Electronic Notes in Theoretical Computer Science

Mathematical Foundations of Programming Semantics
Thirtieth Annual Conference

Cornell University
Ithaca, NY
June 12–15, 2014

Preliminary Proceedings

Guest Editors:

Bart Jacobs, Alexandra Silva and Sam Staton

Contents

Preface

This volume collects papers presented at the 30th Annual Conference on Mathematical Foundations of Programming Semantics (MFPS XXX), held on the campus of Cornell University, Ithaca, New York, USA, from Thursday, June 12 through Sunday, June 15, 2014. The MFPS conferences are devoted to those areas of mathematics, logic, and computer science that are related to models of computation in general and to the semantics of programming languages in particular. The series particularly stresses providing a forum where researchers in mathematics and computer science can meet and exchange ideas about problems of common interest. As the series also strives to maintain breadth in its scope, the conference strongly encourages participation by researchers in neighboring areas.

The Organizing Committee for MFPS consists of Andrej Bauer (Ljubljana), Stephen Brookes (CMU), Achim Jung (Birmingham), Catherine Meadows (NRL), Michael Mislove (Tulane), Joël Ouaknine (Oxford) and Prakash Panangaden (McGill). The local arrangements for MFPS XXX were overseen by Dexter Kozen.

The MFPS XXX Program Committee members are:

Bart Jacobs, Alexandra Silva, Sam Staton (Co-chairs),
Radboud University Nijmegen

- Steve Awodey, CMU
- Andrej Bauer, Ljubljana
- Stephen Brookes, CMU
- Pierre Clairambault, ENS Lyon
- Martín Escardo, Birmingham
- Fabio Gadducci, Pisa
- Ichiro Hasuo, Tokyo
- Martin Hofmann, LMU München
- Achim Jung, Birmingham
- Shin-ya Katsumata, Kyoto
- Naoki Kobayashi, Tokyo
- Dexter Kozen, Cornell

- Conor McBride, Strathclyde
- Guy McCusker, Bath
- Annabelle McIver, Macquarie
- Catherine Meadows, Naval Research Lab
- Stefan Milius, Erlangen-Nürnberg
- Michael Mislove, Tulane
- Rasmus Mogelberg, ITU Copenhagen
- Joël Ouaknine, Oxford
- Prakash Panangaden, McGill
- Daniela Petrisan, ENS Lyon
- Alex Simpson, Edinburgh
- Jamie Vicary, Oxford
- Nobuko Yoshida, Imperial College

The Program Committee selected 16 out of 32 submitted papers for presentation at the meeting. The Organizing Committee selected the following people to give invited plenary talks at the meeting:

- Nick Benton, Microsoft Research, Cambridge, UK
- Andy Gordon, Microsoft Research, Cambridge, UK
- Elham Kashefi, University of Edinburgh, UK
- Dexter Kozen, Cornell University, USA
- Prakash Panangaden, McGill University, Canada
- Alexandra Silva, Radboud University, Nijmegen, The Netherlands

In addition, two of these invited speakers organised special sessions:

- Probabilistic computing, organized by Andy Gordon
- Quantum computing, organized by Elham Kashefi

It remains for us to thank all authors and speakers, the organizers of special sessions, the program committee, and the external referees for their contribution to the success of the conference. We would also like to thank the US Office of Naval Research for its continued support of the MFPS series. Finally, we would like to thank Cornell University for providing the venue and the local staff who helped with the preparations and execution of the conference, especially Michelle Eighmey amd Molly Trufant.

Bart Jacobs
Alexandra Silva
Sam Staton

On Continuous Nondeterminism and State Minimality

Jiří Adámek, Robert S. R. Myers, Henning Urbat

Institut für Theoretische Informatik
Technische Universität Braunschweig
Germany

Stefan Milius

Lehrstuhl für Theoretische Informatik
Friedrich-Alexander-Universität Erlangen-Nürnberg
Germany

Abstract

This paper is devoted to the study of nondeterministic closure automata, that is, nondeterministic finite automata (nfas) equipped with a strict closure operator on the set of states and continuous transition structure. We prove that for each regular language L there is a unique minimal nondeterministic closure automaton whose underlying nfa accepts L. Here minimality means no proper sub or quotient automata exist, just as it does in the case of minimal dfas. Moreover, in the important case where the closure operator of this machine is topological, its underlying nfa is shown to be state-minimal. The basis of these results is an equivalence between the categories of finite semilattices and finite strict closure spaces.

Keywords: Canonical Nondeterministic Automata, State Minimality, Closure Spaces, Semilattices

1 Introduction

Why are state-minimal deterministic finite automata (dfas) easy to construct, whilst no efficient minimization procedure for nondeterministic finite automata (nfas) is known? Let us start with the observation that minimal dfas are built inside the category Set_f of finite sets and functions and are characterized by having no proper subautomata (reachability) and no proper quotient automata (simplicity). Nfas can be regarded as dfas interpreted in the category Rel_f of finite sets and relations, and so one might hope to build minimal nfas in the same way as minimal dfas, but now in Rel_f. However, there is a significant difference: Set_f is both finitely complete and cocomplete, yet Rel_f does not have coequalizers, i.e., canonical quotients. The lack of such canonical constructions provides evidence for the lack of *canonical* state-minimal nfas.

This suggests the following strategy: form the cocompletion of Rel_f obtained by freely adding canonical quotients, which turns out to be the category JSL_f of finite join-semilattices (see Appendix), and build minimal automata in this larger category. Every nfa may be viewed as a dfa in JSL_f via the usual subset construction. In order to obtain more efficient presentations of nfas, avoiding the full power set of states, we make use of a categorical equivalence between JSL_f and the category Cl_f of finite strict closure spaces [2]. The objects of the latter are finite sets Z equipped with a strict closure operator (i.e., an extensive, monotone and idempotent map $\mathbf{cl}_Z : \mathcal{P}Z \to \mathcal{P}Z$ preserving the empty set), and the morphisms are continuous relations, see Definitions 2.9 and 2.10 below. For example, every finite topological space induces a finite strict closure space; these closures are called *topological*.

Just as nfas may be viewed as deterministic automata interpreted in Rel_f, *nondeterministic closure automata* (ncas) are deterministic automata interpreted in Cl_f: an nca is an nfa with a strict closure operator on its set of states, continuous transition relations, an open set of final states and a closed set of initial states. Since the category Cl_f has the same relevant properties as Set_f, we derive for each regular language $L \subseteq \Sigma^*$ the existence of a unique minimal nca $\mathcal{N}(L)$ whose underlying nfa (forgetting the closure operator) accepts L. It is minimal in the sense that it has no proper subautomata (reachability) and no proper quotient automata (simplicity), and can be constructed in a way very much analogous to Brzozowski's classical construction of the minimal dfa [6]: starting with any nca $\mathcal{N}$ accepting L, one has

$$\mathcal{N}(L) = \mathsf{reach} \circ \mathsf{rev} \circ \mathsf{reach} \circ \mathsf{rev}(\mathcal{N})$$

where reach and rev are continuous versions of the reachable subset construction and the reversal operation for nfas, respectively.

The states of $\mathcal{N}(L)$ are the *prime derivatives* of L, i.e., those non-empty derivatives $w^{-1}L = \{v \in \Sigma^* : wv \in L\}$ of L that do not arise as a union of other derivatives. The underlying nfa of $\mathcal{N}(L)$ accepts L, thus it is natural to ask when this nfa is *state-minimal*. Our main result is:

If the closure of $\mathcal{N}(L)$ is topological then the underlying nfa is state-minimal.

In other words, we identify a natural class of regular languages for which *canonical* state-minimal nondeterministic acceptors exist.

Related Work. Our paper is inspired by the work of Denis, Lemay and Terlutte [7] who define a canonical nondeterministic acceptor for each regular language L. In fact, the underlying nfa of our minimal nca $\mathcal{N}(L)$ is precisely their 'canonical residual finite state automaton', and our Brzozowski construction of $\mathcal{N}(L)$ in Section 3 generalizes their construction in [7, Theorem 5.2]. The main conceptual difference is that the latter works on the level of nfas, while our construction takes the continuous structure of nondeterministic closure automata into account. We hope to convince the reader that ncas provide the proper setting in which to study these canonical nfas and their construction.

We have introduced nondeterministic closure automata in [2] where we demonstrated that ncas – as well as related machines like the átomaton of Brzozowski and Tamm [5] – are instances of a uniform coalgebraic construction. There we also gave various simple criteria for nondeterministic state minimality. The present paper can be understood as an in-depth study of ncas, extending the results from [2] and [7] in two ways: firstly, we provide a richer and more conceptual way of constructing the minimal nca $\mathcal{N}(L)$ (compared to [7]) by working with closures. Secondly, we prove that the underlying nfa of $\mathcal{N}(L)$ is state-minimal provided that $\mathcal{N}(L)$ has topological closure, thereby generalizing a much weaker criterion from [2].

2 From Deterministic JSL-Automata to Nondeterministic Closure Automata

In this section we consider deterministic automata interpreted in the category of join-semilattices, and explain how they induce nondeterministic closure automata. We shall assume familiarity with basic concepts from category theory (categories, functors, duality and equivalence).

Notation 2.1 *Throughout this paper we fix an alphabet Σ. The composition of relations $R \subseteq A \times B$ and $S \subseteq B \times C$ is $S \circ R = \{(a,c) : \exists b \in B.(a,b) \in R \wedge (b,c) \in S\}$. Moreover $R[A'] = \{b \in B : \exists a \in A'.(a,b) \in R\}$ denotes the R-image of a subset $A' \subseteq A$, and we write $R[a]$ instead of $R[\{a\}]$.*

Let us first recall deterministic and nondeterministic finite automata and provide them with suitable morphisms.

Definition 2.2 (1) A *nondeterministic finite automaton (nfa)* $N = (Z, R_a, F)$ consists of a finite set Z of states, transition relations $R_a \subseteq Z \times Z$ for every $a \in \Sigma$, and a set $F \subseteq Z$ of final states. A *pointed* nfa (N, I) is additionally equipped with a set of initial states $I \subseteq Z$.

(2) Nfas form a category Nfa whose morphisms $\mathcal{B} : (Z, R_a, F) \to (Z,', R'_a, F')$ are relations $\mathcal{B} \subseteq Z \times Z'$ that preserve and reflect transitions (i.e. $R'_a \circ \mathcal{B} = \mathcal{B} \circ R_a$) and moreover $z \in F$ iff $\mathcal{B}[z] \cap F' \neq \emptyset$. Likewise we have the category Nfa_* of pointed nfas, whose morphisms $\mathcal{B} : (N, I) \to (N', I')$ are additionally required to satisfy $\mathcal{B}[I] = I'$.

Remark 2.3 Our choice of Nfa-morphisms $\mathcal{B}$ is sound: for each $z \in Z$ the pointed nfas $(Z, R_a, F, \{z\})$ and $(Z', R'_a, F', \mathcal{B}[z])$ accept the same language.

Definition 2.4 (1) A *deterministic finite automaton (dfa)* is an nfa $D = (Q, \delta_a, F)$ whose transition relations $\delta_a : Q \to Q$ are functions. A *pointed dfa* (D, q_0) is a dfa equipped with a single initial state $q_0 \in Q$. Morphisms of (pointed) dfas are precisely the Nfa- (resp. Nfa_*-)morphisms that are functions.

(2) A *deterministic automaton (da)* is defined in analogy to to (1), except that the set of states is not required to be finite.

We are mainly interested in d(f)as carrying a semilattice structure.

Notation 2.5 Let JSL denote the category of (join-)semilattices with a bottom element $\bot$, whose morphisms are $\bot$-preserving semilattice homomorphisms. JSL_f is the full subcategory of finite semilattices.

One can view the final states of a da (Q, δ_a, F) as a predicate $f : Q \to \{0, 1\}$ with $F = f^{-1}(\{1\})$. This suggests the following definition: a JSL-*da* is a da whose state set Q carries a semilattice structure, such that the transitions $\delta_a : Q \to Q$ and the final state predicate $f : Q \to \mathbf{2}$ are semilattice morphisms. Here $\mathbf{2}$ denotes the 2-chain $0 < 1$. Note that to give a morphism $f : Q \to \mathbf{2}$ means precisely to give a prime filter $F \subseteq Q$, i.e., $\bot_Q \notin F$ and $q \vee_Q q' \in F$ iff $q \in F$ or $q' \in F$. Indeed, given f, the set $F = f^{-1}(\{1\})$ is a prime filter, and conversely every prime filter of Q arises in this way. Therefore:

Definition 2.6 *(a) A* JSL-da *is a triple* $\mathcal{D} = (Q, \delta_a, F)$ *where* Q *is a semilattice of states, the* a*-transitions* $\delta_a : Q \to Q$ *are semilattice homomorphisms for* $a \in \Sigma$*, and the final states* $F \subseteq Q$ *form a prime filter. Given another* JSL*-da* $\mathcal{D}' = (Q', \delta'_a, F')$*, a* morphism $h : \mathcal{D} \to \mathcal{D}'$ *is a semilattice homomorphism* $h : Q \to Q'$ *such that* $\delta'_a \circ h = h \circ \delta_a$ *and* $q \in F$ *iff* $h(q) \in F'$*. We denote by* Jda *the category of all* JSL*-das, and by* Jdfa *the full subcategory of* JSL-dfas*, that is,* JSL*-das with a finite set of states.*

(b) A pointed JSL-da $(\mathcal{D}, q_0)$ *is a* JSL*-da* $\mathcal{D} = (Q, \delta_a, F)$ *with an initial state* $q_0 \in Q$*. Morphisms of pointed* JSL*-das must additionally preserve the initial state.* Jda_* *denotes the category of pointed* JSL*-das, and* Jdfa_* *the full subcategory of pointed* JSL*-dfas.*

(c) The language accepted by a pointed JSL*-da* $(\mathcal{D}, q_0)$ *is the language accepted by its underlying da, that is, the set*

$$\mathcal{L}_{\mathcal{D}}(q_0) = \{w \in \Sigma^* : \delta_w(q_0) \in F\}$$

where $\delta_w : Q \to Q$ *is the usual inductive extension of the transition function given by* $\delta_\varepsilon = \mathsf{id}_Q$ *and* $\delta_{wa} = \delta_a \circ \delta_w$*.*

Example 2.7 (1) Take any nfa $N = (Z, R_a, F)$. The usual determinization via the subset construction is a JSL-dfa. Indeed, the states $\mathcal{P}Z$ form a semilattice w.r.t. union, the transitions preserve binary unions and the empty set, and the final states $\{A \subseteq Z : A \cap F \neq \emptyset\}$ form a prime filter.

(2) Let $\mathcal{P}\Sigma^*$ be the semilattice (w.r.t. union) of all languages over Σ. It carries the structure of a JSL-da whose transitions are given by $L \to a^{-1}L$ ($a \in \Sigma$) and whose final states F are precisely the languages containing the empty word. This automaton $\mathcal{D}_{\mathcal{P}\Sigma^*} = (\mathcal{P}\Sigma^*, a^{-1}(-), F)$ is easily seen to be the *final* JSL-da: every JSL-da has a unique Jda-morphism into $\mathcal{D}_{\mathcal{P}\Sigma^*}$, namely the function mapping each state to the language it accepts.

(3) Let $\mathcal{D}_{\mathsf{Reg}(\Sigma)}$ be the subautomaton of $\mathcal{D}_{\mathcal{P}\Sigma^*}$ whose states are the regular languages over Σ. It can be characterized up to isomorphism by the property that every JSL-dfa has a unique Jda-morphism into $\mathcal{D}_{\mathsf{Reg}(\Sigma)}$.

Remark 2.8 Readers familiar with the theory of coalgebras will immediately notice that JSL-das correspond to coalgebras for the functor $T = \mathbf{2} \times \mathsf{Id}^{\Sigma} : \mathsf{JSL} \to \mathsf{JSL}$. The examples (2) and (3) above then state precisely that $\mathcal{D}_{\mathcal{P}\Sigma^*}$ is the final T-coalgebra and $\mathcal{D}_{\mathsf{Reg}(\Sigma)}$ is the final locally finite T-coalgebra, see [9].

In [2] it was proved that the category JSL_f of finite semilattices is equivalent to the category of finite strict closure spaces. We recall the necessary concepts.

Definition 2.9 *A* closure operator *(shortly, a* closure*) on a set Z is a monotone, idempotent and extensive function $\mathbf{cl}_Z : \mathcal{P}Z \to \mathcal{P}Z$, that is,*

$$\frac{A \subseteq B}{\mathbf{cl}_Z(A) \subseteq \mathbf{cl}_Z(B)}, \qquad \mathbf{cl}_Z(A) \supseteq A, \qquad \mathbf{cl}_Z \circ \mathbf{cl}_Z = \mathbf{cl}_Z, \qquad \textit{for all } A, B \subseteq Z.$$

A closure space *$(Z, \mathbf{cl}_Z)$ is a set with a closure defined on it. It is* finite *if Z is finite and* strict *if $\mathbf{cl}_Z(\emptyset) = \emptyset$. A subset $A \subseteq Z$ is* closed *if $\mathbf{cl}_Z(A) = A$ and* open *if its complement $\bar{A} \subseteq Z$ is closed. We write $\mathbf{cl}_Z(z)$ for $\mathbf{cl}_Z(\{z\})$.*

Definition 2.10 *Let Z_1 and Z_2 be finite strict closure spaces. Then a relation $\mathcal{B} \subseteq Z_1 \times Z_2$ is said to be* continuous *if:*

(i) For each $z \in Z_1$, the image $\mathcal{B}[z] \subseteq Z_2$ is closed in Z_2.

(ii) $\mathcal{B}[\mathbf{cl}_{Z_1}(A)] \subseteq \mathbf{cl}_{Z_2}(\mathcal{B}[A])$ for all subsets $A \subseteq Z_1$.

Given two continuous relations $\mathcal{B}_1 : Z_1 \to Z_2$ and $\mathcal{B}_2 : Z_2 \to Z_3$ we define their continuous composition *as follows:*

$$\mathcal{B}_2 \bullet \mathcal{B}_1 = \{(z_1, z_3) \in Z_1 \times Z_3 : z_3 \in \mathbf{cl}_{Z_3}(\mathcal{B}_2 \circ \mathcal{B}_1[z_1])\} : Z_1 \to Z_3.$$

That is, one forms the usual relational composition and takes the closure in Z_3. The continuous identity *is defined by $\mathsf{id}_Z = \{(z, z') \in Z \times Z : z' \in \mathbf{cl}_Z(z)\}$.*

Definition 2.11 Let Cl_f denote the category of finite strict closure spaces and continuous relations, with continuous composition and identities.

Remark 2.12 Strict closure spaces can be regarded as generalized topological spaces. Indeed, every topological space Z induces a strict closure space $(Z, \mathbf{cl}_Z)$ where $\mathbf{cl}_Z$ is the usual topological closure operator. It preserves finite unions, i.e.,

$$\mathbf{cl}_Z(A \cup B) = \mathbf{cl}_Z(A) \cup \mathbf{cl}_Z(B) \quad \text{for all } A, B \subseteq Z. \tag{$*$}$$

Conversely, every strict closure space Z satisfying $(*)$ arises from a topological space. Moreover, if $\mathcal{B} : Z_1 \to Z_2$ is a function between topological spaces and Z_2 is a T_1 space, then $\mathcal{B}$ is continuous in the sense of topology iff it is continuous in the sense of Definition 2.10.

Definition 2.13 A closure $\mathbf{cl}_Z$ satisfying $(*)$ is called *topological.*

Definition 2.14 *Let Q be a finite semilattice, so that it is a lattice with meet*

$$q \wedge_Q q' = \bigvee_Q \{r : r \leq_Q q \textit{ and } r \leq_Q q'\}$$

and top element

$$\top_Q = \bigvee_Q \{q : q \in Q\}.$$

Then $q \in Q$ is join-irreducible *(resp.* meet-irreducible*) if (i) $q \neq \bot_Q$ (resp. $q \neq \top_Q$) and (ii) whenever $q = r \vee_Q r'$ (resp. $q = r \wedge_Q r'$) then $q = r$ or $q = r'$. Let $J(Q)$, $M(Q) \subseteq Q$ be the sets of join-irreducible and meet-irreducible elements of Q.*

Lemma 2.15 (see [2]) *The categories of finite semilattices and finite strict closure spaces are equivalent. Indeed, the following functor $G : \mathsf{JSL}_f \to \mathsf{Cl}_f$ is an equivalence:*

$$GQ = (J(Q), \mathbf{cl}_Q) \quad \textit{where } \mathbf{cl}_Q(A) = \{j \in J(Q) : j \leq_Q \bigvee_Q A\}$$

$$Gf = \{(j, j') \in J(Q) \times J(Q') : j' \leq_Q f(j)\} : GQ \to GQ'$$

for any semilattice homomorphism $f : Q \to Q'$.

Remark 2.16 The associated equivalence $C : \mathsf{Cl}_f \to \mathsf{JSL}_f$ maps a finite strict closure space Z to the semilattice CZ of all closed subsets of Z w.r.t. inclusion order. Its bottom is $\emptyset$ and it has joins $A \vee_{CZ} B = \mathbf{cl}_Z(A \cup B)$ (the meet being intersection). A continuous relation $\mathcal{B} : Z_1 \to Z_2$ is mapped to

$$C\mathcal{B} : CZ_1 \to CZ_2, \quad C\mathcal{B}(A) = \mathbf{cl}_{Z_2}(\mathcal{B}[A]).$$

The equivalence G lifts to an equivalence of automata, assigning to each JSL-dfa a corresponding 'nondeterministic closure automaton' in Cl_f.

Definition 2.17 *(a) A* nondeterministic closure automaton *(*nca*) is a triple $\mathcal{N} = (Z, R_a, F)$ where Z is a finite strict closure space (of states), the transition relations $R_a \subseteq Z \times Z$ are continuous for $a \in \Sigma$, and the final states $F \subseteq Z$ form an open set. Given another nca $\mathcal{N}' = (Z', R'_a, F')$, a* morphism *$\mathcal{B} : \mathcal{N} \to \mathcal{N}'$ is a continuous relation $\mathcal{B} : Z \to Z'$ such that $R'_a \bullet \mathcal{B} = \mathcal{B} \bullet R_a$ for each $a \in \Sigma$, and $z \in F$ iff $\mathcal{B}[z] \cap F' \neq \emptyset$. The category of ncas (and the above morphisms with continuous composition) is denoted Nca.*

(b) A pointed nca *$(\mathcal{N}, I)$ is an nca $\mathcal{N} = (Z, R_a, F)$ equipped with a* closed *subset $I \subseteq Z$ of initial states. Morphisms $\mathcal{B}$ between pointed ncas must additionally satisfy $\mathbf{cl}_{Z'}(\mathcal{B}[I]) = I'$. The category of pointed ncas is denoted Nca_*.*

(c) The language accepted by a pointed nca $(\mathcal{N}, I)$ is the set $\mathcal{L}_\mathcal{N}(I) \subseteq \Sigma^*$ of words w such that some state in I has some w-path to a final state.

Remark 2.18 (1) Every nfa $N = (Z, R_a, F)$ may be viewed as an nca where Z is discrete, i.e., it has the identity closure $\mathbf{cl}_Z = \mathsf{id}_{\mathcal{P}Z}$. This nca is well-defined because (i) every relation between discrete closure spaces is continuous and (ii) every subset of a discrete closure space is both open and closed. Therefore we have full embeddings $\mathsf{Nfa} \hookrightarrow \mathsf{Nca}$ and $\mathsf{Nfa}_* \hookrightarrow \mathsf{Nca}_*$.

(2) Every (pointed) nca has an *underlying (pointed) nfa* where we forget the closure. In contrast to the previous statement, this does not define functors $\mathsf{Nca} \to \mathsf{Nfa}$

and $\mathsf{Nca}_* \to \mathsf{Nfa}_*$ because composition of Nca-morphisms is not the relational composition. Note that the language $\mathcal{L}_{\mathcal{N}}(I)$ accepted by a pointed nca $(\mathcal{N}, I)$ is the language accepted by its underlying pointed nfa.

Lemma 2.19 (see [2]) *The categories of (pointed)* JSL*-dfas and (pointed) ncas are equivalent. Indeed, the equivalence* $G : \mathsf{JSL}_f \to \mathsf{Cl}_f$ *described above lifts to equivalences:*

$$\begin{array}{lll} \mathbb{G} : \mathsf{Jdfa} \to \mathsf{Nca} & \textit{with} & \mathbb{G}(Q, \delta_a, F) = (GQ, G\delta_a, F') \\ \mathbb{G}_* : \mathsf{Jdfa}_* \to \mathsf{Nca}_* & \textit{with} & \mathbb{G}_*(Q, \delta_a, F, q_0) = (GQ, G\delta_a, F', I_{q_0}) \end{array}$$

where $F' = J(Q) \cap F$ *and* $I_{q_0} = \{q \in J(Q) : q \leq_Q q_0\}$. *On morphisms we have* $\mathbb{G}f = Gf$ *and* $\mathbb{G}_* f = Gf$.

Proof. [Sketch] This follows from Lemma 2.15. Briefly, $G : \mathsf{JSL}_f \to \mathsf{Cl}_f$ defines an equivalence and one can apply it to the carrier Q and each homomorphism δ_a of a JSL-dfa. Furthermore, the final states arise as a morphism $Q \to \mathbf{2}$ and initial states as a morphism $\mathbf{2} \to Q$, so one may again apply G. The resulting structure is the equivalent (pointed) nca. □

Hence JSL-dfas and ncas are essentially the same structures, although the latter have the significant advantage of having fewer states. For example, if a JSL-dfa has free carrier, then the corresponding nca is exponentially smaller. The languages accepted by JSL-dfas and ncas are by definition just the languages accepted by their underlying dfas and nfas, respectively, and are preserved by the equivalence:

Lemma 2.20 *A pointed* JSL*-dfa accepts the same language as its equivalent pointed nca, i.e.,* $\mathcal{L}_{\mathcal{D}}(q) = \mathcal{L}_{\mathcal{N}}(I)$ *where* $(\mathcal{N}, I) = \mathbb{G}_*(\mathcal{D}, q)$.

Proof. Let $\mathcal{D} = (Q, \delta_a, F)$ and recall $I = J(Q) \cap {\downarrow_Q} q$, where ${\downarrow_Q} q = \{q' \in Q : q' \leq q\}$. Then $w \in \mathcal{L}_{\mathcal{D}}(q)$ iff $\delta_w(q) \in F$ by definition. Equivalently $w \in \mathcal{L}_{\mathcal{D}}(q)$ iff $f \circ \delta_w \circ i = \mathsf{id}_{\mathbf{2}}$ where the JSL_f-morphism $i : \mathbf{2} \to Q$ is defined $i(1) = q$ (and, necessarily, $i(0) = \bot_Q$) and the JSL_f-morphism $f : Q \to \mathbf{2}$ is defined $f(q) = 1$ iff $q \in F$, recalling that F is a prime filter.

Now $f \circ \delta_w \circ i = \mathsf{id}_{\mathbf{2}}$ iff $Gf \bullet G\delta_w \bullet Gi = G\mathsf{id}_{\mathbf{2}} = \mathsf{id}_{\{1\}}$ (where $\{1\}$ has identity closure) because G is faithful, being an equivalence. Observe that $Gi[1] = J(Q) \cap {\downarrow_Q} q = I$ and also $Gf \subseteq J(Q) \times \{1\}$ is such that $Gf[j] = \{1\}$ iff $j \in F$. We now show

$$Gf \bullet G\delta_w \bullet Gi = \mathsf{id}_{\{1\}} \tag{1}$$

iff there exists $j \in I$ such that $G\delta_w[j] \cap F \neq \emptyset$. The latter is equivalent to saying that $\mathbb{G}\mathcal{D}$ accepts w via the initial states I.

Assuming acceptance implies (1) because the relation $G\delta_w \bullet Gi$ contains $G\delta_w \circ Gi$. Conversely, assuming (1), first observe that:

$$\begin{aligned} G\delta_w \bullet Gi[1] &= \mathbf{cl}_Q(G\delta_w \circ Gi[1]) \\ &= \mathbf{cl}_Q(\textstyle\bigcup_{j\in I}\{j_0 \in J(Q) : j_0 \leq_Q \delta_w(j)\}) \\ &= \{j' \in J(Q) : j' \leq \textstyle\bigvee_Q\{\delta_w(j) : j \in I\}\}. \end{aligned}$$

Then (1) implies that $G\delta_w \bullet Gi[1]$ has non-empty intersection with F, i.e., there exists $j' \leq_Q \bigvee_Q\{\delta_w(j) : j \in I\}$ such that $j' \in F \cap J(Q)$, so in particular $j' \neq \bot_Q$. Since F is upwards closed, there is some non-zero $\delta_w(j) \in F$ (where $j \in I$) and hence also some non-zero join-irreducible beneath $\delta_w(j)$ lies in F. This implies $G\delta_w[j] \cap F$ is non-empty. □

3 Reversal, Reachability and Minimality

The purpose of the present section is to prove that every regular language L has an associated *minimal* pointed nca $\mathcal{N}(L)$ accepting L, which is unique up to isomorphism. We also present a construction of this minimal pointed nca, which is analogous to Brzozowski's classical construction of the minimal pointed dfa [6] (see also [4] for a (co-)algebraic view). Recall that the latter takes any pointed nfa (N, I) accepting L and constructs L's minimal dfa as follows:

$$\mathsf{reach} \circ \mathsf{rev} \circ \mathsf{reach} \circ \mathsf{rev}(N, I) \tag{2}$$

Here rev reverses transitions and also swaps the final and initial states,

$$(N, I) = (Z, R_a, F, I) \quad\Longrightarrow\quad \mathsf{rev}(N, I) = (Z, \breve{R}_a, I, F),$$

where $\breve{R}_a$ denotes the converse of the relation R_a. Furthermore, reach performs the reachable subset construction, i.e., it forms the subset dfa and takes its reachable part. In this section we introduce these two operations for pointed ncas. We then prove that the minimal pointed nca $\mathcal{N}(L)$ arises in exactly the same way as (2), only now taking any pointed nca accepting L as input. In particular, any pointed nfa will do.

The above nfa operation rev extends to a self-duality of the category Nfa_* of pointed nfas, defined on objects as above and on morphisms by $\mathcal{B} \mapsto \breve{\mathcal{B}}$. To see that it works on the final/initial states, let $\mathfrak{F} = \{(z, *) : z \in F\} \subseteq Z \times \mathbf{1}$ and $\mathfrak{I} = \{(*, z) : z \in I\} \subseteq \mathbf{1} \times Z$ where $\mathbf{1} = \{*\}$. Then we can rewrite our conditions on Nfa_*-morphisms (see Definition 2.2) as $\mathfrak{F}' \circ \mathcal{B} = \mathfrak{F}$ and $\mathcal{B} \circ \mathfrak{I} = \mathfrak{I}'$, which clearly dualize under converse. Therefore in order to generalize rev to pointed ncas, we describe a suitable self-duality of Nca_*. It is based on the well-known self-duality of JSL_f:

Lemma 3.1 *The following functor $D : \mathsf{JSL}_f^{op} \to \mathsf{JSL}_f$ is an equivalence: on objects let $DQ = Q^{op}$ (which has carrier Q, bottom $\top_Q$ and join $\wedge_Q$) and on morphisms*

$f : Q_1 \to Q_2$ define $Df^{op} : Q_2^{op} \to Q_1^{op}$ by

$$Df^{op}(q_2) = \bigvee_{Q_1}\{q_1 \in Q_1 : f(q_1) \leq_{Q_2} q_2\}.$$

Proof. [Sketch] The self-duality of JSL_f follows from the adjoint functor theorem for posets. Finite join-semilattices are finite posets with all joins (= colimits) and join-semilattice morphisms are monotone maps that preserve all joins. Consequently, each $f : Q_1 \to Q_2$ has a right adjoint $g : Q_2^{op} \to Q_1^{op}$ where the order is reversed because right adjoints preserve all meets. The uniqueness of adjoints implies that this is an equivalence. Its explication yields the above action on the morphisms. □

Since JSL_f is equivalent to Cl_f (see Lemma 2.15 and Remark 2.16), it follows that Cl_f is also self-dual, with dual equivalence

$$\mathbf{D} = (\,\mathsf{Cl}_f^{op} \xrightarrow{C^{op}} \mathsf{JSL}_f^{op} \xrightarrow{D} \mathsf{JSL}_f \xrightarrow{G} \mathsf{Cl}_f\,).$$

We now describe this self-duality explicitly. Recall that CZ denotes the semilattice of closed subsets of a finite strict closure space Z, and that $M(CZ)$ is the set of meet-irreducibles of CZ.

Proposition 3.2 *For any finite strict closure space Z we have*

$$\mathbf{D}Z = M(CZ) \qquad \mathbf{cl}_{\mathbf{D}Z}(X) = \{A \in M(CZ) : \bigcap X \subseteq A\},$$

and for any $\mathcal{B} : Z_1 \to Z_2$, the continuous relation $\mathbf{D}\mathcal{B}^{op} : M(CZ_2) \to M(CZ_1)$ consists of all those $(A_2, A_1) \in M(CZ_2) \times M(CZ_1)$ such that:

$$\mathcal{B}[A] \subseteq A_2 \quad \Longrightarrow \quad A \subseteq A_1 \qquad \text{for every } A \in J(CZ_1).$$

Proof. We have $\mathbf{D}Z = G((CZ)^{op}) = (M(CZ), \mathbf{cl})$ where $M(CZ) = J((CZ)^{op}) \subseteq Q$ are the meet-irreducibles and:

$$\begin{aligned}\mathbf{cl}(S) &= \{j \in J((CZ)^{op}) : j \leq_{(CZ)^{op}} \textstyle\bigvee_{(CZ)^{op}} S\}\\ &= \{m \in M(CZ) : \textstyle\bigwedge_{CZ} S \leq_{CZ} m\}\\ &= \{m \in M(CZ) : \textstyle\bigcap S \subseteq m\}\end{aligned}$$

using the fact that the meet in CZ is intersection. Likewise every JSL_f-morphism $f : Q \to Q'$ has a dual morphism $Df^{op} : Q'^{op} \to Q^{op}$ defined $Df^{op}(q') = \bigvee_Q f^{-1}(\downarrow_{Q'} q')$. Therefore every continuous relation $\mathcal{B} : Z \to Z'$ has a *dual* continuous relation $\mathbf{D}\mathcal{B}^{op} : \mathbf{D}Z' \to \mathbf{D}Z$ defined as follows: let $f = C\mathcal{B}$ be the JSL_f-morphism corresponding to $\mathcal{B}$ and then apply the equivalence $G : \mathsf{JSL}_f \to \mathsf{Cl}_f$ to the homomorphism

Df^{op}. Then

$$\begin{aligned}\mathbf{D}\mathcal{B}^{op} &= \{(j',j) \in J(Q'^{op}) \times J(Q^{op}) : j \leq_{Q^{op}} Df^{op}(j')\} \\ &= \{(m',m) \in M(Q') \times M(Q) : \textstyle\bigvee_Q f^{-1}(\downarrow_{Q'} m') \leq_Q m\} \\ &= \{(m',m) : \forall j \in J(Q).(f(j) \leq_{Q'} m' \Rightarrow j \leq_Q m)\} \\ &= \{(m',m) : \forall j \in J(Q).(\mathbf{cl}_{Z_2}(\mathcal{B}[j]) \leq_{Q'} m' \Rightarrow j \leq_Q m)\} \\ &= \{(m',m) : \forall j \in J(Q).(\mathcal{B}[j] \leq_{Q'} m' \Rightarrow j \leq_Q m)\}\end{aligned}$$

In the last step we use that $B[j]$ is closed since $\mathcal{B}$ is continuous. □

Given any pointed nca (Z, R_a, F, I), the previous duality naturally leads to a pointed nca with states $M(CZ)$ by applying $\mathbf{D}$ to (a) $R_a : Z \to Z$, (b) F considered as a continuous relation $F : Z \to \{1\}$ and (c) I considered as a continuous relation $I : \{1\} \to Z$. This is the *reversal* of pointed ncas: by applying $\mathbf{D}$ to F we get the subset $I^d \subseteq M(CZ)$ of all $A \in M(CZ)$ containing $Z \setminus F$. By applying $\mathbf{D}$ to I we get the subset $F^d \subseteq M(CZ)$ of all $A \in M(CZ)$ with $I \nsubseteq A$.

Definition 3.3 *The* reversal *of a pointed nca is defined by*

$$\mathsf{rev}(Z, R_a, F, I) = (M(CZ), \mathbf{D}R_a^{op}, F^d, I^d).$$

Example 3.4 Take any pointed nfa and view it as a pointed nca with identity closure. Then $CZ = \mathcal{P}Z$ has meet-irreducibles $Z \setminus \{z\}$ for $z \in Z$ and the reversal is the classical nfa reversal, modulo the bijection $z \mapsto Z \setminus \{z\}$.

Theorem 3.5 *The category* Nca_* *is self-dual: the object map* rev *extends to an equivalence* $\mathsf{rev} : \mathsf{Nca}_*^{op} \to \mathsf{Nca}_*$. *It assigns to every morphism* $\mathcal{B} : (Z, R_a, I, F) \to (Z', R'_a, I', F')$ *the morphism* $\mathsf{rev}(\mathcal{B}) \subseteq M(CZ') \times M(CZ)$ *of all pairs* (A', A) *such that*

$$\mathcal{B}[X] \subseteq A' \quad\Longrightarrow\quad X \subseteq A \qquad \textit{for every } X \in J(CZ).$$

Proof. In Definition 3.3 we defined the object part of the dual equivalence $\mathsf{rev} : \mathsf{Nca}_*^{op} \to \mathsf{Nca}$. We now prove that it actually defines a functor.

(i) The action on continuous relations R_a and Nca_*-morphisms $\mathcal{B}$ is the action of $\mathbf{D}$, see Proposition 3.2.

(ii) The initial states form a closed set $I \in CZ$, or equivalently a join-semilattice morphism $i : \mathbf{2} \to CZ$ such that $i(1) = I$. Dually we have the join-semilattice morphism $i' = Di^{op} : (CZ)^{op} \to \mathbf{2}^{op}$ defined by

$$i'(x) = \bigvee_{\mathbf{2}} i^{-1}(\downarrow_{CZ} x) = \begin{cases} \top_{\mathbf{2}^{op}} = 0 & \text{if } i(1) \not\leq_{CZ} x \\ 1 & \text{otherwise} \end{cases}$$

Therefore the new final states are:

$$\begin{aligned} F^d &= J((CZ)^{op}) \cap i'^{-1}(\top_{\mathbf{2}^{op}}) \\ &= \{m \in M(CZ) : I \not\leq_{CZ} m\} \\ &= \{m \in M(CZ) : I \not\subseteq m\}. \end{aligned}$$

(iii) The final states F form an open set in $(Z, \mathbf{cl}_Z)$, or equivalently a join-semilattice morphism $f : CZ \to \mathbf{2}$ such that $Z \setminus F = \bigvee_{CZ} f^{-1}(\{0\})$. The dual join-semilattice morphism $f' = Df^{op} : \mathbf{2}^{op} \to (CZ)^{op}$ is defined by $f'(b) = \bigvee_{CZ} f^{-1}(\downarrow_{\mathbf{2}} b)$. Consequently:

$$\begin{aligned} I^d &= \{j \in J((CZ)^{op}) : j \leq_{(CZ)^{op}} f'(\top_{\mathbf{2}^{op}})\} \\ &= \{m \in M(CZ) : \textstyle\bigvee_{CZ} f^{-1}(\{0\}) \leq_{CZ} m\} \\ &= \{m \in M(CZ) : Z \setminus F \subseteq m\}. \end{aligned}$$

□

Proposition 3.6 *If a pointed nca accepts L then its reverse pointed nca accepts the reversed language* $\mathsf{rev}(L)$*.*

Proof. A pointed JSL-dfa $(\mathcal{D}, q) = (Q, \delta_a, F, q)$ accepts a word w iff $f \circ \delta_w \circ i = \mathsf{id}_{\mathbf{2}}$, where $f : Q \to \mathbf{2}$ represents the final states, δ_w is the inductive extension of the δ_a's and $i : \mathbf{2} \to Q$ represents $q \in Q$. Now Jdfa_* is self-dual (because Nca_* is) with dual pointed JSL-dfa $(Q^{op}, D\delta_a^{op}, \uparrow_{Q^{op}} q, \bigvee_Q (Z \setminus F))$. The dual of the equality above is $Di^{op} \circ D\delta_{w^r}^{op} \circ Df^{op} = \mathsf{id}_{\mathbf{2}^{op}}$ and furthermore Di corresponds to the *final* states in the dual machine, just as Df corresponds to the *initial* states. Consequently $(\mathcal{D}, q)$ accepts w iff its dual machine accepts the reversed word w^r. This property is inherited by Nca_* because the equivalence $\mathbb{G}_* : \mathsf{Jdfa}_* \to \mathsf{Nca}_*$ preserves languages, see Lemma 2.20. □

Next we extend the operation reach from nfas to pointed ncas. A pointed nfa is *reachable* if each state is reached from some initial state by transitions. Equivalently, the pointed nfa has no proper sub nfas. Here 'sub nfa' refers to the category Nfa_* i.e. N' is a sub nfa of N if there is a morphism $\mathcal{B} : N' \to N$ where $\mathcal{B}$ is an injective function. Implicitly one uses the (onto relation, injective function) factorization system in Rel_f and lifts it to Nfa_*. Similar remarks apply to sub dfas: (i) they arise as injective dfa morphisms via the (surjection, injection) factorization system of Set_f, (ii) a pointed dfa is reachable iff it has no proper sub dfas.

In order to define *reachable pointed ncas*, we first need the appropriate concept of *sub nca*. To this end, we take the (epi, mono) = (surjection, injection) factorization system of JSL_f and apply the equivalence of JSL_f and Cl_f to obtain a corresponding factorization system in Cl_f.

Lemma 3.7 *Every continuous relation $\mathcal{B} : Z_1 \to Z_2$ has an essentially unique (epi,* *mono)-factorization in* Cl_f. *Moreover,* $\mathcal{B}$ *is monic (resp. epic) iff the function*

$$C\mathcal{B} : CZ_1 \to CZ_2, \quad C\mathcal{B}(S) = \mathbf{cl}_{Z_2}(\mathcal{B}[S]),$$

is injective (resp. surjective).

Proof. The functor $C : \mathsf{Cl}_f \to \mathsf{JSL}_f$ preserves and reflects monos and epis, being an equivalence. □

Definition 3.8 *(a) A pointed nca* $(\mathcal{N}', I')$ *is a* sub nca *of* $(\mathcal{N}, I)$ *if there exists an* Nca_**-monomorphism* $m : (\mathcal{N}', I') \rightarrowtail (\mathcal{N}, I)$.

(b) A pointed nca $(\mathcal{N}', I')$ *is a* quotient nca *of* $(\mathcal{N}, I)$ *if there exists an* Nca_**-epimorphism* $e : (\mathcal{N}, I) \twoheadrightarrow (\mathcal{N}', I')$.

(c) A pointed nca is called reachable *if it has no proper sub ncas,* simple *if it has no proper quotient ncas, and* minimal *if it is both reachable and simple.*

Proposition 3.9 *Any sub or quotient nca of* $(\mathcal{N}, I)$ *accepts the same language* $\mathcal{L}_{\mathcal{N}}(I)$.

Proof. Viewed as their equivalent pointed JSL-dfa, we have injective or surjective deterministic automata morphisms which preserve the initial state. These certainly preserve the language and by Lemma 2.20 the respective pointed ncas accept the same languages. □

To obtain a more concrete characterization of reachability and simplicity, we shall restrict to ncas whose closure is normalized in the following sense:

Lemma 3.10 *Every finite strict closure space* Y *is isomorphic to another finite strict closure space* Z *such that:*

(i) Z *is separable, that is,* $z \neq z' \in Z$ *implies* $\mathbf{cl}_Z(z) \neq \mathbf{cl}_Z(z')$.

(ii) $S \in CZ$ *is join-irreducible in* CZ *iff* $S = \mathbf{cl}_Z(\{z\})$ *for some* $z \in Z$.

Proof.

(i) Recall from Lemma 2.15 and Remark 2.16 the equivalence $G : \mathsf{JSL}_f \to \mathsf{Cl}_f$ and its associated equivalence $C : \mathsf{Cl}_f \to \mathsf{JSL}_f$. Then Y is isomorphic to the closure space $Z = GCY$ whose carrier $J(CY)$ is the set of join-irreducibles in CY and whose closure is defined by $\mathbf{cl}_{GCY}(S) = \{j \in J(CY) : j \subseteq \mathbf{cl}_Y(\bigcup S)\}$ for any $S \subseteq J(CY)$. This closure space GCY is separable: given distinct join-irreducibles $j \neq j' \in J(CY)$ then wlog $j \nleq_{CY} j'$, hence $j \nsubseteq j' = \mathbf{cl}_Y(j')$ and $j \notin \mathbf{cl}_{GCY}(\{j'\})$. But clearly $j' \in \mathbf{cl}_{GCY}(\{j'\})$.

(ii) In *any* closure space Z', every join-irreducible in CZ' is the closure of some singleton set. For if $S = \mathbf{cl}_{Z'}(S)$ is join-irreducible and $S = S_1 \cup S_2$ then $S = \mathbf{cl}_{Z'}(S_1) \vee_{CZ'} \mathbf{cl}_{Z'}(S_2)$ and hence wlog $S = \mathbf{cl}_{Z'}(S_1)$. Continuing we get either $S = \mathbf{cl}_{Z'}(\emptyset) = \emptyset$ (a contradiction) or S is the closure of a singleton set. For the particular closure space $GCY = (J(CY), \mathbf{cl}_{CY})$ we also have the converse, i.e., the closure of a singleton subset of $J(CY)$ is join-irreducible in $C(GCY)$. This follows because the closed sets of GCY take the form $J(CY) \cap \downarrow_{CY} S$ for some $S \in CY$ and in particular the closure of a singleton $\{j\}$, $j \in J(CY)$, consists of all join-irreducibles smaller than or equal to j. It follows that if $\mathbf{cl}_{GCY}(\{j\}) = K_1 \vee_{C(GCY)} K_2$ then some K_i contains j by join-irreducibility of $\{j\}$, wlog $j \in K_1$. Then $\mathbf{cl}_{GCY}(\{j\}) \subseteq \mathbf{cl}_{GCY}(K_1) = K_1$, and the converse is clear.

□

Definition 3.11 A (pointed) nca is *normalized* if its closure satisfies the conditions of Lemma 3.10.

Corollary 3.12 *Every nca is isomorphic to a normalized nca.*

Proposition 3.13 *A normalized pointed nca (Z, R_a, F, I) is reachable iff for every $z \in Z$ there exists a word $w_z \in \Sigma^*$ such that:*

(i) There is a w_z-path from some initial state to z in the underlying nfa.

(ii) Every w_z-path from every initial state terminates at an element of $\mathbf{cl}_Z(z)$.

Proof. Suppose $(\mathcal{N}, I)$ is a reachable pointed nca. Then by the equivalence of $\mathbb{G}_* : \mathsf{Jdfa}_* \to \mathsf{Nca}_*$, we have a corresponding pointed JSL-dfa $(\mathcal{D}, I) = (CZ, CR_a, F', I)$. Its final states $F' \subseteq CZ$ are defined $F' = \{A \in CZ : A \cap F \neq \emptyset\}$. Note $I \in CZ$ is now a *single state*. By the equivalence of Nca_* and Jdfa_* we know that $(\mathcal{D}, I)$ has no proper subobjects i.e. every injective Jdfa_*-morphism into $(\mathcal{D}, I)$ is bijective. Viewing $\mathcal{D}$ as its underlying dfa, one can list those states reachable from the state $I \in CZ$ via transitions and then construct the join-subsemilattice of CZ generated by this set. This defines a pointed sub JSL-dfa, using the fact that the transition functions $CR_a : CZ \to CZ$ are JSL_f-morphisms, hence an injective Jdfa_*-morphism which is necessarily bijective. It follows that *every* $A \in CZ$ arises as a join of elements reachable from the single state I via transitions. In particular the join-irreducible elements must be reachable from I via transitions, since they form the minimal generating set. So take any element $z \in Z$ and consider its closure $A = \mathbf{cl}_Z(z)$, this being an element of $\mathcal{D}$'s carrier. By our assumption that the closure has been normalized, A is join-irreducible in CZ. Therefore there exists some $w_z \in \Sigma^*$ such that $CR_{w_z}(I) = A$ i.e. we have a deterministic w_z-path in $\mathcal{D}$ from the initial state I to the state A. Then $A = \mathbf{cl}_Z(R_{w_z}[I])$ and since A is join-irreducible and $\mathbf{cl}_Z$ is separable we deduce $z \in R_{w_z}[I]$, i.e., the first condition holds. The second condition follows because $z' \in R_{w_z}[z_0]$ implies $z' \in CR_{w_z}(I) = A = \mathbf{cl}_Z(z)$.

Conversely, suppose the two conditions hold for some pointed nca (Z, R_a, F, I). Consider its equivalent pointed JSL-dfa with carrier CZ. The conditions imply that every join-irreducible in CZ is reachable from the single state I. We can form a sub JSL-dfa by closing under the transitions and then forming the generated subalgebra. Since every join-irreducible is reachable, this gives the original JSL-dfa. Furthermore this is the smallest sub JSL-dfa, which implies the respective pointed nca is reachable. □

Remark 3.14 The above condition (ii) may be felt surprising. However, recall that reachability for pointed ncas was defined by complete analogy with nfas: no proper sub nca exists. For pointed dfas (viewed as ncas with identity closure) this is the usual notion of reachability, since there is exactly one w_z-path from the unique initial state. However the same cannot be said for pointed nfas.

Proposition 3.15 *Viewed as a pointed nca with identity closure, a pointed nfa is reachable iff its reachable subset construction (that is, the reachable part of the subset dfa) contains all singleton sets.*

Proof. Let (Z, R_a, F, I) be a pointed nfa. If every singleton subset lies in its reachable subset construction, then every $z \in Z$ has some $w_z \in \Sigma^*$ such that the unique path from the single state I terminates at z. Since the path is unique and $z \in \mathbf{cl}_Z(z) = \{z\}$, it follows that we have a reachable pointed nca by Proposition 3.13. Conversely, suppose this pointed nfa defines a reachable pointed nca with identity closure. Then by Proposition 3.13 each $z \in Z$ has some w_z such that $I \xrightarrow{w_z} \{z\}$ because no other state lies in the closure of $\{z\}$. □

We now provide further properties of reachable pointed ncas.

Proposition 3.16 *Suppose one has a normalized reachable pointed nca accepting L, then:*

(i) *Its underlying pointed nfa is reachable.*

(ii) *Its individual states accept derivatives of L.*

(iii) *Varying the (closed) set of initial states, the languages it accepts are precisely the unions of L's derivatives.*

Proof. The first statement follows immediately from Proposition 3.13 via the first condition (i). The second statement follows because for each $z \in Z$, its closure is reachable from I in the equivalent pointed JSL-dfa. Thus this closed set accepts a derivative of L and this language is preserved by the equivalence (see Lemma 2.20). Finally, for (iii) observe that the JSL-dfa equivalent to $\mathcal{N}$ accepts precisely the unions of derivatives of L when varying the initial state: indeed, the states reachable via transitions accept precisely the derivatives of L, and all other states arise as a join of such states. Since languages are preserved by the equivalence, (iii) follows. □

Every pointed nfa has a smallest sub nfa, which is necessarily reachable. That is, one simply discards all those states not reachable from the initial states by transitions. From the categorical standpoint this means that the intersection of all pointed sub nfas exists. Similarly, for any pointed nca, the intersection of all pointed sub ncas exists i.e. we can always construct a unique reachable sub nca.

Definition 3.17 *The* reachable part $\mathsf{reach}(\mathcal{N}, I)$ *of a pointed nca* $(\mathcal{N}, I)$ *is the unique reachable pointed sub nca, i.e. the intersection of all pointed sub ncas.*

Notation 3.18 *Given any nca with carrier Z, any word $w \in \Sigma^*$ and any subset $I \subseteq Z$, we write $z \xrightarrow{w} z'$ whenever there is a w-path from z to z' in the underlying nfa. Then $w \cdot I \in CZ$ denotes the closure of $\{z' \in Z : \exists z \in I. z \xrightarrow{w} z'\}$.*

Proposition 3.19 *$\mathsf{reach}(Z, R_a, F, I)$ is isomorphic to (Z', R'_a, F', I') where:*

(i) $Z' \subseteq CZ$ is the set of $w \cdot I$'s not arising as joins of other $v \cdot I$'s in CZ;

(ii) $R'_a = \{(u \cdot I, v \cdot I) \in Z' \times Z' : v \cdot I \subseteq ua \cdot I\}$ for each $a \in \Sigma$;

(iii) $F' = \{A \in Z' : F \cap A \neq \emptyset\}$;

(iv) $I' = \{A \in Z' : A \subseteq I\}$.

Proof. [Sketch] This follows by considering equivalent pointed JSL-dfa i.e. we close under the deterministic transitions from I, form the generated subalgebra, and then convert this JSL-dfa back into a pointed nca. □

Remark 3.20 Applying this to an nfa (i.e. a pointed nca with identity closure), one finds that $\mathsf{reach}(\mathcal{N}, I)$ is never larger than the reachable subset construction $\{w \cdot I : w \in \Sigma^*\}$.

Next we characterize *simple* pointed ncas. Recall that a dfa is simple iff distinct states accept distinct languages. Analogously:

Proposition 3.21 *A pointed nca $(\mathcal{N}, I)$ with carrier Z is simple iff distinct closed subsets accept distinct languages i.e. if $A \neq B \in CZ$ then $\mathcal{L}_{\mathcal{N}}(A) \neq \mathcal{L}_{\mathcal{N}}(B)$.*

Proof. Let $\mathcal{D}$ be the JSL-dfa equivalent to N. It is simple since $\mathcal{N}$ is, so the unique map into the final JSL-da $\mathcal{D}_{\mathcal{P}\Sigma^*}$ (see Example 2.7) is injective. This means that distinct states of $\mathcal{D}$ accept distinct languages, hence the statement follows from Lemma 2.20. □

By Theorem 3.5 we know Nca_* is self-dual. Moreover *reachable* and *simple* are dual concepts, see Definition 3.8. Then if $(\mathcal{N}, I)$ is any pointed nca accepting L, it follows that

$$\mathsf{sim}(\mathcal{N}, I) \overset{\mathsf{def}}{=} \mathsf{rev} \circ \mathsf{reach} \circ \mathsf{rev}(\mathcal{N}, I)$$

is a simple pointed nca accepting L i.e. the *simplification* of $(\mathcal{N}, I)$. Categorically, it is the cointersection of all quotient ncas.

Next consider $\mathsf{reach} \circ \mathsf{sim}(\mathcal{N}, I)$ which is certainly reachable. Importantly it is also simple, using Proposition 3.21 and the fact $\mathsf{reach}(\mathcal{N}, I)$ is a sub nca of $(\mathcal{N}, I)$. Then by definition it is a *minimal* pointed nca accepting L. In fact:

Proposition 3.22 *For every regular languages L, there is up to isomorphism only one minimal pointed nca accepting L.*

Proof. This follows from a more general result in [1, Lemma 3.22] since the minimal nca is an instance of a *well-pointed coalgebra*. Let us briefly sketch the argument. Suppose one has two minimal pointed ncas accepting L. Equivalently one has two minimal pointed finite JSL-dfas accepting L, where minimal now means reachable and simple in Jdfa_*. Each one has a unique Jda-morphism to the deterministic JSL-automaton $\mathcal{D}_{\mathsf{Reg}(\Sigma)}$ of regular languages (see Example 2.7(c)), assigning to each state the language it accepts. These morphisms factorize into a surjective morphism followed by an inclusion i.e. their image defines a sub dfa of $\mathcal{D}_{\mathsf{Reg}(\Sigma)}$. Since they are both simple, they are each isomorphic to their respective image. Since they are both reachable, by Proposition 3.16 the carrier of this image is precisely the set of unions of derivatives of L. Therefore they are isomorphic to each other. □

Notation 3.23 *Let Q_L denote the semilattice of all unions of derivatives of L.*

Proposition 3.24 *The following pointed nca $\mathcal{N}(L)$ for is minimal nca for L:*

(i) *States $Z_L = J(Q_L)$ i.e. the non-empty derivatives of L that are not unions of others derivatives, endowed with the closure:*

$$\mathbf{cl}_L(A) = \{K \in Z_L : K \subseteq \bigcup A\},$$

(ii) *transitions $R_a = \{(K, K') \in Z_L \times Z_L : K' \subseteq a^{-1}K\}$ for each $a \in \Sigma$,*

(iii) *as final states those $K \in Z_L$ containing ε,*

(iv) *as initial states those $K \in Z_L$ which are subsets of L.*

Proof. It is easy to see that the minimal pointed JSL-dfa accepting L is the finite subautomaton of $\mathcal{D}_{\mathsf{Reg}(\Sigma)}$ (see Example 2.7(c)) generated by L. Hence is has states Q_L, transitions $K \to a^{-1}K$ ($K \in Q_L$), initial state L, and the final states are precisely those languages in Q_L containing ε. Now apply the equivalence of Jdfa_* and Nca_*. □

Then the main result in this section follows:

Theorem 3.25 *For any pointed nca $(\mathcal{N}, I)$ accepting L, the pointed nca:*

$$\mathsf{reach} \circ \mathsf{rev} \circ \mathsf{reach} \circ \mathsf{rev}(\mathcal{N}, I)$$

is a minimal pointed nca accepting L, and is hence isomorphic to $\mathcal{N}(L)$.

Proof. By Proposition 3.6, the dual of a pointed nca accepts the reversed language. Since reach preserves the accepted language, the above pointed nca accepts L. so by Proposition 3.22 it suffices to show it is a minimal pointed nca. It is clearly reachable. Moreover, $\mathsf{rev} \circ \mathsf{reach} \circ \mathsf{rev}(\mathcal{N}, I)$ is the dual of a reachable pointed nca and hence is simple. Finally, reach preserves simplicity (as previously explained), so we are done. □

Finally, since Nca_* is self-dual and reachability and simplicity are dual:

Proposition 3.26 *For each regular language L, the reverse of the minimal pointed nca $\mathcal{N}(L)$ is isomorphic to the minimal pointed nca for $\mathsf{rev}(L)$, shortly,*

$$\mathsf{rev}(\mathcal{N}(L)) \cong \mathcal{N}(\mathsf{rev}(L)).$$

Proof. By Proposition 3.6 we know the dual of $\mathcal{N}(L)$ accepts $\mathsf{rev}(L)$. Furthermore minimality is a *self-dual* property because 'no proper subobjects' and 'no proper quotients' dualize. Hence, the dual of $\mathcal{N}(L)$ is minimal and is isomorphic to $\mathcal{N}(\mathsf{rev}(L))$ by Proposition 3.22. □

4 State-minimal Nondeterministic Automata

Each regular language L is accepted by the underlying nfa of its minimal pointed nca $\mathcal{N}(L)$. Although this nfa is never larger than the minimal dfa, it generally need not be a state-minimal *nondeterministic* acceptor [7]. In this section we present a natural sufficient condition for state minimality. Recall that a strict closure $\mathbf{cl}_Z$ is *topological* if it is induced by a topology on Z, i.e., it satisfies the equation

$$\mathbf{cl}_Z(A \cup B) = \mathbf{cl}_Z(A) \cup \mathbf{cl}_Z(B) \quad \text{for all } A, B \subseteq Z.$$

Definition 4.1 *An nca is* topological *if its closure is topological.*

Lemma 4.2 *$\mathcal{N}(L)$ is topological iff the lattice $Q_L \cong CZ_L$ is distributive.*

Proof. If $\mathbf{cl}_{Z_L}$ is topological then $CZ_L \subseteq \mathcal{P}Z_L$ is closed under union and intersection, hence it is a distributive lattice. Conversely suppose that $Q_L \cong CZ_L$ is distributive, and let $A = \{K_1, \dots, K_m\}$ and $B = \{L_1, \dots, L_n\}$ be closed subsets of Z_L. We need to show that $A \cup B$ is closed, so suppose that $K \in \mathbf{cl}_{Z_L}(A \cup B)$, that is, $K \subseteq K_1 \cup \dots \cup K_m \cup L_1 \cup \dots \cup L_n$ by the definition of $\mathbf{cl}_{Z_L}$. Note that for any join-irreducible element x of a finite distributive lattice, $x \leq y \vee z$ implies $x \leq y$ or $x \leq z$ (because $x \leq y \vee z$ implies $x = x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$, hence $x = x \wedge y$ or $x = x \wedge z$ since x is join-irreducible). We conclude that $K \subseteq K_1 \cup \dots \cup K_m$ or $L \subseteq L_1 \cup \dots \cup L_n$, so $K \in \mathbf{cl}_{Z_L}(A) = A$ or $K \in \mathbf{cl}_{Z_L}(B) = B$. Thus $\mathbf{cl}_{Z_L}(A \cup B) \subseteq A \cup B$. □

Example 4.3 (1) If $\mathcal{N}(L)$ is an nfa (i.e., has identity closure) then it is topological. For example, this is the case whenever $d_L = 2^{n_L}$ where d_L (resp. n_L) is the number of states of a state-minimal dfa (resp. nfa) accepting L, see [2].

(2) If $\mathcal{N}(L)$ is topological then so is $\mathcal{N}(\mathsf{rev}(L))$. Indeed, recall that they are duals. Then since $\mathcal{N}(L)$ is topological iff the closed sets CZ_L form a distributive lattice, its order-dual is also distributive and is isomorphic to $CZ_{\mathsf{rev}(L)}$.

(3) If $\mathcal{N}(L)$ is topological and $f : \Delta^* \to \Sigma^*$ is a surjective monoid morphism then $\mathcal{N}(f^{-1}(L))$ is also topological. Here one uses the fact that $f^{-1} : \mathcal{P}\Sigma^* \to \mathcal{P}\Delta^*$ is an injective boolean morphism, providing an isomorphism between the semilattices Q_L and $Q_{f^{-1}(L)}$.

(4) If L is *intersection-closed* (that is, each binary intersection of derivatives of L arises as a union of derivatives of L), then $\mathcal{N}(L)$ is topological by Lemma 4.2. Examples of intersection-closed languages include:
   (a) the languages Σ^* and $\{w\}$ for $w \in \Sigma^*$,
   (b) the tail languages $(a+b)^*b(a+b)^{n-1}$ $(n \geq 1)$,
   (c) linear codes, i.e., linear subspaces of the vector space $\mathbb{Z}_2^n$ (viewed as languages over the binary alphabet);
   (d) the languages $L_f = \{w \in 2^n : f(w) = 1\} \subseteq 2^*$ where $f : 2^n \to 2$ is either the parity function, the majority function or any $\mathbb{R}$-weighted threshold function, i.e.,

$$f(b_1, \dots b_n) = 1 \quad \text{iff} \quad \sum k_i b_i \geq t$$

for some real-valued constants $k_1, \dots, k_n$ and t.

(5) The language $L = a(a+b) + b(b+c)$ is not intersection-closed, yet $\mathcal{N}(L)$ is topological. Indeed, the derivatives of L are $\emptyset$, L, $a+b$, $b+c$ and ε, and $(a+b) \cap (b+c) = b$ is not a union of derivatives. However, the lattice Q_L is isomorphic to the lattice of all subsets of a four-element set, and hence distributive.

In [2] is was proved that $\mathcal{N}(L)$ is state-minimal provided that L is intersection-closed. The following theorem is a generalization of this result, as witnessed by Example 4.3(4),(5) above.

Theorem 4.4 *Let L be a regular language. If the minimal pointed nca $\mathcal{N}(L)$ is topological then its underlying nfa is state-minimal.*

Proof. (1) By Lemma 2.19, the categories of pointed JSL-dfas and pointed nondeterministic closure automata are equivalent. In fact, $\mathbb{G}_*$ has the associated equivalence $\mathbb{C}_* : \mathsf{Nca}_* \to \mathsf{Jdfa}_*$ defined by

$$\mathbb{C}_*(Z, R_a, F, I) = (CZ, CR_a, F', I) \quad \text{and} \quad \mathbb{C}_*\mathcal{B} = C\mathcal{B},$$

where $F' = \{A \in CZ : A \cap F \neq \emptyset\}$. Here we use the equivalence $C : \mathsf{Cl}_f \to \mathsf{JSL}_f$, see Remark 2.16. Observe that $I \in CZ$ is now a *single state* in a JSL-dfa.

(2) Given any pointed nfa $(N, I) = (Z, R_a, F, I)$ accepting L. Viewing (N, I) as a pointed nca $(\mathcal{N}, I)$ with identity closure and applying the above equivalence yields the JSL-dfa $(\mathcal{D}, I)$ where $\mathcal{D} = (\mathcal{P}Z, CR_a, F')$ is the subset construction. Let

$$Q = \{\mathcal{L}_{\mathcal{D}}(q) : q \in \mathcal{P}Z\}$$

be the set of languages accepted by the individual states in $\mathcal{D}$. Since these are precisely those languages accepted by the nfa N (varying the set of initial states), we deduce that Q is closed under finite unions and derivatives, so Q is a semilattice of regular languages under $\emptyset$ and $\cup$. It forms the carrier of a JSL-dfa $\mathcal{D}' = (Q, \delta_a, \hat{F})$ with transitions $K \xrightarrow{a} a^{-1}K$ and final states $\hat{F} := \{K \in Q : \varepsilon \in K\}$. Furthermore, $\mathcal{L}_{\mathcal{D}}$ defines a surjective Jdfa_*-morphism

$$\mathcal{L}_{\mathcal{D}} : (\mathcal{D}, I) \twoheadrightarrow (\mathcal{D}', L),$$

where $L \in Q$ because (N, I) accepts L. Now every surjective morphism between finite semilattices has the property that the domain has at least as many join-irreducibles as the codomain (since the join-irreducibles form the minimal generating set). Hence we have shown that

$$|J(Q)| \leq |J(\mathcal{P}Z)| = |Z|.$$

(3) Next apply the above construction to the underlying nfa of $\mathcal{N}(L)$. We obtain a JSL-dfa $\mathcal{D}_L = (Q_L, a^{-1}(-), \tilde{F})$ where Q_L is the semilattice of languages which this nfa can accept, varying the set of initial states. In fact:

(a) Q_L is precisely the set of unions of derivatives of L, see Proposition 3.16 and use that $\mathcal{N}(L)$ is reachable and (isomorphic to) a normalized nca by Corollary 3.12.

(b) Applying $\mathbb{G}_*$ to $(\mathcal{D}_L, L)$ one obtains the minimal pointed nca $\mathcal{N}(L)$. In particular, Z_L has states $J(Q_L)$ – see Lemma 2.19 and Proposition 3.24.

(c) Therefore $CZ_L = CGQ_L \cong Q_L$, since G and C define an equivalence.

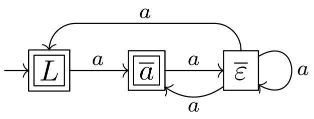
It follows from (a) that we have an injective Jdfa_*-morphism:

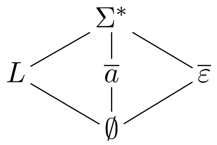
$$\iota : (\mathcal{D}_L, L) \hookrightarrow (\mathcal{D}', L).$$

Indeed $Q_L \subseteq Q$ because $L \in Q$ and the latter is closed under unions and derivatives. This inclusion certainly preserves unions. The transition structure and final states are inherited, so ι is well-defined.

(4) Now assume that $\mathcal{N}(L)$ is topological, hence $Q_L \cong CZ_L$ is distributive by Lemma 4.2. In view of (2), it remains to prove that $|J(Q_L)| \leq |J(Q)|$, for which we establish an auxiliary statement: given any finite distributive lattice D which is a sub semilattice of a finite semilattice S, we prove that $|J(D)| \leq |J(S)|$. Let $|J(S)| = n$, so we have a surjective join-semilattice homomorphism $\mathcal{P}n \twoheadrightarrow S$. By the self-duality of JSL_f we have an embedding of S^{op} into $(\mathcal{P}n)^{op} \cong \mathcal{P}n$. Thus any maximal chain in S (hence also in D) has at most n edges. Since the number of join-irreducibles in a finite distributive lattice equals the number of edges of any maximal chain [8, Corollary II.112], it follows that $|J(D)| \leq n = |J(S)|$. Now Q_L is a distributive sub-semilattice of Q by (3), so applying this result to $D = Q_L$ and $S = Q$ proves the theorem. □

Example 4.5 The converse of Theorem 4.4 is generally false. Let $L = \overline{aa}$ denote the complement of the singleton $\{aa\}$. The underlying nfa of $\mathcal{N}(L)$ and the lattice Q_L are depicted below:





Clearly Q_L is non-distributive but $\mathcal{N}(L)$ is a state-minimal nfa accepting L.

5 Conclusions and Future Work

It has been known since the early days of automata theory that nondeterministic finite automata suffer from two unpleasant phenomena, as opposed to their deterministic counterparts: the lack of canonical machines, and the lack of state-minimization. In this paper, we have demonstrated that both problems disappear when one augments nfas with a closure structure. Based on the equivalence between JSL-dfas and nondeterministic closure automata, we derived the "right" notion of minimality, which allowed us to establish the existence of a unique minimal nca for each regular language along with a Brzozowski-style construction method. Furthermore, by restricting to ncas with topological closure, we identified a very natural class of *canonical* state-minimal nfas.

One open question that we leave for future work is to what extent our main result (Theorem 4.4) can be reversed, that is, under which conditions the state-minimality of $\mathcal{N}(L)$ implies topologicity.

Another issue we aim to address in the future are the complexity-related implications of our results. Although the general state minimization problem for nfas is known to be PSPACE-complete, a good implementation of our operators reach and rev could lead to efficient state minimization procedures for the class of topological automata, and possibly even larger classes of automata.

References

[1] Adámek, J., S. Milius, L. S. Moss and L. Sousa, *Well-pointed Coalgebras* (2013), full version, available at www.stefan-milius.eu, extended abstract in: Proc. FoSSaCS 2012, Lecture Notes Comput. Sci., vol. 7321, pp. 89–103.

[2] Adámek, J., R. S. Myers, S. Milius and H. Urbat, *Canonical Nondeterministic Automata*, in: M. Bonsangue, editor, *Proceedings of the 12th International Workshop on Coalgebraic Methods in Computer Science*, 2014, to appear.

[3] Adámek, J., J. Rosický and E. Vitale, "Algebraic Theories," Cambridge University Press, 2011.

[4] Bonchi, F., M. M. Bonsangue, H. H. Hansen, P. Panangaden, J. J. M. M. Rutten and A. Silva, *Algebra-coalgebra Duality in Brzozowski's Minimization Algorithm*, ACM Trans. Comput. Logic **15** (2014), pp. 3:1–3:29.
URL http://doi.acm.org/10.1145/2490818

[5] Brzozowski, J. and H. Tamm, *Theory of Átomata*, in: *Proc. 15th International Conference on Developments in Language Theory (DLT'11)*, Lecture Notes Comput. Sci. **6795** (2011), pp. 105–116.

[6] Brzozowski, J. A., *Canonical regular expressions and minimal state graphs for definite events*, in: *Mathematical theory of Automata*, Volume 12 of MRI Symposia Series, Polytechnic Press, Polytechnic Institute of Brooklyn, N.Y., 1962 pp. 529–561.

[7] Denis, F., A. Lemay and A. Terlutte, *Residual Finite State Automata*, Fund. Inform. **XX** (2002), pp. 1–30.

[8] Grätzer, G., "Lattice Theory: Foundation," SpringerLink : Bücher, Birkhauser Boston, 2011.
URL http://books.google.co.uk/books?id=6XJX5-zCoIQC

[9] Milius, S., *A Sound and Complete Calculus for Finite Stream Circuits*, in: *Proc. 25th Annual Symposium on Logic in Computer Science (LICS'10)*, IEEE Computer Society, 2010, pp. 449–458.

A Appendix

We prove the claim made in the Introduction that the category JSL_f of finite semilattices arises as a free cocompletion of Rel_f. More precisely, JSL_f is both

(a) a free cocompletion of Rel_f under reflexive coequalizers, and

(b) a conservative cocompletion of Rel_f under finite colimits.

Let us us first recall the general concepts.

Definition A.1 Let $\mathcal{A}$ be a category. A *free cocompletion under reflexive coequalizers* of $\mathcal{A}$ is a full embedding $E : \mathcal{A} \hookrightarrow \mathcal{A}'$ such that

(i) $\mathcal{A}'$ has reflexive coequalizers (i.e., coequalizers of pairs of retractions $f, g : X \to Y$ having a common coretraction $d : Y \to X$, $fd = \mathsf{id}_Y = gd$).

(ii) Every functor $F : \mathcal{A} \to \mathcal{B}$, where $\mathcal{B}$ has reflexive coequalizers, has an extension $F' : \mathcal{A}' \to \mathcal{B}$ (i.e., $F'E \cong F$) preserving reflexive coequalizers, which is unique up to natural isomorphism.

Definition A.2 Let $\mathcal{A}$ be a category with finite coproducts. A *conservative finite cocompletion* of $\mathcal{A}$ is a full embedding $E : \mathcal{A} \hookrightarrow \mathcal{A}'$ such that

(i*) E preserves finite coproducts and $\mathcal{A}'$ is finitely cocomplete.

(ii*) Every finite-coproduct preserving functor $F : \mathcal{A} \to \mathcal{B}$, where $\mathcal{B}$ is finitely cocomplete, has an extension to a finite-colimit preserving functor $F' : \mathcal{A}' \to \mathcal{B}$ (i.e. $F'E \cong F$) which is unique up to natural isomorphism.

Example A.3 Let $\mathcal{V}$ be a finitary variety, and let $\mathcal{V}_{fp}$ and $\mathcal{V}_{fgf}$ be the full subcategories of all finite presentable and finitely generated free algebras, respectively. Then $\mathcal{V}_{fp}$ is a free cocompletion of $\mathcal{V}_{fgf}$ under reflexive coequalizers, as well as a conservative finite cocompletion of $\mathcal{V}_{fgf}$, see [3, Theorem 7.3 and Theorem 17.11].

Corollary A.4 *JSL_f is a free cocompletion of Rel_f under reflexive coequalizers, as well as a conservative finite cocompletion of Rel_f.*

Proof. Apply the above example to the variety $\mathcal{V} = \mathsf{JSL}$. Here $\mathcal{V}_{fp} = \mathsf{JSL}_f$. Since finitely generated free semilattices are power sets $\mathcal{P}n$ $(n < \omega)$, it is easy to see that $\mathcal{V}_{fgf}$ is equivalent to Rel_f. Indeed, the functor $\mathcal{P} : \mathsf{Rel}_f \to \mathsf{JSL}_{fgf}$ assigning to each finite set n its power set and to each relation $h : n \to m$ the semilattice morphism

$$\mathcal{P}h : \mathcal{P}n \to \mathcal{P}m, \quad A \mapsto h[A],$$

is an equivalence of categories. □

Coalgebraic Update Lenses

Danel Ahman[1]

Laboratory for Foundations of Computer Science, University of Edinburgh,
10 Crichton Street, Edinburgh EH8 9LE, United Kingdom

Tarmo Uustalu[2]

Institute of Cybernetics at Tallinn University of Technology,
Akadeemia tee 21, 12618 Tallinn, Estonia

Abstract

Lenses are mathematical structures used in the context of bidirectional transformations.
In this paper, we introduce update lenses as a refinement of ordinary (asymmetric) lenses in which we distinguish between views and updates. In addition to the set of views, there is a monoid of updates and an action of the monoid on the set of views. Decoupling updates from views allows for other ways of changing the source than just merging a view into the source. We also consider a yet finer dependently typed version of update lenses.
We give a number of characterizations of update lenses in terms of bialgebras and coalgebras, including analogs to O'Connor's coalgebraic and Johnson, Rosebrugh and Wood's algebraic characterizations of ordinary lenses. We consider conversion of views and updates, a tensor product of update lenses and composition of update lenses.

Keywords: lenses, comonads, monads, coalgebras, algebras, bialgebras, distributive laws, liftings, monoid acts, directed containers

1 Introduction

Lenses are studied in the context of bidirectional transformations. A lens is a structure on two sets mediating actions on them. Think of making sure that two copies of a database on a server and a client computer remain in agreement no matter how the database is changed on the client's side; this must also work when the client only sees a part of the database, a "view".

Many different variations of lenses have been considered in the literature. In the seminal work [8], a *lens* between X and S was defined as a pair of maps $\mathsf{lkp} : X \to S$ and $\mathsf{upd} : X \times S \to X$, subject to appropriate round-trip laws. Here, X is to be

[1] d.ahman@ed.ac.uk

[2] tarmo@cs.ioc.ee

thought of as a set of sources and S as a set of views. O'Connor [18] showed that such lenses are nothing but coalgebras of the array comonad for S. Johnson, Rosebrugh and Wood [14] and Gibbons and Johnson [10] characterized lenses as algebras for a certain monad on $\mathbf{Set}/S$.

Other variants of lenses include, for example, quotient lenses [9], symmetric lenses [12], (symmetric) edit lenses [11], and (asymmetric) delta lenses [6].

In this paper, we define a variation of ordinary (asymmetric) lenses, which we call *update lenses* (this name is justified by their practical motivation and agrees with the terminology of our prior related work [3] on update monads). In short, update lenses differ from ordinary state-based lenses in that they distinguish between views and updates. In addition to the set of views S, there is also a monoid $(P, \mathsf{o}, \oplus)$ of updates and an action $\downarrow$ of $(P, \mathsf{o}, \oplus)$ on S. The update map thus becomes $\mathsf{upd} : X \times P \to X$, allowing for other ways of changing the source than just merging a view into the source. There is also a dependently typed version where every view $s : S$ comes with its own set of updates $P\,s$.

We give a number of characterizations of update lenses in terms of bialgebras, algebras and coalgebras, which we derive from standard facts about distributive laws between comonads/functors, compatible composition of comonads, liftings of comonads/functors to categories of coalgebras, distributive laws of monads/functors over comonads/functors, cofree comonads, adjoint monads and comonads etc. [7,5,4,20,21]. We study conversion of views and updates, a tensor of update lenses, composition of update lenses. We also look at a specialization, initializable update lenses. For lack of space, this appears in the appendix.

Of the related work, the closest to ours is that of Hofmann, Pierce and Wagner [11] on (symmetric) edit lenses. They analyze symmetric lenses, and recognize that edits (in our terminology updates) often carry a natural monoidal structure, and are thus best modeled using monoids. As a result of the symmetric setting, edit lenses consist of two monoids, with the upd operations given by a suitable variation of monoid homomorphisms. A different line of related work is that of Diskin, Xiong, and Czarnecki [6] and Johnson and Rosebrugh [15], on (asymmetric) delta lenses. The latter group of authors also introduced a subclass of delta lenses, called c-lenses, with a more succinct definition. Both groups of authors investigate adding more structure to the sets of sources and views, taking them to be reflexive graphs (or, equivalently, categories). The nodes model sources or views, and arrows model deltas (in our terminology updates) between them. The lkp and upd operations rely on functoriality to guarantee the correct and composable translation of deltas.

For readability, we develop all our mathematics for $\mathbf{Set}$ (and categories built on $\mathbf{Set}$), but for nearly everything we do it could be replaced by any category with finite products or any Cartesian closed category.

2 Ordinary lenses

Ordinary lenses: the definition

Given a set S (of views), an *ordinary* (asymmetric) *lens* for S is a set X (of sources) and maps $\mathsf{lkp} : X \to S$ (viewing a source) and $\mathsf{upd} : X \times S \to X$ (updating a source)[3] satisfying the conditions

$$x = \mathsf{upd}\,(x, \mathsf{lkp}\,x) \qquad \mathsf{upd}\,(\mathsf{upd}\,(x,s), s') = \mathsf{upd}\,(x, s') \qquad \mathsf{lkp}\,(\mathsf{upd}\,(x,s)) = s$$

i.e., making the following three diagrams commute

$$\begin{tikzcd} X \arrow[d, "\langle \mathsf{id},\mathsf{lkp}\rangle"'] \arrow[dr, equal] & \\ X \times S \arrow[r, "\mathsf{upd}"] & X \end{tikzcd} \qquad \begin{tikzcd} (X \times S) \times S \arrow[r, "\mathsf{upd} \times S"] \arrow[d, "\langle \mathsf{fst},\mathsf{id}\rangle"'] & X \times S \arrow[d, "\mathsf{upd}"] \\ X \times S \arrow[r, "\mathsf{upd}"] & X \end{tikzcd} \qquad \begin{tikzcd} X \times S \arrow[r, "\mathsf{upd}"] \arrow[dr, "\mathsf{snd}"'] & X \arrow[d, "\mathsf{lkp}"] \\ & S \end{tikzcd}$$

A *map* between two lenses $(X, \mathsf{lkp}, \mathsf{upd})$, $(X', \mathsf{lkp}', \mathsf{upd}')$ is a map $h : X \to X'$ (conversion of a source) satisfying

$$\mathsf{lkp}\,x = \mathsf{lkp}'\,(h\,x) \qquad h\,(\mathsf{upd}\,(x,s)) = \mathsf{upd}'(h\,x, s)$$

i.e.,

$$\begin{tikzcd} X \arrow[r, "h"] \arrow[d, "\mathsf{lkp}"'] & X' \arrow[d, "\mathsf{lkp}'"] \\ S \arrow[r, equal] & S \end{tikzcd} \qquad \begin{tikzcd} X \times S \arrow[r, "h \times S"] \arrow[d, "\mathsf{upd}"'] & X' \times S \arrow[d, "\mathsf{upd}'"] \\ X \arrow[r, "h"] & X' \end{tikzcd}$$

Lenses for S and maps between them form a category $\mathbf{Lens}\,S$.

Running example: editing a bookshop database

Consider modelling a database of a small bookshop. We will work with a very simple database, consisting of only one table, containing information about the identity (e.g., book title and author names, or ISBN), price and quantity of every book in the shop. We let *book*, *price*, *quantity* stand for the sets of allowed values of these data.

The set of database states or sources X is

$$X = \Sigma B \subseteq \mathit{book}.\, B \to \mathit{price} \times \mathit{quantity}$$

Every such database state consists of a set of books B currently in the database, and an assignment of *price* and *quantity* to each book in B.

A simple example of a set of views S could be

$$S = \Sigma B \subseteq \mathit{book}.\, B \to \mathit{price}$$

[3] We use the letter $\underline{X}$ for the $\underline{s}$ource set and $\underline{S}$ for the $\underline{v}$iew set, and the names lkp for get and upd for put. This may at first feel confusing, but agrees with what is mnemonic for algebras of update monads [3].

This set of views can be used to define an ordinary lens for editing book prices as follows.

We define $\mathsf{lkp} : X \to S$ by simply discarding the *quantity* field of each book, that is

$$\mathsf{lkp}\,(B, v) = (B, \mathsf{fst} \circ v)$$

We define $\mathsf{upd} : X \times S \to X$ by removing all books that are in the database but not in the view, editing the prices and preserving the quantities of all books that are both in the database and the view, and adding all books that are only in the view with quantity 0, that is

$$\mathsf{upd}\,((B, v), (B', v')) = (B', \lambda b.\,\mathsf{if}\ b \in B\ \mathsf{then}\ (v'\,b, \mathsf{snd}\,(v\,b))\ \mathsf{else}\ (v'\,b, 0))$$

Observe that, for our structure to satisfy the 3rd ordinary lens law, the view argument we pass to upd has to contain all the books we want to leave in the database after the update. Also, as we cannot restrict which books can be edited, we have to allow the lens to be able to add new books to the source, with some default quantity, here 0.

We claim that such forced choices are a real shortcoming of ordinary lenses, and address it in the rest of this paper by developing the theory of update lenses.

Ordinary lenses: some alternative descriptions

Johnson et al. [14] pointed out the category of lenses for S is the same as the category of algebras of the following monad (T_1, η_1, μ_1) on $\mathbf{Set}/S$:

$$\begin{aligned} T_1\,(X, \mathsf{lkp}) &= (X \times S, \mathsf{snd}) \\ \eta_1\,\{X, \mathsf{lkp}\}\,x &= (x, \mathsf{lkp}\,x) \\ \mu_1\,((x, s), s') &= (x, s') \end{aligned}$$

Indeed, an algebra of (T_1, η_1, μ_1) is a map $\mathsf{upd} : (X \times S, \mathsf{snd}) \to (X, \mathsf{lkp})$ in $\mathbf{Set}/S$, i.e., a map $\mathsf{upd} : X \times S \to X$ satisfying $\mathsf{lkp}\,(\mathsf{upd}\,(x, s)) = s$, which must moreover also satisfy the conditions of an algebra structure of (T_1, η_1, μ_1). The latter amount exactly to $x = \mathsf{upd}\,(x, \mathsf{lkp}\,x)$ and $\mathsf{upd}\,(\mathsf{upd}\,(x, s), s') = \mathsf{upd}\,(x, s')$.

O'Connor [18] observed that the category of lenses is isomorphic to the category of coalgebras of the following comonad (D, ε, δ) on $\mathbf{Set}$ (sometimes [19] called the *array comonad* for S):

$$\begin{aligned} D\,X &= S \times (S \to X) \\ \varepsilon\,(s, v) &= v\,s \\ \delta\,(s, v) &= (s, \lambda s'.\,(s', v)) \end{aligned}$$

Explictly, a coalgebra of (D, ε, δ) is a set X and a map $\mathsf{act} : X \to S \times (S \to X)$ satisfying

$$x = \mathsf{let}\ (s, v) \leftarrow \mathsf{act}\,x\ \mathsf{in}\ v\,s$$
$$\mathsf{let}\ (s, v) \leftarrow \mathsf{act}\,x\ \mathsf{in}\ (s, \lambda s'.\,\mathsf{act}\,(v\,s')) = \mathsf{let}\ (s, v) \leftarrow \mathsf{act}\,x\ \mathsf{in}\ (s, \lambda s'.\,(s', v))$$

The isomorphism assigns to a lens structure $(\mathsf{lkp}, \mathsf{upd})$ on a set X the (D, ε, δ)-coalgebra structure $\mathsf{act} = \langle \mathsf{lkp}, \mathsf{cur}\, \mathsf{upd} \rangle$ on X.

A further characterization, due to Power and Shkaravska [19], is only an equivalence of categories, not an isomorphism. The category of lenses for S is equivalent to **Set**. With a set R, one associates the lens $(X, \mathsf{lkp}, \mathsf{upd})$ defined by $X = S \times R$, $\mathsf{lkp}\,(s, r) = s$, $\mathsf{upd}\,((s, r), s') = (s', r)$. In the converse direction, a lens $(X, \mathsf{lkp}, \mathsf{upd})$ is sent to the set $R = \{x : X \mid \mathsf{lkp}\, x = s_0\}$ where s_0 is some chosen element of S. The choice of s_0 only makes the difference of an isomorphism: for any $s, s' : S$, the function $\lambda x.\, \mathsf{upd}\,(x, s') : \{x : X \mid \mathsf{lkp}\, x = s\} \to \{x : X \mid \mathsf{lkp}\, x = s'\}$ is an isomorphism by the 2nd and 3rd lens conditions. (In the special case $S = 0$, we take $R = 0$.)

Intuitively, X is decomposed into two parts: S, which can be both viewed and updated, and R, which can be neither viewed nor updated.

Composition of ordinary lenses

Rather than thinking of S (the set of views) as a fixed set, we can let it vary. We can then define that an (ordinary) lens between two sets X and Y is a pair of maps $\mathsf{lkp} : X \to Y$ and $\mathsf{upd} : X \times Y \to X$ satisfying the three conditions.

For any set X, there is the identity lens $(\mathsf{id}\,\{X\}, \mathsf{fst})$. Given two lenses $(\mathsf{lkp}, \mathsf{upd}) : X \to Y$ and $(\mathsf{lkp}', \mathsf{upd}') : Y \to Z$, we define their composition to be $(\mathsf{lkp}'', \mathsf{upd}'') : X \to Z$ where $\mathsf{lkp}''\, x = \mathsf{lkp}'\,(\mathsf{lkp}\, x)$ and $\mathsf{upd}''\,(x, z) = \mathsf{upd}\,(x, \mathsf{upd}'\,(\mathsf{lkp}\, x, z))$. Sets and lenses between them for a category.

3 Update lenses

In this paper, we are interested in a more fine-grained variation of lenses that we call update lenses.

Update lenses: the definition

We define an *act* to be given by a set S (of views), a monoid $(P, \mathsf{o}, \oplus)$ (of updates) and an action $\downarrow$ of the monoid on the set (describing the outcome of applying any given update on any given view).[4]

An *update lens* for an act $(S, (P, \mathsf{o}, \oplus), \downarrow)$ is a set X (of sources) and maps $\mathsf{lkp} : X \to S$ (viewing a source) and $\mathsf{upd} : X \times P \to X$ (updating a source) satisfying the conditions

$$x = \mathsf{upd}\,(x, \mathsf{o}) \qquad \mathsf{upd}\,(\mathsf{upd}\,(x, p), p') = \mathsf{upd}\,(x, p \oplus p') \qquad \mathsf{lkp}\,(\mathsf{upd}\,(x, p)) = \mathsf{lkp}\, x \downarrow p$$

[4] In the literature, the pair $(S, \downarrow)$ is often called a $(P, \mathsf{o}, \oplus)$-*set*. We will occasionally use this terminology too.

i.e.,

$$\begin{tikzcd}
X \arrow[d, "\rho"'] \arrow[ddr, equal] & \\
X \times 1 \arrow[d, "X \times \mathsf{o}"'] & \\
X \times P \arrow[r, "\mathsf{upd}"] & X
\end{tikzcd}
\qquad
\begin{tikzcd}
(X \times P) \times P \arrow[r, "\mathsf{upd} \times P"] \arrow[d, "\alpha"'] & X \times P \arrow[dd, "\mathsf{upd}"] \\
X \times (P \times P) \arrow[d, "X \times \oplus"'] & \\
X \times P \arrow[r, "\mathsf{upd}"] & X
\end{tikzcd}
\qquad
\begin{tikzcd}
X \times P \arrow[r, "\mathsf{upd}"] \arrow[dd, "\mathsf{lkp} \times P"'] & X \arrow[dd, "\mathsf{lkp}"] \\
& \\
S \times P \arrow[r, "\downarrow"] & S
\end{tikzcd}$$

A *map* between two update lenses $(X, \mathsf{lkp}, \mathsf{upd})$, $(X', \mathsf{lkp}', \mathsf{upd}')$ is a map $h : X \to X'$ (conversion of a source) satisfying

$$\mathsf{lkp}\, x = \mathsf{lkp}'\,(h\, x) \qquad h\,(\mathsf{upd}\,(x, p)) = \mathsf{upd}'(h\, x, p)$$

i.e.,

$$\begin{tikzcd}
X \arrow[r, "h"] \arrow[d, "\mathsf{lkp}"'] & X' \arrow[d, "\mathsf{lkp}'"] \\
S \arrow[r, equal] & S
\end{tikzcd}
\qquad
\begin{tikzcd}
X \times P \arrow[r, "h \times P"] \arrow[d, "\mathsf{upd}"'] & X' \times P \arrow[d, "\mathsf{upd}'"] \\
X \arrow[r, "h"] & X'
\end{tikzcd}$$

Update lenses for an act $(S, (P, \mathsf{o}, \oplus), \downarrow)$ and maps between them form a category $\mathbf{ULens}\ (S, (P, \mathsf{o}, \oplus), \downarrow)$.

Running example: editing a bookshop database

We now revisit the bookshop example from Section 2.

We first recall that it was somewhat inconvenient to use the ordinary bookshop lens to edit the price of a particular book. In particular, we could not simply list the books whose prices we wanted to edit, but had to also list all the other books. The main cause for this inconvenience was the single set of views S that got used in both lkp and upd. Here we illustrate how decoupling the updates from the views helps us reuse the same set of views for different update strategies.

We keep the set of sources X, the set of views S and the definition of lkp as before, but can let $(P, \mathsf{o}, \oplus)$, $\downarrow$, upd depend on what we want to do with the database.

For the first example, we reconsider editing book prices by taking P to be $P = (\mathit{book} \times \mathit{price})^*$. Any update $\mathit{ps} : P$ is to be read as a sequence of price modifications of single books. The monoid structure $(\mathsf{o}, \oplus)$ is that of free monoids (with $\mathsf{o} = []$ and $\oplus = +\!\!+$), while $\downarrow$ and upd are defined similarly.

$$\begin{array}{l}
(B, v) \downarrow [] = (B, v) \\
(B, v) \downarrow (b, p) :: \mathit{ps}) = (B, \lambda b'.\, \mathsf{if}\ b' = b\ \mathsf{then}\ p\ \mathsf{else}\ v\, b') \downarrow \mathit{ps}
\end{array}$$

$$\begin{array}{l}
\mathsf{upd}\,((B, v), []) = (B, v) \\
\mathsf{upd}\,((B, v), (b, p) :: \mathit{ps}) = \mathsf{upd}\,((B, \lambda b'.\, \mathsf{if}\ b' = b\ \mathsf{then}\ (p, \mathsf{snd}\,(v\, b'))\ \mathsf{else}\ v\, b'), \mathit{ps})
\end{array}$$

(With this definition, if a book to modify is not in the database, it is not added!)

Thanks to the freedom offered by update lenses, we can equally well modify the book prices relative to the prices in the database. For this purpose, we can define P to be $(\mathit{book} \times \mathit{change})^*$ (let us say a price can only be a nonnegative number, but a change can also be negative). The monoid structure $(P, \mathsf{o}, \oplus)$ is as above, but the definitions of $\downarrow$ and upd are adjusted.

$$
\begin{array}{l}
(B, v) \downarrow [] = (B, v) \\
(B, v) \downarrow (b, c) :: ps) = (B, \lambda b'.\, \mathsf{if}\ b' = b\ \mathsf{then}\ \max(0, v\, b' + c))\ \mathsf{else}\ v\, b') \downarrow ps
\end{array}
$$

$$
\begin{array}{l}
\mathsf{upd}\,((B, v), []) = (B, v) \\
\mathsf{upd}\,((B, v), (b, c) :: ps) = \\
\qquad\qquad \mathsf{upd}\,((B, \lambda b'.\, \mathsf{if}\ b' = b\ \mathsf{then}\ (\max(0, \mathsf{fst}\,(v\, b') + c), \mathsf{snd}\,(v\, b'))\ \mathsf{else}\ v\, b'), ps)
\end{array}
$$

As an additional treat, we can also use the same set of views S to perform both deletions and additions of books. Here we choose P to be $((\mathit{book} \times \mathit{price} \times \mathit{quantity}) + \mathit{book})^*$ where the first summand stands for addition of a book and the second summand for deletion of a book. The monoid structure $(P, \mathsf{o}, \oplus)$ is again that of free monoids.

$$
\begin{array}{l}
(B, v) \downarrow [] = (B, v) \\
(B, v) \downarrow (\mathsf{inl}\,(b, p, q) :: ps) = (B \cup \{b\}, \lambda b'.\, \mathsf{if}\ b' = b\ \mathsf{then}\ p\ \mathsf{else}\ v\, b') \downarrow ps \\
(B, v) \downarrow (\mathsf{inr}\, b :: ps) = (B \backslash \{b\}, v|_{B \backslash \{b\}}) \downarrow ps
\end{array}
$$

$$
\begin{array}{l}
\mathsf{upd}\,((B, v), []) = (B, v) \\
\mathsf{upd}\,((B, v), \mathsf{inl}\,(b, p, q) :: ps) = \mathsf{upd}\,((B \cup \{b\}, \lambda b'.\, \mathsf{if}\ b' = b\ \mathsf{then}\ (p, q)\ \mathsf{else}\ v\, b'), ps) \\
\mathsf{upd}\,((B, v), \mathsf{inr}\, b :: ps) = \mathsf{upd}\,((B \backslash \{b\}, v|_{B \backslash \{b\}}), ps)
\end{array}
$$

Of course, we could also combine addition and deletion of entries with modification of entries, if we wanted.

Notice that changing a book's price and adding a book are implemented somewhat arbitrarily, if the absolute value of a negative change is greater than the book's price resp. if the book is already in the database. This is because every update must always be applicable. In Section 7, we will address this deficiency.

Update lenses as bialgebras of a functor and monad

The category of lenses for $(S, (P, \mathsf{o}, \oplus), \downarrow)$ admits several alternative characterizations. We will now consider these in turn. The ultimate insight will be that update lenses are coalgebras of an appropriate comonad. We will arrive there in several small steps through intermediate characterizations that are also of independent value. We only use standard facts about monads, comonads, distributive laws, liftings, cofree comonads, adjoint monads and comonads. These appear scattered in the literature, some references include [7,5,4,20,21].

We begin by noting that a map lkp from a set X to S is nothing but a coalgebra structure on X of the functor F_0 defined by $F_0\, X = S$. Also an action upd of $(P, \mathsf{o}, \oplus)$

on X is exactly an algebra structure on X of the monad (T_1, η_1, μ_1) defined by

$$\begin{aligned} T_1\, X &= X \times P \\ \eta_1\, x &= (x, \mathsf{o}) \\ \mu_1\, ((x, p), p') &= (x, p \oplus p') \end{aligned}$$

Finally, an action $\downarrow : S \times P \to S$ of $(P, \mathsf{o}, \oplus)$ on S is exactly a distributive law of the monad (T_1, η_1, μ_1) over the functor F_0.

Therefore, an update lens $(X, \mathsf{lkp}, \mathsf{upd})$ for $(S, (P, \mathsf{o}, \oplus), \downarrow)$ is nothing but a $\downarrow$-bialgebra of the functor F_0 and monad (T_1, η_1, μ_1), i.e., a triple of a set X, map $\mathsf{lkp} : X \to S$ and algebra structure $\mathsf{upd} : X \times P \to X$ of (T_1, η_1, μ_1) cohering in the sense of the condition

$$\begin{tikzcd} X \times P \arrow[r, "\mathsf{upd}"] \arrow[d, "\mathsf{lkp} \times P"'] & X \arrow[r, "\mathsf{lkp}"] & S \arrow[d, equal] \\ S \times P \arrow[rr, "\downarrow"] & & S \end{tikzcd}$$

In the same way, update lens maps are bialgebra maps. So the category of update lenses is the same as

(0) the category of $\downarrow$-bialgebras of the functor F_0 and monad (T_1, η_1, μ_1).

It follows that this category can also be described as:

(0') the category of coalgebras of the $\downarrow$-lifting $F_0^{\downarrow}$ of F_0 to the category of algebras of (T_1, η_1, μ_1);

(0") the category of algebras of the $\downarrow$-lifting $(T_1^{\downarrow}, \eta_1^{\downarrow}, \mu_1^{\downarrow})$ of (T_1, η_1, μ_1) to the category of coalgebras of F_0.

Let us spell out (0') and (0").

(0'): The category of algebras of (T_1, η_1, μ_1) is $(P, \mathsf{o}, \oplus)$-Set. The $\downarrow$-lifting $F_0^{\downarrow}$ of F_0 to $(P, \mathsf{o}, \oplus)$-Set is defined by $F_0^{\downarrow}(X, \mathsf{upd}) = (S, \downarrow)$. A coalgebra of $F_0^{\downarrow}$ is therefore given by a $(P, \mathsf{o}, \oplus)$-set (X, upd) with a map lkp to $(S, \downarrow)$. So the category of coalgebras of $F_0^{\downarrow}$ is $(P, \mathsf{o}, \oplus)$-$\mathsf{Set}/(S, \downarrow)$.

(0"): The category of coalgebras of F_0 is $\mathbf{Set}/S$. The $\downarrow$-lifting $(T_1^{\downarrow}, \eta_1^{\downarrow}, \mu_1^{\downarrow})$ of (T_1, η_1, μ_1) is defined by

$$T_1^{\downarrow}\, (X, \mathsf{lkp}) = (X \times P, \lambda(x, p).\, \mathsf{lkp}\, x \downarrow p)$$

An algebra of $(T_1^{\downarrow}, \eta_1^{\downarrow}, \mu_1^{\downarrow})$ is an object (X, lkp) of $\mathbf{Set}/S$ with a map upd from $(X \times P, \lambda(x, p).\, \mathsf{lkp}\, x \downarrow p)$ satisfying the conditions of a monad algebra.

Update lenses as bialgebras of a comonad and monad

Let $(D_0, \varepsilon_0, \delta_0)$ be the cofree comonad on the functor F_0, i.e., explicitly

$$\begin{aligned} D_0\, X &= S \times X \\ \varepsilon_0\, (s, x) &= x \\ \delta_0\, (s, x) &= (s, (s, x)) \end{aligned}$$

The category of coalgebras of F_0 (i.e., the category $\mathbf{Set}/S$) is isomorphic to the category of coalgebras of $(D_0, \varepsilon_0, \delta_0)$, whose objects are given by a set X and map $\mathsf{dlkp} : X \to S \times X$ satisfying

$$\begin{tikzcd}
X \arrow[r, "\mathsf{dlkp}"] \arrow[dr, equal] & S \times X \arrow[d, "\mathsf{snd}"] \\
& X
\end{tikzcd}
\qquad
\begin{tikzcd}
X \arrow[r, "\mathsf{dlkp}"] \arrow[d, "\mathsf{dlkp}"'] & S \times X \arrow[d, "\langle \mathsf{fst}, \mathsf{id} \rangle"] \\
S \times X \arrow[r, "S \times \mathsf{dlkp}"'] & S \times (S \times X)
\end{tikzcd}$$

The isomorphism assigns to any F_0-coalgebra structure $\mathsf{lkp} : X \to S$ on a set X the $(D_0, \varepsilon_0, \delta_0)$-coalgebra structure $\mathsf{dlkp} = \langle \mathsf{lkp}, \mathsf{id} \rangle : X \to S \times X$.

Furthermore, the distributive law $\downarrow$ of (T_1, η_1, μ_1) over F_0 induces a distributive law λ of (T_1, η_1, μ_1) over $(D_0, \varepsilon_0, \delta_0)$, explicitly defined by

$$\begin{array}{c}
\lambda : (S \times X) \times P \to S \times (X \times P) \\
\lambda\,((s, x), p) = (s \downarrow p, (x, p))
\end{array}$$

As a result, the category of update lenses, i.e, the category of $\downarrow$-bialgebras of F_0 and (T_1, η_1, μ_1), is isomorphic to

(1) the category of λ-bialgebras of $(D_0, \varepsilon_0, \delta_0)$ and (T_1, η_1, μ_1).

Explictly, the objects of this category are triples of a set X, $(D_0, \varepsilon_0, \delta_0)$-coalgebra structure $\mathsf{dlkp} : X \to S \times X$ and (T_1, η_1, μ_1)-algebra structure $\mathsf{upd} : X \times P \to X$ satisfying

$$\begin{tikzcd}
X \times P \arrow[r, "\mathsf{upd}"] \arrow[d, "\mathsf{dlkp} \times P"'] & X \arrow[r, "\mathsf{dlkp}", "= \langle \mathsf{lkp}, \mathsf{id} \rangle"'] & S \times X \\
(S \times X) \times P \arrow[rr, "\lambda"] & & S \times (X \times P) \arrow[u, "S \times \mathsf{upd}"']
\end{tikzcd}$$

This category can also be described as:

(1') the category of coalgebras of the λ-lifting $(D_0^\lambda, \varepsilon_0^\lambda, \delta_0^\lambda)$ of $(D_0, \varepsilon_0, \delta_0)$ to the category of algebras of (T_1, η_1, μ_1);

(1") the category of algebras of the λ-lifting $(T_1^\lambda, \eta_1^\lambda, \mu_1^\lambda)$ of (T_1, η_1, μ_1) to the category of coalgebras of $(D_0, \varepsilon_0, \delta_0)$.

Update lenses as pairs of coalgebras of a functor and comonad

Instead of replacing the functor F_0 with a comonad, we could replace the monad (T_1, η_1, μ_1) with a different comonad. This is what we will embark on now.

Consider the following comonad $(D_1, \varepsilon_1, \delta_1)$:

$$\begin{array}{c}
D_1\, X = P \to X \\
\varepsilon_1\, v = v\, \mathsf{o} \\
\delta_1\, v = \lambda p.\, \lambda p'.\, v\, (p \oplus p')
\end{array}$$

The functor D_1 is the right adjoint of the functor T_1. What is more, the comonad $(D_1, \varepsilon_1, \delta_1)$ is the corresponding "right adjoint" of the monad (T_1, η_1, μ_1) in the sense of [7]. As a consequence, the category of algebras of (T_1, η_1, μ_1) is isomorphic to the category of coalgebras of $(D_1, \varepsilon_1, \delta_1)$. Explicitly, a coalgebra of $(D_1, \varepsilon_1, \delta_1)$ is an object X with a map $\mathsf{cupd} : X \to P \to X$ satisfying

$$\begin{tikzcd}
X \arrow[r, "\mathsf{cupd}"] \arrow[ddr, equal] & P \to X \arrow[d, "\mathsf{o} \to X"] \\
& 1 \to X \arrow[d] \\
& X
\end{tikzcd}
\qquad
\begin{tikzcd}
X \arrow[rr, "\mathsf{cupd}"] \arrow[dd, "\mathsf{cupd}"'] & & P \to X \arrow[d, "\oplus \to X"] \\
& & P \times P \to X \arrow[d] \\
P \to X \arrow[rr, "P \times \mathsf{cupd}"] & & P \to (P \to X)
\end{tikzcd}$$

The isomorphism assigns to a (T_1, η_1, μ_1)-algebra structure $\mathsf{upd} : X \times P \to X$ on a set X the $(D_1, \varepsilon_1, \delta_1)$-coalgebra structure $\mathsf{cupd} = \mathsf{cur}\,\mathsf{upd} : X \to P \to X$.

Further the distributive law $\downarrow$ of (T_1, η_1, μ_1) over F_0 induces a distributive law $\lightning$ of F_0 over $(D_1, \varepsilon_1, \delta_1)$, namely $\lightning = \mathsf{cur} \downarrow$. It follows that the category of $\downarrow$-bialgebras of F_0 and (T_1, η_1, μ_1) is isomorphic to

(2) the category of $\lightning$-matching pairs of coalgebras of F_0 and $(D_1, \varepsilon_1, \delta_1)$.

Explicitly, the objects of this category are given by triples of a set X, map $\mathsf{lkp} : X \to S$ and coalgebra structure $\mathsf{cupd} : X \to P \to X$ of $(D_1, \varepsilon_1, \delta_1)$ satisfying

$$\begin{tikzcd}
S \arrow[d, equal] & X \arrow[l, "\mathsf{lkp}"'] \arrow[r, "\mathsf{cupd}", "= \mathsf{cur}\,\mathsf{upd}"'] & P \to X \arrow[d, "P \to \mathsf{lkp}"] \\
S \arrow[rr, "\lightning"] & & P \to S
\end{tikzcd}$$

This category is also describable as

(2') the category of coalgebras of the lifting $F_0^{\lightning}$ of the functor F_0 to the category of coalgebras of $(D_1, \varepsilon_1, \delta_1)$.

Update lenses as pairs of coalgebras of two comonads

We now combine what we did in the last two paragraphs. We replace both F_0 with $(D_0, \varepsilon_0, \delta_0)$ and (T_1, η_1, μ_1) with $(D_1, \varepsilon_1, \delta_1)$. Both the distributive law λ of (T_1, η_1, μ_1) over $(D_0, \varepsilon_0, \delta_0)$ and the distributive law $\lightning$ of F_0 over $(D_1, \varepsilon_1, \delta_1)$ give us the following distributive law θ of $(D_0, \varepsilon_0, \delta_0)$ over $(D_1, \varepsilon_1, \delta_1)$:

$$\begin{gathered}
\theta : S \times (P \to X) \to P \to S \times X \\
\theta\,(s, v) = \lambda p.\,(s \downarrow p, v\,p)
\end{gathered}$$

The category of update lenses, i.e., the category of $\downarrow$-bialgebras of the functor F_0 and monad (T_1, η_1, μ_1), is therefore isomorphic to

(3) the category of θ-matching pairs of coalgebras of $(D_0, \varepsilon_0, \delta_o)$ and $(D_1, \varepsilon_1, \delta_1)$.

Explicitly, an object of this category is given by a set X, $(D_0, \varepsilon_0, \delta_0)$-coalgebra structure $\mathsf{dlkp} : X \to S \times X$ and $(D_1, \varepsilon_1, \delta_1)$-coalgebra structure $\mathsf{cupd} : X \to P \to X$

satisfying

$$\begin{tikzcd}
S \times X \arrow[d, "S\times\mathsf{cupd}"'] & X \arrow[l, "\mathsf{dlkp}"', "=\langle\mathsf{lkp},\mathsf{id}\rangle"] \arrow[r, "\mathsf{cupd}", "=\mathsf{cur}\,\mathsf{upd}"'] & P \to X \arrow[d, "P\to\mathsf{dlkp}"] \\
S \times (P \to X) \arrow[rr, "\theta"] & & P \to S \times X
\end{tikzcd}$$

The same category can also be described as

(3') the category of coalgebras of the θ-lifting $(D_0^\theta, \varepsilon_0^\theta, \delta_0^\theta)$ of the comonad $(D_0, \varepsilon_0, \delta_0)$ to the category of coalgebras of $(D_1, \varepsilon_1, \delta_1)$.

Update lenses as coalgebras of a comonad

We have two comonads $(D_0, \varepsilon_0, \delta_0)$ and $(D_1, \varepsilon_1, \delta_1)$ with a distributive law θ of the former over the latter. This gives a compatible composite comonad (D, ε, δ):

$$\begin{gathered}
D\,X = S \times (P \to X) \\
\varepsilon\,(s, v) = v\,\mathsf{o} \\
\delta\,(s, v) = (s, \lambda p.\,(s \downarrow p, \lambda p'.\,v\,(p \oplus p')))
\end{gathered}$$

The category of update lenses, i.e., of θ-matching pairs of coalgebras of $(D_0, \varepsilon_0, \delta_0)$ and $(D_1, \varepsilon_1, \delta_1)$, is isomorphic to

(4) the category of (D, ε, δ)-coalgebras.

Explicitly, a (D, ε, δ)-coalgebra is a set X with a map $\mathsf{act} : X \to S \times (P \to X)$ satisfying

$$\begin{gathered}
x = \mathsf{let}\ (s, v) \leftarrow \mathsf{act}\,x\ \mathsf{in}\ v\,\mathsf{o} \\
\mathsf{let}\ (s, v) \leftarrow \mathsf{act}\,x\ \mathsf{in}\ (s, \lambda p.\,\mathsf{act}\,(v\,p)) = \mathsf{let}\ (s, v) \leftarrow \mathsf{act}\,x\ \mathsf{in}\ (s, \lambda p.\,(s \downarrow p, \lambda p'.\,v\ (p \oplus p')))
\end{gathered}$$

i.e.,

$$\begin{tikzcd}
X \arrow[r, "\mathsf{act}"] \arrow[dr, equal] & S \times (P \to X) \arrow[d, "\varepsilon"] \\
& X
\end{tikzcd}
\qquad
\begin{tikzcd}
X \arrow[r, "\mathsf{act}"] \arrow[d, "\mathsf{act}"'] & S \times (P \to X) \arrow[d, "\delta"] \\
S \times (P \to X) \arrow[r, "S\times(P\to\mathsf{act})"] & S \times (P \to S \times (P \to X))
\end{tikzcd}$$

The isomorphism associates to an update lens structure $(\mathsf{lkp}, \mathsf{upd})$ on a set X a (D, ε, δ)-coalgebra structure $\mathsf{act} = \langle \mathsf{lkp}, \mathsf{cur}\,\mathsf{upd} \rangle$.

Following two routes (0)-(1)-(2)-(4) and (0)-(1)-(3)-(4), we have derived that update lenses for $(S, (P, \mathsf{o}, \oplus), \downarrow)$ are essentially the same as coalgebras of the comonad (D, ε, δ). The algebraic characterization (0") is analogous to that of Johnson et al. [14] for ordinary lenses for S and the coalgebraic characterization (4) is analogous to O'Connor's [18], while perhaps the most basic one is (0'), which says that an update lens is a $(P, \mathsf{o}, \oplus)$-set with a map to $(S, \downarrow)$. Many of the other alternative characterizations of update lenses are without counterparts for ordinary lenses.

We have no opportunity to discuss this here in any detail, but update lenses $(S, (P, \mathsf{o}, \oplus), \downarrow)$, i.e., coalgebras of the comonad (D, ε, δ), comodel the same (generally large) Lawvere theory that is modelled by the algebras of what we have elsewhere [3] called the update monad for $(S, (P, \mathsf{o}, \oplus), \downarrow)$. This monad (T, η, μ) is defined by $T\,X = S \to P \times X$, $\eta\,x = \lambda s.\,(\mathsf{o}, x)$, $\mu\,f = \lambda s.\,\mathsf{let}\ \{(p, g) \leftarrow f\,s; (p', x) \leftarrow g\,(s \downarrow p)\}\ \mathsf{in}\ (p \oplus p', x)$.

Update lenses: an equivalent characterization

Here is a different characterization, which is only an equivalence of categories, not an isomorphism.

We first observe that any act $(S, (P, \mathsf{o}, \oplus), \downarrow)$ defines a category $\langle\!\langle S, (P, \mathsf{o}, \oplus), \downarrow \rangle\!\rangle$ where an object is an element s of S and a map between $s, s' : S$ is an element p of P such that $s \downarrow p = s'$. The identity on s is o and the composition of p and p' is $p \oplus p'$.

The category $\mathbf{ULens}\,(S, (P, \mathsf{o}, \oplus), \downarrow)$ is equivalent to the functor category $[\langle\!\langle S, (P, \mathsf{o}, \oplus), \downarrow \rangle\!\rangle, \mathbf{Set}]$. A functor $R : \langle\!\langle S, (P, \mathsf{o}, \oplus), \downarrow \rangle\!\rangle \to \mathbf{Set}$ is mapped to the update lens $(X, \mathsf{lkp}, \mathsf{upd})$ defined by $X = \Sigma s : S.\, R\,s$, $\mathsf{lkp}\,(s, r) = s$ and $\mathsf{upd}\,((s, r), p) = (s \downarrow p, R\,p\,r)$. In the converse direction, an update lens $(X, \mathsf{lkp}, \mathsf{upd})$ is mapped to the functor R defined by $R\,s = \{x : X \mid \mathsf{lkp}\,x = s\}$, $R\,p\,x = \mathsf{upd}\,(x, p)$.

Turning ordinary lenses into update lenses

Given an act $(S, (P, \mathsf{o}, \oplus), \downarrow)$, any ordinary lens $(X, \mathsf{lkp} : X \to S, \mathsf{upd} : X \times S \to X)$ for S defines an update lens $(X, \mathsf{lkp}, \mathsf{upd}' : X \times P \to X)$ for $(S, (P, \mathsf{o}, \oplus), \downarrow)$ via $\mathsf{upd}'\,(x, p) = \mathsf{upd}\,(x, \mathsf{lkp}\,x \downarrow p)$. And any map h between two ordinary lenses for S is also a map between the corresponding update lenses for $(S, (P, \mathsf{o}, \oplus), \downarrow)$. This gives us a functor from $\mathbf{Lens}\,S$ to $\mathbf{ULens}\,(S, (P, \mathsf{o}, \oplus), \downarrow)$.

This functor is "caused" by a morphism τ between the comonads (D, ε, δ), $(D', \varepsilon', \delta')$ where $D\,X = S \times (S \to X)$ and $D'\,X = S \times (P \to X)$. It is defined by $\tau\,(s, v) = (s, \lambda p.\, v\,(s \downarrow p))$. A morphism between two comonads gives a functor between the corresponding categories of coalgebras.

4 Conversions of views and updates

So far we have seen that update lense maps $h : (X, \mathsf{lkp}, \mathsf{upd}) \to (X', \mathsf{lkp}', \mathsf{upd}')$ model the conversion from a source set X to a source set X', for a fixed act $(S, (P, \mathsf{o}, \oplus), \downarrow)$. We will now discuss how maps between acts give rise to conversions between view and update sets, for a fixed source set X.

A *map* (t, q) between acts $(S, (P, \mathsf{o}, \oplus), \downarrow)$ and $(S', (P', \mathsf{o}', \oplus'), \downarrow')$ consists of a function (conversion of views) $t : S \to S'$ and a monoid homomorphism (conversion of updates) $q : (P', \mathsf{o}', \oplus') \to (P, \mathsf{o}, \oplus)$, such that

$$t\,(s \downarrow q\,p) = t\,s \downarrow' p$$

(Notice the reversed direction of the monoid homomorphism! In the literature, e.g.,

[17], one typically requires that $q : (P, \mathsf{o}, \oplus) \to (P', \mathsf{o}', \oplus')$ and $t\,(s \downarrow p) = t\,s \downarrow' q\,p$, but this is not what we need here.) Acts form a category **Act**.

We say that a *conversion of views and updates* induced by a morphism (t, q) between acts $(S, (P, \mathsf{o}, \oplus), \downarrow)$, $(S', (P', \mathsf{o}', \oplus'), \downarrow')$ is a functor

$$\mathbf{ULens}\,(t, q) : \mathbf{ULens}\,(S, (P, \mathsf{o}, \oplus), \downarrow) \to \mathbf{ULens}\,(S', (P', \mathsf{o}', \oplus'), \downarrow')$$

defined as

$$\begin{gathered}\mathbf{ULens}\,(t, q)\,(X, \mathsf{lkp}, \mathsf{upd}) = (X, t \circ \mathsf{lkp}, \mathsf{upd} \circ (X \times q)) \\ \mathbf{ULens}\,(t, q)\,h = h\end{gathered}$$

ULens forms a functor from **Act** to **CAT**.

From the functor **ULens** we can build the total category **ULensTot** of all update lenses with the Grothendieck construction (see, e.g., [13, §1.10]), i.e., $\mathbf{ULensTot} = \int \mathbf{ULens}$.

In detail, an object of **ULensTot** is an act $(S, (P, \mathsf{o}, \oplus), \downarrow)$ together with an update lens $(X, \mathsf{lkp}, \mathsf{upd})$ for that act, i.e., an object of $\mathbf{ULens}\,(S, (P, \mathsf{o}, \oplus), \downarrow)$. A map between two objects $((S, (P, \mathsf{o}, \oplus), \downarrow), (X, \mathsf{lkp}, \mathsf{upd}))$ and $((S', (P', \mathsf{o}', \oplus'), \downarrow'), (X', \mathsf{lkp}', \mathsf{upd}'))$ is an act map $(t, q) : (S, (P, \mathsf{o}, \oplus), \downarrow) \to (S', (P', \mathsf{o}', \oplus'), \downarrow')$ paired with a map $h : \mathbf{ULens}\,(t, q)\,(X, \mathsf{lkp}, \mathsf{upd}) \to (X', \mathsf{lkp}', \mathsf{upd}')$ of update lenses for $(S', (P', \mathsf{o}', \oplus'), \downarrow')$.

Conversions of views and updates as mappings of comonad coalgebras to comonad coalgebras

Just as every act $(S, (P, \mathsf{o}, \oplus), \downarrow)$ defines a comonad $[\![S, (P, \mathsf{o}, \oplus), \downarrow]\!] = (D, \varepsilon, \delta)$, every morphism (t, q) between two acts $(S, (P, \mathsf{o}, \oplus), \downarrow)$ and $(S', (P', \mathsf{o}', \oplus'), \downarrow')$ defines a morphism $[\![t, q]\!] = \tau$ between the comonads $[\![S, (P, \mathsf{o}, \oplus), \downarrow]\!]$ and $[\![S', (P', \mathsf{o}', \oplus'), \downarrow']\!]$. Explictly, this natural transformation is defined by

$$\begin{gathered}\tau : \forall\{X\}.\,(S \times (P \to X)) \to S' \times (P' \to X) \\ \tau\,(s, v) = (t\,s, v \circ q)\end{gathered}$$

$[\![-]\!]$ forms a functor from **Act** to **Comonads**(**Set**).

A morphism τ between two comonads (D, ε, δ) and $(D', \varepsilon', \delta')$ determines a functor between the corresponding categories of coalgebras, sending a (D, ε, δ)-coalgebra (X, act) to the $(D', \varepsilon', \delta')$-coalgebra $(X, \tau\,\{X\} \circ \mathsf{act})$.

While the category $\mathbf{ULens}\,(S, (P, \mathsf{o}, \oplus), \downarrow)$ is isomorphic to the category of coalgebras of the comonad $[\![S, (P, \mathsf{o}, \oplus), \downarrow]\!]$, the functor $\mathbf{ULens}\,(t, q) : \mathbf{ULens}\,(S, (P, \mathsf{o}, \oplus), \downarrow) \to \mathbf{ULens}\,(S', (P', \mathsf{o}', \oplus'), \downarrow')$ is isomorphic to the functor between the categories of coalgebras of $[\![S, (P, \mathsf{o}, \oplus), \downarrow]\!]$ and $[\![S', (P', \mathsf{o}', \oplus'), \downarrow']\!]$, induced by the comonad morphism $[\![t, q]\!]$. In summary, the following diagram commutes up to isomorphism

$$\begin{tikzcd}
\mathbf{Act} \arrow[rr, "\llbracket - \rrbracket"] \arrow[dr, "\mathbf{ULens}"'] & & \mathbf{Comonads}(\mathbf{Set}) \arrow[dl, "\mathbf{Coalg}"] \\
& \mathbf{CAT} &
\end{tikzcd}$$

5 Tensoring update lenses

Sometimes we might want to collect multiple update lenses into a single update lens structure. We show how this can be done with a tensor product on the category of all update lenses.

We define the unit act I to be the act $(1, (1, !, !), !)$.

For any two acts $(S_0, (P_0, \mathsf{o}_0, \oplus_0), \downarrow_0)$ and $(S_1, (P_1, \mathsf{o}_1, \oplus_1), \downarrow_1)$, we define their tensor $(S_0, (P_1, \mathsf{o}_1, \oplus_1), \downarrow_1) \otimes (S_1, (P_1, \mathsf{o}_1, \oplus_1), \downarrow_1)$ to be $(S_0 \times S_1, (P_0 \times P_1, \mathsf{o}, \oplus), \downarrow)$ where

$$\mathsf{o} = (\mathsf{o}_0, \mathsf{o}_1) \qquad (p_0, p_1) \oplus (p'_0, p'_1) = (p_0 \oplus_0 p'_0, p_1 \oplus_1 p'_1)$$
$$(s_0, s_1) \downarrow (p_0, p_1) = (s_0 \downarrow_0 p_0, s_1 \downarrow_1 p_1)$$

I and $\otimes$ make **Act** a monoidal category.

Now for the act I we have a unit lens $J = (1, !, !)$.

And for two update lenses $(X_0, \mathsf{lkp}_0, \mathsf{upd}_0)$ and $(X_1, \mathsf{lkp}_1, \mathsf{upd}_1)$ for acts $(S_0, (P_0, \mathsf{o}_0, \oplus_0), \downarrow_0)$ resp. $(S_1, (P_1, \mathsf{o}_1, \oplus_1), \downarrow_1)$, there is the tensor update lens $(X_0, \mathsf{lkp}_0, \mathsf{upd}_0) \boxtimes (X_1, \mathsf{lkp}_1, \mathsf{upd}_1) = (X_0 \times X_1, \mathsf{lkp}, \mathsf{upd})$ for the act $(S_0, (P_0, \mathsf{o}_0, \oplus_0), \downarrow_0) \otimes (S_1, (P_1, \mathsf{o}_1, \oplus_1), \downarrow_1)$ where

$$\mathsf{lkp}\,(x_0, x_1) = (\mathsf{lkp}_0\, x_0, \mathsf{lkp}_1\, x_1) \qquad \mathsf{upd}\,(x_0, x_1)\,(p_0, p_1) = (\mathsf{upd}_0\,(x_0, p_0), \mathsf{upd}_1\,(x_1, p_1))$$

With this we have shown that **ULens** is a lax monoidal functor from **Act** to **CAT** (wrt. the $(I, \otimes)$ monoidal structure on **Act** and the product monoidal structure on **CAT**), where the monoidality witnesses are

$$J : 1 \to \mathbf{ULens}\, I$$
$$\boxtimes : \mathbf{ULens}\,(S_0, (P_0, \mathsf{o}_0, \oplus_0), \downarrow_0) \times \mathbf{ULens}\,(S_1, (P_1, \mathsf{o}_1, \oplus_1), \downarrow_1)$$
$$\to \mathbf{ULens}((S_0, (P_0, \mathsf{o}_0, \oplus_0), \downarrow_0) \otimes (S_1, (P_1, \mathsf{o}_1, \oplus_1), \downarrow_1))$$

It is evident that $(I, \otimes)$ and $(J, \boxtimes)$ also endow the total category **ULensTot** with a monoidal structure. The unit is (I, J) and the tensor of $((S_0, (P_0, \mathsf{o}_0, \oplus_0), \downarrow_0), (X_0, \mathsf{lkp}_0, \mathsf{upd}_0))$ and $((S_1, (P_1, \mathsf{o}_1, \oplus_1), \downarrow_1), (X_1, \mathsf{lkp}_1, \mathsf{upd}_1))$ is $((S_0, (P_0, \mathsf{o}_0, \oplus_0), \downarrow_0) \otimes (S_1, (P_1, \mathsf{o}_1, \oplus_1), \downarrow_1)), (X_0, \mathsf{lkp}_0, \mathsf{upd}_0) \boxtimes (X_1, \mathsf{lkp}_1, \mathsf{upd}_1))$.

The tensor product on **ULensTot** is a "parallel composition" of lenses.

The category of acts also carries a product monoidal structure. However, due to its involved definition and lack of space, we refrain from discussing it here. For a dependently typed version of acts discussed in Section 7, we worked the details out in the journal version of [2].

6 Composition of update lenses

Above we have seen how to convert of sources (for a fixed act) and views/updates (for a fixed set of sources). We will now discuss a notion of composition of two update lenses where the view set of one update lens is the source set of another.

We develop the definition in two steps, based on the alternative characterization (0') of update lenses from Section 3.

We first recall from (0') that an update lens $(X, \mathsf{lkp} : X \to S, \mathsf{upd} : X \times P \to X)$ for an act $(S, (P, \mathsf{o}, \oplus), \downarrow)$ is essentially the same as an object $((X, \mathsf{upd}), \mathsf{lkp} : (X, \mathsf{upd}) \to (S, \downarrow))$ of $(P, \mathsf{o}, \oplus)$-$\mathsf{Set}/(S, \downarrow)$. If we keep $(P, \mathsf{o}, \oplus)$ fixed, but let $(S, \downarrow)$ vary, we can say that an update lens for $(P, \mathsf{o}, \oplus)$ is a $(P, \mathsf{o}, \oplus)$-set map $\mathsf{lkp} : (X, \mathsf{upd}) \to (Y, \mathsf{upd}')$. The identity update lens on (X, upd) is then $\mathsf{id}\,\{X\}$ and the composition of two update lenses $\mathsf{lkp} : (X, \mathsf{upd}) \to (Y, \mathsf{upd}')$ and $\mathsf{lkp}' : (Y, \mathsf{upd}') \to (Z, \mathsf{upd}'')$ is $\mathsf{lkp}' \circ \mathsf{lkp}$.

To be able to compose update lenses for possibly different monoids, we need to let $(P, \mathsf{o}, \oplus)$ vary too. We achieve this by switching to a more involved category.

An object is just a set. A map between two sets X, Y is given by a monoid $(P, \mathsf{o}, \oplus)$, actions upd and $\downarrow$ on X resp. Y (up to isomorphism in the choice of these data) and a map $\mathsf{lkp} : X \to Y$ satisfying $\mathsf{lkp}\,(\mathsf{upd}\,(x, p)) = \mathsf{lkp}\,x \downarrow p$.

The identity map on X is $(P, \mathsf{o}, \oplus, \mathsf{upd}, \downarrow, \mathsf{lkp})$ where $P = X \to X$, $\mathsf{o} = \mathsf{id}$, $f \oplus g = g \circ f$, $\mathsf{upd}\,(x, f) = x \downarrow f = f\,x$, $\mathsf{lkp} = \mathsf{id}$. The composition of two maps $(P_0, \mathsf{o}_0, \oplus_0, \mathsf{upd}_0, \downarrow_0, \mathsf{lkp}_0) : X \to Y$ and $(P_1, \mathsf{o}_1, \oplus_1, \mathsf{upd}_1, \downarrow_1, \mathsf{lkp}_1) : Y \to Z$ is $(P, \mathsf{o}, \oplus, \mathsf{upd}, \downarrow, \mathsf{lkp})$ where $P = \{(p_0, p_1) : P_0 \times P_1 \mid \forall y : Y.\, y \downarrow_0 p_0 = \mathsf{upd}_1\,(y, p_1)\}$, $\mathsf{o} = (\mathsf{o}_0, \mathsf{o}_1)$, $(p_0, p_1) \oplus (p'_0, p'_1) = (p_0 \oplus_0 p'_0, p_1 \oplus_1 p'_1)$, $\mathsf{upd}\,(x, (p_0, p_1)) = \mathsf{upd}_0\,(x, p_0)$, $z \downarrow (p_0, p_1) = z \downarrow_1 p_1$, $\mathsf{lkp} = \mathsf{lkp}_1 \circ \mathsf{lkp}_0$.

7 A dependently typed generalization

It often happens that every particular view comes with a specific set of safe or allowed updates. For example, if a view is a table, then one expects to be able to edit and delete only those rows appearing in the view, and not to be able to edit rows after they have been deleted. As we saw in Section 3, update lenses do not capture such intended behavior, albeit it can be accommodated by defining suitably "truncating" $\downarrow$ and upd. The reason for this is that there is a single monoid of updates that has to act on every view.

We propose a fix to these limitations of update lenses by introducing a dependently typed version. This development is based on our previous work with Chapman [2] on directed containers, a specialization of Abbott, Altenkirch and Ghani's containers [1]. Due to lack of space, we must keep the presentation very brief and refer the interested reader to our earlier work [2,3].

Containers, directed containers

Recall that a *container* is given by a set S and a S-indexed family P of sets. A *map* between two containers (S, P) and (S', P') is given by functions $t : S \to S'$ and

$q : \Pi\{s : S\}.\, P'\,(t\,s) \to P\,s$. Containers and maps between them form a category **Cont**. Any container (S, P) gives rise to a set functor $[\![S, P]\!]^{\mathrm{c}}$ defined by

$$[\![S, P]\!]^{\mathrm{c}} X = \Sigma s : S.P\,s \to X$$
$$[\![S, P]\!]^{\mathrm{c}} h = \lambda(s, v).(s, h \circ v)$$

Any container map (t, q) defines a natural transformation $[\![t, q]\!]^{\mathrm{c}} : [\![S, P]\!]^{\mathrm{c}} \to [\![S', P']\!]^{\mathrm{c}}$ by $[\![t, q]\!]^{\mathrm{c}} = \lambda(s, v).\,(t\,s, v \circ q)$. $[\![-]\!]^{\mathrm{c}}$ is a functor from **Cont** to $[\mathbf{Set}, \mathbf{Set}]$.

A *directed container* $(S, P, \downarrow, \mathsf{o}, \oplus)$ is given by a container (S, P) and operations

$$\downarrow : \Pi s : S.\, P\,s \to S$$
$$\mathsf{o} : \Pi\{s : S\}.\, P\,s$$
$$\oplus : \Pi\{s : S\}.\, \Pi p : P\,s.\, P\,(s \downarrow p) \to P\,s$$

satisfying five conditions

$$s \downarrow \mathsf{o} = s \qquad s \downarrow (p \oplus p') = (s \downarrow p) \downarrow p'$$
$$p \oplus \mathsf{o} = p \qquad \mathsf{o} \oplus p = p \qquad (p \oplus p') \oplus p'' = p \oplus (p' \oplus p'')$$

Directed containers are a dependently typed generalization of acts. Here we do not have a single monoid and not a family of monoids either, but rather a single monoid-like structure spread out over multiple carriers $P\,s$. If one ignores the dependent typing of the above five equations, they are exactly the equations of a monoid and its action on S.

A *map* between directed containers $(S, P, \downarrow, \mathsf{o}, \oplus)$ and $(S', P', \downarrow', \mathsf{o}', \oplus')$ is a map (t, q) between the underlying containers (S, P) and (S', P') that satisfies

$$t\,(s \downarrow q\,p) = t\,s \downarrow' p \qquad \mathsf{o} = q\,\mathsf{o}' \qquad q\,p \oplus q\,p' = q\,(p \oplus' p')$$

Of course, a map between directed containers is a dependently typed generalization of a map between acts. Directed containers form a category **DCont**.

A directed container $(S, P, \downarrow, \mathsf{o}, \oplus)$ determines a comonad $[\![S, P, \downarrow, \mathsf{o}, \oplus]\!]^{\mathrm{dc}} = (D, \varepsilon, \delta)$ given by

$$D\,X = [\![S, P]\!]^{\mathrm{c}}\,X = \Sigma s : S.\, P\,s \to X$$
$$\varepsilon\,(s, v) = v\,\mathsf{o}$$
$$\delta\,(s, v) = (s, \lambda p.\,(s \downarrow p, \lambda p'.\,v\,(p \oplus p')))$$

For a map (t, q) between two directed containers $(S, P, \downarrow, \mathsf{o}, \oplus)$ and $(S', P', \downarrow', \mathsf{o}', \oplus')$, the natural transformation $[\![t, q]\!]^{\mathrm{c}}$ is a comonad map between $[\![S, P, \downarrow, \mathsf{o}, \oplus]\!]^{\mathrm{dc}}$ and $[\![S', P', \downarrow', \mathsf{o}', \oplus']\!]^{\mathrm{dc}}$. So the functor $[\![-]\!]^{\mathrm{c}} : \mathbf{Cont} \to [\mathbf{Set}, \mathbf{Set}]$ lifts to a functor $[\![-]\!]^{\mathrm{dc}} : \mathbf{DCont} \to \mathbf{Comonads}(\mathbf{Set})$.

In fact, $[\![-]\!]^{\mathrm{dc}}$ is the pullback in **CAT** of $[\![-]\!]^{\mathrm{c}}$ along $U : \mathbf{Comonads}(\mathbf{Set}) \to [\mathbf{Set}, \mathbf{Set}]$.

Dependently typed update lenses

A *dependently typed update lens* (X, act) for a directed container $(S, P, \downarrow, \mathsf{o}, \oplus)$ is a set X (of sources) and a map $\mathsf{act} : X \to \Sigma s : S.\, P\, s \to X$ satisfying

$$x = \mathsf{let}\ (s, v) \leftarrow \mathsf{act}\, x\ \mathsf{in}\ v\, \mathsf{o}$$
$$\mathsf{let}\ (s, v) \leftarrow \mathsf{act}\, x\ \mathsf{in}\ (s, \lambda p.\, \mathsf{act}\, (v\, p)) = \mathsf{let}\ (s, v) \leftarrow \mathsf{act}\, x\ \mathsf{in}\ (s, \lambda p.\, (s \downarrow p, \lambda p'.\, v\ (p \oplus p')))$$

A *map* between dependently typed update lenses (X, act) and (X', act') is a map $h : X \to X'$ (conversion of a source) satisfying

$$\mathsf{act}'\, (h\, x) = \mathsf{let}\ (s, v) \leftarrow \mathsf{act}\, x\ \mathsf{in}\ (s, h \circ v)$$

The category of dependently typed update lenses is precisely the category of coalgebras of the comonad (D, ε, δ) defined above.

While simply-typed update lenses fail to subsume ordinary lenses (since an arbitrary unstructured set S does not carry monoid structure), an ordinary lens $(X, \mathsf{lkp}, \mathsf{upd})$ for a set S is a dependently typed updated monad for $(S, P, \downarrow, \mathsf{o}, \oplus)$ given by $P\, s = S$, $s \downarrow s' = s'$, $\mathsf{o}\, \{s\} = s$, $s' \oplus \{s\}\, s'' = s''$.

Running example: editing a bookshop database

We briefly revisit our bookshop example in the context of dependently typed update lenses. We define a dependently typed update lens structure for editing book prices, in a type-safe way.

We keep the set of sources X and the set of views S as before. We define the S-indexed family P of updates inductively as heterogeneous lists by the two rules

$$\frac{}{[] : P\, (B, v)} \qquad \frac{v\, b + c \geq 0 \quad ps : P\, (B, \lambda b' \in B.\, \mathsf{if}\ b = b'\ \mathsf{then}\ v'\, b + c\ \mathsf{else}\ v'\, b)}{(b, c) :: ps : P\, (B, v)}$$

$P\, (B, v)$ encodes sequences of single book price changes whose application is guaranteed to lead to no negative price in the database, if all prices in the given view (B, v) of the database are nonnegative.

The monoid structure $(\mathsf{o}, \oplus)$ is again that of nil and append, but the dependently typed $\downarrow$ and act are defined as below.

$$(B, v) \downarrow [] = (B, v)$$
$$(B, v) \downarrow ((b, c) :: ps) = (B, \lambda b' \in B.\, \mathsf{if}\ b = b'\ \mathsf{then}\ v\, b' + c\ \mathsf{else}\ v\, b') \downarrow ps$$

$$\mathsf{act}\, (B, v) = ((B, \mathsf{fst} \circ v), \lambda ps.\, \mathsf{act}'\, (B, v)\, ps)$$

where

$$\mathsf{act}' : \Pi(B, v) : X.\, P\, (B, \mathsf{fst} \circ v) \to X$$
$$\mathsf{act}'\, ((B, v), []) = (B, v)$$
$$\mathsf{act}'\, ((B, v), (b, c) :: ps) =$$
$$\mathsf{act}'\, ((B, \lambda b'.\, \mathsf{if}\ b' = b\ \mathsf{then}\ (\mathsf{fst}\, (v\, b') + c, \mathsf{snd}\, (v\, b'))\ \mathsf{else}\ v\, b'), ps)$$

8 Conclusions and future work

Combining insights from our work on update monads [3] with existing knowledge about ordinary (asymmetric) lenses [18,14,10], we were led to a concept of update lenses.

Update lenses are a refinement of ordinary lenses that we find both practically meaningful and theoretically elegant. Most interestingly perhaps, update lenses admit decompositions that ordinary lenses do not enjoy: they can be seen as pairs of matching coalgebras, bialgebras etc. These characterizations arise naturally from various kinds of distributive laws.

We wish to continue this work by turning to symmetric variations of update lenses. We expect to be able to build on both the original symmetric lenses works [12,11] and the newest ideas of Johnson and Rosebrugh [16]. We also plan to find out the precise connections to delta lenses and c-lenses [15].

Acknowledgements

Tarmo Uustalu thanks Andreas Abel for reminding him of O'Connor's work. This ongoing work is being supported by the University of Edinburgh Principal's Career Development PhD Scholarship, the ERDF funded Estonian CoE project EXCS and ICT National Programme project Coinduction, and the Estonian Research Council target-financed research theme 0140007s12 and grant no. 9475.

References

[1] Abbott, M., T. Altenkirch, N. Ghani. *Containers: constructing strictly positive types,* Theor. Comput. Sci. **342**(1), 2005, pp. 3–27.

[2] Ahman, D., J. Chapman, and T. Uustalu, *When is a container a comonad?* Log. Methods in Comput. Sci., to appear. Conference version in L. Birkedal, ed., "Proc. of 15th Int. Conf. on Foundations of Software Science and Computation Structures, FoSSaCS 2012 (Tallinn, March 2012)," Lect. Notes in Comput. Sci. **7213**, Springer, 2012, pp. 74–88.

[3] Ahman, D. and T. Uustalu, *Update monads: cointerpreting directed containers,* in: R. Matthes and A. Schubert, eds., "Proc. of 19th Conf. on Types for Proofs and Programs, TYPES '13 (Toulouse, Apr. 2013)," Leibniz Proc. in Informatics, Schloss Dagstuhl, to appear.

[4] Barr, M. and C. Wells, *Toposes, Triples and Theories,* Grundlehren der mathematischen Wissenschaften **278**, Springer, 1984.

[5] Beck, J., *Distributive laws,* in: B. Eckmann, ed., "Seminar on Triples and Categorical Homology, ETH 1966/67," Lect. Notes in Math. **80**, Springer, 1969, pp. 119–140.

[6] Diskin, Z., Y. Xiong, and K. Czarnecki, *From state- to delta-based bidirectional model transformations: the asymmetric case,* J. of Object Technology **10**, 2011, article 6.

[7] Eilenberg, S. and J. Moore, *Adjoint functors and triples,* Illinois J. of Math. **9**(3), 1965, pp 381–398.

[8] Foster, J. N., M. B. Greenwald, J. T. Moore, B. C. Pierce, and A. Schmitt, *Combinators for bidirectional tree transformations: a linguistic approach to the view-update problem,* ACM Trans. on Program. Lang. and Syst. **29**(3), 2007, article 17.

[9] Foster, J. N., A. Pilkiewicz, and B. C. Pierce, *Quotient lenses,* in: "Proc. of 13th ACM SIGPLAN Int. Conf. on Functional Programming, ICFP '08 (Victoria, BC, Sept. 2008)," ACM, 2008, pp. 383–396.

[10] Gibbons, J. and M. Johnson, *Relating algebraic and coalgebraic descriptions of lenses,* in: F. Hermann and J. Voigtländer, eds., "Proc. of 1st Int. Wksh. on Bidirectional Transformations, BX 2012 (Tallinn, March 2012)," Electron. Commun. of EASST **49**, 2012, 16 pp.

[11] Hofmann, M., B. C. Pierce, and D. Wagner, *Edit lenses,* in: "Proc. of 39th Ann. ACM SIGPLAN-SIGACT Symp. on Principles of Programming Languages, POPL '12 (Philadelphia, PA, January 2012)," ACM, 2012, pp. 495–508.

[12] Hofmann, M., B. C. Pierce, and D. Wagner, *Symmetric lenses,* in: "Proc. of 38th Ann. ACM SIGPLAN-SIGACT Symp. on Principles of Programming Languages, POPL '11 (Austin, TX, Jan. 2011)," ACM, 2011, pp. 371–384.

[13] Jacobs, B., *Categorical Logic and Type Theory,* Studies in Logic and the Foundations of Mathematics **141**, North Holland, 1999.

[14] Johnson, M., R. Rosebrugh, and R. J. Wood, *Algebras and update strategies,* J. of Univ. Comput. Sci **16**(5), 2010, pp. 729–748.

[15] Johnson, M. and R. Rosebrugh, *Delta lenses and opfibrations,* in: P. Stevens and J. F. Terwilliger, "Proc. of 2nd Int. Wksh. on Bidirectional Transformations, BX 2013 (Rome, March 2013)," Electron. Commun. of EASST **57**, 2013, 18 pp.

[16] Johnson, M. and R. Rosebrugh, *Spans of lenses,* in: K. Selcuk-Candan et al., eds., "Proc. of Wkshs. of EDBT/ICDT 2014 Joint Conf. (Athens, March 2014)," *CEUR Wksh. Proc.* **1133**, RWTH Aachen, 2014, pp. 112–118.

[17] Kilp, M., U. Knauer, A. V. Mikhalev, *Monoids, Acts and Categories: With Applications to Wreath Products and Graphs,* De Gruyter Expositions in Mathematics **29**, De Gruyter, 2000.

[18] O'Connor, R., *Functor is to lens as applicative is to biplate: introducing multiplate,* arXiv:1103.2841, 2011. (Paper presented at 2011 ACM SIGPLAN Wksh. on Generic Programming, WGP '11, Tokyo, Sept. 2011.)

[19] Power, J. and O. Shkaravska, *From comodels to coalgebras: state and arrays,* in: J. Adámek and S. Milius, eds., "Proc. of 7th Int. Wksh. on Coalgebraic Methods in Computer Science, CMCS '04 (Barcelona, March 2004)," Electr. Notes in Theor. Comput. Sci. **106**, Elsevier, 2004, pp. 297–314.

[20] Power, J. and H. Watanabe, *Combining a monad and a comonad,* Theor. Comput. Sci. **280**(1-2), 2002, pp. 137–162.

[21] Tanaka, M., *Pseudo-distributive laws and unified framework for variable binding,* PhD thesis, LFCS, Univ. of Edinburgh, 2005.

A Initializable ordinary and update lenses

In this section we discuss a specialization that is applicable to both ordinary and update lenses, initializability.

A.1 Initializable ordinary lenses

In the lenses literature [9,12], one sometimes equips an ordinary lens $(X, \mathsf{lkp}, \mathsf{upd})$ for a set S with an additional map $\mathsf{create} : S \to X$ (creation of a source from a view) satisfying

$$\mathsf{lkp}\,(\mathsf{create}\,s) = s \qquad \mathsf{upd}\,(\mathsf{create}\,s, s') = \mathsf{create}\,s'$$

i.e.,

$$\begin{tikzcd}
S \arrow[r, "\mathsf{create}"] \arrow[dr, equal] & X \arrow[d, "\mathsf{lkp}"] \\
& S
\end{tikzcd}
\qquad
\begin{tikzcd}
S \times S \arrow[r, "\mathsf{create} \times S"] \arrow[d, "\mathsf{snd}"'] & X \times S \arrow[d, "\mathsf{upd}"] \\
S \arrow[r, "\mathsf{create}"] & X
\end{tikzcd}$$

We call such a structure an *initializable lens.*

A *map* between two initializable lenses $(X, \mathsf{lkp}, \mathsf{upd}, \mathsf{create})$ and $(X', \mathsf{lkp}', \mathsf{upd}', \mathsf{create}')$ is a map h between $(X, \mathsf{lkp}, \mathsf{upd})$ and $(X', \mathsf{lkp}', \mathsf{upd}')$ that also satisfies

$$h\,(\mathsf{create}\,s) = \mathsf{create}'\,s$$

i.e.,

$$\begin{tikzcd}
S \arrow[r, equal] \arrow[d, "\mathsf{create}"'] & S \arrow[d, "\mathsf{create}'"] \\
X \arrow[r, "h"] & X'
\end{tikzcd}$$

Of course initializable lenses for S and maps between them form a category.

A.2 Initializable update lenses

Update lenses admit this specialization too.

Initializable update lenses: the definition

We define an *initializable update lens* $(X, \mathsf{lkp}, \mathsf{upd}, \mathsf{create})$ for an act $(S, (P, \mathsf{o}, \oplus), \downarrow)$ to be given by an update lens $(X, \mathsf{lkp}, \mathsf{upd})$ together with an additional map $\mathsf{create} : S \to X$ satisfying

$$\mathsf{lkp}\,(\mathsf{create}\,s) = s \qquad \mathsf{upd}\,(\mathsf{create}\,s, p) = \mathsf{create}\,(s \downarrow p)$$

i.e.,

$$\begin{tikzcd}
S \arrow[r, "\mathsf{create}"] \arrow[dr, equal] & X \arrow[d, "\mathsf{lkp}"] & S \times P \arrow[r, "\mathsf{create} \times P"] \arrow[d, "\downarrow"'] & X \times P \arrow[d, "\mathsf{upd}"] \\
& S & S \arrow[r, "\mathsf{create}"] & X
\end{tikzcd}$$

A *map* between two initializable update lenses $(X, \mathsf{lkp}, \mathsf{upd}, \mathsf{create})$ and $(X', \mathsf{lkp}', \mathsf{upd}', \mathsf{create}')$ is a map h between $(X, \mathsf{lkp}, \mathsf{upd})$ and $(X', \mathsf{lkp}', \mathsf{upd}')$ that also satisfies

$$h\,(\mathsf{create}\,s) = \mathsf{create}'\,s$$

i.e.,

$$\begin{tikzcd}
S \arrow[r, equal] \arrow[d, "\mathsf{create}"'] & S \arrow[d, "\mathsf{create}'"] \\
X \arrow[r, "h"] & X'
\end{tikzcd}$$

Initializable update lenses for $(S, (P, \mathsf{o}, \oplus), \downarrow)$ and maps between them form a category.

Initializable update lenses: alternative descriptions

Recall the definitions of F_0, (T_1, η_1, μ_1) and (D, ε, δ) from Section 3.

It is immediate from the definition that the category of initializable update lenses is the same as category of triples of a coalgebra of F_0, algebra of (T_1, η_1, μ_1) and algebra of F_0 on the same carrier pairwise matched by the distributive law $\downarrow$ of (T_1, η_1, μ_1) over F_0 (used twice) and the distributive law id of F_0 over itself, i.e., of quadruples of a set X, map $\mathsf{lkp} : X \to S$, (T_1, η_1, μ_1)-algebra structure $\mathsf{upd} : X \times P \to X$ and map $\mathsf{create} : S \to X$ satisfying

$$\begin{tikzcd}
X \times P \arrow[r, "\mathsf{upd}"] \arrow[d, "\mathsf{lkp} \times P"] & X \arrow[r, "\mathsf{lkp}"] & S \arrow[d, equal] & S \arrow[r, "\mathsf{create}"] \arrow[d, equal] & X \arrow[r, "\mathsf{lkp}"] & S \arrow[d, equal] & X \times P \arrow[r, "\mathsf{upd}"] & X & S \arrow[l, "\mathsf{create}"'] \arrow[d, equal] \\
S \times P \arrow[rr, "\downarrow"] & & S & S \arrow[rr, equal] & & S & S \times P \arrow[u, "\mathsf{create} \times P"] \arrow[rr, "\downarrow"] & & S
\end{tikzcd}$$

(remember that a (T_1, η_1, μ_1)-algebra structure on X is an action of $(P, \mathsf{o}, \oplus)$ on X).

We already know that the category of $\downarrow$-matching bialgebras of F_0 and (T_1, η_1, μ_1) is isomorphic to the category of coalgebras of (D, ε, δ). It is also the case that the distributive laws id of F_0 over itself and $\downarrow$ of (T_1, η_1, μ_1) over F_0 make $[\mathsf{id}, \mathsf{cur} \downarrow]$ a distributive law of F_0 over (D, ε, δ). It follows that the category of initializable update lenses is also isomorphic to the category of bialgebras of (D, ε, δ) and F_0 matched by the distributive law $\langle \mathsf{id}, \mathsf{cur} \downarrow \rangle$ of F_0 over (D, ε, δ), i.e., of triples of a set X, (T_1, η_1, μ_1)-coalgebra structure $\mathsf{act} : X \to S \times (P \to X)$ and map $\mathsf{create} : S \to X$ satisfying

$$\begin{tikzcd}
S \arrow[r, "\mathsf{create}"] \arrow[d, equal] & X \arrow[r, "\mathsf{act}", "= \langle \mathsf{lkp}, \mathsf{cur}\, \mathsf{upd} \rangle"'] & S \times (P \to X) \\
S \arrow[rr, "\langle \mathsf{id}, \mathsf{cur} \downarrow \rangle"] & & S \times (P \to S) \arrow[u, "S \times P \to \mathsf{create}"']
\end{tikzcd}$$

But notably we can also pack together upd and create instead of upd and lkp. This is done in two steps.

First we observe that the category of algebras of F_0 is isomorphic to the category of algebras of the free monad (T_0, η_0, μ_0) on F_0, explicitly defined by

$$\begin{gathered}
T_0\, X = S + X \\
\eta_0\, x = \mathsf{inr}\, x \\
\mu_0\, (\mathsf{inl}\, s) = \mathsf{inl}\, s \\
\mu_0\, (\mathsf{inl}\, (\mathsf{inl}\, s)) = \mathsf{inl}\, s \\
\mu_0\, (\mathsf{inl}\, (\mathsf{inr}\, x)) = \mathsf{inr}\, x
\end{gathered}$$

The isomorphism assigns to a F_0-algebra (X, create) the (T_0, η_0, μ_0)-algebra $(X, \mathsf{dcreate})$ where $\mathsf{dcreate} = [\mathsf{create}, \mathsf{id}]$.

The distributive laws id of F_0 over itself and $\downarrow$ of (T_1, η_1, μ_1) over F_0 induce distributive laws $[\mathsf{id}, \mathsf{id}]$ of (T_0, η_0, μ_0) over F_0 and ϕ of (T_1, η_1, μ_1) over (T_0, η_0, μ_0)

where

$$\begin{aligned} \psi : (S + X) \times P &\to S + X \times P \\ \psi\,(\mathsf{inl}\,s, p) &= \mathsf{inl}\,(s \downarrow p) \\ \psi\,(\mathsf{inr}\,x, p) &= \mathsf{inr}\,(x, p) \end{aligned}$$

The category of initializable update lenses is therefore isomorphic to the category of triples of a coalgebra of F_0, algebra of (T_1, η_1, μ_1) and algebra of (T_0, η_0, μ_0) on a common carrier pairwise matched by the distributive laws $\downarrow$, $[\mathsf{id}, \mathsf{id}]$ and ψ, i.e., quadruples of a set X, map $\mathsf{lkp} : X \to S$, (T_1, η_1, μ_1)-algebra structure $\mathsf{upd} : X \times P \to X$ and (T_0, η_0, μ_0)-algebra structure $\mathsf{dcreate} : S + X \to X$ satisfying

$$\begin{tikzcd} X \times P \arrow[r, "\mathsf{upd}"] \arrow[d, "\mathsf{lkp} \times P"] & X \arrow[r, "\mathsf{lkp}"] & S \arrow[d, equal] \\ S \times P \arrow[rr, "\downarrow"] & & S \end{tikzcd}
\qquad
\begin{tikzcd} S + X \arrow[r, "\mathsf{dcreate}", "{=[\mathsf{create},\mathsf{id}]}"'] \arrow[d, "S + \mathsf{lkp}"] & X \arrow[r, "\mathsf{lkp}"] & S \arrow[d, equal] \\ S + S \arrow[rr, "{[\mathsf{id},\mathsf{id}]}"] & & S \end{tikzcd}
\qquad
\begin{tikzcd} X \times P \arrow[r, "\mathsf{upd}"] & X & S + X \arrow[l, "\mathsf{dcreate}"'] \\ (S + X) \times P \arrow[u, "\mathsf{dcreate} \times P"] \arrow[rr, "\psi"] & & S + X \times P \arrow[u, "S + \mathsf{upd}"'] \end{tikzcd}$$

Second we notice that the distributive law ψ of (T_1, η_1, μ_1) over (T_0, η_0, μ_0) defines a compatible composition (T, η, μ) of the two monads, explicitly

$$\begin{aligned} TX &= S + X \times P \\ \eta\, x &= \mathsf{inr}\,(x, \mathsf{o}) \\ \mu\,(\mathsf{inl}\,s) &= \mathsf{inl}\,s \\ \mu\,(\mathsf{inr}\,(\mathsf{inl}\,s, p)) &= \mathsf{inl}\,(s \downarrow p) \\ \mu\,(\mathsf{inr}\,(\mathsf{inr}(x, p), p') &= \mathsf{inr}\,(x, p \oplus p') \end{aligned}$$

The category of algebras of (T, η, μ) is isomorphic to the category of ψ-matching pairs of algebras of (T_0, η_0, μ_0) and (T_1, η_1, μ_1). The algebra of (T, η, μ) corresponding to a ψ-matching pair $(X, \mathsf{create}, \mathsf{upd})$ of algebras of (T_0, η_0, μ_0) and (T_1, η_1, μ_1) is $(X, \mathsf{maintain})$ where $\mathsf{maintain} = [\mathsf{create}, \mathsf{upd}]$.

The distributive laws $\downarrow$ of (T_1, η_1, μ_1) over F_0 and $[\mathsf{id}, \mathsf{id}]$ of (T_0, η_0, μ_0) over F_0 determine a distributive law $[\mathsf{id}, \downarrow]$ of (T, η, μ) over F_0. As a result, the category of initializable update lenses is isomorphic to the category of bialgebras of F_0 and (T, η, μ) for the distributive law $[\mathsf{id}, \downarrow]$, i.e., triples of a set X, map $\mathsf{lkp} : X \to S$ and (T, η, μ)-algebra structure $\mathsf{maintain} : S + X \times P \to X$ satisfying

$$\begin{tikzcd} S + X \times P \arrow[r, "\mathsf{maintain}", "{=[\mathsf{create},\mathsf{upd}]}"'] \arrow[d, "S + \mathsf{lkp} \times P"'] & X \arrow[r, "\mathsf{lkp}"] & S \arrow[d, equal] \\ S + S \times P \arrow[rr, "{[\mathsf{id},\downarrow]}"'] & & S \end{tikzcd}$$

We see that we can construct initializable update lenses by first giving ourselves the means to initialize and update the source, and only then to view the source.

Applicative Bisimilarities for Call-by-Name and Call-by-Value $\lambda\mu$-Calculus

Dariusz Biernacki [1]

Institute of Computer Science
University of Wrocław
Wrocław, Poland

Sergueï Lenglet [2]

LORIA
Université de Lorraine
Nancy, France

Abstract

We propose the first sound and complete bisimilarities for the call-by-name and call-by-value untyped $\lambda\mu$-calculus, defined in the applicative style. We give equivalence examples to illustrate how our relations can be used; in particular, we prove David and Py's counter-example, which cannot be proved with Lassen's preexisting normal form bisimilarities for the $\lambda\mu$-calculus.

Keywords: contextual equivalence, applicative bisimulation, continuation, control operator

1 Introduction

Contextual equivalence [13] is considered as the most natural behavioral equivalence in languages based on the λ-calculus. Two terms are equivalent if an outside observer cannot tell them apart when they are evaluated within any *context* (a term with a hole). However, the quantification over contexts makes proving the equivalence of two given programs cumbersome. Consequently, other characterizations of contextual equivalence are sought for, such as coinductively defined *bisimilarities*.

Several kinds of bisimilarity have been proposed, such as, e.g., *applicative* bisimilarity [1], which relates terms by reducing them to values (if possible), and then compares these values by applying them to an arbitrary argument. The idea is the

[1] Email: dabi@cs.uni.wroc.pl

[2] Email: serguei.lenglet@univ-lorraine.fr

same for *environmental* bisimilarity [18], except the values are tested with arguments built from an environment, which represents the knowledge of an observer about the tested terms. Finally, *normal form* bisimilarity [11] (initially called *open* bisimilarity [17]) reduces open terms to normal forms and then compares their subterms. Applicative and environmental bisimilarities still contain some quantification over arguments, and usually coincide with contextual equivalence. In contrast, normal form bisimilarity is easier to use, as its definition does not contain any quantification over arguments, but it is generally not *complete*, i.e., there exist equivalent terms that are not normal form bisimilar.

This article treats the behavioral theory of the untyped $\lambda\mu$-calculus [15]. The $\lambda\mu$-calculus provides a computational interpretation of classical natural deduction and thus extends the Curry-Howard correspondence from intuitionistic to classical logic. Operationally, the reduction rules of the calculus express not only function applications but also capturing of the current context of evaluation. Therefore, when considered in the untyped setting, the calculus offers an approach to the semantics of abortive control operators such as *call/cc* known from the Scheme programming language and it may be viewed as a closely related alternative to Felleisen and Hieb's syntactic theory of control [6].

So far no characterization of contextual equivalence has been proposed for either call-by-value or call-by-name $\lambda\mu$-calculus. Lassen defined normal form bisimilarities for call-by-name weak-head reduction [10], for head reduction [12], and, with Støvring, for call-by-value weak-head reduction [19] that are not complete. However, normal form bisimilarity is complete for the $\Lambda\mu$-calculus [5] with head reduction [12], and also for the $\lambda\mu$-calculus with store [19] under call-by-value weak-head reduction. Lassen also defined an incomplete applicative bisimilarity for call-by-name weak-head reduction in [10]. A definition of applicative bisimilarity has also been proposed for a call-by-value typed μPCF [14], but the resulting relation is neither sound nor complete.

In this work, we propose the first characterizations of contextual equivalence for $\lambda\mu$-calculus for both call-by-name and call-by-value weak-head reduction semantics. The applicative bisimilarities we define are harder to use than Lassen's normal form bisimilarity to prove the equivalence of two given terms, but because they are complete, we can equate terms that cannot be related with normal form bisimilarity, such as David and Py's counter-example [4]. Even though the two applicative bisimilarities we define are built along the same principles, the relation we obtain in call-by-value is much more difficult to use than the one for call-by-name. However, we provide counter-examples showing that simplifying the call-by-value case so that it matches the call-by-name one leads to an unsound definition.

The paper is organized as follows. We first discuss the behavioral theory of the call-by-name (abbreviated as CBN) $\lambda\mu$-calculus in Section 2. We propose a notion of contextual equivalence (in Section 2.2) which observes top-level names, and we then characterize it with an applicative bisimilarity (Section 2.3). In particular, we compare our definition of bisimilarity with Lassen's work and we prove David and Py's counter-example using our relation. We then discuss call-by-value (CBV) in

Section 3. We propose a definition of applicative bisimilarity (Section 3.2) which coincides with contextual equivalence. We also provide counter-examples showing that the definition cannot be naively simplified to match the one for call-by-name. Although the relation we obtain is harder to use than the one for call-by-name, we can still prove some interesting equivalences of terms, as we demonstrate in Section 3.3. We conclude in Section 4, and the appendices contain some of the proofs missing from the main text (Appendix A for call-by-name and Appendix B for call-by-value). An accompanying research report [3] also discusses environmental bisimilarity for call-by-name.

2 Call-by-Name $\lambda\mu$-calculus

2.1 Syntax and Semantics

The $\lambda\mu$-calculus [15] extends the λ-calculus with named terms and a μ constructor that binds names in terms. We assume a set X of *variables*, ranged over by x, y, etc., and a distinct set A of *names*, ranged over by a, b, etc. Terms (T) and named terms (U) are defined by the following grammar:

$$\begin{aligned} \text{Terms:} \quad & t ::= x \mid \lambda x.t \mid t\,t \mid \mu a.u \\ \text{Named terms:} \quad & u ::= [a]t \end{aligned}$$

Values (V), ranged over by v, are terms of the form $\lambda x.t$. A λ-abstraction $\lambda x.t$ binds x in t and a μ-abstraction $\mu a.t$ binds a in t. We equate terms up to α-conversion of their bound variables and names, and we assume bound names to be pairwise distinct, as well as distinct from free names. We write $\mathsf{fv}(t)$ and $\mathsf{fv}(u)$ for the set of free variables of, respectively, t and u, and we write $\mathsf{fn}(t)$ and $\mathsf{fn}(u)$ for their set of free names. A term t or named term u is said *closed* if, respectively, $\mathsf{fv}(t) = \emptyset$ or $\mathsf{fv}(u) = \emptyset$. Note that a closed (named) term may contain free names. The sets of closed terms, closed values, and named terms are T^0, V^0, and U^0, respectively. In any discussion or proof, we say a variable or a name is *fresh* if it does not occur in any term under consideration.

We distinguish several kinds of contexts, represented outside-in, as follows:

$$\begin{aligned} \text{Contexts:} \quad & C ::= \Box \mid C\ t \mid t\ C \mid \lambda x.C \mid \mu a.\mathbb{C} \\ \text{Named contexts:} \quad & \mathbb{C} ::= [a]C \\ \text{CBN evaluation contexts:} \quad & E ::= \Box \mid E\ t \\ \text{Named evaluation contexts:} \quad & \mathbb{E} ::= [a]E \end{aligned}$$

The syntax of (named) evaluation contexts reflects the chosen reduction strategy, here call-by-name. Contexts can be filled only with a term t, to produce either regular terms $C[t]$, $E[t]$, or named terms $\mathbb{C}[t]$, $\mathbb{E}[t]$; the free names and free variables of t may be captured in the process.

We write $t_0\{t_1/x\}$ and $u_0\{t_1/x\}$ for the usual capture-avoiding substitution of terms for variables. We define the capture-avoiding substitution of named contexts for names, written $t\langle{}^{\mathbb{E}}\!/_{a}\rangle$ and $u\langle{}^{\mathbb{E}}\!/_{a}\rangle$, as follows. Note that the side-condition in the μ-binding case can always be fulfilled using α-conversion.

$$\begin{aligned}
x\langle{}^{\mathbb{E}}\!/_{a}\rangle &\overset{\text{def}}{=} x & (\mu b.u)\langle{}^{\mathbb{E}}\!/_{a}\rangle &\overset{\text{def}}{=} \mu b.u\langle{}^{\mathbb{E}}\!/_{a}\rangle \text{ if } b \notin \mathsf{fn}(\mathbb{E}) \cup \{a\}\\
(\lambda x.t)\langle{}^{\mathbb{E}}\!/_{a}\rangle &\overset{\text{def}}{=} \lambda x.t\langle{}^{\mathbb{E}}\!/_{a}\rangle & ([b]t)\langle{}^{\mathbb{E}}\!/_{a}\rangle &\overset{\text{def}}{=} \begin{cases} [b]t\langle{}^{\mathbb{E}}\!/_{a}\rangle & \text{if } a \neq b \\ \mathbb{E}[t\langle{}^{\mathbb{E}}\!/_{a}\rangle] & \text{if } a = b \end{cases}\\
(t_0\, t_1)\langle{}^{\mathbb{E}}\!/_{a}\rangle &\overset{\text{def}}{=} t_0\langle{}^{\mathbb{E}}\!/_{a}\rangle\; t_1\langle{}^{\mathbb{E}}\!/_{a}\rangle
\end{aligned}$$

We define the CBN reduction relation $\rightarrow_{\mathrm{n}}$ inductively by the following rules:

$$\begin{aligned}
(\beta_n) &\quad [a](\lambda x.t_0)\, t_1 \rightarrow_{\mathrm{n}} [a]t_0\{t_1/x\}\\
(\mu) &\quad [a]\mu b.u \rightarrow_{\mathrm{n}} u\langle{}^{[a]\Box}\!/_{b}\rangle\\
(\mathit{app}) &\quad [a]t_0\, t_1 \rightarrow_{\mathrm{n}} u\langle{}^{[a]\Box\, t_1}\!/_{b}\rangle \text{ if } [b]t_0 \rightarrow_{\mathrm{n}} u \text{ and } b \notin \mathsf{fn}([a]t_0\, t_1)
\end{aligned}$$

Reduction is defined on named terms only. The rule (β_n) is the usual call-by-name β-reduction. In rule (μ), the current continuation, represented by a, is captured and substituted for b in u. In an application (cf. rule (app)), we reduce the term t_0 in function position by introducing a fresh name b which represents the top level. We then replace b with $[a]\Box\; t_1$ in the result u of the reduction of $[b]t_0$. We can also express reduction with top-level evaluation contexts as follows.

Lemma 2.1 *$u \rightarrow_{\mathrm{n}} u'$ iff $u = \mathbb{E}[(\lambda x.t_0)\, t_1]$ and $u' = \mathbb{E}[t_0\{t_1/x\}]$, or $u = \mathbb{E}[\mu a.u'']$ and $u' = u''\langle\mathbb{E}/a\rangle$.*

Reduction is also compatible with evaluation contexts in the following sense.

Lemma 2.2 *If $u \rightarrow_{\mathrm{n}} u'$, then $u\langle{}^{\mathbb{E}}\!/_{a}\rangle \rightarrow_{\mathrm{n}} u'\langle{}^{\mathbb{E}}\!/_{a}\rangle$.*

We write $\rightarrow_{\mathrm{n}}^{*}$ for the transitive and reflexive closure of $\rightarrow_{\mathrm{n}}$, and we define the evaluation relation of the calculus as follows.

Definition 2.3 We write $u \Downarrow_{\mathrm{n}} u'$ if $u \rightarrow_{\mathrm{n}}^{*} u'$ and u' cannot reduce further.

If $u \Downarrow_{\mathrm{n}} u'$, then u' is a named value. If u admits an infinite reduction sequence, we say it *diverges*, written $u \Uparrow_{\mathrm{n}}$. For example, let $\Omega \overset{\text{def}}{=} (\lambda x.x\, x)\, (\lambda x.x\, x)$; then $[a]\Omega \Uparrow_{\mathrm{n}}$ for all a.

2.2 Contextual Equivalence

As in the λ-calculus, contextual equivalence in the $\lambda\mu$-calculus is defined in terms of convergence. However, unlike previous definitions [10,12], we define contextual equivalence on named terms first, before extending it to any terms.

Definition 2.4 Two closed terms u_0, u_1 are contextually equivalent, written $u_0 \approx_c u_1$, if for all closed contexts $\mathbb{C}$ and names a, there exist b, v_0, and v_1 such that $\mathbb{C}[\mu a.u_0] \Downarrow_{\mathrm{n}} [b]v_0$ iff $\mathbb{C}[\mu a.u_1] \Downarrow_{\mathrm{n}} [b]v_1$.

Note that we can plug only terms in a context, therefore we prefix u_0 and u_1 with a μ-abstraction. Definition 2.4 is not as generic as it could be, because we require the resulting named values to have the same top-level name b; a more general definition would simply say "$\mathbb{C}[\mu a.u_0] \Downarrow_{\mathrm{n}}$ iff $\mathbb{C}[\mu a.u_1] \Downarrow_{\mathrm{n}}$." Our definition is strictly finer than the general one, because contexts cannot discriminate upon top-level names in some cases, as we can see with the next example.

Example 2.5 Let $\Theta \stackrel{\mathrm{def}}{=} (\lambda x.\lambda y.y\ (x\ x\ y))\ (\lambda x.\lambda y.y\ (x\ x\ y))$ be Turing's CBN fixed-point combinator, and let $v \stackrel{\mathrm{def}}{=} \lambda x.\lambda y.x$. The terms $u_0 \stackrel{\mathrm{def}}{=} [a]\lambda x.\mu c.[b]\lambda y.\Theta\ v$ and $u_1 \stackrel{\mathrm{def}}{=} [b]\lambda y.\Theta\ v$ are distinguished by Definition 2.4 if $a \neq b$, but we show they are related by the general contextual equivalence. To do so, we verify that $\mathbb{E}[\mu c.u_0] \Downarrow_{\mathrm{n}}$ iff $\mathbb{E}[\mu c.u_1] \Downarrow_{\mathrm{n}}$ holds for all $\mathbb{E}$ and c, and we can then conclude that u_0 and u_1 are in the general equivalence with David and Py's context lemma [4]. Let $\mathbb{E}$ be of the form $[d]E\ t$ for some d, E, t. Then $\mathbb{E}[\mu a.u_0] \Downarrow_{\mathrm{n}} [b]\lambda y.\Theta\ v$ and $\mathbb{E}[\mu a.u_1] \Downarrow_{\mathrm{n}} [b]\lambda y.\Theta\ v$, $\mathbb{E}[\mu b.u_0] \Downarrow_{\mathrm{n}} [a]\lambda x.\mu c.\mathbb{E}[\lambda y.\Theta\ v]$ and $\mathbb{E}[\mu b.u_1] \Downarrow_{\mathrm{n}} [d]\lambda y.\Theta\ v$, and finally $\mathbb{E}[\mu c.u_0] \Downarrow_{\mathrm{n}} u_0$ and $\mathbb{E}[\mu c.u_1] \Downarrow_{\mathrm{n}} u_1$ for $c \notin \{a, b\}$. The case $\mathbb{E} = [d]\square$ is easy to check as well.

We choose Definition 2.4 because it gives more information on the behaviors of terms than the general equivalence. Besides, only very peculiar terms u_0 and u_1 are related by the general equivalence but not by Definition 2.4. These terms are like black holes: they reduce (in some context $\mathbb{C}$) to values $[a]v_0$ and $[b]v_1$ with $a \neq b$ that never evaluate their arguments. Indeed, if $\mathbb{E} = [c]\square\ t_0 \ldots t_n$, then $\mathbb{E}[\mu a.[a]v_0] \to_{\mathrm{n}} \mathbb{E}[v_0\langle {}^{\mathbb{E}}\!/\!_{a}\rangle]$, and $\mathbb{E}[\mu a.[b]v_1] \Downarrow_{\mathrm{n}} [b]v_1\langle {}^{\mathbb{E}}\!/\!_{a}\rangle$. Suppose that when evaluating $\mathbb{E}[v_0\langle {}^{\mathbb{E}}\!/\!_{a}\rangle]$, we evaluate one of the t_i's. Then by replacing t_i with Ω, we obtain a context $\mathbb{E}'$ such that $\mathbb{E}'[\mu a.[a]v_0] \Uparrow_{\mathrm{n}}$ (because Ω will be evaluated), and $\mathbb{E}'[\mu a.[b]v_1] \Downarrow_{\mathrm{n}}$, which is in contradiction with the fact that u_0 and u_1 are in the general equivalence (they are distinguished by $\mathbb{E}'[\mu a.\mathbb{C}]$).

We extend Definition 2.4 to any closed terms t_0, t_1, by saying that $t_0 \approx_c t_1$ if $[a]t_0 \approx_c [a]t_1$ for any fresh a. Other versions of the extension are possible, for example by replacing "for any a" by "for some a", or by dropping the freshness requirement; as can be shown using the results of Section 2.3, all these definitions are equivalent. We can also define contextual equivalence on open terms, using the notion of *open extension*, which extends any relation on closed (named) terms to open (named) terms. We say a substitution σ closes t (or u) if σ replaces the variables in $\mathsf{fv}(t)$ (or $\mathsf{fv}(u)$) with closed terms.

Definition 2.6 Let $\mathcal{R}$ be a relation on closed (named) terms. Two terms t_0 and t_1 are in the open extension of $\mathcal{R}$, written $t_0\ \mathcal{R}^\circ\ t_1$, if for all substitutions σ closing t_0 and t_1, we have $t_0\sigma\ \mathcal{R}\ t_1\sigma$ (and similarly for $u_0\ \mathcal{R}^\circ\ u_1$).

2.3 Applicative Bisimilarity

We propose a notion of applicative bisimulation, which tests values by applying them to a random closed argument. As with contextual equivalence, we give the definitions for named terms, before extending it to regular terms.

Definition 2.7 A relation $\mathcal{R}$ on closed named terms is an applicative bisimulation if $u_0 \mathrel{\mathcal{R}} u_1$ implies

- if $u_0 \rightarrow_{\mathrm{n}} u_0'$, then there exists u_1' such that $u_1 \rightarrow_{\mathrm{n}}^{*} u_1'$ and $u_0' \mathrel{\mathcal{R}} u_1'$;
- if $u_0 = [a]\lambda x.t_0$, then there exists t_1 such that $u_1 \rightarrow_{\mathrm{n}}^{*} [a]\lambda x.t_1$ and for all t, we have $[a]t_0\langle {}^{[a]\Box\, t}/_{a}\rangle\{t/x\} \mathrel{\mathcal{R}} [a]t_1\langle {}^{[a]\Box\, t}/_{a}\rangle\{t/x\}$;
- the symmetric conditions on u_1.

Applicative bisimilarity, written $\approx$, is the largest applicative bisimulation.

For regular terms, we write $t_0 \mathrel{\mathcal{R}} t_1$ if $[a]t_0 \mathrel{\mathcal{R}} [a]t_1$ for any $a \notin \mathsf{fn}(t_0, t_1)$. The first item of Definition 2.7 plays the bisimulation game for named terms which are not named values. If u_0 is a named value $[a]\lambda x.t_0$, then u_1 has to reduce to a named value $[a]\lambda x.t_1$, and we compare the values by applying them to an argument t. However, a context cannot interact with $[a]\lambda x.t_0$ and $[a]\lambda x.t_1$ by simply applying them to t, because $([a]\lambda x.t_0)\, t$ is not allowed by the syntax. Consequently, we have to prefix them first with μa. As a result, we consider the named terms $[a](\mu a.[a]\lambda x.t_0)\, t$ and $[a](\mu a.[a]\lambda x.t_1)\, t$, which reduce to, respectively, $[a](\lambda x.t_0\langle {}^{[a]\Box\, t}/_{a}\rangle)\ t$ and $[a](\lambda x.t_1\langle {}^{[a]\Box\, t}/_{a}\rangle)\ t$, and then to $[a]t_0\langle {}^{[a]\Box\, t}/_{a}\rangle\{t/x\}$ and $[a]t_1\langle {}^{[a]\Box\, t}/_{a}\rangle\{t/x\}$; we obtain the terms in the clause for values of Definition 2.7.

Remark 2.8 When considering $[a](\mu a.[a]\lambda x.t_0)\, t$ and $[a](\mu a.[a]\lambda x.t_1)\, t$, we use the same top-level name a as the one of the named values $[a]\lambda x.t_0$ and $[a]\lambda x.t_1$. We could use a fresh name b instead; reusing the same name makes the bisimulation proofs easier (we do not have to introduce unnecessary fresh names).

We can also define a big-step version of the bisimulation, where we consider only evaluation to a value.

Definition 2.9 A relation $\mathcal{R}$ on closed named terms is a big-step applicative bisimulation if $u_0 \mathrel{\mathcal{R}} u_1$ implies

- if $u_0 \rightarrow_{\mathrm{n}}^{*} [a]\lambda x.t_0$, then there exists t_1 such that $u_1 \rightarrow_{\mathrm{n}}^{*} [a]\lambda x.t_1$ and for all t, we have $[a]t_0\langle {}^{[a]\Box\, t}/_{a}\rangle\{t/x\} \mathrel{\mathcal{R}} [a]t_1\langle {}^{[a]\Box\, t}/_{a}\rangle\{t/x\}$;
- the symmetric condition on u_1.

Lemma 2.10 *If $\mathcal{R}$ is a big-step applicative bisimulation, then $\mathcal{R} \subseteq \approx$.*

As a first property, we prove that reduction (and therefore, evaluation) is included in bisimilarity.

Lemma 2.11 *We have $\rightarrow_{\mathrm{n}}^{*} \subseteq \approx$.*

Proof. By showing that $\{(u, u') \mid u \rightarrow_{\mathrm{n}}^{*} u'\} \cup \{(u, u)\}$ is a big-step bisimulation. □

We give a basic example to show how applicative bisimulation can be used.

Example 2.12 For all closed v and $b \notin \mathsf{fn}(v)$, we prove that $[a]v \approx [a]\lambda x.\mu b.[a]v$ by showing that $\{([a]v, [a]\lambda x.\mu b.[a]v) \mid b \notin \mathsf{fn}(v)\} \cup \approx$ is an applicative bisimulation. Indeed, if $v = \lambda x.t$, then for all t', we have $[a]t\{t'/x\} \approx [a]\mu b.[a]v\ t'$, because $[a]\mu b.[a]v\, t' \rightarrow_{\mathrm{n}}^{*} [a]t\{t'/x\}$ (and by Lemma 2.11).

2.4 Soundness and Completeness

We now prove that $\approx$ coincides with $\approx_c$. We first show that $\approx$ is a congruence using Howe's method [8,7], which is a classic proof method to show that an applicative bisimilarity is a congruence. As in [10], we need to slightly adapt the proof to the $\lambda\mu$-calculus. Here we only sketch the application of the method, all the details can be found in Appendix A.1.

The principle of the method is to prove that a relation called the *Howe's closure* of $\approx$, which is a congruence by construction, is also a bisimulation. The definition of Howe's closure relies on an auxiliary relation, called the *compatible refinement* $\widetilde{\mathcal{R}}$ of a relation $\mathcal{R}$, and inductively defined by the following rules:

$$\frac{}{x \mathrel{\widetilde{\mathcal{R}}} x} \qquad \frac{t_0 \mathrel{\mathcal{R}} t_1}{\lambda x.t_0 \mathrel{\widetilde{\mathcal{R}}} \lambda x.t_1} \qquad \frac{t_0 \mathrel{\mathcal{R}} t_1 \qquad t'_0 \mathrel{\mathcal{R}} t'_1}{t_0\ t'_0 \mathrel{\widetilde{\mathcal{R}}} t_1\ t'_1} \qquad \frac{u_0 \mathrel{\mathcal{R}} u_1}{\mu a.u_0 \mathrel{\widetilde{\mathcal{R}}} \mu a.u_1}$$

$$\frac{t_0 \mathrel{\mathcal{R}} t_1}{[a]t_0 \mathrel{\widetilde{\mathcal{R}}} [a]t_1} \qquad \frac{t_0 \mathrel{\mathcal{R}} t_1 \qquad \mathbb{E}_0 \mathrel{\widetilde{\mathcal{R}}} \mathbb{E}_1}{t_0\langle \mathbb{E}_0/a\rangle \mathrel{\widetilde{\mathcal{R}}} t_1\langle \mathbb{E}_1/a\rangle} \qquad \frac{u_0 \mathrel{\mathcal{R}} u_1 \qquad \mathbb{E}_0 \mathrel{\widetilde{\mathcal{R}}} \mathbb{E}_1}{u_0\langle \mathbb{E}_0/a\rangle \mathrel{\widetilde{\mathcal{R}}} u_1\langle \mathbb{E}_1/a\rangle}$$

$$\frac{}{\Box \mathrel{\widetilde{\mathcal{R}}} \Box} \qquad \frac{E_0 \mathrel{\widetilde{\mathcal{R}}} E_1 \qquad t_0 \mathrel{\mathcal{R}} t_1}{E_0\ t_0 \mathrel{\widetilde{\mathcal{R}}} E_1\ t_1} \qquad \frac{E_0 \mathrel{\widetilde{\mathcal{R}}} E_1}{[a]E_0 \mathrel{\widetilde{\mathcal{R}}} [a]E_1}$$

In the original definition of compatible refinement [7], two terms are related by $\widetilde{\mathcal{R}}$ if they have the same outer language constructor, and their subterms are related by $\mathcal{R}$. In the $\lambda\mu$-calculus, compatible refinement is extended to (named) evaluation contexts, and we allow for the substitution of names with related named contexts.

Given two relations $\mathcal{R}_1$ and $\mathcal{R}_2$, we write $\mathcal{R}_1\mathcal{R}_2$ for their composition, e.g., $t_0 \mathrel{\mathcal{R}_1\mathcal{R}_2} t_2$ holds if there exists t_1 such that $t_0 \mathrel{\mathcal{R}_1} t_1$ and $t_1 \mathrel{\mathcal{R}_2} t_2$. We can now define Howe's closure of $\approx$, written $\approx^\bullet$, as follows.

Definition 2.13 The Howe's closure $\approx^\bullet$ is the smallest relation verifying:

$$\approx^\circ \subseteq \approx^\bullet \qquad\qquad \approx^\bullet\approx^\circ \subseteq \approx^\bullet \qquad\qquad \widetilde{\approx^\bullet} \subseteq \approx^\bullet$$

Howe's closure is defined on open (named) terms as well as on (named) evaluation contexts. Because it contains its compatible refinement, $\approx^\bullet$ is a congruence. To prove it is a bisimulation, we need a stronger result, called a pseudo-simulation lemma, where we test named values not with the same argument, but with arguments t'_0, t'_1 related by $\approx^\bullet$.

Lemma 2.14 *Let $(\approx^\bullet)^c$ be $\approx^\bullet$ restricted to closed terms, and let $u_0\ (\approx^\bullet)^c\ u_1$.*

- *If $u_0 \rightarrow_{\mathrm{n}} u'_0$, then $u_1 \rightarrow^*_{\mathrm{n}} u'_1$ and $u'_0\ (\approx^\bullet)^c\ u'_1$.*
- *If $u_0 = [a]\lambda x.t_0$, then $u_1 \rightarrow^*_{\mathrm{n}} [a]\lambda x.t_1$ and for all $t'_0\ (\approx^\bullet)^c\ t'_1$, we have $[a]t_0\langle {}^{[a]\Box\ t'_0}/a\rangle\{t'_0/x\}\ (\approx^\bullet)^c\ [a]t_1\langle {}^{[a]\Box\ t'_1}/a\rangle\{t'_1/x\}$.*

With this result, we can prove that $(\approx^\bullet)^c$ is a bisimulation, and therefore included in $\approx$. Because it also contains $\approx$ by definition, we have $\approx=(\approx^\bullet)^c$, and this implies that $\approx$ is a congruence. As a result, $\approx$ is sound w.r.t. to $\approx_c$.

Theorem 2.15 $\approx \subseteq \approx_c$.

To simplify the proof of completeness (the reverse inclusion), we consider an alternate definition of contextual equivalence, where we test terms with named evaluation contexts only. By doing so, we prove a context lemma in the process. [3]

Definition 2.16 Let u_0, u_1 be closed terms. We write $u_0 \mathrel{\dot{\approx}_c} u_1$ if for all closed contexts $\mathbb{E}$ and names a, there exist b, v_0, v_1 such that $\mathbb{E}[\mu a.u_0] \Downarrow_{\mathrm{n}} [b]v_0$ iff $\mathbb{E}[\mu a.u_1] \Downarrow_{\mathrm{n}} [b]v_1$.

Theorem 2.17 $\approx_c \subseteq \dot{\approx}_c \subseteq \approx$.

The first inclusion is by definition, and the second one is by showing that $\dot{\approx}_c$ is a big-step applicative bisimulation.

2.5 Comparison with Lassen's Work

In [10], Lassen also proposes a definition of applicative bisimilarity that he proves sound, but he conjectures that it is not complete. We discuss here the differences between the two approaches.

Lassen defines a notion of bisimulation for regular terms only, and not for named terms. The definition is as follows.

Definition 2.18 A relation $\mathcal{R}$ on closed terms is a Lassen applicative bisimulation if $t_0 \mathrel{\mathcal{R}} t_1$ implies:

- for all a, if $[a]t_0 \rightarrow_{\mathrm{n}}^{*} [b]\lambda x.t_0'$, then there exists t_1' such that $[a]t_1 \rightarrow_{\mathrm{n}}^{*} [b]\lambda x.t_1'$, and for all t, we have $t_0'\langle {}^{[b]\Box\, t}\!/_{b}\rangle\{t/x\} \mathrel{\mathcal{R}} t_1'\langle {}^{[b]\Box\, t}\!/_{b}\rangle\{t/x\}$;
- the symmetric condition on t_1.

Lassen's definition is quite similar to our definition of big-step applicative bisimulation (Definition 2.9), except it requires $t_0\langle {}^{[b]\Box\, t}\!/_{b}\rangle\{t/x\} \mathrel{\mathcal{R}} t_1\langle {}^{[b]\Box\, t}\!/_{b}\rangle\{t/x\}$, which implies that these terms must be related when reduced with any top-level name a. This is more restrictive than our definition, where we compare these terms only with the top-level name b (or, as discussed in Remark 2.8, we could instead compare $[c]t_0\langle {}^{[c]\Box\, t}\!/_{b}\rangle\{t/x\}$ and $[c]t_1\langle {}^{[c]\Box\, t}\!/_{b}\rangle\{t/x\}$ for some fresh name c). To illustrate the difference, we consider Lassen's counter-example from [10].

Example 2.19 Let $t_0 \overset{\text{def}}{=} (\lambda x.\lambda y.x\, x)\,(\lambda x.\lambda y.x\, x)$, and $t_1 \overset{\text{def}}{=} \mu a.[a]\lambda y.\mu c.[a]t_0$ (with $c \neq a$). These terms are not bisimilar according to Lassen's definition. For all b, we have $[b]t_0 \rightarrow_{\mathrm{n}}^{*} [b]\lambda y.t_0$ and $[b]t_1 \rightarrow_{\mathrm{n}}^{*} [b]\lambda y.\mu c.[b]t_0$. With Lassen's definition, one has to relate t_0 and $\mu c.[b]t_0\, t$ for any t, which means comparing $[d]t_0$ and $[d]\mu c.[b]t_0\, t$ for all d. But these two terms are not equivalent if $d \neq b$.

Lassen conjectures in [10] that these terms are contextually equivalent, and we can indeed prove that they are (big-step) bisimilar with our definition: we just have

[3] We cannot directly use David and Py's context lemma [4], because we use a different notion of contextual equivalence.

to compare $[b]t_0$ and $[b]\mu a'.[b]t_0\, t$ (or $[c]t_0$ and $[c]\mu a'.[c]t_0\, t$ for some fresh c) for any t, and both terms evaluate to $[b]\lambda x.t_0$ (or $[c]\lambda x.t_0$) and are therefore equivalent.

By comparing primarily named terms, as we do in our definition, we can keep track of what happens to the top level, and especially of any connection between the top level and a subterm. In Example 2.19, we can see that it is essential to remember that b represents the top level in $\mu c.[b]t_0\, t$, and therefore it does not make sense to compare $[d]t_0$ and $[d]\mu c.[b]t_0 t$ for any $d \neq b$, as we have to do with Lassen's definition. We believe that comparing named terms is essential to obtain completeness w.r.t. contextual equivalence; note that the sound and complete normal form bisimilarity for the $\lambda\mu\rho$-calculus [19] is also defined on named terms.

2.6 David and Py's Counter-Example

In [4], David and Py give a counter-example showing that Böhm's theorem fails in CBN $\lambda\mu$-calculus. They prove that their terms are contextually equivalent using a context lemma. Here we slightly simplify their counter-example, and prove equivalence using applicative bisimilarity. Note that these terms cannot be proved equivalent with (a CBN variant of) eager normal form bisimilarity [10,19].

Example 2.20 Let $0 \overset{\text{def}}{=} \lambda x.\lambda y.y$, $1 \overset{\text{def}}{=} \lambda x.\lambda y.x$, and $t_a \overset{\text{def}}{=} \mu c.[a]0$. Then we have $\lambda x.\mu a.[a]x\; \mu b.[a]x\; t_a\; 0 \approx \lambda x.\mu a.[a]x\; \mu b.[a]x\; t_a\; 1$ [4] .

Proof. [Sketch] We only give the main ideas here, the complete equivalence proof can be found in Appendix A.2. First, $\lambda x.\mu a.[a]x\; \mu b.[a]x\; t_a\; 0$ is not normal form bisimilar to $\lambda x.\mu a.[a]x\, \mu b.[a]x\, t_a\, 1$, because the subterms of these two terms are not normal form bisimilar (0 is not equivalent to 1).

To prove applicative bisimilarity, let c be a fresh name and t be a closed term. We want to relate $[c]\mu a.[a]t\, \mu b.[a]t t_a 0$ and $[c]\mu a.[a]t\, \mu b.[a]t t_a 1$, which reduce respectively to $[c]t\, \mu b.[c]t\, t_c 0$ (1) and $[c]t\, \mu b.[c]t\, t_c 1$ (2). Let $d \notin \mathsf{fn}(t)$; we distinguish several cases depending on the behavior of $[d]t$. The interesting case is when $[d]t \Downarrow_{\mathrm{n}} [d]\lambda y.t'$; then $\mu b.[c]t\; t_c\; 0$ or $\mu b.[c]t\; t_c\; 1$ is passed as an argument to $\lambda y.t'$ in respectively (1) and (2). If t' executes its argument (that is, if t' reduces to $E[y]$ for some E), then (1) reduces to $[c]t\, t_c\, 0$ (3), and (2) to $[c]t\, t_c\, 1$ (4). But we know that $[d]t \Downarrow_{\mathrm{n}} [d]\lambda y.t'$, and t' executes its argument, so when evaluating (3) and (4), t_c will be reduced, and therefore (3) and (4) will evaluate to $[c]0$.

In the other cases (e.g., $[d]t \Downarrow_{\mathrm{n}} [e]\lambda y.t'$ with $e \neq d$), either (1) and (2) eventually get to a point similar to the situation above where t_c is executed, or they diverge. In all cases, they are applicative bisimilar. □

[4] The terms David and Py consider in their work are $\lambda x.\mu a.[a]x\mu b.[a](xt_a 0)t_a$ and $\lambda x.\mu a.[a]x\mu b.[a](xt_a 1)t_a$. However, the additional argument t_a would not come into play in the proof we present, so we have elided it.

3 Call-by-Value $\lambda\mu$-calculus

3.1 Semantics and Contextual Equivalence

In this section, we use CBV left-to-right evaluation, which is encoded in the syntax of the CBV evaluation contexts:

$$E ::= \Box \mid E\,t \mid v\,E$$

The CBV reduction relation $\rightarrow_{\mathrm{v}}$ is defined by the following rules.

$$\begin{array}{rrll}
(\beta_v) & [a](\lambda x.t)\,v \rightarrow_{\mathrm{v}} [a]t\{v/x\} & & \\
(\mu) & [a]\mu b.u \rightarrow_{\mathrm{v}} u\langle{}^{[a]\Box}/_{b}\rangle & & \\
(app) & [a]t_0\,t_1 \rightarrow_{\mathrm{v}} u\langle{}^{[a]\Box\,t_1}/_{b}\rangle & \text{if } [b]t_0 \rightarrow_{\mathrm{v}} u \text{ and } b \notin \mathsf{fn}([a]t_0\,t_1) \\
(app_v) & [a]v\,t \rightarrow_{\mathrm{v}} u\langle{}^{[a]v\,\Box}/_{b}\rangle & \text{if } [b]t \rightarrow_{\mathrm{v}} u \text{ and } b \notin \mathsf{fn}([a]v\,t)
\end{array}$$

With rule (app_v), we reduce arguments to values, to be able to apply CBV β-reduction (rule (β_v)). The rules (μ) and (app) are unchanged. We could also express reduction with top-level named evaluation contexts, as in Lemma 2.1. Furthermore, CBV reduction is compatible with CBV contexts, as in Lemma 2.2. We write $\rightarrow_{\mathrm{v}}^*$ for the reflexive and transitive closure of $\rightarrow_{\mathrm{v}}$, $\Downarrow_{\mathrm{v}}$ for CBV evaluation, and $\Uparrow_{\mathrm{v}}$ for CBV divergence.

We use the same definition of contextual equivalence as in CBN.

Definition 3.1 Let u_0, u_1 be closed named terms. We write $u_0 \approx_c u_1$, if for all closed contexts $\mathbb{C}$ and names a, there exist b, v_0, and v_1 such that $\mathbb{C}[\mu a.u_0] \Downarrow_{\mathrm{v}} [b]v_0$ iff $\mathbb{C}[\mu a.u_1] \Downarrow_{\mathrm{v}} [b]v_1$.

However, unlike in CBN, this definition (where we require the resulting values to have the same top-level names) coincides with the general definition where we simply say "$\mathbb{C}[\mu a.u_0] \Downarrow_{\mathrm{v}}$ iff $\mathbb{C}[\mu a.u_1] \Downarrow_{\mathrm{v}}$." Indeed, if $\mathbb{C}[\mu a.u_0] \Downarrow_{\mathrm{v}} [b]v_0$ and $\mathbb{C}[\mu a.u_1] \Downarrow_{\mathrm{v}} [c]v_1$ with $c \neq b$, then we can easily distinguish them, because $[b]\mu b.\mathbb{C}[\mu a.u_0]\;\Omega \rightarrow_{\mathrm{v}}^* [b]v_0\langle{}^{[b]\Box\,\Omega}/_{b}\rangle\;\Omega \Uparrow_{\mathrm{v}}$, and $[b]\mu b.\mathbb{C}[\mu a.u_1]\;\Omega \Downarrow_{\mathrm{v}} [c]v_1\langle{}^{[b]\Box\,\Omega}/_{b}\rangle$.

We extend $\approx_c$ to any closed terms as in CBN. The definition of open extension is slightly changed in CBV, compared to CBN: we close open terms by substituting their variables with closed *values* only, and not any closed terms.

3.2 Applicative Bisimilarity

Before giving its complete definition, we explain how applicative bisimilarity $\approx$ should compare two named values $[a]\lambda x.t_0$ and $[a]\lambda x.t_1$. The following reasoning explains and justifies the clauses in Definition 3.4. In particular, we provide counter-examples to show that we cannot simplify this definition.

In CBV λ-calculus (and also with delimited control [2]), values are tested by applying them to an arbitrary value argument. Following this principle, it is natural to propose the following clause for CBV $\lambda\mu$-calculus.

(1) For all v, we have $[a]t_0\langle{}^{[a]\Box\ v}/_{a}\rangle\{v/x\} \approx [a]t_1\langle{}^{[a]\Box\ v}/_{a}\rangle\{v/x\}$.

As with Definition 2.7, we in fact compare $[a](\mu a.[a]\lambda x.t_0)\, v$ with $[a](\mu a.[a]\lambda x.t_1)\, v$, which reduce to the terms in clause (1). However, such a clause would produce an unsound applicative bisimilarity; it would relate terms that are not contextually equivalent, like the ones in the next example.

Example 3.2 Let $v_0 \stackrel{\text{def}}{=} \lambda x.\mu b.[a]w\ x$, $v_1 \stackrel{\text{def}}{=} \lambda x.w\ x\ x$, with $w \stackrel{\text{def}}{=} \lambda y.\lambda z.z\ y$. Then we have $([a]v_0)\langle{}^{[a]\Box\ v}/_{a}\rangle = [a](\lambda x.\mu b.[a]w\ x\ v)\ v \rightarrow_{\mathrm{v}}^{*} [a]w\ v\ v$ and $([a]v_1)\langle{}^{[a]\Box\ v}/_{a}\rangle = [a](\lambda x.w\ x\ x)\ v \rightarrow_{\mathrm{v}}^{*} [a]w\ v\ v$. Because they reduce to the same term, $([a]v_0)\langle{}^{[a]\Box\ v}/_{a}\rangle$ is contextually equivalent to $([a]v_1)\langle{}^{[a]\Box\ v}/_{a}\rangle$, and using clause (1) would lead us to conclude that $[a]v_0$ and $[a]v_1$ are equivalent as well.

However, $[a]v_0$ and $[a]v_1$ can be distinguished with $t \stackrel{\text{def}}{=} \mu d.[d]\lambda y.\mu c.[d]w'$, where $w' \stackrel{\text{def}}{=} \lambda x.\mu c.[d]x\ w''$ and $w'' \stackrel{\text{def}}{=} \lambda x.\lambda y.\lambda z.\Omega$. Indeed, we can check that $([a]v_0)\langle{}^{[a]\Box\ t}/_{a}\rangle \rightarrow_{\mathrm{v}}^{*} \lambda z.\Omega$ and $([a]v_1)\langle{}^{[a]\Box\ t}/_{a}\rangle \rightarrow_{\mathrm{v}}^{*} \Omega$. This discrepancy comes from the fact that, in $([a]v_1)\langle{}^{[a]\Box\ t}/_{a}\rangle$, t is reduced to a value once, capturing $[a]v_1\ \Box$ in the process, while t is reduced twice to a value in $([a]v_0)\langle{}^{[a]\Box\ t}/_{a}\rangle$, and each time it captures a different context. Therefore, $[a]v_0$ and $[a]v_1$ are distinguished by the context $[a](\mu a.\Box)\ t$, and they are consequently not contextually equivalent.

Example 3.2 suggests that we should compare $[a]\lambda x.t_0$ and $[a]\lambda x.t_1$ with contexts of the form $[a]\Box\ t$, instead of $[a]\Box\ v$. Therefore, we should compare $u_0 \stackrel{\text{def}}{=} [a](\lambda x.t_0\langle{}^{[a]\Box\ t}/_{a}\rangle)\ t$ with $u_1 \stackrel{\text{def}}{=} [a](\lambda x.t_1\langle{}^{[a]\Box\ t}/_{a}\rangle)\ t$. However, we can restrict a bit the choice of the testing term t, based on its behavior. Let $b \notin \mathsf{fn}(t)$; if $[b]t$ diverges, then u_0 and u_1 diverge as well, and we gain no information on $[a]\lambda x.t_0$ and $[a]\lambda x.t_1$ themselves. If $[b]t \rightarrow_{\mathrm{v}}^{*} [c]v$ with $b \neq c$, then $u_0 \rightarrow_{\mathrm{v}}^{*} [c]v\langle{}^{[a]\lambda x.t_0\langle{}^{[a]\Box\ t}/_{a}\rangle\ \Box}/_{b}\rangle$, and similarly with u_1. The values $[a]\lambda x.t_0$ and $[a]\lambda x.t_1$ are captured by $[b]t$, and no interaction between t and the two named values takes place in the process ($[a]\lambda x.t_0$ and $[a]\lambda x.t_1$ are not applied to any value); again, we do not gain any new knowledge on the behavior of $[a]\lambda x.t_0$ and $[a]\lambda x.t_1$. Finally, if $[b]t \rightarrow_{\mathrm{v}}^{*} [b]v$, then $u_0 \rightarrow_{\mathrm{v}}^{*} [b](\lambda x.t_0\langle{}^{[a]\Box\ t}/_{a}\rangle)\ v\langle{}^{[a]\lambda x.t_0\langle{}^{[a]\Box\ t}/_{a}\rangle\ \Box}/_{b}\rangle$, and similarly with u_1; in this case, a value is indeed passed to $[a]\lambda x.t_0$ and $[a]\lambda x.t_1$, and we can compare their respective behaviors. Therefore, an interaction happens between t and the tested values iff $[b]t \rightarrow_{\mathrm{v}}^{*} [b]v$, and the results of the interaction (after β-reduction) are the two terms in the clause below.

(2) For all t, b, v such that $[b]t \rightarrow_{\mathrm{v}}^{*} [b]v$ and $b \notin \mathsf{fn}(t)$, we have

$$[a]t_0\langle{}^{[a]\Box\ t}/_{a}\rangle\{v\langle{}^{[a]\lambda x.t_0\langle{}^{[a]\Box\ t}/_{a}\rangle\ \Box}/_{b}\rangle/x\} \approx [a]t_1\langle{}^{[a]\Box\ t}/_{a}\rangle\{v\langle{}^{[a]\lambda x.t_1\langle{}^{[a]\Box\ t}/_{a}\rangle\ \Box}/_{b}\rangle/x\}.$$

Unfortunately, clause (2) is not enough to obtain a sound bisimilarity. The next example shows that an extra clause is needed.

Example 3.3 Let $v_0 \stackrel{\text{def}}{=} \lambda x.\mu b.[a](\lambda y.\lambda z.wy)x$ and $v_1 \stackrel{\text{def}}{=} w$ with $w \stackrel{\text{def}}{=} \lambda x.w'(x\lambda y.y)$, and $w' \stackrel{\text{def}}{=} \lambda y.y\ \lambda z.\Omega$. We first show that $[a]v_0$ and $[a]v_1$ are related by clause (2). Let t such that $[b]t \Downarrow_{\mathrm{v}} [b]v$ for $b \notin \mathsf{fn}(t)$. Then we have $([a]v_0)\langle{}^{[a]\Box\ t}/_{a}\rangle \rightarrow_{\mathrm{v}}^{*} [a]w'\ (v\langle{}^{[a]v_0\langle{}^{[a]\Box\ t}/_{a}\rangle\ \Box}/_{b}\rangle\ \lambda x.x)$ and $([a]v_1)\langle{}^{[a]\Box\ t}/_{a}\rangle \rightarrow_{\mathrm{v}}^{*} [a]w'\ (v\langle{}^{[a]v_1\ \Box}/_{b}\rangle\ \lambda x.x)$. We can prove that the two resulting terms are contextually equivalent by showing that the relation $\{(u\langle {}^{[a]E[v_0\langle {}^{[a]E[\Box\; t]}/_{a}\rangle\; \Box]}/_{b}\rangle, u\langle {}^{[a]E[v_1\; \Box]}/_{b}\rangle) \;\mid\; [b]t \Downarrow_{\mathrm{v}} [b]v, b \notin \mathsf{fn}(t)\}$ is an applicative bisimulation according to Definition 3.4, and by using Theorem 3.9. The proof can be found in Appendix B.1. Because $([a]v_0)\langle {}^{[a]\Box\; t}/_{a}\rangle$ and $([a]v_1)\langle {}^{[a]\Box\; t}/_{a}\rangle$ are contextually equivalent, using only clause (2) would lead us to conclude that $[a]v_0$ and $[a]v_1$ are also equivalent.

However, these two named values can be distinguished with the context $[a](\lambda x.x\; x)\; \mu a.\Box$, because in one case we have $([a]v_0)\langle {}^{[a](\lambda x.x\; x)\; \Box}/_{a}\rangle \rightarrow^{*}_{\mathrm{v}} \lambda z.\Omega$, and in the other $([a]v_1)\langle {}^{[a](\lambda x.x\; x)\; \Box}/_{a}\rangle \rightarrow^{*}_{\mathrm{v}} \Omega$. As in Example 3.2, when evaluating $([a]v_0)\langle {}^{[a](\lambda x.x\; x)\; \Box}/_{a}\rangle$, the body of v_0 is evaluated twice, and two different contexts are captured each time. In contrast, v_1 does not contain any control effect, so when its body is evaluated twice, we get the same result.

Example 3.3 shows that we have to compare two values $[a]\lambda x.t_0$ and $[a]\lambda x.t_1$ by also testing them with contexts of the form $[a]v\; \Box$, i.e., by considering $[a]v\; \lambda x.t_0\langle {}^{[a]v\; \Box}/_{a}\rangle$ and $[a]v\; \lambda x.t_1\langle {}^{[a]v\; \Box}/_{a}\rangle$. If $v = \lambda x.t$, then these terms reduce in one β-reduction step into $[a]t\{\lambda x.t_0\langle {}^{[a]v\; \Box}/_{a}\rangle/x\}$, and $[a]t\{\lambda x.t_1\langle {}^{[a]v\; \Box}/_{a}\rangle/x\}$. Taking this and clause (2) into account, we obtain the following definition of applicative bisimulation.

Definition 3.4 A relation $\mathcal{R}$ on closed named terms is an applicative bisimulation if $u_0 \;\mathcal{R}\; u_1$ implies

- if $u_0 \rightarrow_{\mathrm{v}} u_0'$, then there exists u_1' such that $u_1 \rightarrow^{*}_{\mathrm{v}} u_1'$ and $u_0' \;\mathcal{R}\; u_1'$;
- if $u_0 = [a]\lambda x.t_0$, then there exists t_1 such that $u_1 \rightarrow^{*}_{\mathrm{v}} [a]\lambda x.t_1$, and:

(i) for all t, b, v such that $[b]t \rightarrow^{*}_{\mathrm{v}} [b]v$ and $b \notin \mathsf{fn}(t)$, we have

$$[a]t_0\langle {}^{[a]\Box\; t}/_{a}\rangle\{v\langle {}^{[a]\lambda x.t_0\langle {}^{[a]\Box\; t}/_{a}\rangle\; \Box}/_{b}\rangle/x\} \;\mathcal{R}\; [a]t_1\langle {}^{[a]\Box\; t}/_{a}\rangle\{v\langle {}^{[a]\lambda x.t_1\langle {}^{[a]\Box\; t}/_{a}\rangle\; \Box}/_{b}\rangle/x\};$$

(ii) for all $v = \lambda x.t$, we have

$$[a]t\{\lambda x.t_0\langle {}^{[a]v\; \Box}/_{a}\rangle/x\} \;\mathcal{R}\; [a]t\{\lambda x.t_1\langle {}^{[a]v\; \Box}/_{a}\rangle/x\};$$

- the symmetric conditions on u_1.

Applicative bisimilarity, written $\approx$, is the largest applicative bisimulation.

The definition is extended to regular terms t_0, t_1 as in CBN, by using a fresh top-level name a. Note that clause (ii) implies that a bisimulation $\mathcal{R}$ is a congruence w.r.t. (regular) values; indeed, if $v_0 \;\mathcal{R}\; v_1$, then $[a]v_0 \;\mathcal{R}\; [a]v_1$ for a fresh a, and so we have $[a]t\{v_0/x\} \;\mathcal{R}\; [a]t\{v_1/x\}$ for all t (by clause (ii)). This property simplifies the congruence proof of $\approx$ with Howe's method.

As in CBN, we can define a big-step version of the bisimulation (where we use evaluation instead of reduction), and bisimilarity contains reduction.

Lemma 3.5 *We have* $\rightarrow^{*}_{\mathrm{v}} \subseteq \approx$.

The applicative bisimulation for CBV is more difficult to use than the one for CBN, as we can see by considering again the terms of Example 2.12.

Example 3.6 Let $v = \lambda x.t$ and $b \notin \mathsf{fn}(v)$; then $[a]v \approx [a]\lambda x.\mu b.[a]v$. To prove clause (i), we consider t' be such that $[b]t' \rightarrow_{\mathrm{v}}^{*} [b]v'$ for $b \notin \mathsf{fn}(t')$; we have to compare $[a]t\{v'\langle{}^{[a]v\,\Box}/_{b}\rangle/x\}$ with $[a]\mu b.[a]v\ t'$. But $[a]\mu b.[a]v\ t' \rightarrow_{\mathrm{v}} [a]v\ t' \rightarrow_{\mathrm{v}}^{*} [a]t\{v'\langle{}^{[a]v\,\Box}/_{b}\rangle/x\}$, therefore we can conclude with Lemma 3.5.

For clause (ii), we have to relate $[a]t'\{v/y\}$ and $[a]t'\{\lambda x.\mu b.[a]v'\ v/y\}$ for all $v' = \lambda y.t'$. We proceed by case analysis on t'; the most interesting case is $t' = E[y\ v'']$. In this case, we have $[a]t'\{\lambda x.\mu b.[a]v'\ v/y\} \rightarrow_{\mathrm{v}}^{*} [a]v'\ v \rightarrow_{\mathrm{v}} [a]t'\{v/y\}$, therefore we can conclude with Lemma 3.5. To handle all the possible cases, we prove in Appendix B.1 that $\{(u\{v/y\}, u\{\lambda x.\mu b.[a]t_0/y\}) \mid [a]t_0 \rightarrow_{\mathrm{v}}^{*} u\{v/y\}\} \cup \approx$ is an applicative bisimulation.

In the next example, we give two terms that can be proved equivalent with applicative bisimilarity but not with eager normal form bisimilarity [19].

Example 3.7 Let $u_0 \overset{\text{def}}{=} [b]\lambda xy.\Omega$, $v \overset{\text{def}}{=} \lambda y.\mu a.[b]\lambda x.y$, and $u_1 \overset{\text{def}}{=} [b]\lambda xy.\Theta_{\mathsf{v}}\ v\ y$, where $\Theta_{\mathsf{v}} \overset{\text{def}}{=} (\lambda xy.y\ (\lambda z.xx\ y\ z))\ (\lambda xy.y\ (\lambda z.xx\ y\ z))$ is Turing's call-by-value fixed-point combinator. For u_0 and u_1 to be normal form bisimilar, we need $[c]\Omega$ to be related to $[c]\Theta_{\mathsf{v}}\, v\, y$ for a fresh c, but $[c]\Theta_{\mathsf{v}}\, v\, y \Downarrow_{\mathrm{v}} [b]\lambda y.\Theta_{\mathsf{v}} v\, y$ and $[c]\Omega \Uparrow_{\mathrm{v}}$. In contrast, we can prove that $u_0 \approx u_1$ (see Appendix B.1).

We now briefly sketch the proofs of soundness and completeness; more details can be found in Appendix B.2. The application of Howe's method is easier than in CBN because, as already pointed out, an applicative bisimulation (and, therefore, the applicative bisimilarity) is already a congruence for regular values by definition. What is left to prove is congruence for (named) terms. We use the same definitions of compatible refinement and Howe's closure $\approx^{\bullet}$ as in CBN. However, because $\approx$ is a congruence for values, we can prove directly that the restriction of $\approx^{\bullet}$ to closed terms (written $(\approx^{\bullet})^{c}$) is an applicative bisimulation, without having to prove a pseudo-simulation lemma (similar to Lemma 2.14) beforehand.

Lemma 3.8 *The relation $(\approx^{\bullet})^{c}$ is an applicative bisimulation.*

As in CBN, we can conclude that $(\approx^{\bullet})^{c} {=} \approx$, and therefore $\approx$ is a congruence. We can then deduce that $\approx$ is sound w.r.t. $\approx_{c}$. For the reverse inclusion, we use an alternate definition of contextual equivalence where we test terms with evaluation contexts (see Definition 2.16), and we prove it is an applicative bisimulation. As a result, $\approx$ coincides with $\approx_{c}$.

Theorem 3.9 $\approx {=} \approx_{c}$.

Remark 3.10 In [9], Koutavas *et al.* show that applicative bisimilarity cannot be sound in a CBV λ-calculus with exceptions, a mechanism that can be seen as a form of control. Our work agrees with their conclusions, as their definition of applicative bisimilarity compares λ-abstractions by applying them to values only, and Example 3.2 shows that it is indeed not sufficient.

3.3 Examples

Even if applicative bisimulation for CBV is difficult to use, we can still prove some equivalences with it. Here we give some examples inspired from Sabry and Felleisen's axiomatization of call/cc [16]. Given a name a, we write $a^\dagger$ for the term $\lambda x.\mu b.[a]x$, and we encode call/cc into $\lambda x.\mu a.[a]x\ a^\dagger$. Given a named context $\mathbb{E}$, we also write $\mathbb{E}^\dagger$ for $\lambda x.\mu b.\mathbb{E}[x]$, where $b \notin \mathsf{fn}(\mathbb{E})$. The first example is the axiom C_{tail} of [16], where call/cc is exchanged with a λ-abstraction.

Example 3.11 If $y \notin \mathsf{fv}(t_1)$ and b is fresh, then $[b](\lambda x.\mu a.[a]x\ a^\dagger)\ (\lambda y.(\lambda z.t_0)\ t_1) \approx [b](\lambda z.(\lambda x.\mu a.[a]x\ a^\dagger)\ (\lambda y.t_0))\ t_1$.

Proof. Let $v_0 \overset{\text{def}}{=} \lambda z.t_0\{b^\dagger/y\}$ and $v_1 \overset{\text{def}}{=} \lambda z.(\lambda x.\mu a.[a]x\, a^\dagger)\,(\lambda y.t_0)$. The term on the left reduces to $[b]v_0\, t_1$, so we relate this term to the one the right, i.e., $[b]v_1\, t_1$. We distinguish several cases depending on t_1. Let c be a fresh name. If $[c]t_1 \Downarrow_{\mathrm{v}} [c]v$, then $[b]v_0\, t_1 \rightarrow^*_{\mathrm{v}} [b]t_0\{b^\dagger/y\}\{v\langle {}^{[b]v_0\,\Box}/_{c}\rangle/z\}$ and $[b]v_1\, t_1 \rightarrow^*_{\mathrm{v}} [b]t_0\{b^\dagger/y\}\{v\langle {}^{[b]v_1\,\Box}/_{c}\rangle/z\}$; because c is fresh, it does not occur in t_0, and the previous terms can be written $u\langle {}^{[b]v_0\,\Box}/_{c}\rangle$ and $u\langle {}^{[b]v_1\,\Box}/_{c}\rangle$ with $u \overset{\text{def}}{=} [b]t_0\{b^\dagger/y\}\{v/z\}$.

Similarly, if $[c]t_1 \Downarrow_{\mathrm{v}} [d]v$ with $c \neq d$, then $[b]v_0\, t_1 \Downarrow_{\mathrm{v}} [d]v\langle {}^{[b]v_0\,\Box}/_{c}\rangle$ and $[b]v_1\, t_1 \Downarrow_{\mathrm{v}} [d]v\langle {}^{[b]v_1\,\Box}/_{c}\rangle$. When testing these two values with clauses (i) and (ii), we obtain each time terms of the form $u\langle {}^{[b]v_0\,\Box}/_{c}\rangle$ and $u\langle {}^{[b]v_1\,\Box}/_{c}\rangle$ for some u. With this reasoning, we can prove that $\{(u\langle {}^{[b]v_0\,\Box}/_{c}\rangle, u\langle {}^{[b]v_1\,\Box}/_{c}\rangle) \mid u \in U^0\}$ is an applicative bisimulation, by case analysis on u. □

With the next example and congruence of $\approx$, we can prove the axiom C_{abort}.

Example 3.12 Let $a \neq b$; we have $[b]E[a^\dagger\ t] \approx [b]a^\dagger\ t$.

Proof. We prove that the relation $\mathcal{R} \overset{\text{def}}{=} \{(u\langle {}^{[b]E[\mathbb{E}^\dagger\,\Box]}/_{c}\rangle, u\langle {}^{[b]\mathbb{E}^\dagger\,\Box}/_{c}\rangle) \mid u \in U^0\}$ is an applicative bisimulation by case analysis on u. For example, if $u = [c]t$ and $u \Downarrow_{\mathrm{v}} [c]v$, then $u\langle {}^{[b]E[\mathbb{E}^\dagger\,\Box]}/_{c}\rangle \rightarrow^*_{\mathrm{v}} [b]E[\mathbb{E}^\dagger\ v\langle {}^{[b]E[\mathbb{E}^\dagger\,\Box]}/_{c}\rangle] \rightarrow_{\mathrm{v}} \mathbb{E}[v\langle {}^{[b]E[\mathbb{E}^\dagger\,\Box]}/_{c}\rangle]$ and $u\langle {}^{[b]\mathbb{E}^\dagger\,\Box}/_{c}\rangle \rightarrow^*_{\mathrm{v}} [b]\mathbb{E}^\dagger\ v\langle {}^{[b]\mathbb{E}^\dagger\,\Box}/_{c}\rangle \rightarrow_{\mathrm{v}} \mathbb{E}[v\langle {}^{[b]\mathbb{E}^\dagger\,\Box}/_{c}\rangle]$. If $\mathbb{E} \neq [d]\Box$ for all d, then the resulting terms are in $\mathcal{R}$, otherwise we get two named values; when checking clauses (i) and (ii), we obtain terms of the form $u'\langle {}^{[b]E[\mathbb{E}'^\dagger\,\Box]}/_{c}\rangle$ and $u'\langle {}^{[b]\mathbb{E}'^\dagger\,\Box}/_{c}\rangle$ that are in $\mathcal{R}$. The remaining cases are similar. □

Example 3.13 [axiom C_{lift}] We have $[b]E[(\lambda x.\mu a.[a]x\ a^\dagger)\ t] \approx [b]E[t\ (\lambda x.b^\dagger\ E[x])]$.

Proof. In this proof, we use an intermediary result, proved in Appendix B.1: if $\mathbb{E} = \mathbb{E}_0[E_1]$, then $\mathbb{E}^\dagger \approx \lambda x.\mathbb{E}_0^\dagger\ (E_1[x])$. The proof of the axiom itself is by case analysis on t. An interesting case is when $[d]t \Downarrow_{\mathrm{v}} [d]\lambda y.t'$ where $d \notin \mathsf{fn}(t)$. Then $[b]E[(\lambda x.\mu a.[a]x\ a^\dagger)\ t] \rightarrow^*_{\mathrm{v}} [b]E[(\lambda x.\mu a.[a]x\ a^\dagger)\ \lambda y.t'\langle {}^{[b]E[(\lambda x.\mu a.[a]x\ a^\dagger)\ \Box]}/_{d}\rangle] \rightarrow^*_{\mathrm{v}} [b]E[t'\langle {}^{[b]E[(\lambda x.\mu a.[a]x\ a^\dagger)\ \Box]}/_{d}\rangle\{\mathbb{E}^\dagger/y\}]$ (with $\mathbb{E} = [b]E$), and $[b]E[t\ (\lambda x.b^\dagger\ E[x])] \rightarrow^*_{\mathrm{v}} [b]E[t'\langle {}^{[b]E[\Box\ (\lambda x.b^\dagger\ E[x])]}/_{d}\rangle\{\lambda x.b^\dagger\ E[x]/y\}]$. From the intermediary result, and because $\approx$ is a congruence, we know that $[b]E[t'\{\mathbb{E}^\dagger/y\}] \approx [b]E[t'\{\lambda x.b^\dagger\ E[x]/y\}]$. Hence, to conclude the proof, one can show that

$$\{(u_0\langle {}^{\mathbb{E}[(\lambda x.\mu a.[a]x\ a^\dagger)\ \Box]}/_{d}\rangle, u_1\langle {}^{\mathbb{E}[\Box\ (\lambda x.\mathbb{E}_0^\dagger\ E_1[x])]}/_{d}\rangle) \ \mid\ u_0 \approx u_1, \mathbb{E} = \mathbb{E}_0[E_1]\}$$

is an applicative bisimulation. □

4 Conclusion

In this work we propose a definition of applicative bisimilarity for CBN and CBV $\lambda\mu$-calculus. Even if the two definitions seem quite different, they follow the same principles. First, we believe it is essential for completeness to hold to relate primarily named terms, and then extend the definition to all terms, as explained when discussing Lassen's definition of applicative bisimilarity (Section 2.5). The top-level names allow to keep track of how the top level is captured and manipulated in the compared terms.

Then, the idea is to test named values with elementary contexts, $[a]\Box\ t$ for CBN, and $[a]\Box\ t$ and $[a]v\ \Box$ for CBV. In the CBV case, we slightly restrict the terms t tested when considering $[a]\Box\ t$, but the resulting definition remains complex to use compared to CBN, as we can see with Examples 2.12 and 3.6. However, we provide counter-examples showing that we cannot simplify it further (see Examples 3.2 and 3.3). In CBV as well as in CBN, applicative bisimilarity is harder to use than eager normal form bisimilarity [19], but our relations are complete characterizations of contextual equivalence, and we can therefore prove equivalences of terms that cannot be related with normal form bisimilarity, such as David and Py's example (see Example 2.20) and Example 3.7. To prove the equivalence between two given $\lambda\mu$-terms, one should start with the bisimulation of [19], and if it fails, try next our applicative (or environmental [3]) bisimulations.

We believe the relations we define remain complete w.r.t. contextual equivalence in other variants of the $\lambda\mu$-calculus (perhaps with some slight variations), such as $\lambda\mu$ with different reduction semantics (like, e.g., in [4]), typed $\lambda\mu$-calculus [15], or de Groote's extended calculus ($\Lambda\mu$-calculus [5]). However, any direct implications of this work for other calculi for abortive continuations such as the syntactic theory of control [6] are unclear and remain to be investigated. The reason is that our approach hinges on the syntactic notion of names, unique to the $\lambda\mu$-calculus, that allows one to keep track of the whereabouts of the top level.

Acknowledgment

We thank Wojciech Jedynak and the anonymous referees for helpful comments on the presentation of this work. The first author has been supported by the Polish NCN grant number DEC-011/03/B/ST6/00348.

References

[1] S. Abramsky. The lazy lambda calculus. In D. A. Turner, editor, *Research Topics in Functional Programming*, pages 65–116. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1990.

[2] D. Biernacki and S. Lenglet. Applicative bisimulations for delimited-control operators. In L. Birkedal, editor, *FOSSACS'12*, number 7213 in LNCS, pages 119–134, Tallinn, Estonia, Mar. 2012. Springer-Verlag.

[3] D. Biernacki and S. Lenglet. Sound and complete bisimilarities for call-by-name and call-by-value $\lambda\mu$-calculus. Research report RR-8447, Inria, Nancy, France, Jan. 2014. Available at http://hal.inria.fr/hal-00926100.

[4] R. David and W. Py. $\lambda\mu$-calculus and Böhm's theorem. *Journal of Symbolic Logic*, 66(1):407–413, 2001.

[5] P. de Groote. On the relation between the $\lambda\mu$-calculus and the syntactic theory of sequential control. In F. Pfenning, editor, *LPAR'94*, number 822 in LNAI, pages 31–43, Kiev, Ukraine, July 1994. Springer-Verlag.

[6] M. Felleisen and R. Hieb. The revised report on the syntactic theories of sequential control and state. *Theoretical Computer Science*, 103(2):235–271, 1992.

[7] A. D. Gordon. Bisimilarity as a theory of functional programming. *Theoretical Computer Science*, 228(1-2):5–47, 1999.

[8] D. J. Howe. Proving congruence of bisimulation in functional programming languages. *Information and Computation*, 124(2):103–112, 1996.

[9] V. Koutavas, P. B. Levy, and E. Sumii. From applicative to environmental bisimulation. *Electronic Notes in Theoretical Computer Science*, 276:215–235, 2011.

[10] S. B. Lassen. Bisimulation for pure untyped $\lambda\mu$-caluclus (extended abstract). Unpublished note, Jan. 1999.

[11] S. B. Lassen. Eager normal form bisimulation. In P. Panangaden, editor, *LICS'05*, pages 345–354, Chicago, IL, June 2005. IEEE Computer Society Press.

[12] S. B. Lassen. Head normal form bisimulation for pairs and the $\lambda\mu$-calculus. In R. Alur, editor, *LICS'06*, pages 297–306, Seattle, WA, Aug. 2006. IEEE Computer Society Press.

[13] J. H. Morris. *Lambda Calculus Models of Programming Languages*. PhD thesis, Massachusets Institute of Technology, 1968.

[14] C.-H. L. Ong and C. A. Stewart. A Curry-Howard foundation for functional computation with control. In N. D. Jones, editor, *POPL*, pages 215–227, Paris, France, Jan. 1997. ACM Press.

[15] M. Parigot. $\lambda\mu$-calculus: an algorithmic interpretation of classical natural deduction. In A. Voronkov, editor, *LPAR'92*, number 624 in LNAI, pages 190–201, St. Petersburg, Russia, July 1992. Springer-Verlag.

[16] A. Sabry and M. Felleisen. Reasoning about programs in continuation-passing style. *Lisp and Symbolic Computation*, 6(3/4):289–360, 1993.

[17] D. Sangiorgi. The lazy lambda calculus in a concurrency scenario. In A. Scedrov, editor, *LICS'92*, pages 102–109, Santa Cruz, California, June 1992. IEEE Computer Society.

[18] D. Sangiorgi, N. Kobayashi, and E. Sumii. Environmental bisimulations for higher-order languages. *ACM Transactions on Programming Languages and Systems*, 33(1):1–69, Jan. 2011.

[19] K. Støvring and S. B. Lassen. A complete, co-inductive syntactic theory of sequential control and state. In M. Felleisen, editor, *POPL'07*, SIGPLAN Notices, Vol. 42, No. 1, pages 161–172, Nice, France, Jan. 2007. ACM Press.

A Call-by-Name $\lambda\mu$-calculus

A.1 Soundness and Completeness of Applicative Bisimilarity

Let $(\approx^\bullet)^c$ be the restriction of $\approx^\bullet$ to closed terms.

Lemma A.1 *If $t_0 \approx^\bullet t_1$, then there exists a substitution σ which closes t_0 and t_1 such that $t_0\sigma \ (\approx^\bullet)^c \ t_1\sigma$, and the size of the derivation of $t_0\sigma \ (\approx^\bullet)^c \ t_1\sigma$ is equal to the size of the derivation of $t_0 \approx^\bullet t_1$. A similar result holds if $u_0 \approx^\bullet u_1$.*

Proof. As usual. □

Lemma A.2 *Let $t_0 \ (\approx^\bullet)^c \ t_1$, and $a \notin \mathsf{fn}(t_0) \cup \mathsf{fn}(t_1)$.*

- *If* $[a]t_0 \to_{\mathrm{n}} u_0$*, then* $[a]t_1 \to_{\mathrm{n}}^* u_1$ *and* $u_0 \, (\approx^\bullet)^c \, u_1$*.*
- *If* $t_0 = \lambda x.t'_0$*, then* $[a]t_1 \to_{\mathrm{n}}^* [a]\lambda x.t'_1$ *and for all* $t''_0 \, (\approx^\bullet)^c \, t''_1$*, we have* $[a]t'_0\{t''_0/x\} \, (\approx^\bullet)^c \, [a]t'_1\langle [a]\Box\, t''_1/a\rangle\{t''_1/x\}$*.*

Proof. By induction on $t_0 \, (\approx^\bullet)^c \, t_1$.

Suppose $t_0 \approx t_1$. Then $[a]t_0 \approx [a]t_1$. If $[a]t_0 \to_{\mathrm{n}} u_0$, then by bisimilarity, we have $[a]t_1 \to_{\mathrm{n}}^* u_1$ and $u_0 \approx u_1$, i.e., $u_0 \, (\approx^\bullet)^c \, u_1$, as required.

Suppose $t_0 = \lambda x.t'_0$, and let $t''_0 \, (\approx^\bullet)^c \, t''_1$. By the bisimilarity definition, we have $[a]t_1 \to_{\mathrm{n}}^* [a]\lambda x.t'_1$ and $[a]t'_0\{t''_1/x\} \approx [a]t'_1\langle [a]\Box\, t''_1/a\rangle\{t''_1/x\}$ (using the fact that $a \notin \mathsf{fn}(t'_0)$). From $t''_0 \, (\approx^\bullet)^c \, t''_1$, we deduce $t'_0\{t''_0/x\} \, (\approx^\bullet)^c \, t'_0\{t''_1/x\}$, and then $[a]t'_0\{t''_0/x\} \, (\approx^\bullet)^c \approx [a]t'_1\langle [a]\Box\, t''_1/a\rangle\{t''_1/x\}$, which implies $[a]t'_0\{t''_0/x\} \, (\approx^\bullet)^c \, [a]t'_1\langle [a]\Box\, t''_1/a\rangle\{t''_1/x\}$, as required.

Suppose $t_0 \approx^\bullet t \approx^\circ t_1$, so that t is closed (using Lemma A.1 if necessary). In fact, we have $t_0 \, (\approx^\bullet)^c \, t \approx t_1$. If $[a]t_0 \to_{\mathrm{n}} u_0$, then by the induction hypothesis, there exists u such that $[a]t \to_{\mathrm{n}}^* u$ and $u_0 \, (\approx^\bullet)^c \, u$. By bisimilarity, there exists u_1 such that $[a]t_1 \to_{\mathrm{n}}^* u_1$ and $u \approx u_1$. From $u_0 \, (\approx^\bullet)^c \, u \approx u_1$, we deduce $u_0 \, (\approx^\bullet)^c \, u_1$, as wished.

Suppose $t_0 = \lambda x.t'_0$, and let $t''_0 \, (\approx^\bullet)^c \, t''_1$. By induction, there exists t' such that $[a]t \to_{\mathrm{n}}^* [a]\lambda x.t'$ and $[a]t'_0\{t''_0/x\} \, (\approx^\bullet)^c \, [a]t'\langle [a]\Box\, t''_1/a\rangle\{t''_1/x\}$. By bisimilarity, there exists t'_1 such that $[a]t_1 \to_{\mathrm{n}}^* [a]\lambda x.t'_1$ and $[a]t'\langle [a]\Box\, t''_1/a\rangle\{t''_1/x\} \approx [a]t'_1\langle [a]\Box\, t''_1/a\rangle\{t''_1/x\}$. Hence, we have $[a]t'_0\{t''_0/x\} \, (\approx^\bullet)^c \approx [a]t'\langle [a]\Box\, t''_1/a\rangle\{t''_1/x\}$, i.e., $[a]t'_0\{t''_0/x\} \, (\approx^\bullet)^c \, [a]t'_1\langle [a]\Box\, t''_1/a\rangle\{t''_1/x\}$ as required.

If $t_0 \, \widetilde{(\approx^\bullet)^c} \, t_1$, then we have several cases to consider.

Suppose $t_0 = \lambda x.t'_0$ and $t_1 = \lambda x.t'_1$ with $t'_0 \approx^\bullet t'_1$. Let $t''_0 \, (\approx^\bullet)^c \, t''_1$. We have $[a]t'_0\{t''_0/x\} \, (\approx^\bullet)^c \, [a]t'_1\{t''_1/x\}$, hence the result holds (note that $a \notin \mathsf{fv}(t'_1)$ because $a \notin \mathsf{fv}(t_1)$).

Suppose $t_0 = \mu b.u_0$ and $t_1 = \mu b.u_1$ with $u_0 \, (\approx^\bullet)^c \, u_1$. We have $[a]t_0 \to_{\mathrm{n}} u_0\langle [a]\Box/b\rangle$, $[a]t_1 \to_{\mathrm{n}} u_1\langle [a]\Box/b\rangle$, and $u_0\langle [a]\Box/b\rangle \, (\approx^\bullet)^c \, u_1\langle [a]\Box/b\rangle$, hence the result holds.

Suppose $t_0 = t_0^1\, t_0^2$, $t_1 = t_1^1\, t_1^2$ with $t_0^1 \, (\approx^\bullet)^c \, t_1^1$ and $t_0^2 \, (\approx^\bullet)^c \, t_1^2$. We distinguish two cases.

- If $[b]t_0^1 \to_{\mathrm{n}} u_0$ (for some fresh b), then $[a]t_0 \to_{\mathrm{n}} u_0\langle [a]\Box\, t_0^2/b\rangle$. By the induction hypothesis, there exists u_1 such that $[b]t_1^1 \to_{\mathrm{n}}^* u_1$ and $u_0 \, (\approx^\bullet)^c \, u_1$. Consequently, we have $[a]t_1 \to_{\mathrm{n}}^* u_1\langle [a]\Box\, t_1^2/b\rangle$, and by definition of $\approx^\bullet$, we have $u_0\langle [a]\Box\, t_0^2/b\rangle \, (\approx^\bullet)^c \, u_1\langle [a]\Box\, t_1^2/b\rangle$, as required.
- If $t_0^1 = \lambda x.t'_0$, then we have $[a]t_0 \to_{\mathrm{n}} [a]t'_0\{t_0^2/x\}$. By the induction hypothesis, there exists t'_1 such that $[a]t_1^1 \to_{\mathrm{n}}^* [a]\lambda x.t'_1$ and $[a]t'_0\{t_0^2/x\} \, (\approx^\bullet)^c \, [a]t'_1\langle [a]\Box\, t_1^2/a\rangle\{t_1^2/x\}$. From $[a]t_1^1 \to_{\mathrm{n}}^* [a]\lambda x.t'_1$, we deduce

$$[a]t_1 \to_{\mathrm{n}}^* [a](\lambda x.t'_1\langle [a]\Box\, t_1^2/a\rangle)t_1^2 \to_{\mathrm{n}} [a]t'_1\langle [a]\Box\, t_1^2/a\rangle\{t_1^2/x\},$$

  hence the result holds.

Suppose $t_0 = t_0'\langle{}^{\mathbb{E}_0}/_{b}\rangle$, $t_1 = t_1'\langle{}^{\mathbb{E}_1}/_{b}\rangle$ with $t_0' \; (\approx^\bullet)^c \; t_1'$, $\mathbb{E}_0 \; (\approx^\bullet)^c \; \mathbb{E}_1$. If $[a]t_0 \rightarrow_{\mathrm{n}} u_0$, then in fact $[a]t_0' \rightarrow_{\mathrm{n}} u_0'$ with $u_0 = u_0'\langle{}^{\mathbb{E}_0}/_{b}\rangle$. By the induction hypothesis, there exists u_1' such that $[a]t_1' \rightarrow_{\mathrm{n}}^* u_1'$ and $u_0' \; (\approx^\bullet)^c \; u_1'$. Consequently, we have $[a]t_1 \rightarrow_{\mathrm{n}}^* u_1'\langle{}^{\mathbb{E}_1}/_{b}\rangle$, and $u_0'\langle{}^{\mathbb{E}_0}/_{b}\rangle \; (\approx^\bullet)^c \; u_1'\langle{}^{\mathbb{E}_1}/_{b}\rangle$ holds, as wished. If $[a]t_0$ is a named value, then in fact $t_0' = \lambda x.t_0'''$ and $t_0 = \lambda x.t_0'''\langle{}^{\mathbb{E}_0}/_{b}\rangle$. Let $t_0'' \; (\approx^\bullet)^c \; t_1''$. Then $t_0''\langle{}^{[c]\Box}/_{b}\rangle \; (\approx^\bullet)^c \; t_1''\langle{}^{[c]\Box}/_{b}\rangle$ for a fresh c. By the induction hypothesis, there exists t_1''' such that $[a]t_1' \rightarrow_{\mathrm{n}}^* [a]\lambda x.t_1'''$ and

$$[a]t_0'''\{t_0''\langle{}^{[c]\Box}/_{b}\rangle/x\} \; (\approx^\bullet)^c \; [a]t_1'''\langle{}^{[a]\Box\; t_1''\langle{}^{[c]\Box}/_{b}\rangle}/_{a}\rangle\{t_1''\langle{}^{[c]\Box}/_{b}\rangle/x\}.$$

Therefore, we have $[a]t_1 \rightarrow_{\mathrm{n}}^* [a]\lambda x.t_1'''\langle{}^{\mathbb{E}_1}/_{b}\rangle$, and

$$[a]t_0'''\{t_0''\langle{}^{[c]\Box}/_{b}\rangle/x\}\langle{}^{\mathbb{E}_0}/_{b}\rangle \; (\approx^\bullet)^c \; [a]t_1'''\langle{}^{[a]\Box\; t_1''\langle{}^{[c]\Box}/_{b}\rangle}/_{a}\rangle\{t_1''\langle{}^{[c]\Box}/_{b}\rangle/x\}\langle{}^{\mathbb{E}_1}/_{b}\rangle.$$

Because b does not occur in $t_0''\langle{}^{[c]\Box}/_{b}\rangle$, $t_1''\langle{}^{[c]\Box}/_{b}\rangle$ thanks to the renaming to a fresh c, we can switch the substitutions around, and in fact

$$[a]t_0'''\langle{}^{\mathbb{E}_0}/_{b}\rangle\{t_0''\langle{}^{[c]\Box}/_{b}\rangle/x\} \; (\approx^\bullet)^c \; [a]t_1'''\langle{}^{\mathbb{E}_1}/_{b}\rangle\langle{}^{[a]\Box\; t_1''\langle{}^{[c]\Box}/_{b}\rangle}/_{a}\rangle\{t_1''\langle{}^{[c]\Box}/_{b}\rangle/x\}$$

holds. Renaming c back into b, we obtain

$$[a]t_0'''\langle{}^{\mathbb{E}_0}/_{b}\rangle\{t_0''/x\} \; (\approx^\bullet)^c \; [a]t_1'''\langle{}^{\mathbb{E}_1}/_{b}\rangle\langle{}^{[a]\Box\; t_1''}/_{a}\rangle\{t_1''/x\},$$

which gives us the required result.

□

Lemma A.3 *Let $u_0 \; (\approx^\bullet)^c \; u_1$.*

- *If $u_0 \rightarrow_{\mathrm{n}} u_0'$, then $u_1 \rightarrow_{\mathrm{n}}^* u_1'$ and $u_0' \; (\approx^\bullet)^c \; u_1'$.*
- *If $u_0 = [a]\lambda x.t_0$, then $u_1 \rightarrow_{\mathrm{n}}^* [a]\lambda x.t_1$ and for all $t_0' \; (\approx^\bullet)^c \; t_1'$, we have $[a]t_0\langle{}^{[a]\Box\; t_0'}/_{a}\rangle\{t_0'/x\} \; (\approx^\bullet)^c \; [a]t_1\langle{}^{[a]\Box\; t_1'}/_{a}\rangle\{t_1'/x\}$.*

Proof. By induction on $u_0 \; (\approx^\bullet)^c \; u_1$.

Suppose $u_0 \approx u_1$. The first item holds by bisimilarity. Suppose $u_0 = [a]\lambda x.t_0$, and let $t_0' \; (\approx^\bullet)^c \; t_1'$. By definition of the bisimilarity, we have $u_1 \rightarrow_{\mathrm{n}}^* [a]\lambda x.t_1$ and $[a]t_0\langle{}^{[a]\Box\; t_1'}/_{a}\rangle\{t_1'/x\} \approx [a]t_1\langle{}^{[a]\Box\; t_1'}/_{a}\rangle\{t_1'/x\}$. From $t_0' \; (\approx^\bullet)^c \; t_1'$, we deduce $t_0\langle{}^{[a]\Box\; t_0'}/_{a}\rangle\{t_0'/x\} \; (\approx^\bullet)^c \; t_0\langle{}^{[a]\Box\; t_1'}/_{a}\rangle\{t_1'/x\}$, which implies

$$t_0\langle{}^{[a]\Box\; t_0'}/_{a}\rangle\{t_0'/x\} \; (\approx^\bullet)^c{\approx} \; [a]t_1\langle{}^{[a]\Box\; t_1'}/_{a}\rangle\{t_1'/x\},$$

which in turn implies

$$[a]t_0\langle{}^{[a]\Box\; t_0'}/_{a}\rangle\{t_0'/x\} \; (\approx^\bullet)^c \; [a]t_1\langle{}^{[a]\Box\; t_1'}/_{a}\rangle\{t_1'/x\},$$

as required.

Suppose $u_0 \approx^\bullet u \approx^\circ u_1$, so that u is closed (using Lemma A.1 if necessary). In fact, we have $u_0 \; (\approx^\bullet)^c \; u \approx u_1$. If $u_0 \rightarrow_{\mathrm{n}} u_0'$, then by the induction hypothesis, there exists u' such that $u \to_{\mathrm{n}}^{*} u'$ and $u'_0 \; (\approx^{\bullet})^{c} \; u'$. By bisimilarity, there exists u'_1 such that $u_1 \to_{\mathrm{n}}^{*} u'_1$ and $u' \approx u'_1$. From $u'_0 \; (\approx^{\bullet})^{c} \; u' \approx u'_1$, we deduce $u'_0 \; (\approx^{\bullet})^{c} \; u'_1$, as wished.

Suppose $u_0 = [a]\lambda x.t_0$, and let $t'_0 \; (\approx^{\bullet})^{c} \; t'_1$. By induction, there exists t such that $u \to_{\mathrm{n}}^{*} [a]\lambda x.t$ and $[a]t_0\langle{}^{[a]\Box\, t'_0}\!/_{a}\rangle\{t'_0/x\} \; (\approx^{\bullet})^{c} \; [a]t\langle{}^{[a]\Box\, t'_1}\!/_{a}\rangle\{t'_1/x\}$. By bisimilarity, there exists t_1 such that $u_1 \to_{\mathrm{n}}^{*} [a]\lambda x.t_1$ and $[a]t\langle{}^{[a]\Box\, t'_1}\!/_{a}\rangle\{t'_1/x\} \approx [a]t_1\langle{}^{[a]\Box\, t'_1}\!/_{a}\rangle\{t'_1/x\}$. Consequently, we have

$$[a]t_0\langle{}^{[a]\Box\, t'_0}\!/_{a}\rangle\{t'_0/x\} \; (\approx^{\bullet})^{c} \approx [a]t\langle{}^{[a]\Box\, t'_1}\!/_{a}\rangle\{t'_1/x\},$$

i.e., $[a]t_0\langle{}^{[a]\Box\, t'_0}\!/_{a}\rangle\{t'_0/x\} \; (\approx^{\bullet})^{c} \; [a]t_1\langle{}^{[a]\Box\, t'_1}\!/_{a}\rangle\{t'_1/x\}$ holds as required.

If $u_0 \; \widetilde{(\approx^{\bullet})^{c}} \; u_1$, then we have two cases.

Suppose $u_0 = [a]t_0$ and $u_1 = [a]t_1$ with $t_0 \; (\approx^{\bullet})^{c} \; t_1$. If $a \notin \mathsf{fn}(t_0) \cup \mathsf{fn}(t_1)$, then we can apply Lemma A.2 directly to get the required result. Otherwise, let $b \notin \mathsf{fn}(t_0) \cup \mathsf{fn}(t_1)$. If $u_0 \to_{\mathrm{n}} u'_0$, then $[b]t_0 \to_{\mathrm{n}} u''_0$ and $u'_0 = u'_0\langle{}^{[a]\Box}\!/_{b}\rangle$. We can apply Lemma A.2 to $[b]t_0$ and $[b]t_1$, and then rename b into a.

Suppose $u_0 = u'_0\langle{}^{\mathbb{E}_0}\!/_{a}\rangle$, $u_1 = u'_1\langle{}^{\mathbb{E}_1}\!/_{a}\rangle$ with $u'_0 \; (\approx^{\bullet})^{c} \; u'_1$ and $\mathbb{E}_0 \; (\approx^{\bullet})^{c} \; \mathbb{E}_1$.

If u_0 is a named value, then we distinguish two cases. First, we may have $u'_0 = [a]\lambda x.t_0$, $\mathbb{E}_0 = \mathbb{E}_1 = [b]\Box$, and $u_0 = [b]\lambda x.t_0\langle{}^{\mathbb{E}_0}\!/_{a}\rangle$. Let $t'_0 \; (\approx^{\bullet})^{c} \; t'_1$. Let t''_0, t''_1 be t'_0 and t'_1 where a, b are renamed into fresh c, d to avoid some name clashes (we still have $t''_0 \; (\approx^{\bullet})^{c} \; t''_1$). By the induction hypothesis, there exists t_1 such that $u'_1 \to_{\mathrm{n}}^{*} [a]\lambda x.t_1$ and $[a]t_0\langle{}^{[a]\Box\, t''_0}\!/_{a}\rangle\{t''_0/x\} \; (\approx^{\bullet})^{c} \; [a]t_1\langle{}^{[a]\Box\, t''_1}\!/_{a}\rangle\{t''_1/x\}$. This implies

$$[a]t_0\langle{}^{[a]\Box\, t''_0}\!/_{a}\rangle\{t''_0/x\}\langle{}^{[a]\Box\, t''_0}\!/_{b}\rangle \; (\approx^{\bullet})^{c} \; [a]t_1\langle{}^{[a]\Box\, t''_1}\!/_{a}\rangle\{t''_1/x\}\langle{}^{[a]\Box\, t''_1}\!/_{b}\rangle,$$

which is the same as

$$[a]t_0\langle{}^{[a]\Box\, t''_0}\!/_{a}\rangle\langle{}^{[a]\Box\, t''_0}\!/_{b}\rangle\{t''_0/x\} \; (\approx^{\bullet})^{c} \; [a]t_1\langle{}^{[a]\Box\, t''_1}\!/_{a}\rangle\langle{}^{[a]\Box\, t''_1}\!/_{b}\rangle\{t''_1/x\}$$

because b does not occur in t''_0, t''_1. In turn, we have

$$\begin{aligned}&[b]t_0\langle{}^{[a]\Box\, t''_0}\!/_{a}\rangle\langle{}^{[a]\Box\, t''_0}\!/_{b}\rangle\{t''_0/x\}\langle{}^{\mathbb{E}_0}\!/_{a}\rangle\\ &\qquad\qquad\qquad\qquad (\approx^{\bullet})^{c} \; [b]t_1\langle{}^{[a]\Box\, t''_1}\!/_{a}\rangle\langle{}^{[a]\Box\, t''_1}\!/_{b}\rangle\{t''_1/x\}\langle{}^{\mathbb{E}_1}\!/_{a}\rangle,\end{aligned}$$

which is equal to

$$[b]t_0\langle{}^{\mathbb{E}_0}\!/_{a}\rangle\langle{}^{[b]\Box\, t''_0}\!/_{b}\rangle\{t''_0/x\} \; (\approx^{\bullet})^{c} \; [b]t_1\langle{}^{\mathbb{E}_1}\!/_{a}\rangle\langle{}^{[b]\Box\, t''_1}\!/_{b}\rangle\{t''_1/x\}$$

because a does not occur in t''_0, t''_1. Renaming c, d back into a, b, we obtain

$$[b]t_0\langle{}^{\mathbb{E}_0}\!/_{a}\rangle\langle{}^{[b]\Box\, t'_0}\!/_{b}\rangle\{t'_0/x\} \; (\approx^{\bullet})^{c} \; [b]t_1\langle{}^{\mathbb{E}_1}\!/_{a}\rangle\langle{}^{[b]\Box\, t'_1}\!/_{b}\rangle\{t'_1/x\},$$

and because $u_1 \to_{\mathrm{n}}^{*} [b]\lambda x.t_1\langle{}^{\mathbb{E}_1}\!/_{a}\rangle$, the result holds.

In the second case, we have $u'_0 = [b]\lambda x.t_0$ and $u_0 = [b]\lambda x.t_0\langle {}^{\mathbb{E}_0}/_{a}\rangle$ with $b \neq a$. Let $t'_0 \ (\approx^\bullet)^c \ t'_1$. Let t''_0, t''_1 be t'_0 and t'_1 where a is renamed into a fresh c. By the induction hypothesis, there exists t_1 such that $u'_1 \rightarrow^*_{\mathrm{n}} [b]\lambda x.t_1$ and

$$[b]t_0\langle {}^{[b]\Box\ t''_0}/_{b}\rangle\{t''_0/x\} \ (\approx^\bullet)^c \ [b]t_1\langle {}^{[b]\Box\ t''_1}/_{b}\rangle\{t''_1/x\}.$$

From $\mathbb{E}_0\langle {}^{[b]\Box\ t''_0}/_{b}\rangle \ (\approx^\bullet)^c \ \mathbb{E}_1\langle {}^{[b]\Box\ t''_1}/_{b}\rangle$ and the previous relation, we can deduce

$$[b]t_0\langle {}^{[b]\Box\ t''_0}/_{b}\rangle\{t''_0/x\}\langle {}^{\mathbb{E}_0\langle {}^{[b]\Box\ t''_0}/_{b}\rangle}/_{a}\rangle \ (\approx^\bullet)^c \ [b]t_1\langle {}^{[b]\Box\ t''_1}/_{b}\rangle\{t''_1/x\}\langle {}^{\mathbb{E}_1\langle {}^{[b]\Box\ t''_1}/_{b}\rangle}/_{a}\rangle.$$

Because a does not occur in t''_0, t''_1, this can be rewritten into

$$[b]t_0\langle {}^{\mathbb{E}_0}/_{a}\rangle\langle {}^{[b]\Box\ t''_0}/_{b}\rangle\{t''_0/x\} \ (\approx^\bullet)^c \ [b]t_1\langle {}^{\mathbb{E}_1}/_{a}\rangle\langle {}^{[b]\Box\ t''_1}/_{b}\rangle\{t''_1/x\}.$$

By renaming c back into a, we obtain

$$[b]t_0\langle {}^{\mathbb{E}_0}/_{a}\rangle\langle {}^{[b]\Box\ t'_0}/_{b}\rangle\{t'_0/x\} \ (\approx^\bullet)^c \ [b]t_1\langle {}^{\mathbb{E}_1}/_{a}\rangle\langle {}^{[b]\Box\ t'_1}/_{b}\rangle\{t'_1/x\},$$

and because $u_1 \rightarrow^*_{\mathrm{n}} [b]\lambda x.t_1\langle {}^{\mathbb{E}_1}/_{a}\rangle$, the result holds.

If $u_0 \rightarrow_{\mathrm{n}}$, then again we distinguish two cases. First, suppose $u'_0 \rightarrow_{\mathrm{n}} u''_0$; then $u_0 \rightarrow_{\mathrm{n}} u''_0\langle {}^{E_0}/_{a}\rangle$. By the induction hypothesis, there exists u''_1 such that $u'_1 \rightarrow^*_{\mathrm{n}} u''_1$ and $u''_0 \ (\approx^\bullet)^c \ u''_1$. Then $u_1 \rightarrow^*_{\mathrm{n}} u''_1\langle {}^{\mathbb{E}_1}/_{a}\rangle$ and $u''_0\langle {}^{\mathbb{E}_0}/_{a}\rangle \ (\approx^\bullet)^c \ u''_1\langle {}^{\mathbb{E}_1}/_{a}\rangle$, hence the result holds.

Otherwise, $u'_0 = [a]\lambda x.t_0$, $\mathbb{E}_0 = \mathbb{E}'_0[\Box\ t'_0]$, and $u_0 \rightarrow_{\mathrm{n}} \mathbb{E}'_0[t_0\langle {}^{\mathbb{E}_0}/_{a}\rangle\{t'_0/x\}]$. We can prove by induction on $\mathbb{E}_0 \ (\approx^\bullet)^c \ \mathbb{E}_1$ that $\mathbb{E}_1 = \mathbb{E}'_1[\Box\ t'_1]$ with $\mathbb{E}'_0 \ (\approx^\bullet)^c \ \mathbb{E}'_1$ and $t'_0 \ (\approx^\bullet)^c \ t'_1$. Let $\mathbb{E}''_i$, t''_i be $\mathbb{E}'_i$, t'_i with a renamed into a fresh b. By the induction hypothesis, there exists t_1 such that $u'_1 \rightarrow^*_{\mathrm{n}} [a]\lambda x.t_1$, and

$$[a]t_0\langle {}^{[a]\Box\ t''_0}/_{a}\rangle\{t''_0/x\} \ (\approx^\bullet)^c \ [a]t_1\langle {}^{[a]\Box\ t''_1}/_{a}\rangle\{t''_1/x\}.$$

This implies

$$([a]t_0\langle {}^{[a]\Box\ t''_0}/_{a}\rangle\{t''_0/x\})\langle {}^{\mathbb{E}''_0}/_{a}\rangle \ (\approx^\bullet)^c \ ([a]t_1\langle {}^{[a]\Box\ t''_1}/_{a}\rangle\{t''_1/x\})\langle {}^{\mathbb{E}''_1}/_{a}\rangle,$$

i.e., $\mathbb{E}''_0[t_0\langle {}^{\mathbb{E}''_0[\Box\ t''_0]}/_{a}\rangle\{t''_0/x\}] \ (\approx^\bullet)^c \ \mathbb{E}''_1[t_1\langle {}^{\mathbb{E}''_1[\Box\ t''_1]}/_{a}\rangle\{t''_1/x\}]$ because a does not occurs in t''_0, t''_1. Renaming b into a, we obtain

$$\mathbb{E}'_0[t_0\langle {}^{\mathbb{E}_0}/_{a}\rangle\{t'_0/x\}] \ (\approx^\bullet)^c \ \mathbb{E}'_1[t_1\langle {}^{\mathbb{E}_1}/_{a}\rangle\{t'_1/x\}],$$

and because $u_1 \rightarrow^*_{\mathrm{n}} \mathbb{E}'_1[t_1\langle {}^{\mathbb{E}_1}/_{a}\rangle\{t'_1/x\}]$, we have the required result.

□

From there, we can prove that $(\approx^\bullet)^c{=}\approx$ using the usual techniques [7], and then we deduce soundness of $\approx$. To prove completeness, we show that $\dot{\approx}_c$ is an applicative bisimulation.

Lemma A.4 *The relation $\dot{\approx}_c$ is a big step applicative bisimulation.*

Proof. Suppose $u_0 \dot{\approx}_c u_1$. If $u_0 \Downarrow_{\mathrm{n}} [a]\lambda x.t_0$, then $u_1 \Downarrow_{\mathrm{n}} [a]\lambda x.t_1$. We have $[a](\mu a.u_0)\ t \rightarrow_{\mathrm{n}}^{*} [a]t_0\langle{}^{[a]\Box\ t}/_{a}\rangle\{t/x\}$, $[a](\mu a.u_1)\ t \rightarrow_{\mathrm{n}}^{*} [a]t_1\langle{}^{[a]\Box\ t}/_{a}\rangle\{t/x\}$, but also $[a](\mu a.u_0)\ t \dot{\approx}_c [a](\mu a.u_1)\ t$. From $\rightarrow_{\mathrm{n}}^{*} \subseteq \approx \subseteq \dot{\approx}_c$, we have $[a]t_0\langle{}^{[a]\Box\ t}/_{a}\rangle\{t/x\} \dot{\approx}_c [a]t_1\langle{}^{[a]\Box\ t}/_{a}\rangle\{t/x\}$ as wished. □

A.2 David and Py's Counter-Example

Lemma A.5 *Let* $0 \stackrel{def}{=} \lambda x.\lambda y.y$, $1 \stackrel{def}{=} \lambda x.\lambda y.x$, *and* $t_a \stackrel{def}{=} \mu c.[a]0$. *Then we have* $\lambda x.\mu a.[a]x\ \mu b.[a]x\ t_a\ 0 \approx \lambda x.\mu a.[a]x\ \mu b.[a]x\ t_a\ 1$.

Proof. We fix a name c, and for all t, we want to relate $[c]\mu a.[a]t\ \mu b.[a]t\ t_a\ 0$ and $[c]\mu a.[a]t\ \mu b.[a]t\ t_a\ 1$, which reduce respectively to $[c]t\ \mu b.[c]t\ t_c\ 0$ and $[c]t\ \mu b.[c]t\ t_c\ 1$. Let $s_0^t \overset{\text{def}}{=} \mu b.[c]t\ t_c\ 0$, $s_1^t \overset{\text{def}}{=} \mu b.[c]t\ t_c\ 1$, $\mathbb{E}_0^t \overset{\text{def}}{=} [c]\Box\ s_0^t$, and $\mathbb{E}_1^t \overset{\text{def}}{=} [c]\Box\ s_1^t$. We define a relation $[d]t \leadsto^k u$ as $[d]t \rightarrow_{\mathrm{n}}^{*} u$ if $k = 0$ and as $[d]t \Downarrow_{\mathrm{n}} [d]\lambda x_1.t_1$, $[c]t_1 \Downarrow_{\mathrm{n}} [d]\lambda x_2.t_2$, $\dots [c]t_k \rightarrow_{\mathrm{n}}^{*} u$ for some $t_1 \dots t_k$ if $k > 0$. The rationale behind this relation appears in the proof. Note that if $[d]t \leadsto^k u$, then $\mathsf{fv}(u) \subseteq \{x_1, \dots x_k\}$, and $[d]t\langle{}^{\mathbb{E}}/_{e}\rangle \leadsto^k u\langle{}^{\mathbb{E}}/_{e}\rangle$ for all $\mathbb{E}$ and $e \neq d$. We define $\mathcal{R}$ as

$$\begin{aligned}
&\{(u\langle{}^{\mathbb{E}_0^t}/_{d}\rangle\{s_0^t/x_i\}_{i=1}^k, u\langle{}^{\mathbb{E}_1^t}/_{d}\rangle\{s_1^t/x_i\}_{i=1}^k) \mid u \in U, t \in T^0, d \notin \mathsf{fn}(t), [d]t \leadsto^k u\}\\
\cup\ &\{([c]t\ t_c\ 0, [c]t\ t_c\ 1) \mid t \in T^0, d \notin \mathsf{fn}(t), [d]t \leadsto^k \mathbb{E}[x_i], k \geq 1, 1 \leq i \leq k\}\\
\cup\ &\{(u, u) \mid u \in U^0\}
\end{aligned}$$

where $t\{t_i/x_i\}_{i=1}^k$ stands for $t\{t_1/x_1\} \dots \{t_k/x_k\}$, and we show that $\mathcal{R}$ is an applicative bisimulation.

Let $u\langle{}^{\mathbb{E}_0^t}/_{d}\rangle\{s_0^t/x_i\}_{i=1}^k\ \mathcal{R}\ u\langle{}^{\mathbb{E}_1^t}/_{d}\rangle\{s_1^t/x_i\}_{i=1}^k$. If $u \rightarrow_{\mathrm{n}} u'$, then it is easy to conclude. Otherwise, we distinguish several cases.

If $u = [e]\lambda y.t_0$ with $e \neq d$, then $[e]t_0\langle{}^{[e]\Box\ t'}/_{e}\rangle\{t'/y\}\langle{}^{\mathbb{E}_0^{t''}}/_{d}\rangle\{s_0^{t''}/x_i\}_{i=1}^k\ \mathcal{R}\ [e]t_0\langle{}^{[e]\Box\ t'}/_{e}\rangle\{t'/y\}\langle{}^{\mathbb{E}_1^{t''}}/_{d}\rangle\{s_1^{t''}/x_i\}_{i=1}^k$ (where $t'' = t\langle{}^{[e]\Box\ t'}/_{e}\rangle$) for all t', because the relation $[d]t \leadsto^k [e]\lambda y.t_0$ implies $[d]t'' \leadsto^k [e]t_0\langle{}^{[e]\Box\ t'}/_{e}\rangle\{t'/y\}$.

If $u = [d]\lambda x_{k+1}.t_{k+1}$, then we have the reduction $u\langle{}^{\mathbb{E}_0^t}/_{d}\rangle\{s_0^t/x_i\}_{i=1}^k \rightarrow_{\mathrm{n}} [c]t_{k+1}\{s_0^t/x_{k+1}\}\langle{}^{\mathbb{E}_0^t}/_{d}\rangle\{s_0^t/x_i\}_{i=1}^k$ as well as the reduction $u\langle{}^{\mathbb{E}_1^t}/_{d}\rangle\{s_1^t/x_i\}_{i=1}^k \rightarrow_{\mathrm{n}} [c]t_{k+1}\{s_1^t/x_{k+1}\}\langle{}^{\mathbb{E}_1^t}/_{d}\rangle\{s_1^t/x_i\}_{i=1}^k$. We obtain terms in $\mathcal{R}$ because we can rewrite them into, respectively, $[c]t_{k+1}\langle{}^{\mathbb{E}_0^t}/_{d}\rangle\{s_0^t/x_i\}_{i=1}^{k+1}$ and $[c]t_{k+1}\langle{}^{\mathbb{E}_1^t}/_{d}\rangle\{s_1^t/x_i\}_{i=1}^{k+1}$, and $[d]t \leadsto^{k+1} [c]t_{k+1}$ holds.

Finally, if $u = \mathbb{E}[x_i]$, for $1 \leq i \leq k$ (assuming $k \geq 1$), then we have $u\langle{}^{\mathbb{E}_0^t}/_{d}\rangle\{s_0^t/x_i\}_{i=1}^k \rightarrow_{\mathrm{n}} [c]t\,t_c 0$ and $u\langle{}^{\mathbb{E}_0^t}/_{d}\rangle\{s_0^t/x_i\}_{i=1}^k \rightarrow_{\mathrm{n}} [c]t\,t_c 1$, and $[c]t\,t_c 0\ \mathcal{R}\ [c]t\,t_c 1$ holds.

Let $[c]t\ t_c\ 0\ \mathcal{R}\ [c]t\ t_c\ 1$ with $[d]t \leadsto^k \mathbb{E}[x_i]$. There exist $t_1, \dots, t_k$ such that $[d]t \Downarrow_{\mathrm{n}} [d]\lambda x_1.t_1$, $[c]t_1 \Downarrow_{\mathrm{n}} [d]\lambda x_2.t_2$, $\dots [c]t_{k-1} \Downarrow_{\mathrm{n}} [d]\lambda x_k.t_k$, and $[c]t_k \rightarrow_{\mathrm{n}}^{*} \mathbb{E}[x_i]$. Then $[c]t\ t_c\ 0 \rightarrow_{\mathrm{n}}^{*} [c](\lambda x_1.t_1\langle{}^{[c]\Box\ t_c\ 0}/_{d}\rangle)\ t_c\ 0 \rightarrow_{\mathrm{n}} [c]t_1\langle{}^{[c]\Box\ t_c\ 0}/_{d}\rangle\{t_c/x_1\}\ 0 \rightarrow_{\mathrm{n}}^{*} [c]t_2\langle{}^{[c]\Box\ t_c\ 0}/_{d}\rangle\langle{}^{[c]\Box\ 0}/_{c}\rangle\{t_c/x_i\}_{i=1}^2\ 0 \rightarrow_{\mathrm{n}}^{*} [c]t_k\langle{}^{[c]\Box\ t_c\ 0}/_{d}\rangle\langle{}^{[c]\Box\ 0}/_{c}\rangle\{t_c/x_i\}_{i=1}^k\ 0 \rightarrow_{\mathrm{n}}^{*} \mathbb{E}'[t_c] \rightarrow_{\mathrm{n}} [c]0$ with $\mathbb{E}' \overset{\text{def}}{=} \mathbb{E}\langle{}^{[c]\Box\ t_c\ 0}/_{d}\rangle\langle{}^{[c]\Box\ 0}/_{c}\rangle\{t_c/x_i\}_{i=1}^k$. Similarly $[c]t\ t_c\ 1 \rightarrow_{\mathrm{n}}^{*} \mathbb{E}''[t_c] \rightarrow_{\mathrm{n}} [c]0$ with $\mathbb{E}'' \overset{\text{def}}{=} \mathbb{E}\langle{}^{[c]\Box\ t_c\ 1}/_{d}\rangle\langle{}^{[c]\Box\ 1}/_{c}\rangle\{t_c/x_i\}_{i=1}^k$. Because the two terms evaluate to $[c]0$, it is easy to conclude. □

B Call-by-Value $\lambda\mu$-calculus

B.1 Equivalence Proofs

Lemma B.1 (see Example 3.6) *The relation* $\mathcal{R} \overset{def}{=} \{(u\{v/y\}, u\{\lambda x.\mu b.[a]t_0/y\}) \mid [a]t_0 \to_{\mathrm{v}}^{*} u\{v/y\}\} \cup \approx$ *is an applicative bisimulation.*

Proof. We proceed by case analysis on u. If $u \to_{\mathrm{v}} u'$ or if $u = \mathbb{E}[v' y]$, then it is easy to conclude. If $u = \mathbb{E}[y\, v']$, then $u\{\lambda x.\mu b.[a]t_0/y\} \to_{\mathrm{v}} [a]t_0$ and $[a]t_0 \to_{\mathrm{v}}^{*} u\{v/y\}$ by definition, so we can conclude with Lemma 3.5. Similarly, the transition from $u\{v/y\}$ is matched by $u\{\lambda x.\mu b.[a]t_0/y\}$.

If $u = [c]v'$, then we have to compare (the result of the reduction of) $u\langle{}^{\mathbb{E}}/_{c}\rangle\{v\langle{}^{\mathbb{E}}/_{c}\rangle/y\}$ and $u\langle{}^{\mathbb{E}}/_{c}\rangle\{\lambda x.\mu b.([a]t_0)\langle{}^{\mathbb{E}}/_{c}\rangle/y\}$ for some $\mathbb{E}$ (which depends on the clause we check). The resulting terms are in $\mathcal{R}$, because $[a]t_0 \to_{\mathrm{v}}^{*} u\{v/y\}$ implies $([a]t_0)\langle{}^{\mathbb{E}}/_{c}\rangle \to_{\mathrm{v}}^{*} u\langle{}^{\mathbb{E}}/_{c}\rangle\{v\langle{}^{\mathbb{E}}/_{c}\rangle/y\}$.

□

We decompose the equivalence proof of Example 3.7 into several lemmas to improve readability. We remind that $u_0 \overset{\text{def}}{=} [b]\lambda xy.\Omega$, $v \overset{\text{def}}{=} \lambda y.\mu a.[b]\lambda x.y$, $u_1 \overset{\text{def}}{=} [b]\lambda xy.\Theta_{\mathsf{v}}\, v\, y$, and $\Theta_{\mathsf{v}} \overset{\text{def}}{=} (\lambda xy.y\,(\lambda z.xx\, y\, z))\,(\lambda xy.y\,(\lambda z.xx\, y\, z))$.

Lemma B.2 *Let* t_1*,* t_2 *be such that* $[c]t_1 \Downarrow_{\mathrm{v}} [c]v_1$ *(*$c \notin \mathsf{fn}(t_1)$*), and* $[d]t_2 \Downarrow_{\mathrm{v}} [d]v_2$ *(*$d \notin \mathsf{fn}(t_2)$*). For all* v'*, we have* $[b]\Omega \approx [b]\Theta_{\mathsf{v}}\, v\langle{}^{[b]\Box\, t_1\, t_2}/_{b}\rangle v'$ *.*

Proof. We have

$$\begin{aligned}
&[b]\Theta_{\mathsf{v}}\, v\langle{}^{[b]\Box\, t_1\, t_2}/_{b}\rangle v' \\
\to_{\mathrm{v}}^{*}&[b]v\langle{}^{[b]\Box\, t_1\, t_2}/_{b}\rangle\, (\lambda x.\Theta_{\mathsf{v}}\, v\langle{}^{[b]\Box\, t_1\, t_2}/_{b}\rangle\, x)\, v' \\
\to_{\mathrm{v}}^{*}&[b](\lambda zx.\Theta_{\mathsf{v}}\, v\langle{}^{[b]\Box\, t_1\, t_2}/_{b}\rangle\, x)\, t_1\, t_2 \\
\to_{\mathrm{v}}^{*}&[b](\lambda zx.\Theta_{\mathsf{v}}\, v\langle{}^{[b]\Box\, t_1\, t_2}/_{b}\rangle\, x)\, v_1'\, t_2 \\
\to_{\mathrm{v}}^{*}&[b](\lambda zx.\Theta_{\mathsf{v}}\, v\langle{}^{[b]\Box\, t_1\, t_2}/_{b}\rangle\, x)\, v_1'\, v_2' \\
\to_{\mathrm{v}}^{*}&[b]\Theta_{\mathsf{v}}\, v\langle{}^{[b]\Box\, t_1\, t_2}/_{b}\rangle\, v_2'
\end{aligned}$$

for some v_1', v_2' (which depend on v_1, v_2). So for all v', there exists v'' such that $[b]\Theta_{\mathsf{v}}\, v\langle{}^{[b]\Box\, t_1\, t_2}/_{b}\rangle v' \;\to_{\mathrm{v}}^{*}\; [b]\Theta_{\mathsf{v}}\, v\langle{}^{[b]\Box\, t_1\, t_2}/_{b}\rangle v''$; from that, we deduce that $[b]\Theta_{\mathsf{v}}\, v\langle{}^{[b]\Box\, t_1\, t_2}/_{b}\rangle v'$ is diverging, and therefore $[b]\Omega \approx [b]\Theta_{\mathsf{v}}\, v\langle{}^{[b]\Box\, t_1\, t_2}/_{b}\rangle v'$ holds.

□

Lemma B.3 *Let* $v' \overset{def}{=} \lambda z.t'$*, and* t_1 *such that there exists* v_1 *such that* $[c]t_1 \Downarrow_{\mathrm{v}} [c]v_1$ *for* $c \notin \mathsf{fn}(t_1)$*. We have*

$$[b]t'\{\lambda y.\Omega/z\} \approx [b]t'\{\lambda y.\Theta_{\mathsf{v}}\, v\langle{}^{[b]v'\,(\Box\, t_1)}/_{b}\rangle\, y/z\}.$$

Proof. Let $\mathcal{R} \overset{\text{def}}{=} \{(u\{\lambda y.\Omega/z\}, u\{\lambda y.\Theta_{\mathsf{v}}\, v\langle{}^{\mathbb{E}[v'\,(\Box\, t_1)]}/_{b}\rangle\, y/z\}) \mid \exists v_1.[c]t_1 \Downarrow_{\mathrm{v}} [c]v_1, c \notin \mathsf{fn}(t_1), v' = \lambda z.t', \mathbb{E}[t'] \to_{\mathrm{v}}^{*} u\} \cup \approx$. We prove $\mathcal{R}$ is an applicative bisimilarity, by case analysis on u.

The case $u \rightarrow_{\mathrm{v}} u'$ is easy. Suppose $u = [d]v_2$, with $v_2 \overset{\text{def}}{=} \lambda z_2.t_2$. Then we have two items to prove.

- Let $[e]t_3 \Downarrow_{\mathrm{v}} [e]v_3$ ($e \notin \mathsf{fn}(t_3)$). Then we have to relate

$$[d]t_2\{\lambda y.\Omega/z\}\langle{}^{[d]\square\ t_3}/_{d}\rangle\{v_3\langle{}^{[d]v_2\langle{}^{[d]\square\ t_3}/_{d}\rangle\ \square}/_{e}\rangle/z_2\}$$

to

$$[d](t_2\{\lambda y.\Theta_{\mathsf{v}}\ v\langle{}^{\mathbb{E}[v'\ (\square\ t_1)]}/_{b}\rangle\ y/z\})\langle{}^{[d]\square\ t_3}/_{d}\rangle\{v_3\langle{}^{[d]v_2\langle{}^{[d]\square\ t_3}/_{d}\rangle\ \square}/_{e}\rangle/z_2\}.$$

These terms can be rewritten into

$$[d]t_2\langle{}^{[d]\square\ t_3}/_{d}\rangle\{v_3\langle{}^{[d]v_2\langle{}^{[d]\square\ t_3}/_{d}\rangle\ \square}/_{e}\rangle/z_2\}\{\lambda y.\Omega/z\} \tag{B.1}$$

and

$$[d]t_2\langle{}^{[d]\square\ t_3}/_{d}\rangle\{v_3\langle{}^{[d]v_2\langle{}^{[d]\square\ t_3}/_{d}\rangle\ \square}/_{e}\rangle/z_2\}\{\lambda y.\Theta_{\mathsf{v}}\ v\langle{}^{\mathbb{E}'[v''\ (\square\ t_1')]}/_{b}\rangle\ y/z\} \tag{B.2}$$

where $\mathbb{E}' \overset{\text{def}}{=} \mathbb{E}\langle{}^{[d]\square\ t_3}/_{d}\rangle$, $v'' \overset{\text{def}}{=} v'\langle{}^{[d]\square\ t_3}/_{d}\rangle$, and $t_1' \overset{\text{def}}{=} t_1\langle{}^{[d]\square\ t_3}/_{d}\rangle$. We have $v'' = \lambda z.t'\langle{}^{[d]\square\ t_3}/_{d}\rangle$, and

$$\begin{aligned}
&\mathbb{E}'[t'\langle{}^{[d]\square\ t_3}/_{d}\rangle]\\
\rightarrow_{\mathrm{v}}^{*}&[d]v_2\langle{}^{[d]\square\ t_3}/_{d}\rangle\ t_3\ \text{(because } \mathbb{E}[t'] \rightarrow_{\mathrm{v}}^{*} u = [d]v_2\text{)}\\
\rightarrow_{\mathrm{v}}^{*}&[d]v_2\langle{}^{[d]\square\ t_3}/_{d}\rangle\ v_3\langle{}^{[d]v_2\langle{}^{[d]\square\ t_3}/_{d}\rangle\ \square}/_{e}\rangle\ \text{(because } [e]t_3 \Downarrow_{\mathrm{v}} [e]v_3\text{)}\\
\rightarrow_{\mathrm{v}}&[d]t_2\langle{}^{[d]\square\ t_3}/_{d}\rangle\{v_3\langle{}^{[d]v_2\langle{}^{[d]\square\ t_3}/_{d}\rangle\ \square}/_{e}\rangle/z_2\}.
\end{aligned}$$

The side-conditions are satisfied, therefore (B.1) and (B.2) are in $\mathcal{R}$.

- Let $v_3 = \lambda z_3.t_3$. We have to relate

$$[d]t_3\{{}^{v_2\{\lambda y.\Omega/z\}\langle{}^{[d]v_3\ \square}/_{d}\rangle}/_{z_3}\}$$

to

$$[d]t_3\{{}^{(v_2\{\lambda y.\Theta_{\mathsf{v}}\ v\langle{}^{\mathbb{E}[v'\ (\square\ t_1)]}/_{b}\rangle\ y/z\})\langle{}^{[d]v_3\ \square}/_{d}\rangle}/_{z_3}\}$$

These terms can be rewritten into

$$[d]t_3\{{}^{v_2\langle{}^{[d]v_3\ \square}/_{d}\rangle}/_{z_3}\}\{\lambda y.\Omega/z\} \tag{B.3}$$

and

$$[d]t_3\{{}^{v_2\langle{}^{[d]v_3\ \square}/_{d}\rangle}/_{z_3}\}\{\lambda y.\Theta_{\mathsf{v}}\ v\langle{}^{\mathbb{E}'[v''\ (\square\ t_1')]}/_{b}\rangle\ y/z\} \tag{B.4}$$

where $\mathbb{E}' \overset{\text{def}}{=} \mathbb{E}\langle{}^{[d]v_3\ \square}/_{d}\rangle$, $v'' \overset{\text{def}}{=} v'\langle{}^{[d]v_3\ \square}/_{d}\rangle$, and $t_1' \overset{\text{def}}{=} t_1\langle{}^{[d]v_3\ \square}/_{d}\rangle$. We have $v'' = \lambda z.t'\langle{}^{[d]v_3\ \square}/_{d}\rangle$, and

$$\begin{aligned}
&\mathbb{E}'[t'\langle{}^{[d]v_3\ \square}/_{d}\rangle]\\
\rightarrow_{\mathrm{v}}^{*}&[d]v_3\ v_2\langle{}^{[d]v_3\ \square}/_{d}\rangle\ \text{(because } \mathbb{E}[t'] \rightarrow_{\mathrm{v}}^{*} u = [d]v_2\text{)}\\
\rightarrow_{\mathrm{v}}&[d]t_3\{v_2\langle{}^{[d]v_3\ \square}/_{d}\rangle/z_3\}
\end{aligned}$$

The side-conditions are satisfied, therefore (B.3) and (B.4) are in $\mathcal{R}$.

The last case is when $u = \mathbb{E}'[x\, v'']$. Then $u\{\lambda y.\Omega/z\} \rightarrow_{\mathrm{v}} \mathbb{E}''[\Omega]$ for some $\mathbb{E}''$, and

$$\begin{aligned}
&u\{\lambda y.\Theta_{\mathsf{v}}\ v\langle{}^{\mathbb{E}[v'\ (\Box\ t_1)]}/_{b}\rangle\ y/z\}\\
\rightarrow_{\mathrm{v}}&\mathbb{E}^{(3)}[\Theta_{\mathsf{v}}\ v\langle{}^{\mathbb{E}[v'\ (\Box\ t_1)]}/_{b}\rangle\ v^{(3)}] \text{ for some } \mathbb{E}^{(3)}, v^{(3)}\\
\rightarrow_{\mathrm{v}}^{*}&\mathbb{E}^{(3)}[v\langle{}^{\mathbb{E}[v'\ (\Box\ t_1)]}/_{b}\rangle\ (\lambda y.\Theta_{\mathsf{v}}\ v\langle{}^{\mathbb{E}[v'\ (\Box\ t_1)]}/_{b}\rangle\ y)v^{(3)}]\\
\rightarrow_{\mathrm{v}}^{*}&\mathbb{E}[v'\ ((\lambda zy.\Theta_{\mathsf{v}}\ v\langle{}^{\mathbb{E}[v'\ (\Box\ t_1)]}/_{b}\rangle\ y)\ t_1)]\\
\rightarrow_{\mathrm{v}}^{*}&\mathbb{E}[v'\ ((\lambda zy.\Theta_{\mathsf{v}}\ v\langle{}^{\mathbb{E}[v'\ (\Box\ t_1)]}/_{b}\rangle\ y)\ v_1')] \text{ for some } v_1',\ \text{ because } [c]t_1 \Downarrow_{\mathrm{v}} [c]v_1\\
\rightarrow_{\mathrm{v}}&\mathbb{E}[v'\ (\lambda y.\Theta_{\mathsf{v}}\ v\langle{}^{\mathbb{E}[v'\ (\Box\ t_1)]}/_{b}\rangle\ y)]\\
\rightarrow_{\mathrm{v}}&\mathbb{E}[t'\{\lambda y.\Theta_{\mathsf{v}}\ v\langle{}^{\mathbb{E}[v'\ (\Box\ t_1)]}/_{b}\rangle\ y/z\}]\\
\rightarrow_{\mathrm{v}}^{*}&u\{\lambda y.\Theta_{\mathsf{v}}\ v\langle{}^{\mathbb{E}[v'\ (\Box\ t_1)]}/_{b}\rangle\ y/z\} \text{ because } \mathbb{E}[t'] \rightarrow_{\mathrm{v}}^{*} u
\end{aligned}$$

We obtain two non-terminating terms that are therefore bisimilar.

□

Lemma B.4 *Let $v' \overset{\mathit{def}}{=} \lambda z.t'$. We have*

$$[b]t'\{\lambda xy.\Omega/z\} \approx [b]t'\{\lambda xy.\Theta_{\mathsf{v}}\ v\langle{}^{[b]v'\ \Box}/_{b}\rangle\ y/z\}$$

Proof. Let $\mathcal{R} \overset{\mathrm{def}}{=} \{(u\{\lambda xy.\Omega/z\}, u\{\lambda xy.\Theta_{\mathsf{v}}\ v\langle{}^{\mathbb{E}[v'\ \Box]}/_{b}\rangle\ y/z\}) \mid v' = \lambda z.t', \mathbb{E}[t'] \rightarrow_{\mathrm{v}}^{*} u\} \cup \approx$. We prove $\mathcal{R}$ is an applicative bisimilarity, by case analysis on u. The proof is the same as for Lemma B.3; we only detail the last case, where $u = \mathbb{E}'[x\ v_1\ v_2]$. Then $u\{\lambda xy.\Omega/z\} \rightarrow_{\mathrm{v}}^{2} \mathbb{E}''[\Omega]$ for some $\mathbb{E}''$, and

$$\begin{aligned}
&u\{\lambda xy.\Theta_{\mathsf{v}}\ v\langle{}^{\mathbb{E}[v'\ \Box]}/_{b}\rangle\ y/z\}\\
\rightarrow_{\mathrm{v}}^{2}&\mathbb{E}^{(3)}[\Theta_{\mathsf{v}}\ v\langle{}^{\mathbb{E}[v'\ \Box]}/_{b}\rangle\ v_2'] \text{ for some } \mathbb{E}^{(3)}, v_2'\\
\rightarrow_{\mathrm{v}}^{*}&\mathbb{E}^{(3)}[v\langle{}^{\mathbb{E}[v'\ \Box]}/_{b}\rangle\ (\lambda y.\Theta_{\mathsf{v}}\ v\langle{}^{\mathbb{E}[v'\ \Box]}/_{b}\rangle\ y)v_2']\\
\rightarrow_{\mathrm{v}}^{*}&\mathbb{E}[v'\ (\lambda xy.\Theta_{\mathsf{v}}\ v\langle{}^{\mathbb{E}[v'\ \Box]}/_{b}\rangle\ y)]\\
\rightarrow_{\mathrm{v}}&\mathbb{E}[t'\{\lambda xy.\Theta_{\mathsf{v}}\ v\langle{}^{\mathbb{E}[v'\ \Box]}/_{b}\rangle\ y/z\}]\\
\rightarrow_{\mathrm{v}}^{*}&u\{\lambda xy.\Theta_{\mathsf{v}}\ v\langle{}^{\mathbb{E}[v'\ \Box]}/_{b}\rangle\ y/z\} \text{ because } \mathbb{E}[t'] \rightarrow_{\mathrm{v}}^{*} u
\end{aligned}$$

We obtain two non-terminating terms that are therefore bisimilar.

□

Lemma B.5 (Example 3.7) *The relation $\{([b]\lambda xy.\Omega, [b]\lambda xy.\Theta_{\mathsf{v}}\ v\ y)\} \cup \approx$ is an applicative bisimulation.*

Proof. We have to prove two items

- Let t_1 such that $[c]t_1 \Downarrow_{\mathrm{v}} [c]v_1$ ($c \notin \mathsf{fn}(t_1)$). We have to prove that $[b]\lambda y.\Omega$ is bisimilar to $[b]\lambda y.\Theta_{\mathsf{v}}\ v\langle{}^{[b]\Box\ t_1}/_{b}\rangle y$, which, in turn, requires that
  - · for all $[d]t_2 \Downarrow_{\mathrm{v}} [d]v_2$ ($d \notin \mathsf{fn}(t_2)$), we need $[b]\Omega \approx [b]\Theta_{\mathsf{v}}\ v\langle{}^{[b]\Box\ t_1\ t_2}/_{b}\rangle v_2'$ for some v_2'. This holds by Lemma B.2;
  - · for all $v' \overset{\mathrm{def}}{=} \lambda z.t'$, we need $[b]t'\{\lambda y.\Omega/z\} \approx [b]t'\{\lambda y.\Theta_{\mathsf{v}}\ v\langle{}^{[b]v'\ (\Box\ t_1)}/_{b}\rangle\ y/z\}$. This holds by Lemma B.3.

- Let $v' \overset{\text{def}}{=} \lambda z.t'$. We have to prove that $[b]t'\{\lambda xy.\Omega/z\}$ is related to the term $[b]t'\{\lambda xy.\Theta_{\mathrm{v}}\, v\langle {}^{[b]v'\,\Box}/_{b}\rangle\; y/z\}$. This a consequence of Lemma B.4.

□

For the next lemma and its proof, we use the same definitions of terms as in Example 3.3.

Lemma B.6 (see Example 3.3) *The relation*

$$\mathcal{R} \overset{def}{=} \{(u\langle {}^{[a]E[v_0\langle {}^{[a]E[\Box\; t]}/_{a}\rangle\; \Box]}/_{b}\rangle, u\langle {}^{[a]E[v_1\; \Box]}/_{b}\rangle) \;\mid\; [b]t \Downarrow_{\mathrm{v}} [b]v, b \notin \mathsf{fn}(t)\}$$

is an applicative bisimulation.

Proof. We proceed by case analysis on u. If $u \rightarrow_{\mathrm{v}} u'$, then the result holds. If $u = [c]v'$ with $c \neq a$ and $c \neq b$, then when checking clauses i and ii, we obtain terms that can be written $u'\langle {}^{[a]E[v_0\langle {}^{[a]E[\Box\; t]}/_{a}\rangle\; \Box]}/_{b}\rangle$ and $u'\langle {}^{[a]E[v_1\; \Box]}/_{b}\rangle$ for some u', and are therefore in $\mathcal{R}$.

If $u = [a]v'$, then when checking clauses i and ii, we obtain terms that can be written $u'\langle {}^{[a]E'[E[v_0\langle {}^{[a]E'[E[\Box\; t]]}/_{a}\rangle\; \Box]]}/_{b}\rangle$ and $u'\langle {}^{[a]E'[E[v_1\; \Box]]}/_{b}\rangle$ for some u' and E' (depending on which clause we check). We obtain terms in $\mathcal{R}$.

If $u = [b]v'$, then

$$\begin{aligned} u\langle {}^{[a]E[v_0\langle {}^{[a]E[\Box\; t]}/_{a}\rangle\; \Box]}/_{b}\rangle &= [a]E[v_0\langle {}^{[a]E[\Box\; t]}/_{a}\rangle\; v'\langle {}^{[a]E[v_0\langle {}^{[a]E[\Box\; t]}/_{a}\rangle\; \Box]}/_{b}\rangle] \\ &\rightarrow_{\mathrm{v}}^{*} [a]E[w'\; (v'\langle {}^{[a]E[v_0\langle {}^{[a]E[\Box\; t]}/_{a}\rangle\; \Box]}/_{b}\rangle\; \lambda x.x)] \end{aligned}$$

and $u\langle {}^{[a]E[v_1\; \Box]}/_{b}\rangle = [a]E[v_1\;\; v'\langle {}^{[a]E[v_1\; \Box]}/_{b}\rangle] \rightarrow_{\mathrm{v}}^{*} [a]E[w'\; (v'\langle {}^{[a]E[v_1\; \Box]}/_{b}\rangle\; \lambda x.x)]$; we obtain terms in $\mathcal{R}$.

□

The next lemma uses the notations of Section 3.3.

Lemma B.7 (see Example 3.13) *Let $\mathbb{E}$, $\mathbb{E}_0$, and E_1 be such that $\mathbb{E} = \mathbb{E}_0[E_1]$. Then $\mathbb{E}^{\dagger} \approx \lambda x.\mathbb{E}_0^{\dagger}\; E_1[x]$.*

Proof. Let a be a fresh name. Let t such that $[d]t \Downarrow_{\mathrm{v}} [d]v$. To check clause i, we have to relate $[a]\mu c.\mathbb{E}[v\langle {}^{[a]\mathbb{E}^{\dagger}\; \Box}/_{d}\rangle]$ and $[a]\mathbb{E}_0^{\dagger}\; E_1[v\langle {}^{[a]\lambda x.\mathbb{E}_0^{\dagger}\; E_1[x]\; \Box}/_{d}\rangle]$. These terms reduce respectively to $\mathbb{E}[v\langle {}^{[a]\mathbb{E}^{\dagger}\; \Box}/_{d}\rangle]$ and $\mathbb{E}_0[E_1[v\langle {}^{[a]\lambda x.\mathbb{E}_0^{\dagger}\; E_1[x]\; \Box}/_{d}\rangle]]$, which we can rewrite into respectively $u\langle {}^{[a]\mathbb{E}^{\dagger}\; \Box}/_{d}\rangle$ and $u\langle {}^{[a]\lambda x.\mathbb{E}_0^{\dagger}\; E_1[x]\; \Box}/_{d}\rangle$. To check clause ii, we have to relate $[a]t\{\mathbb{E}^{\dagger}/y\}$ and $[a]t\{\lambda x.\mathbb{E}_0^{\dagger}\, E_1[x]/y\}$ for all t. The most interesting case is when $t = E[y\, v]$; then these terms reduce to $\mathbb{E}[v]\{\mathbb{E}^{\dagger}/y\}$ and $\mathbb{E}[v]\{\lambda x.\mathbb{E}_0^{\dagger}\; E_1[x]/y\}$ respectively. In fact, one can prove the result by showing that

$$\{(u\langle {}^{[a]\mathbb{E}^{\dagger}\; \Box}/_{d}\rangle, u\langle {}^{[a]\lambda x.\mathbb{E}_0^{\dagger}\; E_1[x]\; \Box}/_{d}\rangle), (u\{\mathbb{E}^{\dagger}/y\}, u\{\lambda x.\mathbb{E}_0^{\dagger}\; E_1[x]/y\}) \;\mid\; \mathbb{E} = \mathbb{E}_0[E_1]\}$$

is an applicative bisimulation. □

B.2 Proof of Soundness

Lemma B.8 *If* $[a]v_0 \approx [a]v_1$, *then for all* $[b]t \rightarrow_{\mathrm{v}}^{*} u$ *(*$b \notin \mathsf{fn}(t)$*), we have* $u\langle {}^{[a]v_0\langle {}^{[a]\Box\ t}/_{a}\rangle\ \Box}/_{b}\rangle \approx u\langle {}^{[a]v_1\langle {}^{[a]\Box\ t}/_{a}\rangle\ \Box}/_{b}\rangle$.

Proof. The relation $\{(u\langle {}^{[a]v_0\langle {}^{[a]\Box\ t}/_{a}\rangle\ \Box}/_{b}\rangle, u\langle {}^{[a]v_1\langle {}^{[a]\Box\ t}/_{a}\rangle\ \Box}/_{b}\rangle) \mid \forall u, [b]t \rightarrow_{\mathrm{v}}^{*} u\} \cup \approx$ is a bisimulation. □

Lemma B.9 *If* $\lambda x.t_0\ (\approx^{\bullet})^{c}\ t_1$, *then there exists* t *such that* $t_0 \approx^{\bullet} t$, $\mathsf{fv}(t) \subseteq \{x\}$ *and* $\lambda x.t \approx t_1$.

Proof. By induction on $\lambda x.t_0\ (\approx^{\bullet})^{c}\ t_1$. □

Lemma B.10 *If* $[a]t_0\ (\approx^{\bullet})^{c}\ u_1$, *then there exists a closed* t *such that* $t_0\ (\approx^{\bullet})^{c}\ t$ *and* $[a]t \approx u_1$.

Proof. By induction on $[a]t_0\ (\approx^{\bullet})^{c}\ u_1$. □

Lemma B.11 *Let* $t_0\ (\approx^{\bullet})^{c}\ t_1$, *and* $a \notin \mathsf{fn}(t_0) \cup \mathsf{fn}(t_1)$.

- *If* $[a]t_0 \rightarrow_{\mathrm{n}} u_0$, *then* $[a]t_1 \rightarrow_{\mathrm{n}}^{*} u_1$ *and* $u_0\ (\approx^{\bullet})^{c}\ u_1$.
- *If* $t_0 = \lambda x.t_0'$, *then* $[a]t_1 \rightarrow_{\mathrm{n}}^{*} [a]\lambda x.t_1'$ *and for all* t, b, v *such that* $[b]t \rightarrow_{\mathrm{v}}^{*} [b]v$ *and* $b \notin \mathsf{fn}(t)$, *we have*

$$[a]t_0'\langle {}^{[a]\Box\ t}/_{a}\rangle\{v\langle {}^{[a]\lambda x.t_0\langle {}^{[a]\Box\ t}/_{a}\rangle\ \Box}/_{b}\rangle/x\}\ (\approx^{\bullet})^{c}$$
$$[a]t_1'\langle {}^{[a]\Box\ t}/_{a}\rangle\{v\langle {}^{[a]\lambda x.t_1\langle {}^{[a]\Box\ t}/_{a}\rangle\ \Box}/_{b}\rangle/x\}$$

and for all $v' = \lambda x.t'$, *we have*

$$[a]t'\{\lambda x.t_0'\langle {}^{[a]v\ \Box}/_{a}\rangle/x\}\ (\approx^{\bullet})^{c}\ [a]t'\{\lambda x.t_1'\langle {}^{[a]v\ \Box}/_{a}\rangle/x\}.$$

Proof. By induction on $t_0\ (\approx^{\bullet})^{c}\ t_1$. If $t_0 \approx t_1$, then the result holds by bisimilarity. If $t_0 \approx^{\bullet} t \approx^{\circ} t_1$, so that t is closed (using Lemma A.1 if necessary), then we can conclude with the induction hypothesis and the bisimilarity definition.

If $t_0\ \widetilde{(\approx^{\bullet})^{c}}\ t_1$, then we have several cases to consider.

If $t_0 = \lambda x.t_0'$ and $t_1 = \lambda x.t_1'$ with $t_0' \approx^{\bullet} t_1'$, we can prove the required result because $\approx^{\bullet}$ is substitutive. Suppose $t_0 = \mu b.u_0$ and $t_1 = \mu b.u_1$ with $u_0\ (\approx^{\bullet})^{c}\ u_1$. We have $[a]t_0 \rightarrow_{\mathrm{v}} u_0\langle[a]\Box/b\rangle$, $[a]t_1 \rightarrow_{\mathrm{v}} u_1\langle[a]\Box/b\rangle$, and $u_0\langle[a]\Box/b\rangle\ (\approx^{\bullet})^{c}\ u_1\langle[a]\Box/b\rangle$, hence the result holds.

Suppose $t_0 = t_0^1\, t_0^2$, $t_1 = t_1^1\, t_1^2$ with $t_0^1\ (\approx^{\bullet})^{c}\ t_1^1$ and $t_0^2\ (\approx^{\bullet})^{c}\ t_1^2$. We distinguish three cases.

- If $[b]t_0^1 \rightarrow_{\mathrm{v}} u_0$ (for some fresh b), then $[a]t_0 \rightarrow_{\mathrm{v}} u_0\langle[a]\Box\ t_0^2/b\rangle$. By the induction hypothesis, there exists u_1 such that $[b]t_1^1 \rightarrow_{\mathrm{v}}^{*} u_1$ and $u_0\ (\approx^{\bullet})^{c}\ u_1$. Consequently, we have $[a]t_1 \rightarrow_{\mathrm{v}}^{*} u_1\langle[a]\Box\ t_1^2/b\rangle$, and by definition of $\approx^{\bullet}$, we have $u_0\langle[a]\Box\ t_0^2/b\rangle\ (\approx^{\bullet})^{c}\ u_1\langle[a]\Box\ t_1^2/b\rangle$, as required.
- Suppose $t_0^1 = \lambda x.t_0'$ and $[b]t_0^2 \rightarrow_{\mathrm{v}} u_0$ for some fresh b; then we have $[a]t_0 \rightarrow_{\mathrm{v}} u_0\langle[a]\lambda x.t_0'\ \ \Box/b\rangle$. By the induction hypothesis, there exists u_1 such that

$[b]t_1^2 \to_{\mathrm{v}}^* u_1$ and $u_0\ (\approx^\bullet)^c\ u_1$. Because $\lambda x.t_0'\ (\approx^\bullet)^c\ t_1^1$, there exists a t such that $t_0' \approx^\bullet t$ and $\lambda x.t \approx t_1^1$ by Lemma B.9. From $t_0' \approx^\bullet t$, we deduce $[a]\lambda x.t_0'\ \Box\ (\approx^\bullet)^c\ [a]\lambda x.t\langle [a]\Box\ t_1^2/a\rangle\ \Box$, which implies $u_0\langle {}^{[a]\lambda x.t_0'\ \Box}/b\rangle\ (\approx^\bullet)^c\ u_1\langle {}^{[a]\lambda x.t\langle {}^{[a]\Box\ t_1^2}/a\rangle\ \Box}/b\rangle$. Because $\lambda x.t \approx t_1^1$, there exists v_1 such that $[a]t_1^1 \to_{\mathrm{v}}^* [a]v_1$, and we have $[a]\lambda x.t \approx [a]v_1$. By Lemma B.8, $u_1\langle {}^{[a]\lambda x.t\langle {}^{[a]\Box\ t_1^2}/a\rangle\ \Box}/b\rangle \approx u_1\langle {}^{[a]v_1\langle {}^{[a]\Box\ t_1^2}/a\rangle\ \Box}/b\rangle$ holds, so $u_0\langle {}^{[a]\lambda x.t_0'\ \Box}/b\rangle\ (\approx^\bullet)^c\ u_1\langle {}^{[a]v_1\langle {}^{[a]\Box\ t_1^2}/a\rangle\ \Box}/b\rangle$ holds as well. Besides, we have $[a]t_1 \to_{\mathrm{v}}^* u_1\langle {}^{[a]v_1\langle {}^{[a]\Box\ t_1^2}/a\rangle\ \Box}/b\rangle$, hence we have the required result.

- If $t_0^1 = \lambda x.t_0'$ and $t_0^2 = \lambda y.t_0''$, then $[a]t_0 \to_{\mathrm{v}} [a]t_0'\{\lambda x.t_0''/x\}$. By Lemma B.9, there exist closed t, t' such that $t_0'\ (\approx^\bullet)^c\ t$, $\lambda x.t \approx t_1^1$, $t_0''\ (\approx^\bullet)^c\ t'$, and $\lambda y.t' \approx t_1^2$. From $t_0'\ (\approx^\bullet)^c\ t$, we deduce $t_0'\ (\approx^\bullet)^c\ t\langle [a]\Box\ t_1^2/a\rangle$, and from $t_0''\ (\approx^\bullet)^c\ t'$, we deduce $\lambda y.t_0''\ (\approx^\bullet)^c\ \lambda y.t'\langle {}^{[a]\lambda x.t\langle {}^{[a]\Box\ t_1^2}/a\rangle\ \Box}/a\rangle$. Consequently, we have $[a]t_0'\{\lambda y.t_0''/x\}\ (\approx^\bullet)^c\ [a]t\langle [a]\Box\ t_1^2/a\rangle\{\lambda y.t'\langle {}^{[a]\lambda x.t\langle {}^{[a]\Box\ t_1^2}/a\rangle\ \Box}/a\rangle/x\}$. From $\lambda y.t' \approx t_1^2$, by bisimilarity, there exists v_1 such that $[a]t_1^2 \to_{\mathrm{v}}^* [a]v_1$, and

  $$\begin{aligned}&[a]t\langle [a]\Box\ t_1^2/a\rangle\{\lambda y.t'\langle {}^{[a]\lambda x.t\langle {}^{[a]\Box\ t_1^2}/a\rangle\ \Box}/a\rangle/x\}\\ &\qquad\qquad \approx [a]t\langle [a]\Box\ t_1^2/a\rangle\{v_1\langle {}^{[a]\lambda x.t\langle {}^{[a]\Box\ t_1^2}/a\rangle\ \Box}/a\rangle/x\}\end{aligned}$$

  holds (clause ii). From $\lambda x.t \approx t_1^1$, by bisimilarity, there exists t_1 such that $[a]t_1^1 \to_{\mathrm{v}}^* [a]\lambda x.t_1$, and

  $$\begin{aligned}&[a]t\langle [a]\Box\ t_1^2/a\rangle\{v_1\langle {}^{[a]\lambda x.t\langle {}^{[a]\Box\ t_1^2}/a\rangle\ \Box}/a\rangle/x\}\\ &\qquad\qquad \approx [a]t_1\langle [a]\Box\ t_1^2/a\rangle\{v_1\langle {}^{[a]\lambda x.t_1\langle {}^{[a]\Box\ t_1^2}/a\rangle\ \Box}/a\rangle/x\}\end{aligned}$$

  holds (clause i). Finally, we have

  $$[a]t_0'\{\lambda y.t_0''/x\}\ (\approx^\bullet)^c\ [a]t_1\langle [a]\Box\ t_1^2/a\rangle\{v_1\langle {}^{[a]\lambda x.t_1\langle {}^{[a]\Box\ t_1^2}/a\rangle\ \Box}/a\rangle/x\},$$

  and because $[a]t_1 \to_{\mathrm{v}}^* [a]t_1\langle [a]\Box\ t_1^2/a\rangle\{v_1\langle {}^{[a]\lambda x.t_1\langle {}^{[a]\Box\ t_1^2}/a\rangle\ \Box}/a\rangle/x\}$, we have the required result.

Suppose $t_0 = t_0'\langle \mathbb{E}_0/b\rangle$, $t_1 = t_1'\langle \mathbb{E}_1/b\rangle$ with $t_0'\ (\approx^\bullet)^c\ t_1'$, $\mathbb{E}_0\ (\approx^\bullet)^c\ \mathbb{E}_1$. If $[a]t_0 \to_{\mathrm{v}} u_0$, then in fact $[a]t_0' \to_{\mathrm{v}} u_0'$ with $u_0 = u_0'\langle \mathbb{E}_0/b\rangle$. By the induction hypothesis, there exists u_1' such that $[a]t_1' \to_{\mathrm{v}}^* u_1'$ and $u_0'\ (\approx^\bullet)^c\ u_1'$. Consequently, we have $[a]t_1 \to_{\mathrm{v}}^* u_1'\langle \mathbb{E}_1/b\rangle$, and $u_0'\langle \mathbb{E}_0/b\rangle\ (\approx^\bullet)^c\ u_1'\langle \mathbb{E}_1/b\rangle$ holds, as wished. If $[a]t_0$ is a named value, then in fact $t_0' = \lambda x.t_0'''$ and $t_0 = \lambda x.t_0'''\langle \mathbb{E}_0/b\rangle$. The result holds by using the induction hypothesis (and with some renaming of b into a fresh c to avoid name clashes).

□

Lemma B.12 *The relation $(\approx^\bullet)^c$ is an applicative simulation.*

Proof. Let $u_0\ (\approx^\bullet)^c\ u_1$; we prove the simulation clause by induction on $u_0\ (\approx^\bullet)^c\ u_1$.

If $u_0 \approx u_1$, then the result holds by bisimilarity. If $u_0 \approx^\bullet u \approx^\circ u_1$, then we can conclude using the induction hypothesis and the bisimilarity definition.

If $u_0\ \widetilde{(\approx^\bullet)^c}\ u_1$, then we have two cases. Suppose $u_0 = [a]t_0$ and $u_1 = [a]t_1$ with $t_0\ (\approx^\bullet)^c\ t_1$. If $a \notin \mathsf{fn}(t_0) \cup \mathsf{fn}(t_1)$, then we can apply Lemma B.11 directly to get the required result. Otherwise, let $b \notin \mathsf{fn}(t_0) \cup \mathsf{fn}(t_1)$. If $u_0 \to_{\mathrm{v}} u_0'$, then $[b]t_0 \to_{\mathrm{v}} u_0''$ and $u_0' = u_0'\langle [a]\Box/b\rangle$. We can apply Lemma B.11 to $[b]t_0$ and $[b]t_1$, and then rename b into a.

Suppose $u_0 = u_0'\langle \mathbb{E}_0/a\rangle$, $u_1 = u_1'\langle \mathbb{E}_1/a\rangle$ with $u_0' \; (\approx^\bullet)^c \; u_1'$ and $\mathbb{E}_0 \; (\approx^\bullet)^c \; \mathbb{E}_1$.

If u_0 is a named value, then we use the induction hypothesis, with some renaming (as in the call-by-name case) to avoid some name clashes.

If $u_0 \to_{\mathrm{v}}$, then again we distinguish several cases. First, suppose $u_0' \to_{\mathrm{v}} u_0''$; then $u_0 \to_{\mathrm{v}} u_0''\langle E_0/a\rangle$. By the induction hypothesis, there exists u_1'' such that $u_1' \to_{\mathrm{v}}^* u_1''$ and $u_0'' \; (\approx^\bullet)^c \; u_1''$. Then $u_1 \to_{\mathrm{v}}^* u_1''\langle \mathbb{E}_1/a\rangle$ and $u_0''\langle \mathbb{E}_0/a\rangle \; (\approx^\bullet)^c \; u_1''\langle \mathbb{E}_1/a\rangle$, hence the result holds.

Second, assume $u_0' = [a]\lambda x.t_0$, $\mathbb{E}_0 = \mathbb{E}_0'[\Box\ v_0]$ and $u_0 \to_{\mathrm{v}} \mathbb{E}_0'[t_0\langle \mathbb{E}_0/a\rangle\{v_0/x\}]$. By Lemma B.10, there exists t such that $\lambda x.t_0 \; (\approx^\bullet)^c \; t$ and $[a]t \approx u_1'$. By Lemma B.9, there exists t' such that $t_0 \approx^\bullet t'$ and $\lambda x.t' \approx t$. We can prove by induction on $\mathbb{E}_0 \; (\approx^\bullet)^c \; \mathbb{E}_1$ that $\mathbb{E}_1 = \mathbb{E}_1'[\Box\ t_1']$ with $\mathbb{E}_0' \; (\approx^\bullet)^c \; \mathbb{E}_1'$ and $v_0 \; (\approx^\bullet)^c \; t_1'$. Suppose $v_0 = \lambda x.t_0'$. By Lemma B.9, there exists s such that $t_0' \approx^\bullet s$ and $\lambda x.s \approx t_1'$. Let $\mathbb{E}_i''$, s', t_i'' be $\mathbb{E}_i'$, s, and t_i' with a renamed into a fresh c. From $t_0 \; (\approx^\bullet)^c \; t'$, $\lambda x.t_0'' \; (\approx^\bullet)^c \; t_1''$, and $\lambda x.t_0'' \; (\approx^\bullet)^c \; \lambda x.s'$, we deduce

$$[a]t_0\langle {}^{[a]\Box\ \lambda x.t_0''}/a\rangle\{\lambda x.t_0''/x\} \; (\approx^\bullet)^c \; [a]t'\langle {}^{[a]\Box\ t_1''}/a\rangle\{\lambda x.s'/x\}.$$

Because $\lambda x.s' \approx t_1''$, there exists v_1' such that $[b]t_1'' \to_{\mathrm{v}}^* [b]v_1'$ (for a fresh b), and $[a]t'\langle {}^{[a]\Box\ t_1''}/a\rangle\{\lambda x.s'/x\} \approx [a]t'\langle {}^{[a]\Box\ t_1''}/a\rangle\{v_1'\langle {}^{[a]\lambda x.t'\langle {}^{[a]\Box\ t_1''}/a\rangle\ \Box}/b\rangle/x\}$ (using the second item of the value case of the bisimulation definition). Because $\lambda x.t' \approx t$, there exists t'' such that $[a]t \to_{\mathrm{v}}^* [a]\lambda x.t''$ and

$$\begin{aligned} &[a]t'\langle {}^{[a]\Box\ t_1''}/a\rangle\{v_1'\langle {}^{[a]\lambda x.t'\langle {}^{[a]\Box\ t_1''}/a\rangle\ \Box}/b\rangle/x\} \\ &\qquad\qquad \approx [a]t''\langle {}^{[a]\Box\ t_1''}/a\rangle\{v_1'\langle {}^{[a]\lambda x.t''\langle {}^{[a]\Box\ t_1''}/a\rangle\ \Box}/b\rangle/x\} \end{aligned}$$

(by clause i). Because $[a]\lambda x.t' \approx u_1'$, there exists t_1 such that $u_1' \to_{\mathrm{v}}^* [a]\lambda x.t_1$ and we have also

$$\begin{aligned} &[a]t''\langle {}^{[a]\Box\ t_1''}/a\rangle\{v_1'\langle {}^{[a]\lambda x.t''\langle {}^{[a]\Box\ t_1''}/a\rangle\ \Box}/b\rangle/x\} \\ &\qquad\qquad \approx [a]t_1\langle {}^{[a]\Box\ t_1''}/a\rangle\{v_1'\langle {}^{[a]\lambda x.t_1\langle {}^{[a]\Box\ t_1''}/a\rangle\ \Box}/b\rangle/x\} \end{aligned}$$

by clause i. Finally, we obtain

$$[a]t_0\langle {}^{[a]\Box\ \lambda x.t_0''}/a\rangle\{\lambda x.t_0''/x\} \; (\approx^\bullet)^c \; [a]t_1\langle {}^{[a]\Box\ t_1''}/a\rangle\{v_1'\langle {}^{[a]\lambda x.t_1\langle {}^{[a]\Box\ t_1''}/a\rangle\ \Box}/b\rangle/x\}$$

by transitivity of $\approx$ and definition of $\approx^\bullet$, from which we can deduce

$$\begin{aligned} &\mathbb{E}_0''[t_0\langle {}^{\mathbb{E}_0''[\Box\ \lambda x.t_0'']}/a\rangle\{\lambda x.t_0''/x\}] \\ &\qquad\qquad (\approx^\bullet)^c \; \mathbb{E}_1''[t_1\langle {}^{\mathbb{E}_1''[\Box\ t_1'']}/a\rangle\{v_1'\langle {}^{\mathbb{E}_1''[\lambda x.t_1\langle {}^{\mathbb{E}_1''[\Box\ t_1'']}/a\rangle\ \Box]}/b\rangle/x\}]. \end{aligned}$$

Renaming c back into a, we get

$$\mathbb{E}_0'[t_0\langle{}^{\mathbb{E}_0'[\Box\ \lambda x.t_0']}/_{a}\rangle\{\lambda x.t_0'/x\}]$$
$$(\approx^\bullet)^c\ \mathbb{E}_1'[t_1\langle{}^{\mathbb{E}_1'[\Box\ t_1']}/_{a}\rangle\{v_1\langle{}^{\mathbb{E}_1'[\lambda x.t_1\langle{}^{\mathbb{E}_1'[\Box\ t_1']}/_{a}\rangle\ \Box]}/_{b}\rangle/x\}]$$

(assuming v_1 is the result of renaming c into a in v_1'), which is the same as

$$\mathbb{E}_0'[t_0\langle{}^{\mathbb{E}_0}/_{a}\rangle\{\lambda x.t_0'/x\}]\ (\approx^\bullet)^c\ \mathbb{E}_1'[t_1\langle{}^{\mathbb{E}_1}/_{a}\rangle\{v_1\langle{}^{\mathbb{E}_1'[\lambda x.t_1\langle{}^{\mathbb{E}_1}/_{a}\rangle\ \Box]}/_{b}\rangle/x\}].$$

One can check that $u_1 \to_{\mathrm{v}}^* \mathbb{E}_1'[t_1\langle{}^{\mathbb{E}_1}/_{a}\rangle\{v_1\langle{}^{\mathbb{E}_1'[\lambda x.t_1\langle{}^{\mathbb{E}_1}/_{a}\rangle\ \Box]}/_{b}\rangle/x\}]$, hence the result holds.

Finally, the last case is $u_0' = [a]v_0$, $\mathbb{E}_0 = \mathbb{E}_0'[\lambda x.t_0\ \Box]$, which give $u_0 \to_{\mathrm{v}} \mathbb{E}_0'[t_0\{v_0\langle\mathbb{E}_0/a\rangle/x\}]$. This case is similar to the previous one and is left to the reader.

□

On Grainless Footprint Semantics for Shared-memory Programs

Stephen Brookes

Carnegie Mellon University [1]

1 Outline

We develop an improved grainless denotational semantics for shared-memory parallel programs, building on ideas from earlier trace-based models with local states and footprints [4]. The key new idea is a more refined approach to race detection, leading to a model with better abstraction properties. Rather than treat a race condition as a "global" catastrophe [3,4], we track information about variables whose value may be tainted by a race, and retain accurate information about unaffected variables. As in the prior work, we abstract away from state changes that occur in between synchronization points, in a manner consistent with Dijkstra's Principle [5]. Our semantics supports compositional program analysis based on "sequential" reasoning for sequential code fragments, even when this code occurs in parallel contexts, and yields a simple semantic characterization of race-free code. The semantics validates the static constraints on "critical variables" imposed in concurrent programming methodology [6,9] and serves as a foundation for reasoning about safe partial correctness, as in concurrent separation logic [8]. The new treatment of race detection allows for more refined analysis of racy programs. By framing our ideas and concepts in a general manner we hope that our results may be applied in a wider setting.

Introduction

Shared-memory programs are difficult to reason about, because of the potential for concurrent interference. Early semantic models assume fixed granularity of exe-

[1] This research was sponsored by the National Science Foundation under grant no. CCF-1017011. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

cution, for example atomic assignments or atomic reads and writes [2,10]. Such assumptions may not hold in practice, and program analysis based on such models is only valid for implementations enforcing the required degree of granularity. In contrast, traditional semantic models for sequential programs ignore granularity and focus on the relationship between initial and final states of a program execution, interpreting a program as a state transformation. Such semantics is simple and easy to use for compositional reasoning about sequential programs, but inadequate for parallel programs [10]: the state transformation denoted by $c_1 \| c_2$ cannot be determined from the state transformations denoted by c_1 and c_2. Ideally we need a semantic model for parallel programs that enjoys the simplicity of state transformation semantics while retaining enough information about intermediate states to allow proper modeling of concurrent interference, without making granularity assumptions.

Such concerns have stimulated an effort to design a “grainless” semantic model for shared-memory concurrency, notably by John Reynolds [11] and this author [4]. Reynolds sought to avoid granularity by breaking atomic actions into instantaneous fragments, an approach that leads to a semantics based on very small steps, and therefore likely to suffer from combinatorial problems. This author developed a “footstep trace” model, a pre-cursor to the approach offered in this paper, but in retrospect we can now see that this model is overly complex and fails full abstraction. Here we offer a more streamlined version of footstep trace semantics, with better abstraction properties. The main new idea involves a more refined account of race conditions, an apparently simple idea with deep ramifications in the construction of the semantics. We classify our semantic model as “grainless” because the model construction abstracts away from irrelevant scheduling details in such a way that there is no need to retain information about what constitutes an “atomic” action, other than the usual assumption that primitive operations for synchronization are atomic.

We deal here with a simple shared-memory language, omitting pointers. Our development builds on our prior work on trace models [2] and on concurrent separation logic [3,4], in which we combined mutable state with concurrency, so we expect to be able to adapt the new ideas presented here accordingly. It is straightforward to incorporate locally scoped declarations. For space reasons we defer these extensions and some semantic details and proofs to a fuller version of the paper.

2 Syntax and Static Semantics

We will deal with a simple shared-memory parallel language, extending the familiar class of sequential while-programs with parallel composition and a well known synchronization construct: conditional critical regions. Identifiers (or program variables) i are assignable integer-valued variables, and region names (or resources) r take values 0 and 1, representing “available” and “unavailable”. The sets **Ide** of identifiers and **Res** of resource names are disjoint. The syntax of integer expressions e and boolean expressions b is conventional, including the usual arithmetic and boolean constructs. We define the sets $\texttt{free}(e)$ and $\texttt{free}(b)$ of identifiers having a free syntactic occurrence in e and b, as usual, by structural induction.

The syntax of commands (or processes) c is given by:

$$\begin{aligned} c \;::=\;& \textbf{skip} \mid i{:=}e \mid c_1;c_2 \mid \textbf{if } b \textbf{ then } c_1 \textbf{ else } c_2 \mid \textbf{while } b \textbf{ do } c \mid \\ & c_1\|c_2 \mid \textbf{with } r \textbf{ when } b \textbf{ do } c \end{aligned}$$

Resources behave like binary semaphores with atomic operations for locking and unlocking, so critical regions can be used to ensure mutually exclusive access to shared variables. As usual we abbreviate **with** r **when true do** c as **with** r **do** c. Resource names are not assignable. The sets $\texttt{res}(c)$ and $\texttt{free}(c)$ of resource names and identifiers with a free occurrence in c are defined by structural induction in the obvious manner.

3 Dynamic Semantics

A program denotes a set of traces representing interactive computations in which the program and its environment make changes to the shared state. Each step in a trace represents the effect of a finite sequence of actions performed by the program, and records just the overall footprint. We detect the potential for race conditions, involving a write to a shared variable whose value is used in a concurrent update. A race condition can lead to unpredictable behavior, so we use $\top$ to represent the value of a variable whose value is race-dependent, a special value which taints all future computations involving that variable. In this sense we treat races as "locally" catastrophic, and we track accurately the values of variables unaffected by races.

States

States are finite partial functions from identifiers and resource names to values. For simplicity we let V be the set of integers and use values 0 ("available") and 1 ("unavailable") for resources. Let $V^\top = V \cup \{\top\}$. Let $\mathbf{Var} = \mathbf{Ide} \cup \mathbf{Res}$. We use ι to range over $\mathbf{Var}$, i over $\mathbf{Ide}$ and r over $\mathbf{Res}$. We use a list-like notation $[\iota_1 : v_1, \ldots, \iota_k : v_k]$ for states, and we may also use set-theoretic notation such as $\{(\iota_1, v_1), \ldots, (\iota_k, v_k)\}$.

Definition 1 *The set Σ of states is given by:*

$$\Sigma = \{\sigma : \mathbf{Var} \rightharpoonup_{\mathit{fin}} V^\top \mid \forall r \in \mathit{dom}(\sigma) \cap \mathbf{Res}.\ \sigma(r) \in \{0,1\}\}.$$

We use σ and τ to range over states, and let $\mathit{res}(\sigma) = \mathit{dom}(\sigma) \cap \mathbf{Res}$ be the set of resources used in σ. For a set $X \subseteq \mathbf{Var}$ we let $\sigma \backslash X = \{(\iota, v) \in \sigma \mid \iota \notin X\}$ and $\sigma \upharpoonright X = \{(\iota, v) \in \sigma \mid \iota \in X\}$. When X is a singleton we write $\sigma\backslash\iota$ for $\sigma\backslash\{\iota\}$.

Definition 2 *A state σ is race-free if $\top \notin \mathit{rge}(\sigma)$.*

Definition 3 *Two states σ and τ are consistent, $\sigma \Uparrow \tau$, iff they agree on the values of all relevant variables, i.e. $\forall \iota \in \mathit{dom}(\sigma) \cap \mathit{dom}(\tau).\ \sigma(\iota) = \tau(\iota)$.*

Consistency is the same as requiring that $\sigma \upharpoonright dom(\tau) = \tau \upharpoonright dom(\sigma)$, or that $\sigma \cup \tau$ is also a well-defined state. States with disjoint domains are always consistent.

Definition 4 *We let* $[\sigma \mid \iota : v] = (\sigma \backslash \iota) \cup \{(\iota, v^{\top})\}$*, where* $v^{\top} = \top$ *if* $(\iota, \top) \in \sigma$*, and* $v^{\top} = v$ *otherwise. This is the state obtained by updating* σ *with* $\iota : v$*, unless* $\sigma(\iota) = \top$*, in which case the update has no effect.*

We generalize to multiple updates, writing $[\sigma \mid \tau]$ for the state obtained by updating σ with the updates in τ, and $[\sigma \mid X \mapsto \top]$ for updating σ with $\{(x, \top) \mid x \in X\}$. Updating is associative, i.e. $[[\sigma \mid \tau] \mid \rho] = [\sigma \mid [\tau \mid \rho]]$, so we can write $[\sigma \mid \tau \mid \rho]$ without ambiguity.

Steps

Steps represent the effect of state changes, and to model footprints of programs we record just the portion of state relevant to a step, rather than the entire global state. So a step will involve a pair of states (σ, σ'), where σ is the piece of state *read* and σ' is the piece of state *written*, and we decorate this pair with a *flow relation* $R \subseteq dom(\sigma) \times dom(\sigma')$ indicating in particular which reads influence the value of each write. We require that if $(\iota, \top) \in \sigma$ & $(\iota, \iota') \in R$ then $(\iota', \top) \in \sigma'$, since an update based on a tainted value is also deemed to be tainted. Further, if $\iota \in dom(\sigma')$ we insist that $(\iota, \iota) \in R$, since a write can only be performed if its target is present in the initial state. Coupled with the previous requirement this means that once a variable has been involved in a race its value never "recovers". Note that $dom(\sigma') \subseteq dom(R^{-1})$: for every variable ι written in the step the set $R^{-1}(\iota) \subseteq dom(\sigma)$ indicates the variables whose values in σ influence the update.

Definition 5 *The set* Λ *of steps consists of all triples* (σ, R, σ') *with* $\sigma, \sigma' \in \Sigma$*, such that:*

- $R \subseteq dom(\sigma) \times dom(\sigma')$.
- *For all* $\iota \in dom(\sigma')$*,* $(\iota, \iota) \in R$.
- *If* $(\iota, \top) \in \sigma$ *and* $(\iota, \iota') \in R$ *then* $(\iota', \top) \in \sigma'$.

We use λ and μ to range over the set of steps.

Definition 6 *For a step* $\lambda = (\sigma, R, \sigma')$*, let* $reads(\lambda) = dom(\sigma)$*,* $writes(\lambda) = dom(\sigma')$*, and* $res(\lambda) = dom(\sigma) \cap \mathbf{Res}$*. By assumption,* $writes(\lambda) \subseteq reads(\lambda)$*.*

For a step (σ, R, σ') and $\iota \in dom(\sigma')$, $R^{-1}(\iota)$ is the set of variables used to compute the update for ι. Since R is surjective, we can specify the dependency relation by listing $R^{-1}(\iota)$ for each $\iota \in dom(\sigma')$. We omit the flow relation when we intend the smallest relation satisfying the requirements, often the identity relation on $dom(\sigma')$ or the empty relation when $dom(\sigma') = \{\}$. Where relevant we use U for the universal relation on $dom(\sigma) \times dom(\sigma')$. It is helpful to introduce names for some simple steps.

In each case the intended flow relation is obvious:

$$\begin{aligned} \mathit{lock}(r) &= ([r:0],[r:1]) \\ \mathit{unlock}(r) &= ([r:1],[r:0]) \\ \mathit{read}(i,v) &= ([i:v],[\,]) \\ \mathit{write}(i,v,v') &= ([i:v],[i:v']) \\ \delta &= ([\,],[\,]) \end{aligned}$$

δ is an "idle" step. For steps in which multiple reads and writes occur, there may be several possible choices of flow relation, expressing different dependencies. For example, in the step $([x:0,y:0],\{(y,y)\},[y:1])$ the update to y does not depend on the read of x, whereas in $([x:0,y:0],\{(x,y),(y,y)\},[y:1])$ the update to y depends on both x and y. There are also racy steps such as $([x:0],[x:\top])$ in which a race condition affects the value of an identifier. Note that $([x:\top,y:0],[x:\top,y:1])$ is also a valid step, but that $([x:\top,y:0],[x:0,y:1])$ is not.

In the forthcoming semantic development, steps involving resources play a special rôle consistent with the underlying assumption that operations to lock and unlock resources are atomic actions. Steps with $\mathit{res}(\lambda) = \mathit{res}(\mu) = \{\}$ may be composable either consecutively or concurrently, to produce a single step representing the composite effect, whereas a step with $\mathit{res}(\lambda) \neq \{\}$ is atomic.

Executing steps

Although steps describe footprints, programs operate on a shared global state, and we need to characterize the effect on the global state of executing a step. Global states are also finite partial functions from variables to values, so Σ as defined before also represents the set of global states.

Given a state σ we can characterize the steps that are *executable* from σ, and their *effect*, with an enabling relation $\Rightarrow \subseteq \Sigma \times \Lambda \times \Sigma$. We write this as an infix relation, writing $\sigma \overset{\lambda}{\Longrightarrow} \sigma'$ when step λ is enabled from σ, and its execution causes the state to change to σ'. The definition is intuitive: a step can only be executed from a state consistent with its reads, and its effect is described by its writes, leaving all other variables unchanged. We say that state σ *enables* step λ if σ satisfies the read properties required to execute λ, i.e. if $\mathit{reads}(\lambda) \subseteq \sigma$.

Definition 7 *The enabling relation $\Rightarrow \subseteq \Sigma \times \Lambda \times \Sigma$ is given by:*

$$\sigma \xrightarrow{(\tau,R,\tau')} \sigma' \textit{ iff } \tau \subseteq \sigma \ \&\ \sigma' = [\sigma \mid \tau'].$$

Referring again to the examples introduced above, note the key facts that

$$\begin{array}{lll} \sigma \xRightarrow{read(i,v)} \sigma & & \text{iff } i : v \in \sigma \\ \sigma \xRightarrow{write(i,v,v')} [\sigma \mid i : v'] & & \text{iff } i : v \in \sigma \\ \sigma \xRightarrow{lock(r)} [\sigma \mid r : 1] & & \text{iff } r : 0 \in \sigma \\ \sigma \xRightarrow{unlock(r)} [\sigma \mid r : 0] & & \text{iff } r : 1 \in \sigma \end{array}$$

Also note that $lock(r)$ is not enabled in σ if $\sigma(r) = 1$, and $unlock(r)$ is not enabled in σ if $\sigma(r) = 0$. It is obvious that $\sigma \xRightarrow{\delta} \sigma$ always holds. In general when $\sigma \xRightarrow{\lambda} \sigma'$ we have $dom(\sigma') = dom(\sigma)$.

Consecutive steps

Two steps $\lambda = (\sigma, R, \sigma')$ and $\mu = (\tau, S, \tau')$ are *consecutive* (or sequentially executable) if their effects are composable, λ first followed by μ. This is the case when μ *follows* λ, i.e. μ can be enabled after λ, as characterized below.

Definition 8 *For steps $\lambda = (\sigma, R, \sigma')$ and $\mu = (\tau, S, \tau')$ we say that μ follows λ, written $\lambda \smile \mu$, if $\tau \Uparrow [\sigma \mid \sigma']$.*

The requirement that $\tau \Uparrow [\sigma \mid \sigma']$ says that the reads of μ are consistent with the effect of λ. This is equivalent to requiring that $\sigma \Uparrow (\tau \backslash dom(\sigma'))$ and $\tau \Uparrow \sigma'$. Note that the sequential composition operation on steps is *partial*, only defined on steps that satisfy the imposed constraints.

Definition 9 *When $\lambda = (\sigma, R, \sigma')$ and $\mu = (\tau, S, \tau')$ and $\lambda \smile \mu$, we define*

$$\lambda; \mu = (\sigma \cup (\tau \backslash dom(\sigma')),\ R; S,\ [\sigma' \mid \tau' \mid \rho]),$$

where $\rho = \{(\iota', \top) \mid \exists(\iota, \top) \in \sigma'.\ (\iota, \iota') \in S\}$, and where $R; S$ is the relation

$$\begin{array}{l} \{(\iota, \iota'') \mid \exists \iota'.\ (\iota, \iota') \in R \ \&\ (\iota', \iota'') \in S\} \\ \cup\ \{(\iota, \iota') \in R \mid \iota' \notin dom(S)\} \\ \cup\ \{(\iota', \iota'') \in S \mid \iota' \notin rge(R)\} \end{array}$$

The term ρ here propagates the effect of racy updates from the first step. If λ is race-free ρ degenerates to the empty state, and the write effect of the cumulative step is $[\sigma' \mid \tau']$ as expected. The flow relation $R; S$ composes R and S, bearing in mind that the first step may write to a variable not used in the second step, and the second step may read a variable not influenced by the first step.

Lemma 10 $\lambda_1 \smile \lambda_2$ & $(\lambda_1; \lambda_2) \smile \lambda_3$ *iff* $\lambda_2 \smile \lambda_3$ & $\lambda_1 \smile (\lambda_2; \lambda_3)$. *When these equations hold $\lambda_1; (\lambda_2; \lambda_3) = (\lambda_1; \lambda_2); \lambda_3$. Further, $\delta = ([\,], [\,])$ is a unit for sequential composition: $\lambda \smile \delta$, $\delta \smile \lambda$ and $\lambda; \delta = \delta; \lambda = \lambda$ for all $\lambda \in \Lambda$.*

Proof: An easy calculation using the definitions. For all relations R, S, T we have $(R; S); T = R; (S; T)$, and $R; \{\} = \{\}; R = R$ for all R.

Examples

In each of the following cases, the two steps are sequentially executable and we show the resulting step. As usual the intended flow relations are obvious:

$$\begin{aligned}
&\mathit{read}(x,v);\mathit{read}(x,v) = \mathit{read}(x,v)\\
&\mathit{read}(x,v_1);\mathit{read}(y,v_2) = ([x:v_1, y:v_2],[\,])\\
&\mathit{read}(x,v);\mathit{write}(x,v,v') = \mathit{write}(x,v,v')\\
&\mathit{write}(x,v,v');\mathit{read}(x,v') = \mathit{write}(x,v,v')\\
&\mathit{write}(x,v_1,v_2);\mathit{write}(x,v_2,v_3) = \mathit{write}(x,v_1,v_3)\\
&\mathit{write}(x,v_1,v_1');\mathit{write}(y,v_2,v_2') = ([x:v_1, y:v_2],[x:v_1', y:v_2'])
\end{aligned}$$

In particular, reads and writes to distinct identifiers have the same effect in either sequential order. Note also that writes to the same identifier are only sequentially composable if the value read by the second step agrees with the value written in the first step.

As expected, there is a simple relationship between sequential composition of steps and execution.

Lemma 11 *If $\lambda \smile \mu$, then $\sigma \xRightarrow{\lambda;\mu} \sigma''$ iff $\exists\sigma'.\ (\sigma \xRightarrow{\lambda} \sigma'\ \&\ \sigma' \xRightarrow{\mu} \sigma'')$.*

Concurrent steps

Steps λ and μ with $\mathit{res}(\lambda) \cap \mathit{res}(\mu) = \{\}$ and $\mathit{reads}(\lambda) \Uparrow \mathit{reads}(\mu)$ are said to be *concurrently executable*, or *concurrent* for short; they have compatible reads so there is a state from which both steps are enabled, and since they use different resources they can be executed simultaneously without violating the mutual exclusion constraints on resource use. The steps *conflict* if one writes to an identifier upon which the other one depends. When this happens we record the value of the race-sensitive identifier as $\top$, using the flow relations as a guide. In this way we obtain a composite step $\lambda \otimes \mu$ describing the concurrent combination of λ and μ. Again this operation on steps is *partial*, only defined for steps with compatible read requirements and needing disjoint resources.

Definition 12 *Two steps $\lambda = (\sigma, R, \sigma')$ and $\mu = (\tau, S, \tau')$ are concurrently executable, λ **co** μ, if $\sigma \Uparrow \tau$ & $\mathit{res}(\sigma) \cap \mathit{res}(\tau) = \{\}$. When this holds we define*

$$\lambda \otimes \mu = (\sigma \cup \tau,\ R \cup S,\ [(\sigma' \cup \tau') \mid X \mapsto \top]),$$

where X is the set of identifiers whose value is susceptible to a race condition, i.e. those identifiers written by one of the steps with a value dependent on an identifier affected by the other step:

$$\begin{aligned}
X &= \{\iota \in \mathit{dom}(\sigma') \mid R^{-1}(\iota) \cap \mathit{dom}(\tau') \neq \{\}\}\\
&\cup \{\iota \in \mathit{dom}(\tau') \mid S^{-1}(\iota) \cap \mathit{dom}(\sigma') \neq \{\}\}.
\end{aligned}$$

Since σ and τ are assumed to be compatible ($\sigma \Uparrow \tau$) the union $\sigma \cup \tau$ is a partial function, hence a valid state. The flow relation $R \cup S$ combines the flow information from the two steps in the obvious way. Even though the union $(\sigma' \cup \tau')$ may not be a partial function when $dom(\sigma') \cap dom(\tau') \neq \{\}$, the update $X \mapsto \top$ does produce a partial function, hence a valid state, since $X \supseteq dom(\sigma') \cap dom(\tau')$, $dom(\sigma') \subseteq dom(R^{-1})$, and $dom(\tau') \subseteq dom(S^{-1})$.

We can paraphrase the above definition, perhaps in a more readable manner, as saying that $\lambda \otimes \mu = (\sigma \cup \tau, R \cup S, \theta)$, where θ is the state with $dom(\theta) = dom(\sigma') \cup dom(\tau')$ and the following properties:

- $\theta(\iota) = \sigma'(\iota)$ if $\iota \in dom(\sigma')$ and $R^{-1}(\iota) \cap dom(\tau') = \{\}$;
- $\theta(\iota) = \tau'(\iota)$ if $\iota \in dom(\tau')$ and $S^{-1}(\iota) \cap dom(\sigma') = \{\}$;
- $\theta(\iota) = \top$ otherwise.

Lemma 13 λ_1 **co** λ_2 *iff* λ_2 **co** λ_1*, and when these hold* $\lambda_1 \otimes \lambda_2 = \lambda_2 \otimes \lambda_1$*. Further,* ($\lambda_1$ **co** λ_2 *and* $(\lambda_1 \otimes \lambda_2)$ **co** λ_3) *iff* (λ_2 **co** λ_3 *and* λ_1 **co** $(\lambda_2 \otimes \lambda_3)$)*, and when these hold* $(\lambda_1 \otimes \lambda_2) \otimes \lambda_3 = \lambda_1 \otimes (\lambda_2 \otimes \lambda_3)$*.*

It is also obvious that $\delta \otimes \lambda = \lambda \otimes \delta = \lambda$ always holds, so the idle step is a unit for concurrent composition.

Examples

- Reads and writes with compatible start states can be executed in parallel, and reads and writes to distinct variables x and y never conflict:

$$write(x, v_1, v_1') \otimes write(y, v_2, v_2') = ([x : v_1, y : v_2], [x : v_1', y : v_2'])$$
$$read(x, v_1) \otimes read(y, v_2) = ([x : v_1, y : v_2], [\,])$$
$$read(x, v_1) \otimes write(y, v_2, v_2') = ([x : v_1, y : v_2], [y : v_2'])$$
$$read(x, v_1) \otimes read(y, v_2) = ([x : v_1, y : v_2], [\,])$$

- For steps affecting the same single variable, concurrent reads are benign but concurrent writes constitute a race:

$$read(x, v) \otimes read(x, v) = read(x, v) = ([x : v], [\,])$$
$$write(x, v, v') \otimes read(x, v) = write(x, v, v') = ([x : v], [x : v'])$$
$$write(x, v, v_1') \otimes write(x, v, v_2') = ([x : v], [x : \top])$$

- A write to x concurrent with a read of x only constitutes a significant race if the read influences the value of some identifier. The following examples show how races get handled, specifically that the correct identifiers get tainted, when the flow relations are non-trivial. To facilitate comparison with the above definition we enumerate the flow relations explicitly.

(i) Consider a step that updates y and reads x but does not use the value of x in the update. If we concurrently write to x, the combined effect is to update both

x and y as intended. For example,

$$\begin{aligned}&([x:1,y:0],\{(y,y)\},[y:1])\otimes([x:1],\{(x,x)\},[x:2])\\&=([x:1,y:0],\{(x,x),(y,y),[x:2,y:1])\end{aligned}$$

(ii) Now consider a step that updates y using a value obtained by reading x. If we concurrently write to x the value of y is tainted. For example,

$$\begin{aligned}&([x:1,y:0],\{(x,y),(y,y)\},[y:1])\otimes([x:1],\{(x,x)\},[x:2])\\&=([x:1,y:0],\{(x,x),(x,y),(y,y)\},[x:2,y:\top])\end{aligned}$$

In each case the combined step accurately reflects the overall effect.

There is an obvious connection between concurrent composition and enabling.

Theorem 14
*When λ and μ have identity flow relations, λ **co** μ, and $\sigma \overset{\lambda\otimes\mu}{\Longrightarrow} \sigma'$ there are states σ_1, σ_2 such that $\sigma \overset{\lambda}{\Longrightarrow} \sigma_1, \sigma \overset{\mu}{\Longrightarrow} \sigma_2$, and σ' is uniquely determined by the following properties:*

- $\sigma'(\iota) = \sigma_1(\iota)$ *for* $\iota \in \mathit{writes}(\lambda) - \mathit{reads}(\mu)$
- $\sigma'(\iota) = \sigma_2(\iota)$ *for* $\iota \in \mathit{writes}(\mu) - \mathit{reads}(\lambda)$
- $\sigma'(\iota) = \top$ *for* $\iota \in \mathit{writes}(\lambda) \cap \mathit{reads}(\mu)$ *or* $\mathit{writes}(\mu) \cap \mathit{reads}(\lambda)$
- $\sigma'(\iota) = \sigma(\iota)$ *for* $\iota \in \mathit{dom}(\sigma) - (\mathit{writes}(\lambda) \cup \mathit{writes}(\mu))$.

A more general relationship is derivable, involving R and S in a natural manner.

Traces

Traces are finite sequences of steps, where each step is either an atomic resource action or represents the cumulative effect of a finite computation. We want to abstract away from the intermediate states in between resource actions, in the spirit of Dijkstra's Principle: *processes should be regarded as independent, except when they synchronize* [5]. The basic information about reads, writes and dependencies contained in a step is sufficient for this purpose, and we can work with traces in which adjacent resource-free steps are consecutively executable and mumbled together. Thus we abstract away from the order in which intermediate reads and writes occur, since the mumbled step only reports the "initial" reads and the "final" writes. In doing this we lose no generality when considering the effect on the shared state as viewed by other process running concurrently. The only way another process could be influenced by or affect an intermediate stage would be by reading or writing to a variable written by this step, causing a race condition. The mumbled step also writes to this variable, so we would also get a race condition under these circumstances and we lose no generality by keeping only the mumbled step.

To build traces with this reduced structure we introduce a modified form of concatenation, a partial operation · that combines traces whose concatenation can be properly reduced by such mumbling, and implements this reduction.

Definition 15 *λ precedes μ, written $\lambda \lhd \mu$, if $\lambda \smile \mu$ or $res(\lambda) \cup res(\mu) \neq \{\}$. When this holds we define $\lambda \cdot \mu$ to be the step given by:*

$$\begin{aligned} \lambda \cdot \mu &= \lambda\mu \quad \textit{if } res(\lambda) \cup res(\mu) \neq \{\} \\ &= \lambda; \mu \ \textit{if } res(\lambda) = res(\mu) = \{\} \textit{ and } \lambda \smile \mu \end{aligned}$$

We refer to this operation from now on simply as concatenation. We extend to finite traces in the obvious inductive manner. Let ϵ be the empty trace.

Definition 16 *For all $\alpha, \beta, \lambda, \mu$ we have $\epsilon \lhd \beta$, $\alpha \lhd \epsilon$, and $(\alpha\lambda) \lhd (\mu\beta)$ iff $\lambda \lhd \mu$. Further, $\epsilon \cdot \beta = \beta, \alpha \cdot \epsilon = \alpha$, and when $\lambda \lhd \mu$ we let $(\alpha\lambda) \cdot (\mu\beta) = \alpha(\lambda \cdot \mu)\beta$.*

It is easy to verify that $\delta \cdot \delta = \delta$ and that $\cdot$ is associative.

Theorem 17 *For all traces α, β, γ, $\alpha \lhd \beta$ & $(\alpha \cdot \beta) \lhd \gamma$ iff $\beta \lhd \gamma$ & $\alpha \lhd (\beta \cdot \gamma)$ and when these hold, $(\alpha \cdot \beta) \cdot \gamma = \alpha \cdot (\beta \cdot \gamma)$.*

Examples
Clearly $\mathit{write}(x, 0, 1) \cdot \mathit{write}(x, 1, 2) = \mathit{write}(x, 0, 2)$, and

$$\mathit{lock}(r) \cdot \mathit{write}(x, 0, 1) = \mathit{lock}(r)\ \mathit{write}(x, 0, 1)$$

$$\mathit{lock}(r) \cdot \mathit{write}(x, 0, 1) \cdot \mathit{write}(x, 1, 2) \cdot \mathit{unlock}(r) = \mathit{lock}(r)\ \mathit{write}(x, 0, 2)\ \mathit{unlock}(r)$$

Reducing traces
We say that a trace α is *reduced* (or "mumbled") iff for all pairs of successive steps $\lambda\mu$ in α either $res(\lambda) \neq \{\}$ or $res(\mu) \neq \{\}$. It is easy to see that any trace built using $\cdot$ is reduced, because when $\alpha\lambda$ and $\mu\beta$ are reduced traces and $\lambda \lhd \mu$, $\alpha(\lambda \cdot \mu)\beta$ is also a reduced trace. We say that $\lambda_1 \dots \lambda_n$ is *feasible* iff its steps can be combined using $\cdot$, and when this happens we obtain the reduced trace $red(\alpha) = \lambda_1 \cdot \ldots \cdot \lambda_n$.

For example, the trace

$$\alpha = \mathit{lock}(r)\ \mathit{write}(x, 0, 1)\ \mathit{write}(x, 1, 2)\ \mathit{unlock}(r)\ \mathit{lock}(r)\ \mathit{write}(x, 0, 1)\ \mathit{unlock}(r)$$

is feasible, and $red(\alpha)$ is the trace

$$\mathit{lock}(r)\ \mathit{write}(x, 0, 2)\ \mathit{unlock}(r)\ \mathit{lock}(r)\ \mathit{write}(x, 0, 1)\ \mathit{unlock}(r).$$

However the trace $\mathit{write}(x, 0, 2)\ \mathit{write}(x, 0, 1)$ is not feasible.

Every reduced trace α is also feasible and satisfies the equation $red(\alpha) = \alpha$. Whenever α is feasible, $red(\alpha)$ is a reduced trace.

Executing traces

We extend the effect relation to traces in the obvious way:

$$\begin{aligned} \sigma \xRightarrow{\lambda_1 \ldots \lambda_n} \sigma' \text{ iff } \exists \sigma_0, \ldots, \sigma_n.\ \sigma = \sigma_0 \ \&\ \sigma_n = \sigma' \\ \&\ \sigma_0 \xRightarrow{\lambda_1} \sigma_1 \xRightarrow{\lambda_2} \sigma_2 \cdots \ \sigma_{n-1} \xRightarrow{\lambda_n} \sigma_n \end{aligned}$$

When $n = 0$ we get $\sigma \overset{\epsilon}{\Longrightarrow} \sigma$. When $\sigma \overset{\alpha}{\Longrightarrow} \sigma'$ we say that α is enabled in σ, and σ' is the result of executing α from σ'. A trace is *executable* if it is enabled by some state.

We are mainly interested in executable traces, yet our semantics deals with *feasible* traces, including both executable and non-executable traces. This is necessary because a parallel program may have an executable trace that arises from non-executable traces of its constituent processes. For example, the traces $\alpha = \mathit{lock}(r)\,\mathit{lock}(r)$ and $\beta = \mathit{unlock}(r)\,\mathit{unlock}(r)$ are both non-executable, but they have an interleaving of the form $\mathit{lock}(r)\,\mathit{unlock}(r)\,\mathit{lock}(r)\,\mathit{unlock}(r)$, which is executable. The executable traces of $c_1\|c_2$ can be determined from the *feasible* traces of c_1 and c_2. Executable traces are also feasible, but the converse fails. For example, the trace

$$\mathit{lock}(r)\,\mathit{write}(x,0,1)\,\mathit{unlock}(r)\,\mathit{lock}(r)\,\mathit{write}(x,0,1)\,\mathit{unlock}(r)$$

is feasible but not executable. The claim that we lose no generality by dropping infeasible traces is validated by the facts that: when α is infeasible there are no states σ such that $\sigma \overset{\alpha}{\Longrightarrow} \sigma'$, and there is no way to fill the gaps in α without concurrently writing to a variable read by α.

Parallel composition

We adapt the definition of resource-sensitive fair merge from our prior work, adjusted to generate mumbled traces. For traces α and β, and disjoint finite sets of resources A and B, we define the set of feasible merges

$$\alpha\,{}_A\|_B\,\beta \subseteq \Lambda^*$$

by induction on trace length. This set consists of all traces in which a process holding resources A runs concurrently with a process holding resources B. We first define the relation $A \xrightarrow[B]{\lambda} A'$ that characterizes when the process holding A can do step λ:

$$\begin{array}{lll} A \xrightarrow[B]{\mathit{lock}(r)} A \cup \{r\} & \text{if } r \notin A \cup B \\ A \xrightarrow[B]{\mathit{unlock}(r)} A - \{r\} & \text{if } r \in A \\ A \xrightarrow[B]{\lambda} A & \text{if } \mathit{res}(\lambda) = \{\} \end{array}$$

When one (or both) of the traces is empty we define:

$$\begin{aligned} \alpha\,{}_A\|_B\,\epsilon &= \{\alpha \mid A \xrightarrow[B]{\alpha} A'\} \\ \epsilon\,{}_A\|_B\,\beta &= \{\beta \mid B \xrightarrow[A]{\alpha} B'\} \end{aligned}$$

For the inductive case we define:

$$\begin{aligned} (\lambda\alpha)\,{}_A\|_B\,(\mu\beta) = {} & \{\lambda\cdot\gamma \mid A \xrightarrow[B]{\lambda} A' \mathbin{\&} \gamma \in \alpha_{A'}\|_B\,(\mu\beta) \mathbin{\&} \lambda \triangleleft \gamma\} \\ & \cup \{\mu\cdot\gamma \mid B \xrightarrow[A]{\mu} B' \mathbin{\&} \gamma \in (\lambda\alpha)\,{}_A\|_{B'}\,\beta \mathbin{\&} \mu \triangleleft \gamma\} \\ & \cup \{(\lambda\otimes\mu)\cdot\gamma \mid \lambda \;\mathbf{co}\; \mu, \gamma \in \alpha\,{}_A\|_B\,\beta \mathbin{\&} (\lambda\otimes\mu)\triangleleft\gamma\} \end{aligned}$$

When $A = B = \{\}$ we omit the subscripts and write $\alpha \| \beta$.

The first two terms produce interleavings in which one process does a step first; the third term allows concurrent combinations when enabled, and takes account of the potential for race conditions. Use of $\cdot$ ensures that we only include feasible traces. The reader can check that when α and β are feasible traces, every trace belonging to $\alpha \| \beta$ is also feasible.

When λ and μ are concurrently enabled resource-free steps, $\lambda \| \mu = \{\lambda \otimes \mu\}$. If in addition the steps are conflict-free, we have $\lambda; \mu = \mu; \lambda = \lambda \otimes \mu$.

When λ and μ are resource-free steps that are not consecutively executable in either order, and not concurrently executable, $\lambda \| \mu = \{\}$.

When at least one of λ and μ is a resource step, $\lambda \| \mu = \{\lambda\mu, \mu\lambda\}$.

Examples

- The following examples check that the above definitions conform with intuition in cases involving concurrent writes. In each case we obtain the set of all (reduced) feasible combinations. Note that several distinct interleavings may lead to the same reduced trace, showing the succinctness of our construction.

(i) Concurrent writes to distinct variables: $\mathit{write}(x, 0, 1) \| \mathit{write}(y, 0, 1)$ yields three feesible interleavings:

$$\mathit{write}(x, 0, 1) \cdot \mathit{write}(y, 0, 1)$$
$$\mathit{write}(y, 0, 1) \cdot \mathit{write}(x, 0, 1)$$
$$\mathit{write}(x, 0, 1) \otimes \mathit{write}(y, 0, 1)$$

and each of these reduces to the same trace $([x : 0, y : 0], [x : 1, y : 1])$. So $\mathit{write}(x, 0, 1) \| \mathit{write}(y, 0, 1) = \{([x : 0, y : 0], [x : 1, y : 1])\}$.

(ii) Concurrently executable writes to the same variable:

$$\begin{aligned} &\mathit{write}(x, 0, 1) \| \mathit{write}(x, 0, 1) \\ &\quad = \{\mathit{write}(x, 0, 1) \otimes \mathit{write}(x, 0, 1)\} \\ &\quad = \{([x : 0], [x : \top])\}. \end{aligned}$$

In this case the updates are not sequentially composable.

(iii) Non-concurrently executable writes to the same variable:

$$\begin{aligned} &\mathit{write}(x, 0, 1) \| \mathit{write}(x, 1, 2) \\ &\quad = \{\mathit{write}(x, 0, 1) \cdot \mathit{write}(x, 1, 2)\} \\ &\quad = \{\mathit{write}(x, 0, 2)\} \end{aligned}$$

because there is only one feasible interleaving. On the other hand, $\mathit{write}(x, 0, 1) \| \mathit{write}(x, 2, 1) = \{\}$, because the two steps cannot be composed either sequentially or concurrently.

- Let α be the trace $\mathit{lock}(r)\mathit{write}(x, 0, 1)\mathit{unlock}(r)$. The trace set $\alpha \| \alpha$ is $\{\alpha\alpha\}$. Other interleavings begin with consecutive $\mathit{lock}(r)$ steps and fall afoul of the resource constraints built into the interleaving operator; such traces are irrelevant in any case because none are executable, and none can be used to construct an executable trace by interleaving. Similarly, let β be $\mathit{lock}(r)\ \mathit{write}(x,1,0)\ \mathit{unlock}(r)$. The set $\alpha \| \beta$ is just $\{\alpha\beta, \beta\alpha\}$. One interleaving of the (non-executable) trace $\alpha\alpha$ with β is the executable trace $\alpha\beta\alpha$.

Semantics of expressions

Since our semantics is designed to detect race conditions involving concurrent access to shared variables, we can work with a very simple model for expressions. There is no need to keep track of the order in which reads occur during expression evaluation, provided we record the set of reads on which the expression value depends, so we can use a set of state-value pairs. For an integer expression e we will let $[\![e]\!] \subseteq \Sigma \times V$, and for a boolean expression b we let $[\![b]\!] \subseteq \Sigma \times \{\mathit{true}, \mathit{false}\}$. An entry $(\sigma, v) \in [\![e]\!]$ represents the fact that evaluation of e in an τ such that $\tau \supseteq \sigma$ only reads the portion σ of the state and produces the integer value v, and similarly for boolean expressions. So each entry is a minimal piece of computational information about expression evaluation, in line with our desire to deal with footprints. We define $[\![e]\!]$ and $[\![b]\!]$ by structural induction as usual:

$$[\![n]\!] = \{([\,], n)\}$$

$$[\![i]\!] = \{([i : v], v) \mid v \in V\}$$

$$[\![e_1 + e_2]\!] = \{(\sigma_1 \cup \sigma_2, v_1 + v_2) \mid (\sigma_1, v_1) \in [\![e_1]\!] \& \ (\sigma_2, v_2) \in [\![e_2]\!] \ \& \ \sigma_1 \Uparrow \sigma_2\}$$

$$[\![\mathbf{true}]\!] = \{([\,], \mathit{true})\}$$

$$[\![e_1 = e_2]\!] = \{(\sigma_1 \cup \sigma_2, v_1 = v_2) \mid (\sigma_1, v_1) \in [\![e_1]\!] \ \& \ (\sigma_2, v_2) \in [\![e_2]\!] \& \ \sigma_1 \Uparrow \sigma_2\}$$

We remark that when $(\sigma, v) \in [\![e]\!]$ the piece of state σ contains values for the identifiers occurring free in e whose values are used to compute v.

Theorem 18 *For all expressions e, if $(\sigma, v) \in [\![e]\!]$ then $\mathit{dom}(\sigma) \subseteq \mathtt{free}(e)$.*

Semantics of commands

Commands denote sets of feasible traces. We define the trace set of a command, $[\![c]\!] \subseteq \mathcal{P}(\Lambda^*)$, by structural induction. To simplify the presentation we introduce the abbreviations $[\![b]\!]_{true}$ for $\{(\sigma, [\,]) \mid (\sigma, true) \in [\![b]\!]\}$ and similarly for $[\![b]\!]_{false}$.

$$[\![\mathbf{skip}]\!] = \{\delta\}$$

$$[\![i{:=}e]\!] = \{(\sigma \cup [i : v], U, [i : v']) \mid (\sigma, v') \in [\![e]\!] \;\&\; \sigma \Uparrow [i : v]\}$$

$$\text{where } U^{-1}(i) = dom(\sigma) \cup \{i\}$$

$$[\![c_1; c_2]\!] = [\![c_1]\!] \cdot [\![c_2]\!] = \{\alpha_1 \cdot \alpha_2 \mid \alpha_1 \in [\![c_1]\!],\ \alpha_2 \in [\![c_2]\!] \;\&\; \alpha_1 \lhd \alpha_2\}$$

$$[\![\mathbf{if}\ b\ \mathbf{then}\ c_1\ \mathbf{else}\ c_2]\!] = [\![b]\!]_{true} \cdot [\![c_1]\!] \ \cup\ [\![b]\!]_{false} \cdot [\![c_2]\!]$$

$$[\![\mathbf{while}\ b\ \mathbf{do}\ c]\!] = ([\![b]\!]_{true} \cdot [\![c]\!])^* \cdot [\![b]\!]_{false}$$

$$[\![c_1 \| c_2]\!] = [\![c_1]\!] \| [\![c_2]\!] = \bigcup \{\alpha_1 \| \alpha_2 \mid \alpha_1 \in [\![c_1]\!] \;\&\; \alpha_2 \in [\![c_2]\!]\}$$

$$[\![\mathbf{with}\ r\ \mathbf{when}\ b\ \mathbf{do}\ c]\!] = \{lock(r)\,\beta \cdot \gamma\, unlock(r) \mid \beta \in [\![b]\!]_{true},\ \gamma \in [\![c]\!],\ \beta \lhd \gamma\}$$

Note the careful use of $\lhd$ and $\cdot$ (crucially in the semantic clauses for sequential and parallel composition, but throughout the above presentation) to ensure that the trace set $[\![c]\!]$ consists of feasible traces. Also note the implicit use of the sequencing and concurrent composition operations on steps, to reduce traces and abstract away from the order of action occurrences in between resource steps. The executable traces of c can be extracted from $[\![c]\!]$ as a subset.

The semantic clause for critical regions shows how entry to and exit from a region are modeled as $lock(r)$ and $unlock(r)$, which together with the way we defined resource-sensitive interleaving ensures that the semantic clause for $c_1 \| c_2$ correctly models concurrent execution while obeying the atomicity and mutual exclusion constraints on resources. Since we only care here about partial correctness behavior there is no need to include infinite traces and no need to include busy waiting traces caused by unavailable resources and/or the falsity of the entry condition b. Similarly the clause for while-loops does not include infinite iteration.

The semantic clause for assignment conceals a slight subtlety: commands do not allocate storage during execution, so an assignment can only be executed from a state in which it target variable already has a value. So the footprint of $i{:=}e$ needs a read state in which i has a value and from which e can be evaluated. To allow for cases where i is not relevant to the value of e as well as for cases where e needs the value of i, we use $\sigma \cup [i : v]$ (with $\sigma \Uparrow [i : v]$) as the read state in the footprint of $i{:=}e$, where $(\sigma, v) \in [\![e]\!]$). When $i \notin dom(\sigma)$, so that i is not needed by e, the consistency constraint holds for all v; when $i \in dom(\sigma)$, so the value of e may depend on i, the consistency constraint only holds for $v = \sigma(i)$ and the state $\sigma \cup [i : v]$ is the same as σ.

Examples

(i) We illustrate how our semantics ignores the relative order of individual assignments and takes account of race conditions:

$$[\![x{:=}1]\!] = \{([x:v],[x:1]) \mid v \in V\}$$

$$[\![x{:=}1; x{:=}2]\!] = [\![x{:=}2]\!] = \{([x:v],[x:2]) \mid v \in V\}$$

$$[\![x{:=}1; y{:=}1]\!] = [\![x{:=}1 \| y{:=}1]\!] = \{([x:v_1, y:v_2],[x:1, y:1]) \mid v_1, v_2 \in V\}$$

$$[\![x{:=}1 \| x{:=}1]\!] = \{([x:v],[x:\top]) \mid v \in V\}$$

(ii) The next example illustrates the relevance of flow relations. It is easy to see that $[\![x{:=}1; y{:=}x]\!] = \{([x:v_1, y:v_2], R, [x:1, y:1]) \mid v_1, v_2 \in V\}$, where $R = \{(x,x),(y,y),(x,y)\}$. When we run this command in parallel with $x{:=}2$ there is a race condition involving x and y:

$$[\![(x{:=}1; y{:=}x) \| x{:=}2]\!] = \{([x:v_1, y:v_2],[x:\top, y:\top]) \mid v_1, v_2 \in V\}.$$

Our semantics does distinguish between $x{:=}1; y{:=}x$ and $x{:=}1; y{:=}1$, and this is necessary because in the latter command the value of y is not influenced by the value of x, leading to different behavior in contexts that write to x. Indeed, in contrast to the previous example, we have

$$[\![(x{:=}1; y{:=}1) \| x{:=}2]\!] = \{([x:v_1, y:v_2],[x:\top, y:1]) \mid v_1, v_2 \in V\}.$$

(iii) A slightly more complex example shows how expression evaluation and control flow fit in: $[\![\textbf{if } x > 0 \textbf{ then } y{:=}1 \textbf{ else } z{:=}1]\!]$ is the set

$$\{([x:v_1, y:v_2], U_y, [y:1]) \mid v_1 > 0, v_2 \in V\}$$
$$\cup \{([x:v_1, z:v_2], U_z, [z:1]) \mid v_1 \leq 0, v_2 \in V\}$$

where $U_y^{-1}(y) = \{x, y\}$ and $U_z^{-1}(z) = \{x, z\}$.

(iv) For a while-loop with no critical regions, our semantics abstracts away from the intermediate states generated by successive iterations:

$$[\![\textbf{while } y > 0 \textbf{ do } (x{:=}x+1; y{:=}y-1)]\!]$$
$$= \{([x:v_1, y:v_2], U, [x:v_1+v_2, y:0]) \mid v_2 > 0\} \cup \{([y:v],[\,]) \mid v \leq 0\}$$

where $U^{-1}(x) = \{x, y\}$.

(v) The only traces of the program $(\textbf{with } r \textbf{ do } x{:=}1) \| (\textbf{with } r \textbf{ do } x{:=}2)$ that are executable from a state in which the values of r and x are 0 are

$$\mathit{lock}(r)\, \mathit{write}(x,0,1)\, \mathit{unlock}(r)\, \mathit{lock}(r)\, \mathit{write}(x,1,2)\, \mathit{unlock}(r)$$
$$\text{and} \quad \mathit{lock}(r)\, \mathit{write}(x,0,2)\, \mathit{unlock}(r)\, \mathit{lock}(r)\, \mathit{write}(x,2,1)\, \mathit{unlock}(r).$$

(vi) Consider the program $(x{:=}1; x{:=}2) \| \textbf{with } r \textbf{ do } y{:=}x$. Each of its traces arises by interleaving a trace $([x:v],[x:2])$ of $x{:=}1; x{:=}2$ with a trace of form

$\mathit{lock}(r)\,([x : v_1, y : v_2], U_y, [x : v_1, y : v_1])\,\mathit{unlock}(r)$ for some v, v_1, v_2, where $U_y^{-1}(y) = \{x, y\}$. The feasible interleavings are all traces of the following forms:

- $([x : v], [x : 2])\,\mathit{lock}(r)\,([x : v_1, y : v_2], U_y, [x : v_1, y : v_1])\,\mathit{unlock}(r)$
- $\mathit{lock}(r)\,([x : v_1, y : v_2], U_y, [x : 2, y : \top])\,\mathit{unlock}(r)$
- $\mathit{lock}(r)\,([x : v_1, y : v_2], U_y, [x : 2, y : v_1])\,\mathit{unlock}(r)$
- $\mathit{lock}(r)\,([x : v_1, y : v_2], U_y, [x : 2, y : 2])\,\mathit{unlock}(r)$
- $\mathit{lock}(r)\,([x : v_1, y : v_2], U_y, [x : v_1, y : v_1])\,\mathit{unlock}(r)\,([x : v], [x : 2])$

There is no trace in which y gets set to 1; instead there is a racy trace in which y gets set to $\top$. The remaining traces represent computations in which steps are taken sequentially.

Note also that the traces of $(x{:=}1; x{:=}2)\|$**with** r **do** $y{:=}x$ are identical to the traces of $x{:=}2\|$**with** r **do** $y{:=}x$. This can be verified by construction, but it also follows by compositionality of the semantics, since $[\![x{:=}1; x{:=}2]\!]$ is the same as $[\![x{:=}2]\!]$.

(vii) Consider the program **with** r **do** $(x{:=}x + 1; x{:=}x + 1)$. Our semantics does not distinguish this from **with** r **do** $x{:=}x + 2$. There is no need to distinguish them, because no other process can tell them apart without causing a race.

Semantic properties

Theorem 19 *If $\alpha(\sigma, R, \sigma')\beta \in [\![c]\!]$ then $\mathit{dom}(\sigma) \subseteq \mathtt{free}(c)$, $\mathit{dom}(\sigma') \subseteq \mathtt{writes}(c)$, $R \subseteq \mathit{dom}(\sigma) \times \mathit{dom}(\sigma')$, and $\mathit{res}(\sigma) \subseteq \mathtt{res}(c)$.*

As a corollary, if c synchronization-free, i.e. $\mathtt{res}(c) = \{\}$, the traces of c are singletons consisting of single step (τ, R, τ'). Such a command determines a state transformation, expressible as

$$|c| = \{(\sigma, \sigma') \mid \exists(\tau, R, \tau') \in [\![c]\!].\ \sigma \subseteq \tau \ \&\ \sigma' = [\sigma \mid \tau']\}.$$

If in addition to this c has no free variables, the only possible non-empty trace for c is $([\,], [\,])$, and $|c|$ is either the identity function on states or the empty function. For example, $|$**skip**; **skip**$| = \{(\sigma, \sigma) \mid \sigma \in \Sigma\}$ and $|$**while true do skip**$| = \{\}$.

Example 3.1 Suppose $x \notin \mathtt{free}(e_2)$ and $y \notin \mathtt{free}(e_1)$. Then

$$[\![x{:=}e_1; y{:=}e_2]\!] =$$

$$\{(\sigma_1 \cup \sigma_2, R, [x : v_1, y : v_2]) \mid (\sigma_1, v_1) \in [\![e_1]\!],\ (\sigma_2, v_2) \in [\![e_2]\!],\ \sigma_1 \Uparrow \sigma_2,$$
$$R^{-1}(x) = \mathit{dom}(\sigma_1)\ \&\ R^{-1}(y) = \mathit{dom}(\sigma_2)\}$$

It follows that $[\![x{:=}e_1; y{:=}e_2]\!] = [\![y{:=}e_2; x{:=}e_1]\!] = [\![x{:=}e_1 \| y{:=}e_2]\!]$.

Recall that a state σ is race-free iff $\top \notin \mathit{rge}(\sigma)$.

Definition 20 *c is race-free from σ iff $\forall\alpha \in [\![c]\!].\forall\sigma'.\ (\sigma \overset{\alpha}{\Longrightarrow} \sigma'$ implies σ' race-free). When this holds for all states σ we say that c is race-free.*

Theorem 21

- **skip** *and $i{:=}e$ are race-free.*

- *If c_1 and c_2 are race-free, so are $c_1; c_2$ and* **if** b **then** c_1 **else** c_2.
- *If c is race-free, so are* **while** b **do** c *and* **with** r **when** b **do** c.
- *If* $\mathtt{writes}(c_1) \cap \mathtt{free}(c_2) = \mathtt{writes}(c_2) \cap \mathtt{free}(c_1) = \{\}$,
  and c_1 and c_2 are race-free, then $c_1 \| c_2$ is race-free.

Corollary Every sequential program c is race-free.

A resource context Γ is a set of entries of form $r(X)$, where r is a resource name and X is a finite set of identifiers, called a protection list. We say that c respects Γ iff every free occurrence in c of an identifier protected by r in Γ is inside a conditional critical region naming r. The next result shows that our semantics validates the Owicki-Gries static constraints on critical variables [9].

Theorem 22
Let c_1 and c_2 be race-free commands. If c_1 and c_2 respect Γ and

$$\mathtt{writes}(c_1) \cap \mathtt{free}(c_2) \subseteq \mathtt{owned}(\Gamma) \ \& \ \mathtt{writes}(c_2) \cap \mathtt{free}(c_1) \subseteq \mathtt{owned}(\Gamma)$$

then $c_1 \| c_2$ is race-free.

Safe partial correctness

Let p and q be boolean-valued expressions in which identifiers may occur free. (We do not allow resource names to appear free in such expressions.) Let $|p|$ be the set of states satisfying p.

The *safe partial correctness* formula $\{p\}c\{q\}$ is *valid* iff all finite executions of c from a state satisfying p are race-free and end in a state satisfying q.

Definition 23 *$\{p\}c\{q\}$ is valid for all $\alpha \in [\![c]\!]$, and all states σ such that $dom(\sigma) \supseteq$ $\mathtt{free}(p, c, q)$, if $\sigma \in |p|$ and $\sigma \xRightarrow{\alpha} \sigma'$, then σ' is race-free and $\sigma' \in |q|$.*

Theorem 24 *If $[\![c_1]\!] = [\![c_2]\!]$ then c_1 and c_2 satisfy the same assertions in all program contexts, i.e. for all p and q, and all contexts $C[-]$, $\{p\}C[c_1]\{q\}$ is valid iff $\{p\}C[c_2]\{q\}$ is valid.*

Corollary 25 *If $[\![c_1]\!] = [\![c_2]\!]$ then c_1 and c_2 satisfy the same assertions, i.e. for all p and q, $\{p\}c_1\{q\}$ is valid iff $\{p\}c_2\{q\}$ is valid.*

Observing the state

For a trace α, let $|\alpha|$ be the set of state sequences obtainable by executing α. When $\alpha = \lambda_1 \dots \lambda_n$ we have $|\alpha| = \{\sigma_1 \dots \sigma_n \mid \sigma_1 \xRightarrow{\lambda_1} \sigma_2 \cdots \sigma_{n-1} \xRightarrow{\lambda_n} \sigma_n\}$. When α is non-executable this set is obviously empty. We define the set $\mathcal{S}(c)$ of state sequences observable during executions of c, to be $\mathcal{S}(c) = \bigcup\{|\alpha| \mid \alpha \in [\![c]\!]\}$. This is observing the state at start and finish and at all synchronization points, since we are dealing here with mumbled traces.

Theorem 26 *If $[\![c_1]\!] = [\![c_2]\!]$ then for all contexts $C[-]$, $\mathcal{S}(C[c_1]) = \mathcal{S}(C[c_2])$.*

4 Conclusions

Our more refined treatment of races leads to a footprint trace semantics that supports compositional reasoning about safe partial correctness, and accurately tracks the variables whose values are immune to race conditions. The new model enjoys better abstraction properties than the earlier version of footprint semantics [4]. We can pinpoint the differences by looking at the notions of semantic equivalence induced by the old semantics and the new. In the earlier model a race condition was interpreted as a global catastrophe, leading to a special **abort** state. This earlier model gave the same meaning, namely

$$\{([x : v_1, y : v_2], [x : 1, y : 1]) \mid v_1, v_2 \in V\},$$

to (i) $x{:=}1; y{:=}x$ and (ii) $x{:=}1; y{:=}1$, and hence the same meaning, namely

$$\{([x : v_1, y : v_2], \mathbf{abort}) \mid v_1, v_2 \in V\},$$

to programs (i)$\|x{:=}2$ and (ii)$\|x{:=}2$. The new semantics distinguishes between (i) and (ii), because they induce different flow relations: only in (i) is the value of y dependent on x. (We made a similar comment in an example, earlier.) Moreover, this distinction is worth making, because when run in parallel with a program that (only) writes to x, for (i) the value of y gets tainted but for (ii) it does not. Thus there is a program context in which (i) and (ii) have different observable behavior. In particular, the new semantics distinguishes (correctly) between the programs (i)$\|x{:=}2$ and (ii)$\|x{:=}2$.

We refer to our semantics as "grainless". We argue that this appellation is accurate, because the steps employed in the traces of programs are obtained by "mumbling" consecutive actions together in between resource actions, so that we abstract away from irrelevant details concerning scheduling and atomicity. We do still assume that resource actions $lock(r)$ and $unlock(r)$ are implemented atomically. Further, we only include feasible traces, leading to a more succinct trace set that subscribes roughly speaking to the slogan: fewer traces, shorter traces, bigger steps. Thus our semantics strives to avoid the combinatorial explosion inherent in interleaving traces, by working with smaller trace sets and shorter traces. Our more localized account of races allows a more liberal view of safe partial correctness. We could say that $\{p\}c\{q\}$ is valid iff for all states σ satisfying p, and all $\alpha \in [\![c]\!]$, if $\sigma \overset{\alpha}{\Longrightarrow} \sigma'$ then σ' satisfies q. Assuming that we define satisfaction so that $\sigma \in |p|$ implies $\forall i \in \texttt{free}(p).\ \sigma(i) \neq \top$, this notion of validity does not imply absolute race-freedom of c from states satisfying the pre-condition, just that the state relevant to the post-condition is race-free. We plan to explore this idea further, perhaps leading to a new variant of concurrent separation logic that deals more flexibly with race conditions.

We dealt here with a simple shared-memory parallel language: no pointers or mutable state, no dynamic allocation of storage. We believe that the foundations laid in our prior development of *action trace* semantics for concurrent separation logic [3] and our earlier grainless model [4] can be adapted to combine the new approach to race-detection and tainting with mutable state, and we plan to investigate further. We also intend to explore existing work using flow analysis in other settings, such as secure information flow and program analysis. Insights from related fields may help us to assess the scope and limitations of our work, and may suggest some improvements.

We have so far assumed that programs are executed on an architecture that provides sequential memory consistency [7]:

> ...the result of any execution is the same as if the operations of all the processors were executed in some sequential order, and the operations of each individual processor appear in this sequence in the order specified by its program.

This assumption leads naturally to the use of interleaving to represent parallel composition, a common feature in denotational models of shared-memory dating back to Park [10]. We have not yet applied our semantic framework to deal with weaker memory assumptions, such as those actually supported by some modern concurrency architectures [1]; it would be interesting to see to what extent our semantics can help support reasoning about language implementations that offer weaker guarantees about the observable effect of concurrent updates. In any case we would argue that even in (perhaps especially in) dealing with more relaxed implementations it would be useful to have a semantics like ours, that yields a (machine-independent) characterization of race-free programs and predicts (semantically) to what extent a program's behavior is susceptible to uncertainty about the values of variables.

Acknowledgements

We thank the referees for some helpful suggestions and pointers to potentially interesting related work.

References

[1] S. V. Adve and K. Gharachorloo. *Shared Memory Consistency Models: A Tutorial.* IEEE Computer 29 (12): 66–76 (December 1996).

[2] S. Brookes. *Full abstraction for a shared-variable parallel language.* Proc. 8th IEEE Symposium on Logic in Computer Science, IEEE Computer Society Press (1993), 98–109. Journal version in: *Inf. Comp.*, vol 127(2):145-163, Academic Press, June 1996.

[3] S. Brookes. *A Semantics for Concurrent Separation Logic.* Invited paper, CONCUR 2004, Philippa Gardner and Nobuko Yoshida (Eds.), Springer LNCS 3170, London, August/September 2004, pp. 16-34.

[4] S. Brookes. *A Grainless Semantics for Parallel Programs with Shared Mutable State.* Proc. 21^{st} Conference on Mathematical Foundations of Programming Semantics (MFPS XXI). Birmingham, 17–21 May, 2005.

[5] E. W. Dijkstra. *Cooperating sequential processes.* In: **Programming Languages**, F. Genuys (editor), pp. 43-112. Academic Press, 1968.

[6] C.A.R. Hoare. *Towards a Theory of Parallel Programming.* In **Operating Systems Techniques**, C. A. R. Hoare and R. H. Perrott, editors, pp. 61-71, Academic Press, 1972.

[7] L. Lamport. *How to Make a Multiprocessor Computer That Correctly Executes Multiprocess Programs*, IEEE Trans. Comput. C-28,9 (Sept. 1979), pp. 690-691.

[8] P.W. O'Hearn. *Resources, Concurrency, and Local Reasoning.* Invited paper, CONCUR 2004, Philippa Gardner and Nobuko Yoshida (Eds.), Springer LNCS 3170, London, August/September 2004, pp. 49-67.

[9] S. Owicki and D. Gries. *Verifying properties of parallel programs: An axiomatic approach*, Comm. ACM. 19(5):279-285, May 1976.

[10] D. Park. *On the semantics of fair parallelism.* In: **Abstract Software Specifications**, Springer-Verlag LNCS vol. 86, 504–526, 1979.

[11] J. C. Reynolds. *Towards a Grainless Semantics for Shared-Variable Concurrency.* Proc. 24th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2004), Chennai, India. Springer-Verlag, December 2004.

Strategies as Concurrent Processes

Simon Castellan[a] Jonathan Hayman[a] Marc Lasson[b]
Glynn Winskel[a]

[a] Computer Laboratory, University of Cambridge, UK

[b] INRIA Paris-Rocquencourt, PiR2, Univ Paris Diderot,
Sorbonne Paris Cité F-78153, Le Chesnay, France

Abstract

Concurrent strategies are shown to support operations yielding an economic yet rich higher-order concurrent process language, which shares features both with process calculi and nondeterministic dataflow. Its operational semantics and 'may and must' equivalence require that we take internal (neutral) moves seriously, leading to the introduction of 'partial strategies' which may contain neutral moves. Through partial strategies, we can present a transition semantics for a language of strategies and can formulate their 'may and must' behaviour. While partial strategies compose, in a way extending that of strategies, in general composition introduces extra neutral moves; in particular, copy-cat is no longer strictly an identity w.r.t. composition. However, a simple extension of concurrent strategies (with stopping configurations) maintains the fact that they form a bicategory while still capturing 'may and must' behaviour.

1 Introduction

There are several reasons for extending games and strategies, with behaviour based on trees, to concurrent games and strategies, based on event structures — the concurrent analogue of trees. One reason is to provide a foundation for a generalized domain theory, in which concurrent games and strategies take over the roles of domains and continuous functions. The motivation is to repair the divide between denotational and operational semantics and tackle anomalies like nondeterministic dataflow, which are beyond traditional domain theory. Another is that strategies are as potentially fundamental as relations and functions. It is surely because of our limited mental capacity, and not because of its unimportance, that the mathematical concept of strategy has been uncovered relatively late. It is hard to think about the successive contingencies involved in playing a game in the same way that is hard to think about interacting processes. Developing strategies in the extra generality demanded by concurrency reveals more clearly their essential nature and enables us to harness computer-science expertise in structure and concurrency in their understanding and formalization.

The extra generality of concurrency reveals new structure and a mathematical robustness to the concept of strategy, in particular showing strategies are essentially special profunctors [19]. Profunctors themselves provide a rich framework in which to generalize domain theory, in a way that is arguably closer to that initiated by Dana Scott than game semantics [9,2]. However, the mathematical abstraction of profunctors comes at a price: it can be hard to give an operational reading to denotations as profunctors. There are examples of semantics of higher-order process languages and "strong correspondence" where elements of profunctor denotations correspond to derivations in an operational semantics [11,15]. But in general it is hard to extract operational semantics from the profunctor denotations alone because they have abstracted too far.

Connections between forms of strategy and process models have been the subject of a large body of prior research — see *e.g.* [8,6,7]. In this paper, we begin a study of concurrent strategies from the perspective of concurrent processes, considering how concurrent games and strategies are objects which we can program. They are shown to support operations yielding an economic yet rich higher-order concurrent process language, which shares features both with process calculi and nondeterministic dataflow. The operations allow recursion to be interpreted using a trace and novel duplication operation.

Indeed, seen from a concurrent-process perspective, in some respects concurrent strategies have abstracted too far. Both in obtaining a transition semantics for the language of strategies and in analysing its behaviour w.r.t. 'may and must' testing, we need to take internal (neutral) moves, introduced in the composition of strategies, seriously. Through the more refined model of *partial* concurrent strategies we obtain a correspondence between events of a strategy and derivations of atomic steps in a transition semantics. Via partial strategies we can justify a simple extension of concurrent strategies (with stopping configurations) which maintains the fact that they form a bicategory, while still capturing 'may and must' behaviour.

2 Event structures and their maps

An *event structure* comprises $(E, \mathrm{Con}, \leq)$, a set E of *events* which are partially ordered by $\leq$, the *causal dependency relation*, and a nonempty *consistency relation* Con consisting of finite subsets of E, which satisfy

$$\{e' \mid e' \leq e\} \text{ is finite for all } e \in E \qquad \{e\} \in \mathrm{Con} \text{ for all } e \in E$$

$$Y \subseteq X \in \mathrm{Con} \implies Y \in \mathrm{Con} \qquad X \in \mathrm{Con} \ \& \ e \leq e' \in X \implies X \cup \{e\} \in \mathrm{Con}.$$

The *configurations*, $\mathcal{C}^\infty(E)$, of an event structure E consist of those subsets $x \subseteq E$ which satisfy $\forall X \subseteq x.\ X$ is finite $\Rightarrow X \in \mathrm{Con}$ (*consistency*) and $\forall e, e'.\ e' \leq e \in x \implies e' \in x$. (*down-closure*). Often we shall be concerned with just the finite configurations of an event structure. We write $\mathcal{C}(E)$ for the *finite* configurations of an event structure E.

We say an event structure is *elementary* when the consistency relation consists of all finite subsets of events. Two events which are both consistent and incomparable w.r.t. causal dependency in an event structure are regarded as *concurrent.* In games the relation of *immediate* dependency $e \rightarrowtriangle e'$, meaning e and e' are distinct with $e \leq e'$ and no event in between, will play a very important role. For $X \subseteq E$ we write $[X]$ for $\{e \in E \mid \exists e' \in X.\ e \leq e'\}$, the down-closure of X; note if $X \in \mathrm{Con}$, then $[X] \in \mathrm{Con}$. We write $[a]$ for $[\{a\}]$ where $a \in E$. For configurations x, y, we use $x {-\!\!\!\subset} y$ to mean y covers x, *i.e.* $x \subset y$ with nothing in between, and $x \overset{e}{\longrightarrow\!\!\!\!\subset} y$ to mean $x \cup \{e\} = y$ for an event $e \notin x$. We sometimes use $x \overset{e}{\longrightarrow\!\!\!\!\subset}$, expressing that event e is enabled at configuration x, when $x \overset{e}{\longrightarrow\!\!\!\!\subset} y$ for some configuration y.

Certain 'structural' maps of event structures, which have a long history [14], play a key role in the development of nondeterministic concurrent strategies. A *map* of event structures $f : E \to E'$ is a partial function $f : E \rightharpoonup E'$ such that $fx \in \mathcal{C}^\infty(E')$ for all $x \in \mathcal{C}^\infty(E)$, and for all $e_1, e_2 \in x$

$$f(e_1) = f(e_2)\ \&\ f(e_1), f(e_1) \text{ both defined} \implies e_1 = e_2\,.$$

Above, it is sufficient to restrict to finite configurations. Note that, when f is total, it restricts to a bijection $x \cong fx$ for any $x \in \mathcal{C}^\infty(E)$. A total map is *rigid* when it preserves causal dependency.

A map $f : E \to E'$ of event structures has *partial-total factorization* as a composition $E \xrightarrow{p} E \downarrow V \xrightarrow{t} E'$ where $V =_{\mathrm{def}} \{e \in E \mid f(e) \text{ is defined}\}$ is the domain of definition of f; the event structure $E{\downarrow}V =_{\mathrm{def}} (V, \leq_V, \mathrm{Con}_V)$, where $v \leq_V v'$ iff $v \leq v'\ \&\ v, v' \in V$ and $X \in \mathrm{Con}_V$ iff $X \in \mathrm{Con}\ \&\ X \subseteq V$; the *partial* map $p : E \to E \downarrow V$ acts as identity on V and is undefined otherwise; and the *total* map $t : E \downarrow V \to E'$, called the *defined part* of f, acts as f. The event structure $E{\downarrow}V$ is the *projection* of E to V.

It shall be convenient to construct event structures using *rigid families.*

Proposition 2.1 *Let $\mathcal{Q}$ be a non-empty family of finite partial orders closed under rigid inclusions, i.e. if $q \in \mathcal{Q}$ and $q' \hookrightarrow q$ is a rigid inclusion (regarded as a map of elementary event structures) then $q' \in \mathcal{Q}$. The family $\mathcal{Q}$ determines an event structure* $\mathrm{Pr}(\mathcal{Q}) =_{\mathrm{def}} (P, \leq, \mathrm{Con})$ *as follows:*

- *the events P are primes, i.e. finite partial orders in $\mathcal{Q}$ with a top element;*
- *the causal dependency relation $p' \leq p$ holds precisely when there is a rigid inclusion from $p' \hookrightarrow p$;*
- *a finite subset $X \subseteq P$ is consistent,* $X \in \mathrm{Con}$*, iff there is $q \in \mathcal{Q}$ and rigid inclusions $p \hookrightarrow q$ for all $p \in X$.*

If $x \in \mathcal{C}(P)$ then $\bigcup x$, the union of the partial orders in x, is in $\mathcal{Q}$. The function $x \mapsto \bigcup x$ is an order-isomorphism from $\mathcal{C}(P)$, ordered by inclusion, to $\mathcal{Q}$, ordered by rigid inclusions.

$$\begin{tikzcd}
& P \arrow[dl, "\pi_1"'] \arrow[dr, "\pi_2"] \arrow[dd, phantom, "\vee"] & \\
A \arrow[dr, "f"'] & & B \arrow[dl, "g"] \\
& C\,. &
\end{tikzcd}$$

Pullbacks of total maps Maps $f : A \to C$ and $g : B \to C$ have pullbacks in the category of event structures, and are simple to describe in the case where f and g are total. In this situation, finite configurations of P correspond to the composite bijections

$$\theta : x \cong fx = gy \cong y$$

between configurations $x \in \mathcal{C}(A)$ and $y \in \mathcal{C}(B)$ s.t. $fx = gy$ for which the transitive relation generated on θ by $(a,b) \leq (a',b')$ if $a \leq_A a'$ or $b \leq_B b'$ is a partial order; the correspondence taking $z \in \mathcal{C}(P)$ to the composite bijection $\pi_1 z \cong f\pi_1 z = g\pi_2 z \cong \pi_2 z$ respects inclusion.

2.1 Affine maps

In considering the dynamics of processes we shall need to relate a process to a process it may become. For this we generalize the earlier structural maps of event structures to *affine* maps, in which we need no longer preserve the empty configuration [16].

Definition 2.2 Let A be an event structure. Let $x \in \mathcal{C}^\infty(A)$. Write A/x for the event structure which remains after the occurrence of x. Precisely, A/x comprises

- events, $\{a \in A \smallsetminus x \mid x \cup [a]_A \in \mathcal{C}^\infty(A)\}$,
- consistency relation, $X \in \text{Con}$ iff $X \subseteq_{\text{fin}} A/x$ & $x \cup [X]_A \in \mathcal{C}^\infty(A)$, and
- causal dependency, the restriction of that on A.

We extend the notation to configurations regarding them as elementary event structures. If $y \in \mathcal{C}^\infty(A)$ with $x \subseteq y$ then by y/x we mean the configuration $y \smallsetminus x \in \mathcal{C}^\infty(A/x)$. In the case of a singleton configuration $\{a\}$ of A — when a is an *initial* event of A — we shall often write A/a and x/a instead of $A/\{a\}$ and $x/\{a\}$.

An *affine* map of event structures f from A to B comprises a pair (f_0, f_1) where $f_0 \in \mathcal{C}(B)$ and f_1 is a map of event structures $f_1 : A \to B/f_0$. It determines a function from $\mathcal{C}(A)$ to $\mathcal{C}(B)$ given by $fx = f_0 \cup f_1 x$ for $x \in \mathcal{C}(A)$. The allied f_0 and f_1 can be recovered from the action of f on configurations: $f_0 = f\varnothing$ and f_1 is that unique map of event structures $f_1 : A \to B/f\varnothing$ which on configurations $x \in \mathcal{C}(A)$ returns $fx/f\varnothing$. It is simplest to describe the composition gf of affine maps $f = (f_0, f_1)$ from A to B and $g = (g_0, g_1)$ from B to C in terms of its action on configurations: the composition takes a configuration $x \in \mathcal{C}(A)$ to $g(f\,x)$. Alternatively, the composition gf can be described as comprising $(g_0 \cup g_1 f_0, h)$ where h is that unique map of event structures $h : A \to C/(g_0 \cup g_1 f_0)$ which sends $x \in \mathcal{C}(A)$ to $g_1(f_0 \cup f_1 x)/g_1 f_0$. Note that traditional maps can be identified with those special affine maps (f_0, f_1) in which $f_0 = \varnothing$. We reserve the term 'map' for the traditional structural maps of event structure and shall say explicitly when a map is affine.

3 Concurrent games and strategies

A *game* is represented by an event structure A in which an event $a \in A$ carries a polarity $pol(a)$, + for Player and − for Opponent. Maps and affine maps of event structures extend to event structures with polarity: their underlying partial functions on events are required, where defined, to preserve polarity. A number of constructions on games will play an important role in the coming semantics:

Dual $A^\perp$, of an event structure with polarity A is a copy of the event structure A with a reversal of polarities.

Simple parallel composition $A \| B$, by juxtaposition. Its unit is the empty game

$\varnothing$. More generally, we can define, $\|_{i \in I} A_i$, the simple parallel composition of a indexed family of games, in which the set of events comprises the disjoint union $\bigcup_{1 \leq i \leq m} \{i\} \times A_i$.

Sums and recursive definitions Sums $\Sigma_{i \in I} A_i$ of an indexed family of games, which coincide with coproducts in the categories of event structures, are obtained in a similar way to simple parallel compositions, but now with events from distinct components being inconsistent (*i.e.* no set in the consistency relation contains elements from distinct components). We shall not make explicit use of recursively-defined games, but they are dealt with in exactly the same way as recursively-defined event structures [14].

3.1 A bicategory of games and strategies

A (nondeterministic concurrent) *strategy in* a game A is represented by a total map of event structures $\sigma : S \to A$ which preserves polarities and is

Receptive: if $\sigma x \overset{a}{\longrightarrow\!\!\subset}$ & $pol_A(a) = -$ then there is unique $s \in S$ s.t. $x \overset{s}{\longrightarrow\!\!\subset}$ & $\sigma(s) = a$;
Innocent: if $s \rightarrowtriangle_S s'$ & $(pol(s) = + \ or \ \ pol(s') = -)$ then $\sigma(s) \rightarrowtriangle_A \sigma(s')$.

Receptivity expresses that Player cannot hinder moves of Opponent, while innocence says a strategy should only adjoin immediate causal dependencies of the form $\ominus \rightarrowtriangle \oplus$. A map between strategies from $\sigma : S \to A$ to $\sigma' : S' \to A$ is a total map $f : S \to S'$ of event structures with polarity such that $\sigma = \sigma' f$. Accordingly, the strategies are isomorphic iff f is an isomorphism of event structures.

The conditions of receptivity and innocence are necessary and sufficient to ensure that the copy-cat strategy behaves as identity w.r.t. composition [12], which we now proceed to define.

We follow Conway and Joyal, and define a *strategy from* a game A *to* a game B, written $\sigma : A \mathrel{+\!\!\!\!\!\rightarrow} B$, as a strategy σ in the game $A^{\perp} \| B$.

Let $\sigma : S \to A^{\perp} \| B$, $\tau : T \to B^{\perp} \| C$ be strategies. Their composition is defined via the pullback drawn below. Ignoring polarities, the composite partial map has defined part $T \odot S$, which yields the composition of strategies $\tau \odot \sigma : T \odot S \to A^{\perp} \| C$ once polarities are reinstated.

$$\begin{tikzcd}
 & A \parallel T \arrow[dr, "A \parallel \tau"] & & \\
P \arrow[ur, "\pi_2"] \arrow[dr, "\pi_1"'] \arrow[r, phantom, ">"] & & A \parallel B \parallel C \arrow[r, rightharpoonup] & A \parallel C \\
 & S \parallel C \arrow[ur, "\sigma \parallel C"'] & &
\end{tikzcd}$$

Let A be a game. The copy-cat strategy from A to A is a total map $\gamma_A : \mathrm{C\!\!C}_A \to A^{\perp} \| A$, based on the idea that Player moves, of +ve polarity, always copy previous corresponding moves of Opponent, of −ve polarity. For $c \in A^{\perp} \| A$ we use $\overline{c}$ to mean the corresponding copy of c, of opposite polarity, in the alternative component. Define $\mathrm{C\!\!C}_A$ to comprise the event structure with polarity $A^{\perp} \| A$ together with extra causal dependencies $\overline{c} \leq_{\mathrm{C\!\!C}_A} c$ for all events c with $pol_{A^{\perp} \| A}(c) = +$. A finite subset of $\mathrm{C\!\!C}_A$ is consistent if its down-closure is consistent in $A^{\perp} \| A$.

The characterisation of configurations of $\mathrm{C\!\!C}_A$ reveals an important partial order on configurations of A. Let x and y be configurations of an event structure with polarity. Write $x \subseteq^{-} y$ to mean $x \subseteq y$ and $pol(y \smallsetminus x) \subseteq \{-\}$, *i.e.* the configuration y extends the configuration x solely by events of −ve polarity. Similarly, $x \subseteq^{+} y$ means $x \subseteq y$ and $pol(y \smallsetminus x) \subseteq \{+\}$. Use $\supseteq^{-}$ to denote the converse order to $\subseteq^{-}$. Define the *Scott order* on configurations by $x \sqsubseteq y$ iff $x \supseteq^{-} \cdot \subseteq^{+} \cdot \supseteq^{-} \cdots \supseteq^{-} \cdot \subseteq^{+} y$. Then, $\sqsubseteq$ is a partial order and part of a factorization system: if $x \sqsubseteq y$ then $\exists! z.\ x \supseteq^{-} z \subseteq^{+} y$.

Proposition 3.1 *[19] Let A be a game. Then, $x \in \mathcal{C}(\mathbb{C}\mathbb{C}_A)$ iff $x_2 \sqsubseteq_A x_1$, where $x_1 \in \mathcal{C}(A^{\perp})$ and $x_2 \in \mathcal{C}(A)$ are the projections of $x \in \mathcal{C}(A^{\perp} \| A)$ to its components.*

Theorem 3.2 *[19] Strategies $\sigma : S \to A$ correspond to discrete fibrations denoted $\sigma\text{“} : (\mathcal{C}(S), \sqsubseteq_S) \to (\mathcal{C}(A), \sqsubseteq_A)$, preserving $\supseteq^{-}$, $\subseteq^{+}$ and $\varnothing$.*

The theorem says we can view strategies in a game as (certain) discrete fibrations, so equivalently as presheaves over finite configurations with the Scott order. In particular, a strategy from a game A to a game B corresponds to a presheaf over $(\mathcal{C}(A^{\perp} \| B), \sqsubseteq_{A^{\perp}\|B}) \cong (\mathcal{C}(A), \sqsubseteq_A)^{\mathrm{op}} \times (\mathcal{C}(B), \sqsubseteq_B)$, so to a profunctor from $(\mathcal{C}(A), \sqsubseteq_A)$ to $(\mathcal{C}(B), \sqsubseteq_B)$. This correspondence yields a lax functor from strategies to profunctors. The view of strategies as (special) profunctors — explained further in [19] — will guide our later work.

We obtain a bicategory of concurrent games and strategies in which the objects are event structures with polarity — the games, the arrows from A to B are strategies $\sigma : A \mathrel{+\!\!\!\!\rightarrow} B$ and the 2-cells are maps of strategies. The vertical composition of 2-cells is the usual composition of maps. Horizontal composition is given by the composition of strategies $\odot$ (which extends to a functor on 2-cells via the universality of pullback).

A strategy in the game $A^{\perp} \| B$ corresponds to a strategy in the game $(B^{\perp})^{\perp} \| A^{\perp}$. Hence a strategy $A \mathrel{+\!\!\!\!\rightarrow} B$ corresponds to a dual strategy $B^{\perp} \mathrel{+\!\!\!\!\rightarrow} A^{\perp}$. The bicategory is rich in structure, in particular, it is compact-closed (so has a trace, a feedback operation).

3.2 Operations on strategies and duplication

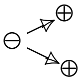
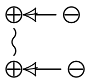
Beyond composition there are many other useful operations on strategies. Several of these have appeared previously in, for example, establishing determinacy [3] or the value theorem for games with pay-off [4] where proofs often hinge on constructing appropriate strategies.

We can form the *sum of strategies* $\mathop{[\,]}_{i \in I} \sigma_i$ of a family of strategies $\sigma_i : S_i \to A$, $i \in I$, in a common game A [4]. This is formed as the sum of the event structures S_i but where the initial −ve events are identified to maintain receptivity. A sum of strategies only commits to a particular component strategy once a Player move is made there. The empty sum $\perp$ essentially comprises the initial segment of the game A consisting of all the initial −ve events of A.

The pullback of a strategy σ across a (possibly partial) map f of event structures is itself a strategy $f^{*}\sigma$ [18]:

$$\begin{tikzcd} S' \arrow[r] \arrow[d, "f^{*}\sigma"'] \arrow[dr, phantom, "\lrcorner", very near start] & S \arrow[d, "\sigma"] \\ A \arrow[r, "f"'] & B\,. \end{tikzcd}$$

This operation can adjoin extra events and causal links to the original strategy; it subsumes, for example, operations on strategies in which we prefix an initial event.

We shall also use a previously unnoticed strategy δ_A that exists from a game A to $A\|A$, and expresses a form of duplication. Like copy-cat it introduces extra 'causal wiring' but its definition is much more subtle, though is easy to describe in special cases. For example if the game A comprises a single Player move $\oplus$ the strategy $\delta_A : A \mathrel{+\!\!\!\!\!\rightarrow} A\|A$ takes the form  . A similar construction applies in the case where all the moves of a game are that of Player. If the game A comprises a single Opponent move $\ominus$ then $\delta_A : A \mathrel{+\!\!\!\!\!\rightarrow} A\|A$ takes the form  , where the wiggly line indicates the inconsistency between the two Player moves.

Duplication The definition of $\delta_A : A \mathrel{+\!\!\!\!\!\rightarrow} A\|A$ in general is via rigid families. For each triple (x, y_1, y_2), where $x \in \mathcal{C}(A^\perp)$ and $y_1, y_2 \in \mathcal{C}(A)$, which is *balanced*, *i.e.*

$$\forall a \in y_1 \cup y_2.\ pol_A(a) = + \implies a \in x \text{ and } \forall a \in x.\ pol_{A^\perp}(a) = + \implies a \in y_1 \text{ or } a \in y_2,$$

and *choice* function $\chi : x^+ \to \{1,2\}$, from the positive events of x denoted by x^+, such that $\chi(a) = 1 \implies a \in y_1$ and $\chi(a) = 2 \implies a \in y_2$, the order $q(x, y_1, y_2; \chi)$ is defined to have underlying set $\{0\} \times x \cup \{1\} \times y_1 \cup \{2\} \times y_2$ with order generated by that inherited from $A^\perp\|A\|A$ together with

$$\{((0,a), (1,a)) \mid a \in y_1\} \quad \cup \quad \{((0,a),(2,a)) \mid a \in y_2\} \quad \cup$$
$$\{((\chi(a),a), (0,a)) \mid a \in x \ \&\ pol_{A^\perp}(a) = +\}\,.$$

The rigid family $\mathcal{Q}$ consists of all such $q(x, y_1, y_2; \chi)$ for balanced (x, y_1, y_2) and choice functions χ. From $\mathcal{Q}$ we obtain the event structure $\Pr(\mathcal{Q})$ in which events are prime orders, *i.e.* with a top element; events of $\Pr(\mathcal{Q})$ inherit the polarity of their top elements to obtain an event structure with polarity. We define the strategy $\delta_A : A \mathrel{+\!\!\!\!\!\rightarrow} A\|A$ to be the map $\Pr(\mathcal{Q}) \to A^\perp\|A\|A$ sending a prime to its top element. We remark that the meaning of a triple of configurations x, y_1, y_2 of C being balanced is almost $y_1 \cup y_2 \sqsubseteq_C x$ but is not this in general as $y_1 \cup y_2$ need not itself be a configuration of C.

The operation δ_A forms a comonoid with counit $\perp : A \mathrel{+\!\!\!\!\!\rightarrow} \varnothing$. In fact, they form part of a Frobenius algebra in which the accompanying monoid is obtained by duality.

4 A language for strategies

We describe, somewhat schematically, a language for describing strategies based on the constructions above. In fact, it is based on an earlier language for profunctors [15], taking advantage of the view of strategies as special profunctors.

4.1 Types

Types are games $A, B, C, \cdots$. We have type operations corresponding to the operations on games of forming the dual $A^\perp$, simple parallel composition $A\|B$, sum $\Sigma_{i\in I} A_i$ as well as recursively-defined types $\mu X.A(X)$, although we shall largely ignore the latter as it rests on well-established techniques [14].

One way to relate types is through the affine maps between them. There will be operations for shifting between types related by affine maps (described by configuration expressions), enabling us to shift strategies either forwards or backwards across affine maps.

A *type environment* is a finite partial function from variables to types, for convenience written typically as $\Gamma \equiv x_1 : A_1, \cdots, x_m : A_m$, in which the (configuration) variables $x_1, \cdots, x_m$ are distinct. It denotes a (simple) parallel composition $\|_{x_i} A_i$. In describing the semantics we shall sometimes write Γ for the parallel composition it denotes.

4.2 Configuration expressions

Configuration expressions denote finite configurations of games in an environment. A typing judgement $\Gamma \vdash p : B$ for a configuration expression p in a type environment Γ denotes an affine map from Γ to B. In particular, the judgement $\Gamma, x : A \vdash x : A$ denotes the partial map of event structures projecting to the single component A. The special case $x : A \vdash x : A$ denotes the identity map.

We shall allow configuration expressions to be built from affine maps $f = (f_0, f_1) : A \to_a B$ in the judgement $\Gamma, x : A \vdash fx : B$ and the equivalent judgement $\Gamma, x : A \vdash f_0 \cup f_1 x : B$. In particular, f_1 may be completely undefined, allowing *configuration expressions* to be built from constant configurations, as *e.g.* in the judgement for the empty configuration $\Gamma \vdash \varnothing : A$ or a singleton configuration $\Gamma \vdash \{a\} : A$ when a is an initial event of A. The expression $\{a\} \cup x'$ associated with the judgement

$$\Gamma, x' : A/a \vdash \{a\} \cup x' : A \,,$$

where a is an initial event of A, is used later in the transition semantics.

The three inductive rules for configuration expressions are as follows, where $\Gamma^\perp$ is $x_1 : A_1^\perp, \cdots, x_m : A_m^\perp$:

$$\frac{\Gamma \vdash p : A_j}{\Gamma \vdash jp : \Sigma_{i \in I} A_i} \; j \in I \qquad \frac{\Gamma \vdash p : A \qquad \Delta \vdash q : B}{\Gamma, \Delta \vdash (p, q) : A \| B} \qquad \frac{\Gamma \vdash p : B}{\Gamma^\perp \vdash p : B^\perp}$$

For a sum $\Sigma_{i \in I} A_i$, the left-hand rule gives configuration expressions jp where $j \in I$ and p is a configuration expression of type A_j: In the central rule for simple parallel composition we exploit the fact that configurations of simple parallel compositions are essentially pairsof configurations of the components. Finally, in the right-hand rule configurations of $B^\perp$ can be taken to be the same as configurations of B.

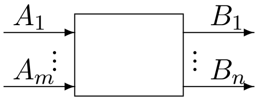
4.3 Terms for strategies

Terms denoting strategies have typing judgements:

$$x_1 : A_1, \cdots, x_m : A_m \vdash \; t \; \dashv y_1 : B_1, \cdots, y_n : B_n \;,$$

where all the variables are distinct, interpreted as a strategy from the game $x_1 : A_1, \cdots, x_m : A_m$ denotes to the game $y_1 : B_1, \cdots, y_n : B_n$ denotes.

We can think of the term t as a box with input and output wires for the typed variables:



Duality The duality of input and output is caught by the rules:

$$\frac{\Gamma, x : A \vdash t \dashv \Delta}{\Gamma \vdash t \dashv x : A^{\perp}, \Delta} \qquad \frac{\Gamma \vdash t \dashv x : A, \Delta}{\Gamma, x : A^{\perp} \vdash t \dashv \Delta}$$

Composition The composition of strategies is described in the rule

$$\frac{\Gamma \vdash t \dashv \Delta \qquad \Delta \vdash u \dashv \mathrm{H}}{\Gamma \vdash \exists \Delta. [\, t \parallel u \,] \dashv \mathrm{H}}$$

which, in the picture of strategies as boxes, joins the output wires of one strategy to input wires of the other. Note that the simple parallel composition of strategies arises as a special case when Δ is empty.

Nondeterministic sum We can form the nondeterministic sum of strategies of the same type:

$$\frac{\Gamma \vdash t_i \dashv \Delta \quad i \in I}{\Gamma \vdash []_{i \in I}\, t_i \dashv \Delta}$$

We shall use $\perp$ for the empty nondeterministic sum, when the rule above specialises to $\Gamma \vdash \perp \dashv \Delta$. The term $\perp$ denotes the minimum strategy in the game $\Gamma^{\perp} \| \Delta$.

Pullback We can form the pullback of two strategies of the same type:

$$\frac{\Gamma \vdash t_1 \dashv \Delta \quad \Gamma \vdash t_2 \dashv \Delta}{\Gamma \vdash t_1 \wedge t_2 \dashv \Delta}$$

In the case where t_1 and t_2 denote the respective strategies $\sigma_1 : S_1 \to \Gamma^{\perp} \| \Delta$ and $\sigma_1 : S_1 \to \Gamma^{\perp} \| \Delta$, the strategy $t_1 \wedge t_2$ denotes the pullback of σ_1 and σ_2. Informally, such a strategy acts as the two component strategies agree to act.

Hom-set The hom-set rule is a powerful way to lift affine maps or relations expressed in terms of cospans of affine maps to strategies. Write $p[\varnothing]$ for the substitution of the empty configuration $\varnothing$ for all configuration variables appearing in a configuration expression p. The hom-set rule

$$\frac{\Gamma \vdash p' : C \qquad \Delta \vdash p : C}{\Gamma \vdash p \sqsubseteq_C p' \dashv \Delta} \quad p[\varnothing] \sqsubseteq_C p'[\varnothing]$$

introduces a term standing for the hom-set $(\mathcal{C}(C), \sqsubseteq_C)(p, p')$. It relies on configuration expressions p, p' and their typings. If $\Delta \vdash p : C$ denotes the affine map $g = (g_0, g_1)$ and $\Gamma \vdash p' : C$ denotes the affine map $f = (f_0, f_1)$, the side condition of the rule ensures that $g_0 \sqsubseteq_C f_0$. A term for copy-cat arises as a special case of the hom-set rule: $x : A \vdash y \sqsubseteq_A x \dashv y : A$.

The hom-set rule is very expressive — see Section 4.4. The precise definition of the strategy which the hom-set rule yields is given in the next section.

Duplication Duplication terms are described by the rule

$$\frac{\Gamma \vdash p : C \quad \Delta_1 \vdash q_1 : C \quad \Delta_2 \vdash q_2 : C}{\Gamma \vdash \delta_C(p, q_1, q_2) \dashv \Delta_1, \Delta_2}$$

provided $p[\varnothing], q_1[\varnothing], q_2[\varnothing]$ is balanced in the sense of Section 3.2. The term for the duplication strategy is, in particular, $x : A \vdash \delta_A(x, y_1, y_2) \dashv y_1 : A, y_2 : A$.

4.3.1 Hom-set terms: semantics

The definition of the strategy which $\Gamma \vdash p \sqsubseteq_C p' \dashv \Delta$ denotes is quite involved. We first simplify notation. W.l.o.g. assume $\Delta \vdash p : C$ and $\Gamma \vdash p' : C$ — using duality we can always rearrange the environment to achieve this. Write A for the denotation of the environment Γ and B for the denotation of Δ. Let $\Delta \vdash p : C$ and $\Gamma \vdash p' : C$ denote respectively the affine maps $g = (g_0, g_1) : B \to_a C$ and $f = (f_0, f_1) : A \to_a C$. Note that we have that $g_0 \sqsubseteq_C f_0$ from the typing of $p \sqsubseteq_C p'$. We build the strategy out of a rigid family $\mathcal{Q}$ with elements as follows. First, define a pre-element to be a finite preorder comprising a set $\{1\} \times x \,\cup\, \{2\} \times y$, for which $x \in \mathcal{C}(A^\perp)$ & $y \in \mathcal{C}(B)$ & $gy \sqsubseteq_c fx$, with order that induced by $\leq_{A^\perp}$ on x, $\leq_B$ on y, with additional causal dependencies

$$\begin{aligned}(1, a) &\leq (2, b) \text{ if } f_1(a) = g_1(b) \ \ \& \ b \text{ is +ve, and}\\ (2, b) &\leq (1, a) \text{ if } f_1(a) = g_1(b) \ \ \& \ b \text{ is −ve.}\end{aligned}$$

As elements of the rigid family $\mathcal{Q}$ we take those pre-elements for which the order $\leq$ is a partial order (*i.e.* is antisymmetric). The elements of $\mathcal{Q}$ are closed under rigid inclusions, so $\mathcal{Q}$ forms a rigid family. We now take $S =_{\mathrm{def}} \Pr(\mathcal{Q})$; the events of S (those elements of $\mathcal{Q}$ with a top event) map to their top events in $A^\perp \| B$ from where they inherit polarities. This map can be checked to be a strategy: innocence follows directly from the construction, while receptivity follows from the constraint that $gy \sqsubseteq_c fx$.

It is quite easy to choose an example where antisymmetry fails in a pre-element, in other words, in which the preorder is not a partial order. However, when either p or p' is just a variable, no nontrivial causal loops are introduced and all pre-elements are elements. More generally, if one of p or p' is associated with a partial rigid map (*i.e.* a map which preserves causal dependency when defined), then no nontrivial causal loops are introduced and all pre-elements are elements.

4.3.2 Duplication terms: semantics

Consider now the semantics of a term $\Gamma \vdash \delta_C(p, q_1, q_2) \dashv \Delta$. W.l.o.g. we may assume that the environment is arranged so $\Delta \equiv \Delta_1, \Delta_2$ with judgements $\Gamma \vdash p : C$, $\Delta_1 \vdash q_1 : C$ and $\Delta_2 \vdash q_2 : C$. To simplify notation assume the latter judgements for configuration expressions denote the respective affine maps $f = (f^0, f^1) : A \to_a C$, $g_1 = (g_1^0, g_1{}^1) : B_1 \to C$ and $g_2 = (g_2^0, g_2{}^1) : B_2 \to C$. From the typing of $\delta_C(p, q_1, q_2)$ we have that (f^0, g_1^0, g_2^0) forms a balanced triple in C. We build the strategy out of a rigid family $\mathcal{Q}$ with elements as follows. We construct pre-elements from $x \in \mathcal{C}(A^\perp)$, $y_1 \in \mathcal{C}(B_1)$ and $y_2 \in \mathcal{C}(B_2)$ where (fx, g_1y_1, g_2y_2) is a balanced triple in C with a choice function χ. There are three kinds of elements of x:

$$\begin{aligned} x^- &= \{a \in x \mid pol_{A^\perp}(a) = -\}\,,\\ x_0^+ &= \{a \in x \mid pol_{A^\perp}(a) = + \ \& \ f^1(a) \in g^0_{\chi(f^1(a))}\} \text{ and}\\ x_1^+ &= \{a \in x \mid pol_{A^\perp}(a) = + \ \& \ f^1(a) \in g^1_{\chi(f^1(a))} y_{\chi(f^1(a))}\}\end{aligned}$$

We define a typical pre-element to be a finite preorder on the set

$$\{0\} \times (x^- \cup x_1^+ \cup \{(\chi(f^1(a)), a) \mid a \in x_0^+\}) \,\cup\, \{1\} \times y_1 \,\cup\, \{2\} \times y_2\,,$$

with order that induced by that of the game $A^\perp \| B_1 \| B_2$ with additional causal dependencies

$$\begin{aligned}(0,a) &\leq (1,b) \text{ if } f^1(a) = g_1^1(b) \ \&\ \text{ b is +ve in } B_1,\\ (0,a) &\leq (2,b) \text{ if } f^1(a) = g_2^1(b) \ \&\ \text{ b is +ve in } B_2, \text{ and}\\ (\chi(f^1(a)),b) &\leq (0,a) \text{ if } a \in x_1^+ \ \&\ f^1(a) = g^1_{\chi(f^1(a))}(b)\,,\end{aligned}$$

for b −ve in $B_{\chi(f^1(a))}$. As elements of the rigid family $\mathcal{Q}$ we take those pre-elements for which the order $\leq$ is a partial order (*i.e.* is antisymmetric). Once $\mathcal{Q}$ is checked to be a rigid family, we can take $S =_{\text{def}} \Pr(\mathcal{Q})$; the events of S map to the events in the game $A^\perp \| B_1 \| B_2$ associated with their top events, from where they inherit polarities. This map defines the strategy denoting the original duplication term.

4.4 Expressivity

The terms for strategies are surprisingly expressive and potentally rich in laws, able to support a form of equational reasoning, that we can only touch on here. For example the Frobenius algebra associated with duplication immediately yields laws. Other laws capture basic facts about the Scott order. For instance, assuming $z \subseteq x, y$ in $\mathcal{C}(A)$, we have $y \sqsubseteq_A x$ iff $y/z \sqsubseteq_{A/z} x/z$.

As we shall see, we can derive the laws expected of a recursion operator provided the recursion involves a homorphism w.r.t. the duplication comonad, and this fact too we could hope to derive. Some of the reasoning can be made diagrammatic, using the techniques of string diagrams.

Hom-set terms provide many basic strategies. The denotation of $x : A \vdash \varnothing \sqsubseteq_A \varnothing \dashv y : B$ is the strategy in the game $A^\perp \| B$ given by the identity map $\mathrm{id}_{A^\perp \| B} : A^\perp \| B \to A^\perp \| B$. The denotation of $\vdash y \sqsubseteq_A \varnothing \dashv y : A$ is $\bot_A$, the minimum strategy in the game A comprising just the initial −ve events of A.

The judgement $x : A_j \vdash y \sqsubseteq_{\Sigma_{i\in I} A_i} jx \dashv y : \Sigma_{i\in I} A_i$ denotes the injection strategy: its application to a strategy in A_j fills out the strategy according to the demands of receptivity to a strategy in $\Sigma_{i\in I} A_i$. Its converse $x : \Sigma_{i\in I} A_i \vdash jy \sqsubseteq_{\Sigma_{i\in I} A_i} x \dashv y : A_j$ applied to a strategy of $\Sigma_{i\in I} A_i$ projects the strategy to a strategy in A_j.

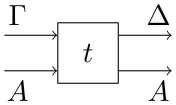
More is obtained by combining hom-set with other operations such as composition. Assume $\vdash t \dashv y : B$. When $f : A \to B$ is a map of event structures with polarity, the composition $\vdash\ \exists y : B.[\, t \parallel fx \sqsubseteq_B y\,] \ \dashv x : A$ denotes the pullback $f^*\sigma$ of the strategy σ denoted by t across the map $f : A \to B$. In the case where a map of event structures with polarity $f : A \to B$ is innocent, the composition $\vdash\ \exists x : A.[\, y \sqsubseteq_B fx \parallel t\,] \ \dashv y : B$ denotes the 'relabelling' $f_!\sigma$ of the strategy σ denoted by t.

A great deal is achieved through basic manipulation of the input and output "wiring" afforded by the hom-set rules and input-output duality. For instance, to achieve the effect of *lambda abstraction*: via the hom-set rule we obtain

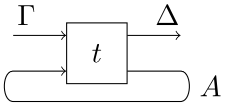
$$x : A^\perp, y : B \vdash\ z \sqsubseteq_{A\|B} (x,y)\ \dashv z : A^\perp \| B\ ,$$

which joins two inputs to a common output, whence:

$$\dfrac{\dfrac{\Gamma, x : A \vdash t \dashv y : B}{\Gamma \vdash t \dashv x : A^{\perp}, y : B} \quad \dfrac{\vdots}{x : A^{\perp}, y : B \vdash z \sqsubseteq_{A^{\perp}\|B} (x,y) \dashv z : A^{\perp}\|B}}{\Gamma \vdash \ \exists x : A^{\perp}, y : B.\,[\, t \parallel z \sqsubseteq_{A^{\perp}\|B} (x,y))\,] \ \dashv z : A^{\perp}\|B}$$

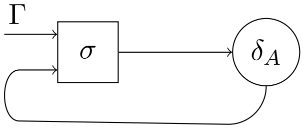
A *trace*, or feedback, operation is another consequence of such "wiring." Given a strategy $\Gamma, x : A \vdash t \dashv y : A, \Delta$ represented by the diagram 

we obtain $\Gamma, \Delta^{\perp} \vdash t \dashv x : A^{\perp}, y : A$, which, post-composed with the term $x : A^{\perp}, y : A \vdash x \sqsubseteq_A y \dashv$ denoting the copy-cat strategy $\gamma_{A^{\perp}}$, yields $\Gamma \vdash \exists x : A^{\perp}, y : A.\,[\, t \parallel x \sqsubseteq_A y \,] \dashv \Delta$, representing its trace:

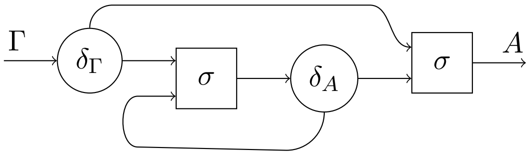


The composition introduces causal links from the +ve events of $y : A$ to the −ve events of $x : A$, and from the +ve events of $x : A$ to the −ve events of $y : A$ — these are the usual links of copy-cat $\gamma_{A^{\perp}}$ as seen from the left of the turnstyle. This trace coincides with the feedback operation which has been used in the semantics of nondeterministic dataflow (where only games comprising solely Player moves are needed) [13].

Recursive definitions can be achieved from trace with the help of duplication. For those strategies which respect δ, *i.e.*$\delta_A \odot \sigma \ \cong \ (\sigma \| \sigma) \odot \delta_{\Gamma\|A}$, and in particular for strategies which are homomorphisms between δ-comonoids, the recursive definition does unfold in the way expected, in the sense that the recursive definition is isomorphic to its unfolding:



$\cong$



This follows as a general fact from the properties of trace monoidal categories and the string-diagram reasoning they support. However, not all strategies are homomorphisms between δ-comonoids, characterised in the following theorem.

Theorem 4.1 *A strategy $\sigma : S \to A^{\perp} \parallel B$ respects δ iff*

- *components $\sigma_1 : S \to A^{\perp}$ and $\sigma_2 : S \to B$ preserve causal dependency when defined,*
- *σ_1 reflects configurations of $A^{\perp}$, i.e. if $x \subseteq S$ is such that $\sigma_1 x \in \mathcal{C}(A^{\perp})$ then $x \in \mathcal{C}(S)$, and*
- *for every +ve event $s \in S$ such that $\sigma(s) \in A^{\perp}$ we have the number of −ve events of $\sigma_2[s]$ equals the number of +ve events of $\sigma_1[s]$.*

In this case, σ also respects counits, i.e. $\bot_B \odot \sigma \cong \bot_A$.

5 A process perspective

The operations on strategies have much in common with the operations of process algebra and can be seen as forming the basis of a higher-order process language. However, from the perspective of concurrent processes, we must address several issues: its *operational semantics*, a suitable form of *equivalence* and *expressivity*. These require we examine the effects of synchronization and the internal, neutral events it produces, more carefully. Composition of strategies can introduce deadlock which is presently hidden. This, for example, affects the reliability of winning strategies; presently a strategy may be deemed winning and yet possibly deadlock before reaching a winning configuration. The next example illustrates how hidden deadlocks may be created in a composition of strategies.

Example 5.1 (i) Deadlock may arise in a composition $\tau \odot \sigma$ through $\sigma : A \relbar\joinrel\mapstochar\joinrel\rightarrow B$ and $\tau : B \relbar\joinrel\mapstochar\joinrel\rightarrow C$ imposing incompatible causal dependencies between events in B. For instance B may contain two concurrent events of opposite polarities $b_1 = \ominus$ and $b_2 = \oplus$. The strategy σ may impose the causal dependency $s_1 \rightarrowtriangle s_2$ between occurrences of b_1 and b_2 respectively. From the point of view of strategy τ, the game B has changed polarity to $B^\perp$ and τ may impose the reverse causal dependency $s_2 \rightarrowtriangle s_1$ between occurrences of b_2 and b_1 respectively.
(ii) Composition of strategies may hide computation which is stuck. For games $B = \oplus \| \oplus$ and $C = \oplus$, assume strategy $\sigma_1 : \varnothing \relbar\joinrel\mapstochar\joinrel\rightarrow B$ nondeterministically chooses the right or left move in B, strategy $\sigma_2 : \varnothing \relbar\joinrel\mapstochar\joinrel\rightarrow B$ chooses just the right move in B, while strategy $\tau : B \relbar\joinrel\mapstochar\joinrel\rightarrow C$ yields output in C if it gets the right event of B as input. The two strategy compositions $\tau \odot \sigma_1$ and $\tau \odot \sigma_2$ are indistinguishable. □

5.1 Partial strategies

To treat such phenomena explicitly and in order to obtain a transition semantics, we extend strategies with neutral events. Extend event structures with polarity to include a neutral polarity 0; as before, maps preserve polarities when defined. However within games we shall still assume that all events have +ve or −ve polarity.

Definition 5.2 A *partial strategy from* a game A *to* a game B comprises a total map $\sigma : S \to A^\perp \| N \| B$ of event structures with polarity (in which S may also have neutral events) where
(i) N is an event structure consisting solely of neutral events;
(ii) σ is *receptive*, *i.e.* if $\sigma x \stackrel{c}{\longrightarrow}\!\!\subset$ in $\mathcal{C}(S)$ with c −ve, then $x \stackrel{s}{\longrightarrow}\!\!\subset$ and $\sigma(s) = c$, for some unique $s \in S$;
(iii) in the partial-total factorization of the composition of σ with the projection $A^\perp \| N \| B \to A^\perp \| B$, drawn below, the defined part σ_0 is a strategy.

$$\begin{tikzcd}
S \arrow[r] \arrow[d, "\sigma"'] & S_0 \arrow[d, "\sigma_0"] \\
A^\perp \| N \| B \arrow[r] & A^\perp \| B
\end{tikzcd}$$

Partial strategies in a game A correspond to partial strategies from the empty game to A. Strategies between games correspond to those partial strategies in which the neutral events N form the empty event structure.

It may seem odd that partial strategies are total maps. Why have we not taken a partial strategy to be undefined on events which are sent to N? Because such partial maps do not behave well under pullback, and this would complicate the definition of composition and spoil later results such as that the pullback of a partial strategy is a partial strategy. With our choice of definition we are able to localise neutral events to the games over which they occur; with the alternative definition, different forms of undefined would become conflated.

5.2 Operations on partial strategies

The operations on strategies extend and give an interpretation of the language of Section 4 in terms of partial strategies. The defined parts of the operations on partial strategies coincide with the operations on the defined parts.

We can compose two partial strategies $\sigma : S \to A^\perp \| N_S \| B$ and $\tau : T \to B^\perp \| N_T \| C$ by pullback. Ignoring polarities temporarily, and padding with identity maps, we obtain $\tau \circledast \sigma$ via the pullback

$$\begin{tikzcd}
& T \circledast S \arrow[dl] \arrow[dr] \arrow[dd, bend left, "\tau\circledast\sigma"] \arrow[d, phantom, "\lrcorner"] & \\
S \| N_T \| C \arrow[dr, "\sigma \| N_T \| C"'] & {} & A \| N_S \| T \arrow[dl, "A \| N_S \| \tau"] \\
& A \| N_S \| B \| N_T \| C . &
\end{tikzcd}$$

once we reinstate polarities and make the events of B neutral. Receptivity of $\tau \circledast \sigma$ follows directly from that of σ and τ. That the defined part of $\tau \circledast \sigma$ is a strategy follows once it is shown that the defined part of the composite $T \circledast S \xrightarrow{\tau\circledast\sigma} A^\perp \| (N_S \| B \| N_T) \| C \longrightarrow A^\perp \| C$ is isomorphic to $\tau_0 \odot \sigma_0$, the composition of the defined parts of σ and τ.

With partial strategies we no longer generally have that composition with copycat yields the same strategy up to isomorphism: there will generally be extra neutral events introduced through synchronizations. However, as demonstrated in Section 6, a bicategory may be recovered through the use of may/must equivalence.

Let $\sigma_i : S_i \to A^\perp \| N_i \| B$, where $i \in I$, be a family of partial strategies. Their *sum* is the partial strategy $\mathop{[]}_{i\in I} \sigma_i : S \to A^\perp \| (\|_{i\in I} N_i) \| B$. Its events are obtained as the disjoint union of the S_i but where the initial −ve events are identified to maintain receptivity; they map under $\mathop{[]}_{i\in I} \sigma_i$ as directed by the component maps σ_i. Causal dependency is inherited from the components S_i with a finite subset of events consistent iff its down-closure contains +ve events from at most one S_i. As such, the nondeterministic sum only commits to a component through the occurrence of a positive event: from the perspective of tracking potential deadlocks, it is not necessary to view neutral events as committing to a particular component since, in isolation, a neutral event cannot introduce deadlock when composed with a counterstrategy due to receptivity.

The *pullback* of partial strategies $\sigma_1 : S_1 \to A^\perp \| N_1 \| B$ and $\sigma_2 : S_2 \to A^\perp \| N_2 \| B$ is obtained as to the right.

$$\begin{tikzcd}
 & S_1 \wedge S_2 \arrow[dl] \arrow[dr] \arrow[dd, dotted, "\sigma_1 \wedge \sigma_2"] & \\
S_1 \| N_2 \arrow[dr, "\sigma_1 \| N_2"'] & & S_2 \| N_1 \, . \arrow[dl, "\sigma_2 \| N_1"] \\
 & A^\perp \| N_1 \| N_2 \| B &
\end{tikzcd}$$

5.3 Transition semantics

We now discuss transition semantics for partial strategies presented in Figure 1. For brevity, the rules presented require the left-hand environment to be empty; this can always be achieved using the rules for duality. In the transition rules, we write $t \dashv \Delta$ instead of $\varnothing \vdash t \dashv \Delta$. Transitions are associated with two kinds of actions, either an action o associated with a hidden neutral action $t \dashv \Delta \xrightarrow{o} t' \dashv \Delta$ or an initial event located in the environment $t \dashv x : A, \Delta \xrightarrow{x : a : x'} t' \dashv x' : A/a, \Delta$. Notice that a neutral action leaves the types unchanged but may affect the term. An action $x : a : x'$ is associated with an initial event $ev(x : a : x') =_{\text{def}} x : a$ at the x-component of the environment. On its occurrence, the component of the environment $x : A$ is updated to $x' : A/a$ in which x', a fresh resumption variable, stands for the configuration remaining in the remaining game A/a. Say an action $x : a : x'$ is +ve/−ve according as a is +ve/−ve. In the ruless for composition, we use α for o or an action of the form $x : a : x'$ where x is in the domain of Γ and use β for o or an action of the form $y : b : y'$ where y is in the domain of H.

In typed judgements of $\delta_C(p, q_1, q_2)$, a variable can appear free in at most one of p, q_1, q_2. Write, for example, $y \in \text{fv}(p)$ for y is free in p, and $q_1(y : b) \in p[\varnothing]$ to mean the image of b under the map q_1 denotes is in the configuration denoted by $p[\varnothing]$.

Theorem 5.3 *Assume certain primitive strategies $\varnothing \vdash \sigma_0 \dashv \Delta$, so as a map, $\sigma_0 : S \to \Delta$, for which we assume rules,*

$$\frac{}{\sigma_0 \dashv \Delta \xrightarrow{\epsilon} \sigma_0' \dashv \Delta'}\, s \textit{ is initial in } S \;\&\; \sigma_0(s) = ev(\epsilon).$$

Then, derivations in the operational semantics

$$\frac{\vdots}{t \dashv \Delta \xrightarrow{\epsilon} t' \dashv \Delta',}$$

up to α-equivalence, in which t denotes the partial strategy $\sigma : S \to \Delta$, are in 1-1 correspondence with initial events s in S such that $\sigma(s) = ev(\epsilon)$ when $ev(\epsilon) \neq o$ or s is neutral when $ev(\epsilon) = o$.

6 May and Must equivalence

We now study 'may' and 'must' equivalence and how it may be used to recover bicategorical structure on strategies equipped with *stopping configurations.*

Consider three partial strategies in a game comprising a single +ve event. For this discussion it will be sufficient to consider just the event structures of the partial strategies:

$$S_1 \quad \oplus \qquad\qquad S_2 \quad \circledcirc \rightarrowtriangle \oplus \qquad\qquad S_3 \quad \circledcirc \sim \circledcirc \rightarrowtriangle \oplus$$

Composition:

$$\frac{t \dashv y : B, \Delta, \Gamma \xrightarrow{y:b:y'} t' \dashv y' : B/b, \Delta, \Gamma \qquad u \dashv y : B^{\perp}, \Delta^{\perp}, \mathrm{H} \xrightarrow{y:b:y'} u' \dashv y' : B^{\perp}/b, \Delta^{\perp}, \mathrm{H}}{\exists y : B, \Delta.\, [\, t \parallel u\,] \dashv \Gamma, \mathrm{H} \xrightarrow{o} \exists y' : B/b, \Delta.\, [\, t' \parallel u'\,] \dashv \Gamma, \mathrm{H}}$$

$$\frac{t \dashv \Gamma, \Delta \xrightarrow{\alpha} t' \dashv \Gamma', \Delta}{\exists \Delta.\, [\, t \parallel u\,] \dashv \Gamma \xrightarrow{\alpha} \exists \Delta.\, [\, t' \parallel u\,] \dashv \Gamma'}$$

$$\frac{u \dashv \mathrm{H}, \Delta^{\perp} \xrightarrow{\beta} u' \dashv \mathrm{H}', \Delta^{\perp}}{\exists \Delta.\, [\, t \parallel u\,] \dashv \mathrm{H} \xrightarrow{\beta} \exists \Delta.\, [\, t \parallel u'\,] \dashv \mathrm{H}'}$$

Hom-sets: Assuming a is an initial event of A for which $p[\{a\}/x][\varnothing] \sqsubseteq_C p'[\{a\}/x][\varnothing]$,

$$p \sqsubseteq_C p' \dashv x : A, \Delta \xrightarrow{x:a:x'} p[\{a\} \cup x'/x] \sqsubseteq_C p'[\{a\} \cup x'/x] \dashv x' : A/a, \Delta$$

Above, the variable x will only appear in one of p and p'.

Sum of partial strategies:

$$\frac{t_j \dashv \Delta \xrightarrow{o} t'_j \dashv \Delta'}{\Box_{i\in I}\, t_i \dashv \Delta \xrightarrow{o} \Box_{i\in I}\, t'_i \dashv \Delta}\, j \in I, \text{ where } t'_i = t_i \text{ if } i \neq j$$

$$\frac{t_i \dashv \Delta \xrightarrow{\epsilon} t'_i \dashv \Delta' \quad \forall i \in I}{\Box_{i\in I}\, t_i \dashv \Delta \xrightarrow{\epsilon} \Box_{i\in I}\, t'_i \dashv \Delta}\, \epsilon \text{ is } -\text{ve}$$

$$\frac{t_j \dashv \Delta \xrightarrow{\epsilon} t'_j \dashv \Delta'}{\Box_{i\in I}\, t_i \dashv \Delta \xrightarrow{\epsilon} t'_j \dashv \Delta'}\, j \in I \ \&\ \epsilon \text{ is } +\text{ve}$$

Pullback:

$$\frac{t_1 \dashv \Delta \xrightarrow{o} t'_1 \dashv \Delta}{t_1 \wedge t_2 \dashv \Delta \xrightarrow{o} t'_1 \wedge t_2 \dashv \Delta} \qquad \frac{t_2 \dashv \Delta \xrightarrow{o} t'_2 \dashv \Delta}{t_1 \wedge t_2 \dashv \Delta \xrightarrow{o} t_1 \wedge t'_2 \dashv \Delta} \qquad \frac{t_i \dashv \Delta \xrightarrow{z:c:z'} t'_i \dashv \Delta' \ \forall i \in \{1,2\}}{t_1 \wedge t_2 \dashv \Delta \xrightarrow{z:c:z'} t'_1 \wedge t'_2 \dashv \Delta}$$

Duplication: $\delta_C(p, q_1, q_2) \dashv x : A, \Delta \xrightarrow{x:a:x'} \delta_C(p, q_1, q_2)[\{a\} \cup x'/x] \dashv x' : A/a, \Delta$ if either

- a is an initial −ve event of A, or
- a is an initial +ve event of A and there exists $i \in \{1,2\}$ s.t. either
  - $x \in \mathrm{fv}(q_i)$ and $q_i(x : a) \in p[\varnothing]$ or
  - $x \in \mathrm{fv}(p)$ and $p(x : a) \in q_i[\varnothing]$.

Fig. 1. Transition semantics

Neutral events are drawn as ⊚. Conflict between pairs of events (meaning that there is no set in the consistency relation containing them both) is drawn as $\sim\sim$. All three partial strategies have the same strategy as their defined parts. However, from the point of view of observing the move in the game, the first two partial strategies differ from the third. In a maximal play both S_1 and S_2 will result in the observation of the single move of the game. However, in S_3 one maximal play is that in which the leftmost neutral event has occurred, in conflict with observing the single move of the game.

We follow [5] in making these ideas precise. For configurations x, y of an event structure with polarity which may have neutral events write $x \subseteq^p y$ to mean $x \subseteq y$ and all events of $y \setminus x$ have polarity + or 0. We write $\subseteq^0$ to mean the inclusion involves only neutral events

Definition 6.1 Let σ be a partial strategy in a game A. Let $\tau : T \to A^{\perp} \| \oplus$ be a 'test' strategy from A to a the game consisting of a single Player move $\checkmark$.

Say σ *may pass* τ iff there exists $x \in \mathcal{C}^{\infty}(T \circledast S)$ with image $(\tau \circledast \sigma)x$ containing $\checkmark$.

Say σ *must pass* τ iff all $x \in \mathcal{C}^{\infty}(T \circledast S)$ which are $\subseteq^p$-maximal have image $(\tau \circledast \sigma)x$ containing $\checkmark$.

Say two partial strategies are *'may' ('must') equivalent* iff the tests they may (respectively, must) pass are the same.

Two partial strategies with the same strategy as their defined part are 'may' equivalent but need not be 'must' equivalent. 'Must' inequivalence is lost in moving from partial strategies to strategies. Moreover, as we have seen, partial strategies lack identities w.r.t. composition, so they do not even form a bicategory. Fortunately, for 'may' and 'must' equivalence it is not necessary to use partial strategies; it is sufficient to carry with a strategy the extra structure of 'stopping' configurations (= images of p-maximal configurations in a partial strategy). Composition and copy-cat on strategies extend to composition and copy-cat on strategies with stopping configurations, while maintaining a bicategory, in the following way.

First, to deal with races, we are forced to introduce a refinement of the Scott order. We write $x \lhd^* y$ for the transitive closure of the relation $\lhd \subseteq \mathcal{C}^\infty(S) \times \mathcal{C}^\infty(S)$ defined as

$$x \lhd y \iff x \sqsubseteq y \text{ and } y \parallel x \text{ is +-maximal in } \mathrm{C\!\!C}_S.$$

On race-free games, $\lhd$ is the identity relation on $\mathcal{C}^\infty(S)$.

Let $\sigma : S \to A^\perp \| N \| B$ be a partial strategy from a game A to a game B. Recall its associated partial-total factorization has defined part a strategy $\sigma_0 : S_0 \to A^\perp \| B$. Define the (possibly) *stopping* configurations in $\mathcal{C}^\infty(S_0)$ to be

$$\mathrm{Stop}(\sigma) =_{\mathrm{def}} \downarrow \{dx \mid x \in \mathcal{C}^\infty(S) \text{ is } p\text{-maximal}\},$$

where $d : S \to S_0$ is the partial map that is undefined where σ is undefined and $\downarrow S$ is the down-closure of S for the order $\lhd^*$. Note that $\mathrm{Stop}(\sigma)$ will include all the +-maximal configurations of S_0: any +-maximal configuration y of S_0 is the image under p of its down-closure $[y]$ in S, and by Zorn's lemma this extends (necessarily by neutral events) to a maximal configuration x of S with image y under d; by maximality, if $x \overset{s}{\longrightarrow\!\!\subset}$ then s cannot be neutral, nor can it be +ve as this would violate the +-maximality of y.

The operation $\mathrm{St} : \sigma \mapsto (\sigma_0, \mathrm{Stop}(\sigma))$ above, from partial strategies to strategies with stopping configurations, motivates the following definitions.

A *strategy with stopping configurations* in a game A comprises a strategy $S \to A$ together with a subset $M_S \subseteq \mathcal{C}^\infty(S)$ which is the down-closure with respect to $\lhd^*$ of a set of +-maximal configurations. As usual, a *strategy with stopping configurations* from a game A to game B is a strategy with stopping configurations in the game $A^\perp \| B$. We can define 'may' and 'must' testing of strategies with stopping configurations analogously to above.

Given two strategies with stopping configurations $\sigma : S \to A^\perp \| B$, M_S and $\tau : T \to B^\perp \| C$, M_T we define their composition by $(\tau, M_T) \odot (\sigma, M_S) =_{\mathrm{def}} (\tau \odot \sigma, M_T \odot M_S)$ where $x \in M_T \odot M_S$ iff

$$\exists z \in \mathcal{C}^\infty(T \circledast S).\ [x]_{T \circledast S} \subseteq^0 z \ \&\ \Pi_1 z \in M_S \ \&\ \Pi_2 z \in M_T\,.$$

The stopping configurations of copy-cat are obtained as for any other strategy, and in particular $M_{CC_A} = \{y \parallel x \mid x, y \in \mathcal{C}^\infty(A) \ \&\ x \lhd^* y\}$.

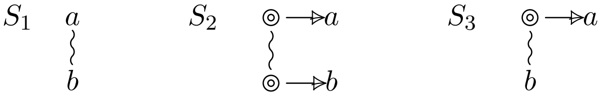
Proposition 6.2 *$\gamma_A, M_{\mathrm{C\!\!C}_A}$ is an identity w.r.t. the composition on strategies with stopping configurations.*

Proposition 6.3 *Let σ be a partial strategy from A to B and τ a partial strategy* *from B to C. Then*

$$\mathrm{Stop}(\tau \circledast \sigma) = \mathrm{Stop}(\tau) \odot \mathrm{Stop}(\sigma)\,.$$

Corollary 6.4 *A partial strategy σ 'may' (respectively 'must') pass a test τ iff $St(\sigma)$ 'may' ('must') pass τ. The operation St preserves 'may' and 'must' equivalence.*

Example 6.5 It is tempting to think of neutral events as behaving like the internal "tau" events of CCS [10]. However, in the context of strategies they behave rather differently. Consider three partial strategies, over a game comprising of just two concurrent +ve events, say a and b. The partial strategies have the following event structures in which we have named events by the moves they correspond to in the game:



All three become isomorphic under St so are 'may' and 'must' equivalent to each other. □

Strategies with stopping configurations inherit the structure of a bicategory from strategies. We can interpret the metalanguage directly in terms of strategies with stopping configurations in such a way that the denotation of a term as a strategy with stopping configurations is the image under St of its denotation as a partial strategy. To achieve this, we specify the stopping configurations of both the sum and pullback of strategies.

For the sum of strategies $[]_{i\in I}\,\sigma_i$ with stopping configurations σ_i, a configuration of the sum is stopping iff it is the image of a stopping configuration under the injection from a component.

Consider strategies $\sigma : S \to A$ and $\tau : T \to A$ with stopping configurations M_S and M_T respectively. Let their pullback be denoted by $\sigma \wedge \tau : P \to A$ with projection morphisms $\pi_1 : P \to S$ and $\pi_2 : P \to T$. A configuration of P is defined to be stopping iff there exist x_1, x_2 such that $\pi_1 x \subseteq^+ x_1$ and $\pi_2 x \subseteq^+ x_2$ and $x_1 \in M_S$ and $x_2 \in M_T$, and furthermore there exists a partition $x^+ = Y_1 \cup Y_2$ satisfying $x_i \cap Y_i = \emptyset$. The set of stopping configurations of P coincides with the stopping configurations obtained via St from the pullback of partial strategies.

The treatment of winning strategies of [3] generalises straightforwardly, with the role of +-maximal configurations replaced by that of stopping configurations.

7 Extensions & concluding remarks

We have seen a range of constructions on concurrent strategies that support a rich higher-order language for them, with a corresponding operational semantics. For the latter a central part has been the introduction of partial strategies. Though composition of a partial strategy with copy-cat does not in general yield the same strategy, we have seen how a bicategory can be obtained that respects the may/must behaviour of partial strategies by using *stopping configurations*.

The bicategorical structure of strategies is largely undisturbed by extensions to *probabilistic* and *quantum* games [18], *imperfect information* [17] and *symmetry* [1] — though compact-closure becomes $*$-autonomy under extensions by *winning conditions* [3] and *pay-off* [4]. The language of strategies is applicable in these extensions, with minor modifications. The constraints of linearity can be alleviated in games with symmetry which support (co)monads for copying [1]. The constructions of the language extend fairly directly to games with symmetry, though to exploit symmetry fully the language needs to be extended to accommodate (co)monads up to symmetry.

References

[1] Castellan, S., P. Clairambault and G. Winskel, *Symmetry in concurrent games*, in: *LICS 2014* (2014).

[2] Cattani, G. L. and G. Winskel, *Profunctors, open maps and bisimulation*, Mathematical Structures in Computer Science **15** (2005), pp. 553–614.

[3] Clairambault, P., J. Gutierrez and G. Winskel, *The winning ways of concurrent games*, in: *LICS '12* (2012).

[4] Clairambault, P. and G. Winskel, *On concurrent games with payoff*, in: *MFPS '13*, number 298 in ENTCS (2013).

[5] De Nicola, R. and M. Hennessy, *Testing equivalences for processes*, Theoretical Computer Science **34** (1984).

[6] Faggian, C. and M. Piccolo, *Partial orders, event structures and linear strategies*, in: *TLCA '09*, LNCS **5608**, 2009.

[7] Hirschowitz, T., *Full abstraction for fair testing in ccs*, in: *CALCO '13*, LNCS **8089**, 2013 .

[8] Honda, K., *Processes and games*, ENTCS **71** (2004), {WRLA} 2002, Rewriting Logic and Its Applications.
URL http://www.sciencedirect.com/science/article/pii/S1571066105825289

[9] Hyland, M., *Some reasons for generalising domain theory*, Mathematical Structures in Computer Science **20** (2010), pp. 239–265.

[10] Milner, R., "Communication and concurrency," Prentice Hall, 1989.

[11] Nygaard, M. and G. Winskel, *Linearity in process languages*, in: *LICS '02* (2002).

[12] Rideau, S. and G. Winskel, *Concurrent strategies*, in: *LICS 2011* (2011).

[13] Saunders-Evans, L. and G. Winskel, *Event structure spans for nondeterministic dataflow*, ENTCS **175** (2007).

[14] Winskel, G., *Event structure semantics for CCS and related languages*, in: *ICALP'82*, LNCS **140** (1982).

[15] Winskel, G., *Relations in concurrency*, in: *LICS '05* (2005).

[16] Winskel, G., *Event structures with symmetry*, Electr. Notes Theor. Comput. Sci. 172: 611-652 (2007).

[17] Winskel, G., *Winning, losing and drawing in concurrent games with perfect or imperfect information*, in: *Festschrift for Dexter Kozen*, LNCS **7230** (2012).

[18] Winskel, G., *Distributed probabilistic and quantum strategies*, in: *MFPS '13*, number 298 in ENTCS (2013).

[19] Winskel, G., *Strategies as profunctors*, in: *FOSSACS '13*, LNCS (2013).

On a Categorical Framework for Coalgebraic Modal Logic

Liang-Ting Chen[1,2]

Institute of Information Science
Academia Sinica
Taipei, Taiwan

Achim Jung[3]

School of Computer Science
University of Birmingham
Birmingham, U.K.

Abstract

A category of one-step semantics is introduced to unify different approaches to coalgebraic logic parametric in a contravariant functor that assigns to the state space its collection of predicates with propositional connectives. Modular constructions of coalgebraic logic are identified as colimits, limits, and tensor products, extending known results for predicate liftings. Generalised predicate liftings as modalities are introduced. Under common assumptions, the logic of all predicate liftings together with a complete axiomatisation exists for any type of coalgebras, and it is one-step expressive for finitary functors. Colimits and compositions of one-step expressive coalgebraic logics are shown to remain one-step expressive.

Keywords: Predicate liftings, coalgebras, coalgebraic logic, modal logic, Stone duality, compositionality, expressivity

1 Introduction

Two syntax-oriented approaches to coalgebraic modal logic — Moss' cover modality [23] and Pattinson's predicate liftings [24,25,26] — are successful in producing a wide range of modal logics parametric in a Set functor. Subsequently, the algebraic semantics of the logics of predicate liftings was formulated elegantly as a particular form of natural transformations using Stone duality [15]. To explain it, let BA denote the category of Boolean algebras and $\mathcal{Q}\colon \mathsf{Set} \to \mathsf{BA}$ the contravariant powerset

[1] The first author was supported by EPSRC Doctoral Training Grant EP/G041296/1 and the School of Computer Science, University of Birmingham, during his studies.

[2] Email: lxc@iis.sinica.edu.tw

[3] Email: A.Jung@cs.bham.ac.uk

functor. For an endofunctor T, a family of sets Λ_n of n-ary predicate liftings indexed by $\mathbb{N}$ amounts to a natural transformation δ to $U\mathcal{Q}T$ from the coproduct indexed by all n-ary predicate liftings λ of n-fold product of $U\mathcal{Q}$, i.e. $\coprod_{n\in\mathbb{N}}\coprod_{\lambda\in\Lambda_n} U\mathcal{Q}^n$ where U is the forgetful functor, so by adjunction this family (Λ_n) gives rise to an *interpretation of modalities* $\delta\colon L\mathcal{Q}\Rightarrow\mathcal{Q}T$ for some functor L; it introduced what we call *one-step semantics* (Definition 3.1), an expression first coined by Cîrstea and Pattinson [7] in a different but equivalent form. Later Moss' cover modality was also formulated in this way [21].

This abstract functorial framework was further developed in [17,4,14,19,18] with the aim of finding suitable modal logics for various coalgebras, e.g. [5,11,13,20], replacing $\mathcal{Q}$ with other contravariant functors. For example, Markov processes are coalgebras of the Giry monad, and propositional connectives for measurable spaces can be specified by the contravariant functor $\mathbb{S}\colon \mathsf{Meas}\to\mathsf{MSL}$ mapping a space to its σ-algebra, considered as a meet semilattice.

Adequacy and soundness of the functorial framework follow from its very formulation as shown by Kurz [17], and a sufficient condition of *expressiveness* was first established by Klin [14] for finitary type functors on locally finitely presentable categories (to be presented in Subsection 5.2 below). For example, Boolean logic extended with the possibility modality is expressive for all image-finite Kripke frames, i.e. coalgebras for the finite powerset functor $\mathbb{P}_\omega$. The restriction to finitary functors is not necessary, however. Multi-modal logic is expressive for image-finite *A-labelled* Kripke frames, but Klin's condition does not cover this case, since its corresponding type functor, the A-fold product $\mathbb{P}^A$, is not necessarily finitary.

Another important line of research investigated *modularity* of predicate liftings [7,27]: not only expressiveness but also completeness are stable under certain constructions. With regards to the example above, expressiveness of multi-modal logic for $\mathbb{P}_\omega^A$-coalgebras is *inherited* from modal logic for $\mathbb{P}_\omega$-coalgebras. The idea has since been incorporated into the functorial framework over Stone duality in [19], where a subset of constructions is considered, focusing on completeness. In [18], Kurz and Leal show how to translate Moss' cover modality into predicate liftings and vice versa; thus making it amenable to their completeness analysis.

In the present paper, we put forward a *fully categorical treatment in a syntax-independent fashion beyond Stone duality*, so that existing results and concepts can be further applied without further effort to richer structures such as ordered, topological, or measurable spaces. Our running examples will be mostly over Set, however, in the hope that the reader will be able to make a direct comparison with known results and concepts.

In Section 3, we introduce a category CoLog of one-step semantics, which includes Pattinson's predicate liftings and Moss' cover modalities as objects, and exhibit its rich structure. Modularity of coalgebraic modal logics are recognised as standard categorical constructions in this category, avoiding any syntactic bookkeeping. A "full logic" for each type of coalgebras, using the basic properties of adjunction, is identified: every other logic for the same type can be (uniquely) translated to it. In Section 4, we then focus on *equational* one-step semantics, which are found to be isomorphic to those determined by predicate liftings. Notions of modalities and equations are derived naturally from the analysis of categorical structures, as is the characterisation of the full equational logic. Logical properties of one-step semantics are discussed in Section 5. Klin's condition of expressiveness is adapted. The categorical viewpoint allows us to formulate general preservation principles for coalgebraic *expressiveness* that apply to *all* logics in CoLog.

The framework is parametric in a contravariant functor mapping state spaces to "algebras" for the base logic about which very little needs to be assumed to cover a large variety of examples. Indeed, we would like to suggest that our work provides the right level of abstraction for understanding coalgebraic modal logic.

This paper summaries the first author's PhD thesis [6], to which we point for most of the proofs.

2 Preliminaries

We follow Mac Lane's terminology of category theory [22] except that for us a (co)reflective subcategory is defined to be *full*. For an endofunctor T, a **coalgebra** for T is a morphism $\xi\colon X \to TX$; a **coalgebra homomorphism** from (X, ξ) to (Y, γ) is a morphism $f\colon X \to Y$ satisfying $Tf \circ \xi = \gamma \circ f$. The category of T-coalgebras is denoted by $\mathscr{X}_T$.

2.1 Dual adjunctions

Definition 2.1 A contravariant functor $P\colon \mathscr{X} \to \mathscr{A}$ is said to be **dual** to a contravariant functor S if together they form an adjunction $S \dashv P\colon \mathscr{X}^{\mathrm{op}} \to \mathscr{A}$ with unit η and counit ϵ.

We use dual adjunctions to set up a link between "state spaces", the objects of $\mathscr{X}$, and (algebras of) "logics", the objects of $\mathscr{A}$. This is a particular instance of "Stone duality"; for a general introduction see [12].

Example 2.2 (i) Consider the powerset 2^{-} as a contravariant functor from Set to Set; it is dual to itself. Alternatively, consider the powerset as a Boolean algebra; we obtain $\mathcal{Q}$, a contravariant functor from Set to BA. The natural dual to $\mathcal{Q}$ is the ultrafilter functor $\mathcal{F}$, equivalently described as $\mathsf{BA}(-, \{\bot \lneq \top\})$. The pair $(\mathcal{Q}, \mathcal{F})$ is our leading example.

(ii) When $\mathscr{X}$ is the category of posets, the upset functor $\mathcal{U}\colon \mathsf{Pos} \to \mathsf{DLat}$ mapping a poset to the distributive lattice of upsets is dual to the prime filter functor, naturally isomorphic to $\mathsf{DLat}(-, \{\bot \lneq \top\})$.

(iii) The contravariant functor $\mathbb{S}\colon \mathsf{Meas} \to \mathsf{MSL}$ mentioned above is dual to the filter functor $\mathcal{F}$ which maps a semi-lattice A to its collection of filters with the σ-algebra generated by the units $(\eta(a))_{a \in A}$.

A wide range of state-based systems can be formulated as Set coalgebras, and beyond Set we have further examples, including *a*) descriptive Kripke frames as Stone coalgebras of the Vietoris topology [16]; *b*) positive Kripke frames as Pos

coalgebras of the convex powerset functor [13]; and *c*) Markov processes as Meas coalgebras of the Giry monad [9].

2.2 Factorisation systems

Definition 2.3 (see [3]) Given two classes $\mathcal{E}$, $\mathcal{M}$ of morphisms of a category $\mathscr{X}$, we say that $(\mathcal{E}, \mathcal{M})$ is a **factorisation system** if

(i) every morphism f has an $(\mathcal{E}, \mathcal{M})$-factorisation;

(ii) $\mathcal{E}$ and $\mathcal{M}$ contain all isomorphisms and are closed under composition.

(iii) it has the **diagonal fill-in property**; i.e., for each equation $g \circ e = m \circ f$ with $e \in \mathcal{E}$ and $m \in \mathcal{M}$ there is a unique morphism d such that the equations $d \circ e = f$ and $m \circ d = g$ hold.

We say that a factorisation system $(\mathcal{E}, \mathcal{M})$ is **proper** if $\mathcal{E}$ is contained in the class of epimorphisms and $\mathcal{M}$ in the class of monomorphisms.

Proposition 2.4 *For any factorisation system $(\mathcal{E}, \mathcal{M})$ on $\mathscr{X}$, the following statements are true:*

(i) *$\mathcal{E}$-morphisms are preserved by pushouts;*

(ii) *$\mathcal{M}$ is closed under limits in the* arrow category *$\mathscr{X}^{\rightarrow}$;*

(iii) *if $f \circ g$ and f are $\mathcal{M}$-morphisms, then so is g.*

2.3 Equationally presentable functors

Let $\mathscr{A}$ be a finitary variety with a left adjoint F to the forgetful functor $U \colon \mathscr{A} \to \mathsf{Set}$, and J the full inclusion of the subcategory $\mathscr{A}_\omega^f$ of $\mathscr{A}$ on free algebras Fn for $n \in \mathbb{N}$.

Definition 2.5 (see [4,29,6]) An endofunctor L of $\mathscr{A}$ is **finitely based** if it satisfies one of the following equivalent statements:

(i) L is finitary and preserves canonical presentations;

(ii) L is of the form $\mathrm{Lan}_J LJ$, a left Kan extension of LJ along J.

Let $\mathsf{FinB}[\mathscr{A}, \mathscr{A}]$ denote the full subcategory of the functor category $[\mathscr{A}, \mathscr{A}]$ on finitely based functors.

The notion of finitely based functors plays an important role in our theory of one-step semantics, since they are precisely the *equationally presentable* ones. Some elementary facts are given first.

Proposition 2.6 $\mathsf{FinB}[\mathscr{A}, \mathscr{A}]$ *is equivalent to* $[\mathscr{A}_\omega^f, \mathscr{A}]$.

Every variety $\mathscr{A}$ is locally presentable and (effectively) regular so the same holds for $\mathsf{FinB}[\mathscr{A}, \mathscr{A}]$ by this equivalence and $\mathscr{A}_\omega^f$ being small. Another useful fact about $\mathsf{FinB}[\mathscr{A}, \mathscr{A}]$ is that every functor has a finitely based coreflection:

Lemma 2.7 $\mathsf{FinB}[\mathscr{A}, \mathscr{A}]$ *is a coreflective subcategory of* $[\mathscr{A}, \mathscr{A}]$ *with the right adjoint* $\mathrm{Lan}_J(- \circ J)$.

It follows that $\mathsf{FinB}[\mathscr{A},\mathscr{A}]$ is closed under colimits of $[\mathscr{A},\mathscr{A}]$ since the inclusion is a left adjoint. Moreover, it can be shown further that the coreflector sends finite products to $\mathsf{FinB}[\mathscr{A},\mathscr{A}]$ unchanged:

Proposition 2.8 *$\mathsf{FinB}[\mathscr{A},\mathscr{A}]$ is closed under finite products of $[\mathscr{A},\mathscr{A}]$.*

The finitely based coreflection of L is written as $\rho_L \colon \widehat{L}_\omega \Rightarrow L$ with $\widehat{L}_\omega = \mathrm{Lan}_J LJ$.

The slogan 'every finitely based functor is equationally presentable' is justified as follows. A **signature** is a functor Σ from the discrete category $\mathbb{N}$ of natural numbers to Set. Every signature defines a finitely based endofunctor on $\mathscr{A}$ by

$$\widehat{H_\Sigma} := FH_\Sigma U \quad \text{where} \quad H_\Sigma X := \coprod_{n\in\mathbb{N}} \Sigma_n \times X^n. \tag{1}$$

Theorem 2.9 *The functor $\mathsf{FinB}[\mathscr{A},\mathscr{A}] \to [\mathbb{N},\mathsf{Set}]$ defined by mapping L to $(n \mapsto ULF)$, is finitary and monadic with a left adjoint defined by (1).*

By monadicity, every finitely based functor L has a *canonical presentation*, i.e. a coequaliser $\widehat{H_\Gamma} \rightrightarrows \widehat{H_\Sigma} \twoheadrightarrow L$ where $\Sigma = ULF$ and $\Gamma = U\widehat{H_\Sigma}F$ are signatures. By adjunction, the two parallel morphisms correspond to

$$\Gamma \rightrightarrows U\widehat{H_\Sigma}F = UFH_\Sigma UF, \tag{2}$$

so H_Σ and the parallel morphisms are the *generator* and the *equation* of the presentation, respectively.

Remark 2.10 For $n \in \mathbb{N}$, an element t of $U\widehat{H_\Sigma}Fn$ is to be understood as a term in $\mathscr{A}$ consisting of at most one layer of operations in Σ at terms of n variables, called a **rank-**1 **Σ-term**:

$$U\widehat{H_\Sigma}Fn = U \quad F \quad \overbrace{H_\Sigma \quad \underbrace{U \quad Fn}_{n\text{-ary terms}}}^{\Sigma\text{-operations}}$$

The two parallel morphisms (2) can be presented by a family of sets $\mathcal{E}_n \subseteq (U\widehat{H_\Sigma}Fn)^2$ of **rank-**1 **equations**, indexed by $n \in \mathbb{N}$.

Example 2.11 Modal algebras can be characterised as algebras for a finitely based functor $\mathbb{M}$, defined by the following BA-presentation [4]

$$\mathbb{M}A := \mathsf{BA}\langle\, \{\blacklozenge a\}_{a\in A} \mid \blacklozenge\bot = \bot, \blacklozenge(a \vee b) = \blacklozenge a \vee \blacklozenge b \,\rangle \tag{3}$$

and $\mathbb{M}f(\blacklozenge a) := \blacklozenge fa$ for each homomorphism f. This functor has an equational presentation consisting of a signature $\Sigma_1 := \{\blacklozenge\}$ and $\Sigma_n := \emptyset$ otherwise; and equations $\mathcal{E}_n$ by

$$\mathcal{E}_n := \begin{cases} \{\, \blacklozenge\bot \sim \bot \,\} & \text{if}\, n = 0 \\ \{\, \blacklozenge(a \vee b) \sim \blacklozenge a \vee \blacklozenge b \,\} & \text{if}\, n = 2 \\ \emptyset & \text{otherwise} \end{cases}$$

[4] A BA-presentation $\mathsf{BA}\langle\, \mathcal{G} \mid \mathcal{E} \,\rangle$ indicates an algebra generated by $\mathcal{G}$ subject to equations in $\mathcal{E}$.

where 2 is presented as $\{a, b\}$.

Remark 2.12 In categories such as Set and the category Vec_K of vector spaces over a field K, every finitely generated algebra is free, so finitary functors are finitely based. Moreover, every finitary endofunctor of BA is naturally isomorphic to a finitely based functor on non-trivial Boolean algebras [20]. However, this coincidence does not hold in general, and a counterexample can be found in [29, Example 3.12].

3 Categories of One-Step Semantics

We study properties of the category of all one-step semantics, including colimits, limits, a monoidal structure, coreflections, and its opposite category. For each type T of coalgebras, we explore the category of one-step semantics of T and show that there is always a 'full' one-step semantics to which every one-step semantics can be translated in a *unique way*.

☡ In the following context, $P \colon \mathscr{X} \to \mathscr{A}$ always denotes a contravariant functor. It is suppressed if an ambiguity is unlikely.

3.1 The category of all one-step semantics

Definition 3.1 A **one-step semantics over** P consists of an endofunctor T of $\mathscr{X}$, called the *type of one-step semantics*; an endofunctor L of $\mathscr{A}$, called the *syntax of modalities*; a natural transformation $\delta \colon LP \Rightarrow PT$, called the *interpretation of modalities*, denoted by (L, T, δ).

Proposition 3.2 *Every one-step semantics (L, T, δ) defines a functor P^δ from the category of T-coalgebras to the category of L-algebras, called the lifting of P.*

Example 3.3 (Classical modal logic) Kripke semantics for modal logic with its algebraic semantics defines a one-step semantics $(\mathbb{M}, \mathbb{P}, \delta)$ over $\mathcal{Q}$ as follows where $\mathbb{M}$ is given in (3) and $\mathbb{P}$ is the covariant powerset functor.

Define a natural transformation $\Diamond \colon 2^- \Rightarrow 2^{\mathbb{P}}$ by $S \mapsto \{\, U \in \mathbb{P}X \mid U \cap S \neq \emptyset \,\}$ for each subset $S \subseteq X$. Then, $\Diamond$ defines a natural transformation δ from $\mathbb{M}\mathcal{Q}$ to $\mathcal{Q}\mathbb{P}$ by mapping $\blacklozenge S$ to $\Diamond_X S$, since $\Diamond_X$ satisfies the above two equations in (3) for every X.

The lifting given by the one-step semantics $(\mathbb{M}, \mathbb{P}, \delta)$ is the functor mapping every $\mathbb{P}$-coalgebra $\xi \colon X \to \mathbb{P}X$ to the $\mathbb{M}$-algebra $\mathcal{Q}\xi \circ \delta_X \colon \mathbb{M}\mathcal{Q}X \to \mathcal{Q}X$ which is equivalent to the *complex algebra* of (X, ξ). The unique algebra homomorphism $[\![-]\!] \colon \Phi \to \mathcal{Q}X$ from the initial $\mathbb{M}$-algebra (Φ, α) to the complex algebra interprets every element in Φ to a predicate on X. The semantics of possibility, $[\![\blacklozenge\varphi]\!] = \{\, x \in X \mid \xi(x) \cap [\![\varphi]\!] \neq \emptyset \,\}$, follows from the commutativity of homomorphisms.

Polyadic predicate liftings for a Set functor T, i.e. natural transformations $\lambda \colon (2^-)^n \Rightarrow 2^T$ for the contravariant powerset functor 2^-, also provide a class of examples:

Example 3.4 (Predicate liftings) Every set Λ of polyadic predicate liftings for T gives rise to a one-step semantics $(L^\Lambda, T, \delta^\Lambda)$ over $\mathcal{Q}$ as follows. Let Λ_n denote the set of n-ary predicate liftings in Λ so that Λ is a signature $\Lambda \colon \mathbb{N} \to \mathsf{Set}$ and F the left adjoint to the forgetful functor $U \colon \mathsf{BA} \to \mathsf{Set}$. Define the syntax functor L^Λ by the signature Λ without equations, and the interpretation $\delta^\Lambda \colon L^\Lambda \mathcal{Q} \Rightarrow \mathcal{Q}T$ on generators by $(\lambda, S) \mapsto \lambda_X(S)$ for $\lambda \in \Lambda_n$ and each n-tuple $S = (S_i \subseteq X)_{i \in n}$.

Example 3.5 (Cover modality [18]) The cover modality for a finitary and weak-pullback preserving Set functor T with Boolean logic also defines a one-step semantics. Let $\in_X$ denote the membership relation on a set X. Define a function $\nabla^T_X \colon T2^X \to 2^{TX}$ by

$$\alpha \mapsto \{\, t \in TX \mid (t, \alpha) \in \overline{T}(\in_X) \,\}$$

where $(t, \alpha) \in \overline{T}(\in_X)$ if there is $w \in T(\in_X)$ with $T\pi_1(w) = t$ and $T\pi_2(w) = \alpha$. ∇^T_X is natural in X because T preserves weak pullbacks. By adjunction and $2^- = U\mathcal{Q}$, the transpose of ∇^T is a natural transformation $\overline{\nabla}^T$ from $FTU\mathcal{Q}$ to $\mathcal{Q}T$. That is, the cover modality defines a one-step semantics $(FTU, T, \overline{\nabla}^T)$.

Definition 3.6 The **category CoLog^P of one-step semantics** over P is defined to be the comma category $(P^* {\downarrow} P_*)$ from the precomposition $P^* \colon L \mapsto LP$ to the postcomposition $P_* \colon T \mapsto PT$. (Following our convention, we will usually suppress the superscript P.)

That is, the objects of CoLog are one-step semantics (L, T, δ) and a morphism from (L, T, δ) to (L', T', δ') is a pair of natural transformations $(\tau \colon L \Rightarrow L', \nu \colon T' \Rightarrow T)$ satisfying $P\nu \circ \delta = \delta' \circ \tau P$. The natural transformation τ is intuitively understood as a **translation** from syntax L to syntax L'. We will justify this intuition in Section 4. We denote the projection $(L, T, \delta) \mapsto L$ with U_L and the contravariant $(L, T, \delta) \mapsto T$ with U_R.

Proposition 3.7 (Coreflection) *Given a coreflective subcategory $\mathscr{B}$ of the functor category $[\mathscr{A}, \mathscr{A}]$, the pullback of the full inclusion functor $J \colon \mathscr{B} \hookrightarrow [\mathscr{A}, \mathscr{A}]$ along the forgetful functor $U_L \colon \mathsf{CoLog} \to [\mathscr{A}, \mathscr{A}]$ is also coreflective.*

Note that the pullback category is the full subcategory of CoLog of those one-step semantics whose syntax functor lies in $\mathscr{B}$.

Proof sketch Let $\rho_L \colon L^\dagger \Rightarrow L$ denote the coreflection of the functor L. Then, the coreflection of a one-step semantics (L, T, δ) is $(L^\dagger, T, \delta \circ \rho_L)$. □

3.1.1 Colimits and limits

A colimit of a J-indexed diagram D in CoLog can be constructed by a pointwise colimit $(\tau_i \colon L_i \Rightarrow L)_{i \in J}$ of $U_L D$ and a pointwise limit $(\nu \colon T \Rightarrow T_i)_{i \in J}$ of $(U_R D)^{\mathrm{op}}$

using the universal property:

$$\begin{tikzcd}
L_iP \arrow[r, "(U_LDf)P"] \arrow[d, "\iota_iP"'] \arrow[rrd, "\delta_i"] & L_jP \arrow[ld, "\iota_jP"] \arrow[rrd, "\delta_j"] & & \\
LP \arrow[rrd, dashed, "\delta"] & & PT_i \arrow[r, "P(U_RDf)"'] \arrow[d, "P\pi_i"'] & PT_j \arrow[ld, "P\pi_j"] \\
& & PT &
\end{tikzcd}$$

since by assumption LPX is a colimit for each component X.

Theorem 3.8 (Colimit) *The pair (U_L, U_R) of projections of* CoLog *creates pointwise colimits.*

A pointwise colimit means that it is constructed by a pointwise colimit in $[\mathscr{A}, \mathscr{A}]$ and a pointwise limit in $[\mathscr{X}, \mathscr{X}]$.

Example 3.9 (Multi-modal logic) For a set A of labels, the A-fold coproduct of classical modal logic $(\mathbb{M}, \mathbb{P}, \delta)$ is a multi-modal logic for A-labelled Kripke frames.

Limits in CoLog can be constructed similarly as

$$\mathrm{Lim}L_iP \overset{\delta=\mathrm{Lim}\delta_i}{\dashrightarrow} \mathrm{Lim}PT_i \cong P(\mathrm{Colim}T_i)$$

and $\mathrm{Lim}\delta_i$ is a pointwise limit in the arrow category, provided that P maps a colimit to a limit:

Theorem 3.10 (Limit) *Suppose that P has a dual adjoint S. Then, the pair (U_L, U_R) of projections creates pointwise limits.*

Example 3.11 An **alternating system** over a set A of actions [10] is a coalgebra for the functor $\mathcal{D} + \mathbb{P}^A$ (where $\mathcal{D}$ is the probability distribution functor). For such systems, a one-step semantics can be obtained as a product of a one-step semantics of type $\mathbb{P}^A$ and one of type $\mathcal{D}$. For the former we may take multi-modal logic, and the latter can be probabilistic modal logic induced by predicate liftings $\langle p \rangle$ for $\mathcal{D}$, indexed by $p \in \mathbb{Q} \cap [0, 1]$:

$$S \mapsto \{ \mu \in \mathcal{D}X \mid \sum \mu(S) \geq p \}$$

for each subset $S \subseteq X$.

3.1.2 Composition

The **composition** $\otimes$ of one-step semantics is defined by *pasting*:

$$\begin{tikzcd}
{} & {} \arrow[l, "L_1"'] & & {} & {} \arrow[l, "L_2"'] & & {} & {} \arrow[l, "L_1L_2"'] \\
{} \arrow[u] & {} \arrow[u] \arrow[l, "T_1"] \arrow[lu, phantom, "\swarrow\delta_1"] & \otimes & {} \arrow[u] & {} \arrow[u] \arrow[l, "T_2"] \arrow[lu, phantom, "\swarrow\delta_2"] & := & {} \arrow[u] & {} \arrow[u] \arrow[l, "T_1T_2"] \arrow[lu, phantom, "\Downarrow\delta_1\otimes\delta_2"]
\end{tikzcd}$$

i.e. $\delta_1 \otimes \delta_2 \coloneqq \delta_1 T_2 \circ L_1 \delta_2$.

Lemma 3.12 *The composition $\otimes$ of one-step semantics is a bifunctor, mapping each pair of morphisms $(\tau_o, \nu_o)\colon \delta_1 \to \delta_3$ and $(\tau_e, \nu_e)\colon \delta_2 \to \delta_4$ to a morphism from $\delta_1 \otimes \delta_2$ to $\delta_3 \otimes \delta_4$ and defined by the horizontal composites*

$$\left(\begin{tikzcd} {} \arrow[r, bend left, "L_2"{name=U1}] \arrow[r, bend right, "L_4"'{name=D1}] & {} \arrow[r, bend left, "L_1"{name=U2}] \arrow[r, bend right, "L_3"'{name=D2}] & {} \arrow[Rightarrow, from=U1, to=D1, "\tau_e"] \arrow[Rightarrow, from=U2, to=D2, "\tau_o"] \end{tikzcd}, \begin{tikzcd} {} \arrow[r, bend left, "T_2^{\mathrm{op}}"{name=U3}] \arrow[r, bend right, "T_4^{\mathrm{op}}"'{name=D3}] & {} \arrow[r, bend left, "T_1^{\mathrm{op}}"{name=U4}] \arrow[r, bend right, "T_3^{\mathrm{op}}"'{name=D4}] & {} \arrow[Rightarrow, from=U3, to=D3, "\nu_e"] \arrow[Rightarrow, from=U4, to=D4, "\nu_o"] \end{tikzcd}\right)$$

Theorem 3.13 (Monoidal structure) *The composition $\otimes$ of one-step semantics with the identity semantics $(\mathcal{I}, \mathcal{I}, id_P)$ is a strict monoidal structure on* CoLog*.*

Example 3.14 A **simple Segala system** over a set A of actions [28] is a coalgebra for the composite $\mathbb{P}^A \circ \mathcal{D}$. Thus, a one-step semantics for simple Segala systems can be obtained as the composite of the A-fold coproduct of $(\mathbb{M}, \mathbb{P}, \delta)$ and the one-step semantics of probabilistic modal logic.

3.1.3 Mate correspondence

To finish our study on CoLog, we study the mate correspondence of one-step semantics, a tool used first by Klin [14] to analyse one-step expressiveness of coalgebraic logic in a categorical approach:

Definition 3.15 Suppose that P has a dual adjoint S. The **mate** δ^* of a natural transformation $\delta\colon LP \Rightarrow PT$ is a natural transformation from TS to SL defined by the pasting diagram

$$\begin{tikzcd} {} & {} \arrow[r, "L"] \arrow[dr, phantom, "\Searrow\delta"] & {} \arrow[r, "S"] \arrow[dr, phantom, "\Searrow\epsilon"] & {} \\ {} \arrow[ur, "\mathcal{I}"] \arrow[r, "S"'] \arrow[ur, phantom, "\Searrow\eta"', shift right=3] & {} \arrow[u, "P"'] \arrow[r, "T^{\mathrm{op}}"'] & {} \arrow[u, "P"'] \arrow[ur, "\mathcal{I}"'] & \end{tikzcd}$$

in the opposite of $\mathscr{X}$, that is, $\delta^* = SL\eta \circ S\delta S \circ \epsilon TS$.

The mate operation maps a one-step semantics to an object of the comma category $\mathsf{CoLog}^* \coloneqq (S^* {\downarrow} S_*)$. Furthermore:

Proposition 3.16 CoLog *is dually isomorphic to* CoLog^* *where a one-step semantics (L, T, δ) is mapped to its mate (T, L, δ^*).*

Remark 3.17 By this isomorphism, colimits and limits in CoLog are transposed to their duals in CoLog^*, but more specifically, it can be shown that every (pointwise) colimit in CoLog can be constructed as a pointwise limit in the arrow category, to be used in Theorem 5.10. A bifunctor $\oplus$ defined by $\theta_1 \oplus \theta_2 \coloneqq \theta_1 L_2 \circ T_1 \theta_2$ with the identity $id\colon S \Rightarrow S$ defines a strict monoidal structure on CoLog^*, which is the image of $(\otimes, id)$ under the mate correspondence.

3.2 Fibre categories of CoLog

Definition 3.18 For every endofunctor T of $\mathscr{X}$, the category CoLog_T is defined to be the fibre category over T. More precisely, the objects of CoLog_T are one-step semantics of type T, denoted (L, δ). A morphism τ from (L, δ) to (L', δ') is a natural transformation satisfying $\delta = \delta' \circ \tau P$ (i.e. a syntax translation).

The type functor being fixed, we focus on the syntax projection functor which maps $U_L \colon (L, \delta) \mapsto L$ and $\tau \mapsto \tau$.

Proposition 3.19 *The following statements hold:*

(i) U_L *reflects isomorphisms.*

(ii) *For every coreflective subcategory* $\mathscr{C}$ *of the functor category* $[\mathscr{A}, \mathscr{A}]$*, the pullback of the full inclusion* $i \colon \mathscr{C} \hookrightarrow [\mathscr{A}, \mathscr{A}]$ *along* $U_L \colon \mathsf{CoLog}_T \to [\mathscr{A}, \mathscr{A}]$ *is coreflective.*

(iii) U_L *creates pointwise colimits.*

We are ready to establish a fundamental theorem for coalgebraic modal logics representable as one-step semantics:

Theorem 3.20 *Suppose that P has a dual adjoint S. Then every fibre CoLog_T has a terminal object*

$$(PTS,\ PT\epsilon \colon PTS\,P \Rightarrow PT)$$

where $\epsilon \colon \mathcal{I} \to SP$ is the counit of the dual adjunction.

In every fibre category CoLog_T, this terminal object is called the **full one-step semantics** for T. It may be too elusive for practical purposes, but conceptually it explains that every collection of modalities may be viewed as a (uniquely determined) fragment of this canonical one-step semantics.

4 Equational One-Step Semantics

We now focus on one-step semantics whose syntax functor is defined by operations and equations, called *equational*. To work with equational one-step semantics, we use finitely based functors (as introduced in Section 2) as syntax functors. Equational one-step semantics are characterised as (generalised) predicate liftings subject to equations. In particular, a full equational one-step semantics exists and is the logic of all predicate liftings subject to a complete axiomatisation *up to isomorphism.*

4.1 The category of equational one-step semantics

Definition 4.1 A one-step semantics $(L \colon \mathscr{A} \to \mathscr{A}, T, \delta)$ is (finitary) **equational** if $\mathscr{A}$ is a variety and L is finitely based. ECoLog is the corresponding full subcategory of CoLog.

The examples given in Section 3 are equational, since BA is a variety and $\mathbb{M}$, L^{Λ} for a set Λ of predicate liftings, and L^{∇^T} for a finitary and weak-pullback preserving functor T, are all finitely based.

It is not hard to show that the composite $L_1 \circ L_2$ of finitely based functors remains finitely based, so every composite of equational one-step semantics remains equational.

Proposition 4.2 *The composition* $\otimes$ *on* ECoLog *with the identity semantics* $(\mathcal{I}, \mathcal{I}, id_P)$ *is a strict monoidal structure on* ECoLog*.*

By applying Proposition 3.7 to the finitely based coreflection $\rho_L \colon \widehat{L}_\omega \Rightarrow L$, every one-step semantics has an equational coreflection:

Proposition 4.3 ECoLog *is a coreflexive subcategory of* CoLog*. Therefore,* ECoLog *is closed under colimits of* CoLog*.*

As we are actually interested in *equational* one-step semantics rather than all one-step semantics, the coreflection ensures that colimits are still constructed as in CoLog. By Proposition 2.8, this is also true for finite products:

Proposition 4.4 *Suppose that* P *has a dual adjoint. Pointwise finite products in* ECoLog *coincide with products in* CoLog*.*

Corollary 4.5 *The category of equational one-step semantics over the contravariant* 2^-*,* $\mathcal{Q}$*,* $\mathcal{U}$*, and* $\mathbb{S}$ *in Example 2.2, respectively, has colimits and finite products constructed pointwise.*

For example, the one-step semantics for alternating systems in Example 3.11 is indeed a product in ECoLog.

The universal property of (co)limits hints at certain optimal conditions. For instance, the fusion of predicate liftings [8], known as the smallest conservative extension of two given logics of predicate liftings, is in fact the coproduct in ECoLog of the corresponding equational one-step semantics:

Example 4.6 (Binary coproduct as fusion) Given two sets Λ_1 and Λ_2 of polyadic predicate liftings for Set functors T_1 and T_2, respectively, the coproduct of the one-step semantics induced by $(\Lambda_i)_{i=1,2}$ consists of $T_1 \times T_2$, as its type, and $L := L^{\Lambda_1} + L^{\Lambda_2} \cong \widehat{H_\Lambda}$ with $\Lambda := \Lambda_1 + \Lambda_2$, as its syntax, and as its interpretation the natural transformation $\delta \colon L\mathcal{Q} \Rightarrow \mathcal{Q}T$ defined for each set X on the generators of L by

$$(\lambda, S) \mapsto \left(\pi_i^{-1} \circ \lambda_X\right)(S)$$

for each $\lambda \in \Lambda_{i,n}$ and n-tuple $S = (S_j)_{j \in n}$.

4.2 Fibre categories of ECoLog

Definition 4.7 For any endofunctor T of $\mathscr{X}$, ECoLog_T is defined to be the fibre category of ECoLog over T.

By Proposition 3.19, each ECoLog_T is a coreflective subcategory of CoLog_T. Now the equational version of Theorem 3.20 follows, as the coreflector preserves limits, including the terminal object:

Corollary 4.8 *Suppose that* P *has a dual adjoint* S*. Every fibre* ECoLog_T *has a terminal object*

$$\left(\widehat{PTS}_\omega,\ PT\epsilon \circ \rho_P \colon \widehat{PTS}_\omega P \to PT\right)$$

where ρ is the finitely based coreflection $\widehat{PTS}_\omega \Rightarrow PTS$ and ϵ is the counit of the dual adjunction.

Again, this terminal object is called the **full equational one-step semantics** for T, and every equational one-step semantics for T is a fragment of it.

The remaining part of this section is used to describe equational one-step semantics as logics of predicate liftings subject to rank-1 equations.

4.2.1 Predicate liftings

Definition 4.9 Let $U\colon \mathscr{A} \to \mathsf{Set}$ be a functor and T an endofunctor of $\mathscr{X}$. A (finitary) **predicate lifting** for T is a natural transformation from $(UP)^n$ to UPT where $n \in \mathbb{N}$.

When $P = \mathcal{Q}$, this definition boils down to the usual definition of polyadic predicate liftings, since the underlying set $U\mathcal{Q}X$ is the powerset 2^X.

Example 4.10 When $P = \mathbb{S}\colon \mathsf{Meas} \to \mathsf{MSL}$, a predicate lifting for T maps a measurable set of a measurable space $X = (X, \mathbb{S}_X)$ to a measurable set on TX natural in X. Take the Giry monad $\mathcal{G}$, for example; $\mathcal{G}$ maps a space $(X, \mathbb{S}_X)$ to the collection of subprobability distributions $\mu\colon \mathbb{S}_X \to [0,1]$ satisfying

$$\mu(\emptyset) = 0 \quad \text{and} \quad \mu(\bigcup_i M_i) = \sum_i \mu(M_i)$$

for countable unions of pairwise disjoint measurable sets M_i (the σ-algebra on $\mathcal{G}X$ is ignored). Then a predicate lifting for $\mathcal{G}$ can be defined for each $p \in [0,1]$ by

$$M \mapsto \{\, \mu \in \mathcal{G}X \mid \mu(M) \geq p \,\}$$

for $M \in \mathbb{S}_X$, which is exactly the modality in [11, Section 4.3].

Assume that $\mathscr{A}$ is a variety and F denotes the left adjoint to the forgetful functor U. Every n-ary predicate lifting λ for T determines a one-step semantics (FU^n, λ^*), called the **unimodal logic** of λ, where $\lambda^*\colon F(UP)^n \Rightarrow PT$ is the transpose of λ by adjunction. A set Λ of predicate liftings for T determines a one-step semantics, called the **logic of predicate liftings**, consisting of

$$\widehat{H_\Lambda} = F\left(\coprod_{n\in\mathbb{N}} \Lambda_n \times (U-)^n\right) \quad \text{and} \quad \delta^\Lambda_X\colon \widehat{H_\Lambda}PX \Rightarrow PTX,$$

mapping generators (λ, S) to $\lambda_X(S)$ for $\lambda \in \Lambda_n$ and $S = (S_i \in UPX)_{i\in n}$.

Moreover, every finitely based functor L is a coequaliser of parallel morphisms $\widehat{H_\mathcal{E}} \rightrightarrows \widehat{H_\Sigma} \twoheadrightarrow L$, so characterisations follow readily:

Corollary 4.11 *For every endofunctor T of $\mathscr{X}$,*

(i) *every logic of predicate liftings for T is a coproduct of unimodal logics;*

(ii) *every object in ECoLog_T is a coequaliser of a logic of predicate liftings.*

4.2.2 Translations between equational one-step semantics

By Corollary 4.11, a morphism from (L', δ') to (L, δ) in ECoLog_T boils down to a family of translations from a unimodal logic to (L, δ), indexed by some set Λ of predicate liftings λ:

$$\begin{tikzcd}
F(UPX)^n \arrow[rr, "\tau_{PX}"] \arrow[dr, "\overline{\lambda}_X"'] & & LPX \arrow[dl, "\overline{\delta}_X"] \\
& PTX. &
\end{tikzcd}$$

Commutativity implies that a translation is not only a syntactic translation but also preserves the interpretation.

Example 4.12 (Continuing Example 3.5) Possibility $\Diamond$ and necessity $\Box$ can be translated to the cover modality ∇ by setting

$$\Box\varphi := \nabla\{\varphi\} \vee \nabla\emptyset \quad \text{and} \quad \Diamond\varphi := \nabla\{\varphi, \top\}.$$

The syntactic translation defines a morphism τ from the one-step semantics $(\mathbb{M}, \delta)$ to the one-step semantics $(F\mathbb{P}_\omega U, \overline{\nabla}^{\mathbb{P}_\omega})$ of the cover modality, i.e. $\overline{\nabla} \circ \tau\mathcal{Q} = \delta$. [5]

4.2.3 Equations valid under an interpretation

Let Λ be a set of predicate liftings. A rank-1 Λ-term in n variables can be interpreted by a function that maps any n-tuple of predicates on X to a predicate on TX as follows. By adjunction, every n-tuple $S = (S_i \in PX)$ is presented as a morphism $(Fn \overset{a}{\to} PX)$. Define the interpretation $[\![-]\!]^S_{\Lambda,X}$ for terms with n variables as the composite

$$U\widehat{H_\Lambda}Fn \xrightarrow{U\widehat{H_\Lambda}a} U\widehat{H_\Lambda}PX \xrightarrow{U\delta^\Lambda_X} UPTX$$

where δ^Λ_X is the logic of predicate liftings in Λ.

Definition 4.13 Given a set Λ of predicate liftings for T, a rank-1 equation $t \sim t'$ of Λ is **valid under the interpretation** of Λ if $[\![t]\!]^S_{\Lambda,X} = [\![t']\!]^S_{\Lambda X}$ for any X and n-tuple $(S_i \in UPX)$.

The universal quantifiers can be simplified to a pair of parallel morphisms:

Theorem 4.14 *Given a set Λ of predicate liftings for T and families of rank-1 Λ-equations $\mathcal{E}_n$ with n variables indexed by $n \in \mathbb{N}$, the following statements are equivalent:*

(i) *For each n, every equation $t \sim t' \in \mathcal{E}_n$ is valid under the interpretation of Λ.*

(ii) *The following diagram commutes*

$$\begin{tikzcd}
\widehat{H_\mathcal{E}}P \arrow[r, shift left, "\pi_1 P"] \arrow[r, shift right, "\pi_2 P"'] & \widehat{H_\Lambda}P \arrow[r, "\delta^\Lambda"] & PT
\end{tikzcd}$$

[5] For consistency, we restrict to the finitary powerset functor $\mathbb{P}_\omega$, otherwise $F\mathbb{P}U$ is not finitely based.

where $\pi_1, \pi_2 \colon \widehat{H_{\mathcal{E}}} \rightrightarrows \widehat{H_\Lambda}$ are the parallel morphisms induced by $\mathcal{E} = (\mathcal{E}_n)_{n\in\mathbb{N}}$.

Therefore, a regular quotient of a one-step semantics in ECoLog_T retains the set of modalities but they are subject to more equations. Moreover, every translation factors through a regular quotient:

Theorem 4.15 *For every T, the category ECoLog_T has $(\mathsf{RegEpi}, U^{-1}\mathsf{Mono})$ as a factorisation system where U is the forgetful functor $U \colon \mathsf{ECoLog}_T \to \mathsf{FinB}[\mathscr{A}, \mathscr{A}]$.*

This factorisation system also leads us to an important notion:

Definition 4.16 An equational one-step semantics for T is said to have a **complete axiomatisation** if it has no proper regular quotient in ECoLog_T.

Informally, a one-step semantics (L, δ) has a complete axiomatisation if every rank-1 equation valid under the interpretation δ is derivable from the presentation of L. Note that this is not model-theoretic completeness.

Example 4.17 The one-step semantics $(\mathbb{M}, \delta)$ of classical modal logic has a complete axiomatisation. Any proper regular quotient of $\mathbb{M}$ would identify a rank-1 equation which is not derivable from $\blacklozenge\bot = \bot$ and $\blacklozenge(a \vee b) = \blacklozenge a \vee \blacklozenge b$. However, classical modal logic is complete with respect to the class of all Kripke frames, so there exists an instance refuting the equation, a contradiction. Thus $(\mathbb{M}, \delta)$ has a complete axiomatisation.

4.2.4 Objects of predicate liftings

Using the Yoneda Lemma, Schöder observes [26] that n-ary predicate liftings in the case where $P = \mathcal{Q}$ are in bijection with the subsets of $T2^n$, because $U\mathcal{Q} = 2^-$ is naturally isomorphic to $\hom(-, 2)$. We generalise and combine this with Klin's *objects of T-modalities* [14], again employing the Yoneda Lemma and the *dual adjunction*:

Lemma 4.18 *Suppose that P has a dual adjoint S and $\mathscr{A}$ is a variety with a left adjoint F to the forgetful functor U. For any endofunctor T of $\mathscr{X}$, the following statements hold:*

(i) *For any $n \in \mathbb{N}$, the set $U(PTS)Fn$ is in bijection with the collection of n-ary predicate liftings.*

(ii) *The bijection is natural in objects n in the Kleisli category of the induced monad UF.*

That is, $PTSFn$ is precisely the *object of n-ary predicate liftings.* This ensures that the collection of all finitary predicate liftings is small, so a coproduct of all unimodal logics exists. Naturality means that for any function $f \colon n \to UFm$ between free algebras, the diagram

$$\begin{tikzcd}
UPTSFn \arrow[r, "{[-]}"] \arrow[d, "UPTS\bar{f}"'] & \mathsf{Nat}(UP^n, UPT) \arrow[d, "\hat{f}^*"] \\
UPTSFm \arrow[r, "{[-]}"'] & \mathsf{Nat}(UP^m, UPT)
\end{tikzcd}$$

commutes, where $[-]$ indicates the bijection from $UPTSFn$ to the collection of n-ary predicate liftings, $\bar{f}$ is the transpose of f, and $\hat{f}^*$ is pre-composition with

$$\hat{f}\colon (m \xrightarrow{a} UP) \mapsto (n \xrightarrow{f} UFm \xrightarrow{Ua} UFUP \xrightarrow{U\epsilon P} UP).$$

Remark 4.19 Note that the forgetful functor $U\colon \mathscr{A} \to \mathsf{Set}$ is naturally isomorphic to $\hom(F1, -)$ by adjunction, so the composite UP is naturally isomorphic to $\hom(-, SF1)$ by dual adjunction. Let Ω be $SF1$. Then, an n-ary predicate lifting is a natural transformation from $\hom(-, \Omega)^n$ to $\hom(T-, \Omega)$, which is in a more familiar form. Specifically, the underlying set of $SF1$ is a two-element set of *truth values* for the dual adjoints in Example 2.2.

4.2.5 Characterising the full equational one-step semantics

The finitely based part $\widehat{PTS}_\omega$ of PTS has a canonical presentation, so the full equational one-step semantics is a regular quotient of the one-step semantics consisting of

$$F\left(\coprod_{n\in\mathbb{N}} UPTSFn \times U^n\right) \quad \text{and} \quad (PT\epsilon \circ \rho P) \circ e \tag{4}$$

where $\rho\colon \widehat{PTS}_\omega \Rightarrow PTS$ is the coreflection and e the regular quotient. We call (4) the **logic of all predicate liftings** for T.

Lemma 4.20 *The following statements hold:*

(i) *The logic of all predicate liftings is a coproduct of all unimodal logics.*

(ii) *The full equational one-step semantics has a complete axiomatisation.*

Theorem 4.21 *An equational one-step semantics (L, δ) of type T is isomorphic to the full equational one-step semantics iff (L, δ) has a complete axiomatisation and every unimodal logic has a translation to (L, δ).*

Proof sketch By the previous lemmas, the 'only if' part follows. For the converse, let (L, δ) be a one-step semantics with a complete axiomatisation such that every unimodal logic has a translation to it. By assumption, there exists a mediating morphism τ from the logic $(L^\Lambda, \delta^\Lambda)$ of all predicate liftings to it, since $(L^\Lambda, \delta^\Lambda)$ is a coproduct of all unimodal logics. Consider the pushout of τ and the regular quotient

e from $(L^{\Lambda}, \delta^{\Lambda})$ to the full equational one-step semantics:

$$\begin{tikzcd}
(L^{\Lambda}, \delta^{\Lambda}) \arrow[r, two heads, "e"] \arrow[d, dashed, "\tau"'] & (\widehat{PTS}_{\omega}, PT\epsilon \circ \rho P) \arrow[d] \\
(L, \delta) \arrow[r, two heads] & (L', \delta').
\end{tikzcd}$$

Using the factorisation system on ECoLog_T and the fact that the full semantics is terminal, the statement follows. □

5 Logic of a One-Step Semantics

We remind the reader how a one-step semantics provides a coalgebraic modal logic, and refine Klin's expressiveness condition [14] for finitary functors. We also address the modularity problem of expressiveness.

For the sake of brevity, we restrict our discussion to the case of a one-step semantics which has a language (see below). Also, we assume that P has a dual adjoint S.

5.1 Logical setup

Definition 5.1 The **language** of (L, T, δ) is the initial L-algebra, denoted (Φ_L, α_L).

The initial L-algebra can be constructed by the *initial sequence* [1]. For a finitary functor L, the following ω-sequence starting from the initial object $\mathbf{0}$:

$$\begin{tikzcd}
\mathbf{0} \arrow[r, dashed, "!"] & L\mathbf{0} \arrow[r, "L!"] & \cdots \arrow[r] & L^i\mathbf{0} \arrow[r, "L^i!"] & \cdots
\end{tikzcd}$$

has a colimit $(f_i \colon L^i\mathbf{0} \to L^{\omega}\mathbf{0})_{i \in \omega}$. Then, by L being finitary, it is not hard to conclude that there exists an isomorphism $\beta \colon L^{\omega}\mathbf{0} \to LL^{\omega}\mathbf{0}$, so we can set $\Phi_L \coloneqq L^{\omega}\mathbf{0}$ and $\alpha_L \coloneqq \beta^{-1}$ for the initial L-algebra.

Definition 5.2 The **semantic interpretation** $[\![-]\!]_{(X,\xi)}$ of a language (Φ_L, α_L) in a T-coalgebra (X, ξ) is the unique L-algebra homomorphism from (Φ_L, α_L) to the L-algebra $(PX, P\xi \circ \delta)$.

The **logic** of (L, T, δ) refers to the language of L and its semantic interpretation. The interpretation $[\![-]\!]$ maps a formula φ, as an element of Φ, to the subset of X consisting of those states that satisfy φ. For example, if the syntax functor is induced by a set $\lambda \in \Lambda$ of (unary) predicate liftings, then $\alpha(\lambda, \varphi)$ represents the modal formula $\lambda\varphi$. The semantics $[\![\lambda\varphi]\!]$ on a T-coalgebra (X, ξ) is given by the diagram

$$\begin{tikzcd}
L\Phi \arrow[rr, "\alpha"] \arrow[d, "L[\![-]\!]"'] & & \Phi \arrow[d, dashed, "[\![-]\!]"] \\
LPX \arrow[r, "\delta_X"'] & PTX \arrow[r, "P\xi"'] & PX
\end{tikzcd}$$

so that $[\![\lambda\varphi]\!] = (P\xi \circ \delta_X \circ L[\![-]\!])(\lambda, \varphi) = P\xi(\lambda_X[\![\varphi]\!])$. By definition, *soundness* is easy to see, since any rank-1 equation encoded in the syntax functor L is valid in the language (Φ_L, α_L).

Definition 5.3 The **theory map** for a T-coalgebra (X, ξ) is the transpose $th\colon X \to S\Phi$ of the semantic interpretation $[\![-]\!]$ under the dual adjunction $S \dashv P$.

Intuitively, the theory map simply maps every state x to the collection of formulae satisfied by x, which is indeed the case for, say, the dual adjoints in Example 2.2.

Assuming that $\mathscr{X}$ has kernel pairs, define **logical equivalence** of a theory map th_ξ to be the kernel of th_ξ; a logic of (L, T, δ) may then be said to be **expressive** if the logical equivalence is contained in some kernel of a coalgebra homomorphism regarded as an $\mathscr{X}$-morphism. For concrete categories, such as Set and Meas, two elements x and y of a coalgebra (X, ξ) are logically equivalent if $th_\xi(x) = th_\xi(y)$; two elements are behaviourally equivalent if there exists a coalgebra homomorphism f with $fx = fy$, and thus a logic is expressive if logically equivalent elements are also behaviourally equivalent. [6]

It is known that behavioural equivalence implies logical equivalence, i.e., *adequacy* holds. The converse, *expressiveness*, is more interesting, and we turn to it next.

5.2 Expressiveness

We have shown that every equational one-step semantics has a unique translation to the full equational one-step semantics (Corollary 4.8), which is in fact the most expressive logic, so its expressiveness is equivalent to the existence of an expressive logic of some one-step semantics:

Theorem 5.4 *Every morphism in* CoLog_T *preserves expressiveness.*

Recall Klin's general condition for expressiveness in the functorial framework (in the formulation by Jacobs and Sokolova [11]):

Theorem 5.5 (see [14,11]) *Suppose that* $\mathscr{X}$ *has a proper factorisation system* $(\mathcal{E}, \mathcal{M})$*. Then, if a)* T *preserves* $\mathcal{M}$*-morphisms and b) the mate* δ^* *is a pointwise* $\mathcal{M}$*-morphism, then* $th(x) = th(y)$ *implies that* x *and* y *are behaviourally equivalent.*

Note that this result only gives sufficient conditions for expressiveness, but on the positive side, these conditions are particularly suitable for further generalisation. Hence, in the presence of a proper factorisation system, we define an equational one-step semantics of T to be **one-step expressive** if*a*) T preserves $\mathcal{M}$-morphisms, and *b*) the mate δ^* is a pointwise $\mathcal{M}$-morphism.

5.3 Expressiveness for finitary functors

We restrict attention to strongly locally finitely presentable categories:

[6] The justification of this point-free formulation may be found in [6, Section 4.1.5] (it assumes the existence of a proper factorisation system). In the following we will simply assume that $\mathscr{X}$ is concrete though this does not necessarily imply the existence of kernel pairs.

Definition 5.6 (see [2]) A locally finitely presentable category is **strongly locally finitely presentable** if for every cofiltered limit $(\sigma_i \colon Y \to Y_i)_{i \in I}$ and every monomorphism $f \colon X \hookrightarrow Y$ with X finitely generated, there is $i \in I$ such that the composite $\sigma_i \circ f$ is monic.

For example, Set, Pos, and Vec_K are strongly locally finitely presentable. Klin showed [14] that when $\mathscr{A}$ is a locally finitely presentable category the full *finitary* one-step semantics[7] of a finitary functor on a strongly locally presentable category is one-step expressive if the counit $\epsilon \colon \mathcal{I} \to \mathcal{SP}$ is pointwise monic. We adapt Klin's argument and apply it to the full *equational* one-step semantics in the case that $\mathscr{A}$ is a *variety*. By Theorem 5.4, the equational version also recovers the finitary one.

Theorem 5.7 (*c.f.* [14, Theorem 4.4]) *Let $\mathscr{X}$ be a strongly locally presentable category, $\mathscr{A}$ a variety, and $T \colon \mathscr{X} \to \mathscr{X}$ a finitary and monomorphism-preserving functor. If the counit $\epsilon \colon \mathcal{I} \to \mathcal{SP}$ is pointwise monic, then the full equational one-step semantics of T is one-step expressive.*

Proof sketch Klin's theorem for full finitary one-step semantics is established in two steps. Let $\mathscr{A}_\omega$ be the full (small) subcategory on finitely presentable objects. First it is shown that if for every A the source

$$\left\{TSf \colon TSA \to TSA_i\right\}_{f \in (\mathscr{A}_\omega \downarrow A)} \tag{5}$$

is jointly monic, then the mate δ^* is pointwise monic where $(\mathscr{A}_\omega \downarrow A)$ is the comma category from $\mathscr{A}_\omega$ to A. Second, the family (5) is shown to be jointly monic by the strong local presentability.

For the same reason, to show that δ^* is pointwise monic, it suffices to show that the source

$$\left\{TSg \colon TSA \to TSFn\right\}_{g \in (\mathscr{A}_\omega^f \downarrow A)} \tag{6}$$

is jointly monic. However, every morphism $(Fn \xrightarrow{g} A)$ factors through a regular epimorphism $e \colon Fn \twoheadrightarrow B$ with B finitely presentable, and TSe is a monomorphism by the dual adjunction and assumption. It is easy to see that (6) is jointly monic if and only if (5) is jointly monic. Then Klin's second step completes this proof. □

5.4 Modularity of one-step expressiveness

As we discussed colimits, finite products, and compositions on CoLog, it is of interest to know if one-step expressiveness is stable under these constructions at this level of generality. Surprisingly, compositions and colimits preserve one-step expressiveness in a straightforward way:

Theorem 5.8 *The composite $\delta_1 \otimes \delta_2$ of one-step expressive semantics (L_1, T_1, δ_1) and (L_2, T_2, δ_2) is one-step expressive for $T_1 T_2$.*

[7] This terminology is defined analogously: A one-step semantics is **finitary** if its syntax functor is finitary on a locally finitely presentable category. The **full finitary one-step semantics** for T is the terminal object in the category of finitary one-step semantics for T by Proposition 3.19.

Proof By Remark 3.17, the mate of $\delta_1 \otimes \delta_2$ is equal to $\delta_1^* L_2 \circ T_1 \delta_2^*$. By assumption, T_i preserves $\mathcal{M}$-morphisms and δ_i^* is a pointwise $\mathcal{M}$-morphism for $i = 1, 2$. Hence the composite $\delta_1^* L_2 \circ T_1 \delta_2^*$ is a pointwise $\mathcal{M}$-morphism. By assumption, $T_1 T_2$ preserves $\mathcal{M}$-morphisms. □

Example 5.9 The double finite powerset functor $\mathbb{P}_\omega \circ \mathbb{P}_\omega$ does not have a separating set of unary predicate liftings [26]. However, we may simply self-compose the usual one-step semantics for $\mathbb{P}_\omega$ to obtain a one-step expressive logic.

Theorem 5.10 *The pointwise colimit of one-step expressive semantics is one-step expressive.*

Proof Let D be a diagram in CoLog. The colimit of D is a one-step semantics of type $\mathrm{Lim}_i T_i$. By assumption, each T_i preserves $\mathcal{M}$-morphisms and by Proposition 3.19, $\mathcal{M}$-morphisms are closed under limits in $\mathscr{X}^{\to}$, so $\mathrm{Lim}T_i$ preserves $\mathcal{M}$-morphisms.

The mate of $(\mathrm{Colim}D)$ is a pointwise $\mathcal{M}$-morphism: Every D_i^* is a pointwise $\mathcal{M}$-morphism by assumption, so the limit of D_i^* in the arrow category $\mathscr{X}^{\to}$ is also a pointwise $\mathcal{M}$-morphism. By Remark 3.17, $\mathrm{Lim}(D_i^*)$ is isomorphic to $(\mathrm{Colim}D_i)^*$ and the latter is a pointwise $\mathcal{M}$-morphism since $\mathcal{M}$ contains isomorphisms. □

Example 5.11 (Labelling T^A) Suppose that $\mathscr{X}$ and $\mathscr{A}$ have products and coproducts, respectively. Let A be a set of labels. Every coalgebra $\xi \colon X \to T^A X$ for the A-fold product of T corresponds to a family $(\xi_a)_{a \in A}$ of T-coalgebras, i.e. an A-labelled T-coalgebra, and the A-fold coproduct of a one-step semantics (L, T, δ) defines a one-step semantics for T^A. Moreover, the coproduct is one-step expressive if and only if (L, T, δ) is one-step expressive. The result applies immediately to $\mathbb{P}$, $\mathcal{D}$, the convex powerset functor $\hat{\mathbb{P}}$, and the Giry monad $\mathcal{G}$, to name but a few.

As for finite products, we are encouraged by the result of Cîrstea's and Pattinson [7] that one-step expressiveness is preserved by finite products for one-step semantics over 2^-, but we do not have a general proof at this point.

Acknowledgement

We would like to thank Paul Levy, Clemens Kupke and Alexander Kurz for insightful discussions and feedback on the current work.

References

[1] Jiří Adámek. Free algebras and automata realizations in the language of categories. *Comment. Math. Univ. Carolinae*, 15(4):589–602, 1974.

[2] Jiří Adámek. On final coalgebras of continuous functors. *Theor. Comput. Sci.*, 294(1-2):3–29, 2003.

[3] Jiří Adámek, Horst Herrlich, and George E. Strecker. *Abstract and Concrete Categories*. Wiley-Interscience, New York, 1990.

[4] Marcello Bonsangue and Alexander Kurz. Presenting functors by operations and equations. In Luca Aceto and Anna Ingólfsdóttir, editors, *Found. Softw. Sci. Comput. Struct.*, volume 3921 of *Lecture Notes in Computer Science*, pages 172–186. Springer Berlin Heidelberg, 2006.

[5] Marcello Bonsangue and Alexander Kurz. Pi-calculus in logical form. In *22nd Annu. IEEE Symp. Log. Comput. Sci.*, pages 303–312. IEEE, 2007.

[6] Liang-Ting Chen. *On a Purely Categorical Framework for Coalgebraic Modal Logic*. PhD thesis, University of Birmingham, 2013. http://www.cs.bham.ac.uk/~axj/pub/papers/thesis_chen.pdf.

[7] Corina Cîrstea and Dirk Pattinson. Modular construction of complete coalgebraic logics. *Theor. Comput. Sci.*, 388(1-3):83–108, 2007.

[8] Fredrik Dahlqvist and Dirk Pattinson. On the fusion of coalgebraic logics. In Andrea Corradini, Bartek Klin, and Corina Cîrstea, editors, *Algebr. Coalgebra Comput. Sci.*, Lecture Notes in Computer Science, pages 161–175. Springer Berlin Heidelberg, 2011.

[9] Josée Desharnais, Abbas Edalat, and Prakash Panangaden. Bisimulation for Labelled Markov Processes. *Inf. Comput.*, 179(2):163–193, 2002.

[10] H. Hansson and B. Jonsson. A calculus for communicating systems with time and probabilities. In *Real-Time Syst. RTSS 90*, pages 278–287. IEEE, 1990.

[11] Bart Jacobs and Ana Sokolova. Exemplaric expressivity of modal logics. *J. Log. Comput.*, 20(5):1041–1068, 2010.

[12] Peter T. Johnstone. *Stone spaces*. Cambridge University Press, 1982.

[13] Krzysztof Kapulkin, Alexander Kurz, and Jiří Velebil. Expressiveness of positive coalgebraic logic. In *Adv. Modal Log.*, pages 368–385, 2012.

[14] Bartek Klin. Coalgebraic modal logic beyond sets. *Electron. Notes Theor. Comput. Sci.*, 173:177–201, 2007.

[15] Clemens Kupke, Alexander Kurz, and Dirk Pattinson. Algebraic semantics for coalgebraic logics. *Electron. Notes Theor. Comput. Sci.*, 106:219–241, 2004.

[16] Clemens Kupke, Alexander Kurz, and Yde Venema. Stone coalgebras. *Theor. Comput. Sci.*, 327(1–2):109–134, 2004.

[17] Alexander Kurz. Coalgebras and their logics. *SIGACT News*, 37(2):57–77, 2006.

[18] Alexander Kurz and Raul A. Leal. Modalities in the stone age: A comparison of coalgebraic logics. *Theor. Comput. Sci.*, 430:88–116, 2012.

[19] Alexander Kurz and Daniela Petrişan. Presenting functors on many-sorted varieties and applications. *Inf. Comput.*, 208(12):1421–1446, 2010.

[20] Alexander Kurz and Jiří Rosický. Strongly complete logics for coalgebras. *Log. Methods Comput. Sci.*, 8:1–32, 2012.

[21] Raul A. Leal. Predicate liftings versus Nabla modalities. *Electron. Notes Theor. Comput. Sci.*, 203(5):195–220, 2008.

[22] Saunders Mac Lane. *Categories for the Working Mathematician*. Springer, 2 edition, 1998.

[23] Lawrence S. Moss. Coalgebraic Logic. *Ann. Pure Appl. Log.*, 99(1-3):241–259, 1999.

[24] Dirk Pattinson. Coalgebraic modal logic: soundness, completeness and decidability of local consequence. *Theor. Comput. Sci.*, 309(1-3):177–193, 2003.

[25] Dirk Pattinson. Expressive logics for coalgebras via terminal sequence induction. *Notre Dame J. Form. Log.*, 45(1):19–33, 2004.

[26] Lutz Schröder. Expressivity of coalgebraic modal logic: The limits and beyond. *Theor. Comput. Sci.*, 390(2-3):230–247, 2008.

[27] Lutz Schröder and Dirk Pattinson. Modular algorithms for heterogeneous modal logics via multi-sorted coalgebra. *Math. Struct. Comput. Sci.*, 21(02):235–266, 2011.

[28] Roberto Segala. *Modeling and Verification of Randomized Distributed Real-Time Systems*. PhD thesis, Massachusetts Institute of Technology, 1995.

[29] Jiří Velebil and Alexander Kurz. Equational presentations of functors and monads. *Math. Struct. Comput. Sci.*, 21(02):363–381, 2011.

Total Maps of Turing Categories

J.R.B. Cockett [1,2]

Department of Computer Science
University of Calgary
Calgary, Alberta, Canada, T2N 1N4

P.J.W. Hofstra[3]

Department of Mathematics and Statistics
University of Ottawa
Ottawa, Ontario, Canada, K1N 6N5

P. Hrubeš[4]

Department of Computer Science and Engineering
University of Washington
Seattle, WA, USA

Abstract

We give a complete characterization of those categories which can arise as the subcategory of total maps of a Turing category. A Turing category provides an abstract categorical setting for studying computability: its (partial) maps may be described, equivalently, as the computable maps of a partial combinatory algebra. The characterization, thus, tells one what categories can be the total functions for partial combinatory algebras. It also provides a particularly easy criterion for determining whether functions, belonging to a given complexity class, can be viewed as the class of total computable functions for some abstract notion of computability.

Keywords: Computability theory, Partial Combinatory Algebra, Turing Category, Complexity Theory.

1 Introduction

Turing categories [1,2,6], provide an abstract categorical setting for computability theory which, unlike partial combinatory algebras (PCAs) and related structures, are presentation-independent and purely formulated in terms of categorical properties. The standard example of a Turing category has objects powers of the natural

[1] This work was partial supported by NSERC Canada

[2] Email: robin@ucalgary.ca

[3] Email: phofstra@uottawa.ca

[4] Email: pahrubes@gmail.com

numbers and maps all the partial recursive functions. However, there are many other less well-known examples deriving from the "computable maps" of PCAs (or more generally relative PCAs) or from syntactical methods. Of special relevance for the purposes of the present work are the Turing categories described in [3] which have as total maps the programs belonging to various complexity classes (PTIME, LOGSPACE, etc.). These examples naturally lead to the question of exactly what categories can be the total maps of a Turing category. Intuitively, as in any Turing category one can simulate all computable functions, it would seem reasonable to suppose that the total maps would have to satisfy some fairly demanding closure properties.

The question is of significance for various reasons. To start with, it is one way to determine the limits of the applicability of Turing categories in studying computability. If it where impossible for the total maps of a Turing category to be exactly, say, the linear time functions, then one cannot hope to use Turing categories as a basis for investigating feasible computation at very low complexity levels. On the other hand, if one knows that the total maps of a Turing category *can* be of such low complexity then Turing categories can be a tool for formally unifying computability and complexity theory and allowing a fluid flow of ideas between the subjects.

A second reason for considering this question in the abstract categorical setting is that it leads to an interesting comparison with the more traditional view on these matters, namely via logic. Given a logical theory, one may ask which relations are provably total: the weaker the theory, the smaller the class of provably total maps. One may also wish for the system to be strong enough to allow for the *representation* of partial computable maps. It is well known that even relatively weak fragments of arithmetic ensure this. For example, Robinson's Arithmetic Q is enough to ensure all the partial recursive functions are represented. The study of complexity, using bounded and two sorted logics [5] for example, has further pushed the limits of these methods. It is, thus, in its own right, an interesting question to know exactly what an absolutely minimal logic for generating these settings really is. Although, this paper does not attempt to answer this question directly, the categorical framework we describe can certainly be backward engineered into a logical form: even a brief perusal indicates that there is a significant difference between these approaches, not least because a Turing category is not *a priori* based on arithmetic.

The third point of interest lies not so much in the question itself as in the methods used here to provide the answer. The proof that a cartesian category satisfying certain conditions can be embedded as a category of total maps in a Turing category makes use of two ideas: first, it uses the Yoneda embedding to create a canonical category of partial maps into which the original category embeds. Next, we use the concept of a *stack machine* in the presheaf category to create a partial combinatory algebra which in turn will generate the desired Turing category. A stack machine can be thought of as a categorical implementation of the canonical rewrite system in combinatory logic (but augmented with additional data). As such, this concept helps clarify the connection between syntactical approaches to generating models of computation and categorical methods.

Below we develop necessary and sufficient conditions for a Cartesian category to be the total maps of a Turing category. The conditions are perhaps a little surprising as apparently very little is actually required: there must be a universal object, U, which has a pair of "disjoint" elements, and which has an "abstract retract" structure which allows coding of maps. A universal object is an object into which every other object can be embedded as a subobject: to have such an object is already a somewhat non-trivial requirement as this, in particular, implies there is an embedding $U \times U \longrightarrow U$ showing that U must be an infinite object. The remaining conditions are somewhat more technical: they are explained in section 3 below. However, it should be noted that in most of the standard applications, as described in section 5, these technical conditions can be by-passed.

We start the exposition by considering the much more general question of how a category can arise as the subcategory of total maps of an arbitrary restriction category. (For background on restriction categories we refer to [4].) This leads to the notion of a totalizing extension of the given category, and we show that the category of totalizing extensions of a particular category has a final object which is naturally a restriction category. This insight allows us to transfer the general question of finding a Turing category which extends a given cartesian category into finding a partial combinatory algebra in this final extension whose total maps include the given maps. This perspective allows us to propose necessary conditions for a Cartesian category to be the total maps of a Turing category, see section 3. To show that these are sufficient we demonstrate that one can build, using a simple abstract machine, a combinatory algebra in the final totalizing extension (actually of a slightly modified category) which has elements representing all the total maps: this suffices in view of Theorem 4.12 of [2]. Finally we provide simpler sufficient conditions to show the wide applicability of the theorem.

2 Totalizing map subcategories

When $\mathbb{X}$ is a restriction category, the inclusion of the subcategory of total maps $\mathrm{Total}(\mathbb{X}) \longrightarrow \mathbb{X}$ satisfies various properties. This leads to the following definition:

Definition 2.1 [Totalizing functor] A functor $T : \mathbb{X}' \longrightarrow \mathbb{X}$ is **totalizing** when it satisfies the following three conditions:

(i) T on objects, $T_{\mathsf{Obj}} : \mathsf{Obj}(\mathbb{X}') \longrightarrow \mathsf{Obj}(\mathbb{X})$, is an isomorphism,

(ii) T is faithful,

(iii) T is left factor[5] closed, meaning that when $Th = gf$ then there is a (necessarily unique) k such that $Tk = f$.

The following observation indicates that this class of functors is reasonably well-behaved.

[5] Note that "left" factor refers to the diagrammatic order of composition: $A \xrightarrow{Th} B = A \xrightarrow{f} B \xrightarrow{g} C$.

Lemma 2.2 *In the category of categories and functors, the class of totalizing functors, $\mathcal{T}$, form a stable system of monics:*

(i) Every $T \in \mathcal{T}$ is monic,

(ii) $\mathcal{T}$ is closed to composition,

(iii) $\mathcal{T}$ contain all isomorphisms,

(iv) $\mathcal{T}$ is closed to pulling back along any functor.

Proof. Routine. □

Recall that given any (small) category $\mathbb{X}$ the Yoneda embedding, $y : \mathbb{X} \longrightarrow \mathsf{Set}^{\mathbb{X}^{\mathrm{op}}}$, gives a full and faithful embedding of $\mathbb{X}$ into presheaves on $\mathbb{X}$. This category of presheaves is finitely complete and therefore one can form the partial map category on all monics $\mathsf{Par}(\mathsf{Set}^{\mathbb{X}^{\mathrm{op}}})$. Notice that the total maps in this category are exactly the morphisms of $\mathsf{Set}^{\mathbb{X}^{\mathrm{op}}}$. Restricting this category to the representables gives a full subcategory which we shall denote $\widetilde{\mathbb{X}}$. While this is in general no longer a partial map category, it certainly is a restriction category. We then have the following pullback:

$$\begin{tikzcd}
\mathbb{X} \arrow[r, "\eta"] \arrow[d, "y"'] & \widetilde{\mathbb{X}} \arrow[d, "\widetilde{y}"] \\
\mathsf{Set}^{\mathbb{X}^{\mathrm{op}}} \arrow[r] & \mathsf{Par}(\mathsf{Set}^{\mathbb{X}^{\mathrm{op}}})
\end{tikzcd}$$

This exhibits $\mathbb{X}$ as a subcategory of total maps of the restriction category $\widetilde{\mathbb{X}}$, whence η is totalizing.

While there may be many other categories $\mathbb{Y}$ into which $\mathbb{X}$ embeds via a totalizing functor, we shall now make precise the sense in which $\eta : \mathbb{X} \longrightarrow \widetilde{\mathbb{X}}$ is the universal such functor.

Consider the category of **totalizing extensions** of $\mathbb{X}$, denoted $\mathsf{Ext}_{\mathcal{T}}(\mathbb{X})$, whose objects are totalizing functors $T : \mathbb{X} \longrightarrow \mathbb{Y}$ and whose morphisms are commuting triangles below $\mathbb{X}$

$$\begin{tikzcd}
 & \mathbb{X} \arrow[dl, "T"'] \arrow[dr, "T'"] & \\
\mathbb{Y} \arrow[rr, "F"'] & & \mathbb{Y}'
\end{tikzcd}
\qquad
\begin{tikzcd}
\mathbb{X} \arrow[r, equal] \arrow[d, "T"'] & \mathbb{X} \arrow[d, "T'"] \\
\mathbb{Y} \arrow[r, "F"'] & \mathbb{Y}'
\end{tikzcd}$$

which **reflect total maps** in the sense that the square on the right is a pullback.

Proposition 2.3 $\mathsf{Ext}_{\mathcal{T}}(\mathbb{X})$ *is a (finitely) complete category in which $\eta : \mathbb{X} \longrightarrow \widetilde{\mathbb{X}}$ is the final object.*

Proof. The verification that pullback exist (and are constructed as in Cat is routine. We shall show that $\eta : \mathbb{X} \longrightarrow \widetilde{\mathbb{X}}$ is the final object in this category. To this end, suppose $T : \mathbb{X} \longrightarrow \mathbb{Y}$ is totalizing. Without loss of generality we may assume that $\mathbb{Y}$ has the same collection of objects as $\mathbb{X}$. It is then obvious how the functor $E : \mathbb{Y} \longrightarrow \widetilde{\mathbb{X}}$ should act on objects. For a morphism $g : X \longrightarrow X'$ in $\mathbb{Y}$, we need to define a partial map $y(X) \longrightarrow y(X')$. First consider the following sieve on X:

$$S_g = \{h : Z \longrightarrow X | gh \in \mathbb{X}\}.$$

This sieve corresponds to a subobject of the representable $y(X)$. Next, define a natural transformation $\tau(g) : S_g \longrightarrow y(X')$ by

$$\tau(g)_Z : S_g(Z) \longrightarrow \mathbb{X}(Z, X'); \qquad \tau(g)_Z(h) = gh.$$

This data defines a partial map $E(g) : y(X) \longrightarrow \mathcal{Y}(X')$, which is total if and only if $g \in \mathbb{X}'$. This in particular shows that E reflects total maps. The verification that E is functorial is straightforward and left to the reader.

To show that E is unique suppose we are given an extension $E' : \mathbb{Y} \longrightarrow \widetilde{\mathbb{X}}$. Since E' must respect maps from $\mathbb{X}$, we verify its action on a map $g : X \longrightarrow X'$ which is not in $\mathbb{X}$. Then $E'(g)$ is a span $y(X) \supseteq S \xrightarrow{\sigma} y(X')$. Now if $h \in S_g$, the composite gh is in $\mathbb{X}$, and therefore $h \in S$, whence $E'(gh) = E'(g)E'(h) = \sigma(h)$ must equal $y(gh)$. This means that $E'(g) \geq E(g)$. To show the converse, assume that $h \in S$ but $h \notin S_g$. Then $gh \notin \mathbb{X}$. But then $E'(gh)$ is a total map, contradicting the reflection of total maps. □

We also note that in case $\mathbb{Y}$ is a restriction category and $T : \mathbb{X} \longrightarrow \mathbb{Y}$ the inclusion of total maps, then in fact $E : \mathbb{Y} \longrightarrow \widetilde{\mathbb{X}}$ is a restriction functor. In addition, we have the following result, which states that if $\mathbb{X}$ has products, then E preserves the induced restriction products.

Lemma 2.4 *Suppose that $\mathbb{X}$ has finite products. Then $\widetilde{\mathbb{X}}$ also has finite restriction products, and η preserves them.*

Proof. Finite products in a split restriction category are completely determined by their counterparts in the total map subcategory. Thus, as the inclusion $\mathsf{Set}^{\mathbb{X}^{\mathrm{op}}}$ $\longrightarrow \mathsf{Par}(\mathsf{Set}^{\mathbb{X}^{\mathrm{op}}})$ preserves products and $y : \mathbb{X} \longrightarrow \mathsf{Set}^{\mathbb{X}^{\mathrm{op}}}$ preserves products cutting down to $\widetilde{\mathbb{X}}$ preserves products. □

With a view towards our aim of characterizing those categories which arise as the total map subcategory of a Turing category, we can now observe the following. When $\mathbb{X} \longrightarrow \mathbb{T}$ is a totalizing extension with $\mathbb{T}$ a Turing category, then we have the following situation:

$$\begin{tikzcd}
\mathbb{X} \arrow[r] \arrow[dr, "\eta"'] & \mathbb{T} \arrow[d, "E"] \\
 & \widetilde{\mathbb{X}}
\end{tikzcd}$$

Then by the above lemma, E is a Cartesian restriction functor; since any such functor preserves partial combinatory algebras, we find that there is a PCA $E(U)$, the image of the Turing object $U \in \mathbb{T}$. Thus we obtain:

Proposition 2.5 *$\mathbb{X}$ is the total map category of a Turing category if and only if there is a combinatory algebra in $\widetilde{\mathbb{X}}$ whose total computable maps include the maps of $\mathbb{X}$.*

Thus we may see our original problem as one of finding a suitable PCA in $\widetilde{\mathbb{X}}$ for which all the maps in $\mathbb{X}$ are computable. Indeed, if such a PCA exists, then we may let $\mathbb{T}$ be the subcategory of $\widetilde{\mathbb{X}}$ on the computable maps. Notice one rather nice aspect of this reformulation of the problem: since the maps of $\mathbb{X}$ already account for *all* the total maps in $\widetilde{\mathbb{X}}$, the total maps represented by a PCA in $\widetilde{\mathbb{X}}$ necessarily lie in $\mathbb{X}$ already; thus we don't have to worry about having too many total maps represented. Therefore, the only thing to verify when constructing a candidate PCA is that it represents all the maps in $\mathbb{X}$.

3 Properties of the total maps of a Turing category

We are now in a position to collect the necessary conditions for being the total maps of a Turing category. Clearly the total map category has to be Cartesian (i.e. has finite products), as Turing categories are Cartesian by definition. The remaining conditions are somewhat more technical. However, it should be stressed that in many cases of interest these conditions greatly simplify.

The purpose of the section is to prove:

Proposition 3.1 *Every total map category of a Turing category has a universal object which has a pair of disjoint elements and is equipped with an abstract retract structure for which there exist codings.*

Below we introduce the required notions of having a *universal object*, *disjoint elements*, *abstract retract structure*, and *codes*, showing each is present in the total map category of a Turing category.

3.1 A universal object

One of the properties of the Turing object U in a Turing category is that every object is a retract of it. More explicitly, given any object A, there are morphisms $\iota_A : A \longrightarrow U$ and $\rho_A : U \longrightarrow A$ for which $\rho_A \iota_A = 1_A$. This forces ι_A to be a total map (in fact, a monomorphism), but ρ_A can still be partial. We will typically denote the situation by $A \prec U$, or by $(\iota_A, \rho_A) : A \prec U$ if necessary.

We call an object U in a category **universal** if for every object A there exists a monomorphism $\iota_A : A \longrightarrow U$. (Thus we don't ask that every object is a retract of U; this is stronger, and will be analyzed in the section on abstract retract structures below.)

Lemma 3.2 *In any Cartesian category $\mathbb{X}$ with a universal object U:*

(i) *There is always an element $\iota_1 : 1 \longrightarrow U$ of the universal object;*

(ii) *There is always an embedding $\iota_{U \times U} : U \times U \longrightarrow U$;*

(iii) *The homset $\mathbb{X}(U, U)$ either has one element, in which case $\mathbb{X}$ is trivial, or is infinite.*

3.2 A pair of disjoint elements

A Turing object U in a Turing category admits the interpretation of all combinatory logic terms. In particular, U has two global elements $\mathsf{t} = \lambda^* xy.x$ and $\mathsf{f} = \lambda^* xy.y$. We shall prove that these two elements have the property of being **disjoint** in the sense that whenever $\mathsf{t}h = \mathsf{f}h$ for a map h, the domain $\overline{h}$ of h must be a *strict* initial object in the idempotent splitting. For the subcategory of total maps, this simply means that for any h with $\mathsf{t}h = \mathsf{f}h$ the domain of h is a strict initial object.

Proposition 3.3 *In every Turing category the total maps have a universal object U with a disjoint pair of elements.*

Notice first that when the Turing category is trivial (in the sense that it is equivalent to the terminal category) all objects are strict initial objects, and hence all elements are disjoint. In general, in a Turing category, an element will be disjoint from itself only when the category is trivial as this forces the initial and final object to be the same.

We start by observing:

Lemma 3.4 *Let $\mathbb{X}$ be a Cartesian category. A map $h : H \longrightarrow 1$ makes*

$$\begin{tikzcd}
H \times A \arrow[r, "h \times 1"] & 1 \times A \arrow[r, "\pi_1"] & A \arrow[r, shift left, "f"] \arrow[r, shift right, "g"'] & B
\end{tikzcd}$$

commute for all $f, g : A \longrightarrow B$ if and only if H is a strict preinitial object.

Recall that an object H is called **preinitial** if there is at most one map from that object to any other object. It is **strict preinitial** if any object with a map to H is itself (strict) preinitial. Preinitial objects are quite common: for example in the category of (commutative) rings $\mathbb{Z}$ is the initial object, while $\mathbb{Z}_n$ is preinitial for each n. However, neither $\mathbb{Z}$ nor $\mathbb{Z}_n$ are strict preinitial.

Proof. Suppose $x, y : H \longrightarrow A$. Then:

$$x = x\pi_1\langle h, 1\rangle = x\pi_1(h \times 1)\Delta = y\pi_1(h \times 1)\Delta = y\pi_1\langle h, 1\rangle = y.$$

So H is preinitial. To show that it is strict, suppose that $q : Q \longrightarrow H$; then the above reasoning applies to qh, making Q preinitial. □

Notice that if H is strict preinitial then, as $H \times Y \xrightarrow{\pi_0} H$, it follows that $H \times Y$ is always strict preinitial. This means in a Cartesian category once one has one strict preinitial there must, for each object, be a preinitial with a map to that object. This does not imply there will be an initial object, but it does force that an initial object, if it exists, is automatically strict.

We are now ready to prove that the elements t and f in a Turing category are disjoint.

Lemma 3.5 *In any Turing category, if $h : H \longrightarrow 1$ is a total map which equalizes* t *and* f *(as chosen above) then H is a strict initial object in the total map category. Thus the elements* t *and* f *are disjoint.*

Proof. Suppose $f, g : A \longrightarrow B$ are total maps. Without loss of generality we may assume $A = B = U$. Note that the diagram

$$\begin{tikzcd}
1 \times U \arrow[r, "\pi"] \arrow[d, shift right=1.5, "\mathsf{t}\times\langle f,g\rangle"'] \arrow[d, shift left=1.5, "\mathsf{f}\times\langle f,g\rangle"] & U \arrow[d, shift right=1.5, "f"'] \arrow[d, shift left=1.5, "g"] \\
U \times U \times U \arrow[r, "\bullet(\bullet\times 1)"'] & U
\end{tikzcd}$$

commutes serially. But then precomposing with $h \times 1 : H \times U \longrightarrow 1 \times U$ shows that h satisfies the conditions of the lemma above and so is a strict preinitial object. It remains only to show that there is a total map $H \longrightarrow B$ for each object B. Each B is a retract of the Turing object and so the composite $\iota_B \rho_B : U \longrightarrow U$ is a split idempotent. As with any map in a Turing category, there is a code $k_B : 1 \longrightarrow U$ such that $\bullet\langle k_B, 1\rangle = \iota_B \rho_B$, as in

$$\begin{tikzcd}
U \arrow[r, "\rho_B"] \arrow[rr, bend right, "\bullet\langle k_B,1\rangle"'] & B \arrow[r, "\iota_B"] & U \, .
\end{tikzcd}$$

While ρ_B may be partial, we argue that $\rho_B h$ is total:

$$\bullet\langle k_B, 1\rangle h = \bullet\langle k_B h, h\rangle = \bullet\langle \mathsf{k}h, h\rangle = \bullet(\mathsf{k} \times 1)\Delta h = \pi_0 \Delta h = h$$

so that $\iota_B \rho_B h$ is total, whence $\rho_B h$ is total. □

It may be useful at this stage to provide an example of a total map category which has a universal object and yet *cannot* be the total maps of a Turing category. The simplest example, which also shows that such a category cannot consist entirely of preinitial objects, is when the category is a meet-semilattice. Then the only element is the identity on the top and this must be disjoint from itself, forcing the top to also be the bottom thereby collapsing the lattice.

3.3 An abstract retract structure

Recall that each object A in a Turing category comes equipped with $(\iota_A, \rho_A) : A \prec U$ exhibiting it as a retract of the (chosen) Turing object. There may be many choices for this family of retractions although we shall assume that $(\iota_U, \rho_U) = (1_U, 1_U)$. Below we describe how this structure introduces an analogous structure on the total map category.

First, consider a span between representable objects $y(X)$ and $y(X')$ in $\mathsf{Set}^{\mathbb{X}^{\mathrm{op}}}$:

$$y(X) \xleftarrow{\;s\;} A \xrightarrow{\;t\;} y(X').$$

The apex A need not be representable, but we may consider the family of spans in $\mathbb{X}$ arising by covering A by representables as follows:

$$\mathcal{S}(s,t) = \{X \xleftarrow{sh} B \xrightarrow{th} y(X') | h : \mathcal{Y}(B) \longrightarrow A\}.$$

This family is then clearly closed under precomposition with maps in $\mathbb{X}$. In fact, we may regard $\mathcal{S}(s,t)$ as a category whose objects are the spans in $\mathbb{X}$ factoring through (s,t), and whose morphisms are morphisms of spans.

Conversely, consider a family $\mathcal{R}$ of spans in $\mathbb{X}$ from X to X' which is closed under precomposition. For the present purposes, we shall call such a family an **abstract span**. Regarding $\mathcal{R}$ as a category, we may consider the functor

$$\mathcal{R} \longrightarrow \mathrm{Span}(y(X), y(X')) \xrightarrow{\mathrm{Apex}} \mathsf{Set}^{\mathbb{X}^{\mathrm{op}}}$$

which sends a span $X \xleftarrow{p} B \xrightarrow{q} X'$ to the $y(B)$. The colimit of this diagram gives a span $y(X) \longleftarrow \hat{\mathcal{R}} \longrightarrow y(X')$ in $\mathsf{Set}^{\mathbb{X}^{\mathrm{op}}}$. The proof of the following is now routine:

Lemma 3.6 *The assignments $(s,t) \mapsto \mathcal{S}(s,t)$ and $\mathcal{R} \mapsto \hat{\mathcal{R}}$ are mutually inverse (up to isomorphism of spans).*

Next, we wish to characterize when the left leg of a span between representables is monic (so that it is a partial map). Call an abstract span $\mathcal{R}$ **deterministic** when $(p,q), (p,q') \in \mathcal{R}$ implies $q = q'$.

Lemma 3.7 *Given an abstract span $\mathcal{R}$ corresponding to a span $y(X) \xleftarrow{s} \hat{\mathcal{R}} \xrightarrow{t} y(X')$ between representables, s is monic if and only if the spans in $\mathcal{R}$ are deterministic.*

Of course, in this situation $\hat{\mathcal{R}}$ may be regarded as a sieve on X: it is precisely the sieve

$$\hat{\mathcal{R}}(Z) = \{h : Z \longrightarrow X | (h,k) \in \mathcal{R} \text{ for some } k : Z \longrightarrow X'\}.$$

So far, we have described partial maps between representables in terms of abstract spans. Next, we wish to characterize when such a partial map is in fact a retraction of a morphism $\iota_A : A \longrightarrow U$. So suppose that in $\mathsf{Set}^{\mathbb{X}^{\mathrm{op}}}$, we have a span $y(X) \supseteq S$ $\xrightarrow{t} y(X')$ which is a retraction of a map $\iota : X' \longrightarrow X$. This of course means that $\iota \in S$ and $t(\iota) = 1_{X'}$. In terms of the corresponding abstract span $\mathcal{S}$, this means that $(\iota, 1) \in \mathcal{S}$ and that for each $(h,f) \in \mathcal{S}$ and each diagram

$$\begin{tikzcd}
X' \arrow[d, "\iota"'] & P \arrow[l, "k"'] \arrow[d, "z"] & \\
X & M \arrow[l, "h"] \arrow[r, "f"'] & X'
\end{tikzcd}$$

with the square commuting, we have $fz = k$. We shall refer to this condition by saying that the abstract span $\mathcal{S}$ is **retracting** on ι. To summarize:

Lemma 3.8 *Given a morphism $\iota : X' \longrightarrow X$ in $\mathbb{X}$ and a span $y(X) \supseteq S \xrightarrow{t} y(X')$, the span is a retraction of ι in $\mathsf{Par}(\mathsf{Set}^{\mathbb{X}^{\mathrm{op}}})$ if and only if the corresponding abstract span contains $(\iota, 1)$ and is retracting on ι.*

We are now ready for the main definition of this section:

Definition 3.9 An **abstract retract structure** on a Cartesian category with a universal object U consists of a choice of embeddings $\iota_A : A \longrightarrow U$ (with $\iota_U = 1_U$) and, in addition, for each object A an **abstract retraction** of ι_A, meaning a family of spans $\mathcal{R}_A$ from U to A satisfying:

[RS.1] Each $\mathcal{R}_A$ is closed under precomposition;

[RS.2] Each $\mathcal{R}_A$ is deterministic;

[RS.3] $U \xleftarrow{\iota_A} A \xrightarrow{1_A} A \in \mathcal{R}_A$;

[RS.4] Each $\mathcal{R}_A$ is retracting on ι_A.

We shall indicate an abstract retraction pair by $(\iota_A, \mathcal{R}_A) : A \prec U$.

Here are some examples abstract retractions for a fixed morphism $\iota_A : A \longrightarrow U$.

Examples 3.1 *(1) If $\mathcal{R}$ is an abstract retraction, then $\mathcal{R}_U$ is always just the family $U \xleftarrow{x} X \xrightarrow{x} U$. Indeed suppose $U \xleftarrow{h} Y \xrightarrow{f} U \in \mathcal{R}_U$ then by* **[RS.4]** *$h1_Y = \iota_U h$, so $f = f1_X = h$. Similarly, if ι_A is an isomorphism then $\mathcal{R}_A$ is always just the family $U \xleftarrow{\iota_A x} X \xrightarrow{x} A$.*

(2) The smallest retraction structure for each A is the one generated by the span $(\iota_A, 1_A)$. Each span in the abstract retraction is of the form $U \xleftarrow{\iota_A x} X \xrightarrow{x} A$; this means the set is deterministic and retracting on ι_A.

(3) Suppose that ι_A has a (total) retraction ρ_A. Then we may generate a retraction structure $\{(x, x\rho_A) | x : X \longrightarrow U\}$. Note that when $x = \iota_A$ we obtain the span $U \xleftarrow{\iota_A} A \xrightarrow{1_A} A$. As all the spans have their right leg determined by the left leg post-composed with ρ_A the set is deterministic.

(4) Finally, consider a totalizing extension $T : \mathbb{X} \longrightarrow \mathbb{Y}$, in which U is a universal object. If, in $\mathbb{Y}$, each object A is equipped with a retraction $(\iota_A, \rho_A) : A \prec U$, then we may consider the abstract span in $\mathbb{X}$ given by $\mathcal{R}_A = \{U \xleftarrow{h} A \xrightarrow{f} A | \rho_A h = f\}$. This is readily seen to be an abstract retraction on ι_A.

The last example shows in particular that whenever $\mathbb{T}$ is a Turing category with total map subcategory $\mathbb{X}$, there is an induced abstract retract structure on $\mathbb{X}$. For the record:

Lemma 3.10 *Every Turing category induces, for any Turing object U and chosen retractions $(\iota_A, \rho_A) : A \prec U$, an abstract retract structure on its total map category.*

3.4 Codes

A Turing object U in $\mathbb{T}$ is not only universal, but it also acts as a weak exponential for every pair of objects in the category. In particular, given $f : A \longrightarrow B$, there exists a code $c[f] : 1 \longrightarrow U$ making

$$\begin{tikzcd}
U \times U \arrow[rr, "\bullet"] & & U \\
U \arrow[u, "{\langle c[f],1\rangle}"] \arrow[r, "\rho_A"'] & A \arrow[r, "f"'] & B \arrow[u, "\iota_B"']
\end{tikzcd}$$

commute. Here, ρ_A and ι_B are part of the chosen retractions $A \prec U$ and $B \prec U$, and the morphism $U \times U \xrightarrow{\bullet} U$ is the universal application map. The code $c[f]$ is required to be total, but not unique. Choosing a code $c[f]$ for each f amounts to giving a family of mappings

$$c = c_{A,B} : \mathbb{T}(A, B) \longrightarrow \mathbb{T}(1, U). \tag{1}$$

Since the inclusion ι_B is total, it in fact suffices to specify codes only for the case $B = U$, that is, for maps $A \longrightarrow U$.

We will now translate this existence of codes to structure on the total maps. Since codes are formulated in terms of the retractions ρ_A, this involves dealing with the abstract retract structure on the total maps. We may cut down the mapping (1) to the total maps to obtain a mapping $c = c_A : \mathbb{X}(A, U) \longrightarrow \mathbb{X}(1, U)$, where $\mathbb{X} = \mathsf{Total}(\mathbb{T})$.

To motivate the coming definition, first we need a bit of notation. Given a span $U \xleftarrow{h} X \xrightarrow{k} A$ we denote by $Z^*(h, k)$ the span $U \xleftarrow{h\pi_X} Z \times X \xrightarrow{k\pi_X} A$, obtained by precomposing each leg with the projection $Z \times X \longrightarrow X$. When $\mathcal{R}$ is a family of spans from U to A we will write $Z^*\mathcal{R} = \{(Z^*(h, k)|(h, k) \in \mathcal{R}\}$. Similarly, given $f : A \longrightarrow U$, we write $f_*\mathcal{R} = \{(h, fk)|(h, k) \in \mathcal{R}\}$.

Now consider two maps $f : A \longrightarrow U$ and $g : B \longrightarrow U$ with codes $c[f], c[g]$, and suppose we have an object Z for which the unique total map $Z \xrightarrow{!} 1$ equalizes $c[f]$ and $c[g]$. Then as per example 3.1 (4), the a typical span in $\mathcal{R}_\mathcal{A}$ is of the form $U \xleftarrow{h} X \xrightarrow{k} A$ satisfying $k = \rho_A h$. Given such (h, k), we may then consider the span

$$\begin{tikzcd}
 & Z \times X \arrow[d, "\pi_X"'] & & \\
U & X \arrow[l, "h"] \arrow[r, "k"'] & A \arrow[r, "f"'] & U
\end{tikzcd}$$

In the above notation, this is $Z^*(fh, k) \in Z^*f_*\mathcal{R}$. Under the given condition on Z, the composite $fk\pi_X$ factors as $g \circ (\rho_B h\pi_X)$, as shown by the following calculation:

$$\begin{aligned}
fk\pi_X &= f\rho_A h\pi_X = f\rho_A\pi_U(z \times h) = \bullet(c_A[f] \times U)(z \times h) \\
&= \bullet(c_A[f]z \times h) = \bullet(c_B[g]z \times h) = \bullet(c_B[g] \times U)(z \times h) \\
&= g\rho_B\pi_U(z \times h) = g\rho_B h\pi_X
\end{aligned}$$

Note that the statement that $fk\pi_X$ factors as indicated implies that $Z^*(h, kf) \in Z^*f_*\mathcal{R}_A$ can be regarded as a span of the form $Z^*(h, gq)$ for $(h, q) \in \mathcal{R}_B$. By symmetry, this implies that $Z^*f_*\mathcal{R}_A = Z^*g_*\mathcal{R}_B$. Thus, informally speaking, when restricting the abstract retractions to Z, the operations of composing with f and with g become equal.

This leads to the following:

Definition 3.11 A Cartesian category $\mathbb{X}$ with universal object U and retract structure $(\iota_A, \mathcal{R}_A) : A \prec U$ has **codes** when there are maps $c_A : \mathbb{X}(A,U) \longrightarrow \mathbb{X}(1,U)$ such that whenever $Z \xrightarrow{!} 1$ satisfies $c[f]z = c[g]z$ for $f : A \longrightarrow U, g : B \longrightarrow U$ then $Z^* f_* \mathcal{R}_A = Z^* g_* \mathcal{R}_B$.

The preceding discussion shows:

Lemma 3.12 *The total map subcategory of a Turing category always has codes.*

4 Building a Turing category from total maps

We shall now construct a PCA in $\widetilde{\mathbb{X}}$ which has all the total maps computable. We use the notion of a "stack object", which is an object satisfying two domain equations which allow for the implementation of a rewriting system. Using the fact that $\widetilde{\mathbb{X}}$ admits a trace, we may then define a universal application morphism which will equip the stack object with a PCA structure. From now on, assume $\mathbb{X}$ is a Cartesian category satisfying the conditions explained in the previous section.

Before we start, however, it is convenient to modify $\widetilde{\mathbb{X}}$ a little. Recall that the Yoneda embedding does not preserve initial objects (if these happen to be present), but that there is always a subcanonical topology J on the category $\mathbb{X}$ which corrects this. (Simply take the smallest topology containing the empty cover on each initial object of $\mathbb{X}$.) This gives an embedding $\mathbb{X} \xrightarrow{y} Sh(\mathbb{X}, J)$ preserving any existing initial object. For the rest of the paper, we will let $\widetilde{\mathbb{X}}$ denote the full subcategory of the partial map category on $Sh(\mathbb{X}, J)$ on the representables. It will also be useful to assume that we have access to the splittings of restriction idempotents and to finite coproducts, thus it will be more convenient to work in $\mathsf{Par}(Sh(\mathbb{X}, J))$, and to restricting to $\widetilde{\mathbb{X}}$ by observing that the structures we build are always on representable objects.

Another important property of a partial map category of a (pre)sheaf category (and hence also of $\widetilde{\mathbb{X}}$) is that it is traced on the coproduct: given $f : X \longrightarrow X + Y$ there is a map $f^\dagger : X \longrightarrow Y$ which is the joint of all the finite partial iterates: $f^\dagger = \bigvee_{i \in \mathbb{N}} f^{(i)}$, where

$$\begin{aligned} f^{(1)} &= \sigma_1^{(-1)} f : X \longrightarrow Y \\ f^{(2)} &= \sigma_1^{(-1)} f \sigma_0^{(-1)} f : X \longrightarrow Y \\ &\dots \\ f^{(n+1)} &= \sigma_1^{(-1)} f (\sigma_0^{(-1)} f)^n : X \longrightarrow Y \end{aligned}$$

where $\sigma_i^{(-1)}$ is the partial inverse of the i^{th} coproduct injection. Intuitively, $f^\dagger$ takes an input $x \in X$, and computes the iterates $f^i(x)$ for as long as the output lies in X. Once the output lies in Y, this is the value of $f^\dagger(x)$.

4.1 Stack objects

An object A in a distributive restriction category is said to be a **stack object** in case there are maps

$$\mathsf{put} : 1 + A \times A \longrightarrow A \qquad \mathsf{get} : A \longrightarrow 1 + A \times A \qquad \text{with } \mathsf{put}\ \mathsf{get} = 1_{1+A\times A}.$$

Thus, a stack object can be indicated by $(\mathsf{put}, \mathsf{get}) : 1 + A \times A \prec A$.

Lemma 4.1 *In any distributive restriction category the following are equivalent conditions for an object:*

(i) $1 + 1 \prec A$ *and* $A \times A \prec A$*;*

(ii) $1 + A \times A \prec A$ *(it is a stack object)*

(iii) *For any polynomial functor P, we have* $P(A) = n_0.1 + n_1.A + ... + n_r.A^r \prec A$

The point is that in $\widetilde{\mathbb{X}}$ we know the first condition is satisfied by the universal object U: indeed, $U \times U \prec U$ because $U \times U$ is an object of $\mathbb{X}$, and $1 + 1 \prec U$ because U has two disjoint elements $\mathsf{t}, \mathsf{f} : 1 \longrightarrow U$, giving $[\mathsf{t}, \mathsf{f}] : 1+1 \longrightarrow U$. A partial retraction may be defined by letting S be the sieve on U generated by $\{\mathsf{t}, \mathsf{f}\}$, and by defining a natural transformation $S \xrightarrow{\sigma} 1 + 1$ by $\sigma_X(m) = in_L(*)$ whenever m factors through t, and $in_R(*)$ otherwise. Note that this is well-defined precisely because t, f are disjoint, so that m factors through both t and f only when its domain is 0; but then $(1 + 1)(0)$ is a singleton since $1 + 1$ is assumed to be a sheaf.

4.2 A stack machine for partial combinatory algebras

We shall now use this trace to define a partial application $A \times A \xrightarrow{\bullet} A$ making A into a PCA. To this end, we will define a partial map $\mathsf{step} : A \times A \times A \longrightarrow A \times A \times A + A$ whose trace then is of the desired type. More precisely, we will define $x \bullet y := \mathsf{step}^\dagger(x, y, [])$. The intuition that should be kept in mind is that the map step executes one step of a program/rewrite system, and that its trace runs the entire computation/rewriting sequence. The components of $A \times A \times A$ will be regarded as a code stack, a value, and a dump stack, respectively. Only when the code is end and the dump stack is empty does the computation halt.

To define step, we use the domain equations for A to fix three different representations of our stack object:

$$\mathsf{end}, \mathsf{k}, \mathsf{s} : 1 \longrightarrow A \quad \mathsf{k}_0, \mathsf{s}_0 : A \longrightarrow A \quad \mathsf{s}_1 : A \times A \longrightarrow A \quad \mathsf{nam} : A \longrightarrow A$$

$$(\langle \mathsf{end}|\mathsf{k}|\mathsf{s}|\mathsf{k}_0|\mathsf{s}_0|\mathsf{s}_1|\mathsf{nam}\rangle, g_0) : 1 + 1 + 1 + A + A + A \times A + A \prec A$$

$$\mathsf{c}_0 : A \longrightarrow A \quad \mathsf{c}_1 : A \times A \longrightarrow A$$

$$(\langle \mathsf{c}_0|\mathsf{c}_1\rangle, g_1) : A + A \times A \prec A$$

$$\mathsf{nil} : 1 \longrightarrow A \quad \mathsf{cons} : A \times A \longrightarrow A$$

$$(\langle \mathsf{nil}|\mathsf{cons}\rangle, g_2) : 1 + A \times A \prec A$$

and now we define step by the following case distinction:

Code	Value	Stack	Code	Value	Stack
end	x	nil	exit with x		
k	x	S	end	$\mathsf{k}_0(x)$	S
$\mathsf{k}_0(x)$	y	S	end	x	S
s	x	S	end	$\mathsf{s}_0(x)$	S
$\mathsf{s}_0(x)$	y	S	end	$\mathsf{s}_1(x,y)$	S
$\mathsf{s}_1(x,y)$	z	S	x	z	$\mathsf{cons}(\mathsf{c}_0(y,z),S)$
$\mathsf{nam}(c_A[f])$	z	S	end	$f(\rho_A(z))$	S
end	v	$\mathsf{cons}(\mathsf{c}_0(y,z),S)$	y	z	$\mathsf{cons}(c_1(v),S)$
end	v'	$\mathsf{cons}(\mathsf{c}_1(v),S)$	v	v'	S

The one aspect which needs explanation concerns how we use the names of maps. The aim is to implement them as the composite of a retraction to the idempotent and the map itself. Without these names, the stack machine can be thought of as an implementation of the usual rewriting system on combinatory logic; adding the names of maps from $\mathbb{X}$ together with the given rule essentially amounts to adding rewrites $c_A[f] \longrightarrow f(a)$ to the system.

The step partial function is the join of all its individual components. For step to be well-defined all these components must be compatible, in the sense that they agree on overlaps of domains. Those which are in separate components of the stack object by design are disjoint and so compatible. However, the names of maps are all in the same component and, thus, we must establish compatibility of the implementation of names.

Lemma 4.2 step *is a well-defined partial map in* $\widetilde{\mathbb{X}}$*.*

Proof. A typical span corresponding to a partial map $U \times U \times U \longrightarrow U \times U \times U$ in the component of step which is defined on a tuple of the form $(\mathsf{nam}(c_A[f]), h, S)$ (for $f : A \longrightarrow U$, $h : X \longrightarrow U$) looks like

$$\begin{tikzcd} U \times U \times U & X \arrow[l, "\langle \mathsf{nam}(c_A[f])!_X, h, S\rangle"'] \arrow[r, "\langle \mathsf{end}, fk, S\rangle"] & U \times U \times U \end{tikzcd}$$

Here, $(h,k) \in \mathcal{R}(A)$. Given another such span

$$\begin{tikzcd} U \times U \times U & X' \arrow[l, "\langle \mathsf{nam}(c_B[g])!_{X'}, h', S'\rangle"'] \arrow[r, "\langle \mathsf{end}, gk', S'\rangle"] & U \times U \times U \end{tikzcd}$$

with $g : B \longrightarrow U$ and $(h', k') \in \mathcal{R}_B$, we must show that these two spans are compatible. This is done by considering generalized elements of $z : Z \longrightarrow X$ and $z' : Z \longrightarrow X'$; we must verify that if $\langle \mathsf{nam}(c_A[f]!_X, h, S\rangle z = \langle \mathsf{nam}(c_B[g])!_{X'}, h', S'\rangle z'$, then also $\langle \mathsf{end}!_X, fk, S\rangle z = \langle \mathsf{end}!_{X'}, gk', S'\rangle z'$ It is clear that this holds for the last component. Thus we must show that if $c_A[f]!_X z = c_B[g]!_{X'} z'$ and $hz = h'z'$ then $fkz = gk'z'$.

The first condition means $c_A[f]!_Z = c_B[g]!_Z$ which, by the requirement on codes, implies (hz, fkz) is in $\mathcal{R}_B g$. This implies $(hz, fkz) = (hz, gv)$. But $hz = h'z'$, as $\mathcal{R}_B$ is deterministic, now gives $v = k'z'$ and so $fkz = gv = gk'z'$ as required. □

We are now ready for the main result:

Theorem 4.3 *Given a Cartesian category $\mathbb{X}$ with a universal object, a pair of disjoint elements and a retract structure with codes then the above definition of $\bullet$ in $\widetilde{\mathbb{X}}$ gives a partial combinatory algebra; the subcategory of $\widetilde{\mathbb{X}}$ on the computable maps is then a Turing category whose total maps are exactly the maps of $\mathbb{X}$.*

The proof consists of a verification that the combinators k and s indeed perform as required. This is relegated to the appendix.

5 Applications

For many of the obvious applications we have proven much more than is actually needed. Here are two corollaries of our main theorem:

Proposition 5.1 *Given a Cartesian category $\mathbb{X}$ in which*

- *Every object has at least one element;*
- *There is a universal object U;*
- *There is a monic (set) map $c : \coprod_{A \in \mathbb{X}} \mathbb{X}(A, U) \longrightarrow \mathbb{X}(1, U)$;*
- *There is a faithful product preserving functor into $U : \mathbb{X} \longrightarrow \mathsf{Set}$.*

Then $\mathbb{X}$ occurs as the total maps of a Turing category.

These conditions include the PTIME maps between binary natural numbers. In this example one actually can obtain a linear time encoding of pairs of binary number into the binary numbers. This may be achieved by interleaving the bit strings while adding an extra bit on each component to indicate termination. This means we have (much more than):

Corollary 5.2 *The PTIME maps between binary numbers occur as the total maps of a Turing category.*

We also have the following:

Proposition 5.3 *Any countable Cartesian category with a universal object U and a pair of disjoint elements is the total maps of a Turing category.*

Here we observe that once one has two distinct elements the fact that one has a stack object allows one to easily obtain countably many distinct and disjoint elements. This means that there is a monic assignment of maps to elements and one can use the smallest abstract retractions.

6 Conclusion

To unify the abstract notion of computability embodied in Turing categories with the study of feasible computation (e.g. LINEAR, LOGSPACE, PTIME, ...) minimally one must know whether these functional complexity classes can form the total maps of a Turing category. In [3] it was shown that both LOGSPACE and PTIME functions had a natural description as the total maps of Turing categories. However, in order to obtain those results, it was necessary to use the well-known facts from complexity theory that transducers and Turing machines can universally simulate themselves with an overhead which can be accommodated within (respectively) LOGSPACE and PTIME. This meant the argument that these complexity classes could be modelled by Turing Categories relied heavily on the details of the machine models and the way in which resources were measured. In particular, as there is no widely accepted machine model which can simulate itself with a *linear* time overhead, the linear time maps, LINEAR, could not be so readily included in this approach.

The power of the results outlined in this paper is, precisely, that they are abstract. That is that they do not depend on the peculiarities of machine models or on the way resources are counted. In particular, Proposition 5.3 applies immediately to the linear time maps because the encoding and decoding of pairs of bit strings can all be done in linear time.

By describing necessary and sufficient conditions for a Cartesian category to be the total maps of a Turing category we have delimited the applicability of Turing categories to feasible computation. Perhaps, somewhat counter-intuitively, the results indicate that Turing categories *are* applicable beyond the traditional confines of computability theory into feasible computation and the domain of complexity theory. In this regard Turing categories, therefore, provide – by more than mere analogy – a medium for the transfer of ideas between computability and complexity theory and, thus, for a potential economy of presentation which may be beneficial to the further development of the subject.

References

[1] J.R.B. Cockett, *Categories and computability.* Lecture notes available at http://pages.cpsc.ucalgary.ca/ robin.

[2] J.R.B. Cockett and P.J.W. Hofstra, *Introduction to Turing categories* Annals of Pure and Applied Logic, Volume 156, Issues 2-3, December 2008, Pages 183-209.

[3] J.R.B. Cockett, J. Diaz-Boïls, J. Gallagher and Pavel Hrubeš *Timed Sets, Functional Complexity, and Computability*. Electronic Notes in Theoretical Computer Science, Volume 286, September 2012, Pages 117137.

[4] J. R. B. Cockett and S. Lack, *Restriction categories I: categories of partial maps.* Theoretical Computer Science 270(1-2): 223-259, 2002.

[5] S. Cook and P. Nguyen, *Logical Foundations of Proof Complexity.* In: Perspectives in Logic, Cambridge University Press, 2010.

[6] R. Di Paola and A. Heller, *Dominical categories: recursion theory without elements.* Journal of Symbolic Logic, Volume 56,1987.

Appendix: Iterating step gives a PCA

We first verify that $\mathsf{k} \bullet x \bullet y = x$ (where $\bullet$, as usual, associates to the left). Here $\mathsf{step}(\mathsf{k}, x, []) = \sigma_0(\mathsf{end}, \mathsf{k}_(x), [])$ has $\mathsf{step}(\mathsf{end}, \mathsf{k}_0(x), []) = \sigma_1(\mathsf{k}_0(x))$ so that $(\mathsf{k} \bullet x) = \mathsf{k}_0(x)$. But now

$$\begin{aligned}
\mathsf{k}_0(x) \bullet y &= \mathsf{step}^{\dagger}(\mathsf{k}_0(x), y, []) \\
&= \left\{ \begin{array}{ll} \sigma_0(c,v,d) & \mapsto \mathsf{step}^{\dagger}(c,v,d) \\ \sigma_1(x) & \mapsto x \end{array} \right\} \mathsf{step}(\mathsf{k}_0(x), y, []) \\
&= \left\{ \begin{array}{ll} \sigma_0(c,v,d) & \mapsto \mathsf{step}^{\dagger}(c,v,d) \\ \sigma_1(x) & \mapsto x \end{array} \right\} \sigma_1(x) \\
&= x
\end{aligned}$$

which verifies that $(\mathsf{k} \bullet x) \bullet y = x$.

Next, we need $\mathsf{s} \bullet x \bullet y$ to be as defined as x and y. But clearly $\mathsf{s} \bullet x = \mathsf{s}_0(x)$ and $\mathsf{s}_0(x) \bullet y = \mathsf{s}_1(x, y)$ so, as s_1 is total this requirement of the combinator is met.

Next, we calculate

$$\begin{aligned}
((\mathsf{s} \bullet x) \bullet y) \bullet z &= \mathsf{s}_1(x,y) \bullet z \\
&= \mathsf{step}^{\dagger}(\mathsf{s}_1(x,y), z, []) \\
&= \left\{ \begin{array}{ll} \sigma_0(c,v,d) & \mapsto \mathsf{step}^{\dagger}(c,v,d) \\ \sigma_1(x) & \mapsto x \end{array} \right\} \mathsf{step}(\mathsf{s}_1(x,y), z, []) \\
&= \mathsf{step}^{\dagger}(x, z, \mathsf{cons}(\mathsf{c}_1(y,z), [])) \\
&= \mathsf{step}^{\dagger}(\mathsf{end}, x \bullet z, \mathsf{cons}(\mathsf{c}_1(y,z), [])) \\
&= \mathsf{step}^{\dagger}(y, z, \mathsf{cons}(\mathsf{c}_0(x \bullet z), [])) \\
&= \mathsf{step}^{\dagger}(\mathsf{end}, y \bullet z, \mathsf{cons}(\mathsf{c}_0(x \bullet z), [])) \\
&= \mathsf{step}^{\dagger}(x \bullet z, y \bullet z, []) \\
&= (x \bullet z) \bullet (y \bullet z)
\end{aligned}$$

Here we use repeatedly the identity:

$$\mathsf{step}^{\dagger}(x, y, S) = \mathsf{step}^{\dagger}(\mathsf{end}, \mathsf{step}^{\dagger}(x, y, []), S) = \mathsf{step}^{\dagger}(\mathsf{end}, x \bullet y, S)$$

which is true by virtue of the fact that the trace is defined inductively. More precisely, we have

$$\mathsf{step}^{(n)}(x, y, s) = \bigsqcup_{i+j=n+1} \mathsf{step}^{(i)}(\mathsf{end}, \mathsf{step}^{(j)}(x, y, []), s).$$

The left hand expression is empty unless the iteration terminates in that number of steps. If there is a (first) stage j at which the left hand iteration returns the stack to its original state and the first coordinate is end then $j+1$ can be used to terminate the inner loop on the right hand side to bring the two iterations to the same state. Subsequently the result will be the same. If there is no such j both sides will be the empty map.

Abstract Local Reasoning for Concurrent Libraries: Mind the Gap

Philippa Gardner, Azalea Raad, Mark Wheelhouse and Adam Wright[1]

Department of Computing, Imperial College London, UK

Abstract

We study abstract local reasoning for concurrent libraries. There are two main approaches: provide a specification of a library by *abstracting* from concrete reasoning about an implementation; or provide a direct abstract library specification, justified by *refining* to an implementation. Both approaches have a significant gap in their reasoning, due to a mismatch between the *abstract connectivity* of the abstract data structures and the *concrete connectivity* of the concrete heap representations. We demonstrate this gap using structural separation logic (SSL) for specifying a concurrent tree library and concurrent abstract predicates (CAP) for reasoning about a concrete tree implementation. The gap between the abstract and concrete connectivity emerges as a mismatch between the SSL tree predicates and CAP heap predicates. This gap is closed by an *interface function I* which links the abstract and concrete connectivity. In the accompanying technical report, we generalise our SSL reasoning and results to arbitrary concurrent data libraries.

Keywords: Concurrency, Abstraction, Refinement, Separation, Translation, Reasoning

1 Introduction

Local reasoning was first introduced in separation logic to reason about the RAM memory model. Since then, there has been considerable work on combining local reasoning with abstraction. There are two main approaches. One approach starts with concrete reasoning about code that manipulates the standard heap and then builds up layers of *abstraction*: this approach is used by concurrent abstract predicates (CAP) and its variants [1,16,17], and is ideal for reasoning about the concurrent library `java.util.concurrent` where the library functions are built up using implementations. The other approach provides direct abstract specifications of abstract code manipulating abstract models, and then justifies the specification by *refining* it to a correct concrete implementation: this approach is used by context logic [7,9], and is ideal for reasoning about libraries such as POSIX and sequential DOM where the library specification is not grounded on implementation.

[1] Email: `{pg, azalea, mjw03, adw07}@doc.ic.ac.uk`

These current abstraction and refinement approaches have a significant gap in their reasoning. With abstraction, the implementation details leak into the abstraction. For example, consider a CAP predicate $\mathsf{tree}(t)(i,j)(l,u,r)$ for describing tree fragments. The predicate is parameterised by an abstract tree t, with concrete pointers $(i,j)(l,u,r)$ describing the concrete interface used to connect the concrete tree fragments. In this case, the concrete interface consists of the first (i) and last (j) nodes of the tree fragment, and the parent (u), left (l) and right (r) siblings. A different implementation, say one using lists, would require a different concrete interface. Thus, this tree predicate is not abstract enough to reason abstractly about updating tree fragments[2]. The missing piece is an abstract way of splitting and combining tree fragments, and a way of linking it to the concrete interface given by $(i,j)(l,u,r)$. With the refinement approach, we have examples where the specification is truly abstract (e.g. [12,19]), but they are typically justified by a soundness result to an operational semantics rather than an implementation. For example, truly abstract reasoning of a tree module (such as DOM) works with predicates based on connecting tree fragments using contexts and place holders. In contrast, the concrete reasoning about an implementation uses pointers. We need to bridge the gap between the *abstract connectivity* of abstract data structures and the *concrete connectivity* of concrete heap representations.

We introduce *Structural Separation Logic (SSL)* for reasoning about concurrent abstract data libraries [2,19]. SSL is underpinned by a particular general approach to abstract connectivity for structured data. In this paper, we use a simple concurrent tree library to illustrate our ideas. We provide concrete reasoning about a tree implementation of the library using CAP [1]. The gap between the abstract and concrete connectivity emerges as a mismatch between SSL tree predicates and CAP tree predicates. This gap is closed by an *interface function* I which links the abstract and concrete connectivity. In the accompanying technical report [18] and Raad's forthcoming thesis, we generalise SSL and our results to arbitrary structured data libraries. The work presented here does depend on the particular SSL approach to abstract connectivity. Our ideas should, however, apply whenever there is a mismatch between abstract and concrete connectivity.

SSL supports reasoning about fine-grained abstract data fragments stored in *abstract heaps*. Abstract heaps contain cells identified by abstract addresses (e.g. address $\mathbf{x}$) whose values are the disjoint data fragments. These data fragments contain context holes, also given by abstract addresses, which are place holders for the data fragments found at the appropriate abstract cells. For example, the SSL predicate $\mathsf{ATree}\,(t)(\mathbf{x})$ describes a tree cell with abstract address $\mathbf{x}$ containing tree context t. We can split (abstractly allocate) this predicate to obtain the semantically-equivalent assertion $\exists \mathbf{y}.\, \mathsf{ATree}\,(t_1)(\mathbf{x}) * \mathsf{ATree}\,(t_2)(\mathbf{y})$ with $t = t_1[t_2/\mathbf{y}]$. The assertion represents the same underlying tree fragment, just in two disjoint parts. Reasoning about semantically-equivalent assertions is only possible due to

[2] The same issue arises with the well-known predicate $\mathsf{listseg}([1,2,3])(i)(r)$ which describes list fragments with concrete address i, abstract contents $1,2,3$, and concrete right pointer r. This predicate is appropriate for implementations using singly-linked lists, but not for those using doubly-linked lists which require an additional concrete left pointer. The predicate is not abstract enough because the concrete interface is leaking into the predicate. The missing bit is the abstract connectivity of the list fragment.

recent advances in concurrent reasoning given by the Views framework [6].

We provide a natural implementation of our tree library and show that it is correct with respect to our abstract SSL specification, by relating abstract SSL specifications of the library functions with CAP-like specifications of the concrete function implementations. To do this, we must extend CAP predicates with a *interface function* I which relates abstract addresses with concrete pointers. For example, consider the abstract tree predicate $\mathsf{ATree}(t)(\mathbf{x})$ and the corresponding concrete CAP predicate $\mathsf{CTree}^{I}(t)(x)$, where the interface function I relates the abstract address $\mathbf{x}$ with the pointer interface $(i, j)(l, u, r)$. We prove that our implementation is correct with respect to our abstract specification, using a $*$-preserving translation (analogous to *locality-preserving* translation in [9]) *parameterised* by I.

In this paper, we concentrate on refinement. It is trivial to adapt our work to the approach of fictional separation logic (FSL) [13], where translations are incorporated within the Hoare derivation. However, our choice of translation differs fundamentally from FSL. FSL is designed to reason about *sequential* programs using $*$*-breaking* translations (analogous to *locality-breaking* translation in [9]): the proofs of FSL assume *all* possible frames are preserved during the execution of program. While this is a reasonable assumption when reasoning about sequential programs, it is non-trivial to establish its soundness for concurrent programs. One possible way to demonstrate its correctness is to provide linearisability proofs to show that all frames are indeed preserved and that the program behaves atomically. In contrast, our $*$-preserving translation ensures that compatible stable resources at the abstract level $(p * q)$ yield compatible stable resources once translated $(p' * q')$[3]. The correctness of concurrent programs then follows immediately from the disjoint concurrency rule of concurrent separation logic [20].

With abstraction using CAP, we start by concretely reasoning about the heap and then build up levels of abstraction in such a way that $*$ is preserved. As demonstrated, current CAP reasoning leaks implementation details into the abstractions. We believe this can be rectified by extending the abstraction rule to hide the interface functions as well as the predicate interpretations.

We believe that interface functions I have been barely studied in the literature. They do appear in [9] for reasoning about *sequential* libraries using context logic. Context logic is not fine-grained enough to extend to concurrency, since it uses a non-commutative separating application unsuitable for use with the disjoint concurrency rule. SSL reasoning makes this extension possible. SSL is used in [19] for specifying sequential POSIX, with the aim to extend to concurrent POSIX in future. However, soundness was proved by comparison with an abstract operational semantics, so the relationship between abstract and concrete connectivity did not arise. In [15,16,13], there is an emphasis on interpretations which relate abstract and concrete states. However, there is no mention of a mapping I between abstract and concrete interfaces since the connectivity in the examples studied is simple.

[3] Note that $p * q$ does not necessarily suggest physically *disjoint* resources; rather two *compatible* resources that can be composed together providing a *fiction of disjointness*.

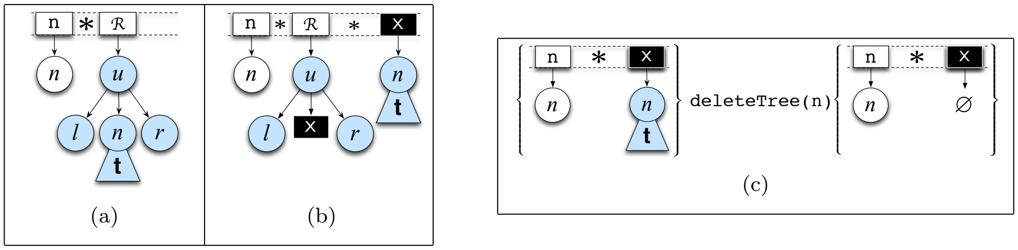


Fig. 1. Abstract (de)allocation in SSL (left); reasoning about `deleteTree(n)` in SSL (right)

2 Structural Separation Logic: Tree Library

Structural separation logic (SSL) is a general program logic for specifying structured data libraries and reasoning locally about client programs which call such libraries. Here, we give the intuition and technical details of SSL using an abstract tree library. We give the general theory of SSL in the accompanying technical report [18]. Further details, including a wide number of examples, can be found in [2].

2.1 Intuition

We give our axiomatic SSL specification of a simple `deleteTree(n)` command. Intuitively, this command removes the entire subtree whose top node identifier corresponds to the value of variable **n**, leaving the rest of the tree unchanged. We formalise this English description using assertions which describe *abstract heaps*.

Abstract heaps store abstract data fragments. For instance, Fig. 1a illustrates an abstract heap describing a variable cell **n** with value n, and a tree cell $\mathcal{R}$ with a complete abstract tree as its value. This tree consists of a subtree $n[t]$ with parent u, and left and right siblings l and r with no children. It abstracts away from how a tree might be concretely represented in a machine.

Intuitively, the `deleteTree(n)` command only affects the subtree identified by n. Abstract heaps enable structured data to be *split* to provide direct access to this subtree by imposing additional instrumentation using *abstract addresses*. Consider the transition from Fig. 1a to 1b. Fig. 1a contains an abstract heap with a complete tree at $\mathcal{R}$. We split this complete tree using *abstract allocation* to obtain the abstract heap in Fig. 1b with subtree $n[t]$ at a fresh, fictional *abstract cell* **x** and an incomplete tree at $\mathcal{R}$ with a *context hole* **x** indicating the position to which the subtree will return. The subtree at n can now be accessed directly. Once the updates have been achieved, the heap can be joined back together using *abstract deallocation*, as in the transition from Fig. 1b to 1a.

The axiomatic specification of `deleteTree(n)` (Fig. 1c) is formalised as:

$$\{\mathsf{var}\,(\mathtt{n}, n) \times \mathsf{ATree}\,(n[isComplete])(\mathbf{x})\}\ \mathtt{deleteTree(n)}\ \{\mathsf{var}\,(\mathtt{n}, n) \times \mathsf{ATree}\,(\varnothing)(\mathbf{x})\}$$

The precondition describes a variable store, in which variable **n** has value n and abstract cell **x** has *complete* subtree with top node n as its value [4]. Since the subtree

[4] Note that the precondition is a pair consisting of a variable assertion and a subtree (heap) assertion. We use the variables-as-resource model [14]. However, in contrast to the assertions of variables-as-resource where heap and variable assertions are combined using $*$, we keep the two components separate. As we

$$\{(\mathsf{var}\,(\mathtt{l}, l) * \mathsf{var}\,(\mathtt{n}, n)) \times (\mathsf{ATree}\,(u[l \otimes n[t] \otimes r])(\mathcal{R}))\}$$

//Abstract allocation (Twice)

$$\{(\mathsf{var}\,(\mathtt{l}, l) * \mathsf{var}\,(\mathtt{n}, n)) \times (\exists \mathbf{x}, \mathbf{y}. \mathsf{ATree}\,(u[\mathbf{y} \otimes \mathbf{x} \otimes r])(\mathcal{R}) * \mathsf{ATree}\,(l)(\mathbf{y}) * \mathsf{ATree}\,(n[t])(\mathbf{x}))\}$$

//Existential elimination and frame rule

$$\{(\mathsf{var}\,(\mathtt{l}, l) * \mathsf{var}\,(\mathtt{n}, n)) \times (\mathsf{ATree}\,(l)(\mathbf{y}) * \mathsf{ATree}\,(n[t])(\mathbf{x}))\}$$

//Disjoint Concurrency rule

$$\begin{array}{c||c}
\{(\mathsf{var}\,(\mathtt{l}, l)) \times (\mathsf{ATree}\,(l)(\mathbf{y}))\} & \{(\mathsf{var}\,(\mathtt{n}, n)) \times (\mathsf{ATree}\,(n[t])(\mathbf{x}))\} \\
\mathtt{deleteTree(l)} & \mathtt{deleteTree(n)} \\
\{(\mathsf{var}\,(\mathtt{l}, l)) \times (\mathsf{ATree}\,(\varnothing)(\mathbf{y}))\} & \{(\mathsf{var}\,(\mathtt{n}, n)) \times (\mathsf{ATree}\,(\varnothing)(\mathbf{x}))\}
\end{array}$$

$$\{(\mathsf{var}\,(\mathtt{l}, l) * \mathsf{var}\,(\mathtt{n}, n)) \times (\mathsf{ATree}\,(\varnothing)(\mathbf{y}) * \mathsf{ATree}\,(\varnothing)(\mathbf{x}))\}$$

//Existential elimination and frame rule

$$\{(\mathsf{var}\,(\mathtt{l}, l) * \mathsf{var}\,(\mathtt{n}, n)) \times (\exists \mathbf{x}, \mathbf{y}.\, \mathsf{ATree}\,(u[\mathbf{y} \otimes \mathbf{x} \otimes r])(\mathcal{R}) * \mathsf{ATree}\,(\varnothing)(\mathbf{y}) * \mathsf{ATree}\,(\varnothing)(\mathbf{x}))\}$$

//Abstract Deallocation (Twice)

$$\{(\mathsf{var}\,(\mathtt{l}, l) * \mathsf{var}\,(\mathtt{n}, n)) \times (\mathsf{ATree}\,(u[r])(\mathcal{R}))\}$$

Fig. 2. Proof derivation of the concurrent program `deleteTree(l) || deleteTree(n)`.

at $\mathbf{x}$ is complete, we own the exclusive right to update its contents. Similarly, the postcondition states that the result of the update is an empty tree at abstract cell $\mathbf{x}$, while variable `n` remains unchanged. Note that the abstract cell $\mathbf{x}$ must be preserved in order to join this tree fragment with the tree it was split from, using abstract deallocation. The footprint of this command is *small* in the sense that it intuitively captures the minimum resources required for safe execution of `deleteTree(n)`.

With this axiomatisation, we can verify that the simple client program `deleteTree(l)` $\|$ `deleteTree(n)` deletes two disjoint subtrees concurrently. Consider the proof derivation in Fig. 2 with program variables `l` and `n`, their values l and n, and an abstract tree predicate $\mathsf{ATree}\,(u[l \otimes n[t] \otimes r])(\mathcal{R})$ denoting a complete tree at $\mathcal{R}$ containing a subtree with top node n, left and right siblings with top nodes l and r, and parent u. The initial precondition describes the complete tree at $\mathcal{R}$. To get at the two subtrees, we split the tree twice using abstract allocation, placing the subtrees at nodes l and n at the freshly allocated cell addresses $\mathbf{y}$ and $\mathbf{x}$, respectively. We apply the standard separation logic rules of existential elimination and frame to temporarily set aside the partial tree at $\mathcal{R}$ as it is not being updated by the program. The resulting state can then be split into two disjoint parts using the disjoint concurrency rule. Both parts are now in the right form to match the precondition of the `deleteTree` axiom. The two updates can happen resulting in the postconditions with empty trees at $\mathbf{y}$ and $\mathbf{x}$ which are joined together by the disjoint concurrency rule. We reintroduce the state set aside through the existential elimination and frame rule, and obtain the complete tree at $\mathcal{R}$ using abstract deallocation twice to remove $\mathbf{y}$ and $\mathbf{x}$.

demonstrate in §4, this is because when translating a library, only heaps are transformed by the translation while program variables are simply preserved.

2.2 Technical Details

We give the technical details of SSL using the tree library $\mathbb{T}$ discussed in §2.1.

Definition 2.1 The set of *abstract atomic tree commands* of $\mathbb{T}$ are defined as:

$$\begin{aligned}\textsc{Atom}_{\mathbb{T}} ::= {} & \texttt{m := getFirst(n)} \mid \texttt{m := getRight(n)} \mid \texttt{m := getUp(n)} \\ & \mid \texttt{m := newNodeAfter(n)} \mid \texttt{deleteTree(n)} \mid \texttt{appendChild(m, n)}\end{aligned}$$

for all program variables $\texttt{n}, \texttt{m} \in \textsc{PVars}$. Our commands are chosen to demonstrate a wide range of structural manipulations. The command `getFirst(n)` returns the identifier of the first child of node `n`, or null if `n` has no children. The `getRight(n)` and `getUp(n)` commands behave analogously with respect to the right sibling and parent of node `n`, respectively. The command `newNodeAfter(n)` creates a new node with a fresh identifier, making it the right sibling of node `n` and returning the fresh identifier. The `deleteTree(n)` command removes the entire subtree identified by `n` from the tree. Finally, the command `appendChild(m,n)` moves the subtree at `n` to be the last child of node `m`. Each of these commands *faults* if any of the nodes given as parameters are not present in the tree. Additionally, the `appendChild(m,n)` command faults if `m` is a descendant of `n`. These commands are intended to be used with any programming language. In this paper, we use the programming language of the Views framework [6] instantiated with $\textsc{Atom}_{\mathbb{T}}$ as the set of atomic commands. We write $\textsc{Prog}_{\mathbb{T}}$ to denote the set of programs written in this language.

We specify the tree commands using abstract heaps with cells whose addresses are either abstract addresses or the tree root address $\mathcal{R}$, and whose values are abstract trees.

Definition 2.2 Given a countably infinite set of abstract addresses $\textsc{AAdd}$ ranged over by $\mathbf{x}, \mathbf{y}, \mathbf{z}$, the set of addresses for the abstract tree heaps is $\textsc{Add}_{\mathbb{T}} \triangleq \{\mathcal{R}\} \uplus \textsc{AAdd}$.

We now define abstract trees, which can either be complete trees or tree fragments with abstract addresses for context holes [5] .

Definition 2.3 Given the set of abstract addresses $\textsc{AAdd}$, the set of *abstract trees* $\textsc{Data}_{\mathbb{T}}$, ranged over by $t, t_1, \cdots t_n$, is defined inductively as:

$$t ::= \varnothing \mid n[t] \mid t_1 \otimes t_2 \mid \mathbf{x}$$

where $n \in \mathbb{N}^+$, $\mathbf{x} \in \textsc{AAdd}$, the sets $\textsc{AAdd}$ and $\mathbb{N}^+$ are disjoint, and no tree contains duplicate node identifiers or abstract addresses. Abstract trees are equal up to the associativity of $\otimes$ with unit $\varnothing$. For brevity, we write n for $n[\varnothing]$. There is an associated *identifiers* function $ids : \textsc{Data}_{\mathbb{T}} \to \wp(\mathbb{N}^+)$ and an associated *addresses* function $addrs : \textsc{Data}_{\mathbb{T}} \to \wp(\textsc{AAdd})$ which respectively, return the sets of node identifiers and abstract addresses present in an abstract tree. The application $t_1 \circ_{\mathbf{x}} t_2$ is standard: it is undefined when $\mathbf{x} \notin addrs(t_1)$ and is otherwise defined as $t_1[t_2/\mathbf{x}]$, denoting the standard substitution of t_2 for $\mathbf{x}$ in t_1.

An abstract tree heap is a mapping from addresses to abstract trees.

[5] Strictly speaking, this grammar describes *forests* or ordered lists of abstract trees. We use the term forest when we wish to emphasise the list with a first and last element.

Definition 2.4 The set of *abstract tree heaps* $H_{\mathbb{T}} : \textsc{Add}_{\mathbb{T}} \overset{\text{fin}}{\rightharpoonup} \textsc{Data}_{\mathbb{T}}$, ranged over by $h, h_1, \cdots, h_n$, is the set of functions from addresses to abstract trees such that for all $h \in H_{\mathbb{T}}$ the following restrictions hold:

$$\begin{array}{l}\forall a_1, a_2 \in \textsc{Add}_{\mathbb{T}}.\ a_1 = a_2 \vee \mathit{addrs}(h(a_1)) \cap \mathit{addrs}(h(a_2)) = \varnothing \\ \forall a_1, a_2 \in \textsc{Add}_{\mathbb{T}}.\ a_1 = a_2\ \vee\ \mathit{ids}(h(a_1)) \cap \mathit{ids}(h(a_2)) = \varnothing \\ \nexists\, \mathbf{x} \in \textsc{AAdd}.\ \mathbf{x}\, D_h^+\, \mathbf{x}\ \wedge\ \forall \mathbf{x} \in \mathrm{dom}(h) \cap \textsc{AAdd}.\ \mathcal{R}\, D_h^+\, \mathbf{x}\end{array}$$

where the descendent relation D for heap h is defined as:

$$a\, D_h\, \mathbf{y} \iff \mathbf{y} \in \mathit{addrs}(h(a))$$

and D_h^+ denotes its transitive closure.

The first two restrictions state that the abstract addresses being used as context holes and the node identifiers are unique across an abstract heap. The last condition states that the abstract addresses join up to produce sensible abstract trees. For instance, $\{\mathbf{x} \to m[\mathbf{y}], \mathbf{y} \to n[\mathbf{x}]\}$ is not an abstract tree heap because of the cycle. An abstract tree heap may be: complete, with no use of abstract addresses; complete, but with the tree split across several heap cells; or *incomplete*, missing some heap cells needed to join some abstract addresses. Incomplete abstract heaps are necessary for local reasoning using the frame rule. In this case, there will be some choice for the missing cells that would render the tree complete.

Given an abstract tree heap h, h^{in} and h^{out} denote the set of abstract cell addresses and context holes, respectively:

$$h^{\mathsf{in}} \triangleq \textsc{AAdd} \cap \mathrm{dom}(h) \qquad h^{\mathsf{out}} \triangleq \textsc{AAdd} \cap \left(\textstyle\bigcup_{t \in \text{co-dom}(h)} \mathit{addrs}(t)\right).$$

By design, abstract tree heaps are similar to standard heaps. The construction of a separation algebra over them is thus straightforward.

Definition 2.5 The *separation algebra of abstract tree heaps* is $\mathcal{A}_{\mathbb{T}} \triangleq (H_{\mathbb{T}}, \bullet_{\mathbb{T}}, \mathbf{0}_{\mathbb{T}})$, where $H_{\mathbb{T}}$ is given in Def. 2.4, $\bullet_{\mathbb{T}}$ is the standard disjoint function union with the proviso that the resulting abstract heap is well-formed as per Def. 2.4; and $\mathbf{0}_{\mathbb{T}}$ is a singleton set consisting of the partial function with an empty domain and co-domain.

Since we use variables as resource [14], we pair variable stores for declaring the values of variables with our abstract tree heaps.

Definition 2.6 The *separation algebra of abstract tree states* $\mathcal{SA}_{\mathbb{T}} \triangleq (\Sigma \times H_{\mathbb{T}}, \bullet_\sigma \times \bullet_{\mathbb{T}}, \mathbf{0}_\sigma \times \mathbf{0}_{\mathbb{T}})$, is the Cartesian product of the separation algebra of variables $(\Sigma, \bullet_\sigma, \mathbf{0}_\sigma)$ [14] and the algebra of abstract tree heaps (Def. 2.5).

To reason about our tree programs, we use the program logic of the Views framework as described in [6]. We instantiate the framework with the separation algebra of abstract tree states (Def. 2.6) as the view monoid and the tree library commands (Def. 2.1) as the atomic commands. What remains is to describe the axioms associated with the tree library commands (Def. 2.7).

Our views are sets of abstract tree states in $\wp(\Sigma \times H_{\mathbb{T}})$. To increase readability of variable sets, we write $\mathsf{var}\,(\mathtt{x}, v)$ for $\{\mathtt{x} \to v\}$ and $p * q$ for $\{\sigma_1 \bullet_\sigma \sigma_2 \mid \sigma_1 \in p \wedge \sigma_2 \in q\}$.

$$\begin{array}{c}\left\{ (\mathsf{var}\,(\mathtt{n}, n) * \mathsf{var}\,(\mathtt{m}, o)) \times (\mathsf{ATree}\,(n[m[\mathbf{y}] \otimes \mathbf{z}])(\mathbf{x})) \right\} \\ \mathtt{m\ :=\ getFirst(n)} \\ \left\{ (\mathsf{var}\,(\mathtt{n}, n) * \mathsf{var}\,(\mathtt{m}, m)) \times (\mathsf{ATree}\,(n[m[\mathbf{y}] \otimes \mathbf{z}])(\mathbf{x})) \right\}\end{array} \qquad \begin{array}{c}\left\{ (\mathsf{var}\,(\mathtt{n}, n) * \mathsf{var}\,(\mathtt{m}, o)) \times (\mathsf{ATree}\,(n[\varnothing])(\mathbf{x})) \right\} \\ \mathtt{m\ :=\ getFirst(n)} \\ \left\{ (\mathsf{var}\,(\mathtt{n}, n) * \mathsf{var}\,(\mathtt{m}, \mathsf{null})) \times (\mathsf{ATree}\,(n[\varnothing])(\mathbf{x})) \right\}\end{array}$$

$$\begin{array}{c}\left\{ (\mathsf{var}\,(\mathtt{n}, n) * \mathsf{var}\,(\mathtt{m}, o)) \times (\mathsf{ATree}\,(n[\mathbf{y}] \otimes m[\mathbf{z}])(\mathbf{x})) \right\} \\ \mathtt{m\ :=\ getRight(n)} \\ \left\{ (\mathsf{var}\,(\mathtt{n}, n) * \mathsf{var}\,(\mathtt{m}, m)) \times (\mathsf{ATree}\,(n[\mathbf{y}] \otimes m[\mathbf{z}])(\mathbf{x})) \right\}\end{array} \qquad \begin{array}{c}\left\{ (\mathsf{var}\,(\mathtt{n}, n) * \mathsf{var}\,(\mathtt{m}, o)) \times (\mathsf{ATree}\,(u[\mathbf{y} \otimes n[\mathbf{z}]])(\mathbf{x})) \right\} \\ \mathtt{m\ :=\ getRight(n)} \\ \left\{ (\mathsf{var}\,(\mathtt{n}, n) * \mathsf{var}\,(\mathtt{m}, \mathsf{null})) \times (\mathsf{ATree}\,(u[\mathbf{y} \otimes n[\mathbf{z}]])(\mathbf{x})) \right\}\end{array}$$

$$\begin{array}{c}\left\{ (\mathsf{var}\,(\mathtt{n}, n) * \mathsf{var}\,(\mathtt{m}, o)) \times (\mathsf{ATree}\,(m[\mathbf{y} \otimes n[\mathbf{z}] \otimes \mathbf{w}])(\mathbf{x})) \right\} \\ \mathtt{m\ :=\ getUp(n)} \\ \left\{ (\mathsf{var}\,(\mathtt{n}, n) * \mathsf{var}\,(\mathtt{m}, m)) \times (\mathsf{ATree}\,(m[\mathbf{y} \otimes n[\mathbf{z}] \otimes \mathbf{w}])(\mathbf{x})) \right\}\end{array} \qquad \begin{array}{c}\left\{ (\mathsf{var}\,(\mathtt{n}, n) * \mathsf{var}\,(\mathtt{m}, o)) \times (\mathsf{ATree}\,(\mathbf{x} \otimes n[\mathbf{y}] \otimes \mathbf{z})(\mathcal{R})) \right\} \\ \mathtt{m\ :=\ getUp(n)} \\ \left\{ (\mathsf{var}\,(\mathtt{n}, n) * \mathsf{var}\,(\mathtt{m}, \mathsf{null})) \times (\mathsf{ATree}\,(\mathbf{x} \otimes n[\mathbf{y}] \otimes \mathbf{z})(\mathcal{R})) \right\}\end{array}$$

$$\begin{array}{c}\{ (\mathsf{var}\,(\mathtt{n}, n) * \mathsf{var}\,(\mathtt{m}, o)) \times (\mathsf{ATree}\,(n[\mathbf{y}])(\mathbf{x})) \} \\ \mathtt{m\ :=\ newNodeAfter(n)} \\ \left\{ \exists m \in \mathbb{N}^+ . \begin{array}{l} (\mathsf{var}\,(\mathtt{n}, n) * \mathsf{var}\,(\mathtt{m}, m)) \ \times \\ (\mathsf{ATree}\,(n[\mathbf{y}] \otimes m[\varnothing])(\mathbf{x})) \end{array} \right\}\end{array} \qquad \begin{array}{c}\left\{ (\mathsf{var}\,(\mathtt{n}, n)) \times (\mathsf{ATree}\,(n[t_c])(\mathbf{x})) \right\} \\ \mathtt{deleteTree(n)} \\ \left\{ (\mathsf{var}\,(\mathtt{n}, n)) \times (\mathsf{ATree}\,(\varnothing)(\mathbf{x})) \right\}\end{array}$$

$$\begin{array}{c}\left\{ (\mathsf{var}\,(\mathtt{m}, m) * \mathsf{var}\,(\mathtt{n}, n)) \times (\mathsf{ATree}\,(m[\mathbf{z}])(\mathbf{y}) * \mathsf{ATree}\,(n[t_c])(\mathbf{x})) \right\} \\ \mathtt{appendChild(m,\ n)} \\ \left\{ (\mathsf{var}\,(\mathtt{m}, m) * \mathsf{var}\,(\mathtt{n}, n)) \times (\mathsf{ATree}\,(m[\mathbf{z} \otimes n[t_c]])(\mathbf{y}) * \mathsf{ATree}\,(\varnothing)(\mathbf{x})) \right\}\end{array}$$

Fig. 3. Axiomatisation of tree library commands: assume arbitrary $\mathtt{m}, \mathtt{n} \in \mathrm{PV{\small ARS}}$, $l, m, n, o \in \mathbb{N}^+ \uplus \{\mathsf{null}\}$, $\mathbf{w}, \mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathrm{AA{\small DD}}$ and $t_c \in \mathrm{C{\small OMP}D{\small ATA}}_{\mathbb{T}}$.

For sets of abstract tree heaps, we write $\mathsf{ATree}\,(d)(\mathbf{x})$ for $\{\mathbf{x} \to d\}$ and $p * q$ for $\{h_1 \bullet_{\mathbb{T}} h_2 \mid h_1 \in p \wedge h_2 \in q\}$. For abstract tree states, we write $\exists v \in V. (p \times q)$ for $\{(\sigma, h) \mid v \in V \wedge \sigma \in p \wedge h \in q\}$ assuming any carrier set V. Finally, we define the *set of complete abstract trees* as the set of abstract trees with no context holes: $\mathrm{C{\small OMP}D{\small ATA}}_{\mathbb{T}} = \{d \in \mathrm{D{\small ATA}}_{\mathbb{T}} \mid \mathit{addrs}(d) = \varnothing\}$.

Definition 2.7 The axiomatisation of atomic tree commands :

$$\mathrm{A{\small XIOM}}_{\mathbb{T}} : \mathrm{A{\small TOM}}_{\mathbb{T}} \to (\Sigma \times H_{\mathbb{T}}) \times (\Sigma \times H_{\mathbb{T}})$$

is defined in Fig. 3. For soundness, each axiom must preserve the set of abstract addresses present in the described data. Failure to do so would give *unstable* commands, as the abstract connectivity described by abstract addresses would be broken. This is evident in the `deleteTree` axiom, which requires that the subtree being removed be *complete*, in that it contains no context holes. If the sub-tree did contain a context hole, it would be destroyed, and there would be some matching abstract heap cell which could not be connected anywhere.

The technical details of the Views framework uses a labelled transition system to describe transitions between states. Transitions are labelled either by atomic commands or by id which labels computation steps in which states are not changed. We extend the behaviour of the id transitions of Views by declaring the relation $\mathrm{A{\small XIOM}}_{\mathrm{ID}}$ for abstract allocation/deallocation. Abstract (de)allocation does not change the underlying program states and can therefore be seen as id transitions.

Definition 2.8 The *identity axiomatisation*: $\mathrm{A{\small XIOM}}_{\mathrm{ID}} : (\Sigma \times H_{\mathbb{T}}) \times \{\mathsf{id}\} \times (\Sigma \times H_{\mathbb{T}})$

is given by the abstract allocation and deallocation axioms:

$$\{\mathsf{ATree}\,(d_1 \circ_{\mathbf{x}} d_2)(a)\}\ \mathsf{id}\ \{\exists \mathbf{y}.\, \mathsf{ATree}\,(d_1 \circ_{\mathbf{x}} \mathbf{y})(a) * \mathsf{ATree}\,(d_2)(\mathbf{y})\}$$

$$\{\exists \mathbf{y}.\, \mathsf{ATree}\,(d_1 \circ_{\mathbf{x}} \mathbf{y})(a) * \mathsf{ATree}\,(d_2)(\mathbf{y})\}\ \mathsf{id}\ \{\mathsf{ATree}\,(d_1 \circ_{\mathbf{x}} d_2)(a)\}$$

The existential quantification of the abstract address $\mathbf{y}$ is analogous to the existential quantification used for heap allocation in separation logic.

Definition 2.9 Given the set of abstract tree heaps $H_{\mathbb{T}}$ (Def. 2.4), the set of atomic commands $\textsc{Atom}_{\mathbb{T}}$ (Def. 2.1) and their axiomatisation $\textsc{Axiom}_{\mathbb{T}}$ (Def. 2.7), the abstract tree library $\mathbb{T}$ is defined as: $\mathbb{T} \triangleq (H_{\mathbb{T}}, \textsc{Atom}_{\mathbb{T}}, \textsc{Axiom}_{\mathbb{T}} \cup \textsc{Axiom}_{\mathrm{ID}})$

Definition 2.10 Given abstract tree states $p, q \in \wp(\Sigma \times H_{\mathbb{T}})$ (Def. 2.6) and program $C \in \textsc{Prog}_{\mathbb{T}}$ (Def. 2.1), an *abstract triple* is $\Omega \vDash_{\mathbb{T}} \{p\}C\{q\}$. $\Omega \in \mathrm{PE}\textsc{nv}_{\mathbb{T}}$ is a *procedure specification environment* which is a set of procedure specifications $\mathtt{f} : p_1 \rightarrowtail q_1$, where $\mathtt{f}$ is a procedure name and $p_1, q_1 \in \wp(\Sigma \times H_{\mathbb{T}})$ are pre/post-conditions of $\mathtt{f}$.

3 Concurrent abstract predicates: Tree Implementation

We use concurrent abstract predicates (CAP) to reason about our implementation. We describe the concrete representation of our abstract trees and the concrete implementation of the tree commands in §3.1. We reason about our implementation in §3.2. We give an informal account of how to establish its correctness with respect to the abstract specification in §3.3, and give the formal justification in §4.

3.1 Concrete Tree Implementation

Tree Representation We give a concrete representation of the abstract tree heaps introduced in §2. Consider the abstract tree heap depicted in Fig. 1a. The corresponding concrete tree heap is illustrated in Fig. 4. Each tree node is represented by a node cell with pointers to its left sibling, parent, first child, last child and right sibling where ⊣ denotes a null pointer. With the abstract tree heap, the deletion of the subtree rooted at `n` is a self-contained operation; it does not rely on the context surrounding the subtree. However, this is not the case for the implementation. Since each node cell maintains pointers to its siblings, deletion of the subtree at n requires altering the outgoing pointers of its siblings (and sometimes the parent). In this example, the right pointer of node cell l must be redirected to r and vice-versa; only then can the subtree at n be discarded. Furthermore, in the implementation of the concurrent client program `deleteTree(l) || deleteTree(n)`, both executing threads need to access the pointers between l and n simultaneously, which calls for a suitable synchronisation technique. In our implementation, we synchronise access to these pointers through locking. Each of the left, first, last and right pointers are protected

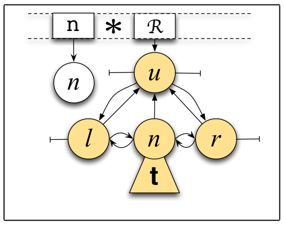


Fig. 4: Concrete tree representation.

```
proc deleteTree(n){
  local  l,u,d,r   in
    //Acquiring the necessary pointer locks.
     u := [n.up] ; lock(n.leftL) ; l:= [n.left] ;
     if l ≠ null then lock(l.rightL) else if u ≠ null then lock(u.firstL) ;
     lock(n.rightL) ; r:= [n.right] ;
     if r ≠ null then lock(r.leftL) ;  else if u ≠ null then lock(u.lastL) ;
    //Pointer Swinging.
     if l ≠ null then [l.right]:= r else if u ≠ null then [u.first]:= r
     if r ≠ null then [r.left]:= l else if u ≠ null then [u.last]:= l
    //Unlocking the acquired pointer locks.
    if l ≠ null then unlock(l.rightL) else if u ≠ null then unlock(u.firstL)
    if r ≠ null then unlock(r.leftL) else if u ≠ null then unlock(u.lastL)
    //Disposing the sub-tree at node n
    d := [n.first] ;  call disposeForest(d) ;  dealloc(n, 9) ;
}
proc disposeForest(n){
  local r,d in
    if n ≠ null then
      r := [n.right] ;  call disposeForest(r) ;
      d := [n.first] ;  call disposeForest(d) ;
      dealloc(n, 9)
}

proc lock(a){
  while(!CAS(a, 0, 1)) skip ;
}
proc unlock(a){
  [a]:= 0 ;
}
```

Fig. 5. Implementation of `deleteTree(n)` and some auxiliary procedures.

by corresponding locks (lL, dL, eL, rL) that are to be acquired by the competing threads beforehand. We refer to these locks as *pointer locks*.

We represent each tree node as a node cell consisting of a block of nine consecutive heap cells, $n \to l, u, d, e, r, lL, dL, eL, rL$ where the first five cells contain pointers to the siblings, parent and children, and the last four cells contain the pointer locks. The primary reason for choosing this particular representation was its resemblance to the representation of tree nodes in the Document Object Model (DOM) implementation of the WebKit project[6]. To increase readability, we write $n.l, n.u, \cdots, n.rL$ for $n, n+1, \cdots, n+8$, respectively.

Tree Implementation We provide a concrete implementation of each of the abstract tree commands. Fig. 5 shows our implementation of the `deleteTree` command with some auxiliary procedures. The implementation needs to cater for various cases regarding the siblings of node `n` such as when `n` does not have a left or right sibling. The implementation of the remaining commands are given in the technical report [18]. Note that this implementation is prone to deadlocks; this can occur with our example program `deleteTree(l) || deleteTree(n)` when `l` and `n` are adjacent siblings. We focus on this simple implementation here as it is sufficient to describe our ideas. We reason about a *deadlock-free* implementation in [18].

[6] However, WebKit supports only sequential DOM manipulation and does not use locks.

3.2 Reasoning about Concrete Tree Implementation

Recall that the pointers between two sibling nodes (or sometimes a parent and child node) are *shared* resources accessible by both nodes. We use concurrent abstract predicates (CAP) [1] to manage this sharing.

Concurrent Abstract Predicates With CAP, the state is modelled as a pair consisting of a thread-local state and a shared state. The shared state is divided into a set of regions, each encompassing some shared portion of the state. Each region is identified by a region identifier R and is governed by a protocol that describes how the resources of the region can be manipulated. For instance, a lock resource at location x can be specified by:

$$\mathsf{lock}\,(x) \triangleq \exists \mathsf{R}, \pi.\, [\mathcal{L}]^{\mathsf{R}}_{\pi} * \boxed{\mathsf{Unlocked}(\mathsf{R}, x) \,\vee\, \mathsf{Locked}(\mathsf{R}, x)}^{\mathsf{R}}_{T(\mathsf{R},x)}$$

where $\mathsf{Unlocked}(\mathsf{R}, x) \triangleq x \mapsto 0 * [\mathcal{U}]^{\mathsf{R}}_{1}$ and $\mathsf{Locked}(\mathsf{R}, x) \triangleq x \mapsto 1$. This lock definition states that there exists a shared region R containing the lock at location x and the thread's local state contains *some* (for permission $\pi \in (0,1]$) locking capability $[\mathcal{L}]^{\mathsf{R}}_{\pi}$ to acquire it. The region is in one of two states: either the lock is unlocked ($x \mapsto 0$) and the region holds the *full* capability $[\mathcal{U}]^{\mathsf{R}}_{1}$ to unlock it; or the lock is taken ($x \mapsto 1$) and the unlocking capability has been claimed by the locking thread.

The protocol of a shared region is specified through a set of actions, such as actions $\mathcal{L}$ and $\mathcal{U}$ for the lock example. A thread can perform an action if it has a non-zero capability for that action (such as the capability $[\mathcal{L}]^{\mathsf{R}}_{\pi}$ for $\pi > 0$). The actions permitted on the lock region are declared in $T(\mathsf{R}, x)$:

$$T(\mathsf{R}, x) \triangleq \left\{ \mathcal{L} : \left(x \mapsto 0 * [\mathcal{U}]^{\mathsf{R}}_{1}\right) \rightsquigarrow x \mapsto 1 \qquad \mathcal{U} : x \mapsto 1 \rightsquigarrow \left(x \mapsto 0 * [\mathcal{U}]^{\mathsf{R}}_{1}\right) \right.$$

The action associated with $\mathcal{L}$ changes the state of the shared region from unlocked to locked, moving the capability $[\mathcal{U}]^{\mathsf{R}}_{1}$ to the thread's local state. The action associated with $\mathcal{U}$ behaves dually, returning $[\mathcal{U}]^{\mathsf{R}}_{1}$ from the local state to the shared region.

The definition of the CAP separation algebra is given in [1,6]. It provides a set of instrumented states $M_{\mathbb{H}}$ consisting of a local state, a shared state and an action relation capturing the ways in which the shared state can be manipulated. The definition is parametrised by an underlying separation algebra for describing the local state. For this paper, we work with the CAP separation algebra instantiated with the standard fractional heap separation algebra.

Definition 3.1 The *CAP separation algebra*, instantiated with fractional heap separation algebra, is $\mathcal{A}_{\mathbb{C}_{\mathbb{H}}} \triangleq (M_{\mathbb{H}}, \bullet_{\mathbb{C}_{\mathbb{H}}}, \mathbf{0}_{\mathbb{C}_{\mathbb{H}}})$. The *separation algebra of CAP states* is $\mathcal{SA}_{\mathbb{C}_{\mathbb{H}}} \triangleq (\Sigma \times M_{\mathbb{H}}, \bullet_{\sigma} \times \bullet_{\mathbb{C}_{\mathbb{H}}}, \mathbf{0}_{\sigma} \times \mathbf{0}_{\mathbb{C}_{\mathbb{H}}})$, where × denotes standard Cartesian product.

Recall that we base our reasoning on the Views framework [6] which provides general reasoning about a generic programming language parametrised by a set of atomic commands and their axiomatisation.

Definition 3.2 Let $\textsc{Atom}_{\mathbb{H}}$ be the set of atomic heap commands including assignment, value lookup, memory allocation/deallocation and the compare and set construct `CAS`. We write $\text{Prog}_{\mathbb{H}}$ for the set of programs generated from Views' programming language instantiated with set $\text{Atom}_{\mathbb{H}}$. Let $\text{Axiom}_{\mathbb{H}}$ be the standard set of separation-logic axiomatisations for these commands based on the fractional heap separation algebra. The Views framework provides the associated reasoning for $\text{Prog}_{\mathbb{H}}$.

Definition 3.3 Given the CAP separation algebra $\mathcal{A}_{\mathbb{C}_{\mathbb{H}}}$ (Def. 3.1), the atomic heap commands $\text{Atom}_{\mathbb{H}}$ and their axiomatisation $\text{Axiom}_{\mathbb{H}}$, the *CAP library* $\mathbb{C}_{\mathbb{H}}$ for fractional heaps is defined as: $\mathbb{C}_{\mathbb{H}} \triangleq (\mathcal{A}_{\mathbb{C}_{\mathbb{H}}}, \text{Atom}_{\mathbb{H}}, \text{Axiom}_{\mathbb{H}})$.

Definition 3.4 Given CAP states $p, q \in \wp(\Sigma \times M_{\mathbb{H}})$ (Def. 3.1) and program $C \in \text{Prog}_{\mathbb{H}}$ (Def. 3.2), a *concrete triple* is $\Omega \vDash_{\mathbb{C}_{\mathbb{H}}} \{p\}C\{q\}$. $\Omega \in \text{PEnv}_{\mathbb{C}_{\mathbb{H}}}$ is a set of procedure specifications $\mathtt{f} : p_1 \rightarrowtail q_1$, where `f` is a procedure name and $p_1, q_1 \in \wp(\Sigma \times M_{\mathbb{H}})$ are pre/post-conditions of `f`.

In §4, we define the library refinement $\tau : \mathbb{T} \to \mathbb{C}_{\mathbb{H}}$ for abstract tree library $\mathbb{T}$ (Def. 2.9) and concrete CAP library $\mathbb{C}_{\mathbb{H}}$ (Def. 3.3). As a first step in defining τ, we identify the concrete CAP states corresponding to the abstract tree fragments. These concrete CAP states rely on an *interface function* I_τ linking abstract addresses with concrete pointers.

Interface Functions. Fig. 4 illustrates a concrete heap representation of a complete abstract tree at root address $\mathcal{R}$. However, many of the abstract tree commands (e.g. `deleteTree`) are specified using tree fragments rather than complete trees. We need to understand the concrete representation of tree fragments.

When an abstract tree heap is split using abstract allocation, the constituent heaps are agnostic to one-another's shapes. For instance, consider the abstract tree heap $\mathcal{R} \mapsto l \otimes i \otimes j \otimes r$ which can be split as $h_1 \bullet_{\mathbb{T}} h_2$ where $h_1 \triangleq \mathcal{R} \mapsto l \otimes \mathbf{x} \otimes r$ and $h_2 \triangleq \mathbf{x} \mapsto i \otimes j$. The abstract tree heap h_1 embodies no knowledge of the tree placed within $\mathbf{x}$; *mutatis mutandis* for h_2. At the concrete level, the situation is different. The concrete representation of h_2 relies on additional knowledge from the concrete representation of h_1, since the representation of node i includes pointers to its left sibling (l) and parent (u). Thus, when translating an abstract heap with abstract addresses, we require supplementary information originating from the concrete heap. We track this additional piece of information associated with each abstract address $\mathbf{x} \in \text{AAdd}$ through a concrete *interface*. Consider Fig. 6, which depicts an abstract tree at abstract cell $\mathbf{x}$ (left) and its concrete representation (right) assuming an *interface function* I_τ. In the concrete representation, the solid lines represent resources held locally, such as node n, its up pointer and the subtree t, while the dashed lines denote shared resources, such as the pointers between node n and its siblings. A concrete *in-interface* records the address of the first (i) and last (j) nodes of the concrete representation of the forest pointed to by $\mathbf{x}$; in our example, the in-interface is $in \triangleq (n, n)$. A concrete *out-interface*

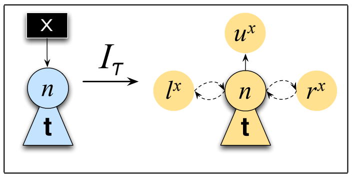


Fig. 6: Tree Fragments.

$$\begin{aligned}
\mathsf{CTree}^{I_\tau}(t)(\mathcal{R}) &\triangleq \exists i,j.\, \mathsf{ICTree}^{I_\tau}\,(t)\,(i,j)\,(\mathsf{null},\mathsf{null},\mathsf{null})\\
\mathsf{CTree}^{I_\tau}(t)(\mathbf{x}) &\triangleq \mathsf{ICTree}^{I_\tau}\,(t)\,(i^x,j^x)\,(l^x,u^x,r^x)\\
&\quad\; * {I_\tau}^{\mathsf{in}}(\mathbf{x}) \doteq (i^x,j^x) * {I_\tau}^{\mathsf{out}}(\mathbf{x}) \doteq (l^x,u^x,r^x)\\[6pt]
\mathsf{ICTree}^{I_\tau}\,(\varnothing)\,(i,j)\,(l,u,r) &\triangleq (i \doteq r) * (j \doteq l)\\
\mathsf{ICTree}^{I_\tau}\,(\mathbf{x})\,(i,j)\,(l,u,r) &\triangleq {I_\tau}^{\mathsf{in}}(\mathbf{x}) \doteq (i,j) * {I_\tau}^{\mathsf{out}}(\mathbf{x}) \doteq (l,u,r)\\
\mathsf{ICTree}^{I_\tau}\,(t_1 \otimes t_2)\,(i,j)\,(l,u,r) &\triangleq \exists p,q.\ \mathsf{ICTree}^{I_\tau}\,(t_1)\,(i,p)\,(l,u,q)\\
&\quad\; * \mathsf{ICTree}^{I_\tau}\,(t_2)\,(q,j)\,(p,u,r)\\
\mathsf{ICTree}^{I_\tau}\,(n[t])\,(i,j)\,(l,u,r) &\triangleq i \doteq j \doteq n \ * \ n.u \to u \ * \exists d,e.\\
&\quad \begin{pmatrix} \mathsf{Left}\,(n,l,u) * \mathsf{First}\,(n,d) * \mathsf{Last}\,(n,e) * \mathsf{Right}\,(n,r,u)\\ * \mathsf{ICTree}^{I_\tau}\,(t)\,(d,e)\,(\mathsf{null},n,\mathsf{null}) \end{pmatrix}
\end{aligned}$$

Fig. 7. CAP Representation of Abstract Tree Heaps: we write $a \doteq b$ for $a = b \wedge \mathsf{emp}$.

records the addresses of the parent node (u) and nodes placed immediately to the left (l) and right (r) of the abstract address $\mathbf{x}$; in our example, the out-interface is $out \triangleq (l^x, u^x, r^x)$. The interface function I_τ maps $\mathbf{x}$ to (in, out).

Definition 3.5 The sets of concrete *in-* and *out-interfaces* are defined by:

$$\textsc{In}_\tau \triangleq (\mathbb{N}^+ \uplus \{\mathsf{null}\})^2 \qquad\qquad \textsc{Out}_\tau \triangleq (\mathbb{N}^+ \uplus \{\mathsf{null}\})^3$$

The set of *in-* and *out-interface functions* are defined by $\mathcal{I}_\tau^{\mathsf{in}} \triangleq \wp(\textsc{AAdd} \rightharpoonup \textsc{In}_\tau)$ and $\mathcal{I}_\tau^{\mathsf{out}} \triangleq \wp(\textsc{AAdd} \rightharpoonup \textsc{Out}_\tau)$. The set of interface functions is $\mathcal{I}_\tau \triangleq \mathcal{I}_\tau^{\mathsf{in}} \times \mathcal{I}_\tau^{\mathsf{out}}$.

CAP Representation of Trees Fig. 7 defines concurrent abstract predicates for describing the concrete representation of abstract tree heaps, parameterised by interface function I_τ. The $\mathsf{CTree}^{I_\tau}(t)(\mathcal{R})$ predicate provides a concrete representation of the tree fragment t at root address $\mathcal{R}$. The predicate $\mathsf{CTree}^{I_\tau}(t)(\mathbf{x})$ provides a concrete representation of the tree fragment at abstract address $\mathbf{x}$. Both predicates are defined using a ICTree predicate indexed by in- and out-interfaces determined by I_τ. For $\mathcal{R}$, the out-interface is $(\mathsf{null}, \mathsf{null}, \mathsf{null})$. The in-interface is unknown at this point and hence is existentially quantified. Its value is later found using the ICTree predicate. The interface of $\mathbf{x}$ is determined by the interface function I_τ.

We now describe the ICTree predicate with the explicit in- and out-interface arguments (i,j) and (l,u,r), following the cases of the inductive definition.

$\boldsymbol{\mathsf{ICTree}^{I_\tau}(\varnothing)}$ There is no resource in the concrete heap representation, simply some pointer equalities. The assertion $i \doteq r$ simply states that the address of the first node cell in the forest is equal to the address of the node cell immediately to the right of the tree; similarly for j and l. This is to ensure the correct concrete representation of trees such as $t_1 \otimes \varnothing \otimes t_2$.

$\boldsymbol{\mathsf{ICTree}^{I_\tau}(\mathbf{x})}$ There is no resource in the concrete heap representation, simply the appropriate connection between abstract address $\mathbf{x}$ and the interfaces given by I_τ.

ICTree$^{I_\tau}(t_1 \otimes t_2)$ This is simply the $*$-composition of the concrete representations of the constituent abstract subtrees, given an appropriate choice of interfaces.

ICTree$^{I_\tau}(n[t])$ The first two lines provide the concrete representation of the node n, and the last line represents the subtree t. For the subtree, the concrete representation is straightforward. For the node, $i \doteq j \doteq n$ since there is only one tree in the forest. The concrete representation has full ownership of the parent pointer $n.u \to u$, as there is no competition for this resource from other nodes. The Left, Right, First and Last predicates represent the pointers and locks associated with the left sibling, right sibling, first child and last child, which are in competition with the representations of nearby nodes and hence must be shared.

We give the definition of Left predicate here; the Right, First and Last predicates are defined analogously and we defer them to the accompanying technical report [18].

The definition of the Left predicate is given in Fig. 8 and is described by the isLLock and ownsR predicates.

isLLock$(n, l, u, 0.5)$ This predicate states that there exists a shared region R_{nl} containing the left pointer lock of node n; the thread's local state contains some locking capability $[\mathcal{L}]_{0.5}^{\mathsf{R}_{nl}}$ associated with the region.

ownsR$(n, l, u, 1)$ Similar to isLLock, this predicate refers to the R_{nl} region where the thread's local state holds *full* permission on the *witness* capability $[\mathcal{W}]_{1}^{\mathsf{R}_{nl}}$. This is to denote that even though the pointers between n and its left sibling l are shared, node n itself is owned by the current thread. Later in the description of LWit predicate we demonstrate how this capability can be used to track the identity of the thread that has claimed the left pointer lock of n.

The contents of the R_{nl} region are analogous to that of the lock example given before. The difference is the additional pointer resource ($\Subset$) contained within the region. The region can be in one of two states: LUnlocked or LLocked.

LUnlocked$(\mathsf{R}_{nl}, n, l, u)$ In this state, the left pointer lock is not taken ($n.lL \to 0$), and the full unlocking capability ($[\mathcal{U}]_{1}^{\mathsf{R}_{nl}}$) and the $\Subset$ resource are in the region.

$\Subset^{n,l,u}$ This predicate describes the pointer resources between node n and its left hand side depending on its position in the tree. When n has a left sibling ($l \neq \mathsf{null}$), it consists of partial ownership of the pointers between n and l, that is, $n.l \overset{0.5}{\mapsto} l * l.r \overset{0.5}{\mapsto} n$. It also contains the capability to acquire the right pointer lock of l, in order to obtain full ownership of the pointers between node n and l. This is captured by the isRLock$\,(l, n, u, 0.5)$ predicate. On the other hand, if n does not have a left sibling and is thus the first child of node u ($l = \mathsf{null} \wedge u \neq \mathsf{null}$), the $\Subset$ resource consists of partial ownership on the left pointer of n and the first pointer of u ($n.l \overset{0.5}{\mapsto} l * u.d \overset{0.5}{\mapsto} n$) as well as the capability to acquire the first pointer lock of u (isDLock$\,(u, n, 0.5)$). Finally, if n is the first child underneath the root address $\mathcal{R}$, and thus both its left sibling and parent correspond to null ($l = u = \mathsf{null}$), the $\Subset$ resource consists of full permission on n's left pointer ($n.l \overset{1}{\mapsto} l$) and the remaining

$$\mathsf{Left}\,(n,l,u) \triangleq \mathsf{isLLock}\,(n,l,u,0.5) * \mathsf{ownsR}\,(n,l,u,1)$$

$$\mathsf{isLLock}\,(n,l,u,\pi) \triangleq \exists \mathsf{R}_{nl}.\, [\mathcal{L}]^{\mathsf{R}_{nl}}_{\pi} * \boxed{\begin{array}{l}\mathsf{LUnlocked}\,(\mathsf{R}_{nl},n,l,u)\\ \vee\, \mathsf{LLocked}\,(\mathsf{R}_{nl},n,l,u)\end{array}}^{\mathsf{R}_{nl}}_{\mathsf{LC}(\mathsf{R}_{nl},n,u)}$$

$$\mathsf{ownsR}\,(n,l,u,\pi) \triangleq \exists \mathsf{R}_{nl}.\, [\mathcal{W}]^{\mathsf{R}_{nl}}_{\pi} * \boxed{\begin{array}{l}\mathsf{LUnlocked}\,(\mathsf{R}_{nl},n,l,u)\\ \vee\, \mathsf{LLocked}\,(\mathsf{R}_{nl},n,l,u)\end{array}}^{\mathsf{R}_{nl}}_{\mathsf{LC}(\mathsf{R}_{nl},n,u)}$$

$$\mathsf{LUnlocked}\,(\mathsf{R}_{nl},n,l,u) \triangleq n.lL \to 0 * [\mathcal{U}]^{\mathsf{R}_{nl}}_{1} * \Subset^{n,l,u}$$

$$\Subset^{n,l,u} \triangleq n.l \overset{0.5}{\mapsto} l \;* \left(\begin{array}{l}\Big(l \neq \mathsf{null} \wedge l.r \overset{0.5}{\mapsto} n * \; \mathsf{isRLock}\,(l,n,u,0.5)\Big)\\ \vee \Big(l = \mathsf{null} \wedge u \neq \mathsf{null} \wedge \; u.d \overset{0.5}{\mapsto} n * \; \mathsf{isDLock}\,(u,n,0.5)\Big)\\ \vee \Big(l \doteq u \doteq \mathsf{null} * n.l \overset{0.5}{\mapsto} l * \mathsf{isLLock}\,(n,l,u,0.5)\Big)\end{array}\right)$$

$$\mathsf{LLocked}\,(\mathsf{R}_{nl},n,l,u) \triangleq n.lL \to 1 * \mathsf{LWit}\,(l,n,u)$$

$$\begin{aligned}\mathsf{LWit}\,(l,n,u) \triangleq\; & \mathsf{ownsR}\,(l,n,u,0.5)\\ & \vee\, (l \neq \mathsf{null} \wedge \mathsf{ownsL}\,(l,n,u,0.5))\\ & \vee (l = \mathsf{null} \wedge u \neq \mathsf{null} \wedge \mathsf{ownsD}\,(u,n,0.5))\end{aligned}$$

$$\mathsf{LC}(\mathsf{R}_{nl},n,u) \triangleq \begin{cases} \mathcal{L}: & \mathsf{LUnlocked}\,(\mathsf{R}_{nl},n,l,u) \rightsquigarrow \mathsf{LLocked}\,(\mathsf{R}_{nl},n,l,u)\\ \mathcal{U}: & \mathsf{LLocked}\,(\mathsf{R}_{nl},n,l,u) \rightsquigarrow \mathsf{LUnlocked}\,(\mathsf{R}_{nl},n,l',u)\\ \mathcal{W}: & \mathsf{LInit}\,(n,l,u) \rightsquigarrow \mathsf{LUnlocked}\,(\mathsf{R}_{nl},n,l,u) \vee \mathsf{LLocked}\,(\mathsf{R}_{nl},n,l,u)\\ & \mathsf{LUnlocked}\,(\mathsf{R}_{nl},n,l,u) \vee \mathsf{LLocked}\,(\mathsf{R}_{nl},n,l,u) \rightsquigarrow \mathsf{LInit}\,(n,l,u)\end{cases}$$

Fig. 8. The definition of Left and its auxiliary predicates.

locking capability on region R_{nl} ($\mathsf{isLLock}\,(n,l,u,0.5)$).

LLocked(R_{nl},n,l,u) In this state, the left pointer lock is taken ($n.lL \to 1$), and the capability $[\mathcal{U}]^{\mathsf{R}_{nl}}_{1}$ and the pointer resources $\Subset$ have been claimed by the locking thread in exchange for the LWit resource.

LWit(l,n,u) Since the left pointer lock of n can only be be acquired by the thread in possession of node n, *or* node l (or node u if n is the first child of u and does not have a left sibling), the LWit predicate is used to track the identity of the locking thread. Recall from the definition of the Left predicate that the full *witness* capability $[\mathcal{W}]^{\mathsf{R}_{nl}}_{1}$ is held by the thread that owns node n, as described by $\mathsf{ownsR}\,(n,l,u,1)$. We thus use the witness capabilities to determine the identity of the locking thread. The first disjunct denotes the case where the lock has been claimed by the thread in possession of node n and corresponds to the witness capability on this region ($\mathsf{ownsR}\,(n,l,u,0.5)$). Analogously, the second disjunct represents the case where the thread in possession of node l has acquired the lock, denoted by $(\mathsf{ownsR}\,(n, l, u, 0.5))$. The third disjunct captures the case where n does not have a left sibling and the lock has been taken by the thread that owns the parent node u $(\mathsf{ownsD}\,(u, n, 0.5))$.

$\mathsf{LC}(\mathsf{R}_{nl}, n, u)$ The R_{nl} region is governed by the $\mathsf{LC}(\mathsf{R}_{nl}, n, u)$ protocol describing the ways in which its contents can be manipulated through actions. The action associated with $\mathcal{L}$ changes the state of R_n from $\mathsf{LUnlocked}$ to $\mathsf{LLocked}$. Dually, the action of $\mathcal{U}$ changes the state of the region from $\mathsf{LLocked}$ to $\mathsf{LUnlocked}$. The first action of $\mathcal{W}$ *initialises* the contents of R_{nl} immediately after its creation. Similarly, the second action *finalises* the contents of the region right before its destruction. The initial/final contents of the region are realised by the LInit predicate. The definition of LInit predicate is nonessential as it bears no relevance in understanding the interactions between the threads and the shared region and is thus omitted here.

Recall that abstract trees can be split and joined using abstract allocation and deallocation. We show that concrete trees can be split (joined) analogously provided that the interface function is extended (reduced) accordingly to capture the interface associated with the freshly allocated (deallocated) abstract address.

Theorem 3.6 (Abstract (de)allocation) *For all interface functions $I_\tau \in \mathcal{I}_\tau$, addresses $a \in \textsc{Add}_{\mathbb{T}}$ and trees $t_1, t_2 \in \textsc{Data}_{\mathbb{T}}$:*

$$\mathit{CTree}^{I_\tau}(t_1 \circ_{\mathbf{x}} t_2)(a) \equiv \exists \mathbf{y} \in \textsc{AAdd}, \mathit{in} \in \textsc{In}_\tau, \mathit{out} \in \textsc{Out}_\tau. \\ \mathit{CTree}^{I'_\tau}(t_1 \circ_{\mathbf{x}} \mathbf{y})(a) * \mathit{CTree}^{I'_\tau}(t_2)(\mathbf{y})$$

where $\textsc{In}_\tau, \textsc{Out}_\tau$ and $\mathcal{I}_\tau$ are given in Def. 3.5 and $I'_\tau \triangleq I_\tau \uplus ([\mathbf{y} \mapsto \mathit{in}], [\mathbf{y} \mapsto \mathit{out}])$.

Proof. The proof of this theorem is provided in the technical report [18].

3.3 Soundness of Concrete Tree Library $\mathbb{C}_{\mathbb{H}}$

In order to show that our concrete CAP library $\mathbb{C}_{\mathbb{H}}$ (Def. 3.3) is sound with respect to the abstract tree library $\mathbb{T}$ (Def. 2.9), we show that everything that can be proved about the abstract library $\mathbb{T}$, can also be proved about the concrete library $\mathbb{C}_{\mathbb{H}}$. That is, for every abstract triple $\Omega \vDash_{\mathbb{T}} \{p\}C\{q\}$ (Def. 2.10), we show that there exists a corresponding concrete triple $\Omega' \vDash_{\mathbb{C}_{\mathbb{H}}} \{p'\}C'\{q'\}$ (Def. 3.4) where C' is the implementation of C and p', q' denote the concrete representations of p, q, respectively. For instance, we need to show that the implementation of the `deleteTree` command in Fig. 5, satisfies its abstract specification as given in Fig. 3.

In §4, we formalise triple transformation (Def. 4.6) and define what it means for the CAP library $\mathbb{C}_{\mathbb{H}}$ to be sound with respect to the abstract tree library $\mathbb{T}$ (Def. 4.8). We then prove that $\mathbb{C}_{\mathbb{H}}$ is sound with respect to $\mathbb{T}$ (Theorem 4.9). In what follows, we give an informal account of establishing the correctness of the `deleteTree` implementation and state a *pseudo-theorem* that *almost* captures the desired result. We defer the formalisation of the correct theorem to §4 (Theorem 4.7), and the full proof to the accompanying technical report [18].

Pseudo-Theorem 1 *Let* $[\![\texttt{deleteTree(n)}]\!]_\tau$ *denote the implementation of* `deleteTree(n)` *command in Fig.* 5*. The following statement* almost *holds:*

$$\forall I_c \in \mathcal{I}_\tau. \quad \left\{ \exists I_d \in \mathcal{I}_\tau^{\texttt{del}}. \left(\mathit{var}(\mathtt{n}, n) \times\ \mathit{CTree}^{I_c \uplus I_d}(n[t])(\mathbf{x}) \right) \right\}$$

$$[\![\texttt{deleteTree(n)}]\!]_\tau$$

$$\left\{ \exists I_d \in \mathcal{I}_\tau^{\texttt{del}}. \left(\mathit{var}(\mathtt{n}, n) \times\ \mathit{CTree}^{I_c \uplus I_d}(\varnothing)(\mathbf{x}) \right) \right\}$$

with $\quad \mathcal{I}_\tau^{\texttt{del}} \triangleq \left\{ I \in \mathcal{I}_\tau \mid \mathit{dom}(I^{\mathit{in}}) = \{\mathbf{x}\} \wedge \mathit{dom}(I^{\mathit{out}}) = \varnothing \right\}.$

The pre-condition contains the variable resource $\mathsf{var}(\mathtt{n}, n)$ and the concrete tree representation $\mathsf{CTree}(n[t])(\mathbf{x})$ of complete subtree at abstract address $\mathbf{x}$. The pre-condition gives the right to control and modify the *inner* interface $I_\tau^{\texttt{del}}$ of address $\mathbf{x}$ (in this case, (n, n)). This freedom to modify $I_\tau^{\texttt{del}}$ is captured by the existential quantification of I_d in both pre- and post-conditions. Note that the control over the *outer* interface of $\mathbf{x}$ as well as the interfaces associated with other abstract addresses lies with the context. We therefore need to be agnostic to the interfaces associated with abstract addresses outside the domain of I_d and show that the implementation is correct for *all* valid choices of context interfaces I_c. This is realised by the universal quantification of I_c outside the Hoare triple. However, since we are reasoning about *concurrent* programs, during the execution of `deleteTree`, the interfaces captured by I_c are potentially changing underfoot. Hence, contrary to what the above triple states, the value of I_c in the pre-condition does not necessarily agree with that of I_c in the post-condition. The correct theorem is given in §4.

4 Correct Library Translation

We state what it means to correctly implement our abstract tree library $\mathbb{T}$ (Def. 2.9) with our CAP library $\mathbb{C}_\mathbb{H}$ (Def. 3.3) by defining a *library translation* $\tau : \mathbb{T} \to \mathbb{C}_\mathbb{H}$. In §4.1, we give our specific library translation τ and, in §4.2, we prove its soundness. In the accompanying technical report [18], we generalise this approach to translate arbitrary abstract libraries, and stipulate a set of general properties that when satisfied, will warrant sound translation of any abstract library. These properties are stated for our specific library translation in Theorem 4.9.

4.1 Tree to CAP Translation

Our goal is to define a translation $\tau : \mathbb{T} \to \mathbb{C}_\mathbb{H}$ in such a way that would allow us to *correctly* transform abstract tree triples to concrete CAP triples, following the spirit of Pseudo-theorem 1. We give an *abstract tree heap translation* that maps abstract tree heaps to CAP heaps. To keep the translation concise, we define it using CAP assertions which can then be interpreted as elements of $\wp(M_\mathbb{H})$ [1].

Definition 4.1 Given the separation algebra of abstract tree heaps (Def. 2.5) $\mathcal{A}_\mathbb{T} \triangleq (H_\mathbb{T}, \bullet_\mathbb{T}, \mathbf{0}_\mathbb{T})$, the tree interface function $\mathcal{I}_\tau$ (Def. 3.5) and the CAP separation algebra (Def. 3.1) $\mathcal{A}_{\mathbb{C}_\mathbb{H}} \triangleq (M_\mathbb{H}, \bullet_{\mathbb{C}_\mathbb{H}}, \mathbf{0}_{\mathbb{C}_\mathbb{H}})$, the *abstract tree heap translation* $\langle . \rangle_\tau^{(\cdot)} : \mathcal{H}_\mathbb{T} \to \mathcal{I}_\tau \to \wp(M_\mathbb{H})$ is a function defined inductively over the structure of abstract tree heaps:

$$\langle \mathbf{0}_{\mathbb{T}} \rangle_{\tau}^{I} \triangleq \mathrm{emp} \qquad \langle \mathcal{R} \to t \rangle_{\tau}^{I} \triangleq \mathsf{CTree}^{I}(t)(\mathcal{R}) \qquad \langle \mathbf{x} \to t \rangle_{\tau}^{I} \triangleq \mathsf{CTree}^{I}(t)(\mathbf{x})$$

$$\langle h_1 \bullet_{\mathbb{T}} h_2 \rangle_{\tau}^{I} \triangleq \exists I_1, I_2.\, I = I_1 \cup I_2 \wedge \langle h_1 \rangle_{\tau}^{I_1} * \langle h_2 \rangle_{\tau}^{I_2}$$

where the definition of the CTree predicate is as given in Fig. 7.

We also need to transform programs in $\textsc{Prog}_{\mathbb{T}}$ to programs in $\textsc{Prog}_{\mathbb{H}}$.

Definition 4.2 The *implementation function* $[\![.]\!]_{\tau} : \textsc{Prog}_{\mathbb{T}} \to \textsc{Prog}_{\mathbb{H}}$, replaces each call to an atomic tree command in $\textsc{Atom}_{\mathbb{T}}$ with a call to a correspondingly named procedure denoting its implementation, while other program constructs remain unchanged. The implementation of `deleteTree` is as given in Fig. 5; the implementation of other atomic commands are given in the technical report [18].

Definition 4.3 The translation $\tau : \mathbb{T} \to \mathbb{C}_{\mathbb{H}}$ is a triple comprising the set of interfaces $\textsc{In}_{\tau} \times \textsc{Out}_{\tau}$ (Def. 3.5), the abstract tree heap translation $\langle . \rangle_{\tau}^{(\cdot)}$ (Def. 4.1) and the implementation function $[\![.]\!]_{\tau}$ (Def. 4.2): $\tau \triangleq \left(\textsc{In}_{\tau} \times \textsc{Out}_{\tau}, \langle . \rangle_{\tau}^{(\cdot)}, [\![.]\!]_{\tau}\right)$.

4.2 Correctness of Translation

Given the translation τ, our goal is to state and prove the correctness of a translation of abstract tree triples $\Omega \vDash_{\mathbb{T}} \{p\}C\{q\}$ to concrete CAP triples $\Omega' \vDash_{\mathbb{C}_{\mathbb{H}}} \{p'\}\, [\![C]\!]_{\tau}\, \{q'\}$ in the spirit of Pseudo-Theorem 1. To do this, we need to define a *state translation* between abstract tree states and concrete CAP states.

Recall from §3.3 that the universal quantification of context interface function I_c in Pseudo-Theorem 1 is incorrect, since interfaces captured by I_c are subject to change by the environment. We define the set of *stable interface functions* $\mathcal{SI}_{\tau}$ whereby an abstract address is associated not with a single interface but with a *set* of interfaces. By associating a set of possible interfaces with an abstract address at any one time, we can account for the potential change of interfaces by the environment.

Definition 4.4 Given the sets of inner and outer interfaces $\textsc{In}_{\tau}$, $\textsc{Out}_{\tau}$ (def. 3.5), the *set of stable inner-interface functions* is $\mathcal{SI}_{\tau}{}^{\mathsf{in}} : \wp(\textsc{AAdd} \rightharpoonup \wp(\textsc{In}_{\tau}))$ and the *set of stable outer-interface functions* is $\mathcal{SI}_{\tau}{}^{\mathsf{out}} : \wp(\textsc{AAdd} \rightharpoonup \wp(\textsc{Out}_{\tau}))$. The set of *stable interface functions* is defined by $\mathcal{SI}_{\tau} \triangleq \mathcal{SI}_{\tau}{}^{\mathsf{in}} \times \mathcal{SI}_{\tau}{}^{\mathsf{out}}$. For $SI \in \mathcal{SI}_{\tau}$, SI^{in} and SI^{out} denote the first and second projections, respectively. The *collapse* of SI is $SI{\downarrow} \triangleq \left(SI{\downarrow}^{\mathsf{in}} \times SI{\downarrow}^{\mathsf{out}}\right)$ where $SI^{\mathsf{in}}{\downarrow}$ (and analogously $SI{\downarrow}^{\mathsf{out}}$) is defined as:

$$SI^{\mathsf{in}}{\downarrow} \triangleq \left\{ I \in \mathcal{I}_{\tau}^{\mathsf{in}} \,\middle|\, \mathrm{dom}(I) = \mathrm{dom}(SI^{\mathsf{in}}) \ \wedge\ \forall \mathbf{x} \in \mathrm{dom}(I).\, I(\mathbf{x}) \in SI^{\mathsf{in}}(\mathbf{x}) \right\}$$

We can now give a state translation between abstract tree states and concrete CAP states, parametrised by the stable interface functions.

Definition 4.5 The *state translation*: $\langle\!| . |\!\rangle_{\tau}^{(\cdot)} : \wp(\Sigma \times H_{\mathbb{T}}) \to \mathcal{SI}_{\tau} \to \wp(\Sigma \times M_{\mathbb{H}})$ is:

$$\langle\!| p |\!\rangle_{\tau}^{SI} \triangleq \left\{ (\sigma, m) \,\middle|\, \begin{array}{l} (\sigma, h) \in p \wedge \\ \exists I_d.\, m \in \bigcup\limits_{I_c \in SI{\downarrow}} \left\{ \langle h \rangle_{\tau}^{I_c \uplus I_d} \,\middle|\, \mathrm{dom}(I_d{}^{\mathsf{in}}) = h^{\mathsf{in}} \ \wedge\ \mathrm{dom}(I_d{}^{\mathsf{out}}) = h^{\mathsf{out}} \right\} \end{array} \right\}$$

where h^{in}, h^{out} are given in Def. 2.4 and $I_1 \uplus I_2 \triangleq \left({I_1}^{\mathsf{in}} \uplus {I_2}^{\mathsf{in}},\ {I_1}^{\mathsf{out}} \uplus {I_2}^{\mathsf{out}}\right)$ with $\uplus$ denoting the standard disjoint function union.

Note that when transforming a tree state, the variable store σ remains unchanged while the tree heap h is translated as $\langle h \rangle_\tau^{I_c \uplus I_d}$. I_d captures the interfaces of abstract addresses within the control of the thread and thus are in the footprint of h. On the other hand, I_c represents the interfaces of abstract addresses controlled by the environment. I_d and I_c are analogous to those of Pseudo-theorem 1, with the exception that they are drawn from the *stable interface function* SI.

We can now formalise the transformation of an abstract triple $\Omega \vDash_{\mathbb{T}} \{p\}C\{q\}$ to a concrete triple $(\!|\Omega|\!)_\tau \vDash_{\mathbb{C}_{\mathbb{H}}} \{\langle\!|p|\!\rangle_\tau\}\ [\![C]\!]_\tau\ \{\langle\!|q|\!\rangle_\tau\}$, where $(\!|.|\!)_\tau$ is defined in Def. 4.6.

Definition 4.6 Given the library translation $\tau : \mathbb{T} \to \mathbb{C}_{\mathbb{H}}$, for all procedure specification environments $\Omega \in \textsc{PEnv}_{\mathbb{T}}$, abstract tree states $p, q \in \wp(\Sigma \times H_{\mathbb{T}})$ and tree programs $C \in \textsc{Prog}_{\mathbb{T}}$, the *translated triple* $\mathcal{T}:\ \Omega \vDash_{\mathbb{T}} \{p\}C\{q\}$ is:

$$\mathcal{T}:\ \Omega \vDash_{\mathbb{T}} \{p\}\ C\ \{q\}\ \triangleq\ \forall SI \in \mathcal{SI}_\tau.\ \ (\!|\Omega|\!)_\tau \vDash_{\mathbb{C}_{\mathbb{H}}} \left\{\langle\!|p|\!\rangle_\tau^{SI}\right\}\ [\![C]\!]_\tau\ \left\{\langle\!|q|\!\rangle_\tau^{SI}\right\}$$

where $(\!|\Omega|\!)_\tau \triangleq \{\mathtt{f} : \langle\!|p|\!\rangle_\tau^{SI} \rightarrowtail \langle\!|q|\!\rangle_\tau^{SI} \mid (\mathtt{f} : p \rightarrowtail q) \in \Omega\ \wedge\ SI \in \mathcal{SI}_\tau\}$.

We can now amend the statement of Pseudo-theorem 1 and formulate the theorem describing the correctness of the `deleteTree` command.

Theorem 4.7 (`deleteTree` Correctness) *The implementation of* `deleteTree` *command as given in Fig. 5 is correct if*

$$\mathcal{T}:\ \Omega \vDash_{\mathbb{T}} \left\{\mathsf{var}(\mathtt{n}, n) \times \mathsf{ATree}(n[t])(\mathbf{x})\right\}\ \mathtt{deleteTree}\ \left\{\mathsf{var}(\mathtt{n}, n) \times \mathsf{ATree}(\varnothing)(\mathbf{x})\right\}$$

Proof. *The correctness proof of the above statement is given in the accompanying technical report [18].*

Definition 4.8 The library translation $\tau : \mathbb{T} \to \mathbb{C}_{\mathbb{H}}$ is *sound* if, for all $\Omega \in \textsc{PEnv}_{\mathbb{T}}$, $p, q \in \wp(\Sigma \times H_{\mathbb{T}})$ and $C \in \textsc{Prog}_{\mathbb{T}}$: $\quad \Omega \vDash_{\mathbb{T}} \{p\}\ C\ \{q\} \implies \mathcal{T}:\ \Omega \vDash_{\mathbb{T}}\ \{p\}\ C\ \{q\}$

Theorem 4.9 (Sound translation) *The library translation $\tau : \mathbb{T} \to \mathbb{C}_{\mathbb{H}}$ is sound.*

Proof. *The proof is by induction over the structure of C in $\Omega \vDash_{\mathbb{T}} \{p\}C\{q\}$ and is given in the technical report [18]. We show that translation τ has the following properties and appeal to these properties in the proof.*

Property 1 (Axiom Correctness) *For all $\Omega \in \textsc{PEnv}_{\mathbb{T}}$, $p, q \in \wp(\Sigma \times H_{\mathbb{T}})$, $A \in \textsc{Atom}_{\mathbb{T}}$,* $\quad \Omega \vDash_{\mathbb{T}} \{p\}\ A\ \{q\} \implies \mathcal{T}:\ \Omega \vDash_{\mathbb{T}} \{p\}\ A\ \{q\}$

This ensures that the translation correctly implements abstract atomic commands, as demonstrated intuitively throughout the paper with the `deletetree` *command.*

Property 2 (Monotonicity of** id **Relation) *For all $h_1, h_2 \in H_{\mathbb{T}}$ and $I \in \mathcal{I}_\tau$:*

$$\left\{\{h_1\}\right\}\ id\ \left\{\{h_2\}\right\} \implies \left\{\ \langle h_1 \rangle_\tau^I\ \right\}\ id\ \left\{\ \langle h_2 \rangle_\tau^I\ \right\}$$

A lifting of this property to abstract tree states is used in proof of rule of consequence.

Property 3 ($*$-Preservation) *For all $h_1, h_2 \in H_{\mathbb{T}}$ and $I \in \mathcal{I}_\tau$*

$$\langle h_1 \bullet_{\mathbb{T}} h_2 \rangle_\tau^I \equiv \exists I_1, I_2.\, I_1 \cup I_2 = I \;\wedge\; \langle h_1 \rangle_\tau^{I_1} \bullet_{\mathbb{C}_{\mathbb{H}}} \langle h_2 \rangle_\tau^{I_2}$$

A lifting of this property to tree states is used in proof of disjoint concurrency rule.

Concluding Remarks. We have highlighted a gap in reasoning between abstract libraries and concrete implementations, due to the mismatch between the abstract connectivity of abstract data fragments and the concrete connectivity of their concrete representations. We have illustrated this gap using SSL reasoning applied to an abstract concurrent tree library $\mathbb{T}$ and CAP reasoning about a concrete implementation. This gap is closed by a refinement translation $\tau : \mathbb{T} \to \mathbb{C}_{\mathbb{H}}$, which depends crucially on an interface function I_τ mapping abstract addresses to pointer interfaces. Our SSL reasoning and results generalise to arbitrary structured data libraries, as we report in [18]. Our work concentrates on the abstract connectivity associated with our SSL reasoning. However, the gap in reasoning occurs whenever there is a difference between abstract and concrete connectivity.

References

[1] Dinsdale-Young, T., M. Dodds, P. Gardner, M. Parkinson and V. Vafeiadis, “Concurrent Abstract Predicates”, ECOOP (2010), 504–528

[2] Wright, A. D., “Structural Separation Logic”, PhD thesis, Imperial College London, 2013.

[3] Wheelhouse, M. J., “Segment Logic”, PhD thesis, Imperial College London, 2011.

[4] Smith, G. D., “Local Reasoning about Web Programs”, PhD thesis, Imperial College London, 2011.

[5] Dinsdale-Young, T., “Abstract Data and Local Reasoning”, PhD thesis, Imperial College London, 2010

[6] Dinsdale-Young, T., L. Birkedal, P. Gardner, M. Parkinson and H. Yang, “Views: Compositional Reasoning for Concurrent Programs”, POPL (2013), 287–300

[7] Calcagno, C., P. Gardner and U. Zarfaty, “Context logic and Tree Update”, POPL (2005), 271–282

[8] Filipovic, I., P. O’Hearn, N. Torp-Smith and H. Yang, *Blaming the Client: on Data Refinement in the Presence of Pointers*, Formal Aspects of Computing **22** (2010), 547–583.

[9] Dinsdale-Young, T., P. Gardner, M. Wheelhouse, “Abstraction and Refinement for Local Reasoning”, VSTTE (2010), 199–215.

[10] Calcagno, C., P. O’Hearn and H. Yang, “Local Action and Abstract Separation Logic”, LICS (2007), 366–378.

[11] Bornat, R., C. Calcagno, P. O’Hearn and M. Parkinson, “Permission Accounting in Separation Logic”, POPL (2005), 259–270.

[12] Gardner, P., G. Smith, M. Wheelhouse and U. Zarfaty, “Local Hoare Reasoning about DOM”, PODS (2008), 261–270.

[13] Jensen, J. and L. Birkedal, “Fictional Separation Logic”, ESOP (2012), 377–396.

[14] Bornat, R., C. Calcagno and H. Yang, *Variables As Resource in Separation Logic*, Electronic Notes in Theoretical Computer Science **155** (2006), 247–276.

[15] Turon, A., D. Dreyer and L. Birkedal, *Unifying Refinement and Hoare-Style Reasoning in a Logic for Higher-Order Concurrency.*

[16] Svendsen, K., L. Birkedal and M. Parkinson, *Impredicative Concurrent Abstract Predicates*, ESOP2014.

[17] Svendsen, K, L. Birkedal and M. Parkinson, “Higher-order Concurrent Abstract Predicates”, Technical report, IT University of Copenhagen (2012).

[18] Raad, A., P. Gardner, A. Wright, M. Wheelhouse, “Abstract Local Reasoning for Concurrent Libraries (Technical Report)”, `http://www.doc.ic.ac.uk/~azalea/MFPS2014/TechnicalReport.pdf`.

[19] Gardner P., G. Ntzik and A. Wrigth, *Local Reasoning for POSIX File System*, ESOP (2014), 169–188.

[20] P. O’Hearn, *Resources, Concurrency and Local Reasoning*, Theor. Comput. Sci., **375** (2007), 271–307.

QRB, QFS, and the Probabilistic Powerdomain

Jean Goubault-Larrecq[1,2]

LSV, ENS Cachan, CNRS, INRIA
ENS Cachan
61 avenue du président Wilson
94230 Cachan, France

Achim Jung[1,3]

School of Computer Science
University of Birmingham
Birmingham, United Kingdom

Abstract

We show that the first author's QRB-domains coincide with Li and Xu's QFS-domains, and also with Lawson-compact quasi-continuous dcpos, with stably-compact locally finitary compact spaces, with sober QFS-spaces, and with sober QRB-spaces. The first three coincidences were discovered independently by Lawson and Xi. The equivalence with sober QFS-spaces is then applied to give a novel, direct proof that the probabilistic powerdomain of a QRB-domain is a QRB-domain. This improves upon a previous, similar result, which was limited to pointed, second-countable QRB-domains.

Keywords: QRB-spaces, QFS-spaces, QRB-domains, QFS-domains, stably compact spaces, probabilistic powerdomain

1 Introduction

An outstanding problem in denotational semantics is whether there is a full subcategory of continuous dcpos that is both Cartesian-closed and closed under the action of the probabilistic powerdomain monad $\mathcal{V}$ [17]. Indeed, there are very few categories of dcpos that are known to be closed under $\mathcal{V}$: the category of all dcpos, that of all continuous dcpos [15], and that of all Lawson compact continuous dcpos [17]. To that list, one must add the pointed, second-countable QRB-domains [10].

[1] We are grateful for the visiting professorship provided by ENS Cachan to the second author in March/April 2013, during which time the work reported here was begun.

[2] Email: goubault@lsv.ens-cachan.fr

[3] Email: A.Jung@cs.bham.ac.uk

While QRB-domains are only quasi-continuous and not continuous domains, and do not form a Cartesian-closed category either, they have attracted considerable attention recently.

QRB-domains are defined by imitating RB-domains. Independently, Li and Xu used a similar process to define QFS-domains [23], imitating the construction of FS-domains [16]. Rather surprisingly, QRB and QFS-domains are the same thing (RB and FS-domains are not known to coincide), and are also exactly the Lawson-compact quasi-continuous domains. This was shown independently by the present authors and J. Lawson and X. Xi. We present our proof in Section 5 below; Lawson's and Xi's proof will appear as [21].

One of our characterizations of QRB is as so-called sober QFS-spaces, and this will turn out to be instrumental in proving that the category of all QRB-domains, and not just the second-countable ones, is closed under the action of the probabilistic powerdomain, as we shall see in Section 6. This improves upon [10], and relies on a rather different proof argument.

Outline. After some brief preliminaries (Section 2), we discuss the notion of functional approximation in Section 3. This is a central concept in domain theory, at the heart of RB-, FS-, QRB-, and QFS-domains. Another domain-theoretic leitmotiv is that one should always topologize (paraphrasing M. Stone), and we introduce QFS-*spaces* in Section 4 as the natural topological counterpart of QFS-domains. We give our proof that QRB-domains and QFS-domains are the same thing (and coincide with four other natural notions, including sober QFS-spaces) in Section 5. We apply this to the promised result that the probabilistic powerdomain of a QRB-domain is a QRB-domain in Section 6.

2 Preliminaries

We refer to the classic texts [6,1] for the required domain-theoretic background, and to [11] for topology.

We agree that a subset of a space is compact if and only if every open cover has a finite subcover, that is, we do not require separation. We take coherence to mean that the intersection of any two compact saturated subsets is compact. (A saturated subset is one that is equal to the intersection of its open neighborhoods.) A space is *stably compact* if it is sober, compact, locally compact and coherent. As is well-known, the patch topology of a stably compact space is compact Hausdorff, see [11, Section 9] or [6, Section VI-6] for more details.

Any sober space is *well-filtered*, meaning that if an open subset U contains a filtered intersection $\bigcap_{i\in I} Q_i$ of compact saturated subsets, then U contains Q_i for some $i \in I$. In a well-filtered space, every such filtered intersection is compact.

Given a T_0 topological space $(X;\tau)$, we will make heavy use of its *specialisation order* defined as $x \leq y$ if $x \in \overline{\{y\}}$. We write $\uparrow E$ for the upward closure (w.r.t. $\leq$) of a subset E. Subsets equal to their upward closure are exactly the saturated one. If E is finite, then $\uparrow E$ is compact, and we call such sets the *finitary compacts* of X.

The set of compact saturated subsets of a topological space $(X;\tau)$ may be equipped with an order by setting $A \leq B$ iff $A \supseteq B$, and we write $\mathcal{Q}(X)$ for the resulting poset. It may also be equipped with the *upper Vietoris* topology, which has a base of opens of the form $\Box U$, $U \in \tau$, where $\Box U$ denotes the collection of compact saturated subsets contained in U. This yields the *upper space* $\mathcal{Q}_{\mathsf{V}}(X)$ of X. Happily, the specialisation order of the upper space is precisely reverse inclusion. Analogously, We write $\mathrm{Fin}(X)$ for the collection of finitary compacts of X, and topologize it with the subspace topology, yielding a space that we write $\mathrm{Fin}_{\mathsf{V}}(X)$. When X is well-filtered (e.g., sober), $\mathcal{Q}(X)$ is a dcpo and directed suprema are computed as intersections.

For a finite subset E of a poset $(X, \leq)$ and $x \in X$, write $E \ll x$ iff every directed family $(x_i)_{i \in I}$ whose supremum $\sup_{i \in I} x_i$ is above x in X contains an element x_i that is above some element z of E. We also write $\uparrow E \ll x$ instead of $E \ll x$, stressing the fact that this is a property of the finitary compact $\uparrow E$, not just of the finite set E. The dcpo X is a *quasi-continuous domain* (see [7] or [6, Definition III-3.2]) if and only if for every $x \in X$, the collection of all $\uparrow E \in \mathrm{Fin}(X)$ that approximate x ($\uparrow E \ll x$) is directed (w.r.t. $\supseteq$) and their least upper bound in $\mathcal{Q}(X)$ is $\uparrow x$.

3 Functional approximation

We are concerned with spaces in which points are "systematically" approximated, by which we mean that we are given functions which produce approximants for each element. In domain theory, the idea goes back to Plotkin's characterization of SFP-domains, [24], as those dcpos X for which there is a chain of Scott-continuous functions $(\varphi_n)_{n \in \mathbb{N}}$ from X to X, satisfying the following properties

(i) for each $n \in \mathbb{N}$, $\varphi_n \leq \mathsf{id}_X$;

(ii) for each $n \in \mathbb{N}$, $\varphi_n \circ \varphi_n = \varphi_n$;

(iii) for each $n \in \mathbb{N}$, φ_n has finite image;

(iv) $\mathsf{id}_X = \bigvee^{\uparrow}_{n \in \mathbb{N}} \varphi_n$.

Plotkin also showed that the retracts of SFP-domains can be characterised similarly, by dropping the idempotency requirement (ii). If instead of a chain, only a directed family of such functions is present, then one obtains *RB-domains*. The concept was further generalized in the work of the second author, [16], where instead of requiring finite image, *finite separation* is stipulated: A function $\varphi\colon X \to X$ is *finitely separated* from id_X if there exists a finite set $M \subseteq X$ such that

$$\forall x \in X.\ \exists m \in M.\ \varphi(x) \leq m \leq x.$$

An FS-domain, then, is a dcpo which contains a directed family $(\varphi_i)_{i \in I}$ of continuous functions finitely separated from identity such that $\mathsf{id}_X = \bigvee^{\uparrow}_{i \in I} \varphi_i$.

In 2010 the first author, [9] realised that the concept of functional approximation could usefully be further generalized by allowing the approximating functions to produce compact neighborhoods rather than points, that is, the φ_i now take values in $\mathcal{Q}_{\mathsf{V}}(X)$ rather than X. With this generalization there are then two choices to be

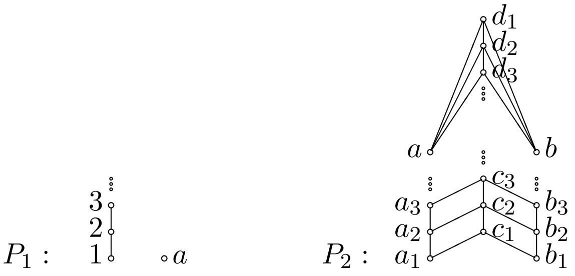


Fig. 1. Two example spaces

made about their *finiteness character*:

Choice 1 One can require the φ_i to have finite image in $\mathcal{Q}_{\mathsf{V}}(X)$ or not make any such restriction.

Choice 2 One can require the φ_i to only produce finitary compacts or allow general compact saturated sets.

Together this means that there are four variants that one might consider and it may come as a relief to the reader that they will in fact all turn out to lead to the same structures. Specifically, we will show that the most liberal notion, arbitrary image of general compact saturated sets, and the most restrictive one, finite image of finitary compacts, coincide. This will be true with and without assuming sobriety.

Definition 3.1 A continuous function $\varphi\colon X \to \mathcal{Q}_{\mathsf{V}}(X)$ is called a *quasi-deflation* if it is has finite image and for each $x \in X$, $x \in \varphi(x) \in \operatorname{Fin} X$. It is called *quasi-finitely separated* (or *qfs* for short) if there exists a finite set $M \subseteq X$ such that for every $x \in X$ there is $m \in M$ such that $x \in {\uparrow} m \subseteq \varphi(x)$. In this case, we say that φ is *separated by* M, or that M is a *separating set* for φ.

We shall agree to order continuous maps from X to $\mathcal{Q}_{\mathsf{V}}(X)$ in the pointwise extension of $\supseteq$. Accordingly, a family $(\varphi_i)_{i\in I}$ of continuous functions from X to $\mathcal{Q}_{\mathsf{V}}(X)$ is directed if and only if it is non-empty and for all $i, j \in I$, there is a $k \in I$ such that, for every $x \in X$, $\varphi_k(x) \subseteq \varphi_i(x), \varphi_j(x)$. We call it *approximating* if it is directed and furthermore, ${\uparrow} x = \bigcap_{i\in I} \varphi_i(x)$ holds for all $x \in X$.

We call a T_0 topological space $(X;\tau)$ a *QRB-space* if there is an approximating family of quasi-deflations for it. It is called a *QFS-space* if there is an approximating family of quasi-finitely separated maps.

A QFS- (or QRB-) space $(X;\tau;(\varphi_i)_{i\in I})$ is called *topological* if for all $U \in \tau$ and $x \in U$ there is $i \in I$ such that $\varphi_i(x) \subseteq U$.

Clearly, every quasi-deflation φ is also qfs because we can take the finitely many minimal elements of the finitely many possible images of φ as the separating set. Therefore, every QRB-space is also QFS. To explain the last part of the definition, we give an example to show that not every QFS-space is topological:

Example 3.2 Consider the poset P_1 in Figure 1 consisting of the natural numbers in their usual order plus an extra element a not related to any of the others. Equip this set with the Alexandroff topology (of all upper sets) and consider the map φ_m

which maps each $n \in \mathbb{N}$ to $\uparrow \min\{m, n\}$ and a to $\uparrow m \cup \{a\}$. Clearly, each φ_m is a quasi-deflation. Furthermore, the family $(\varphi_m)_{m \in \mathbb{N}}$ is approximating and thus P_1 is a QRB-space. However, for no $m \in \mathbb{N}$ do we have that $\varphi_m(a) \subseteq \uparrow a = \{a\}$.

4 QFS-spaces

Proposition 4.1 *QFS-spaces are compact.*

Proof. Let X be a QFS-space and φ be any qfs map on X with separating set M. Then $X = \uparrow M$ since every $x \in X$ is above some $m \in M$ by definition. Since M is finite, we have compactness. □

For local compactness we start with a useful lemma:

Lemma 4.2 *Let φ be a qfs map on a topological space X, separated by the finite set M. Then for every $x \in X$, x is in the interior of $\uparrow(M \cap \varphi(x))$.*

Proof. Fix $x \in X$ and let $U = X \setminus \downarrow(M \setminus \varphi(x))$. Because of the finiteness of M, U is an open set and a neighborhood of $\varphi(x)$. Let $V = \varphi^{-1}(\Box U)$, which is an open set since φ is continuous. By construction, x is a member of V and, furthermore, we claim that $V \subseteq \uparrow(M \cap \varphi(x))$. Indeed, let $y \in V$. Then $\varphi(y) \subseteq U$ and hence the separating element $m \in M$ with $m \in \varphi(y)$ and $m \leq y$ also belongs to U. Hence $m \in M \cap \varphi(x)$ and $y \in \uparrow(M \cap \varphi(x))$ follows. □

A topological space X is *locally finitary compact* if every open neighborhood U of an arbitrary point x contains a locally finitary neighborhood $\uparrow E$ of x: $U \supseteq \uparrow E \supseteq \mathrm{int}(\uparrow E) \ni x$. The notion originates with Isbell [14], and the T_0 such spaces are called qc-spaces in [21]. Every quasi-continuous domain is locally finitary compact, since in this case $\mathrm{int}(\uparrow E) = \{x \in X \mid E \ll x\}$ [6, III-3.6(ii)]. The following is immediate from the definitions and the preceding lemma:

Lemma 4.3 *Every topological QFS-space is locally finitary compact.*

It would be nice if one could also show coherence for QFS-spaces but without further assumptions this is not possible, even for QRB-spaces:

Example 4.4 Consider the poset P_2 in Figure 1 together with the Scott topology (note that the only non-trivial directed suprema are $a = \bigvee^{\uparrow}_{n \in \mathbb{N}} a_n$ and $b = \bigvee^{\uparrow}_{n \in \mathbb{N}} b_n$). The QRB property is established by maps f_m, $m \in \mathbb{N}$, which map

$$\begin{array}{llll} a \mapsto a_m & b \mapsto b_m & c_n \mapsto c_{\min\{m,n\}} & d_n \mapsto c_m \text{ for } n > m \\ a_n \mapsto a_{\min\{m,n\}} & b_n \mapsto b_{\min\{m,n\}} & & d_n \mapsto d_n \text{ for } n \leq m \end{array}$$

and by setting $\varphi_m(x) = \uparrow f_m(x)$. The resulting QRB-space is topological because every Scott neighborhood of a (resp. b) must contain some final segment of a_n's (resp. b_n's). It is not coherent, though, because $\uparrow a \cap \uparrow b = \{d_n \mid n \in \mathbb{N}\}$ is not compact.

The situation is much nicer if we assume our spaces to be sober. First, since sobriety implies well-filteredness, we immediately have the following:

Lemma 4.5 *Sober QFS-spaces are topological.*

Combining the last two lemmas we get that sober QFS-spaces are locally finitary compact, and it is known from [3], or the equivalence between (6) and (11) in [22, Theorem 2], or [21, Corollary 3.6], or [11, Exercise 8.3.39], that the sober, locally finitary compact spaces are exactly the quasi-continuous dcpos in their Scott topology. Thus we have:

Proposition 4.6 *Sober QFS-spaces are quasi-continuous domains, and their given topology coincides with the Scott topology derived from the specialisation order.*

Thus it is appropriate to call sober QFS-spaces, *QFS-domains*, and similarly for sober QRB-spaces. A little amount of work should convince the reader that these QFS-domains are exactly the same of those defined by Li and Xu [23].

How far are (topological) QFS-spaces from QFS-domains? As it turns out, not very far as we will now show that sobrification leads from one to the other.

The sobrification $\hat{X}$ of a topological space $(X;\tau)$ can be described in a number of ways; the most convenient for our purposes is to realise it concretely as the set of closed irreducible[4] subsets of X, together with the topology $\hat{\tau}$ which consists of open sets $\hat{U} = \{A \in \hat{X} \mid A \cap U \neq \emptyset\}$, where U ranges over the open sets in τ. Note that X and $\hat{X}$ have isomorphic frames of opens.

Given a qfs map $\varphi\colon X \to \mathcal{Q}_{\mathsf{V}}(X)$ we replace $\varphi(x)$ with its set of open neighborhoods, defined as $\{U \in \tau \mid \varphi(x) \subseteq U\}$. This is always a Scott-open filter in the frame τ, and the Hofmann-Mislove Theorem tells us that, conversely, every Scott-open filter $\mathcal{F}$ of τ corresponds to a unique compact saturated set $Q_{\mathcal{F}}$ of the sobrification $\hat{X}$ of X. Indeed, $\mathcal{F}$ consists precisely of the opens U such that $\hat{U}$ is a neighborhood of $Q_{\mathcal{F}}$, that is, a closed irreducible set belongs to $Q_{\mathcal{F}}$ if and only if it meets every member of $\mathcal{F}$. For the upper Vietoris topology on $\mathcal{Q}_{\mathsf{V}}(\hat{X})$, the basic open set $\Box\hat{U}$ consists of those compacts $Q_{\mathcal{F}}$ where $\mathcal{F}$ ranges over the Scott-open filters which contain U. Using this setup, we define $\tilde{\varphi}\colon \hat{X} \to \mathcal{Q}_{\mathsf{V}}(\hat{X})$ by mapping $A \in \hat{X}$ to $Q_{\psi(A)}$ where $\psi(A) = \{U \in \tau \mid \exists a \in A.\ \varphi(a) \subseteq U\} = \{U \in \tau \mid A \cap \varphi^{-1}(\Box U) \neq \emptyset\}$.

Lemma 4.7 *For $\varphi\colon X \to \mathcal{Q}_{\mathsf{V}}(X)$ qfs, $\tilde{\varphi}$ is a qfs map.*

Proof. The function $\tilde{\varphi}$, equivalently ψ, is well-defined: if a directed union of opens belongs to $\psi(A)$ then it covers $\varphi(a)$ for some $a \in A$. Because $\varphi(a)$ is compact, one of them does so already. Filteredness follows from $\varphi^{-1}(\Box U \cap \Box V) = \varphi^{-1}(\Box(U \cap V)) = \varphi^{-1}(\Box U) \cap \varphi^{-1}(\Box V)$ and the assumption that A is irreducible.

For continuity, observe that $\tilde{\varphi}^{-1}(\Box\hat{U}) = \psi^{-1}(\{\mathcal{F} \mid U \in \mathcal{F}\}) = \{A \in \hat{X} \mid A \cap \varphi^{-1}(\Box U) \neq \emptyset\} = \widehat{\varphi^{-1}(\Box U)}$.

For finite separation, we assume that M is a separating set for φ. We show that the set $\hat{M} = \{\downarrow m \mid m \in M\}$ is separating for $\tilde{\varphi}$. Let A be a closed irreducible set.

[4] A set is irreducible if it meets every member of a finite family of open sets precisely if it meets their intersection (from which it follows that irreducible sets are non-empty).

For every $U \in \psi(A)$ we have by definition that there is $a \in A$ such that $\varphi(a) \subseteq U$. It follows that $U \cap (M \cap A)$ is non-empty. Hence the family of these sets, indexed by $U \in \psi(A)$, is a proper filter on the finite set $M \cap A$ and so there is $m_A \in M$ belonging to all of them. We clearly have that $\downarrow m_A \subseteq A$ and because m_A is in every $U \in \psi(A)$, $\downarrow m_A$ meets every element of $\psi(A)$, whence $\downarrow m_A \in Q_{\psi(A)} = \tilde{\varphi}(A)$. □

The above construction has been chosen for its brevity but we may point out that the underlying idea relies on a natural transformation T (a "distributive law") from $[\hat{-}] \circ \mathcal{Q}_{\mathsf{V}}$ to $\mathcal{Q}_{\mathsf{V}} \circ [\hat{-}]$. Our map $\tilde{\varphi}$ is the composition $\hat{X} \xrightarrow{\widehat{\varphi}} \widehat{\mathcal{Q}_{\mathsf{V}}(X)} \xrightarrow{T} \mathcal{Q}_{\mathsf{V}}(\hat{X})$. An even more explicit construction is also possible, and it demonstrates nicely the usefulness of the "Topological Rudin Lemma" presented in [13]: We invite the reader to use the latter to show that $T(C) = \{A \in \hat{X} \mid \forall Q \in C.\ Q \cap A \neq \emptyset\}$, and to also use it to reprove Lemma 4.7 with that definition.

Finally, we would like to show that the lifted family $(\tilde{\varphi}_i)_{i \in I}$ is approximating for $\hat{X}$. It is here where we need the condition that the original QFS space be topological, as without this condition this would not be the case. Consider again Example 3.2: The sobrification of the space P_1 consists of the sets $\downarrow x$, $x \in X$ plus one more, the chain $A = \mathbb{N}$. By definition, A belongs to each $\tilde{\varphi}_m(\downarrow a)$: check that, for every $a' \leq a$, A meets every open neighborhood U of $\varphi_m(a')$. Hence A is also in the intersection of all $\tilde{\varphi}_m(\downarrow a)$, but it does not belong to $\uparrow(\downarrow a) = \{\{a\}\}$.

We come to the main result of this section:

Theorem 4.8 *The sobrification of a topological QFS space is a QFS domain.*

Proof. All that remains is to show that the family $(\tilde{\varphi}_i)_{i \in I}$ is approximating for $\hat{X}$. Let $A \in \hat{X}$ be a closed irreducible subset of X and let B be another such, not above A. This means that B does not contain A (as subsets of X), and so let $a \in A \smallsetminus B$. By the definition of topological QFS spaces we obtain an index $i \in I$ such that $\varphi_i(a)$ is contained in the open set $X \smallsetminus B$. Writing $\psi_i(A)$ for $\{U \in \tau \mid \exists a \in A.\ \varphi_i(a) \subseteq U\}$, so that $\tilde{\varphi}_i(A) = Q_{\psi_i(A)}$, we obtain that $U \in \psi_i(A)$ for $U = X \smallsetminus B$. Since U does not meet B, B is not in $\tilde{\varphi}_i(A)$. □

5 QFS-domains

We have already seen that the addition of sobriety to the conditions for a QFS-space results in much nicer structures. The best is still to come, however. We begin by giving a short argument to show that QFS-domains are coherent, a result which appears as Corollary 3.9 in [21]. First a lemma, also from [21]:

Lemma 5.1 *If X is a QFS-domain then $\mathcal{Q}_{\mathsf{V}}(X)$ is an FS-domain.*

Proof. If φ is a qfs map on X separated by M, then $\Phi \colon \mathcal{Q}_{\mathsf{V}}(X) \to \mathcal{Q}_{\mathsf{V}}(X)$, defined by $\Phi(K) = \uparrow \varphi[K]$, is finitely separated: For the separating set consider all sets $\uparrow E$, $E \subseteq M$. □

Proposition 5.2 *The topology of a QFS-domain is coherent.*

Proof. Let K, L be compact saturated sets of X. They are points in $\mathcal{Q}_{\mathsf{V}}(X)$ and generate principal upper, hence compact, sets $\uparrow_{\mathcal{Q}_{\mathsf{V}}(X)} K$ and $\uparrow_{\mathcal{Q}_{\mathsf{V}}(X)} L$. Since $\mathcal{Q}_{\mathsf{V}}(X)$ is an FS-domain, it is coherent, hence the set $\mathcal{K} = \uparrow_{\mathcal{Q}_{\mathsf{V}}(X)} K \cap \uparrow_{\mathcal{Q}_{\mathsf{V}}(X)} L$ is a compact saturated set. The claim follows from the observation that $K \cap L = \bigcup \mathcal{K}$ and the fact that $\bigcup$, as the multiplication of the upper powerspace monad ([25, Chapter 7]), is a continuous map from $\mathcal{Q}_{\mathsf{V}}(\mathcal{Q}_{\mathsf{V}}(X))$ to $\mathcal{Q}_{\mathsf{V}}(X)$. □

For quasi-continuous domains, compactness plus coherence is the same as compactness in the Lawson topology. This follows, for example, from the fact that the Lawson and patch topologies coincide on quasi-continuous dcpos [6, Lemma V-5.15], and that every patch-compact space is coherent and compact [11, Proposition 9.1.27], while conversely quasi-continuous domains are locally compact and sober [11, Exercise 8.2.15]. We thus have the following refinement of Proposition 4.6:

Corollary 5.3 *QFS-domains are Lawson-compact quasi-continuous domains equipped with their Scott topology.*

We now work towards the converse of this:

Proposition 5.4 *Every compact, locally compact, coherent space X has an approximating family of maps $\varphi_{\mathcal{M}} \colon X \to \mathcal{Q}_{\mathsf{V}}(X)$ with finite image. Precisely, $\mathcal{M}$ ranges over the finite $\vee$-semi-lattices $\mathcal{M}$ of compact saturated sets of X, and $\varphi_{\mathcal{M}}$ maps each $x \in X$ to the smallest element of $\mathcal{M}$ whose interior contains x.*

Note that $\varphi_{\mathcal{M}}$ takes values in $\mathcal{Q}(X)$, not in $\mathrm{Fin}(X)$. Smallest is taken with respect to inclusion. A $\vee$*-semi-lattice of compact saturated sets* is a family of sets that is closed under finite intersections (in particular, contains X).

Proof. Define $\varphi_{\mathcal{M}}(x)$ as the intersection of all the elements Q of $\mathcal{M}$ that are neighborhoods of x. Using the fact that $\mathcal{M}$ is finite, $\varphi_{\mathcal{M}}(x)$ is the smallest neighborhood of x in $\mathcal{M}$, so $\varphi_{\mathcal{M}}(x)$ is well defined, and in $\mathcal{Q}(X)$ by coherence and compactness.

For continuity, let U be open and consider $x \in \varphi_{\mathcal{M}}^{-1}(\Box U)$. Let $Q = \varphi_{\mathcal{M}}(x)$. For every $y \in \mathrm{int}(Q)$, $\varphi_{\mathcal{M}}(y) \subseteq Q \subseteq U$, so y is in $\varphi_{\mathcal{M}}^{-1}(\Box U)$. Hence $\mathrm{int}(Q)$ is an open neighborhood of x included in $\varphi_{\mathcal{M}}^{-1}(\Box U)$, so $\varphi_{\mathcal{M}}^{-1}(\Box U)$ is open.

Clearly, if $\mathcal{M} \subseteq \mathcal{M}'$, then $\varphi_{\mathcal{M}}(x) \supseteq \varphi_{\mathcal{M}'}(x)$ for every $x \in X$. The family of all $\varphi_{\mathcal{M}}$ is directed: given $\mathcal{M}$ and $\mathcal{M}'$, there is a smallest semi-lattice $\mathcal{M} \sqcup \mathcal{M}'$ of compact saturated sets containing $\mathcal{M}$ and $\mathcal{M}'$, consisting of the intersections $Q \cap Q'$ with $Q \in \mathcal{M}$ and $Q' \in \mathcal{M}'$; coherence implies that each such $Q \cap Q'$ is compact saturated, and $\varphi_{\mathcal{M} \sqcup \mathcal{M}'}$ is above both $\varphi_{\mathcal{M}}$ and $\varphi_{\mathcal{M}'}$ (w.r.t. $\supseteq$).

All that remains to show is that the maps $\varphi_{\mathcal{M}}$ form an approximating family. Given $x \in X$, $\uparrow x \subseteq \bigcap_{\mathcal{M}} \varphi_{\mathcal{M}}(x)$ is by definition. For the reverse inclusion, we show that every open neighborhood U of x contains $\bigcap_{\mathcal{M}} \varphi_{\mathcal{M}}(x)$. By local compactness, U contains a compact saturated neighborhood Q of x. $\mathcal{M} = \{Q, X\}$ qualifies as a semi-lattice of compact saturated sets, and we have $\varphi_{\mathcal{M}}(x) = Q \subseteq U$. □

The following is standard:

Lemma 5.5 *Let X be a locally finitary compact space. For every compact saturated subset Q of X, and every open neighborhood U of Q, there is a further, finitary compact neighborhood $\uparrow E$ of Q contained in U.*

Proposition 5.6 *Every compact, locally finitary compact, coherent space X has an approximating family of quasi-deflations.*

Proof. Applying Proposition 5.4, we obtain an approximating family of maps $\varphi_{\mathcal{M}}$. We need to replace each compact saturated subset $Q \in \mathrm{im}\,\varphi_{\mathcal{M}}$ by a finitary compact.

Assume first that we are given an open neighborhood U_Q around each of them. Lemma 5.5 allows us to find finitary compact neighborhoods $\uparrow E_Q$ between Q and U_Q. We seek to find $\uparrow E_Q$ so that, additionally, $Q \subseteq Q'$ implies $\uparrow E_Q \subseteq \uparrow E_{Q'}$. To ensure this, we define $\uparrow E_Q$ step by step, always working on the largest $Q \in \mathcal{M}$ that is still to be considered (so we start with X itself, the largest element of $\mathcal{M}$). Given any $Q \in \mathcal{M}$ such that $\uparrow E_{Q'}$ is already defined for every strictly larger $Q' \in \mathcal{M}$, we apply Lemma 5.5 and define $\uparrow E_{Q'}$ as some finitary compact neighborhood of Q contained in $U_Q \cap \bigcap_{\substack{Q' \in \mathcal{M} \\ Q' \supsetneq Q}} \mathrm{int}(\uparrow E_Q)$.

We now replace each $Q \in \mathrm{im}\,\varphi_{\mathcal{M}}$ by the so chosen $\uparrow E_Q$, resulting in a function $\psi_{\mathcal{M},\mathcal{E},\mathcal{U}}$, where $\mathcal{U}$ is the collection of open neighborhoods U_Q we started with, and $\mathcal{E}$ is the collection of finitary compacts $\uparrow E_Q$. We need to check that $\psi_{\mathcal{M},\mathcal{E},\mathcal{U}}$ is continuous, and for that we check that $\psi^{-1}_{\mathcal{M},\mathcal{E},\mathcal{U}}(\Box U)$ is open for every open subset U of X. Let x be an element of $\psi^{-1}_{\mathcal{M},\mathcal{E},\mathcal{U}}(\Box U)$, and $Q = \varphi_{\mathcal{M}}(x)$; in particular, $\uparrow E_Q \subseteq U$. As in the proof of Proposition 5.4, every element y of $\mathrm{int}(Q)$ is such that $\varphi_{\mathcal{M}}(y) \subseteq Q$; for $Q' = \varphi_{\mathcal{M}}(y)$, $Q' \subseteq Q$ implies $\uparrow E_{Q'} \subseteq \uparrow E_Q$, so $\psi_{\mathcal{M},\mathcal{E},\mathcal{U}}(y) \subseteq \uparrow E_Q \subseteq U$. Therefore $\mathrm{int}(Q)$ is an open neighborhood of x included in $\psi^{-1}_{\mathcal{M},\mathcal{E},\mathcal{U}}(\Box U)$.

The family of all maps $\psi_{\mathcal{M},\mathcal{E},\mathcal{U}}$ (namely, with $\mathcal{E} = (\uparrow E_Q)_{Q \in \mathcal{M}}$ monotone, $\mathcal{U} = (U_Q)_{Q \in \mathcal{M}}$, and $Q \subseteq \mathrm{int}(\uparrow E_Q) \subseteq \uparrow E_Q \subseteq U_Q$ for each $Q \in \mathcal{M}$) is approximating, since we can choose the initial neighborhoods U_Q as close to each $Q \in \mathcal{M}$ as we like, and it remains to show that it is directed. It is non-empty: choose $\mathcal{M} = \{X\}$ and $\mathcal{U} = \mathcal{M}$, and define $\uparrow E_X$ as X itself, which is finitary compact as a consequence of Lemma 5.5 with $Q = U = X$. We find an upper bound of $\psi_{\mathcal{M},\mathcal{E},\mathcal{U}}$ and $\psi_{\mathcal{M}',\mathcal{E}',\mathcal{U}'}$ by defining $\mathcal{N} = \mathcal{M} \sqcup \mathcal{M}'$, and for the open neighborhood system $\mathcal{V}$ we let $V_N = \bigcap\{\mathrm{int}(\uparrow E_Q) \mid N \subseteq Q \in \mathcal{M}\} \cap \bigcap\{\mathrm{int}(\uparrow E'_{Q'}) \mid N \subseteq Q' \in \mathcal{M}'\}$ for each $N \in \mathcal{N}$. (We write $\mathcal{E} = (\uparrow E_Q)_{Q \in \mathcal{M}}$, $\mathcal{E}' = (\uparrow E'_{Q'})_{Q' \in \mathcal{M}'}$.) It is clear that $\psi_{\mathcal{N},\mathcal{F},\mathcal{V}}$ is above $\psi_{\mathcal{M},\mathcal{E},\mathcal{U}}$ and $\psi_{\mathcal{M}',\mathcal{E}',\mathcal{U}'}$. □

Theorem 5.7 *Let X be a topological space. The following are equivalent:*

(i) *X is a stably compact, locally finitary compact space.*

(ii) *X is a sober QRB-space.*

(iii) *X is a sober QFS-space.*

(iv) *X is a QRB-domain with its Scott topology.*

(v) *X is a QFS-domain with its Scott topology.*

(vi) *X is a Lawson-compact quasi-continuous dcpo in its Scott topology.*

(vii) *X is a compact, coherent, quasi-continuous dcpo in its Scott topology.*

Proof. $(i) \Rightarrow (ii)$: X is a QRB-space by Proposition 5.6, and sober since stably-compact. $(ii) \Rightarrow (iii)$ and $(iv) \Rightarrow (v)$ are obvious. $(iii) \Rightarrow (v)$ is Proposition 4.6, which also implies $(ii) \Rightarrow (iv)$ since QRB-spaces are instances of QFS-spaces.

$(v) \Rightarrow (vi)$. Every QFS-domain is quasi-continuous [23, Proposition 3.8], and Lawson-compact [23, Theorem 4.9].

$(vi) \Rightarrow (vii)$. For quasi-continuous domains, compactness plus coherence is the same as compactness in the Lawson topology.

$(vii) \Rightarrow (i)$. Every quasi-continuous dcpo is sober [11, Exercise 8.2.15] and locally finitary compact [11, Exercise 5.2.31]. With compactness and coherence, this implies that X is stably-compact. □

Lawson and Xi's result mentioned in the introduction [21] is the equivalence $(iv) \Leftrightarrow (v) \Leftrightarrow (vi)$ above. Items (i)–(iii) offer other, purely topological characterizations of QRB-domains.

Returning to the topological beginnings of our paper, we note the following:

Theorem 5.8 *Topological QFS-spaces are QRB.*

Proof. Let $(X; \tau; (\varphi_i)_{i\in I})$ be a topological QFS-space. Then its sobrification $\hat{X}$ is a QFS-domain and so by the preceding theorem, a QRB-domain. Looking at the proof of Proposition 5.6 we find that we constructed the finitary compacts by invoking Lemma 5.5, so we should have a closer look at that in the case that we are dealing with a locally finitary compact space that is a sobrification. In that case, every element e of E is a closed irreducible set A that meets the open set $U \in \tau$. We may therefore replace e with the irreducible set $\downarrow a$ where a is an arbitrarily chosen element of $A \cap U$. In summary, then, we can make sure that the elements employed in the proof of Proposition 5.6 all stem from the image of the embedding $x \mapsto \downarrow x$ of X into its sobrification. □

6 The Probabilistic Powerdomain over QRB-Domains

Let us turn to the *probabilistic powerdomain* $\mathcal{V}(X)$ over a space X. This was introduced by Jones in her PhD thesis [15] to give semantics to higher-order programs with probabilistic choice. Jones proved that $\mathcal{V}(X)$ was a continuous dcpo for every continuous dcpo X, but also that $\mathcal{V}(X)$ was not a continuous lattice, or even a bc-domain even for very simple continuous lattices or bc-domains X. We still do not know whether $\mathcal{V}(X)$ is an FS-domain, resp. an RB-domain whenever X is one, except in very specific cases [17]. However, the notion of functional approximation offered by QRB-domains is relaxed enough that the probabilistic powerdomain of a QRB-domain is again a QRB-domain. The first author proved this [10], for probability valuations (with total mass 1), and assuming second-countability.

Using Theorem 5.7, we shall see that the latter is an irrelevant assumption. We shall also prove it for spaces of continuous subprobability valuations, and of general, unbounded, continuous valuations. The nature of the proof is very different from [10]: we build an approximating family of qfs maps on $\mathcal{V}(X)$, directly[5].

The elements of $\mathcal{V}(X)$ are a slight variation on the idea of a measure, and are called continuous valuations. A *continuous valuation* ν on a space X is a Scott-continuous, strict, modular map from the complete lattice $\mathcal{O}(X)$ of open subsets of X to $\overline{\mathbb{R}^+} = \mathbb{R} \cup \{+\infty\}$. Strictness means that $\nu(\emptyset) = 0$, modularity states that $\nu(U \cup V) + \nu(U \cap V) = \nu(U) + \nu(V)$ for all $U, V \in \mathcal{O}(X)$.

Let $\mathcal{V}(X)$, the probabilistic powerdomain over X, denote the space of all continuous valuations on X, with the weak topology. We also write $\mathcal{V}^1(X)$ for the subspace of continuous probability valuations ($\nu(X) = 1$) and $\mathcal{V}^{\leq 1}(X)$ for the subspace of continuous subprobability valuations ($\nu(X) \leq 1$). We shall write $\mathcal{V}^\bullet(X)$ for $\mathcal{V}(X)$, $\mathcal{V}^1(X)$, or $\mathcal{V}^{\leq 1}(X)$. The *weak topology* on $\mathcal{V}^\bullet(X)$ has subbasic open sets of the form $[U > r]$, defined as $\{\nu \in \mathcal{V}^\bullet(X) \mid \nu(U) > r\}$ [19, Satz 8.5] (see also [12, Theorem 8.3]). Whatever $\bullet$ is, $\mathcal{V}^\bullet$ is a functor on the category of topological spaces, and its action $\mathcal{V}f$ on continuous maps $f\colon X \to Y$ is defined by $\mathcal{V}f(\nu)(V) = \nu(f^{-1}(V))$.

We again introduce a "distributivity law" θ, this time from $\mathcal{V}^\bullet\mathcal{Q}_\mathsf{V}$ to $\mathcal{Q}_\mathsf{V}\mathcal{V}^\bullet$. Given $\mu \in \mathcal{V}^\bullet\mathcal{Q}_\mathsf{V}(X)$, one may define $\theta(\mu)$ as the set of all $\nu \in \mathcal{V}^\bullet(X)$ such that $\nu(U) \geq \mu(\Box U)$ for every open U. It is not completely trivial that $\theta(\mu)$ is non-empty and compact, or that θ is continuous, but let us accept it for the moment. We may use θ to produce maps $\mathcal{V}^\bullet(X) \xrightarrow{\mathcal{V}\varphi_i} \mathcal{V}^\bullet\mathcal{Q}_\mathsf{V}(X) \xrightarrow{\theta} \mathcal{Q}_\mathsf{V}\mathcal{V}^\bullet(X)$ for an approximating family of quasi-deflations φ_i on X. It will be fairly easy to see that the resulting maps are approximating, but they certainly do not have finite image, and even the image of a single $\nu \in \mathcal{V}^\bullet(X)$ is in general not finitary. However, and up to a minor variation on the theme of θ (θ_f, see below), we will manage to show that these maps are qfs. Hence $\mathcal{V}^\bullet(X)$ will be a QFS space, and the equivalence between (*iii*) and (*iv*) of Theorem 5.7 will allow us to conclude.

Lemma 6.1 *Let $(X;\tau)$ be a stably compact space. If $\bullet$ is "1", assume X pointed, too. For every Scott-continuous map $f\colon \overline{\mathbb{R}^+} \to \overline{\mathbb{R}^+}$ such that $f \leq \mathrm{id}_{\overline{\mathbb{R}^+}}$, the map θ_f defined by $\theta_f(\mu) = \{\nu \in \mathcal{V}^\bullet(X) \mid \forall U \in \tau.\ \nu(U) \geq f(\mu(\Box U))\}$ is a continuous and Scott-continuous map from $\mathcal{V}^\bullet(\mathcal{Q}_\mathsf{V}(X))$ to $\mathcal{Q}_\mathsf{V}(\mathcal{V}^\bullet(X))$.*

Proof. Let $[X \to \overline{\mathbb{R}^+}]$ denote the dcpo of Scott-continuous maps from X to $\overline{\mathbb{R}^+}$, in the pointwise ordering. For any monotonic set function ξ on the open subsets of X with values in $\overline{\mathbb{R}^+}$, and every $h \in [X \to \overline{\mathbb{R}^+}]$, one can define $\int_{x\in X} h(x)d\xi$ by the Choquet formula $\int_0^{+\infty} \xi(h^{-1}(t, +\infty])dt$, where the latter is a Riemann integral. For $\xi = \nu \in \mathcal{V}^\bullet(X)$, the functional $h \mapsto \int_{x\in X} h(x)d\nu$ is Scott-continuous and linear [26, Section 4]. Scott-continuity follows from the fact that Riemann integration of antitonic maps is itself Scott-continuous.

For $\xi(U) = \mu(\Box U)$, notice that $h_*(Q) = \min_{x\in Q} h(x)$ defines a continuous function of $Q \in \mathcal{Q}_\mathsf{V}(X)$, since $h_*^{-1}(t, +\infty] = \Box h^{-1}(t, +\infty]$ (by compactness, the inf of h is attained on Q), so that $\int_{x\in X} h(x)d\xi = \int_0^{+\infty} \mu(\Box h^{-1}(t, +\infty])dt = \int_0^{+\infty} \mu(h_*^{-1}(t,$

[5] This means proving the result using characterization 3 of QRB-domains given in Theorem 5.7. Characterization 1 may seem a better route, since $\mathcal{V}$ is already known to preserve stable compactness: only local finitary compactness remains to be proved. However, that seems quite a formidable effort by itself.

$+\infty])dt = \int_{Q\in\mathcal{Q}_{\mathsf{V}}(X)} h_*(Q)d\mu$. Since $(h_1+h_2)_* \geq h_{1*}+h_{2*}$, the functional $p\colon [X \to \overline{\mathbb{R}^+}] \to \overline{\mathbb{R}^+}$ that maps h to $\int_{Q\in\mathcal{Q}_{\mathsf{V}}(X)} h_*(Q)d\mu$ is superlinear, meaning that $p(h_1+h_2) \geq p(h_1)+p(h_2)$ and $p(ah) = ap(h)$ for every $a \in \mathbb{R}^+$. Since $p(h) = \int_{x\in X} h(x)d\xi = \int_0^{+\infty} \xi(h^{-1}(t,+\infty])dt$, p is also Scott-continuous in h.

$\theta_f(\mu)$ is non-empty. Define $q\colon [X \to \overline{\mathbb{R}^+}] \to \overline{\mathbb{R}^+}$ by: $q(h) = \sup_{x\in X} h(x)$ if $\bullet$ is "1" or "≤ 1", and $q(h) = +\infty.\sup_{x\in X} h(x)$ otherwise, agreeing that $+\infty.0 = 0$. In each case, q is sublinear ($q(h_1+h_2) \leq q(h_1)+q(h_2)$, and $q(ah) = aq(h)$ for every $a \in \mathbb{R}^+$), and $p \leq q$. The space $[X \to \overline{\mathbb{R}^+}]$ is a continuous dcpo, because X is locally compact hence core-compact, and using Proposition 2 of [5], for example. Together with the obvious, pointwise addition and scalar multiplication by non-negative reals, $[X \to \overline{\mathbb{R}^+}]$ is therefore a so-called continuous d-cone [27]. The Sandwich Theorem given there (Theorem 3.2) implies that there is a Scott-continuous linear map $\Lambda\colon X \to \overline{\mathbb{R}^+}$ such that $p \leq \Lambda \leq q$. Defining $\nu(U)$ as $\Lambda(\chi_U)$, where χ_U is the characteristic map of U, yields a continuous valuation ν in $\mathcal{V}^\bullet(X)$ such that $\mu(\Box U) = p(\chi_U) \leq \Lambda(\chi_U) = \nu(U)$ for every open subset U of X.

Since $f \leq \mathrm{id}_{\overline{\mathbb{R}^+}}$, in particular $\theta_f(\mu)$ is non-empty.

$\theta_f(\mu)$ is compact saturated. To show this, we use the following results. Define $\nu^\dagger(Q)$, for $Q \in \mathcal{Q}(X)$, as $\inf\{\nu(U) \mid Q \subseteq U\}$, and $\langle Q \geq r\rangle_\bullet$ as $\{\nu \in \mathcal{V}^\bullet(X) \mid \nu^\dagger(Q) \geq r\}$. The latter sets are compact saturated subsets of $\mathcal{V}^\bullet(X)$: this is a consequence of [8, Lemma 6.6] if $\bullet$ is "≤ 1" or "1", and of [18, Theorem 6.5 (3)] otherwise.

We now notice that:

$$\theta_f(\mu) = \{\nu \in \mathcal{V}^\bullet(X) \mid \forall Q \in \mathcal{Q}(X) \cdot \nu^\dagger(Q) \geq a^*_Q\}, \tag{1}$$

where $a^*_Q = \inf_{U \text{ open} \supseteq Q} f(\mu(\Box U))$. Before we prove this, observe that $\theta_f(\mu)$ is therefore the intersection of the compact saturated subsets $\langle Q \geq a^*_Q\rangle$, $Q \in \mathcal{Q}(X)$, and is therefore itself compact, since $\mathcal{V}^\bullet(X)$ is stably compact. (The latter holds because X is stably compact, see [17,2]. Technically, this is proved there for $\mathcal{V}^1(X)$ and $\mathcal{V}^{\leq 1}(X)$, but the proof is similar for $\mathcal{V}(X)$.)

To prove (1), let $a_U = f(\mu(\Box U))$. Every $\nu \in \theta_f(\mu)$ trivially satisfies $\nu^\dagger(Q) \geq a^*_Q$. Conversely, assume the latter holds for every $Q \in \mathcal{Q}(X)$. For every open subset U of X, by local compactness U is the directed union of all $\mathrm{int}(Q)$, where Q ranges over the compact saturated subsets of U. Since $\Box$ commutes with directed unions, and μ and f are Scott-continuous, $a_U = \sup_{Q\subseteq U} f(\mu(\Box\mathrm{int}(Q)))$. Since $\Box\mathrm{int}(Q) \subseteq U$ for every open neighborhood U of Q, this is less than or equal to $\sup_{Q\subseteq U} a^*_Q$, and the latter is less than or equal to a_U because $a^*_Q \leq a_U$ whenever $Q \subseteq U$. Therefore $a_U = \sup_{Q\subseteq U} a^*_Q$. A similar argument shows that $\nu(U) = \sup_{Q\subseteq U} \nu^\dagger(Q)$ (or see Tix [26, Satz 3.4 (1)]). It follows that $\nu(U) \geq a_U$. As U is arbitrary, ν is in $\theta_f(\mu)$.

θ_f is Scott-continuous. Monotonicity is clear, while for a directed family $(\mu_i)_{i\in I}$ in $\mathcal{V}^\bullet(X)$, $\theta_f(\sup_{i\in I}\mu_i) = \{\nu \in \mathcal{V}^\bullet(X) \mid \forall U \text{ open in } X \cdot \nu(U) \geq \sup_{i\in I} f(\mu_i(\Box U))\}$ (since f is Scott-continuous) $= \bigcap_{i\in I}\theta_f(\mu_i)$.

θ_f is continuous. Since X is T_0, well-filtered, and locally compact, $\mathcal{Q}(X)$ is a continuous dcpo, and the Scott and upper Vietoris topologies coincide [25, Section 7.3.4], i.e., $\mathcal{Q}(X) = \mathcal{Q}_{\mathsf{V}}(X)$. The Kirch-Tix Theorem states that given a con-

tinuous dcpo Y, the Scott and weak topologies coincide on $\mathcal{V}(Y)$ [26, Satz 4.10], and on $\mathcal{V}^{\leq 1}(Y)$ [19, Satz 8.6]; the same happens for $\mathcal{V}^1(Y)$ if additionally Y is pointed, by a trick due to Edalat [4, Section 3]: $Y' = Y \smallsetminus \{\bot\}$ is again a continuous dcpo, and $\mathcal{V}^1(Y)$ is isomorphic to $\mathcal{V}^{\leq 1}(Y')$. Taking $Y = \mathcal{Q}(X) = \mathcal{Q}_{\mathsf{V}}(X)$ (and noticing that this is pointed, as X is compact), we obtain that $\mathcal{V}^{\bullet}(\mathcal{Q}_{\mathsf{V}}(X))$ has the Scott topology of the pointwise ordering. To show that θ_f is continuous, it is therefore enough to show that $\theta_f^{-1}(\Box\mathcal{U})$ is open in the Scott topology for every open subset $\mathcal{U}$ of $\mathcal{V}^{\bullet}(X)$. Since $\Box\mathcal{U}$ is itself Scott-open by well-filteredness, this amounts to the Scott-continuity of θ_f. □

Theorem 6.2 *For every QRB-domain X, $\mathcal{V}(X)$, $\mathcal{V}^{\leq 1}(X)$, and also $\mathcal{V}^1(X)$ if X is pointed, are QRB-domains.*

Proof. Let X be a QRB-domain, and $(\varphi_i)_{i \in I}$ be an approximating family of quasi-deflations on X. By Theorem 5.7 (*iii*), we only need to show that $\mathcal{V}^{\bullet}(X)$ is a QFS-space. It is sober since stably compact, as we have noted earlier.

For $\epsilon \in (0, 1]$, and $t \in \overline{\mathbb{R}^+}$, let $f_\epsilon(t) = \max(0, \min(t, 1/\epsilon) - \epsilon)$. This is a chain of Scott-continuous maps, as $\epsilon \geq \epsilon'$ implies $f_\epsilon \leq f_{\epsilon'}$. Also, $f_\epsilon \leq \mathrm{id}_{\overline{\mathbb{R}^+}}$. For short, write θ_ϵ for the map θ_{f_ϵ} given in Lemma 6.1, and define $\psi_{i\epsilon}$ as $\theta_\epsilon \circ \mathcal{V}\varphi_i \colon \mathcal{V}^{\bullet}(X) \to \mathcal{Q}_{\mathsf{V}}(\mathcal{V}^{\bullet}(X))$. The family $(\psi_{i\epsilon})_{i \in I, \epsilon \in (0,1]}$ is directed, and for every $\nu \in \mathcal{V}^{\bullet}(X)$, we claim that $\bigcap_{i \in I, \epsilon \in (0,1]} \psi_{i\epsilon}(\nu) = {\uparrow} \nu$.

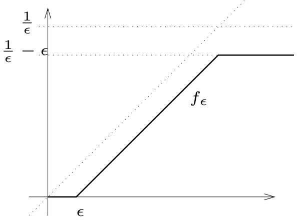


Fig. 2: The function f_ϵ

To this end, we notice that:

(*a*) $\bigcap_{\epsilon \in (0,1]} \theta_\epsilon(\mu) = \theta_{\mathrm{id}}(\mu)$. Indeed, $\bigcap_{\epsilon \in (0,1]} \theta_\epsilon(\mu) = \{\nu' \in \mathcal{V}^{\bullet}(X) \mid \forall U \text{ open in } X \cdot \nu'(U) \geq \sup_{\epsilon \in (0,1]} f_\epsilon(\mu(\Box U))\} = \{\nu' \in \mathcal{V}^{\bullet}(X) \mid \forall U \text{ open in } X \cdot \nu'(U) \geq \mu(\Box U)\} = \theta_{\mathrm{id}}(\mu)$.

(*b*) $\bigcap_{i \in I} \theta_{\mathrm{id}}(\mathcal{V}\varphi_i(\nu)) = {\uparrow} \nu$. This is proved as follows. For every open subset U of X, $\bigcup_{i \in I} \varphi_i^{-1}(\Box U) = U$: the elements x of U are those such that ${\uparrow} x \in \Box U$, and we obtain the desired equality by the defining property of quasi-deflations, plus well-filteredness. It follows that $\sup_{i \in I} \nu(\varphi_i^{-1}(\Box U)) = \nu(\bigcup_{i \in I} \varphi_i^{-1}(\Box U)) = \nu(U)$. The elements ν' of $\bigcap_{i \in I} \theta_{\mathrm{id}}(\mathcal{V}\varphi_i(\nu))$ are those elements of $\mathcal{V}^{\bullet}(X)$ such that, for every $i \in I$, for every open subset U of X, $\nu'(U) \geq \mathcal{V}\varphi_i(\nu)(\Box U)$; equivalently, such that $\nu'(U) \geq \sup_{i \in I} \mathcal{V}\varphi_i(\nu)(\Box U) = \nu(U)$, and we conclude.

Using these, $\bigcap_{i \in I, \epsilon \in (0,1]} \psi_{i\epsilon}(\nu) = \bigcap_{i \in I} \bigcap_{\epsilon \in (0,1]} \theta_\epsilon(\mathcal{V}\varphi_i(\nu)) = \bigcap_{i \in I} \theta_{\mathrm{id}}(\mathcal{V}\varphi_i(\nu)) = {\uparrow} \nu$, as announced.

It remains to show that $\psi_{i\epsilon}$ is qfs. Write δ_x for the Dirac mass at x, namely, the continuous valuation such that $\delta_x(U) = \chi_U(x)$ for every open U. Let E be the finite set of all elements that are minimal in some finitary compact in the image of φ_i, n be its cardinality, and let M be the finite set of continuous valuations of the form $\sum_{x \in E} a_x \delta_x$, where each a_x is an integer multiple of ϵ/n between 0 and $1/\epsilon$ (and with $\sum_{x \in E} a_x \leq 1$ if $\bullet$ is "≤ 1", $\sum_{x \in E} a_x = 1$ if $\bullet$ is "1"). This will be our separating set.

Fix $\nu \in \mathcal{V}^{\bullet}(X)$. We first simplify the expression of $\psi_{i\epsilon}(\nu)$. For $Q \in \mathcal{Q}(X)$, let

$a^*_Q = \inf_{U \text{ open} \supseteq Q} f_\epsilon(\mathcal{V}\varphi_i(\nu)(\Box U)) = \inf_{U \text{ open} \supseteq Q} f_\epsilon(\nu(\varphi_i^{-1}(\Box U)))$. Let $Q_1, \cdots, Q_m$ be the finitely many finitary compacts in the image of φ_i. For $J \subseteq \{1, \cdots, m\}$, write Q_J for $\bigcup_{j \in J} Q_j$. We claim that $\psi_{i\epsilon}(\nu) = \bigcap_{J \subseteq \{1, \cdots, m\}} \langle Q_J \geq a^*_{Q_J} \rangle$. To this end, recall equality (1), which we have used in the course of proving Lemma 6.1: $\theta_f(\mu) = \{\nu' \in \mathcal{V}^\bullet(X) \mid \forall Q \in \mathcal{Q}(X) \cdot {\nu'}^\dagger(Q) \geq \inf_{U \text{ open} \supseteq Q} f(\mu(\Box U))\}$. So $\psi_{i\epsilon}(\nu) = \{\nu' \in \mathcal{V}^\bullet(X) \mid \forall Q \in \mathcal{Q}(X) \cdot {\nu'}^\dagger(Q) \geq a^*_Q\} = \bigcap_{Q \in \mathcal{Q}(X)} \langle Q \geq a^*_Q \rangle$. Looking back at the definition of a^*_Q, we see that, since φ_i has finite image, $\varphi_i^{-1}(\Box U)$ can only take finitely many values when U varies. The family of these values forms a (finite) filtered family of open sets, which therefore has a least element, which happens to be $\varphi_i^{-1}(\Box Q)$ (extending the $\Box$ notation in the obvious way). Hence $a^*_Q = f_\epsilon(\nu(\varphi_i^{-1}(\Box Q)))$. For every $\nu' \in \bigcap_{J \subseteq \{1, \cdots, m\}} \langle Q_J \geq a^*_{Q_J} \rangle$, and every $Q \in \mathcal{Q}(X)$, let J be the set of indices $j \in \{1, \cdots, m\}$ such that $Q_j \subseteq Q$. Since φ_i takes its values among $Q_1, \ldots, Q_m$, $\varphi_i^{-1}(\Box Q) = \varphi_i^{-1}(\Box Q_J)$, so $a^*_Q = a^*_{Q_J}$. It follows that ${\nu'}^\dagger(Q) \geq \nu^\dagger(Q_J) \geq a^*_{Q_J} = a^*_Q$, and as Q is arbitrary, $\nu' \in \bigcap_{Q \in \mathcal{Q}(X)} \langle Q \geq a^*_Q \rangle = \psi_{i\epsilon}(\nu)$. The converse inclusion $\psi_{i\epsilon}(\nu) \subseteq \bigcap_{J \subseteq \{1, \cdots, m\}} \langle Q_J \geq a^*_{Q_J} \rangle$ is obvious.

To show that $\psi_{i\epsilon}$ is qfs, it will therefore be enough to find an element $\sum_{x \in E} a_x \delta_x$ of M below ν and in $\psi_{i\epsilon}(\nu) = \bigcap_{J \subseteq \{1, \cdots, m\}} \langle Q_J \geq a^*_{Q_J} \rangle$.

Let L be the finite lattice of all intersections of sets of the form $\uparrow A$, $A \subseteq E$. Tix observed that $\nu^\dagger$ defined a valuation on the compact saturated subsets of X [26, Satz 3.4 (2–4)]. In particular $\nu^\dagger$ restricts to a valuation on L. Using the Smiley-Horn-Tarski Theorem (see, e.g., [20, Theorem 3.4]), $\nu^\dagger$ extends to an additive measure on the algebra ρL of subsets generated by L. The algebra ρL is the smallest collection of subsets containing L and closed under unions, intersections, and complements. Its elements are the finite disjoint unions of sets of the form $C_A = \bigcap_{x \in A} \uparrow x \smallsetminus \bigcup_{x \in E \smallsetminus A} \uparrow x$, $A \subseteq E$.

For each $x \in E$, let $b_x = \nu^\dagger(C_{A_x})$ where A_x is the unique subset of E such that C_{A_x} contains x. This definition ensures that $\nu^\dagger(B) = \sum_{x \in B \cap E} b_x = (\sum_{x \in E} b_x \delta_x)(B)$ for every $B \in L$. For every open subset U of X, and every $x \in E \cap U$, $C_{A_x} \subseteq \uparrow x \subseteq U$, so $(\sum_{x \in E} b_x \delta_x)(U) = \sum_{x \in E \cap U} \nu^\dagger(C_{A_x}) = \nu^\dagger(\bigcup_{x \in E \cap U} C_{A_x})$ (since the sum is disjoint) $\leq \nu^\dagger(\uparrow(E \cap U)) \leq \nu(U)$.

When $\bullet$ is "≤ 1", we define the desired element $\sum_{x \in E} a_x \delta_x$ of M by letting a_x be the nearest multiple of ϵ/n below b_x, namely $\frac{\epsilon}{n} \lfloor \frac{n}{\epsilon} b_x \rfloor$. Clearly, $\sum_{x \in E} a_x \delta_x \leq \sum_{x \in E} b_x \delta_x \leq \nu$. Moreover, for every $J \subseteq \{1, \cdots, m\}$, $(\sum_{x \in E} a_x \delta_x)^\dagger(Q_J) + \epsilon = \sum_{x \in E \cap Q_J} a_x + \epsilon \geq \sum_{x \in E \cap Q_J} (a_x + \frac{\epsilon}{n}) \geq \sum_{x \in E \cap Q_J} b_x = \nu^\dagger(Q_J)$, since Q_J belongs to ρL. Since $\varphi_i(x) \supseteq \uparrow x$ for every $x \in X$, $\varphi^{-1}(\Box Q_J) \subseteq Q_J$. For every $Q \in \mathcal{Q}(X)$, recall that $\varphi_i^{-1}(\Box Q)$ is the least element of some finite filtered family of open sets, hence is open. It follows that the notation $\nu(\varphi_i^{-1}(\Box Q_J))$ makes sense. From $\varphi_i^{-1}(\Box Q_J) \subseteq Q_J$, we obtain $\nu^\dagger(Q_J) \geq \nu(\varphi_i^{-1}(\Box Q_J))$. We have just shown that $(\sum_{x \in E} a_x \delta_x)^\dagger(Q_J) + \epsilon \geq \nu(\varphi_i^{-1}(\Box Q_J))$, whence $(\sum_{x \in E} a_x \delta_x)^\dagger(Q_J) \geq \max(0, \nu(\varphi_i^{-1}(\Box Q_J)) - \epsilon) = f_\epsilon(\nu(\varphi_i^{-1}(\Box Q_J))) = a^*_{Q_J}$. (We are silently using the fact that $f_\epsilon(t) = \max(0, t - \epsilon)$ for every $t \in [0, 1]$.) Therefore $\sum_{x \in E} a_x \delta_x$ is in $\langle Q_J \geq a^*_{Q_J} \rangle$, and as J is arbitrary, it is in $\psi_{i\epsilon}(\nu)$.

When $\bullet$ is "1" instead, we use the standard trick of putting all the missing mass on the bottom element $\bot$. In other words, we define a_x as above for $x \neq \bot$, and as $1 - \sum_{x \in E, x \neq \bot} a_x$ otherwise. (Note that E contains $\bot$. Indeed, it appears as the minimal element of $\varphi_i(\bot) = X$.) We check that $(\sum_{x \in E} a_x \delta_x)(U) \leq \nu(U)$ as above when U does not contain $\bot$, while the same inequality reduces to the trivial $1 \leq 1$ when U contains $\bot$, namely when $U = X$. Since we are using larger coefficients a_x than in the "≤ 1" case, the fact that $(\sum_{x \in E} a_x \delta_x)^\dagger(Q_J) \geq a^*_{Q_J}$ follows by the same arguments. It follows, again, that $\sum_{x \in E} a_x \delta_x$ is in $\bigcap_{J \subseteq \{1, \cdots, m\}} \langle Q_J \geq a^*_{Q_J} \rangle = \psi_{i\epsilon}(\nu)$.

Finally, when $\bullet$ is neither "≤ 1" not "1", we argue as in the "≤ 1" case, except we now define a_x as $\frac{\epsilon}{n} \lfloor \frac{n}{\epsilon} \min(\frac{1}{\epsilon}, b_x) \rfloor$. To check that $(\sum_{x \in E} a_x \delta_x)^\dagger(Q_J) \geq a^*_{Q_J}$, we compute $(\sum_{x \in E} a_x \delta_x)^\dagger(Q_J) + \epsilon = \sum_{x \in E \cap Q_J} a_x + \epsilon \geq \sum_{x \in E \cap Q_J} (a_x + \frac{\epsilon}{n}) \geq \sum_{x \in E \cap Q_J} \min(\frac{1}{\epsilon}, b_x)$. If every b_x is less than or equal to $\frac{1}{\epsilon}$, the latter is equal to $\nu^\dagger(Q_J)$, hence greater than or equal to $\min(\frac{1}{\epsilon}, \nu^\dagger(Q_J))$. If $b_x > \frac{1}{\epsilon}$ for some $x \in E \cap Q_J$, then $\nu^\dagger(Q_J) \geq \nu^\dagger(C_{A_x}) = b_x > \frac{1}{\epsilon}$, so $\sum_{x \in E \cap Q_J} \min(\frac{1}{\epsilon}, b_x) \geq \frac{1}{\epsilon} = \min(\frac{1}{\epsilon}, \nu^\dagger(Q_J))$. In any case, $(\sum_{x \in E} a_x \delta_x)^\dagger(Q_J) + \epsilon \geq \min(\frac{1}{\epsilon}, \nu^\dagger(Q_J))$. As in the "$\leq 1$" case, this implies $(\sum_{x \in E} a_x \delta_x)^\dagger(Q_J) + \epsilon \geq \min(\frac{1}{\epsilon}, \nu(\varphi_i^{-1}(\Box Q_J)))$, so $(\sum_{x \in E} a_x \delta_x)^\dagger(Q_J) \geq f_\epsilon(\nu(\varphi_i^{-1}(\Box Q_J))) = a^*_{Q_J}$. $\Box$

References

[1] Abramsky, S. and A. Jung, *Domain theory*, in: S. Abramsky, D. M. Gabbay and T. S. E. Maibaum, editors, *Semantic Structures*, Handbook of Logic in Computer Science **3**, Oxford University Press, 1994 pp. 1–168.

[2] Alvarez-Manilla, M., A. Jung and K. Keimel, *The probabilistic powerdomain for stably compact spaces*, Theoretical Computer Science **328** (2004), pp. 221–244.

[3] Banaschewski, B., *Essential extensions of T_0-spaces*, General Topology and Applications **7** (1977), pp. 233–246.

[4] Edalat, A., *Domain theory and integration*, Theoretical Computer Science **151** (1995), pp. 163–193.

[5] Erker, T., M. H. Escardo and K. Keimel, *The way-below relation of function spaces over semantic domains*, Topology and Its Applications **89** (1998), pp. 61–74.

[6] Gierz, G., K. H. Hofmann, K. Keimel, J. D. Lawson, M. Mislove and D. S. Scott, "Continuous Lattices and Domains," Encyclopedia of Mathematics and its Applications **93**, Cambridge University Press, 2003.

[7] Gierz, G., J. D. Lawson and A. Stralka, *Quasicontinuous posets*, Houston Journal of Mathematics **9** (1983), pp. 191–208.

[8] Goubault-Larrecq, J., *De Groot duality and models of choice: Angels, demons, and nature*, Mathematical Structures in Computer Science **20** (2010), pp. 169–237.
URL http://www.lsv.ens-cachan.fr/Publis/PAPERS/PDF/JGL-mscs09.pdf

[9] Goubault-Larrecq, J., *ω**QRB**-domains and the probabilistic powerdomain*, in: *Proceedings of the 25th Annual IEEE Symposium on Logic in Computer Science (LICS'10)* (2010), pp. 352–361.
URL http://www.lsv.ens-cachan.fr/Publis/PAPERS/PDF/JGL-lics10.pdf

[10] Goubault-Larrecq, J., *QRB-domains and the probabilistic powerdomain*, Logical Methods in Computer Science **8** (2012), long and improved version of [9].
URL http://www.lsv.ens-cachan.fr/Publis/PAPERS/PDF/JGL-lmcs12.pdf

[11] Goubault-Larrecq, J., "Non-Hausdorff Topology and Domain Theory — Selected Topics in Point-Set Topology," New Mathematical Monographs **22**, Cambridge University Press, 2013.

[12] Heckmann, R., *Spaces of valuations*, in: S. Andima, R. C. Flagg, G. Itzkowitz, Y. Kong, R. Kopperman and P. Misra, editors, *Papers on General Topology and Applications: 11th Summer Conference at the University of Southern Maine*, Annals of the New York Academy of Sciences **806**, New York, USA, 1996, pp. 174–200.

[13] Heckmann, R. and K. Keimel, *Quasicontinuous domains and the Smyth powerdomain*, in: D. Kozen and M. Mislove, editors, *Proc. 29th Ann. Conf. Mathematical Foundations of Programming Semantics (MFPS'13)*, 2013, pp. 217–316, published in *Electronic Notes in Theoretical Computer Science.*

[14] Isbell, J. R., *Meet-continuous lattices*, Symposia Mathematica **16** (1975), pp. 41–54, convegno sulla Topologica Insiemsistica e Generale, INDAM, Roma, Marzo 1973.

[15] Jones, C., "Probabilistic Non-Determinism," Ph.D. thesis, University of Edinburgh (1990), technical Report ECS-LFCS-90-105.

[16] Jung, A., *The classification of continuous domains*, in: *Proceedings of the Fifth Annual IEEE Symposium on Logic in Computer Science* (1990), pp. 35–40.

[17] Jung, A. and R. Tix, *The troublesome probabilistic powerdomain*, in: A. Edalat, A. Jung, K. Keimel and M. Kwiatkowska, editors, *Proc. 3rd Workshop on Computation and Approximation*, Electronic Lecture Notes in Computer Science **13** (1998), pp. 70–91, 23pp.

[18] Keimel, K., *Topological cones: Foundations for a domain-theoretical semantics combining probability and nondeterminism*, Electronic Notes in Theoretical Computer Science **155** (2006), pp. 423–443.

[19] Kirch, O., "Bereiche und Bewertungen," Master's thesis, Technische Hochschule Darmstadt (1993).

[20] König, H., "Measure and Integration: An Advanced Course in Basic Procedures and Applications," Springer Verlag, 1997.

[21] Lawson, J. and X. Xi, *The equivalence of QRB, QFS, and compactness for quasicontinuous domains*, Order (2013), submitted.

[22] Lawson, J. D., *T_0-spaces and pointwise convergence*, Topology and its Applications **21** (1985), pp. 73–76.

[23] Li, G. and L. Xu, *QFS-domains and their Lawson compactness*, Order **30** (2013), pp. 233–248.

[24] Plotkin, G. D., *A powerdomain construction*, SIAM Journal on Computing **5** (1976), pp. 452–487.

[25] Schalk, A., "Algebras for Generalized Power Constructions," Ph.D. thesis, Technische Universität Darmstadt (1993).

[26] Tix, R., "Stetige Bewertungen auf topologischen Räumen," Diplomarbeit, TH Darmstadt (1995).

[27] Tix, R., K. Keimel and G. Plotkin, *Semantic domains for combining probability and non-determinism*, Electronic Notes in Theoretical Computer Science **129** (2005), pp. 1–104.

Adequacy and Complete Axiomatization for Timed Modal Logic

Samy Jaziri[1] Kim G. Larsen[2] Radu Mardare[2] Bingtian Xue[2]

[1] Lab. Spécification & Vérification, ENS Cachan, France [2] Aalborg University, Denmark

Abstract

In this paper we develop the metatheory for Timed Modal Logic (TML), which is the modal logic used for the analysis of timed transition systems (TTSs). We solve a series of long-standing open problems related to TML. Firstly, we prove that TML enjoys the Hennessy-Milner property and solve one of the open questions in the field. Secondly, we prove that the set of validities are not recursively enumerable. Nevertheless, we develop a strongly-complete proof system for TML. Since the logic is not compact, the proof system contains infinitary rules, but only with countable sets of instances. Thus, we can involve topological results regarding Stone spaces, such as the Rasiowa-Sikorski lemma, to complete the proofs.

Keywords: Non-compact modal logics, complete axiomatization, timed modal logic.

1 Introduction

In the areas of embedded and cyber-physical systems, more than two decades of research have been dedicated to developing quantitative modeling and specification formalisms that allow for the construction of systems with guaranteed functional and non-functional properties. In particular, many embedded systems are highly safety critical, with hard constraints on real-time behaviors being essential. Here, the notion of *timed transition system* and *timed automata* [4] have proven extremely convenient for modeling purposes and are now routinely used for the analysis of real-time communication protocols and control programs.

Accompanying the timed transition systems, a variety of *timed temporal logics* have been introduced as convenient ways of capturing requirements to real-time systems. These logics include MTL [13], MITL [5], TPTL [7, 8] and ECL [21] all providing timed extensions of LTL. Similarly TCTL [2], T$_\mu$ [12] and L$_\nu$ [16] provide timed extensions of CTL and the modal μ-calculus. Emphasis has been on detailed investigation of decidability and complexity of model checking and satisfiability checking, identifying the importance of (absence of) punctual constraints in the logic [5, 15] as well as of the choice of semantics of timed automata (point-wise or continuous) [17–19]. Also, with the purpose of gaining decidability, satisfiability checking given constraints on number of clocks and size of constants in a satisfying timed automata has been considered [16]. In all of the aforementioned logics, quantitative requirements are either obtained by decorating modalities with timing constraints, or by using clocks in formulas.

The presence of time in all these settings makes it difficult to address fundamental meta-theoretical questions regarding the timed logics, such as *adequacy*[1] or the development of complete proof systems. Regarding the axiomatization, there exists a series of results proved in very restricted settings, such as for TPTL [11] that only looks to models with discrete time, or for ECTL [22] under the restriction of models with finite variability – only finitely many state changes can occur in a finite amount of time. Other attempts, such as [9], failed already in achieving soundness results, as argued in [22].

In this paper we take the challenge of developing a strongly-complete proof system for the most basic Timed Modal Logic (TML) defined for the most general model of real-time systems, without any restriction on the nature of time. Our models are timed labelled transition systems (TTSs), which generalize timed automata [1]: their transitions are labeled with actions or time delays (real numbers). Our logic is the non-recursive fragment of L_ν [16], which generalizes the logic of [1]. In these settings we solve a series of open problems.

Firstly, we prove that TML is adequate not only for timed automata, but in general, for entire class of TTSs. This settles an open problem and disproves a belief often found in the literature, e.g., [1] – that such a logic is not sufficiently expressive to characterize timed-bisimulation. In proving this result, we use a novel exploitation of formulas with free clock variables.

Secondly, we prove that the satisfiability problem for TML is undecidable despite its restrictive expressive power, thus generalizing the known undecidability result of satisfiability for TCTL [2]. Moreover, we show that the set of TML-validities is not recursively-enumerable. This implicitly means that any complete proof system will generate a non-recursively enumerable set of provable formulas, and makes one wonder whether this logic can, in fact, be axiomatized at all - see e.g., the discussion in [6].

We prove that TML can be axiomatized and we develop a proof system for it that is *strongly-complete*[2] for the TTS-semantics. However, TML is not *compact*: there exists an infinite set of formulas that admits no model while all its finite subsets have models. For this reason our axiomatization must contains infinitary proof rules; they reflect Archimedean properties of the rational numbers used to interpret

[1] A logic is adequate when its semantical equivalence over the class of models coincides with bisimilarity.

[2] Strong completeness means that [$\Phi \vdash \phi$ iff $\Phi \models \phi$], where $\Phi \models \phi$ denotes that all the models of the set Φ of formulas are also models of the formula ϕ; and $\Phi \vdash \phi$ denotes that ϕ is provable from Φ.

clocks. Due to these infinitary rules, the proofs of our system cannot be enumerated. Nevertheless, these rules have countable sets of instances.

Our axioms provide a set of sufficient conditions that characterizes the concept of time in models like TTSs and timed automata. They reflect properties such as persistence, continuity, linearity, density of time, synchronization, or the fact that all clocks measure the same time flow. Most of these axioms are similar to axioms seen in other modal logics and these relations open interesting further questions.

An other important contribution of the paper is the construction of the canonical model, which generalizes for the case of a (non-compact) higher-order modal logic the classic filtration construction usually used for propositional modal logics. This follows the line opened by the second and the third author in collaboration with Panangaden and Kozen in [14]; it involves complex topological and model-theoretical arguments, such as the Rasiowa-Sikorski lemma, which are essential in achieving the main results and are pointing to a general methodology for constructing canonical models for non-compact modal logics.

This paper does not aim at addressing problems related to timed automata, nor to model verification, model construction or model checking. Our purpose is to clarify the open problems of model theory for TML. The interesting questions regarding timed automata-semantics will be addressed in a future work.

2 Preliminaries

Hereafter we fix the notation used in the paper.

Kleene equality. Given a partial function $f : X \to Y$, we write $f(x) \overset{\sim}{=} f(x')$ for $x, x' \in X$ to denote the fact that $f(x)$ and $f(x')$ are simultaneously well-defined and whenever they are well-defined, they are equal.

Orders on reals. We use $\trianglelefteq$ and $\trianglerighteq$ to range over the set $\{\leq, \geq\}$ such that $\{\trianglelefteq, \trianglerighteq\} = \{\leq, \geq\}$; this means that $\trianglelefteq$ can either be interpreted as $\leq$ or as $\geq$; if $\trianglelefteq$ represents one of the two, then $\trianglerighteq$ denotes the other. Similarly, we use $\lhd$ and $\rhd$ to range over the set $\{<, >\}$ such that $\{\lhd, \rhd\} = \{<, >\}$. Moreover, $x \lhd y$ means $[x \trianglelefteq y$ and $x \neq y]$, and similarly for $\rhd$ and $\trianglerighteq$. We use $\bowtie$ to denote to an arbitrary element of the set $\{\leq, \geq, <, >\}$.

Interpretations on reals. Given a set $\mathcal{K}$, an interpretation of $\mathcal{K}$ (on non-negative reals) is a function $i : \mathcal{K} \to \mathbb{R}_{\geq 0}$; if $x \in \mathcal{K}$ and $r \in \mathbb{R}_{\geq 0}$, we denote by $i[x \mapsto r]$ the interpretation j of $\mathcal{K}$ such that $j(x) = r$ and $j(y) = i(y)$ for $y \neq x$. The arithmetic operations on interpretations are defined pointwise; 0 is used for the constant null interpretation and $[x \mapsto r]$ denotes $0[x \mapsto r]$.

The Rasiowa-Sikorski Lemma [10, 20] is a result with important applications in logic.

Definition 2.1 Let $\mathcal{B}$ be a Boolean algebra and let $T \subseteq \mathcal{B}$ such that T has a greatest lower bound $\bigwedge T$ in $\mathcal{B}$. An ultrafilter (maximal filter) U is said to *respect* T if $T \subseteq U$ implies that $\bigwedge T \in U$.

If $\mathcal{T}$ is a family of subsets of $\mathcal{B}$, we say that an ultrafilter U respects $\mathcal{T}$ if it respects every member of $\mathcal{T}$.

Lemma 2.2 (Rasiowa–Sikorski [20]) *For any Boolean algebra $\mathcal{B}$ and any countable family $\mathcal{T}$ of subsets of $\mathcal{B}$, each member of which has a meet in $\mathcal{B}$, and for any nonzero $x \in \mathcal{B}$, there exists an ultrafilter in $\mathcal{B}$ that contains x and respects $\mathcal{T}$.*

3 Timed Transition Systems

A *timed transition system* (TTS) [1] is a labeled transition system that uses both actions and time delays as transition labels. The delay transitions describe the time flow and consequently are continuous and deterministic. Here we propose an equivalent definition for TTS that encodes the time in an algebraic format and simplifies our future developments.

Definition 3.1 [Timed Transition System] A *timed labeled transition system* (TTS) is a tuple $\mathcal{W} = (M, \Sigma, \theta, \oplus)$ where M is a non-empty set of *states*, Σ a non-empty set of *actions*, $\theta : M \times \Sigma \to 2^M$ is the *action-labelling function* and $\oplus : \mathbb{R}_{\geq 0} \times M \rightharpoonup M$ is a partial function that encodes the *delay transitions*; for arbitrary $m \in M$ and $d, d' \in \mathbb{R}_{\geq 0}$,
1. $0 \oplus m = m$;
2. $d \oplus (d' \oplus m) \overset{\sim}{=} (d + d') \oplus m$.

Whenever it is defined, $d \oplus m$ denotes a time delay d applied to the state m. Condition 1 guarantees that a zero-delay is always well-defined and it does not change the state of the system; and condition 2 expresses that time is both *additive* and *deterministic*. As usual, instead of $m' \in \theta(m, a)$ we will write $m \xrightarrow{a} m'$.

In the rest of this paper we fix the set Σ and omit it in the description of TTSs.

A *timed bisimulation* is a relation that equates states of a TTS with identical behaviours.

Definition 3.2 [Timed Bisimulation] Given a TTS $\mathcal{W} = (M, \theta, \oplus)$, a *timed bisimulation* is an equivalence relation $R \subseteq M \times M$ such that whenever $(m, n) \in R$, for all $a \in \Sigma$ and $d \in \mathbb{R}_{\geq 0}$,
- if $m \xrightarrow{a} m'$, then there exists $n' \in M$ s.t. $n \xrightarrow{a} n'$ and $(m', n') \in R$;
- if $d \oplus m$ is well-defined, then $d \oplus n$ is well-defined and $(d \oplus m, d \oplus n) \in R$.

As for the other types of bisimulation, the previous definition can be extended to define the time bisimulation between distinct TTSs by considering bisimulation relations on their disjoint union. *Time bisimilarity* is the largest time-bisimulation relation; if $\mathcal{W}_i = (M_i, \theta_i, \oplus_i)$, $m_i \in M_i$ for $i = 1, 2$ and m_1 and m_2 are bisimilar, we write $(m_1, \mathcal{W}_1) \sim (m_2, \mathcal{W}_2)$.

4 Timed Modal Logic

In this section we introduce the Timed Modal Logic (TML) that encodes properties of TTSs. It is defined for a countable set $\mathcal{K}$ of *clocks* that we consider fixed in what follows. It contains Henessy-Milner operators $[a]\phi$ for the actions $a \in \Sigma$, where Σ is the fixed set of actions for which we have defined TTSs in the previous section. In addition, it is endowed with *time inequalities* of type $x \trianglelefteq r$ for rational values r that evaluate the clock $x \in \mathcal{K}$ at the current state; with *delay quantifiers* $\forall\!\!\!\forall\phi$ that predicate properties for any time-delay of the current state; and *clock quantifiers* $\forall x.\phi$ that predicate properties for any interpretation of the clock x at the current state.

Definition 4.1 [Syntax] For arbitrary $r \in \mathbb{Q}_{\geq 0}$ and $a \in \Sigma$,

$$\mathcal{L}: \quad \phi := \bot \mid x \trianglelefteq r \mid \phi \to \phi \mid [a]\phi \mid \forall\!\!\!\forall\phi \mid \forall x.\phi.$$

Let $\Im(\mathcal{K})$ be the set of interpretations of $\mathcal{K}$. The semantics of TML is defined for an arbitrary TTS $M = (M, \theta, \oplus)$, $m \in M$ and $i \in \Im(\mathcal{K})$ as follows.

- $M, m, i \models \bot$ – never;
- $M, m, i \models x \trianglelefteq r$ if $i(x) \trianglelefteq r$;
- $M, m, i \models \phi \to \psi$ if $M, m, i \models \psi$ whenever $M, m, i \models \phi$;
- $M, m, i \models [a]\phi$ if [for any $m' \in M$ s.t. $m \xrightarrow{a} m'$, $M, m', i \models \phi$];
- $M, m, i \models \forall\!\!\!\forall\phi$ if [for any $d \in \mathbb{R}_{\geq 0}$ s.t. $d \oplus m$ is well-defined, $M, d \oplus m, i + d \models \phi$];
- $M, m, i \models \forall x.\phi$ if [$\forall t \in \mathbb{R}_{\geq 0}$,$M, m, i[x \mapsto t] \models \phi$]

We also use, in addition, all the boolean operators defined as usual and the De Morgan duals of the modal operators: $\langle a\rangle\phi \overset{df}{=} \neg[a]\neg\phi$, $x \triangleright r \overset{df}{=} \neg(x \trianglelefteq r)$, $\exists\!\!\!\exists\phi \overset{df}{=} \neg(\forall\!\!\!\forall\neg\phi)$, and $\exists x.\phi \overset{df}{=} \neg(\forall x.\neg\phi)$. Other derived operators used in what follows are: $(x = r) \overset{df}{=} (x \leq r) \wedge (x \geq r)$ and $\forall(x \bowtie r).\phi \overset{df}{=} \forall x.(x \bowtie r \to \phi)$, $\bowtie \in \{\leq, \geq, <, >\}$.

In TML we can express the reset operator used in [1, 12, 16] by

$$x \text{ in } \phi \overset{df}{=} \forall x.(x = 0 \to \phi).$$

Whenever it is not the case that $M, m, i \models \phi$, we write $M, m, i \not\models \phi$. We say that a formula ϕ is *satisfiable* if there exists at least one TTS that satisfies it in one of its states under at least one interpretation; ϕ is a *validity* if it is satisfied in all states of any TTS under any interpretation - in this case we write $\models \phi$. For an arbitrary set $\Phi \subseteq \mathcal{L}$, we write $\Phi \models \phi$ if all the models of all the formulas in Φ are also models of ϕ.

4.1 Undecidability of TML.

Individual formulae of TML express properties which only depend on the behaviour of a TTS up to a finite action-depth, thus making TML significantly less expressive than TCTL [3]. In a number of papers it has been shown how recursive extensions of TML – e.g. extensions with the ability to define logical properties recursively – enable the encoding of TCTL, while maintaining decidability of model-checking.

As stated in the theorem below, despite its limited expressive power the question of satisfiability for TML is (highly) undecidable, as is the case of TCTL. Formally, given an arbitrary TML formula $\phi \in \mathcal{L}$, it is undecidable whether there exists a TTS $(M, \theta, \oplus)$ with $m \in M$ and interpretation $i \in \Im(\mathcal{K})$ such that $M, m, i \models \phi$.

Theorem 4.2 (Undecidability of TML) *The satisfiability question for TML is Σ^1_1-hard, hence undecidable.*

Proof. We show that we can reduce the TML satisfiability question into the question as to whether a non-deterministic 2-counter machines has a computation with the initial location being visited infinitely often. This last question is known to be Σ^1_1-hard. Our proof is similar to the one in [3] and for this reason it is presented in the Appendix. □

The undecidability of satisfiability for TML implies, as usual, the undecidability of validity for TML. In fact, Theorem 4.2 proves that the set of validities is not recursively enumerable.

5 Adequacy of TML

The Hennessy-Milner property (H-Mp), which states for a logic that bisimilarity of the models coincides with the semantic equivalence induced by the logic, is currently an open problem for timed logic. In [1] it was proven that the closed formulas (without free clock variables) cannot characterize bisimilarity. In this section we prove the H-Mp for TML, therefore we solve the adequacy problem.

Before proceeding with the proof, observe the essential role of interpretations in the semantics of TML. Consider the two TTSs depicted in Figure 1, where the horizontal lines represent the time flow from the initial states m and m' respectively. The two systems in the initial states can delay forever and they can both take an a-transition to states that satisfy END after each delay $2 - \frac{2}{2^n}$ and $2 + \frac{2}{2^n}$ for each integer $n > 0$. However, the two systems differ: m can take an a-transition after the delay 2, while m' cannot.

If we consider an interpretation $i \in \Im(\mathcal{K})$ s.t. $i(x) \notin \mathbb{Q}_{\geq 0}$ for any $x \in \mathcal{K}$, one can notice that m and m' satisfy exactly the same formulas. However, this is not true if we consider an interpretation i' s.t. $i'(x) = 0$ for some $x \in \mathcal{K}$, since $m, i' \models \exists(x = 2 \wedge \langle a \rangle \top)$ and $m', i' \models \neg(\exists(x = 2 \wedge \langle a \rangle \top))$.

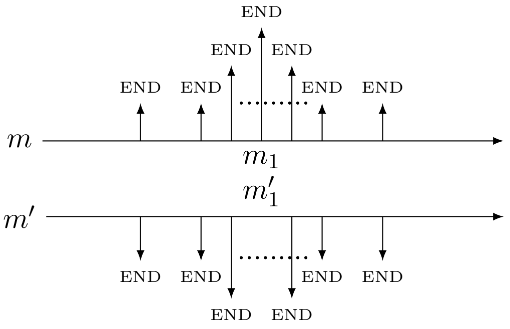


Fig. 1. non-bisimilar TTSs

To clarify this situation, we start from analysing how the formulas satisfied by a model under a certain interpretation change when we change the interpretation.

It is useful in what follows to identify the set $\mathcal{K}(\phi)$ of the free clocks in a formula $\phi \in \mathcal{L}$, defined by: $\mathcal{K}(\bot) = \emptyset$, $\mathcal{K}(x \bowtie r) = \{x\}$, $\mathcal{K}(\phi \to \psi) = \mathcal{K}(\phi) \cup \mathcal{K}(\psi)$, $\mathcal{K}([a]\phi) = \mathcal{K}(\forall\!\!\!\forall \phi) = \mathcal{K}(\phi)$, $\mathcal{K}(\forall x.\phi) = \mathcal{K}(\phi) \setminus \{x\}$.

For a clock variable $y \in \mathcal{K}$ that does not appear in the syntax of ϕ and $x \in \mathcal{K}(\phi)$, we denote by $\phi\{y/x\}$ the formula obtained by uniformly substituting all the occurrences of x in ϕ by y.

Definition 5.1 Given two rational interpretations $f_-, f_+ \colon \mathcal{K} \to \mathbb{Q}$ and a bijection $\sigma : \mathcal{K} \to \mathcal{K}$, for any formula $\phi \in \mathcal{L}$ let $\phi +_\sigma {}^{f_-}\!/_{f_+}$ be defined as follows, where $x \bowtie t$ for $t < 0$ should be read as $x \geq 0$:

$$\begin{array}{llll}
\bot +_\sigma {}^{f_-}\!/_{f_+} & \overset{\text{df}}{=} \bot & (\phi \wedge \psi) +_\sigma {}^{f_-}\!/_{f_+} & \overset{\text{df}}{=} (\phi +_\sigma {}^{f_-}\!/_{f_+}) \wedge (\psi +_\sigma {}^{f_-}\!/_{f_+}) \\
(x \leq r) +_\sigma {}^{f_-}\!/_{f_+} & \overset{\text{df}}{=} \sigma(x) \leq (r + f_+(x)) & (x \geq r) +_\sigma {}^{f_-}\!/_{f_+} & \overset{\text{df}}{=} \sigma(x) \geq (r + f_-(x)) \\
(\neg\phi) +_\sigma {}^{f_-}\!/_{f_+} & \overset{\text{df}}{=} \neg(\phi +_\sigma {}^{f_+}\!/_{f_-}) & ([a]\phi) +_\sigma {}^{f_-}\!/_{f_+} & \overset{\text{df}}{=} [a](\phi +_\sigma {}^{f_-}\!/_{f_+}) \\
(\forall\!\!\!\forall\phi) +_\sigma {}^{f_-}\!/_{f_+} & \overset{\text{df}}{=} \forall\!\!\!\forall(\phi +_\sigma {}^{f_-}\!/_{f_+}) & (\forall x.\phi) +_\sigma {}^{f_-}\!/_{f_+} & \overset{\text{df}}{=} \forall\sigma(x).(\phi +_\sigma {}^{f_-[x \mapsto 0]}\!/_{f_+[x \mapsto 0]})
\end{array}$$

Whenever $f_- = f_+ = f$, we write $+_\sigma f$; whenever σ is the identity on $\mathcal{K}$, we write $+ {}^{f_-}\!/_{f_+}$.

The following lemma, which can be proved by induction over the structure of formulas, and its corollaries characterize the relationships between the formulas satisfied by the same model under different interpretations.

Lemma 5.2 *Let $M = (M, \theta, \oplus)$ be a TTS, $\sigma : \mathcal{K} \to \mathcal{K}$, $\delta : \mathcal{K} \to \mathbb{R}$ and $f_-, f_+ : \mathcal{K} \to \mathbb{Q}$ s.t. $f_- \leq \delta \leq f_+$. Then for any $m \in M$, $\phi \in \mathcal{L}$ and $i \in \Im(\mathcal{K})$, if $i + \delta \geq 0$ then:*

$$M, m, i \models \phi \implies M, m, (i + \delta) \circ \sigma^{-1} \models \phi +_\sigma {}^{f_-}\!/_{f_+}.$$

The implication from right to left is not always true, since f_- and f_+ are approximations of δ. However we have equivalences in some concrete cases.

Corollary 5.3 *Let $\mathcal{M} = (M, \theta, \oplus)$ be a TTS and $f : \mathcal{K} \to \mathbb{Q}$. Then for any $m \in M$, $\phi \in \mathcal{L}$ and $i \in \Im(\mathcal{K})$, if $i + f \geq 0$ then:*

$$\mathcal{M}, m, i \models \phi \iff \mathcal{M}, m, i + f \models \phi + f.$$

Corollary 5.4 *Let $\mathcal{M} = (M, \theta, \oplus)$ be a TTS and $\delta : \mathcal{K} \to \mathbb{R}$. Then for any $m \in M$, $\phi \in \mathcal{L}$ and $i \in \Im(\mathcal{K})$, if $i + \delta \geq 0$ and for any $x \in \mathcal{K}(\phi)$, $\delta(x) = 0$, then:*

$$\mathcal{M}, m, i \models \phi \iff \mathcal{M}, m, i + \delta \models \phi.$$

With these, we can proceed with the proof of the H-Mp. We say that a TTS $\mathcal{M} = (M, \theta, \oplus)$ has the *finite image property* if for any action $a \in \Sigma$ and any $m \in M$, $\theta(m, a)$ is finite.

Theorem 5.5 (Hennessy-Milner Theorem) *Consider a TTS $\mathcal{M} = (M, \theta, \oplus)$ satisfying the finite image property. Then for any $m, n \in M$:*

$$m \sim n \quad \textit{iff} \quad \forall i \in \Im(\mathcal{K}), \phi \in \mathcal{L}, \mathcal{M}, m, i \models \phi \Leftrightarrow \mathcal{M}, n, i \models \phi.$$

Proof. ($\Leftarrow$): We prove that R is a bisimulation

$$R = \{(m, n) \mid \forall i, \forall \phi, \ \mathcal{M}, m, i \models \phi \Leftrightarrow \mathcal{M}, n, i \models \phi\}.$$

Supp. that $m \xrightarrow{a} m'$. If there exists no $n' \in M$ s.t $n \xrightarrow{a} n'$, then $\mathcal{M}, n, i \models [a]\bot$, implying $\mathcal{M}, m, i \models [a]\bot$ - contradicting the assumption. Let $F = \{n_k \mid n \xrightarrow{a} n_k\}$, which is finite since the TTS is image finite. Suppose that $(m', n_k) \notin R$ for any k. Then, there exists $i_k \in \Im(\mathcal{K})$ and $\phi_k \in \mathcal{L}$ s.t. $\mathcal{M}, m', i_k \models \phi_k$ and $\mathcal{M}, n_k, i_k \models \neg\phi_k$, for any k. For every $x \in \mathcal{K}(\phi_k)$, consider a new variable x_k distinct from all the other variables. Let $\phi'_k = \phi_k\{x_k/x\}$, for every k and let $i' \in \Im(\mathcal{K})$ s.t. $i'(x_k) = i_k(x)$ for any k. We have: $\mathcal{M}, m', i' \models \bigwedge_k \phi'_k$ and $\mathcal{M}, n_k, i' \models \neg\phi'_k$ by Corollary 5.4. Then $\mathcal{M}, m, i' \models \langle a \rangle \bigwedge_k \phi'_k$ and $\mathcal{M}, n, i' \models [a] \bigvee \neg\phi'_k$ - contradiction.

Supp. that $m' = d \oplus m$. For $r \in \mathbb{Q}_{\geq 0}$, $r \geq d$, $\mathcal{M}, m, i[x \mapsto r - d] \models \exists\!\!\!\exists\,(x = r)$, which implies $\mathcal{M}, n, i[x \mapsto r - d] \models \exists\!\!\!\exists\,(x = r)$. Hence, there exists $n' \in M$ s.t. $n' = d \oplus n$. We prove that $(m', n') \in R$. For any $i \in \Im(\mathcal{K})$ and $\phi \in \mathcal{L}$, $\mathcal{M}, m', i \models \phi$ implies, using Corollary 5.4, that $\mathcal{M}, m', i[x \mapsto 0] \models \phi$ for any $x \notin \mathcal{K}(\phi)$; further, applying Corollary 5.3, $\mathcal{M}, m', i[x \mapsto 0] + r \models \phi + r$. Then, $\mathcal{M}, m, i[x \mapsto 0] + r - d \models \exists\!\!\!\exists(x = r \wedge (\phi + r))$ and $\mathcal{M}, n, i[x \mapsto 0] + r - d \models \exists\!\!\!\exists(x = r \wedge (\phi + r))$. Consequently, $\mathcal{M}, n, i[x \mapsto 0] + r - d \models \forall\!\!\!\forall\,(x = r \to (\phi + r))$. This implies that $\mathcal{M}, n', i[x \mapsto 0] + r \models x = r \to (\phi + \lceil d \rceil)$ ($\lceil d \rceil$ denotes the smallest natural number bigger than d), which implies $\mathcal{M}, n', i[x \mapsto 0] + r \models (\phi + r)$. So $\mathcal{M}, n', i + r \models \phi + r$, and using Corollary 5.3, $\mathcal{M}, n', i \models \phi$.

The symmetry of R proves the other cases. □

6 Metatheory for TML

In this section we develop a proof system for TML. We prove that TML is not compact and consequently requires infinitary rules. However, we demonstrate that our proof system is strongly-complete for the TTS-semantics. The completeness proofs consist of the construction of a canonical model. Being the role of interpretations in the semantics of TTS, the canonical model is not constructed from maximal consistent sets, as for other modal logics. This is because a maximal consistent set does not identify a state of a TTS; it identifies a set in the presence of a fixed interpretation. Moreover, the same maximal consistent set might be satisfied by non-bisimilar models under different interpretations. To cope with all this complex situation we propose a new method for constructing canonical models.

6.1 Axiomatization for TML

Modal prefixes are words $w \in Mod^*$ over the alphabet

$$Mod = \{[a] \mid a \in \Sigma\} \cup \{\forall\!\!\!\forall\} \cup \{\forall x. \mid x \in \mathcal{K}\},$$

e.g., $\forall x.[a]\forall\!\!\!\forall[b][c], \forall x.\forall\!\!\!\forall, [a], \varepsilon \in Mod^*$, where ε is the empty word. A *context* is a word formed by a modal prefix $w \in Mod^*$ concatenated with the metavariable X; e.g., $[a]X$, $\forall x.\forall\!\!\!\forall X$, $\forall x.[a]\forall\!\!\!\forall[b][c]X$ are contexts. To emphasize the presence of the metavariable we will use the functional representation of type $C[X]$ for contexts; this will allow us to instantiate the metavariable with elements from $\mathcal{L}$. For example, if $C[X] = \forall x.[a]\forall\!\!\!\forall[b][c]X$ is a context, then $C[(x \geq r)] = \forall x.[a]\forall\!\!\!\forall[b][c](x \geq r) \in \mathcal{L}$. Also $\varepsilon[X]$ is a context - the empty one - and for $\phi \in \mathcal{L}$, $\varepsilon[\phi] = \phi$.

The axiomatic system of TML includes, in addition to the axioms and the rules of propositional logic, the axioms and the rules in Table 1. They are stated for arbitrary $\phi, \psi \in \mathcal{L}$, $r, s, t \in \mathbb{Q}_{\geq 0}$, $a \in \Sigma$, $\Box \in Mod$ and arbitrary context $C[X]$.

(A1): $\vdash x \geq 0$

(A2): $\vdash (x \geq r) \vee (x \leq r)$

(A3): $\vdash x \leq r \to \neg(x \geq s),\ r < s$

(A4): $\vdash x \trianglelefteq r \to [a](x \trianglelefteq r)$

(A5): $\vdash \forall\!\!\!\forall\phi \to \phi \wedge \forall\!\!\!\forall\forall\!\!\!\forall\phi$

(A6): $\vdash \forall\!\!\!\forall\phi \to \forall\!\!\!\forall(r \leq x \leq s \to \phi),\ r \leq s$

(A7): $\vdash x \geq r \to \forall\!\!\!\forall(x \geq r)$

(A8): $\vdash \exists\!\!\!\exists(x = r \wedge \phi) \to \forall\!\!\!\forall(x = r \to \phi)$

(A9): $\vdash \exists\!\!\!\exists(x \leq r \wedge \forall\!\!\!\forall\phi) \to \forall\!\!\!\forall(x \geq r \to \phi)$

(A10): $\vdash \exists\!\!\!\exists(x \leq r) \wedge \exists\!\!\!\exists(x \geq r) \to \exists\!\!\!\exists(x = r)$

(A11): $\vdash x \trianglelefteq r \wedge y \trianglelefteq s \to \forall\!\!\!\forall(x \trianglelefteq r + t \to y \trianglelefteq s + t)$

(A12): $\vdash \forall x.\phi \to \phi + {}^{[x \mapsto r]}/_{[x \mapsto s]},\ r \leq s$

(A13): $\vdash \forall(x \leq r).\forall\!\!\!\forall(x = s \to \phi) \to$
$\forall(x = r).\forall\!\!\!\forall(s \leq x \leq s + r \to \phi)$

(A14): $\vdash \Box(\phi \to \psi) \to (\Box\phi \to \Box\psi)$

(R1): If $\vdash \phi$, then $\vdash \Box\phi$

(R2): $\{C[x \trianglelefteq r] \mid r \triangleright s\} \vdash C[x \trianglelefteq s]$

(R3): $\{C[x \geq r] \mid r \in \mathbb{Q}_{\geq 0}\} \vdash C[\bot]$

(R4): $\{C[\phi + {}^{[x \mapsto r]}/_{[x \mapsto s]}] \mid r \leq s\} \vdash C[\forall x.\phi]$

(R5): $\{C[\forall\!\!\!\forall(x \leq s \to \phi)] \mid s \in \mathbb{Q}_{\geq 0}\} \vdash C[\forall\!\!\!\forall\phi]$

Table 1
The Axiomatic System of TML

A formula ϕ is *provable*, denoted by $\vdash \phi$, if it can be proven from the given axioms and rules. We say that ϕ is *consistent*, if $\phi \to \bot$ is not provable. Given a set Φ of formulas, we say that Φ *proves* ϕ, $\Phi \vdash \phi$, if from the formulas of Φ and from the axioms one can prove ϕ, eventually using Boolean or infinitary-Boolean reasoning. In other words, we assume that the provability is closed under the rule

$$\Phi \cup \{\phi\} \vdash \psi \text{ iff } \Phi \vdash \phi \to \psi,$$

for arbitrary (possibly infinite) sets $\Phi \subseteq \mathcal{L}$. Φ is *consistent* if it is not the case that $\Phi \vdash \bot$.

The axioms (A1)-(A3) state simple facts about the clock values. The axiom (A4) reflects the fact that action-transitions in a TTS happen instantaneously.

The axioms (A5)-(A10) describe the nature of time in TTSs. Thus, (A5) states that the time is linear and 0-delays do not perturb the system; (A6) that the time is persistent; (A7) that the flow of time is unidirectional (the past and the future are disjoint); (A8) that the time is deterministic; (A9) and (A10) that the time is continuous. The axiom (A11) guarantees that all the clocks measure the same time flow.

The axiom (A12) together with the infinitary rule (R4) describe the fact that the information provided by a clock variable x in a formula prefixed by $\forall x$ is superfluous. The role of (A13) is to characterize the interaction between the two types of universal quantifiers.

The axiom (A14) and the rule (R1) state that all the box-like operators of TML are normal.

The rules (R2)-(R5) are infinitary and have instances for any possible context. For instance, the formulas below are instances of the rules (R2) and (R3) respectively.

$$\{[a](x \geq r) \mid r < s\} \vdash [a](x \geq s),$$
$$\{\mathbb{V}[a](x \geq r) \mid r \in \mathbb{Q}_{\geq 0}\} \vdash \mathbb{V}[a]\bot.$$

The rule (R2) reflects the Archimedean property of rationals. (R3) guarantees that the value of any clock is finite in any model under any interpretation.

By induction on the structure of possible proofs, we prove the soundness.

Theorem 6.1 (Soundness) The axiomatic system in Table 1 is sound with respect to the TTS-semantics, i.e., for arbitrary $\Phi \subseteq \phi$ and $\phi \in \mathcal{L}$,

$$\Phi \vdash \phi \text{ implies } \Phi \models \phi.$$

6.2 Non-Compactness of TML

We have seen in Section 4 that the set of validities of TML are not recursively enumerable. This means that any complete axiomatization of TML must be infinitary. Otherwise, we could enumerate all the proofs and the set of provable formulas, which in a complete logic coincides with the set of validities, is recursively enumerable - contradiction!

There is also a model theoretic result that ensures us the necessity of having infinitary rules:

Theorem 6.2 (Non-Compactness of TML) *TML with the TTS-semantics is not compact, i.e., there exists an infinite set $\Phi \subseteq \mathcal{L}$ such that each finite subset of Φ admits a model but Φ does not admit any model.*

Proof. The result derives from the soundness of the infinitary rules and each rule can be used to produce examples of such sets. Consider, for example, $s \in \mathbb{Q}_{\geq 0}$ and the set

$$\Phi = \{x \geq r \mid r < s\} \cup \{x < s\}.$$

Since for any $i \in \Im(\mathcal{K})$, $i(x) \geq r$ for each $r < s$ implies $i(x) \geq s$, Φ does not admit any model. However, it is not difficult to construct a model for just any finite subset of Φ. □

6.3 Canonical Model and Completeness

In this section we prove that the axiomatic system of TML is not only sound, but also complete for the TTS-semantics, meaning that for arbitrary $\Phi \subseteq \mathcal{L}$ and $\phi \in \mathcal{L}$, $\Phi \models \phi$ iff $\Phi \vdash \phi$. To complete this proof it is sufficient to show that any consistent formula has a model. In the following we construct a canonical model, which is a TTS such that each consistent formula is satisfied at some state under some interpretation. In modal logics such a construction is usually done using maximally consistent sets of formulas as states.

For some set $\Lambda \subseteq \mathcal{L}$, we say that $\Phi \subseteq \mathcal{L}$ is Λ-*maximally consistent* if Φ is consistent and no formula of Λ can be added to Φ without making it inconsistent. Φ is *maximally-consistent* if it is $\mathcal{L}$-maximally-consistent.

The aforementioned technique to construct canonical models cannot be applied directly for TML because to the same state of a given TTS corresponds different maximally-consistent sets of formulas under different interpretations. We generalize this construction to cope with the complexity of TML. To the best of our knowledge, the following construction is original.

For the beginning, we observe that given a maximally-consistent set of formulas, the information contained about a given clock is complete.

Let Ω be the set of $\mathcal{L}$-maximally consistent sets.

Lemma 6.3 *For arbitrary $\Lambda \in \Omega$ and $x \in \mathcal{K}$,*

$$\sup\{r \in \mathbb{Q}^+ \mid x \geq r \in \Lambda\} = \inf\{r \in \mathbb{Q}^+ \mid x \leq r \in \Lambda\} \in \mathbb{R}_{\geq 0}.$$

Proof. Let $A = \{r \in \mathbb{Q}^+ \mid x \geq r \in \Lambda\}$ and $B = \{r \in \mathbb{Q}^+ \mid x \leq r \in \Lambda\}$. (A1) guarantees that $A \neq \emptyset$ and if $B = \emptyset$, we can derive a contradiction from (R3) for $C[X] = X$.

Since the two sets are non-empty, the sup and inf exist. Moreover, (R3) can also be used to prove that $\sup A < \infty$. Let $\sup A = u$ and $\inf B = v$. If $u < v$, there exists $r \in \mathbb{Q}^+$ such that $u < r < v$. Hence, $x \leq r \in \Lambda$, which contradicts $r \leq v$. If $v < u$, there exists $r_1, r_2 \in \mathbb{Q}^+$ such that $v < r_1 < r_2 < u$. Hence, $x \leq r_i, x \geq r_i \in \Lambda$ for $i = 1, 2$. Since $r_2 - r_1 > 0$, (A3)$\vdash x \geq r_2 \rightarrow \neg(x \leq r_1)$, which proves the inconsistency of Λ - contradiction. □

The previous lemma demonstrates that to each maximally-consistent set corresponds a unique interpretation of clocks that we will identify in what follows using the function $\mathscr{I} : \Omega \longrightarrow \Im(\mathcal{K})$ defined for arbitrary $\Lambda \in \Omega$ and $x \in \mathcal{K}$ by

$$\mathscr{I}(\Lambda)(x) = \sup\{r \in \mathbb{Q}^+ \mid x \geq r \in \Lambda\} \in \mathbb{R}_{\geq 0}.$$

Since $\mathscr{I}(\Lambda)$ synthesize only the information in Λ regarding the clocks, there exist disjoint sets $\Lambda_1, \Lambda_2 \in \Omega$ s.t. $\mathscr{I}(\Lambda_1) = \mathscr{I}(\Lambda_2)$; this equality defines an equivalence relation on Ω and the equivalence classes are in one to one correspondence with the interpretations in $\Im(\mathcal{K})$.

Observe that any state in a model of TML corresponds to a function from $\Im(\mathcal{K})$ to Ω: given a model, each interpretation identifies a maximally-consistent set of formulas satisfied by that model under the given interpretation. Consequently, to construct the canonical model we will have to take as states not maximally-consistent sets of formulas, but functions from interpretations to maximally-consistent sets. However, not just any function $\gamma : \Im(\mathcal{K}) \to \Omega$ is a good candidate for a model, because between the maximally-consistent sets associated to a model under different interpretation there are certain coherence conditions as the ones described in Lemma 5.2 and corollaries 5.3 and 5.4. These coherences are formally described in the next definition.

Definition 6.4 A function $\gamma : \Im(\mathcal{K}) \to \Omega$ is *coherent*, if for any $i \in \Im(\mathcal{K})$, any bijection σ on $\mathcal{K}$, any $\delta : \mathcal{K} \to \mathbb{R}$ s.t. $i + \delta \geq 0$, and any $f_-, f_+ : \mathcal{K} \to \mathbb{Q}$ s.t. $f_- \leq \delta \leq f_+$,
1. $(\mathscr{I} \circ \gamma)(i) = i$;
2. if $\phi \in \gamma(i)$, then $\phi +_\sigma {}^{f_-}/_{f_+} \in \gamma((i + \delta) \circ \sigma^{-1})$.

The first fundamental result that we prove is that any maximally-consistent set Λ belongs to the image $\gamma(\Im(\mathcal{K}))$ of some coherent function γ. After constructing the canonical model on the set of coherent functions, this result will guarantee that any maximally-consistent set is satisfied by some model under some interpretation.

Lemma 6.5 *For any $\Lambda \in \Omega$, there exists a coherent function γ s.t. $\gamma(\mathscr{I}(\Lambda)) = \Lambda$.*

Proof. We say that a set $S \subseteq \Omega$ is *coherent* if there exists a set $T \subseteq \Im(\mathcal{K})$ and a bijection $\gamma : T \to S$ that satisfies the two conditions of Definition 6.4. S is *maximally-coherent* if $T = \Im(\mathcal{K})$. Observe that the bijection that defines a maximally-coherent set is a coherent function in the sense of Definition 6.4.

We prove, using a transfinite induction, that any coherent set has a maximally-coherent extension. And this proves the existence of γ, because $\{\Lambda\}$ is coherent.

I. Firstly, observe that $S \subseteq \Omega$ is coherent iff for any $\Lambda_1, \Lambda_2 \in \Omega$, with $i_s = \mathscr{I}(\Lambda_s)$, $s = 1, 2$,

$$\Lambda_1 + (i_2 - i_1) \subseteq \Lambda_2 \text{ and } \Lambda_2 + (i_1 - i_2) \subseteq \Lambda_1,$$

where for arbitrary $\Lambda' \in \Omega$ and $\delta : \mathcal{K} \to \mathbb{R}$,

$$\Lambda' + \delta = \{\phi + {}^{f_-}/_{f_+} \mid f_-, f_+ : \mathcal{K} \to \mathbb{Q}, f_- \leq \delta \leq f_+\}.$$

Moreover, $\Lambda_1 + (i_2 - i_1) \subseteq \Lambda_2$ iff $\Lambda_2 + (i_1 - i_2) \subseteq \Lambda_1$.

II. If $i = \mathscr{I}(\Lambda)$ and $i' \in \Im(\mathcal{K})$, then there exists $\Lambda' \in \Omega$ s.t. $\mathscr{I}(\Lambda') = i'$ and $\{\Lambda, \Lambda'\}$ is coherent. To prove this, we firstly use Definition 5.1 and the axioms to prove that $\Lambda + (i' - i)$ is consistent. Then, we observe that all the infinitary rules have countable sets of instances, which allows us to apply Rasiowa-Sikorski lemma to conclude that $\Lambda + (i' - i)$ must have a maximal-consistent extension Λ'. Since $\Lambda + (i' - i) \subseteq \Lambda'$, we also have $\Lambda' + (i - i') \subseteq \Lambda$. Hence, $\{\Lambda, \Lambda'\}$ is coherent.

III. If $S = \{\Lambda_0, \Lambda_1, ..\Lambda_k\}$ is a finite coherent set, $i_s = \mathscr{I}(\Lambda_s)$, $s = \overline{1..k}$ and $j \in \Im(\mathcal{K})$, then there exists $\Lambda_j \in \Omega$ s.t. $\mathscr{I}(\Lambda_j) = j$ and $S \cup \{\Lambda_j\}$ is coherent. Supp. $\Lambda = \Lambda_0$. It is sufficient to show that the following set is consistent, which is not a difficult task:

$$X = \bigcup_{s=\overline{0..k}} (\Lambda_s + (j - i_s)).$$

IV. If $S = \{\Lambda_s \mid s \in \mathbb{N}\}$ is a countable coherent set (supp. $\Lambda = \Lambda_0$) and $j \in \Im(\mathcal{K})$. There exists $\Lambda_j \in \Omega$ s.t. $S \cup \{\Lambda_j\}$ is coherent. As before, we need to prove that

$$X = \bigcup_{s \in \mathbb{N}} (\Lambda_s + (j - i_s))$$

is consistent. Let $X_t = \bigcup_{s \leq t} (\Lambda_s + (j - i_s))$ for $t \in \mathbb{N}$. We have previously proved that X_t is consistent for any $t \in \mathbb{N}$. Moreover, observe that $X = \bigcup_{t \in \mathbb{N}} X_t$.

Suppose that X is not consistent. If the contradiction can be proven without involving the infinitary rules, then there must exist one X_t that contains all the formulas in the proof, which contradicts the consistency of X_t. Consequently, if X is inconsistent it is because, when t goes to infinity, the sets $X_t \setminus X_{t-1}$ collect the left-hand side of some of our infinitary rules.

If for arbitrary contexts C, formulas $\phi \in \mathcal{L}$, $r \in \mathbb{Q}_{\geq 0}$ and $x \in \mathcal{K}$ we denote by
$\mathcal{L}_1(C, x) = \{C[x \leq s], C[x \geq s] \mid s \in \mathbb{Q}_{\geq 0}\}$
$\mathcal{L}_2(C, x, \phi) = \{C[\phi + {}^{[x \mapsto s]}/_{[x \mapsto s']}] \mid s, s' \in \mathbb{Q}, s \leq s'\}$,
$\mathcal{L}_3(C, x, \phi, r) = \{C[\forall\!\!\!\forall (r \leq x \leq s \to \phi)] \mid r, s \in \mathbb{Q}_{\geq 0}, r \leq s\}$,
one can prove that for each $t \in \mathbb{N}$, X_t is $\mathcal{L}_1(C, x)$-maximally consistent and for each L of type $\mathcal{L}_2(C, x, \phi)$ or $\mathcal{L}_3(C, x, \phi, r)\}$, either $X_t \cap L = \emptyset$ or X_t is L-maximally consistent. This demonstrates that if in X we can use some infinitary rule, it must be used within some X_t for some $t \in \mathbb{N}$. Consequently, the consistency of X_t proves the consistency of X.

V. If $S \subseteq \Omega$ is an uncountable coherent set and $j \in \Im(\mathcal{K})$, then there exists $\Lambda_j \in \Omega$ s.t. $S \cup \{j\}$ is coherent. Supp. $S = \{\Lambda_s \mid s \in \mathbb{R}\}$, $\Lambda = \Lambda_0$ and $i_s = \mathscr{I}(\Lambda_s)$. As in the other cases it is sufficient to prove that the following set is consistent:

$$Y = \bigcup_{s \in \mathbb{R}} (\Lambda_s + (j - i_s)).$$

Since $Y \subseteq \mathcal{L}$ and $\mathcal{L}$ is countable, Y is countable and consequently we can assume that

$$Y = \bigcup_{s \in \mathbb{N}} (\Lambda_s + (j - i_s)).$$

Further the result is a consequence of IV. □

In what follows we define a TTS using the set

$$\Gamma = \{\gamma : \Im(\mathcal{K}) \to \Omega \mid \gamma \text{ is a coherent function}\}$$

as the support-set of the model and the structure

- $\gamma \xrightarrow{a} \gamma'$ if $[\forall i \in \Im(\mathcal{K}),\ [a]\phi \in \gamma(i) \Rightarrow \phi \in \gamma'(i)]$;
- $\gamma' = d \oplus \gamma$ if $[\forall i,\ \forall\!\!\!\forall\phi \in \gamma(i) \Rightarrow \phi \in \gamma'(i+d)]$.

Theorem 6.6 (Canonical Model) *The tuple $\Gamma = (\Gamma, \theta, \oplus)$ defined above is a TTS.*

Proof. Firstly, we prove that $\gamma_1 = d \oplus \gamma$ and $\gamma_2 = d \oplus \gamma$ implies $\gamma_1 = \gamma_2$.

For any $\forall i \in \Im(\mathcal{K})$ and $x \in \mathcal{K} \setminus \mathcal{K}(\phi)$, $\phi \in \gamma_1(i)$ implies $x = \lceil d \rceil \wedge \phi \in \gamma_1(i[x \mapsto \lceil d \rceil])$, where $\lceil d \rceil$ denote the smallest integer larger than d. Since $\gamma_1 = d \oplus \gamma$,

$$\exists\!\!\!\exists(x = \lceil d \rceil \wedge \phi) \in \gamma(i[x \mapsto \lceil d \rceil] - d),$$

which further implies, due to (A8), that

$$\forall\!\!\!\forall(x = \lceil d \rceil \to \phi) \in \gamma(i[x \mapsto \lceil d \rceil] - d).$$

$x = \lceil d \rceil \to \phi \in \gamma_2(i[x \mapsto \lceil d \rceil])$ since $\gamma_2 = d \oplus \gamma$. Then $\phi \in \gamma_2(i[x \mapsto \lceil d \rceil])$, which implies $\phi \in \gamma_2(i)$ because $x \notin \mathcal{K}(\phi)$. Hence for any $i \in \Im(\mathcal{K})$, $\gamma_1(i) \subseteq \gamma_2(i)$. Similarly, $\gamma_2(i) \subseteq \gamma_2(i)$.

Observe now that due to (A5), $\forall\!\!\!\forall\phi \in \gamma(i)$ implies $\phi \in \gamma(i)$, for any $\gamma \in \Gamma$ and $i \in \Im(\mathcal{K})$. Hence, $0 \oplus \gamma = \gamma$.

Now we prove that $d \oplus (d' \oplus m) \cong (d + d') \oplus m$, i.e. $\exists \gamma_1, \gamma_2$ s.t. $\gamma_1 = d' \oplus \gamma$, $\gamma_2 = d \oplus \gamma_1$ iff $\exists \gamma_3$ s.t. $\gamma_3 = (d + d') \oplus \gamma$.

($\Rightarrow$) Suppose $\exists \gamma_1, \gamma_2$ s.t. $\gamma_1 = d \oplus \gamma$, $\gamma_2 = d' \oplus \gamma_1$. From (A5), $\forall\!\!\!\forall\phi \in \gamma(i)$ implies $\forall\!\!\!\forall\forall\!\!\!\forall\phi \in \gamma(i)$ and since $\gamma_1 = d \oplus \gamma$, $\forall\!\!\!\forall\phi \in \gamma_1(i + d)$. This implies $\phi \in \gamma_2(i + d + d')$ because $\gamma_2 = d' \oplus \gamma_1$. Hence, there exists $\gamma_3 = (d + d') \oplus \gamma$ and, in fact, $\gamma_3 = \gamma_2$.

($\Leftarrow$) Supp. that $\exists \gamma_3$ s.t. $\gamma_3 = (d + d') \oplus \gamma$.

Firstly, we prove that there exists γ_1 s.t. $\gamma_1 = d' \oplus \gamma$ by constructing it. For each $i \in \Im(\mathcal{K})$, $\gamma(i) +_\sigma d' \subseteq \gamma_1(i + d')$. It is not difficult to verify that for arbitrary $i, j \in \Im(\mathcal{K})$, $\gamma(i) +_\sigma d' + (j - i) \subseteq \gamma(j) +_\sigma d'$ and

$$\bigcup_{x \in \mathcal{K}} \{x \trianglelefteq r \mid r \in \mathbb{Q}^+, i(x) \trianglelefteq r\} \subseteq \gamma(i) +_\sigma d'.$$

Now, we can use a similar transfinite construction as in Lemma 6.5 to construct entire γ_1.

Secondly, we need to prove that there exists $d \oplus \gamma_1$. For any $i \in \Im(\mathcal{K})$, $\forall\!\!\!\forall\phi \in \gamma_1(i)$ implies $x \leq r \wedge \forall\!\!\!\forall\phi \in \gamma_1(i)$ for $r \in \mathbb{Q}^+$ s.t. $i(x) \leq r \leq i(x) + d$. Then,

$$x \leq r + \lceil d' \rceil \wedge \forall\!\!\!\forall(\phi + \lceil d' \rceil) \in \gamma_1(i + \lceil d' \rceil).$$

Now because $\gamma_1 = d' \oplus \gamma$, we obtain

$$\exists\!\!\!\exists(x \leq r + \lceil d' \rceil \wedge \forall\!\!\!\forall(\phi + \lceil d' \rceil)) \in \gamma(i + \lceil d' \rceil - d').$$

From this we get, by applying (A9), that

$$\forall\!\!\!\forall(x \geq r + \lceil d' \rceil \to \forall\!\!\!\forall(\phi + \lceil d' \rceil)) \in \gamma(i + \lceil d' \rceil - d').$$

Since $\gamma_3 = (d + d') \oplus \gamma$,

$$x \geq r + \lceil d' \rceil \to \forall\!\!\!\forall(\phi + \lceil d' \rceil) \in \gamma_3(i + \lceil d' \rceil + d).$$

And because $r \leq i(x) + d$, $x \geq r + \lceil d' \rceil \in \gamma_3(i + \lceil d' \rceil + d)$. We then get that

$\forall\!\!\!\forall(\phi + \lceil d' \rceil) \in \gamma_3(i + \lceil d' \rceil + d)$, implying $\forall\!\!\!\forall\phi \in \gamma_3(i + d)$, which using (A5) gives $\phi \in \gamma_3(i + d)$. Hence, there exists $d \oplus \gamma_1 = \gamma_3$. □

Lemma 6.7 (Truth Lemma) *For any $\psi \in \mathcal{L}$, $i \in \Im(\mathcal{K})$ and $\gamma \in \Gamma$,*

$$\Gamma, \gamma, i \models \psi \textit{ iff } \psi \in \gamma(i).$$

Proof. Induction on ψ.

[The case $\psi = \forall x.\phi$]: $\Gamma, \gamma, i \models \forall x.\phi$ iff $\Gamma, \gamma, i[x \mapsto u] \models \phi$ for any $u \in \mathbb{R}_{\geq 0}$, i.e., $\phi \in \gamma(i[x \mapsto u])$ for all $u \in \mathbb{R}_{\geq 0}$.
$(\Longrightarrow)$ $\phi \in \gamma(i[x \mapsto u])$ implies $\phi + {}^{[x \mapsto r]}/_{[x \mapsto s]} \in \gamma(i)$ for all $r, s \in \mathbb{Q}$ s.t $r \leq u - i(x) \leq s$. This must happen for any $u \in \mathbb{R}_{\geq 0}$, so for any $r, s \in \mathbb{Q}$, $\phi + {}^{[x \mapsto r]}/_{[x \mapsto s]} \in \gamma(i)$. Hence, $\forall x.\phi \in \gamma(i)$ by rule (R4).
$(\Longleftarrow)$ Using (A12), $\forall x.\phi \in \gamma(i)$ implies $\phi + {}^{[x \mapsto r]}/_{[x \mapsto s]} \in \gamma(i[x \mapsto u)$ for any $u \in \mathbb{R}_{\geq 0}$ and $r, s \in \mathbb{Q}$ s.t. $r \leq u - i(x) \leq s$. Consequently, $\forall x.\phi \in \gamma(i[x \mapsto u])$ by (R4). And (A12), $\phi \in \gamma(i[x \mapsto u])$ for all $u \in \mathbb{R}_{\geq 0}$.

[The case $\psi = \forall\!\!\!\forall\phi$]: $\Gamma, \gamma, i \models \forall\!\!\!\forall\phi$ iff [for any $d \in \mathbb{R}_{\geq 0}$ and $\gamma' \in \Gamma$ s.t. $\gamma' = d \oplus \gamma$, $\Gamma, \gamma', i + d \models \phi$] iff $\phi \in \gamma'(i + d)$ by the inductive hypothesis.
$(\Longrightarrow)$ $\phi \in \gamma'(i + d)$ implies $(x = r) \wedge \phi \in \gamma'((i + d)[x \mapsto r])$, for any $r \in \mathbb{Q}^+$, $r \geq d$. Applying (A8), $\forall\!\!\!\forall(x = r \to \phi) \in \gamma(i[x \mapsto r - d])$, for any $r \in \mathbb{Q}^+$, $r \geq d$. This implies $x \leq r \to \forall\!\!\!\forall(x = r \to \phi) \in \gamma(i[x \mapsto u])$ for any $u \in \mathbb{R}_{\geq 0}$. Then, $\forall(x \leq r).\forall\!\!\!\forall(x = r \to \phi) \in \gamma(i)$. Applying (A13), $\forall(x = r).\forall\!\!\!\forall(r \leq x \leq 2r \to \phi) \in \gamma(i)$, which further implies $\forall(x = 0).\forall\!\!\!\forall(0 \leq x \leq r \to \phi) \in \gamma(i)$ by (A12) and (R4). Using (R5), $\forall(x = 0).\forall\!\!\!\forall(x \geq 0 \to \phi) \in \gamma(i)$, which implies $\forall(x = 0).\forall\!\!\!\forall(x \geq 0 \to \phi) \in \gamma(i[x \mapsto 0])$. So, $x = 0 \to \forall\!\!\!\forall(x \geq 0 \to \phi) \in \gamma(i[x \mapsto 0])$, which implies $\forall\!\!\!\forall(x \geq 0 \to \phi) \in \gamma(i[x \mapsto 0])$. Using (A1), $\forall\!\!\!\forall\phi \in \gamma(i[x \mapsto 0])$ and further, $\forall\!\!\!\forall\phi \in \gamma(i)$.
$(\Longleftarrow)$ derives from the definition of $\oplus$.

[The case $\psi = [a]\phi$]: $\Gamma, \gamma, i \models [a]\phi$ iff for any $\gamma' \in \Gamma$ s.t. $\gamma \xrightarrow{a} \gamma'$, $\Gamma, \gamma', i \models \phi$, iff $\phi \in \gamma'(i)$ by induction.
$(\Longrightarrow)$ Supp. $\langle a \rangle \neg\phi \in \gamma(i)$. Let $A = \{\gamma' \mid \gamma \xrightarrow{a} \gamma'\}$, $B_i = \{\neg\phi\} \cup \{\psi \mid [a]\psi \in \gamma(i)\} \cup \Delta_i$ and $B_j = \{\psi \mid [a]\psi \in \gamma(j)\} \cup \Delta_i$ for any $j \neq i$, where $\Delta_k = \bigcup_{x \in \ K}\{x \leq r \mid r \geq i(x)\} \cup \{x \geq r \mid r \leq i(x)\}$.
$\{\psi \mid [a]\psi \in \gamma(i)\} \cup \Delta_i$ and B_j, $j \neq i$ are consistent.
Suppose that B_i is inconsistent. Then, there exists a set $F \subseteq B_i$ s.t. $F \vdash \phi$. If F is finite, (R1) guarantees that $[a]F \vdash [a]\phi$, where $[a]F = \{[a]\rho \mid \rho \in F\}$. Otherwise, $F \vdash \phi$ is (modulo Boolean reasoning possible involving infinite meets) an instance of one of the rules (R2)-(R5); in all these cases, $[a]F \vdash [a]\phi$ is an instance of the same rule for the context $C[X] = [a]X$. Since $F \subseteq B_i$, $[a]F \subseteq \gamma(i)$ implying $[a]\phi \in \gamma(i)$, which contradicts the consistency of $\gamma(i)$. Hence, B_i is consistent.
Now we prove that for any $j, j' \in \mathfrak{I}(K)$, B_j and $B_{j'}$ are such that $B_j + (j' - j) \subseteq B_{j'}$. If $j \neq i$, then for arbitrary $\rho \in B_j$ either $[a]\rho \in \gamma(j)$, or $\rho = x \trianglelefteq r$. In the first case, $[a]\rho +_\sigma {}^{f_-}/_{f_+} \in \gamma(j')$, for all $f_- \leq j' - j \leq f_+$. So, $\rho +_\sigma {}^{f_-}/_{f_+} \in B_{j'}$. In the second case, since $\rho = x \trianglelefteq r$ is closed under any interpretation transformation, for

any $f_- \leq j' - j \leq f_+$, $\rho +_\sigma {}^{f_-}/_{f_+} \in B_{j'}$.
If $j = i$, consider an arbitrary $\rho \in B_j$. If $\rho \neq \neg\phi$, we get a similar case as above. Otherwise, $\langle a \rangle \rho \in \gamma(i)$, which implies $\langle a \rangle \rho +_\sigma {}^{f_-}/_{f_+} \in \gamma(j)$ for all $f_- \leq j - i \leq f_+$. So, $\rho +_\sigma {}^{f_-}/_{f_+} \in B_j$.
At this point we can use a similar strategy as in Theorem 6.5 to prove that there exists $\gamma'' \in \Gamma$ s.t. for any $j \in \mathfrak{I}(K)$, $B_j \subseteq \gamma''(j)$. But then, $\gamma'' \in A$, which implies $\phi \in \gamma''(i)$ - contradiction!
Hence, $[a]\phi \in \gamma(i)$.
$(\Longleftarrow)$ derives from the definition of θ. □

Corollary 6.8 *If $\Phi \subseteq \mathcal{L}$ is consistent, there exists $\gamma \in \Gamma$ and $i \in \Im(\mathcal{K})$ s.t. $\Gamma, \gamma, i \models \Phi$.*

Proof. Because the infinitary rules of TML have countable sets of instances, the Rasiowa-Sikorski lemma guarantees the existence of some $\Lambda \in \Omega$ s.t. $\Lambda \supseteq \Phi$. Lemma 6.5 guarantees that there exists $\gamma \in \Gamma$ s.t. $\Lambda = \gamma(\mathscr{I}(\Lambda))$. Hence $\Gamma, \gamma, \mathscr{I}(\Lambda) \models \Phi$ by applying Lemma 6.7. □

Corollary 6.8 is a well known equivalent formulation of the strong completeness theorem.

Theorem 6.9 (Strong Completeness) *TML is strongly-complete with respect to the TTS-semantics, i.e., for arbitrary $\Phi \subseteq \mathcal{L}$ and $\phi \in \mathcal{L}$,*

$$\Phi \models \phi \quad \textit{implies } \Phi \vdash \phi.$$

7 Conclusions

In this paper we addressed and solved a series of open problems regarding the timed logics and real-time systems. We develop the metatheory for the Timed Modal Logic (TML), which is the most basic logic used in practice for specifying and analyzing timed transition systems (TTSs).

In this paper we demonstrate that TML is adequate for the TTS semantics. We show that its satisfiability problem is undecidable and the set of the validities is not recursively enumerable. Despite this, we develop a strongly-complete proof system for TML. Because TML is not compact, the proof system contains necessarily infinitary rules and these rules also explain why the set of validities is not recursively enumerable. Our axioms characterize the concept of time used in the definition of TTS. Our completeness proof is based on a novel method that generalizes the classic filtration technique used in modal logics for the construction of canonical models. Essential in the proof is the use of the Rasiowa-Sikorski lemma.

All these results open new perspective on real-time systems and on their analysis and reveal new research directions. The proof system contains similar axioms to those of other well-known logics, which makes one think about other possible connections and research perspectives.

References

[1] Luca Aceto, Anna Ingólfsdóttir, Kim Guldstrand Larsen, and Jiri Srba. *Reactive Systems: modelling, specification and verification.* Cambridge University Press, 2007.

[2] Rajeev Alur, Costas Courcoubetis, and David L. Dill. Model-checking in dense real-time. *Information and Computation*, 104(1):2–34, May 1993.

[3] Rajeev Alur, Costas Courcoubetis, and David L. Dill. Model-checking in dense real-time. *Inf. Comput.*, 104(1):2–34, 1993.

[4] Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, April 1994.

[5] Rajeev Alur, Tómas Feder, and Thomas A. Henzinger. The benefits of relaxing punctuality. *Journal of the ACM*, 43(1):116–146, January 1996.

[6] Rajeev Alur and Thomas A. Henzinger. Logics and models of real time: A survey. In *REX Workshop*, volume 600, pages 74–106. Springer, 1991.

[7] Rajeev Alur and Thomas A. Henzinger. Real-time logics: Complexity and expressiveness. *Information and Computation*, 104(1):35–77, May 1993.

[8] Rajeev Alur and Thomas A. Henzinger. A really temporal logic. *Journal of the ACM*, 41(1):181–203, January 1994.

[9] Howard Barringer, Ruurd Kuiper, and Amir Pnueli. A really abstract concurrent model and its temporal logic. In *POPL*, pages 173–183. ACM Press, 1986.

[10] R. Goldblatt. On the role of the Baire category theorem in the foundations of logic. *Journal of Symbolic logic*, pages 412–422, 1985.

[11] Thomas A. Henzinger. Half-order modal logic: How to prove real-time properties. In Cynthia Dwork, editor, *PODC*, pages 281–296. ACM, 1990.

[12] Thomas A. Henzinger, Xavier Nicollin, Joseph Sifakis, and Sergio Yovine. Symbolic model checking for real-time systems. In *LICS*, pages 394–406, 1992.

[13] Ron Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Systems*, 2(4):255–299, 1990.

[14] Dexter Kozen, Kim G. Larsen, Radu Mardare, and Prakash Panangaden. Stone duality for markov processes. In *LICS*, pages 321–330. IEEE Computer Society, 2013.

[15] Salvatore La Torre and Margherita Napoli. A decidable dense branching-time temporal logic. In *FSTTCS'00*, volume 1974, pages 139–150. Springer-Verlag, 2000.

[16] François Laroussinie, Kim Guldstrand Larsen, and Carsten Weise. From timed automata to logic - and back. In *MFCS*, volume 969, pages 529–539. Springer, 1995.

[17] Joël Ouaknine and James Worrell. On the decidability of metric temporal logic. In *LICS'05*, pages 188–197. IEEE Comp. Soc. Press, 2005.

[18] Joël Ouaknine and James Worrell. On metric temporal logic and faulty Turing machines. In *FoSSaCS'06*, volume 3921, pages 217–230. Springer-Verlag, 2006.

[19] Joël Ouaknine and James Worrell. On the decidability and complexity of metric temporal logic over finite words. *Logicical Methods in Computer Science*, 3(1), March 2007.

[20] H. Rasiowa and R. Sikorski. A proof of the completeness theorem of Gödel. *Fund. Math*, 37:193–200, 1950.

[21] Jean-François Raskin and Pierre-Yves Schobbens. Real-time logics: Fictitious clock as an abstraction of dense time. In *TACAS*, volume 1217, pages 165–182. Springer, 1997.

[22] Pierre-Yves Schobbens, Jean-François Raskin, and Thomas A. Henzinger. Axioms for real-time logics. *Theor. Comput. Sci.*, 274(1-2):151–182, 2002.

Appendix

This appendix contains the details of the proofs of the major results presented in the paper.

Proof. [Proof of Theorem 4.2] Our reduction is similar to the one in [3]. The question as to whether a non-deterministic 2-counter machine has a computation with the initial location being visited infinitely often is Σ_1^1-hard. We show how to reduce this problem into a TML satisfiability question.

Let M be a 2-counter machine with counters X and Y and with $n+1$ program instructions $\ell_0, \ldots, \ell_n$. The instruction ℓ_n represents termination, and each instruction ℓ_i $(i < n)$ is either an increment of the form $[\ell_i : X := X + 1; \text{goto } \ell_j]$ or a decrement of the form $[\ell_i : \text{if } X \neq 0 \text{ then } X := X - 1; \text{goto } \ell_j \text{ else goto } \ell_k]$ or a non-deterministic jump of the form $[\ell_i : \text{goto } \ell_j \text{ or } \ell_k]$. A configuration of M is a triple $\langle \ell_i, x, y\rangle$, where x and y are natural numbers representing the current values of X and Y. A computation of M is a "valid" sequence of configurations starting in $\langle \ell_0, 0, 0\rangle$ and ending in some configuration of the type $\langle \ell_n, x, y\rangle$.

We encode the computation of M in TML using the actions $\Sigma = \{\ell_0, \ldots, \ell_n, X, Y\}$. We say that a state m of a given TTS $\mathcal{T}$ encodes the configuration $\langle \ell_i, x, y\rangle$ in the interval $[a, b)$, with $a, b \in \mathbb{R}^+$ and $a < b$, iff the following holds:

- $(d \oplus m) \xrightarrow{X}$ for exactly x distinct time-points d in (a, b);
- $(d \oplus m) \xrightarrow{Y}$ for exactly y distinct time-points d in (a, b);
- $(d \oplus m) \xrightarrow{\ell_i}$ for $d = a$;
- $(d \oplus m) \not\xrightarrow{\ell_j}$ for all time-points d in (a, b) and for all j.

Let $\{\langle \ell_{i(j)}, x_j, y_j\rangle : j \geq 0\}$ be a computation of M (i.e. $i(0) = 0$). We may construct a (closed) TML formula ϕ^M, such that for any TTS $\mathcal{T}$ and any state m, $\mathcal{T}, m \models \phi$ if and only if m encodes the j'th configuration over the interval $[j, j+1)$ for all $j \geq 0$. Also ϕ^M will ensure that m encodes a computation where ℓ_0 is visited infinitely often. The formula ϕ^M is obtained as the conjunction of a formula expressing the initial configuration, a formula expressing infinite repetition of the initial location and a formula for each instruction of M ensuring that states $(d \oplus m)$ of $\mathcal{T}$ being separated by a delay of 1 correctly encodes the given instruction. The following formula ensures that instructions of M are unique:

(i) $\forall\!\!\!\forall(\langle \ell_i\rangle \top \rightarrow [\ell_j] \perp)$, for all $i \neq j$ (uniqueness);

The initial configuration may be encoded as the conjunction of the following TML formulae:

(i) $\langle \ell_0\rangle \top$;

(ii) x in $\forall\!\!\!\forall(0 < x < 1 \rightarrow [\ell_i] \perp)$, for all $i \geq 0$;

(iii) x in $\forall(0 < x < 1 \rightarrow ([X] \perp \wedge [Y] \perp))$.

An increment statement of the form $[\ell_i : X := X + 1; \text{goto } \ell_j]$ is reflected by the conjunction of the following formulae relating the behaviour in a unit-interval $\{(d \oplus m) : d \in [j, j+1)\}$ with $(j \oplus m) \xrightarrow{\ell_i}$ with the successor unit-interval:

(i) $\forall\big(\langle \ell_i \rangle \top \rightarrow x \text{ in } \exists(x = 1 \wedge \langle \ell_j \rangle)\top\big)$;

(ii) $\forall\big(\langle \ell_i \rangle \top \rightarrow x \text{ in } \forall(0 < x < 1 \rightarrow [\ell_j] \perp \big)$, for all i, j;

(iii) $\forall\big[\langle \ell_i \rangle \top \rightarrow x \text{ in } \forall(x < 1 \wedge \langle X \rangle \top \rightarrow y \text{ in } \exists(y = 1 \wedge \langle X \rangle \top))\big]$;

(iv) $\forall\big[\langle \ell_i \rangle \top \rightarrow$

$$\begin{aligned}
&x \text{ in } \forall(x < 1 \wedge \langle X \rangle \top \wedge y \text{ in } \forall(y > 0 \wedge x < 1 \rightarrow [X] \perp) \\
&\qquad \rightarrow \\
&\qquad\quad y \text{ in } (\exists(y > 1 \wedge x < 2 \wedge \langle X \rangle \top)) \wedge \\
&\qquad\quad y \text{ in } (\forall(y > 1 \wedge x < 2 \wedge \langle X \rangle \top \rightarrow z \text{ in } \forall(z > 0 \wedge x < 2 \rightarrow [X] \perp))))\big]
\end{aligned}$$

Here (1) encodes the goto of the instruction. (2) ensures that ℓ_i actions are only possible at integer-points. (3) ensures that all X actions in the interval $[j, j+1)$ are copied to the successor interval $[j+1, j+2)$. The most involved formula (4) ensures that exactly one additional X action is inserted in $[j+1, j+2)$ after the copy of the last X action in $[j, j+1)$. The formulae for decrement and non-deterministic choice are similar (and simpler). Infinite repetition of ℓ_0 is easily expressed as $\forall[\langle \ell_0 \rangle \rightarrow x \text{ in } \exists(x > 0 \wedge \langle \ell_0 \rangle \top)]$. □

Proof. [Proof of Lemma 5.2] Induction on ϕ.

[The case $x \geq r$]: $(x \geq r) +_\sigma {}^{f_-}/_{f_+} = \sigma(x) \geq (r + f_-(x))$. $M, m, i \models x \geq r$ implies $i(x) \geq r$. Then $((i+\delta) \circ \sigma^{-1})(x) \geq r + \delta(x) \geq r + f_-(x)$. Hence $M, m, (i+\delta) \circ \sigma^{-1} \models (x \geq r) +_\sigma {}^{f_-}/_{f_+}$.

[The case $x > r$]: $(x > r) +_\sigma {}^{f_-}/_{f_+} = \sigma(x) > (r + f_-(x))$. $M, m, i \models x > r$ implies $i(x) > r$. Then $((i+\delta) \circ \sigma^{-1})(x) > r + \delta(x) \geq r + f_-(x)$. Hence $M, m, (i+\delta) \circ \sigma^{-1} \models (x > r) +_\sigma {}^{f_-}/_{f_+}$.

[The case $x \leq r$]: $(x \leq r) +_\sigma {}^{f_-}/_{f_+} = \sigma(x) \leq (r + f_+(x))$. $M, m, i \models x \leq r$ implies $i(x) \leq r$. Then $((i+\delta) \circ \sigma^{-1})(x) \leq r + \delta(x) \leq r + f_+(x)$. Hence $M, m, (i+\delta) \circ \sigma^{-1} \models (x \leq r) +_\sigma {}^{f_-}/_{f_+}$.

[The case $x < r$]: $(x < r) +_\sigma {}^{f_-}/_{f_+} = \sigma(x) < (r + f_+(x))$. $M, m, i \models x < r$ implies $i(x) < r$. Then $((i+\delta) \circ \sigma^{-1})(x) < r + \delta(x) \geq r + f_+(x)$. Hence $M, m, (i+\delta) \circ \sigma^{-1} \models (x < r) +_\sigma {}^{f_-}/_{f_+}$.

[The case $\phi \wedge \psi$]: $M, m, i \models \phi \wedge \psi$ implies $M, m, i \models \phi$ and $M, m, i \models \psi$. By inductive hypothesis, $M, m, (i+\delta) \circ \sigma^{-1} \models \phi +_\sigma {}^{f_-}/_{f_+}$ and $M, m, (i+\delta) \circ \sigma^{-1} \models \psi +_\sigma {}^{f_-}/_{f_+}$, which imply $M, m, i+\delta \models (\phi +_\sigma {}^{f_-}/_{f_+}) \wedge (\psi +_\sigma {}^{f_-}/_{f_+})$. Hence $M, m, (i+\delta) \circ \sigma^{-1} \models (\phi \wedge \psi) +_\sigma {}^{f_-}/_{f_+}$.

[The case $\phi \vee \psi$]: $M, m, i \models \phi \vee \psi$ implies $M, m, i \models \phi$ or $M, m, i \models \psi$. By inductive hypothesis, $M, m, (i+\delta) \circ \sigma^{-1} \models \phi +_\sigma {}^{f_-}/_{f_+}$ or $M, m, (i+\delta) \circ \sigma^{-1} \models \psi +_\sigma {}^{f_-}/_{f_+}$, which imply $M, m, (i+\delta) \circ \sigma^{-1} \models (\phi +_\sigma {}^{f_-}/_{f_+}) \vee (\psi +_\sigma {}^{f_-}/_{f_+})$. Hence $M, m, (i+\delta) \circ \sigma^{-1} \models (\phi \vee \psi) +_\sigma {}^{f_-}/_{f_+}$.

[The case $[a]\phi$]: $M, m, i \models [a]\phi$ implies for any $m' \in M$ s.t. $m \xrightarrow{a} m'$, $M, m', i \models \phi$. By inductive hypothesis, $M, m', (i+\delta) \circ \sigma^{-1} \models \phi +_\sigma {}^{f_-}/_{f_+}$, which implies $M, m, (i+\delta) \circ \sigma^{-1} \models [a](\phi +_\sigma {}^{f_-}/_{f_+})$. Hence $M, m, (i+\delta) \circ \sigma^{-1} \models ([a]\phi) +_\sigma {}^{f_-}/_{f_+}$.

[The case $\langle a\rangle\phi$]: $M, m, i \models \langle a\rangle\phi$ implies exists $m' \in M$ s.t. $m \xrightarrow{a} m'$ and $M, m', i \models \phi$. By inductive hypothesis, $M, m', (i+\delta) \circ \sigma^{-1} \models \phi +_\sigma {}^{f_-}/_{f_+}$, which implies $M, m, (i+\delta) \circ \sigma^{-1} \models \langle a\rangle(\phi +_\sigma {}^{f_-}/_{f_+})$. Hence $M, m, (i+\delta) \circ \sigma^{-1} \models (\langle a\rangle\phi) +_\sigma {}^{f_-}/_{f_+}$.

[The case $\forall\!\!\!\forall\phi$]: $M, m, i \models \forall\!\!\!\forall\phi$ implies for any $d \in \mathbb{R}_+$ and $m' \in M$ s.t. $m' = d \oplus m$, $M, m', i \models \phi$. By inductive hypothesis, $M, m', (i+\delta) \circ \sigma^{-1} \models \phi +_\sigma {}^{f_-}/_{f_+}$, which implies $M, m, (i+\delta) \circ \sigma^{-1} \models \forall\!\!\!\forall(\phi +_\sigma {}^{f_-}/_{f_+})$. Hence $M, m, (i+\delta) \circ \sigma^{-1} \models (\forall\!\!\!\forall\phi) +_\sigma {}^{f_-}/_{f_+}$.

[The case $\exists\!\!\!\exists\phi$]: $M, m, i \models \exists\!\!\!\exists\phi$ implies exists $d \in \mathbb{R}_+$ and $m' \in M$ s.t. $m' = d \oplus m$, $M, m', i \models \phi$. By inductive hypothesis, $M, m', (i+\delta) \circ \sigma^{-1} \models \phi +_\sigma {}^{f_-}/_{f_+}$, which implies $M, m, (i+\delta) \circ \sigma^{-1} \models \exists\!\!\!\exists(\phi +_\sigma {}^{f_-}/_{f_+})$. Hence $M, m, (i+\delta) \circ \sigma^{-1} \models (\exists\!\!\!\exists\phi) +_\sigma {}^{f_-}/_{f_+}$.

[The case $\forall x.\phi$]: $M, m, i \models \forall x.\phi$ implies for any $t \in \mathbb{Q}_+$, $M, m, i[x \mapsto t] \models \phi$. By inductive hypothesis, $M, m', (i[x \mapsto t] + \delta[x \mapsto 0]) \circ \sigma^{-1} \models \phi +_\sigma {}^{f_-[x \mapsto 0]}/_{f_+[x \mapsto 0]}$, which implies $M, m', ((i+\delta) \circ \sigma^{-1})[x \mapsto t] \models \phi +_\sigma {}^{f_-[x \mapsto 0]}/_{f_+[x \mapsto 0]}$. Then $M, m, (i+\delta) \circ \sigma^{-1} \models \forall x.(\phi +_\sigma {}^{f_-[x \mapsto 0]}/_{f_+[x \mapsto 0]})$. Hence $M, m, (i+\delta) \circ \sigma^{-1} \models (\forall x.\phi) +_\sigma {}^{f_-}/_{f_+}$.

[The case $\exists x.\phi$]: $M, m, i \models \exists x.\phi$ implies exists $t \in \mathbb{Q}_+$ s.t. $M, m, i[x \mapsto t] \models \phi$. By inductive hypothesis, $M, m', (i[x \mapsto t] + \delta[x \mapsto 0]) \circ \sigma^{-1} \models \phi +_\sigma {}^{f_-[x \mapsto 0]}/_{f_+[x \mapsto 0]}$. So $M, m', ((i+\delta) \circ \sigma^{-1})[x \mapsto t] \models \phi +_\sigma {}^{f_-[x \mapsto 0]}/_{f_+[x \mapsto 0]}$. Then $M, m, (i+\delta) \circ \sigma^{-1} \models \exists x.(\phi +_\sigma {}^{f_-[x \mapsto 0]}/_{f_+[x \mapsto 0]})$. Hence $M, m, (i+\delta) \circ \sigma^{-1} \models (\exists x.\phi) +_\sigma {}^{f_-}/_{f_+}$. □

Proof. [Proof of Corollary 5.3] (⇒): from Lemma 5.2.
(⇐): Since $i + f - f \geq 0$, $M, m, i + f \models \phi + f$ implies $M, m, i + f - f \models \phi + f - f$ due to lemma 5.2. □

Proof. [Proof of Corollary 5.4] Let $f_-, f_+ : \mathcal{K} \to \mathbb{Q}$ be such that $f_- \leq \delta \leq f_+$ and $f_-(x) = f_+(x) = 0$ for any $x \in \mathcal{K}\phi)$. Then, $\phi + {}^{f_-}/_{f_+} = \phi + {}^{-f_+}/_{-f_-} = \phi$.
(⇒): derives from Lemma 5.2.
(⇐): Since $i + \delta - \delta \geq 0$, $M, m, i + \delta \models \phi$ implies $M, m, i + \delta - \delta \models \phi$ using Lemma 5.2. □

Proof. [Proof of Theorem 5.5] (⇒): Induction on ϕ.

[The Case $[a]\phi$]: $M, m, i \models [a]\phi$ implies for any $m' \in M$ s.t. $m \xrightarrow{a} m'$, $M, m', i \models \phi$. Since $m \sim n$, so exists $n' \in M$ s.t. $n \xrightarrow{a} n'$ and $m' \sim n'$. By inductive hypothesis, $M, m', i \models \phi$ implies $M, n', i \models \phi$. Hence $M, m, i \models [a]\phi$ implies $M, n, i \models [a]\phi$. Similarly $M, n, i \models [a]\phi$ implies $M, m, i \models [a]\phi$.

[The Case $\forall\!\!\!\forall\, \phi$]: $M, m, i \models [a]\phi$ implies for any $d \in \mathbb{R}_{\geq 0}$ and $m' \in M$ s.t. $m' = d \oplus m$, $M, m', i \models \phi$. Since $m \sim n$, so exists $n' \in M$ s.t. $n' = d \oplus n$ and $m' \sim n'$. By inductive hypothesis, $M, m', i \models \phi$ implies $M, n', i \models \phi$. Hence $M, m, i \models \forall\!\!\!\forall\phi$ implies $M, n, i \models \forall\!\!\!\forall\phi$. Similarly $M, n, i \models \forall\!\!\!\forall\phi$ implies $M, m, i \models \forall\!\!\!\forall\phi$.

[The Case $\forall x.\phi$]: $M, m, i \models \forall x.\phi$ iff for any $j = i[x \mapsto t]$ and $t \in \mathbb{R}_{\geq 0}$, $M, m, j \models \phi$. By inductive hypothesis, $M, m, j \models \phi$ iff $M, n, j \models \phi$. Hence $M, m, i \models \forall x.\phi$ iff $M, n, i \models \forall x.\phi$. □

Proof. [Proof of Lemma 6.3] First, the sets $A = \{r \in \mathbb{Q}^+ \mid x \geq r \in \Gamma\}$ and $B = \{r \in \mathbb{Q}^+ \mid x \leq r \in \Gamma\}$ are both non-empty: Axiom (A2) guarantees that for any $r \in \mathbb{Q}^+$, either $x \geq r \in \Gamma$ or $x \leq r \in \Gamma$. Suppose that there exists $r \in \mathbb{Q}^+$ such that $x \geq r \in \Gamma$. Then, $A \neq \emptyset$. Suppose that $B = \emptyset$, then (A2) implies that for any $r \in \mathbb{Q}^+$, $x \geq r \in \Gamma$. Using (R3) for $C[X] = X$, we derive that $\bot \in \Gamma$ - contradiction. Consequently, $B \neq \emptyset$. Similarly can be proven that $B \neq \emptyset$ implies $A \neq \emptyset$.

Since the two sets are non-empty, the sup and inf exist. Suppose that $\sup A = \infty$. Then applying (A3) we obtain that $\mathbb{Q}^+ \subseteq A$ and (R2) for $C[X] = X$ proves the inconsistency of Γ - contradiction. Similarly one can prove that $\inf B \in \mathbb{R}^+$.

Let $\sup A = u$ and $\inf B = v$. We prove $u = v$. If $u < v$, there exists $r \in \mathbb{Q}^+$ such that $u < r < v$. Since $u < r$, $x \geq r \notin \Gamma$ and (A2) guarantees that $x \leq r \in \Gamma$. But this contradicts the fact that $r \leq v$. If $v < u$, there exists $r_1, r_2 \in \mathbb{Q}^+$ such that $v < r_1 < r_2 < u$. Since $r_1 > v$, $x \leq r_i \in \Gamma$ for $i = 1, 2$ (applying (A3)), and similarly, $r_i < u$ implies $x \geq r_i \in \Gamma$ for $i = 1, 2$. Since $r_2 - r_1 > 0$, we apply (A3) and obtain $\vdash x \geq r_2 \to \neg(x \leq r_1)$. This shows that Γ is inconsistent - contradiction.

Consequently, $u = v$. □

Proof. [Proof of Lemma 6.5]

We say that a set $S \subseteq \Omega$ is *coherent* if there exists a set $T \subseteq \Im(\mathcal{K})$ and a bijection $\gamma : T \to S$ that satisfies the two conditions of Definition 6.4. S is *maximally-coherent* if $T = \Im(\mathcal{K})$. Observe that the bijection that defines a maximally-coherent set is a coherent function in the sense of Definition 6.4.

We prove, using a transfinite induction, that any coherent set has a maximally-coherent extension. And this proves the existence of γ, because the singleton $\{\Lambda\}$ is coherent.

I. Firstly, observe that the set $S \subseteq \Omega$ is coherent iff for arbitrary $\Lambda_1, \Lambda_2 \in \Omega$, with $i_s = \mathscr{I}(\Lambda_s)$, $s = 1, 2$,

$$\Lambda_1 + (i_2 - i_1) \subseteq \Lambda_2 \ \text{ and } \ \Lambda_2 + (i_1 - i_2) \subseteq \Lambda_1,$$

where for arbitrary $\Lambda' \in \Omega$ and $\delta : \mathcal{K} \to \mathbb{R}$,

$$\Lambda' + \delta = \{\phi + {}^{f_-}/_{f_+} \mid f_-, f_+ : \mathcal{K} \to \mathbb{Q}, f_- \leq \delta \leq f_+\}.$$

Moreover $\Lambda_1 + (i_2 - i_1) \subseteq \Lambda_2$ iff $\Lambda_2 + (i_1 - i_2) \subseteq \Lambda_1$. We prove it as follows:
Suppose $\Lambda_1 + (i_2 - i_1) \subseteq \Lambda_2$ but $\Lambda_2 + (i_1 - i_2) \not\subseteq \Lambda_1$, i.e., there exist $\psi \in \Lambda_2$, bijection σ and $f_-, f_+ : \mathfrak{I}(K) \to \mathbb{Q}$ s.t. $f_- \leq i_1 - i_2 \leq f_+$ and $\psi +_\sigma f_-/f_+ \notin \Lambda_1$. Since Λ_1 is

maximal, $\neg(\psi +_\sigma f_-/f_+) \in \Lambda_1$. So $\neg\psi +_\sigma f_+/f_- \in \Lambda_1$. Since $-f_+ \leq i_2 - i_1 \leq -f_-$ and σ is bijection, $(\neg\psi +_\sigma f_+/f_-) +_{\sigma^{-1}} -f_+/-f_- \in \Lambda_2$. Then $\neg\psi \in \Lambda_2$ - contradiction! Hence $\Lambda_2 + (i_1 - i_2) \subseteq \Lambda_1$.
Similarly for the other direction.

II. If $i = \mathscr{I}(\Lambda)$ and $i' \in \Im(\mathcal{K})$, then there exists $\Lambda' \in \Omega$ s.t. $\mathscr{I}(\Lambda') = i'$ and $\{\Lambda, \Lambda'\}$ is coherent, i.e., $\Lambda + (i' - i) \subseteq \Lambda'$. To prove this, we firstly prove the following two properties which will also be used in **IV**:

(a) For arbitrary $\phi \in \Lambda$, σ a bijection on $\mathcal{K}$, and $f_-, f_+ : \mathcal{K} \to \mathbb{Q}$ s.t. for any $x \in \mathcal{K}(\phi)$, either $f_-(x) = f_+(x) = 0$ or $f_-(x) \leq (i' - i)(x) \leq f_+(x)$. Then,

$$\vdash (\phi +_\sigma {}^{f_-}/_{f_+}) +_{\sigma^{-1}} {}^{-(f_-\circ\sigma^{-1})}/_{-(f_+\circ\sigma^{-1})} \to \phi.$$

(b) For any $x \trianglelefteq r \in \mathcal{L}$,

$$\{(x \trianglelefteq r) + {}^{f_-}/_{f_+} \mid f_-, f_+ : \mathcal{K} \to \mathbb{Q}, f_- < 0 < f_+\} \vdash x \trianglelefteq r.$$

[Proof of (a)]: Induction on ϕ.
[The Case $x \geq r$]:

$$(x \geq r) +_\sigma {}^{f_-}/_{f_+} = \begin{cases} \sigma(x) \geq (r + f_-(x)), \; r + f_-(x) \geq 0 \\ \sigma(x) \geq 0, \qquad\qquad \text{otherwise} \end{cases}$$

(i). If $r + f_-(x) \geq 0$:
$((x \geq r) +_\sigma {}^{f_-}/_{f_+}) +_{\sigma^{-1}} {}^{-(f_-\circ\sigma^{-1})}/_{-(f_+\circ\sigma^{-1})} = (\sigma(x) \geq (r + f_-(x))) +_{\sigma^{-1}} {}^{-(f_-\circ\sigma^{-1})}/_{-(f_+\circ\sigma^{-1})} = x \geq r$;
(ii). If $r + f_-(x) < 0$:
$((x \geq r) +_\sigma {}^{f_-}/_{f_+}) +_{\sigma^{-1}} {}^{-(f_-\circ\sigma^{-1})}/_{-(f_+\circ\sigma^{-1})} = (\sigma(x) \geq 0) +_{\sigma^{-1}} {}^{-(f_-\circ\sigma^{-1})}/_{-(f_+\circ\sigma^{-1})} = x \geq -f_-(x)$.
$r + f_-(x) < 0$ implies $-f_-(x) > r$ implies $x \geq r$.
So $\vdash ((x \geq r) +_\sigma {}^{f_-}/_{f_+}) +_{\sigma^{-1}} {}^{-(f_-\circ\sigma^{-1})}/_{-(f_+\circ\sigma^{-1})} \to (x \geq r)$.
[The Case $x \leq r$]: If $f_-(x) = f_+(x) = 0$, then obviously

$$((x \leq r) +_\sigma {}^{f_-}/_{f_+}) +_{\sigma^{-1}} {}^{-(f_-\circ\sigma^{-1})}/_{-(f_+\circ\sigma^{-1})} = x \leq r.$$

Otherwise,

$$(x \leq r) +_\sigma {}^{f_-}/_{f_+} = \begin{cases} \sigma(x) \leq (r + f_+(x)), \; r + f_+(x) \geq 0 \\ \sigma(x) \geq 0, \qquad\qquad \text{otherwise} \end{cases}$$

$x \leq r \in \Lambda$, $r \geq i_\Lambda(x)$, so $r + f_+(x) \geq i_\Lambda(x) + \delta(x) = i(x) \geq 0$. Then $(x \leq r) +_\sigma {}^{f_-}/_{f_+} = \sigma(x) \leq (r + f_+(x))$, which implies $((x \leq r) +_\sigma {}^{f_-}/_{f_+}) +_{\sigma^{-1}} {}^{-(f_-\circ\sigma^{-1})}/_{-(f_+\circ\sigma^{-1})} = x \leq r$.
So $\vdash ((x \leq r) +_\sigma {}^{f_-}/_{f_+}) +_{\sigma^{-1}} {}^{-(f_-\circ\sigma^{-1})}/_{-(f_+\circ\sigma^{-1})} \to (x \leq r)$.
[The Case $x > r$ and $x < r$]: similar as the above two.
[The Case $\phi \wedge \psi$ and $\phi \vee \psi$]: Obviously.
[The Case $[a]\phi$]: $(([a]\phi) +_\sigma {}^{f_-}/_{f_+}) +_{\sigma^{-1}} {}^{-(f_-\circ\sigma^{-1})}/_{-(f_+\circ\sigma^{-1})} = [a]((\phi +_\sigma {}^{f_-}/_{f_+}) +_{\sigma^{-1}} {}^{-(f_-\circ\sigma^{-1})}/_{-(f_+\circ\sigma^{-1})})$. By inductive hypothesis: $\vdash (\phi +_\sigma {}^{f_-}/_{f_+}) +_{\sigma^{-1}} {}^{-(f_-\circ\sigma^{-1})}/_{-(f_+\circ\sigma^{-1})} \to \phi$. By (R1) and (A5) $\vdash [a]((\phi +_\sigma {}^{f_-}/_{f_+}) +_{\sigma^{-1}} {}^{-(f_-\circ\sigma^{-1})}/_{-(f_+\circ\sigma^{-1})}) \to [a]\phi$.
Hence $\vdash (([a]\phi) +_\sigma {}^{f_-}/_{f_+}) +_{\sigma^{-1}} {}^{-(f_-\circ\sigma^{-1})}/_{-(f_+\circ\sigma^{-1})} \to [a]\phi$.
[The Case $\langle a\rangle\phi$]: $((\langle a\rangle\phi) +_\sigma {}^{f_-}/_{f_+}) +_{\sigma^{-1}} {}^{-(f_-\circ\sigma^{-1})}/_{-(f_+\circ\sigma^{-1})} = \langle a\rangle((\phi +_\sigma$

${}^{f_-}/_{f_+}) +_{\sigma^{-1}} {}^{-(f_-\circ\sigma^{-1})}/_{-(f_+\circ\sigma^{-1})})$. By inductive hypothesis: $\vdash (\phi +_\sigma {}^{f_-}/_{f_+}) +_{\sigma^{-1}} {}^{-(f_-\circ\sigma^{-1})}/_{-(f_+\circ\sigma^{-1})} \to \phi$. So $\vdash \neg\phi \to \neg((\phi +_\sigma {}^{f_-}/_{f_+}) +_{\sigma^{-1}} {}^{-(f_-\circ\sigma^{-1})}/_{-(f_+\circ\sigma^{-1})})$. By (R1) and (A5), $\vdash [a]\neg\phi \to [a]\neg((\phi +_\sigma {}^{f_-}/_{f_+}) +_{\sigma^{-1}} {}^{-(f_-\circ\sigma^{-1})}/_{-(f_+\circ\sigma^{-1})})$. Then $\vdash \langle a\rangle((\phi +_\sigma {}^{f_-}/_{f_+}) +_{\sigma^{-1}} {}^{-(f_-\circ\sigma^{-1})}/_{-(f_+\circ\sigma^{-1})}) \to \langle a\rangle\phi$.
Hence $\vdash (((\langle\phi\rangle) +_\sigma {}^{f_-}/_{f_+}) +_{\sigma^{-1}} {}^{-(f_-\circ\sigma^{-1})}/_{-(f_+\circ\sigma^{-1})} \to \langle a\rangle\phi$.
[The Case $\forall\!\!\!\forall\ \phi$]: $((\forall\!\!\!\forall\ \phi) +_\sigma {}^{f_-}/_{f_+}) +_{\sigma^{-1}} {}^{-(f_-\circ\sigma^{-1})}/_{-(f_+\circ\sigma^{-1})} = \forall\!\!\!\forall\ ((\phi +_\sigma {}^{f_-}/_{f_+}) +_{\sigma^{-1}} {}^{-(f_-\circ\sigma^{-1})}/_{-(f_+\circ\sigma^{-1})})$. By inductive hypothesis: $\vdash (\phi +_\sigma {}^{f_-}/_{f_+}) +_{\sigma^{-1}} {}^{-(f_-\circ\sigma^{-1})}/_{-(f_+\circ\sigma^{-1})} \to \phi$. By (R2) and (A7) $\vdash \forall\!\!\!\forall((\phi +_\sigma {}^{f_-}/_{f_+}) +_{\sigma^{-1}} {}^{-(f_-\circ\sigma^{-1})}/_{-(f_+\circ\sigma^{-1})}) \to \forall\!\!\!\forall\phi$.
Hence $\vdash ((\forall\!\!\!\forall\phi) +_\sigma {}^{f_-}/_{f_+}) +_{\sigma^{-1}} {}^{-(f_-\circ\sigma^{-1})}/_{-(f_+\circ\sigma^{-1})} \to \forall\!\!\!\forall\phi$.
[The Case $\exists\!\!\!\exists\ \phi$]: $((\exists\!\!\!\exists\ \phi) +_\sigma {}^{f_-}/_{f_+}) +_{\sigma^{-1}} {}^{-(f_-\circ\sigma^{-1})}/_{-(f_+\circ\sigma^{-1})} = \exists\!\!\!\exists\ ((\phi +_\sigma {}^{f_-}/_{f_+}) +_{\sigma^{-1}} {}^{-(f_-\circ\sigma^{-1})}/_{-(f_+\circ\sigma^{-1})})$. By inductive hypothesis: $\vdash (\phi +_\sigma {}^{f_-}/_{f_+}) +_{\sigma^{-1}} {}^{-(f_-\circ\sigma^{-1})}/_{-(f_+\circ\sigma^{-1})} \to \phi$. So $\vdash \neg\phi \to \neg((\phi +_\sigma {}^{f_-}/_{f_+}) +_{\sigma^{-1}} {}^{-(f_-\circ\sigma^{-1})}/_{-(f_+\circ\sigma^{-1})})$. By (R2) and (A7) $\vdash \forall\!\!\!\forall\neg\phi \to \forall\!\!\!\forall\neg((\phi +_\sigma {}^{f_-}/_{f_+}) +_{\sigma^{-1}} {}^{-(f_-\circ\sigma^{-1})}/_{-(f_+\circ\sigma^{-1})})$. Then $\vdash \exists\!\!\!\exists((\phi +_\sigma {}^{f_-}/_{f_+}) +_{\sigma^{-1}} {}^{-(f_-\circ\sigma^{-1})}/_{-(f_+\circ\sigma^{-1})}) \to \exists\!\!\!\exists\phi$.
Hence $\vdash ((\exists\!\!\!\exists\phi) +_\sigma {}^{f_-}/_{f_+}) +_{\sigma^{-1}} {}^{-(f_-\circ\sigma^{-1})}/_{-(f_+\circ\sigma^{-1})} \to \exists\!\!\!\exists\phi$.
[The Case $\forall x.\phi$]:
$((\forall x.\phi) +_\sigma {}^{f_-}/_{f_+}) +_{\sigma^{-1}} {}^{-(f_-\circ\sigma^{-1})}/_{-(f_+\circ\sigma^{-1})} =$
$\forall x.((\phi +_\sigma {}^{f_-[x\mapsto 0]}/_{f_+[x\mapsto 0]}) +_{\sigma^{-1}} {}^{-(f_-[x\mapsto 0]\circ\sigma^{-1})}/_{-(f_+[x\mapsto 0]\circ\sigma^{-1})})$. By inductive hypothesis: $\vdash (\phi +_\sigma {}^{f_-[x\mapsto 0]}/_{f_+[x\mapsto 0]}) +_{\sigma^{-1}} {}^{-(f_-[x\mapsto 0]\circ\sigma^{-1})}/_{-(f_+[x\mapsto 0]\circ\sigma^{-1})} \to \phi$.
By (R3), (R6) and (A13) $\vdash \forall x.((\phi +_\sigma {}^{f_-[x\mapsto 0]}/_{f_+[x\mapsto 0]}) +_{\sigma^{-1}} {}^{-(f_-[x\mapsto 0]\circ\sigma^{-1})}/_{-(f_+[x\mapsto 0]\circ\sigma^{-1})}) \to \forall x.\phi$.
Hence $\vdash ((\forall x.\phi) +_\sigma {}^{f_-}/_{f_+}) +_{\sigma^{-1}} {}^{-(f_-\circ\sigma^{-1})}/_{-(f_+\circ\sigma^{-1})} \to \forall x.\phi$.
[The Case $\exists x.\phi$]:
$((\exists x.\phi) +_\sigma {}^{f_-}/_{f_+}) +_{\sigma^{-1}} {}^{-(f_-\circ\sigma^{-1})}/_{-(f_+\circ\sigma^{-1})} =$
$\exists x.((\phi +_\sigma {}^{f_-[x\mapsto 0]}/_{f_+[x\mapsto 0]}) +_{\sigma^{-1}} {}^{-(f_-[x\mapsto 0]\circ\sigma^{-1})}/_{-(f_+[x\mapsto 0]\circ\sigma^{-1})})$.
By inductive hypothesis: $\vdash (\phi +_\sigma {}^{f_-[x\mapsto 0]}/_{f_+[x\mapsto 0]}) +_{\sigma^{-1}} {}^{-(f_-[x\mapsto 0]\circ\sigma^{-1})}/_{-(f_+[x\mapsto 0]\circ\sigma^{-1})} \to \phi$.
So $\vdash \neg\phi \to \neg((\phi +_\sigma {}^{f_-}/_{f_+}) +_{\sigma^{-1}} {}^{-(f_-\circ\sigma^{-1})}/_{-(f_+\circ\sigma^{-1})})$. By (R3), (R6) and (A13), we have $\vdash \forall x.(\neg\phi) \to \forall x.\neg((\phi +_\sigma {}^{f_-[x\mapsto 0]}/_{f_+[x\mapsto 0]}) +_{\sigma^{-1}} {}^{-(f_-[x\mapsto 0]\circ\sigma^{-1})}/_{-(f_+[x\mapsto 0]\circ\sigma^{-1})})$.
Then $\vdash \exists x.((\phi +_\sigma {}^{f_-}/_{f_+}) +_{\sigma^{-1}} {}^{-(f_-\circ\sigma^{-1})}/_{-(f_+\circ\sigma^{-1})}) \to \exists x.\phi$.
Hence $\vdash ((\exists x.\phi) +_\sigma {}^{f_-}/_{f_+}) +_{\sigma^{-1}} {}^{-(f_-\circ\sigma^{-1})}/_{-(f_+\circ\sigma^{-1})} \to \exists x.\phi$.

[Proof of (b)]: **[The Case $x \geq r$]:**

$$(x \geq r) + {}^{f_-}/_{f_+} = \begin{cases} x \geq (r + f_-(x)), & r + f_-(x) \geq 0 \\ x \geq 0, & \text{otherwise} \end{cases}$$

For any $r + f_-(x) \geq 0$, we have $x \geq (r + f_-(x))$. Since $f_-(x) < 0$, we have exists $s \in \mathbb{Q}$ s.t. $s = r + f_-(x) < r$ and $x \geq s$. By rule (R2), $x \geq r$.
So $\{(x \geq r) + {}^{f_-}/_{f_+} \mid \text{for any } f_-, f_+ : \mathcal{K} \to \mathbb{Q} \text{ s.t. for any}$
$x \in \mathcal{K}, f_-(x) < 0 < f_+(x)\} \vdash (x \geq r)$.
[The Case $x \leq r$]:

$$(x \leq r) + {}^{f_-}/_{f_+} = \begin{cases} x \leq (r + f_+(x)), \; r + f_+(x) \geq 0 \\ x \geq 0, \qquad\qquad \text{otherwise} \end{cases}$$

For any $r + f_+(x) \geq 0$, we have $x \leq (r + f_+(x))$. Since $f_+(x) > 0$, we have exists $s \in \mathbb{Q}$ s.t. $s = r + f_+(x) > r$, $x \leq s$. By rule (R2), $x \leq r$.
So $\{(x \leq r) + {}^{f_-}/_{f_+} \mid$ for any $f_-, f_+ : \mathcal{K} \to \mathbb{Q}$ s.t. for any $x \in \mathcal{K}, f_-(x) < 0 < f_+(x)\} \vdash (x \leq r)$.
[The Case $x > r$ and $x < r$]: similar as the above two.

Now we are ready to prove that there exists $\Lambda' \in \Omega$ s.t. $\mathscr{I}(\Lambda') = i'$ and $\Lambda + (i' - i) \subseteq \Lambda'$.
Suppose not, i.e., for any $\Lambda' \in \Omega$, either $A \not\subseteq \Lambda'$ or $A \subseteq \Lambda'$ but $\mathscr{I}(\Lambda') \neq i'$, where $A = \{\phi +_\sigma {}^{f_-}/_{f_+} \mid \phi \in \Lambda, f_-, f_+ : \mathcal{K} \to \mathbb{Q}$ s.t. $f_- \leq i' - i \leq f_+\}$.
* Suppose $A \not\subseteq \Lambda'$, i.e., $A \vdash \bot$ (otherwise by *Rasiowa-Sikorski Lemma*, there exists one maximal consistent set that contains it). Then $B \vdash \bot$, where $B = A +_{\sigma^{-1}} {}^{-f_- \circ \sigma^{-1}}/_{-f_+ \circ \sigma^{-1}} = \{\phi +_{\sigma^{-1}} {}^{-f_- \circ \sigma^{-1}}/_{-f_+ \circ \sigma^{-1}} \mid \phi \in A\}$. For any $\phi \in \Lambda$, $\phi +_\sigma {}^{f_-}/_{f_+} +_{\sigma^{-1}} {}^{-f_- \circ \sigma^{-1}}/_{-f_+ \circ \sigma^{-1}} \in B$. By (a), $\phi \in B$. So $\Lambda \subseteq B$. Since Λ is maximal, $\Lambda = B$. So $\Lambda \vdash \bot$ - contradiction!
* Suppose $A \subseteq \Lambda'$ but $\mathscr{I}(\Lambda') \neq i$.
First we prove $x \trianglelefteq r \notin A$ implies $\neg(x \trianglelefteq r) \in A$ for any $x \in \mathcal{K}$ and $r \in \mathbb{Q}^+$ as follows: $x \trianglelefteq r \notin A$ implies $(x \trianglelefteq r) +_{\sigma^{-1}} {}^{-f_- \circ \sigma^{-1}}/_{-f_+ \circ \sigma^{-1}} \notin \Lambda$. So $\neg((x \trianglelefteq r) +_{\sigma^{-1}} {}^{-f_- \circ \sigma^{-1}}/_{-f_+ \circ \sigma^{-1}}) \in \Lambda$. Then $(\neg(x \trianglelefteq r) +_{\sigma^{-1}} {}^{-f_+ \circ \sigma^{-1}}/_{-f_- \circ \sigma^{-1}}) +_\sigma {}^{f'_-}/_{f'_+} \in A$. So $\neg(x \trianglelefteq r) + {}^{f'_- - f_+}/_{f'_+ - f_-} \in A$, where $f_-, f'_- \leq i' - i \leq f_+, f'_+$ which implies $f'_- - f_+ \leq 0 \leq f'_+ - f_-$. If $i' - i \in \mathbb{Q}^+$, then there exists f'_-, f_+ s.t. $f'_- - f_+ = 0$ (and f_-, f'_+ s.t. $f'_+ - f_- = 0$). Hence $\neg(x \trianglelefteq r) \in A$. If not, $f'_- - f_+ < 0 < f'_+ - f_-$. By (b), $\neg(x \trianglelefteq r) \in A$.
Then for all $r \geq i(x) \geq 0$, suppose $x > r \in \Lambda'$. So $x > r \in A_i$ by the above result. Then $x > r - f_-(x) \in \Lambda$, which implies $r - f_-(x) \leq i_\Lambda(x)$. Hence $r \leq i_\Lambda(x) + f_-(x) \leq i(x)$ - contradiction! So $x \leq r \in \Lambda'$.
For all $0 < r \leq i(x)$, suppose $x < r \in \Lambda'$. Then $x < r \in A_i$ by the above result. Then $x < r - f_+(x) \in \Lambda$, which implies $r - f_+(x) \geq i_\Lambda(x)$. Then $r \geq i_\Lambda(x) + f_+(x) \geq i(x)$ - contradiction! So $x \geq r \in \Lambda'$.
If $r = 0$, apparently $x \geq 0 \in \Lambda'$.
So $\mathscr{I}(\Lambda') = i$ - contradiction!

IV. If $S = \{\Lambda_s \mid s \in \mathbb{N}\}$ is a countable coherent set (supp. $\Lambda = \Lambda_0$) and $j \in \Im(\mathcal{K})$. There exists $\Lambda_j \in \Omega$ s.t. $S \cup \{\Lambda_j\}$ is coherent. As before, we need to prove that

$$X = \bigcup_{s \in \mathbb{N}} (\Lambda_s + (j - i_s))$$

is consistent. Let $X_t = \bigcup_{s \leq t} (\Lambda_s + (j - i_s))$ for $t \in \mathbb{N}$. We have previously proved that X_t is consistent for any $t \in \mathbb{N}$. Moreover, observe that $X = \bigcup_{t \in \mathbb{N}} X_t$.

Suppose that X is not consistent. If the contradiction can be proven without involving the infinitary rules, then there must exist one X_t that contains all the formulas in the proof, which contradicts the consistency of X_t. Consequently, if X is inconsistent it must be because, when t goes to infinite, the sets $X_t \setminus X_{t-1}$ collect the left-hand side of some of our infinitary rules.

For arbitrary contexts C, formulas $\phi \in \mathcal{L}$, $r \in \mathbb{Q}_{\geq 0}$ and $x \in \mathcal{K}$ we denote by
$\mathcal{L}_1(C,x) = \{C[x \leq s], C[x \geq s] \mid s \in \mathbb{Q}_{\geq 0}\}$
$\mathcal{L}_2(C,x,\phi) = \{C[\phi + {}^{[x \mapsto s]}/_{[x \mapsto s']}] \mid s, s' \in \mathbb{Q}, s \leq s'\}$,
$\mathcal{L}_3(C,x,\phi,r) = \{C[\forall(r \leq x \leq s \to \phi)] \mid r, s \in \mathbb{Q}_{\geq 0}, r \leq s\}$.
We already proved that for each $t \in \mathbb{N}$, X_t is $\mathcal{L}_1(C,x)$-maximally consistent in II. Similarly, for $\mathcal{L}_2(C,x,\phi)$ either $X_t \cap \mathcal{L}_2(C,x,\phi) = \emptyset$ or X_t is $\mathcal{L}_2(C,x,\phi)$-maximally consistent. To prove the same result for $\mathcal{L}_3(C,x,\phi)$, we rely on the following fact: If Λ_1, Λ_2 are coherent and $t_1 = \sup\{s \in \mathbb{Q} \mid \forall(x \leq s \to \phi) \in \Lambda_1\}$, then there exists no $t_2 > t_1 + (i_2(x) - i_1(x))$ such that for any $s' \leq t_2$ and any $f_- \leq i_2 - i_1 \leq f_+$,

$$\neg\forall(x \leq s' \to (\phi + f_-/f_+)) \in \Lambda_2.$$

This demonstrates that if in X we can use some infinitary rule, it must be used within some X_t for some $t \in \mathbb{N}$. Consequently, the consistency of X_t proves the consistency of X.

V. If $S \subseteq \Omega$ is an uncountable coherent set and $j \in \Im(\mathcal{K})$, then there exists $\Lambda_j \in \Omega$ s.t. $S \cup \{j\}$ is coherent. Supp. $S = \{\Lambda_s \mid s \in \mathbb{R}\}$, $\Lambda = \Lambda_0$ and $i_s = \mathscr{I}(\Lambda_s)$. As in the other cases it is sufficient to prove that the set

$$Y = \bigcup_{s \in \mathbb{R}} (\Lambda_s + (j - i_s))$$

is consistent. Since $Y \subseteq \mathcal{L}$ and $\mathcal{L}$ is countable, Y is countable and consequently we can assume that

$$Y = \bigcup_{s \in \mathbb{N}} (\Lambda_s + (j - i_s)).$$

Further the result is a consequence of IV. □

Proof. [Proof of Theorem 6.6] We need to prove that $\oplus$ is well-defined and satisfies the required conditions 1 and 2 in Definition 3.1. That is to prove the following three conditions:

1. $\oplus$ is a well-defined partial function: $\mathbb{R}_{\geq 0} \times \Gamma \to \Gamma$, i.e. if $\gamma_1 = d \oplus \gamma$ and $\gamma_2 = d \oplus \gamma$, then $\gamma_1 = \gamma_2$.

For any $\forall i \in \mathfrak{I}(\mathcal{K})$, $\phi \in \gamma_1(i)$ implies $x = \lceil d \rceil \wedge \phi \in \gamma_1(i[x \mapsto \lceil d \rceil])$, where $x \in \mathcal{K} \backslash \mathcal{K}(\phi)$ is a new clock different from those in ϕ. It implies that $\forall(x = \lceil d \rceil \to \phi) \in \gamma(i[x \mapsto \lceil d \rceil] - d)$ because $\gamma_1 = d \oplus \gamma$ and axiom (A10). So $x = \lceil d \rceil \to \phi \in \gamma_2(i[x \mapsto \lceil d \rceil])$ because $\gamma_2 = d \oplus \gamma$. Then $\phi \in \gamma_2(i[x \mapsto \lceil d \rceil])$, which implies $\phi \in \gamma_2(i)$ since $x \in \mathcal{K} \backslash \mathcal{K}(\phi)$.

Hence for any $i \in \mathfrak{I}(\mathcal{K})$, $\gamma_1(i) \subseteq \gamma_2(i)$. Similarly $\gamma_2 \subseteq \gamma_2(i)$. Then for any $i \in \mathfrak{I}(\mathcal{K})$, $\gamma_1(i) = \gamma_2(i)$. Hence $\gamma_1 = \gamma_2$.

2. For any $\gamma \in \Gamma$, $0 \oplus \gamma = \gamma$, i.e. for any $i \in \mathfrak{I}(\mathcal{K})$, $\forall\phi \in \gamma$ implies $\phi \in \gamma$. It is obviously true by (A9).

3. $d \oplus (d' \oplus m) \stackrel{\sim}{=} (d + d') \oplus m$, i.e. $\exists \gamma_1, \gamma_2$ s.t. $\gamma_1 = d' \oplus \gamma$, $\gamma_2 = d \oplus \gamma_1$ iff $\exists \gamma_3$ s.t. $\gamma_3 = (d + d') \oplus \gamma$.

($\Rightarrow$): suppose $\exists\gamma_1, \gamma_2$ s.t. $\gamma_1 = d \oplus \gamma$, $\gamma_2 = d' \oplus \gamma_1$. Then for any $i \in \mathfrak{I}(\mathcal{K})$, for any $\forall\!\!\!\forall\phi \in \gamma(i)$ implies $\forall\!\!\!\forall\,\forall\!\!\!\forall\phi \in \gamma(i)$ by (A8). So $\forall\!\!\!\forall\phi \in \gamma_1(i+d)$ because $\gamma_1 = d \oplus \gamma$, which further implies $\phi \in \gamma_2(i+d+d')$ because $\gamma_2 = d' \oplus \gamma_1$. Hence $\exists\gamma_3 = \gamma_2$ s.t. $\gamma_3 = (d+d') \oplus \gamma$.

($\Leftarrow$): suppose $\exists\gamma_3$ s.t. $\gamma_3 = (d+d') \oplus \gamma$.

- First, we prove: if $\gamma_3 = (d+d') \oplus \gamma$ then $\exists\gamma_1$ s.t. $\gamma_1 = d' \oplus \gamma$.

For any $i \in \mathfrak{I}(\mathcal{K})$, let $\lambda_i = \{\phi \mid \forall\!\!\!\forall\phi \in \gamma(i)\}$, $\Delta_i = \bigcup_{x\in\mathcal{K}}\{x \trianglelefteq r \mid i(x) \trianglelefteq r, r \in \mathbb{Q}^+\}$, $\Theta_{i+d'} = \{\phi \mid \lambda_i \cup \Delta_{i+d'} \vdash \phi\}$.

We construct γ_1 according to the following steps:

(1) We prove $\Theta_{i+d'}$ is consistent.

We only need to show that $\lambda_i \cup \Delta_{i+d'}$ is consistent.

Suppose not, i.e. $\lambda_i \cup \Delta_{i+d'} \vdash \bot$.

Since both λ_i and $\Delta_{i+d'}$ are consistent, exists $\rho \in \mathcal{L}$ s.t. $\lambda_i \vdash \neg\rho$ and $\Delta_{i+d'} \vdash \rho$.

$\lambda_i \vdash \neg\rho$ implies $\forall\!\!\!\forall\lambda_i \vdash \forall\!\!\!\forall\neg\rho$ by rule (R2) or (R4), which further implies $\forall\!\!\!\forall\neg\rho \in \gamma(i)$.

$\Delta_{i+d'} \vdash \rho$ implies $\rho \in \gamma'(i+d')$ for any $\gamma' \in \Gamma$.

Let $\delta = (i+d')[\mathcal{K}(\rho) \mapsto \lceil d' \rceil] - (i+d')$, $f_-, f_+ : \mathcal{K} \to \mathbb{Q}$ defined as:

$$f_-(x) = \begin{cases} \lceil d' \rceil - r_x, & x \in \mathcal{K}(\rho) \\ 0, & \text{otherwise} \end{cases}$$

$$f_=(x) = \begin{cases} \lceil d' \rceil - s_x, & x \in \mathcal{K}(\rho) \\ 0, & \text{otherwise} \end{cases}$$

where $r_x = \max\{x \geq r_x$ is in ρ with $\neg$ on atoms$\}$, $s_x = \min\{x \leq s_x$ is in ρ with $\neg$ on atoms$\}$.

Then $\rho + {}^{f_-}/_{f_+} \in \gamma'((i+d')[\mathcal{K}(\rho) \mapsto \lceil d' \rceil])$, which implies $\rho + {}^{f_-}/_{f_+} \in \gamma'(i'[\mathcal{K}(\rho) \mapsto \lceil d' \rceil])$ for any $i' \in \mathfrak{I}(\mathcal{K})$. Then $\bigwedge_{x\in\mathcal{K}(\rho)}(x = \lceil d' \rceil) \to \rho + {}^{f_-}/_{f_+} \in \gamma'(i')$ for any γ' and i'. By rule (R2) $\forall\!\!\!\forall(\bigwedge_{x\in\mathcal{K}(\rho)}(x = \lceil d' \rceil) \to \rho + {}^{f_-}/_{f_+}) \in \gamma'(i')$.

$x \leq \lceil d' \rceil \in \gamma(i[\mathcal{K}(\rho) \mapsto \lceil d' \rceil - d'])$ for any $x \in \mathcal{K}(\rho)$. Then $\bigvee_{x\in\mathcal{K}(\rho)}(x \leq \lceil d' \rceil) \in \gamma(i[\mathcal{K}(\rho) \mapsto \lceil d' \rceil - d'])$. And similarly $\bigvee_{x\in\mathcal{K}(\rho)}(x \geq \lceil d' \rceil) \in \gamma(i[\mathcal{K}(\rho) \mapsto \lceil d' \rceil - d'] + (d+d'))$, which implies $\exists\!\!\!\exists(\bigvee_{x\in\mathcal{K}(\rho)}(x \geq \lceil d' \rceil)) \in \gamma(i[\mathcal{K}(\rho) \mapsto \lceil d' \rceil - d'])$. So $\bigvee_{x\in\mathcal{K}(\rho)}(x \leq \lceil d' \rceil) \wedge \exists\!\!\!\exists(\bigvee_{x\in\mathcal{K}(\rho)}(x \geq \lceil d' \rceil)) \in \gamma(i[\mathcal{K}(\rho) \mapsto \lceil d' \rceil - d'])$. By axiom (A11) $\exists\!\!\!\exists(\bigvee_{x\in\mathcal{K}(\rho)}(x = \lceil d' \rceil)) \in \gamma(i[\mathcal{K}(\rho) \mapsto \lceil d' \rceil - d'])$.

Together with $\forall\!\!\!\forall(\bigwedge_{x\in\mathcal{K}(\rho)}(x = \lceil d' \rceil) \to \rho + {}^{f_-}/_{f_+}) \in \gamma(i[\mathcal{K}(\rho) \mapsto \lceil d' \rceil])$, we get $\exists\!\!\!\exists(\rho + {}^{f_-}/_{f_+}) \in \gamma(i[\mathcal{K}(\rho) \mapsto \lceil d' \rceil])$ by axiom (A10).

So $\exists\!\!\!\exists\rho \in \gamma(i)$ - contradict that $\forall\!\!\!\forall\neg\rho \in \gamma(i)$. Hence $\lambda_i \cup \Delta_{i+d'}$ is consistent, which implies $\Theta_{i+d'}$ is consistent.

(2) Extend to maximal consistent set Λ_i for each $i \in I$.

(i) If there exists $x \in \mathcal{K}$ s.t. $i(x) \geq d'$ and $i(x) \in \mathbb{Q}^+$, Θ_i is maximal, i.e. for any $\phi \in \mathcal{L}$, $\phi \notin \Theta_i$ implies $\neg\phi \in \Theta_i$.

By definition of Δ_i, $x = i(x) \in \Delta_i \subseteq \Theta_i$. Together with $\phi \notin \Theta_i$, $x = i(x) \to \phi \notin \Theta_i$. By definition of $\lambda_{i-d'}$, $\forall\!\!\!\forall\,(x = i(x) \to \phi) \in \gamma(i-d')$ implies $x = i(x) \to \phi \in \lambda_{i-d'} \subseteq \Theta_i$. So $\forall\!\!\!\forall\,(x = i(x) \to \phi) \notin \gamma(i-d')$. Since $\gamma(i-d')$ is maximal and $\neg(\exists\!\!\!\exists\,(x = i(x) \wedge \neg\phi)) = \forall\!\!\!\forall\,(x = i(x) \to \phi)$, we have $\forall\!\!\!\forall\,(x = i(x) \to \phi) \notin \gamma(i-d')$ implies $\exists\!\!\!\exists\,(x = i(x) \wedge \neg\phi) \in \gamma(i-d')$. By (A10) $\forall\!\!\!\forall\,(x = i(x) \to \neg\phi) \in \gamma(i-d')$. So $x = i(x) \to \neg\phi \in \lambda_{i-d'} \subseteq \Theta_i$ by the definition of $\lambda_{i-d'}$. Then we have $\neg\phi \in \Theta_i$.

So $\Lambda_i = \Theta_i$, if there exists $x \in \mathcal{K}$ s.t. $i(x) \geq d'$ and $i(x) \in \mathbb{Q}^+$.

(ii) If for all $x \in \mathcal{K}$, $i(x) \geq d'$ and $i(x) \in \mathbb{R}_{\geq 0}$ but $\notin \mathbb{Q}^+$, define the following function, which given an interpretation$i$ and finite set of clocks $\mathcal{K}_f$, maps to another interpretation:

$$i^{\mathcal{K}_f} = \begin{cases} i(x) & x \in \mathcal{K}_f \\ \lceil d' \rceil & \text{otherwise} \end{cases}$$

Then we define $\Lambda_i = \bigcup_{\mathcal{K}_f \subset \mathcal{K}} \{\phi \mid \mathcal{K}(\phi) = \mathcal{K}_f,\ \phi \in \Pi_{i^{\mathcal{K}_f}}\}$.

Now we prove that Λ_i above is maximal consistent.

Obviously for any $x \in \mathcal{K}_f$ and any finite $\mathcal{K}_f \subset \mathcal{K}$, $i^{\mathcal{K}_f}(x) - d' \geq 0$. And according to the first case, $\Lambda_{i^{\mathcal{K}_f}}$ is maximal consistent. So Λ_i is maximal.

Suppose Λ_i is not consistent, i.e. $\Lambda_i \vdash \bot$. Suppose $\Psi \subseteq \Lambda_i$ s.t. $\Psi \vdash \bot$, and $\mathcal{K}(\Psi)$ is the set of clocks of Ψ. Since even in the infinitary rules, the number of the clocks is finite, $\mathcal{K}(\Psi)$ is finite. For any $\psi \in \Psi$, $\psi \in \Lambda_{i^{\mathcal{K}(\psi)}}$ by the definition of Λ_i.

By the first case, $\Lambda_{i^{\mathcal{K}(\psi)}} = \Theta_{i^{\mathcal{V}(\psi)}}$, so $\lambda_{i^{\mathcal{K}(\psi)}-d'} \cup \Delta_{i^{\mathcal{K}(\psi)}} \vdash \psi$. Suppose $\Psi' \subseteq \lambda_{i^{\mathcal{K}(\psi)}-d'} \cup \Delta_{i^{\mathcal{K}(\psi)}}$ s.t. $\Psi' \vdash \psi$. $\Psi'\{{}^{y'}\!/_{y}\} \vdash \psi$ for any $y \in \mathcal{K}(\Psi) \backslash \mathcal{K}(\psi), y' \notin \mathcal{K}(\Psi)$. Since $\Psi'\{{}^{y'}\!/_{y}\}$ has no clocks in $\mathcal{K}(\Psi) \backslash \mathcal{K}(\psi)$, and $i^{\mathcal{K}(\Psi)}$ and $i^{\mathcal{K}(\psi)}$ only differ for the clocks in $\mathcal{K}(\Psi) \backslash \mathcal{K}(\psi)$, it is easy to have $\Psi'\{{}^{y'}\!/_{y}\} \in \lambda_{i^{\mathcal{K}(\Psi)}-d'} \cup \Delta_{i^{\mathcal{K}(\Psi)}}$. Then $\psi \in \Theta_{i^{\mathcal{K}(\Psi)}}$. Hence $\Psi \subseteq \Theta_{i^{\mathcal{K}(\Psi)}}$, which implies $\Theta_{i^{\mathcal{K}(\Psi)}} \vdash \bot$ - contradiction.

(iii) If for all $x \in \mathcal{K}$, $i(x) < d'$.

$\Lambda_i = \{\phi \mid \phi + \lceil d' \rceil \in \Lambda_{i+\lceil d' \rceil}\}$.

Obviously Λ_i is a consistent set since $\Lambda_{i+\lceil d' \rceil}$ is consistent. Now we prove Λ_i is also maximal. Suppose $\phi \notin \Lambda_i$, so $\phi + \lceil d' \rceil \notin \Lambda_{i+\lceil d' \rceil}$, which implies $\neg(\phi + \lceil d' \rceil) \in \Lambda_{i+\lceil d' \rceil}$ since $\Lambda_{i+\lceil d' \rceil}$ is maximal. Then we have $\neg\phi \in \Lambda_i$. So Λ_i is maximal.

(3) Let $\gamma_1(i) = \Lambda_i$.

We prove that γ_1 is a coherent function, i.e., γ_1 satisfies the two conditions in Definition 6.4.

The first condition holds obviously.

Before we prove γ_1 satisfies the second condition, we first prove the following:

(a) For any bijection σ on $\mathcal{K}$, any $\delta : \mathcal{K} \to \mathbb{R}$ s.t. $\forall x \in \mathcal{K}$, $i(x) \geq d'$, $i(x)+\delta(x) \geq d'$, and any $f_-, f_+ : \mathcal{K} \to \mathbb{Q}$ s.t. $f_- \leq \delta \leq f_+$, we have: $\phi \in \Theta_i \Rightarrow \phi +_\sigma {}^{f_-}/_{f_+} \in \Theta_{(i+\delta)\circ\sigma^{-1}}$.

$\phi \in \Theta_i$ implies $\lambda_{i-d'} \cup \Delta_i \vdash \phi$. Because γ is interpretation coherent, so $\lambda_{i-d'} +_\sigma {}^{f_-}/_{f_+} \subseteq \lambda_{(i+\delta)\circ\sigma^{-1}-d'}$. $\Delta_i +_\sigma {}^{f_-}/_{f_+} \subseteq \Delta_{(i+\delta)\circ\sigma^{-1}}$ by the definition of Δ_i. So we have $\lambda_{(i+\delta)\circ\sigma^{-1}-d'} \cup \Delta_{(i+\delta)\circ\sigma^{-1}} \vdash \phi +_\sigma {}^{f_-}/_{f_+}$. Hence $\phi \in \Theta_i \Rightarrow \phi +_\sigma {}^{f_-}/_{f_+} \in \Theta_{(i+\delta)\circ\sigma^{-1}}$.

(b) For any bijection σ on $\mathcal{K}$, any $\delta : \mathcal{K} \to \mathbb{R}$ s.t. $\forall x \in \mathcal{K}$, $i(x) \geq d'$, $i(x)+\delta(x) \geq d'$, and any $f_-, f_+ : \mathcal{K} \to \mathbb{Q}$ s.t. $f_- \leq \delta \leq f_+$, we have: $\phi \in \Lambda_i \Rightarrow \phi +_\sigma {}^{f_-}/_{f_+} \in \Lambda_{(i+\delta)\circ\sigma^{-1}}$.

If $\Lambda_i = \Theta_i$, it's the (a) case.

If not, $\phi \in \Lambda_{i^{\mathcal{K}(\phi)}} = \Theta_{i^{\mathcal{V}(\phi)}}$. $i(x) + \delta(x) \geq 0$ implies $i^{\mathcal{K}(\phi)}(x) + \delta^{\mathcal{K}(\phi)}(x) \geq 0$, so by the first case, $\phi +_\sigma {}^{f_-}/_{f_+} \in \Theta_{(i^{\mathcal{K}(\phi)}+\delta^{\mathcal{K}(\phi)})\circ\sigma^{-1}}$.

$(i^{\mathcal{K}(\phi)} + \delta^{\mathcal{K}(\phi)}) \circ \sigma^{-1}(x) = i^{\mathcal{K}(\phi)}(\sigma^{-1}(x)) + \delta^{\mathcal{K}(\phi)}(\sigma^{-1}(x)) = (i \circ \sigma^{-1})^{\mathcal{K}(\phi+_\sigma {}^{f_-}/_{f_+})}(x) + (\delta \circ \sigma^{-1})^{\mathcal{K}(\phi+_\sigma {}^{f_-}/_{f_+})}(x)$ (because $\sigma^{-1}(x) \in \mathcal{K}(\phi)$ iff $x \in \mathcal{K}(\phi +_\sigma {}^{f_-}/_{f_+})$) $= ((i+\delta) \circ \sigma^{-1})^{\mathcal{K}(\phi+_\sigma {}^{f_-}/_{f_+})}(x)$.

So we have $\phi +_\sigma {}^{f_-}/_{f_+} \in \Theta_{((i+\delta)\circ\sigma^{-1})^{\mathcal{K}(\phi+_\sigma {}^{f_-}/_{f_+})}}$.

Hence $\phi \in \Lambda_i \Rightarrow \phi +_\sigma {}^{f_-}/_{f_+} \in \Lambda_{(i+\delta)\circ\sigma^{-1}}$.

Now we are ready to prove the second condition: for any bijection σ on $\mathcal{K}$, any $\delta : \mathcal{K} \to \mathbb{R}$ s.t. $\forall x \in \mathcal{K}$, $i(x) + \delta(x) \geq 0$, and any $f_-, f_+ : \mathcal{K} \to \mathbb{Q}$ s.t. $f_- \leq \delta \leq f_+$, we have: $\phi \in \gamma(i) \Rightarrow \phi +_\sigma {}^{f_-}/_{f_+} \in \gamma((i+\delta)\circ\sigma^{-1})$.

- If $i \geq d'$ and $i + \delta \geq d'$, it's the (b) case above.
- If $i \ngeq d'$ and $i + \delta \geq d'$:

  $\phi + \lceil d' \rceil \in \gamma_1(i + \lceil d' \rceil)$ by the definition of γ_1. Let $i' = i + \lceil d' \rceil$, $\delta' = \delta - \lceil d' \rceil$, $f_-^{'} = f_- - \lceil d' \rceil$ and $f_+^{'} = f_+ - \lceil d' \rceil$. Then we have $i' \geq d', i' + \delta' = i + \delta \geq d'$. So we have $(\phi + \lceil d' \rceil) +_\sigma {}^{f'_-}/_{f'_+} \in \gamma_1((i' + \delta') \circ \sigma^{-1})$, which implies $\phi +_\sigma {}^{f_-}/_{f_+} \in \gamma((i+\delta)\circ\sigma^{-1})$.
- If $i \geq d'$ and $i + \delta \ngeq d'$:

  Then $\phi +_\sigma {}^{f_- + \lceil d' \rceil}/_{f_+ + \lceil d' \rceil} \in \gamma_1((i + \delta + \lceil d' \rceil) \circ \sigma^{-1})$.

  Induction on ϕ:

  [The case $x \geq r$]:

  $(x \geq r) +_\sigma {}^{f_-}/_{f_+} + \lceil d' \rceil =$
  $$\begin{cases} \sigma(x) \geq r + f_-(x) + \lceil d' \rceil, & \text{if } r + f_-(x) \geq 0 \\ \sigma(x) \geq \lceil d' \rceil, & \text{otherwise} \end{cases}$$

So $(x \geq r) +_\sigma {}^{f_-}\!/_{f_+} + \lceil d' \rceil \in \gamma_1((i + \delta + d') \circ \sigma^{-1})$.

[The case $x > r$**]:** similar as above.

[The case $x \leq r$**]:**

$$(x \leq r) +_\sigma {}^{f_-}\!/_{f_+} + \lceil d' \rceil = \begin{cases} \sigma(x) \leq r + f_+(x) + \lceil d' \rceil, & \text{if } r + f_+(x) \geq 0 \\ \sigma(x) \leq \lceil d' \rceil, & \text{otherwise} \end{cases}$$

So $(x \leq r) +_\sigma {}^{f_-}\!/_{f_+} + \lceil d' \rceil \in \gamma_1((i + \delta + d') \circ \sigma^{-1})$.

[The case $x < r$**]:** similar as above.

Other cases hold obviously.

So we have $\phi +_\sigma {}^{f_-}\!/_{f_+} \in \gamma_1((i + \delta) \circ \sigma^{-1})$.

- If $i \ngeq d'$ and $i + \delta \ngeq d'$:
  $\phi \in \gamma_1(i)$ implies $\phi + \lceil d' \rceil \in \gamma_1(i + \lceil d' \rceil)$. Then $(\phi + \lceil d' \rceil) +_\sigma {}^{f_-}\!/_{f_+} \in \gamma_1((i + \lceil d' \rceil + \delta) \circ \sigma^{-1})$. So $\phi +_\sigma {}^{f_- + \lceil d' \rceil}\!/_{f_+ + \lceil d' \rceil} \in \gamma_1((i + \lceil d' \rceil + \delta) \circ \sigma^{-1})$. Then we can do the same deduction as the above case.

So we proved that there exists γ_1 constructed as above, s.t. $\gamma_1 = d' \oplus \gamma$.

- Now we prove that: $\exists \gamma_2$ s.t. $\gamma_2 = d \oplus \gamma_1$.

In the following we prove that $\gamma_3 = d \oplus \gamma_1$, i.e. for any $i \in \mathfrak{I}(\mathcal{K})$, $\forall\!\!\forall \phi \in \gamma_1(i)$ implies $\phi \in \gamma_3(i + d)$.

Secondly we need to prove that exists γ_2 s.t. $\gamma_2 = d \oplus \gamma_1$. We prove that $\gamma_3 = d \oplus \gamma_1$, i.e. for any $i \in \mathfrak{I}(\mathcal{K})$, for any $\forall\!\!\forall \phi \in \gamma_1(i)$ implies $\phi \in \gamma_3(i + d)$.

For any $i \in \mathfrak{I}(\mathcal{K})$, $\forall\!\!\forall\, \phi \in \gamma_1(i)$ implies $x \leq r \wedge \forall\!\!\forall\, \phi \in \gamma_1(i)$ for $r \in \mathbb{Q}^+$ s.t. $i(x) \leq r \leq i(x) + d$. Then $x \leq r + \lceil d' \rceil \wedge \forall\!\!\forall(\phi + \lceil d' \rceil) \in \gamma_1(i + \lceil d' \rceil)$. Since $\gamma_1 = d' \oplus \gamma$, we get $\exists\!\!\exists(x \leq r + \lceil d' \rceil \wedge \forall\!\!\forall(\phi + \lceil d' \rceil)) \in \gamma(i + \lceil d' \rceil - d')$. By (A9), $\forall\!\!\forall(x \geq r + \lceil d' \rceil \to \forall\!\!\forall(\phi + \lceil d' \rceil)) \in \gamma(i + \lceil d' \rceil - d')$. So $x \geq r + \lceil d' \rceil \to \forall\!\!\forall(\phi + \lceil d' \rceil) \in \gamma_3(i + \lceil d' \rceil + d)$ since $\gamma_3 = (d + d') \oplus \gamma$. And $r \leq i(x) + d$, which implies $x \geq r + \lceil d' \rceil \in \gamma_3(i + \lceil d' \rceil + d)$. Then $\forall\!\!\forall(\phi + \lceil d' \rceil) \in \gamma_3(i + \lceil d' \rceil + d)$. Hence $\forall\!\!\forall \phi \in \gamma_3(i + d)$.

So there exists $\gamma_2 = \gamma_3$ s.t. $\gamma_2 = d \oplus \gamma_1$.

Hence $\mathcal{W} = (\Gamma, \theta, \oplus)$ defined above is a timed labeled transition system. □

A Metrized Duality Theorem for Markov Processes

Dexter Kozen
Cornell University, USA

Radu Mardare
Aalborg University, Denmark

Prakash Panangaden
McGill University, Canada

Abstract

We extend our previous duality theorem for Markov processes by equipping the processes with a pseudometric and the algebras with a notion of metric diameter. We are able to show that the isomorphisms of our previous duality theorem become isometries in this quantitative setting. This opens the way to developing theories of approximate reasoning for probabilistic systems.

Keywords: Markov processes, Aumann algebras, Stone duality, metrics, isometry, probabilistic reasoning, quantitative reasoning.

1 Introduction

Stone-type dualities are recognized as being ubiquitous, especially in computer science. For example Plotkin [22] and Smyth [23] (see also [21]) emphasized that the duality between state-transformer semantics and predicate-transformer semantics is an instance of Stone-type duality. A similar duality was observed for probabilistic transition systems [14]. Recently several authors, see for example [3], have emphasized the duality between logics and transition systems from a coalgebraic perspective. Mislove et al. [19] have found a duality between labelled Markov processes and C^*-algebras based on the closely related classical Gelfand duality.

In a recent paper [15], a Stone-type duality was developed for Markov processes defined on continuous state spaces. The algebraic counterpart of the Markov processes were called Aumann algebras in honour of Aumann's work on probabilistic reasoning [1]. Aumann algebras capture, in algebraic form, a modal logic in which bounds on probabilities enter into the modalities. This logic can be stripped down to a very spartan core – just the modalities and finite conjunction – and still characterize bisimulation for labelled Markov processes [5, 6]. However, to obtain the strong completeness properties that are implied by the duality theorems, one needs infinitary proof principles [12].

One of the critiques [11] of logics and equivalences used for the treatment of probabilistic systems is that boolean logic is not robust with respect to small perturbations of the real-valued system parameters. Accordingly, a theory of metrics [7, 8, 24, 25] was developed and metric reasoning principles were advocated. In the present paper we extend our exploration of duality theory with an investigation into the role of metrics and exhibit a metric analogue of the duality theory. This opens the way for an investigation into quantitative aspects of *approximate reasoning*.

In the present paper we integrate quantitative information into the duality of [15] by endowing Markov processes with a (pseudo)metric and Aumann algebras with a quantitative "norm-like" structure called a *metric diameter*. The interplay between the pseudometric and the boolean algebra is somewhat delicate and had to be carefully examined for the duality to emerge. The final results have easy proofs but the correct way to impose quantitative structure on Aumann algebras was elusive. The key idea is to axiomatize the notion of metric diameter on the Aumann algebra side. This is a concept more like a norm than a distance, but one can derive a metric from it. The idea comes from a paper by Banaschewski and Pultr [2] on Stone duality for metric spaces. However, our formulation is not quite the same as theirs.

2 Background

Given a relation $\mathfrak{R} \subseteq M \times M$, the set $N \subseteq M$ is $\mathfrak{R}$-closed iff

$$\{m \in M \mid \exists n \in N, (n, m) \in \mathfrak{R}\} \subseteq N.$$

We assume that the reader is familiar with the definitions of field of sets, σ-algebra, measurable function, measurable space and measure.

If (M, Σ) is a measurable space and $\mathfrak{R} \subseteq M \times M$ is a relation on M, then $\Sigma(\mathfrak{R})$ denotes the set of measurable $\mathfrak{R}$-closed subsets of M.

Given a measurable space $\mathcal{M} = (M, \Sigma)$, we view the set $\Delta(M, \Sigma)$ of measures defined on Σ, as itself being a measurable space by endowing it with the σ-algebra $\mathfrak{F}$ generated by the sets $F(S, r) = \{\mu \in \Delta(M, \Sigma) : \mu(S) \geq r\}$ for arbitrary $S \in \Sigma$ and $r > 0$.

Measure theory works most smoothly in conjunction with certain topological assumptions. A *Polish space* is the topological space underlying a complete separable metric space. An *analytic* space is the image of a Polish space X under a continuous function $f : X \to Y$, where Y is also a Polish space. The special properties of analytic spaces were crucial in the proof of the logical characterization of bisimulation [6].

Let M be a set and $d : M \times M \to \mathbb{R}$.

Definition 2.1 We say that d is a *pseudometric* on M if for arbitrary $x, y \in M$,

$$(1)\colon\ d(x,x) = 0$$

$$(2)\colon\ d(x,y) = d(y,x)$$

$$(3)\colon\ d(x,y) \leq d(x,z) + d(z,y)$$

We say that (M, d) is a *pseudometric space.*

Pseudometrics arise as metric analogues of bisimulation [7, 8]. A pseudometric defines an equivalence relation called the *kernel* of the pseudometric, by $x \sim y$ iff $d(x, y) = 0$. The metrics defined in [7, 8] had bisimulation as the kernel.

In a pseudometric space (M, d), the *open ball* with center $x \in M$ and radius $\varepsilon > 0$ is the set $\{y \in M \mid d(x, y) < \varepsilon\}$. The collection of open balls forms a base for a topology called the *metric topology.* We can extend the pseudometric to sets in a manner analogous to the way in which one extends metrics to compact sets.

Definition 2.2 For a pseudometric d on M we define the *Hausdorff pseudometric* d^H on the class of subsets of X by

$$d^H(X,Y) = \max(\sup_{x\in X}\inf_{y\in Y} d(x,y), \sup_{y\in Y}\inf_{x\in X} d(x,y)).$$

We need to verify that these are indeed pseudometrics. The proof of the following lemma is omitted here; it is not too hard.

Lemma 2.3 *If $d : M \times M \to [0, 1]$ is a pseudometric on M, then d^H is a pseudometric on subsets of M.*

Markov processes (MPs) are models of probabilistic systems with a continuous state space and probabilistic transitions [6, 9, 20]. In earlier papers, they were called *labeled* Markov processes to emphasize the fact that there were multiple possible actions, but here we will suppress the labels, as they do not contribute any relevant structure for our results.

Definition 2.4 [Markov process] A *Markov process (MP)* is a tuple $\mathcal{M} = (M, \Sigma, \theta)$, where (M, Σ) is an analytic space and $\theta \in [\![M \to \Delta(M, \Sigma)]\!]$.

In a Markov process $\mathcal{M} = (M, \Sigma, \theta)$, M is the *support set*, denoted by supp$\mathcal{M}$, and θ is the *transition function.* For $m \in M$, $\theta(m) : \Sigma \to [0, 1]$ is a probability measure on the state space (M, Σ). For $N \in \Sigma$, the value $\theta(m)(N) \in [0, 1]$ represents the probability of a transition from m to a state in N.

The condition that θ is a measurable function $[\![M \to \Delta(M, \Sigma)]\!]$ is equivalent to the condition that for fixed $N \in \Sigma$, the function $m \mapsto \theta(m)(N)$ is a measurable function $[\![M \to [0, 1]]\!]$ (see e.g. Proposition 2.9 of [9]).

Markovian logic (ML) is a multi-modal logic for semantics based on MPs [1, 4, 10, 13, 16, 18, 26]. In addition to the Boolean operators, this logic is equipped with *probabilistic modal operators* L_r for $r \in \mathbb{Q}_0$ that bound the probabilities of transitions. Intuitively, the formula $L_r\varphi$ is satisfied by $m \in \mathcal{M}$ whenever the probability of a transition from m to a state satisfying φ is at least r.

Definition 2.5 [Syntax] The formulas of $\mathcal{L}$ are defined by the grammar

$$\varphi \ ::= \ \bot \mid \varphi \to \varphi \mid L_r\varphi$$

where r can be any element of $\mathbb{Q}_0$.

The Boolean operators $\vee$, $\wedge$, $\neg$, and $\top$ are defined from $\to$ and $\bot$ as usual. For $r_1, \ldots, r_n \in \mathbb{Q}_0$ and $\varphi \in \mathcal{L}$, let

$$L_{r_1 \cdots r_n}\varphi = L_{r_1} \cdots L_{r_n}\varphi.$$

The *semantics* for $L_r\varphi$ is defined as follows; the semantics of the other constructs are obvious. For MP $\mathcal{M} = (M, \Sigma, \theta)$ and $m \in M$,

$$\mathcal{M}, m \models L_r\varphi \text{ if } \theta(m)([\![\varphi]\!]) \geq r,$$

where $[\![\varphi]\!] = \{m \in M \mid \mathcal{M}, m \models \varphi\}$.

For this to make sense, $[\![\varphi]\!]$ must be measurable, this is readily verified. We define negation in the obvious fashion and use the words *valid* and *satisfiable* in the usual way.

This logic can be axiomatized using Hilbert-style axioms. The (strong) completeness of this logic is proved in [12, 18, 26] by assuming Lindenbaum's lemma (every consistent set can be expanded to a maximal consistent set) as a meta-axiom. The duality theorem of Kozen et al. [15] implies strong completeness without needing this assumption.

The logical equivalence induced by ML on the class of MPs coincides with bisimulation equivalence [6, 20]. The proof requires that the state space be an analytic space.

3 Stone Duality for Markov Processes

In this section we briefly summarize the results of our previous duality paper [15].

We introduce an algebraic version of Markovian logic consisting of a Boolean algebra with operators F_r for $r \in \mathbb{Q}_0$ corresponding to the operators L_r of ML. We call these algebras *Aumann algebras*. They are dual to certain Markov processes constructed from zero-dimensional Hausdorff spaces called *Stone–Markov processes* (SMPs).

Definition 3.1 [Aumann algebra] An *Aumann algebra (AA)* is a structure $\mathcal{A} = (A, \to, \bot, \{F_r\}_{r \in \mathbb{Q}_0}, \leq)$ where

- $(A, \to, \bot, \leq)$ is a Boolean algebra;
- for each $r \in \mathbb{Q}_0$, $F_r : A \to A$ is a unary operator; and
- the axioms in Table 1 hold for all $a, b \in A$ and $r, s, r_1, \ldots, r_n \in \mathbb{Q}_0$.

The Boolean operations $\vee$, $\wedge$, $\neg$, and $\top$, are defined from $\to$ and $\bot$ as usual.

Morphisms of Aumann algebras are Boolean algebra homomorphisms that commute with the operations F_r. The category of Aumann algebras and Aumann algebra homomorphisms is denoted **AA**.

We abbreviate $F_{r_1} \cdots F_{r_n} a$ by $F_{r_1 \cdots r_n} a$.

(AA1) $\top \leq F_0 a$

(AA2) $\top \leq F_r \top$

(AA3) $F_r a \leq \neg F_s \neg a$, $r + s > 1$

(AA4) $F_r(a \wedge b) \wedge F_s(a \wedge \neg b) \leq F_{r+s} a$, $r + s \leq 1$

(AA5) $\neg F_r(a \wedge b) \wedge \neg F_s(a \wedge \neg b) \leq \neg F_{r+s} a$, $r + s \leq 1$

(AA6) $a \leq b \Rightarrow F_r a \leq F_r b$

(AA7) $\left(\bigwedge_{r<s} F_{r_1 \cdots r_n r} a\right) = F_{r_1 \cdots r_n s} a$

Table 1
Aumann algebra

The operator F_r is the algebraic counterpart of the logical modality L_r. The first two axioms state tautologies, while the third captures the way F_r interacts with negation. Axioms (AA4) and (AA5) assert finite additivity, while (AA6) asserts monotonicity.

The most interesting axiom is the infinitary axiom (AA7). It asserts that $F_{r_1 \cdots r_n s} a$ is the greatest lower bound of the set $\{F_{r_1 \cdots r_n r} a \mid r < s\}$ with respect to the natural order $\leq$. In SMPs, it implies countable additivity.

As expected, the formulas of Markovian logic modulo logical equivalence form a free countable Aumann algebra. Define $\equiv$ on formulas in the usual way and let $[\varphi]$ denote the equivalence class of φ modulo $\equiv$, and let $\mathcal{L}{\equiv} = \{[\varphi] \mid \varphi \in \mathcal{L}\}$.

Theorem 3.2 *The structure*

$$(\mathcal{L}/{\equiv}, \to, [\bot], \{L_r\}_{r \in \mathbb{Q}_0}, \leq)$$

is an Aumann algebra, where $[\varphi] \leq [\psi]$ *iff* $\vdash \varphi \to \psi$.

A *Stone–Markov process* (SMP) is a Markov process $(M, \mathcal{A}, \theta)$, where $\mathcal{A}$ is a distinguished countable base of clopen sets that is closed under the set-theoretic Boolean operations and the operations

$$F_r(A) = \{m \mid \theta(m)(A) \geq r \mid\}, \quad r \in \mathbb{Q}_0.$$

The measurable sets Σ are the Borel sets of the topology generated by $\mathcal{A}$. Morphisms of such spaces are required to preserve the distinguished base; thus a morphism $f : \mathcal{M} \to \mathcal{N}$ is a continuous function such that

- for all $m \in M$ and $B \in \Sigma_{\mathcal{N}}$,

$$\theta_{\mathcal{M}}(m)(f^{-1}(B)) = \theta_{\mathcal{N}}(f(m))(B);$$

- for all $A \in \mathcal{A}_{\mathcal{N}}$, $f^{-1}(A) \in \mathcal{A}_{\mathcal{M}}$.

Unlike Stone spaces, SMPs are not topologically compact, but we do postulate a completeness property that is a weak form of compactness, which we call *saturation.* One can saturate a given SMP by a completion procedure that is reminiscent of Stone–Čech compactification [15]. Intuitively, one adds points to the structure without changing the represented algebra. An MP is *saturated* if it is maximal with respect to this operation.

Definition 3.3 [Stone–Markov Process] A Markov process $\mathcal{M} = (M, \mathcal{A}, \theta)$ with distinguished base is a *Stone–Markov process (SMP)* if it is saturated.

The morphisms of SMPs are just the morphisms of MPs with distinguished base as defined above.

The category of SMPs and SMP morphisms is denoted **SMP**.

Fix an arbitrary countable Aumann algebra

$$\mathcal{A} = (A, \to, \bot, \{F_r\}_{r \in \mathbb{Q}_0}, \leq).$$

Let $\mathcal{U}^*$ be the set of all ultrafilters of $\mathcal{A}$. The classical Stone construction gives a Boolean algebra of sets isomorphic to $\mathcal{A}$ with elements

$$(\!|a|\!)^* = \{u \in \mathcal{U}^* \mid a \in u\}, \qquad (\!|\mathcal{A}|\!)^* = \{(\!|a|\!)^* \mid a \in A\}.$$

The sets $(\!|a|\!)^*$ generate a Stone topology τ^* on $\mathcal{U}^*$, and the $(\!|a|\!)^*$ are exactly the clopen sets of the topology.

Let $\mathcal{F}$ be the set of elements of the form $\alpha^r = F_{t_1 \cdots t_n r} a$ for $a \in A$ and $t_1, \ldots, t_n, r \in \mathbb{Q}_0$. As before, we consider this term as parameterized by r; that is, if $\alpha^r = F_{t_1 \cdots t_n r} a$, then α^s denotes $F_{t_1 \cdots t_n s} a$. The set $\mathcal{F}$ is countable since A is. Axiom (AA7) asserts all infinitary conditions of the form

$$\alpha^s = \bigwedge_{r<s} \alpha^r. \tag{1}$$

for $\alpha^s \in \mathcal{F}$. Let us call an ultrafilter u *bad* if it violates one of these conditions in the sense that for some $\alpha^s \in \mathcal{F}$, $\alpha^r \in u$ for all $r < s$ but $\alpha^s \notin u$. Otherwise, u is called *good*. Let $\mathcal{U}$ be the set of good ultrafilters of $\mathcal{A}$.

Let $\tau = \{B \cap \mathcal{U} \mid B \in \tau^*\}$ be the subspace topology on $\mathcal{U}$, and let

$$(\!|a|\!) = \{u \in \mathcal{U} \mid a \in u\} = (\!|a|\!)^* \cap \mathcal{U} \qquad\qquad (\!|\mathcal{A}|\!) = \{(\!|a|\!) \mid a \in A\}.$$

Then τ is countably generated by the sets $(\!|a|\!)$ and all $(\!|a|\!)$ are clopen in the subspace topology.

We can now form a Markov process $\mathbb{M}(\mathcal{A}) = (\mathcal{U}, \Sigma, \theta)$, where Σ is the σ-algebra generated by $(\!|\mathcal{A}|\!)$ and $\theta : \mathcal{U} \to (\!|\mathcal{A}|\!) \to [0,1]$ is defined on the generators by

$$\theta(u)((\!|a|\!)) = \sup\{r \in \mathbb{Q}_0 \mid F_r a \in u\} = \inf\{r \in \mathbb{Q}_0 \mid \neg F_r a \in u\}.$$

It can be shown that θ extends uniquely to a transition function [15].

Theorem 3.4 *If $\mathcal{A}$ is a countable Aumann algebra, then $\mathbb{M}(\mathcal{A}) = (\mathcal{U}, (\!|\mathcal{A}|\!), \theta)$ is a Stone Markov process.*

Most of the technical difficulties of our earlier paper are in the proof of this theorem.

Let $\mathcal{M} = (M, \mathcal{B}, \theta)$ be a Stone Markov process with distinguished base $\mathcal{B}$. By definition, $\mathcal{B}$ is a field of clopen sets closed under the operations

$$F_r(A) = \{m \in M \mid \theta(m)(A) \geq r\}.$$

Theorem 3.5 *The structure $\mathcal{B}$ with the set-theoretic Boolean operations and the operations F_r, $r \in \mathbb{Q}_0$ is a countable Aumann algebra.*

We denote this algebra by $\mathbb{A}(\mathcal{M})$. Now we have the duality theorem.

Theorem 3.6 (Duality Theorem [15])

(i) *Any countable Aumann algebra $\mathcal{A}$ is isomorphic to $\mathbb{A}(\mathbb{M}(\mathcal{A}))$ via the map $\beta : \mathcal{A} \to \mathbb{A}(\mathbb{M}(\mathcal{A}))$ defined by*

$$\beta(a) = \{u \in \operatorname{supp}(\mathbb{M}(\mathcal{A})) \mid a \in u\} = (\!|a|\!).$$

(ii) *Any Stone Markov process $\mathcal{M} = (M, \mathcal{A}, \theta)$ is homeomorphic to $\mathbb{M}(\mathbb{A}(\mathcal{M}))$ via the map $\alpha : \mathcal{M} \to \mathbb{M}(\mathbb{A}(\mathcal{M}))$ defined by*

$$\alpha(m) = \{A \in \mathcal{A} \mid m \in A\}.$$

In [15], we also give a categorical version of this theorem with the contravariant functors between the two categories given explicitly.

4 Extending the Duality to Metrized Markov Processes

We add quantitative structure to both Markov processes and Aumann algebras. We prove an extended version of the representation theorem for metrized Markov processes versus metrized Aumann algebras. This theorem states that starting from an arbitrary metrized Markov process, we can extend the Aumann algebra constructed in the previous section to a metrized Aumann algebra that preserves the pseudometric and conversely. In other words the natural isomorphisms that arise in the duality of [15] will turn out to be isometries.

We equip Stone Markov processes with a pseudometric that measures distances between the states of the MP. The key condition that we impose is that for a particular state m, the diameters of the clopens containing m converges to 0.

Definition 4.1 [Metrized Markov process] A metrized Markov process is a tuple $(\mathcal{M}, d)$, where $\mathcal{M} = (M, \mathcal{A}, \theta)$ is a Stone Markov process with $\mathcal{A}$ its countable base of clopens and $d : M \times M \to [0,1]$ is a pseudometric on M satisfying for arbitrary $m \in M$ the property

$$\text{(M)} \quad \inf_{c \in \mathcal{A}, m \in c} \sup\{d(n,n') \mid n, n' \in c\} = 0.$$

The following lemma gives a number of conditions equivalent to (M). In particular, it shows the connection between the topology of the Stone Markov space and the pseudometric topology.

Lemma 4.2 *For a metrized MP $(\mathcal{M}, d)$, where $\mathcal{M} = (M, \mathcal{A}, \theta)$, the following are equivalent:*

(i) $\forall m, \quad \inf_{c \in \mathcal{A}, m \in c} \sup\{d(n,n') \mid n, n' \in c\} = 0$

(ii) $\forall m, m' \quad \inf_{c \in \mathcal{A}, m, m' \in c} \sup\{d(n,n') \mid n, n' \in c\} = d(m, m')$

(iii) $\forall m,\ \forall \varepsilon > 0\ \exists c \in \mathcal{A}\ (m \in c \wedge \forall n, n' \in c\ d(n,n') < \varepsilon)$

(iv) $\forall m\ \forall \varepsilon > 0\ \exists c \in \mathcal{A}\ (m \in c \wedge \forall n \in c\ d(m,n) < \varepsilon)$

(v) *The topology generated by $\mathcal{A}$ refines the pseudometric topology generated by d.*

(vi) *The pseudometric d is continuous in both arguments with respect to the $\mathcal{A}$-topology.*

Proof. Note that (i) is just (M).

(i) ⇔ (iii) is immediate from the definitions.

For (iii) ⇒ (iv), we can substitute m, n for n, n' in (iii).

For (iv) $\Rightarrow$ (iii), let m and $\varepsilon > 0$ be arbitrary, and let $c \in \mathcal{A}$ such that $m \in c$ and for all $n \in c$, $d(m,n) < \varepsilon/2$ and $d(n,m) < \varepsilon/2$. Then for any $n, n' \in c$, $d(n,n') \le d(n,m) + d(m,n') < \varepsilon/2 + \varepsilon/2 = \varepsilon$.

For (iv) $\Leftrightarrow$ (v), let $N_\varepsilon(m) = \{x \mid d(m,x) < \varepsilon\}$. Then,

$$\begin{aligned}
&\forall m\ \forall \varepsilon > 0\ \exists c \in \mathcal{A}\ (m \in c \wedge \forall n \in c\ d(m,n) < \varepsilon \wedge d(n,m) < \varepsilon)\\
\Leftrightarrow\ &\forall m\ \forall \varepsilon > 0\ \exists c \in \mathcal{A}\ (m \in c \wedge \forall n \in c\ n \in N_\varepsilon(m))\\
\Leftrightarrow\ &\forall m\ \forall \varepsilon > 0\ \exists c \in \mathcal{A}\ (m \in c \wedge c \subseteq N_\varepsilon(m)).
\end{aligned}$$

The last statement says that every basic open neighborhood of the pseudometric topology contains a basic open neighborhood of the $\mathcal{A}$-topology, which says exactly that the $\mathcal{A}$-topology refines the pseudometric topology.

For (iii) $\Rightarrow$ (vi), to show that d is continuous in its second argument, let m, x and $\varepsilon > 0$ be arbitrary and let $c \in \mathcal{A}$ such that $x \in c$ and for all $n, n' \in c$, $d(n,n') < \varepsilon$. Then for all $y \in c$,

$$\begin{aligned}
d(m,y) &\le d(m,x) + d(x,y) < d(m,x) + \varepsilon\\
d(m,x) &\le d(m,y) + d(y,x) < d(m,y) + \varepsilon
\end{aligned}$$

so $d(m,y) \in (d(m,x) - \varepsilon, d(m,x) + \varepsilon)$. That d is continuous in its first argument follows from symmetry.

For the other direction, suppose d is continuous in its second argument. Then for all m and $\varepsilon > 0$, the set $N_\varepsilon(m)$ is open and contains m, thus there exists a basic open set $c \in \mathcal{A}$ such that $m \in c$ and $c \subseteq N_\varepsilon(m)$. Thus the $\mathcal{A}$-topology refines the pseudometric topology of d.

Statement (ii) implies (i) immediately by taking $m' = m$ in (ii).

To show (i) implies (ii), let m, m' and $\varepsilon > 0$ be arbitrary. From (iv), we have $c \in \mathcal{A}$ such that $m \in c$ and from (iii) we have that for all $n, n' \in c$, $d(n,n') < \varepsilon/2$ and we have $c' \in \mathcal{A}$ such that $m' \in c'$ and for all $n, n' \in c'$, $d(n,n') < \varepsilon/2$. We claim that for all $n, n' \in c \cup c'$, $d(n,n') < d(m,m') + \varepsilon$, which will establish (ii).

If $n, n' \in c$, then

$$d(n,n') \le d(n,m) + d(m,n') < \varepsilon/2 + \varepsilon/2 = \varepsilon \le d(m,m') + \varepsilon.$$

If $n, n' \in c'$, the argument is the same, replacing m by m'. If $n \in c$ and $n' \in c'$, then

$$\begin{aligned}
d(n',n) = d(n,n') &\le d(n,m) + d(m,m') + d(m',n')\\
&< \varepsilon/2 + d(m,m') + \varepsilon/2 = d(m,m') + \varepsilon.
\end{aligned}$$

□

Definition 4.3 [Isometric Markov processes] Given two metrized MPs $(\mathcal{M}_i, d_i)$, where $\mathcal{M}_i = (M_i, \Sigma_i, \theta_i)$ for $i = 1, 2$, an isometry from $\mathcal{M}_1$ to $\mathcal{M}_2$ is a map $f : M_1 \to M_2$ such that for arbitrary $m, n \in M_1$,

$$d_1(m, n) = d_2(f(m), f(n)).$$

Now we introduce the metrized Aumann algebras. Despite their name, the metrized AAs are not directly equipped with a pseudometric structure, but with a concept of metric diameter. Later we will prove that the metric diameter can indeed define a pseudometric.

Definition 4.4 [Metrized Aumann algebra] A *metrized Aumann algebra* is a tuple $(\mathcal{A}, |\ |)$, where $\mathcal{A} = (A, \to, \bot, \{F_r\}_{r\in\mathbb{Q}_0}, \leq)$ is an Aumann algebra and $|\ | : A \to [0, 1]$ is a metric diameter on A, which is a map satisfying, for arbitrary $a, b \in A$ and ultrafilter u, the following properties

(A1) $|\bot| = 0$;

(A2) if $a \leq b$, then $|a| \leq |b|$;

(A3) if $a \wedge b \neq \bot$, then $|a \vee b| \leq |a| + |b|$;

(A4) $\inf\{|a| \mid a \in u\} = 0$.

Definition 4.5 [Isometric Aumann Algebras] Given two metrizable Aumann Algebras $(\mathcal{A}_i, |\ |_i)$ for $i = 1, 2$, an isometry from $\mathcal{A}_1$ to $\mathcal{A}_2$ is a map $f : A_1 \to A_2$ such that for any $a \in A_1$,

$$|a|_1 = |f(a)|_2.$$

We can now extend the duality results presented in the previous section to include the metric structure.

Consider a metrizable MP $(\mathcal{M}, d)$, where $\mathcal{M} = (M, \mathcal{A}, \theta)$. As before, let $\mathbb{A}(\mathcal{M})$ be the AA constructed from $\mathcal{M}$. We extend this construction so that $\mathbb{A}(\mathcal{M})$ becomes a metrized AA. For arbitrary $a \in \mathcal{A}$, let

$$|a|_d = \sup\{d(m, n) \mid m, n \in a\},$$

under the assumption that $\sup \varnothing = 0$.

Lemma 4.6 $(\mathbb{A}(\mathcal{M}), |\ |_d)$ *is a metrized Aumann Algebra.*

Proof. (A1) $|\bot|_d = 0$ follows from the assumption that $\sup \varnothing = 0$.

(A2) If $c_1 \subseteq c_2$, then $\sup\{d(m, n) \mid m, n \in c_1\} \leq \sup\{d(m, n) \mid m, n \in c_2\}$.

(A3) Suppose that $c_1 \cap c_2 \neq \varnothing$. Let $z \in c_1 \cap c_2$, $x \in c_1$ and $y \in c_2$. Then, the triangle inequality for d guarantees that

$$d(x, y) \leq d(x, z) + d(z, y) \leq \sup\{d(x, z) \mid x, z \in c_1\} + \sup\{d(z, y) \mid z, y \in c_2\} = |c_1| + |c_2|.$$

Consequently, $\sup\{d(x,y) \mid x \in c_1, y \in c_2\} \leq |c_1| + |c_2|$ and similarly, $\sup\{d(y,x) \mid x \in c_1, y \in c_2\} \leq |c_1| + |c_2|$. Since

$$|c_1 \cup c_2| = \max\{|c_1|, |c_2|, \sup\{d(x,y) \mid x \in c_1, y \in c_2\}, \sup\{d(x,y) \mid x \in c_2, y \in c_1\}\},$$

$$|c_1 \cup c_2| \leq |c_1| + |c_2|.$$

(A4) Using the notation of the previous duality theorem 3.6, we know that for any ultrafilter u of $\mathbb{A}(\mathcal{M})$ and any clopen $c \in \mathcal{A}$, we have that

$$\alpha^{-1}(u) \in c \text{ iff } c \in u.$$

Then, $\inf\{|c|_d \mid c \in u\} = \inf\{|c|_d \mid \alpha^{-1}(u) \in c\}$.

Since α is a bijection, $\alpha^{-1}(u) \in \mathcal{M}$ and $|c|_d = \sup\{d(n,n') \mid n, n' \in c\}$, using (M) we obtain

$$\inf_{c \in \mathcal{A}, c \ni \alpha^{-1}(u)} \sup\{d(n,n') \mid n, n' \in c\} = 0,$$

therefore $\inf\{|c|_d \mid c \in u\} = 0$. □

Consider a metrizable AA $(\mathcal{A}, |\ |)$ and, as before, let $\mathbb{M}(\mathcal{A})$ be the MP constructed from $\mathcal{A}$. We extend this construction so that $\mathbb{M}(\mathcal{A})$ will become a metrizable MP.

For arbitrary ultrafilters u, v of $\mathcal{A}$, let

$$\delta_{|\ |}(u,v) = \inf\{|a| \mid a \in u \cap v\}.$$

Lemma 4.7 $(\mathbb{M}(\mathcal{A}), \delta_{|\ |})$ *is a metrized MP.*

Proof. First, we prove that $\delta_{|\ |}$ is a pseudometric over the space of ultrafilters. From (A4) we can simply infer that $\delta_{|\ |}(u,u) = 0$, while the symmetry of $\delta_{|\ |}$ follows from the definition.

We now prove the triangle inequality: let u, v, w be three arbitrary ultrafilters. Let $a \in u \cap v$, $b \in u \cap w$ and $c \in w \cap v$. Obviously $b \cup c \in u \cap v$. Then,

$$\inf_{a \in u \cap v} |a| \leq |b \cup c|.$$

Since $b \cap c \neq \varnothing$, using (A3) we get $|b \cup c| \leq |b| + |c|$ which guarantees that for any $b \in u \cap w$ and any $c \in w \cap v$,

$$\inf_{a \in u \cap v} |a| \leq |b| + |c|,$$

implying

$$\inf_{a \in u \cap v} |a| \leq \inf_{b \in u \cap w} |b| + \inf_{c \in w \cap v} |c|.$$

Hence,

$$\delta_{|\ |}(u,v) \leq \delta_{|\ |}(u,w) + \delta_{|\ |}(w,v)$$

which proves that $\delta_{|\ |}$ is a pseudometric.

It remains to verify (M). Since from theorem 3.6 we know that $a \in u$ iff $u \in \beta(a)$, (M) follows directly from (A4). □

Finally, we extend the representation theorem 3.6 to include the metric structure. Essentially, we show that the isomorphisms α and β of the duality theorem are isometries.

Theorem 4.8 (The metric duality theorem) (i) *Any metrizable countable Aumann algebra* $(\mathcal{A}, |\ |)$ *is isomorphic to* $(\mathbb{A}(\mathbb{M}(\mathcal{A})), |\ |_{\delta_{|\ |}})$ *via the map* $\beta : \mathcal{A} \to \mathbb{A}(\mathbb{M}(\mathcal{A}))$ *defined by*

$$\beta(a) = \{u \in \mathrm{supp}(\mathbb{M}(\mathcal{A})) \mid a \in u\} = (\!|a|\!).$$

Moreover, β *is an isometry of metrizable Aumann algebras, i.e., for arbitrary* $a \in A$,

$$|a| = |\beta(a)|_{\delta_{|\ |}}.$$

(ii) *Any metrizable Stone Markov process* $(\mathcal{M}, d)$, *where* $\mathcal{M} = (M, \mathcal{A}, \theta)$ *is homeomorphic to* $(\mathbb{M}(\mathbb{A}(\mathcal{M})), \delta_{|\ |_d})$ *via the map* $\alpha : \mathcal{M} \to \mathbb{M}(\mathbb{A}(\mathcal{M}))$ *defined by*

$$\alpha(m) = \{A \in \mathcal{A} \mid m \in A\}.$$

Moreover, α *is an isometry of MPs, i.e., for arbitrary* $m, n \in M$,

$$d(m, n) = \delta_{|\ |_d}(\alpha(m), \alpha(n)).$$

Proof. We only need to prove the two isometries.

(i). The isometry of AAs. We need to prove that $|a| = |\beta(a)|_{\delta_{|\ |}}$.

Observe that

$$|\beta(a)|_{\delta_{|\ |}} = \sup_{u,v \in \beta(a)} \delta_{|\ |}(u, v) = \sup_{u,v \in \beta(a)} \inf_{a' \in u \cap v} |a'|.$$

Since $\beta(a)$ is the set of all ultrafilters containing a, a' quantifies over all elements that belong to the intersection of all ultrafilters containing a. But this intersection is nothing else but the principal filter $\uparrow a$ of a. Hence, the previous equality became

$$|\beta(a)|_{\delta_{|\ |}} = \inf_{a' \in \uparrow a} |a'|.$$

Now the monotonicity stated by (A2) guarantees that

$$\inf_{a' \in \uparrow a} |a'| = |a|.$$

(ii). The isometries of MPs. We need to prove that $d(m, n) = \delta_{|\ |_d}(\alpha(m), \alpha(n))$.

From Lemma 4.2(ii) we know that

$$d(m,n) = \inf_{c \in \mathcal{A}, c \ni m,n} |c|_d.$$

From Theorem 3.6 we also know that

$$m, n \in c \text{ iff } c \in \alpha(m) \cap \alpha(n).$$

Consequently,

$$d(m,n) = \inf\{|c|_d \mid c \in \alpha(m) \cap \alpha(n)\} = \delta_{|\ |_d}(\alpha(m), \alpha(n)).$$

□

We have claimed earlier that the metric diameter on an Aumann algebra induces a pseudometric. We now demonstrate this.

Let $(\mathcal{A}, |\ |)$ be a metrized AA. For arbitrary $a, b \in \mathcal{A}$ and $\varepsilon > 0$, let

$$B_\varepsilon(b) = \bigcup\{\beta(b') \mid b' \in \mathcal{A}, |b'| \leq \varepsilon, b \wedge b' \neq \bot\}.$$

Intuitively, $B_\varepsilon(b)$ is a ball that contains all ultrafilters that are at distance at most ε from some ultrafilter containing b. This definition allows us to define a natural distance on $\mathcal{A}$ by

$$d_{|\ |}(a,b) = \inf\{\varepsilon > 0 \mid B_\varepsilon(b) \supseteq \beta(a) \text{ and } B_\varepsilon(a) \supseteq \beta(b)\}.$$

Intuitively, if in the light of the duality we think of the elements of $\mathcal{A}$ as sets of ultrafilters, then the previous distance is just the Hausdorff pseudometric of the distance between ultrafilters.

To prove that the previous construction is not void, we show in the next lemma that for any non-zero element $a \in \mathcal{A}$ the ball $B_\varepsilon(a)$ is not empty for any ε.

Lemma 4.9 *If $a \neq \bot$, then for any $\varepsilon > 0$ there exists $a' \neq \bot$ such that $a \wedge a' \neq \bot$ and $|a'| \leq \varepsilon$.*

Proof. Since $a \neq \bot$, there exists an ultrafilter u such that $a \in u$. For any other $a' \in u$, $a \cap a' \in u$, hence $a \cap a' \neq \bot$. Moreover, using (A4) there exists $a' \in u$ such that $|a'| \leq \varepsilon$. □

Now we prove that $d_{|\ |}$ is indeed a pseudometric: it is the Hausdorff pseudometric of the pseudometric $\delta_{|\ |}$ on ultrafilters.

Lemma 4.10 *The function $d_{|\ |}$ previously defined on the support set of a metrizable Aumann Algebra is a pseudometric. Moreover,*

$$d_{|\ |}(a,b) = \max\{\sup_{u \in \beta(a)} \inf_{v \in \beta(b)} \delta_{|\ |}(u,v), \ \sup_{u \in \beta(a)} \inf_{v \in \beta(b)} \delta_{|\ |}(v,u)\}.$$

We omit the proof, which is not completely trivial, from this abstract.

4.1 Metric Duality in Categorical Form

We present the previous results in a more categorical format. The categories of metrized Aumann algebras (**MAA**) and metrized Markov processes (**MMP**) are defined as follows.

The objects of **MAA** are metrized AAs and their morphisms are expansive morphisms of AAs, i.e., morphisms $f : \mathcal{A}_1 \to \mathcal{A}_2$ of AAs such that for any $a \in \mathcal{A}_1$,

$$|a|_1 \le |f(a)|_2.$$

The objects of **MMP** are metrized MPs and their morphisms are non-expansive morphisms of MPs, i.e., morphisms $f : \mathcal{M}_1 \to \mathcal{M}_2$ of SMPs such that for any $m, n \in \mathcal{M}_1$,

$$d_1(m, n) \ge d_2(f(m), f(n)).$$

We define contravariant functors $\mathbb{A} : \mathbf{MMP} \to \mathbf{MAA}^{\mathrm{op}}$ and $\mathbb{M} : \mathbf{MAA} \to \mathbf{MMP}^{\mathrm{op}}$. The functor $\mathbb{A}$ on an object $\mathcal{M}$ produces the Aumann algebra $\mathbb{A}(\mathcal{M})$ defined in Theorem 3.5. On arrows $f : \mathcal{M} \to \mathcal{N}$ we define $\mathbb{A}(f) = f^{-1} : \mathbb{A}(\mathcal{N}) \to \mathbb{A}(\mathcal{M})$. We have previously proved that this is an Aumann algebra homomorphism. To see that it is also expansive, consider a morphism $f : \mathcal{M} \to \mathcal{N}$ such that for any $m, n \in \mathcal{M}$,

$$d_{\mathcal{M}}(m, n) \ge d_{\mathcal{N}}(f(m), f(n)).$$

Observe that for arbitrary $a \in \mathbb{A}(\mathcal{N})$,

$$|a|_{\mathbb{A}(\mathcal{N})} = |a|_{d_{\mathcal{N}}} \text{ and } |f^{-1}(a)|_{\mathbb{A}(\mathcal{M})} = |f^{-1}(a)|_{d_{\mathcal{M}}}$$

and in this context the previous inequality guarantees that

$$|a|_{\mathbb{A}(\mathcal{N})} \le |f^{-1}(a)|_{\mathbb{A}(\mathcal{M})}.$$

The functor $\mathbb{M} : \mathbf{MAA} \to \mathbf{MMP}^{\mathrm{op}}$ on an object $\mathcal{A}$ gives the Stone–Markov process $\mathbb{M}(\mathcal{A})$ defined in Theorem 3.4. On morphisms $h : \mathcal{A} \to \mathcal{B}$, it maps ultrafilters to ultrafilters by $\mathbb{M}(h) = h^{-1} : \mathbb{M}(\mathcal{B}) \to \mathbb{M}(\mathcal{A})$; that is,

$$\mathbb{M}(h)(u) = h^{-1}(u) = \{A \in \mathcal{A}_{\mathcal{N}} \mid h(A) \in u\}.$$

Another way to view $\mathbb{M}(h)$ is by composition, recalling that an ultrafilter can be identified with a homomorphism $\overline{u} : \mathcal{A} \to 2$ by $u = \{a \mid \overline{u}(a) = 1\}$. In this view,

$$\mathbb{M}(h)(\overline{u}) = \overline{u} \circ h,$$

where $\circ$ denotes function composition.

We have proven in our previous paper [15] that this is a morphism of SMPs.

That it is also non-expansive can be demonstrated as follows.

Let $h : \mathcal{A} \to \mathcal{B}$ be a morphism such that for arbitrary $a \in \mathcal{A}$,

$$|a|_{\mathcal{A}} \leq |f(a)|_{\mathcal{B}}.$$

Using the previous results it is not difficult to verify that for arbitrary ultrafilters u, v of $\mathcal{B}$,

$$\delta_{|\ |_{\mathcal{B}}}(u, v) \geq \delta_{|\ |_{\mathcal{A}}}(h^{-1}(u), h^{-1}(v)).$$

And since

$$\delta_{|\ |_{\mathcal{B}}}(u, v) = \delta_{\mathbb{M}(\mathcal{B})}(u, v) \text{ and } \delta_{|\ |_{\mathcal{A}}}(h^{-1}(u), h^{-1}(v)) = \delta_{\mathbb{M}(\mathcal{A})}(h^{-1}(u), h^{-1}(v)),$$

We obtain that $\mathbb{M}(h)$ is non-expansive.

Theorem 4.11 *The functors $\mathbb{M}$ and $\mathbb{A}$ define a dual equivalence of categories.*

$$\begin{tikzcd}
\mathbf{MMP} \arrow[r, bend left, "\mathbb{A}"] & \mathbf{MAA}^{\mathrm{op}} \arrow[l, bend left, "\mathbb{M}"]
\end{tikzcd}$$

5 Conclusions

We have extended the duality theory of [15] to a quantitative setting. It is important to note that the conditions we have imposed on the pseudometric relate the topology of the Markov process to the pseudometric topology. This can be seen from the fact that the pseudometric topology is refined by the Stone topology. We have defined our Stone Markov processes to be Hausdorff spaces, which was necessary for the duality theory. In effect, this means that the clopens separate points; in other words, one cannot have two states that satisfy exactly the same formulas. In view of the logical characterization of bisimulation, this implies that no two distinct states are bisimilar; that is, the process is already minimal with respect to bisimulation. If we look at a broader class of Markov processes, then we would have possibly nontrivial bisimulations on the space. The Stone Markov processes would be a reflective subcategory with the reflector sending each Markov process to a version of the process with all the bisimulation equivalence classes collapsed to point; this would be a Stone Markov process. How does the topology on a Stone Markov process "know" about the transition structure? Note that the base of clopens is required to be closed under the F_r operations, which are defined in terms of the transition function.

The only work of which we are aware similar to this is a paper by Banaschewski and Pultr [2] called "A Stone duality for metric spaces." They are not working with Markov processes, so there is nothing like the Aumann algebra structure there. They gave us the idea of using a metric diameter, but their axiomatization is different and the proofs that we developed do not resemble theirs.

The main impact of this work is to put quantitative reasoning about Markov processes on a firmer footing. It has been over a decade since metric analogues of bisimulation were developed, but they have not had the impact that they might have had. One reason is that with ordinary logical reasoning, one has a clear understanding of what completeness means, thus users of these logics have a good understanding of the power of the principles they are using. What does completeness mean for metric reasoning and approximate reasoning in general? The standard Stone-type duality theorem captures the concept of completeness; it is our hope that the present work will pave the way towards a similar understanding of approximate reasoning principles. There is much to be done, however. In a previous paper [17] we began investigating the relationship between the logic and metrics on Markov processes. The results of the present paper could perhaps strengthen and deepen these preliminary results.

Acknowledgments

Panangaden's research was supported by an NSERC grant. Mardare's research was supported by VKR Center of Excellence MT-LAB and by the Sino-Danish Basic Research Center IDEA4CPS. Kozen's research was supported by the National Security Agency.

References

[1] Robert Aumann. Interactive epistemology II: probability. *International Journal of Game Theory*, 28:301–314, 1999.

[2] B. Banaschewski and A. Pultr. Stone duality for metric spaces. In R. A. G. Seely, editor, *Category Theory 1991*, volume 13 of *CMS Conference Proceedings*, pages 33–42. CMS, 1991.

[3] M. M. Bonsangue and A. Kurz. Duality for logics of transition systems. In *FoSSaCS*, pages 455–469, 2005.

[4] Luca Cardelli, Kim G. Larsen, and Radu Mardare. Continuous markovian logic - from complete axiomatization to the metric space of formulas. In *CSL*, pages 144–158, 2011.

[5] J. Desharnais, A. Edalat, and P. Panangaden. A logical characterization of bisimulation for labelled Markov processes. In *proceedings of the 13th IEEE Symposium On Logic In Computer Science, Indianapolis*, pages 478–489. IEEE Press, June 1998.

[6] J. Desharnais, A. Edalat, and P. Panangaden. Bisimulation for labeled Markov processes. *Information and Computation*, 179(2):163–193, Dec 2002.

[7] J. Desharnais, V. Gupta, R. Jagadeesan, and P. Panangaden. Metrics for labeled Markov systems. In *Proceedings of CONCUR99*, number 1664 in Lecture Notes in Computer Science. Springer-Verlag, 1999.

[8] Josée Desharnais, Vineet Gupta, Radhakrishnan Jagadeesan, and Prakash Panangaden. A metric for labelled Markov processes. *Theoretical Computer Science*, 318(3):323–354, June 2004.

[9] E.-E. Doberkat. *Stochastic Relations. Foundations for Markov Transition Systems.* Chapman and Hall, New York, 2007.

[10] R. Fagin and J. Y. Halpern. Reasoning about knowledge and probability. *Journal of the ACM*, 41(2):340–367, 1994.

[11] A. Giacalone, C. Jou, and S. Smolka. Algebraic reasoning for probabilistic concurrent systems. In *Proceedings of the Working Conference on Programming Concepts and Methods*, IFIP TC2, 1990.

[12] R. Goldblatt. Deduction systems for coalgebras over measurable spaces. *Journal of Logic and Computation*, 20(5):1069–1100, 2010.

[13] Aviad Heifetz and Philippe Mongin. Probability logic for type spaces. *Games and Economic Behavior*, 35(1-2):31–53, April 2001.

[14] D. Kozen. A probabilistic PDL. *Journal of Computer and Systems Sciences*, 30(2):162–178, 1985.

[15] Dexter Kozen, Kim G. Larsen, Radu Mardare, and Prakash Panangaden. Stone duality for Markov processes. In *Proceedings of the 28th Annual IEEE Symposium On Logic In Computer Science*. IEEE Press, 2013.

[16] K. G. Larsen and A. Skou. Bisimulation through probablistic testing. *Information and Computation*, 94:1–28, 1991.

[17] Kim G. Larsen, Radu Mardare, and Prakash Panangaden. Taking it to the limit: Approximate reasoning for markov processes. In *Proceedings of the 37th International Symposium on the Mathematical Foundations of Computer Science*, volume 7464 of *Lecture Notes In Computer Science*, pages 681–692, 2012.

[18] Radu Mardare, Luca Cardelli, and Kim G. Larsen. Continuous markovian logics - axiomatization and quantified metatheory. *Logical Methods in Computer Science*, 8(4), 2012.

[19] M. Mislove, J. Ouaknine, D. Pavlovic, and J. Worrell. Duality for labelled Markov processes. In I. Walukiewicz, editor, *Foundations of Software Science and Computation Structures, FOSSACS*, volume 2987 of *Lecture Notes In Computer Science*, pages 393–407, 2004.

[20] Prakash Panangaden. *Labelled Markov Processes*. Imperial College Press, 2009.

[21] G. D. Plotkin. Lecture notes on domain theory. Available from his home page as The Pisa Notes, 1983.

[22] Gordon D. Plotkin. Dijkstra's predicate transformers and Smyth's power domains. In *Abstract Software Specifications*, volume 86 of *Lecture Notes in Computer Science*, pages 527–553. Springer, 1979.

[23] M. Smyth. Powerdomains and predicate transformers. In J. Diaz, editor, *Proceedings of the International Colloquium On Automata Languages And Programming*, pages 662–676. Springer-Verlag, 1983. Lecture Notes In Computer Science 154.

[24] Franck van Breugel and James Worrell. An algorithm for quantitative verification of probabilistic systems. In K. G. Larsen and M. Nielsen, editors, *Proceedings of the Twelfth International Conference on Concurrency Theory - CONCUR'01*, number 2154 in Lecture Notes In Computer Science, pages 336–350. Springer-Verlag, 2001.

[25] Franck van Breugel and James Worrell. Towards quantitative verification of probabilistic systems. In *Proceedings of the Twenty-eighth International Colloquium on Automata, Languages and Programming*. Springer-Verlag, July 2001.

[26] C. Zhou. *A complete deductive system for probability logic with application to Harsanyi type spaces*. PhD thesis, Indiana University, 2007.

Canonicity of Weak ω-Groupoid Laws Using Parametricity Theory

Marc Lasson

INRIA Paris-Rocquencourt, PiR2, Univ Paris Diderot, Sorbonne Paris Cité
F-78153 Le Chesnay France

Abstract

We show that terms witnessing a groupoid law from the ω-groupoid structure of types are all propositionally equal. Our proof reduce this problem to the unicity of the canonical point in the n-th loop space and conclude using Bernardy's parametricity theory for dependent types.

Keywords: type theory, parametricity, loop spaces, groupoids, identity types

1 Introduction

The synthetic approach to weak ω-groupoids promoted by the univalent foundation program [8] is the idea that (homotopy) type theory should be the primitive language in which spaces, points, paths, homotopies are derived. Following this approach, spaces are represented by types, points by inhabitants and paths by equalities between points (also known as identity types) and algebraic properties of these objects should not be enforced *a priori* (for instance by axioms) but should be derived directly from the language. To justify the synthetic approach, one should prove the canonicity of each definition, in the sense that no important choice should be made by choosing a particular implementation of a definition over another.

Garner, van den Berg [9] and Lumsdsaine [5] independently showed that in type theory, each type can be equipped with a structure of weak ω-groupoids. For this, they show that a minimal fragment of Martin-Löf type theory, where identity types are the only allowed type constructors, bears a weak ω-category structure. Informally, these results state the possibility to express algebraic properties of weak ω-groupoids as types and in each case to find a canonical inhabitant of these types reflecting the fact that the property holds. Identities, inversion and concatenation of

[1] Email: marc.lasson@inria.fr

path, associativities, involution of inversion, horizontal and vertical compositions of 2-paths, are all examples of groupoid laws. The *canonicity* of witnesses of groupoid laws here means there is a path between any two inhabitants of the law witnessing their equality, but also that this path should be canonical: there should be a path between any two paths between two inhabitants of the same law, and so on ... This canonicity is already a known fact within the fragment. The main result of this article is to extend the canonicity to the whole Martin-Löf type theory.

In this work, we follow a syntactic approach inspired by Brunerie [3] to formalize the notion of groupoid law. We call *groupoid law* any closed type $\forall\Gamma.c$ such that the sequent $\Gamma \vdash c : \mathsf{Type}$ is derivable in the minimal fragment and such that the context Γ is contractible. A *contractible context* is a context of the following shape: $A : \mathsf{Type}, a : A, x_1 : C_1, y_1 : M_1 = x_1, \ldots, x_n : C_n, y_n : M_n = x_n$ where x_i does not occur in M_i. The shape of these contexts is stable by path-induction, which allows to find an inhabitant of any groupoid law by successive path inductions. We show that this inhabitant is canonical, even outside of the fragment.

The main idea of the proof is to use successive path inductions to reduce the problem of the uniqueness of inhabitants of a given groupoid law to the uniqueness of the canonical point inhabiting a parametric loop space. Given a base type A and a point $a : A$, the n-th *loop space* and its *canonical point* are inductively defined by:

$$\Omega_0(A,a) := A \qquad\qquad \omega_0(A,a) := a$$
$$\Omega_{n+1}(A,a) := \Omega_n(a = a, 1_a) \qquad \omega_{n+1}(A,a) := \omega_n(a = a, 1_a)$$

where $1_a : a = a$ denotes the reflexivity. Thus for any integer n, $\forall X : \mathsf{Type}, x : X.\Omega_n(X,x)$ is a groupoid law inhabited by $\lambda X : \mathsf{Type}, x : X.\omega_n(X,x)$ (note that using one universe, it is possible to internalize the quantification over n; everything that we state here will be true whether or not this is used). We call this groupoid law the n-th *parametric loop space*.

The 0-th parametric loop space, is the polymorphic type $\forall X : \mathsf{Type}.X \to X$ of identity functions, and its canonical inhabitant is $\lambda X : \mathsf{Type}, x : X.x$, ie. the identity function. This term is the only one up to function extensionality inhabiting its type. The standard tool to prove this kind of properties is by using Reynold's parametricity theory [7] which was introduced to study the behavior of type quantifications within polymorphic λ-calculus (a.k.a. System F). It refers to the concept that well-typed programs cannot inspect types; they must behave uniformly with respect to abstract types. Reynolds formalizes this notion by showing that polymorphic programs satisfy the so-called logical relations defined by induction on the structure of types. This tool has been extended by Bernardy et al. [2] to dependent type systems. It provides a uniform translation of terms, types and contexts preserving typing (the so-called *abstraction theorem*). In its unary version (the only needed for this work), logical relations are defined by associating to any well-formed type $A : \mathsf{Type}$ a predicate $[\![A]\!] : A \to \mathsf{Type}$ and to any inhabitant $M : A$ a witness $[\![M]\!] : [\![A]\!]\, M$ that the M satisfies the predicate. This translation may be extended to cope with identity types by setting $[\![a = b]\!] : a = b \to \mathsf{Type}$ to be the predicate $\lambda p : a = b.p_*([\![a]\!]) = [\![b]\!]$ where p_* is the transport along p of the predicate generated by the common type of a and b. Then, it is easy -although quite verbose- to find a translation of introduction and elimination rules of identity types as well as checking that these translations preserve computation rules. This allows to extend Bernardy's abstraction theorem to identity types. Using this framework, we are able to generalize the uniqueness property of the polymorphic identity type to any parametric loop space. The proof proceed by induction on the index of the loop space and uses algebraic properties of transport.

Outline of the paper.

In Section 2, we introduce the type theoretical setting that is used in the article. Section 3 is devoted to the proof that Bernardy's parametricity may be extended to cope with identity types. In Section 4, we use this translation to prove the canonicity result for loop spaces; we prove that all inhabitants of parametric loop spaces are propositionally equal (Theorem 4.3). In Section 5, we introduce the fragment MLID of type theory to define our notion of groupoid laws and we show that the result of Section 4 may be generalized to all groupoid laws (Theorem 5.6). Finally Section 6 is devoted to various discussions.

2 Presentation of the syntax

We give a presentation of Martin-Löf type theory with identity types and universes which is close to the syntax of pure type systems [1] in order to reuse the parametricity theory presented in [2]. In this framework, computation rules are treated in an untyped way and subtyping of universes is achieved using Luo's cumulativity relation introduced for the extended calculus of constructions [6]. The reader may be more used to judgemental presentations of type theory where each computation steps are checked to be well-typed. This is just a matter of presentation; all the material presented here could be adapted without much effort to suit type systems using a judgemental equality. Also, the use of cumulativity is not really needed but makes our results more general and closer to implementations such as coq.
The terms of the system are given by the following grammar:

$$\begin{array}{rllllllllll} A,B,C,M,N,U,V := & x & \mid & (M\,N) & \mid & \lambda x:A.M & \mid & \forall x:A.B & \mid & \mathsf{Type}_i \\ & \mid & M =_A N & \mid & \mathbf{1}_M^C & \mid & \mathbf{J}_{\forall x:C,y:M=x.A}(B,U,V) \end{array}$$

where universes Type_i are indexed by $i \in \mathbb{N}$. Variables are considered up to α-conversion and we write $M[N/x]$ to denote the term obtained by substituting all free occurrences of x in M by N. To ease the reading of terms, we allow ourselves to omit some typing annotations when they could be guessed from the context (in particular $M = N$, $\mathbf{1}_M$ and $\mathbf{J}(B,U,V)$ will be used often in this text).

The grammar of terms is obtained by adding to the syntax of pure type systems:

- The type constructor $M =_A N$ for forming identity types,
- The introduction rule for identity types, $\mathbf{1}_M^C$, to witness the reflexivity $M =_C M$,

- The elimination rule (also known as "path-induction") $\mathbf{J}_{\forall x:C,y:M=x.A}(B,U,V)$. Given any dependent type $A(x,y)$ which depends on a point x and a path y from a base point M to x, given a witness for $A(M,\mathbf{1}_M)$ (the base case of the induction), given a point U and a path V, the path-induction provides an inhabitant of $A(U,V)$. This formulation of path-induction, due to Paulin-Möhring (also called *based path induction*), is equivalent to the other version where A is parametrized by two points and a path between them.

We use the symbol $\equiv$ to denote the syntactic equality (up to α-conversion) between terms. The conversion between terms will be denoted by $M \equiv_\beta N$, it is defined as the smallest congruence containing the usual β-reduction $(\lambda x : A.M)\, N \equiv_\beta M[N/x]$ and the computation rule for identity types: $\mathbf{J}_{\forall x:c,y:M=x,\Delta.P}(N,M,\mathbf{1}_M^C) \equiv_\beta N$. And the cumulativity order is defined as the smallest partial order $\preccurlyeq$ compatible with $\equiv_\beta$ and satisfying for $i \leq j$:

$$\forall x_1 : A_1, ..., x_n : A_n.\mathsf{Type}_i \preccurlyeq \forall x_1 : A_1, ..., x_n : A_n.\mathsf{Type}_j$$

Like in the extended calculus of constructions, the cumulativity rule is not fully contravariant with respect to the domain of functions ($A' \preccurlyeq A$ and $B \preccurlyeq B'$ does not imply $\forall x : A.B \preccurlyeq \forall x : A'.B'$) otherwise it would break the decidability of type checking. Contexts are finite lists of the form $x_1 : A_1, \cdots, x_n : A_n$ mapping a variable to its type. The rules of the type system are given in Figure 1. As we follow standard lines, we do not develop in details the metatheory of the system.

The non-dependent version of path-induction is called *transport* and is defined by

$$P_*^{x:C.X}(M) \equiv \mathbf{J}_{\forall x:C,y:U=x.X}(M,V,P)$$

where y does not occur in X. It is often used to coerce between type families: given a type family $X : C \to \mathsf{Type}_i$, a path $P : U = V$ between two points $U, V : C$, and a term M in $X\,U$, the transport $P_*^{x:C.X}(M)$ of M along P inhabits $X\,V$. It satisfies the following derivable rule :

$$\frac{\Gamma, x : C \vdash X : \mathsf{Type}_j \qquad \Gamma \vdash P : U =_C V \qquad \Gamma \vdash M : X[U/x]}{\Gamma \vdash P_*^{x:C.X}(M) : X[V/x]}\ \text{TRANSPORT}$$

We sometimes also omit the type family when it can be guessed from the context. The computation rule tells us that transporting along a reflexivity is same as doing nothing : $\mathbf{1}_{U*}(M) \equiv_\beta M$.

3 Extending Relational Parametricity to Identity types

In this section, we explain how to extend Bernardy's parametricity translation below to primitive identity types. Then we prove that this extension preserves typing (The-

$$\frac{}{\langle\rangle\ \mathbf{wf}}\ \text{WF-EMPTY} \qquad \frac{\Gamma\ \mathbf{wf} \qquad \Gamma \vdash B : \mathsf{Type}_i}{\Gamma, x : B\ \mathbf{wf}}\ \text{WF}$$

$$\frac{\Gamma\ \mathbf{wf}}{\Gamma \vdash \mathsf{Type}_i : \mathsf{Type}_{i+1}}\ \text{UNIV} \qquad \frac{\Gamma\ \mathbf{wf} \qquad (x : A) \in \Gamma}{\Gamma \vdash x : A}\ \text{VARIABLES}$$

$$\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x : A.M : \forall x : A.B}\ \text{ABSTRACTION} \qquad \frac{\Gamma \vdash M : \forall x : A.B \qquad \Gamma \vdash N : A}{\Gamma \vdash (M\,N) : B[N/x]}\ \text{APPLICATION}$$

$$\frac{\Gamma \vdash M : A' \qquad \Gamma \vdash A : \mathsf{Type}_i}{\Gamma \vdash M : A}\ A' \preccurlyeq A\ \ \text{CONVERSION} \qquad \frac{\Gamma \vdash A : \mathsf{Type}_i \qquad \Gamma, x : A \vdash B : \mathsf{Type}_i}{\Gamma \vdash \forall x : A.B : \mathsf{Type}_i}\ \text{PRODUCT}$$

$$\frac{\Gamma \vdash M : C \qquad \Gamma \vdash N : C \qquad \Gamma \vdash C : \mathsf{Type}_i}{\Gamma \vdash M =_C N : \mathsf{Type}_i}\ \text{IDENTITY} \qquad \frac{\Gamma \vdash M : C}{\Gamma \vdash \mathbf{1}_M^C : M =_C M}\ \text{REFLEXIVITY}$$

$$\frac{\begin{array}{c}\Gamma \vdash M : C \\ \Gamma, x : C, y : M = x \vdash P : \mathsf{Type}_i\end{array} \qquad \Gamma \vdash B : P[M/x, \mathbf{1}_M^C/y] \qquad \begin{array}{c}\Gamma \vdash U : C \\ \Gamma \vdash V : M = U\end{array}}{\Gamma \vdash \mathbf{J}_{\forall x:C,y:M=x.P}(B, U, V) : P[U/x, V/y]}\ \text{PATH-INDUCTION}$$

Fig. 1. Type theory with identity types (MLTT).

orem 3.2).

$$\begin{aligned}
\llbracket \forall x : A.B \rrbracket &\equiv \lambda f : \forall x : A.B.\forall x : A, x_R : x \in \llbracket A \rrbracket.(f\,x) \in \llbracket B \rrbracket \\
\llbracket \mathsf{Type}_i \rrbracket &\equiv \lambda x : \mathsf{Type}_i.x \to \mathsf{Type}_i \\
\llbracket \lambda x : A.M \rrbracket &\equiv \lambda x : A, x_R : x \in \llbracket A \rrbracket.\llbracket M \rrbracket \\
\llbracket M\,N \rrbracket &\equiv \llbracket M \rrbracket\, N\, \llbracket N \rrbracket \\
\llbracket x \rrbracket &\equiv x_R
\end{aligned}$$

where $M \in \llbracket A \rrbracket$ simply stands as a notation for $\llbracket A \rrbracket\, M$ (it makes formulas a bit easier to read). As said in the introduction, the predicate generated by an identity type $M =_C N$ is defined by the type family over $M =_C N$ selecting paths transporting $\llbracket M \rrbracket$ to $\llbracket N \rrbracket$:

$$\begin{aligned}
&\llbracket M =_C N \rrbracket\ :\ M =_C N \to \mathsf{Type} \\
&\llbracket M =_C N \rrbracket \equiv \lambda p : M =_C N.{p_*}^{x:C.x\in\llbracket C \rrbracket}\,(\llbracket M \rrbracket) =_{\llbracket C \rrbracket\, N} \llbracket N \rrbracket
\end{aligned}$$

Thanks to the computational rule

$$\mathbf{1}_M^C \in \llbracket M =_C M \rrbracket\ \equiv_\beta\ {\mathbf{1}_{M*}^C}^{x:c.x\in\llbracket C \rrbracket}\,(\llbracket M \rrbracket) =_{M\in\llbracket C \rrbracket} \llbracket M \rrbracket\ \equiv_\beta\ \llbracket M \rrbracket =_{M\in\llbracket C \rrbracket} \llbracket M \rrbracket$$

the translation $\llbracket \mathbf{1}_M^C \rrbracket : \mathbf{1}_M^C \in \llbracket M =_C M \rrbracket$ of reflexivity is a reflexivity: $\llbracket \mathbf{1}_M^C \rrbracket \equiv \mathbf{1}_{\llbracket M \rrbracket}^{M\in\llbracket C \rrbracket}$. And finally, the elimination rule is translated in terms of nested path-inductions:

$$\begin{array}{l}[\![\mathbf{J}_{\forall x:C,y:M=x.P}(B,U,V)]\!] \equiv \\ \quad \mathbf{J}_{\forall x_R:U\in[\![C]\!],y_R:V_*([\![M]\!])=x_R.\mathbf{J}_{\forall x:C,y:M=x.P}(B,U,V)\in[\![P]\!][U/x,V/y]} \\ \qquad (\mathbf{J}_{\forall x:C,y:M=x.\mathbf{J}_{\forall x:C,y:M=x.P}(B,x,y)\in[\![P]\!][y_*([\![M]\!])/x_R,\mathbf{1}_{y_*([\![M]\!])}/y_R]}([\![B]\!],U,V),[\![U]\!],[\![V]\!])\end{array}$$

The translation of the predicate of path inductions are quite verbose and make the translation hard to read. However if we ignore the annotation, we see that the translation $[\![\mathbf{J}(B,U,V)]\!] \equiv \mathbf{J}(\mathbf{J}([\![B]\!],U,V),[\![U]\!],[\![V]\!])$ is simply a duplication of path induction which should be compared to the duplication in abstractions and applications. We can check that this translation behaves well with respect to substitution and conversion :

Lemma 3.1 (Substitution and conversion lemma) *We have :*

(i) $[\![M[N/x]]\!] \equiv [\![M]\!][N/x,[\![N]\!]/x_R]$,

(ii) *If* $M \equiv_\beta M'$, *then* $[\![M]\!] \equiv_\beta [\![M']\!]$.

(iii) *If* $M \preccurlyeq M'$, *then* $[\![M]\!] \preccurlyeq [\![M']\!]$.

Proof. The proof of (i) is a routine proof by induction on M. And (iii) is a rather direct consequence of (ii). To prove (ii), we only do here the only check that is not in Bernardy's translation :

$$\begin{array}{rl}[\![\mathbf{J}(N,M,\mathbf{1}^C_M)]\!] & \equiv \mathbf{J}(\mathbf{J}([\![N]\!],M,\mathbf{1}^C_M),[\![M]\!],[\![\mathbf{1}^C_M]\!]) \\ & \equiv \mathbf{J}(\mathbf{J}([\![N]\!],M,\mathbf{1}^C_M),[\![M]\!],\mathbf{1}^{M\in[\![C]\!]}_{[\![M]\!]}) \\ & \equiv_\beta \mathbf{J}([\![N]\!],[\![M]\!],\mathbf{1}^{M\in[\![C]\!]}_{[\![M]\!]}]\!]) \\ & \equiv_\beta [\![N]\!]\end{array}$$ □

Now we can check that this extension preserve typing :

Theorem 3.2 (Abstraction) *If* $\Gamma \vdash M : A$, *then*

$$\begin{cases}[\![\Gamma]\!] \vdash M : A & (a) \\ [\![\Gamma]\!] \vdash [\![M]\!] : M \in [\![A]\!] & (b)\end{cases}$$

moreover if Γ **wf** *then* $[\![\Gamma]\!]$ **wf** *(c).*

Proof. The abstraction theorem is proved by induction on derivations. In each cases, proving the statement (a) is straightforward. Since the treatment of other rules is now standard, we only deal here with the rules concerning identity types. Even though we do not detailed it here, the treatment of CONVERSION uses previous lemma.

- IDENTITY: The induction hypothesis gives us $[\![\Gamma]\!] \vdash [\![M]\!] : M \in [\![C]\!]$ (1), $[\![\Gamma]\!] \vdash [\![N]\!] : N \in [\![C]\!]$ (2) and $[\![\Gamma]\!], x : C \vdash x \in [\![C]\!] : \mathsf{Type}_i$ (3). Using TRANSPORT we derive from (1) and (3), that $[\![\Gamma]\!], p : M =_C N \vdash p_*{}^{\lambda x:C.x\in C}([\![M]\!]) : N \in [\![C]\!]$. Then, using IDENTITY and (2) we build a derivation for $[\![\Gamma]\!], p : M =_C N \vdash$

${p_*}^{\lambda x:C.x\in C}(\llbracket M\rrbracket) =_{N\in\llbracket C\rrbracket} \llbracket N\rrbracket$ and finally we conclude $\llbracket\Gamma\rrbracket \vdash \llbracket M =_C N\rrbracket : M =_C N \to \mathsf{Type}_i$ with ABSTRACTION.

- REFLEXIVITY: By induction hypothesis, we have $\llbracket\Gamma\rrbracket \vdash \llbracket M\rrbracket : M \in \llbracket C\rrbracket$ (1). Using (1) we can check that $\llbracket\Gamma\rrbracket \vdash \mathbf{1}_{\llbracket M\rrbracket}^{M\in\llbracket C\rrbracket} : \llbracket M\rrbracket =_{M\in\llbracket C\rrbracket} \llbracket M\rrbracket$ which is convertible to $\llbracket\Gamma\rrbracket \vdash \mathbf{1}_{\llbracket M\rrbracket}^{M\in\llbracket C\rrbracket} : \mathbf{1}_{M*}^{C}(\llbracket M\rrbracket) =_{M\in\llbracket C\rrbracket} \llbracket M\rrbracket$. So, we conclude $\llbracket\Gamma\rrbracket \vdash \llbracket\mathbf{1}_M^C\rrbracket : \mathbf{1}_M^C \in \llbracket C\rrbracket$.
- PATH-INDUCTION: The induction hypothesis for (b) are :

$$\llbracket\Gamma\rrbracket \vdash \llbracket M\rrbracket : M \in \llbracket C\rrbracket \quad (1) \qquad \llbracket\Gamma, x : C, y : M = x\rrbracket \vdash \llbracket P\rrbracket : P \to \mathsf{Type}_i \quad (2)$$
$$\llbracket\Gamma\rrbracket \vdash \llbracket B\rrbracket : B \in \llbracket P[M/x, \mathbf{1}_M^C/y]\rrbracket \quad (3) \qquad \llbracket\Gamma\rrbracket \vdash \llbracket U\rrbracket : U \in \llbracket C\rrbracket \quad (4)$$
$$\llbracket\Gamma\rrbracket \vdash \llbracket V\rrbracket : V \in \llbracket M = U\rrbracket \quad (5)$$

and the induction hypothesis for (a) are :

$$\llbracket\Gamma\rrbracket \vdash M : C \quad (6) \qquad \llbracket\Gamma, x : C, y : M = x\rrbracket \vdash P : \mathsf{Type}_i \quad (7)$$
$$\llbracket\Gamma\rrbracket \vdash B : P[M/x, \mathbf{1}_M^C/y] \quad (8) \qquad \llbracket\Gamma\rrbracket \vdash U : C \quad (9)$$
$$\llbracket\Gamma\rrbracket \vdash V : M = U \quad (10)$$

Let T be the following term :

$$\mathbf{J}_{\forall x:C,y:M=x.\mathbf{J}(B,x,y)\in\llbracket P\rrbracket[y_*(\llbracket M\rrbracket)/x_R,\mathbf{1}_{y_*(\llbracket M\rrbracket)}/y_R]}(\llbracket B\rrbracket, U, V)$$

First, we have to typecheck T by showing that

$$\llbracket\Gamma\rrbracket \vdash T : \mathbf{J}(B, U, V) \in \llbracket P\rrbracket[y_*(\llbracket M\rrbracket)/x_R, \mathbf{1}_{y_*(\llbracket M\rrbracket)}/y_R, U/x, V/y] \qquad (*)$$

which means, using PATH-INDUCTION, checking :
· The predicate is well-formed :

$$\llbracket\Gamma\rrbracket, x : C, y : M = x \vdash \mathbf{J}(B, x, y) \in \llbracket P\rrbracket[y_*(\llbracket M\rrbracket)/x_R, \mathbf{1}_{y_*(\llbracket M\rrbracket)}/y_R] : \mathsf{Type}$$

This is obtained by substituting x_R and y_R in (2) using a derivation of

$$\llbracket\Gamma\rrbracket, x : C, y : M = x \vdash y_*(\llbracket M\rrbracket) : x \in \llbracket C\rrbracket$$

and of $\llbracket\Gamma\rrbracket, x : C, y : M = x \vdash \mathbf{1}_{y_*(\llbracket M\rrbracket)} : y_*(\llbracket M\rrbracket) = y_*(\llbracket M\rrbracket)$ which are easily derived from (1) and by applying $\llbracket\Gamma\rrbracket, x : C, y : M = x \vdash \mathbf{J}(B, x, y) : P$ to the substituted derivation.
· The arguments are correct : We derive

$$\llbracket\Gamma\rrbracket \vdash \llbracket B\rrbracket : \mathbf{J}(B, M, \mathbf{1}_M) \in \llbracket P\rrbracket[y_*(\llbracket M\rrbracket)/x_R, \mathbf{1}_{y_*(\llbracket M\rrbracket)}/y_R, M/x, \mathbf{1}_M/y]$$

by noticing using computation rules that :

$$\mathbf{J}(B, M, \mathbf{1}_M) \in \llbracket P\rrbracket[y_*(\llbracket M\rrbracket)/x_R, \mathbf{1}_{y_*(\llbracket M\rrbracket)}/y_R, M/x, \mathbf{1}_M/y]$$
$$\equiv_\beta B \in \llbracket P\rrbracket[M/x, \llbracket M\rrbracket/x_R, \mathbf{1}_M/y, \mathbf{1}_{y_*(\llbracket M\rrbracket)}/y_R]$$

and therefore CONVERSION and (3) allow us to conclude. Finally, we have to check that target point and path are correct ie. $[\![\Gamma]\!] \vdash U : C$ and $[\![\Gamma]\!] \vdash V : M = U$ which are given by (9) and (10).

Let K be the following term:

$$\mathbf{J}_{\forall x_R : U \in [\![C]\!], y_R : V_*([\![M]\!]) = x_R . \mathbf{J}_{\forall x:C, y:M=x.P}(B,U,V) \in [\![P]\!][U/x, V/y]}(T, [\![U]\!], [\![V]\!])$$

We now want to check that : $[\![\Gamma]\!] \vdash K : \mathbf{J}_{\forall x:C,y:M=x.P}(B, U, V) \in [\![P[U/x, V/y]]\!]$ using PATH-INDUCTION, we have to show that :

· The predicate is well-formed :

$$\begin{aligned}&[\![\Gamma]\!], x_R : U \in [\![C]\!], y_R : V_*([\![M]\!]) = x_R \vdash \\ &\qquad \mathbf{J}_{\forall x:C,y:M=x.P}(B, U, V) \in [\![P]\!][U/x, V/y] : \mathsf{Type}_i\end{aligned}$$

which is obtained by substituting x and y in (2) using (9) and (10) and by applying $[\![\Gamma]\!] \vdash \mathbf{J}_{\forall x:C,y:M=x.P}(B, U, V) : P[U/x, V/y]$ to the substituted derivation.

· The arguments are correct : We use (*) for type checking T and we use (4) and (5) for type checking $[\![U]\!]$ and $[\![V]\!]$. □

4 Canonicity in parametric loop spaces

The goal of this section is to prove Theorem 4.3. The following lemma is needed to perform the main induction in Lemma 4.2. The proofs terms are described in a semi-formal style for reading purpose, they could be easily constructed from the prose. However, as a consequence of Thereom 4.3, the precise shape of these terms is not important. In this section, we use $p \bullet q$ and p^{-1} to denote respectively the concatenation of paths and the inverse.

Lemma 4.1 *Let $P : A \to \mathsf{Type}$ be a type family and $u : P\,a$ for some $a : A$ and assume we have $\phi : \forall x : A. P\,x \to a = x$. Then, for all $p : a = a$ such that $p_*(u) = u$, we have $1 = p$.*

Proof. By applying the functorial action of path on $p_*(u) = u$ using ϕ_a we obtain a two dimensional path $\phi_a(p_*(u)) = \phi_a(u)$ and by the naturality of transport (Lemma 2.3.11 in [8]), we obtain a path $p_*(\phi_a(u)) = \phi_a(u)$. The left-hand side path $p_*(\phi_a(u))$ is transport over an identity type, we therefore have $p_*(\phi_a(u)) = \phi_a(u) \bullet p$ and therefore $\phi_a(u) \bullet p = \phi_a(u)$. So, we conclude $1 = p$ by cancelling by $\phi_a(u)$ and by symmetry. □

Here is the main induction which allows us to derive Theorem 4.3:

Lemma 4.2 *Let A, P, a, u and ϕ defined as in previous lemma. Then for all n-dimensional loop $p : \Omega_n(A, a)$, the type family $[\![\Omega_n]\!](A, P, a, u) : \Omega_n(A, a) \to \mathsf{Type}$ satisfies : $\forall p : \Omega_n(A, a).[\![\Omega_n]\!](A, P, a, u)\,p \to \omega_n(A, a) = p$.*

Proof. By induction on n, the base case is exactly witness by ϕ. For the inductive step, we need to prove $\forall p : \Omega_{n+1}(A, a).p \in [\![\Omega_{n+1}]\!](A, P, a, u) \to \omega_{n+1}(A, a) = p$

which is convertible to :

$$\forall p : \Omega_n(a = a, 1).p \in [\![\Omega_n]\!](a = a, \lambda q : a = a.q_*(u) = u, 1, 1) \to \omega_n(a = a, 1) = p$$

We may therefore apply the induction hypothesis by providing a proof that $\forall q : a = a.q_*(u) = u \to 1 = q$ which is given by previous lemma. □

We can deduce :

Theorem 4.3 (Canonicity for loop spaces) *If* $\vdash M : \forall A : \mathsf{Type}, a : A.\Omega_n(A, a)$ *then there is a term* π *such that* $\vdash \pi : \forall A : \mathsf{Type}, a : A.\omega_n(A, a) = M\,A\,a$.

Proof. The abstraction theorem gives us a proof $[\![M]\!]$ that $\forall A : \mathsf{Type}, A_R : A \to \mathsf{Type}, a : A, a_R : A_R\,a.(M\,A\,a) \in [\![\Omega_n]\!](A, A_R, a, a_r)$ by instantiating A_R with $\lambda x : A.a = x$ we can conclude by applying previous lemma. □

Note that as a corollary, the proof π is unique up to propositional equality (and so on). If we have two such proofs π and π' then $(\pi\,A\,a)\boldsymbol{\cdot}(\pi'\,A\,a)^{-1}$ is of type $\omega_n(A, a) = \omega_n(A, a)$ which is $\Omega_{n+1}(A, a)$. Therefore, by applying the previous theorem we obtain a proof of $(\pi\,A\,a)\boldsymbol{\cdot}(\pi'\,A\,a)^{-1} = \omega_{n+1}(A, a)$ which is also $(\pi\,A\,a)\boldsymbol{\cdot}(\pi'\,A\,a)^{-1} = \mathbf{1}_{\omega_n(A,a)}$. Therefore we conclude $(\pi\,A\,a) = (\pi'\,A\,a)$.

5 Groupoid laws

In this section, we start by describing a fragment MLID of the previous type system MLTT. This sub-system is used to characterise groupoid laws. Informally, it is obtained from MLTT by removing the rules ABSTRACTION, APPLICATION and UNIVERSES and by restricting valid sequents to the contractible contexts defined in the introduction. In the absence of function spaces, the rule PATH-INDUCTION has to be strengthened in order to be able to make a path induction along a path which is not the last one of the context. Therefore, we need to extend the grammar of terms in MLID with terms of the shape $\mathbf{J}_{\forall x:C,y:M=x,\Delta.P}(B, U, V, \overrightarrow{W})$ where Δ is a context and where the vectorial notation $\overrightarrow{W}$ denotes a tuple $(W_1, ..., W_n)$ of terms.

The typing rules are given in Figure 2. Formally a context Γ is said to be *contractible* if Γ **contr** is derivable; the reader should notice that all contexts occurring in derivations of MLID are contractible. In the typing rules, we write $\Gamma \vdash \overrightarrow{W} : \Delta$ to denote the conjunction for $k = 1, \cdots, n$ of $\Gamma \vdash W_k[A_1/y_1, \cdots, A_{k-1}/y_{k-1}] : A_k$ when Δ is of the shape $y_1 : A_1, \cdots, y_n : A_n$. Moreover, $[\overrightarrow{W}/\Delta]$ denotes the iterated substitution $[W_1/y_1, \cdots, W_n[A_1/y_1, \cdots, A_{n-1}/y_{n-1}]/y_n]$.

In order to embed MLID into MLTT, we translate extended path inductions into normal ones according to the following translation :

$$\begin{aligned}&\mathbf{J}_{\forall x:C,y:M=x,\Delta.P}(B, U, V, \overrightarrow{W}) \equiv \\ &\qquad \lambda \overrightarrow{x} : \Delta.(\mathbf{J}_{\forall x:C,y:M=x.\forall\Delta.P}(\lambda x : \Delta[M/x, \mathbf{1}_M^C/y].B, U, V)\,\overrightarrow{x})\end{aligned}$$

where $\overrightarrow{y} : \Delta$ means that $y_1, \cdots, y_n$ are the variables assigned in Δ, and where $\forall\Delta$, $\lambda\Delta$, and $(J\,\overrightarrow{x})$ denote respectively iterated products, abstractions and appli-

$$\frac{}{A : \mathsf{Type}_0, x : A\ \mathbf{contr}}$$

INIT

$$\frac{\Gamma\ \mathbf{contr} \qquad \Gamma \vdash_{\mathrm{id}} B : \mathsf{Type}_0 \qquad \Gamma \vdash_{\mathrm{id}} M : B}{\Gamma, x : B, p : M =_B x\ \mathbf{contr}}$$

CONTR

$$\frac{\Gamma \vdash_{\mathrm{id}} M : C \qquad \Gamma \vdash_{\mathrm{id}} N : C \qquad \Gamma \vdash_{\mathrm{id}} C : \mathsf{Type}_0}{\Gamma \vdash_{\mathrm{id}} M =_C N : \mathsf{Type}_0}$$

IDENTITY

$$\frac{\Gamma \vdash_{\mathrm{id}} M : C}{\Gamma \vdash_{\mathrm{id}} \mathbf{1}^C_M : M =_C M}$$

REFLEXIVITY

$$\frac{\Gamma\ \mathbf{contr} \qquad (x : A) \in \Gamma}{\Gamma \vdash_{\mathrm{id}} x : A}$$

VARIABLE

$$\frac{\Gamma \vdash_{\mathrm{id}} M : A' \qquad \Gamma \vdash_{\mathrm{id}} A' : \mathsf{Type}_0}{\Gamma \vdash_{\mathrm{id}} M : A}\ A' \equiv_\beta A$$

CONVERSION

$$\frac{\begin{array}{cc} \Gamma \vdash_{\mathrm{id}} M : C & \Gamma \vdash_{\mathrm{id}} U : C \\ \Gamma, x : C, y : M = x, \Delta \vdash_{\mathrm{id}} P : \mathsf{Type}_0 & \Gamma \vdash_{\mathrm{id}} V : M = U \\ \Gamma, \Delta[M/x, \mathbf{1}^C_M/y] \vdash_{\mathrm{id}} B : P[M/x, \mathbf{1}^C_M/y] & \Gamma \vdash_{\mathrm{id}} \overrightarrow{W} : \Delta \end{array}}{\Gamma \vdash_{\mathrm{id}} \mathbf{J}_{\forall x:C, y:M=x, \Delta . P}(B, U, V, \overrightarrow{W}) : P[U/x, V/y, \overrightarrow{W}/\Delta]}$$

EXTENDED-PATH-INDUCTION

Fig. 2. The minimal fragment MLID of type theory with identity types.

cations. It is then straightforward to check that EXTENDED-PATH-INDUCTION is an admissible rule of MLTT.

Groupoid laws are characterized by a contractible context and a derivable type in MLID (Figure 3 contains some examples of groupoid laws):

Definition 5.1 [Groupoid law] A *groupoid law* is a term of the shape $\forall \Gamma . C$ such that $\Gamma \vdash_{\mathrm{id}} C : \mathsf{Type}_0$.

The only groupoid laws in the "initial" contractible context $A : \mathsf{Type}, a : A$ are loop spaces. Since we do not change the base type and the base point of loop spaces in this section we simply denote by ω_n (resp. Ω_n) the terms $\omega_n(A, a)$ (resp. $\Omega_n(A, a)$). Using these notations we have :

Lemma 5.2 *If* $A : \mathsf{Type}, a : A \vdash_{\mathrm{id}} T : \mathsf{Type}_0$*, then*

(i) *there exists* $|T| \in \mathbb{N}$ *such that* $T \equiv_\beta \Omega_{|T|}$*,*

(ii) *moreover, if* $A : \mathsf{Type}, a : A \vdash_{\mathrm{id}} W : T$ *then* $W \equiv_\beta \omega_{|T|}$.

Proof. We proceed by induction on the derivation of $A : \mathsf{Type}, a : A \vdash_{\mathrm{id}} T : \mathsf{Type}_0$. We notice that the only two possible last used rules are IDENTITY and VARIABLE since Type_0 does not inhabit Type_0 it is not possible to invoke EXTENDED-PATH-INDUCTION nor CONVERSION.

(i) In the VARIABLE case, we necessarily have $T \equiv A$ and therefore we can conclude by taking $|T| = 0$. In the IDENTITY case, T is of the shape $M =_C N$ and by induction hypothesis $M \equiv_\beta \omega_{|C|}$, $N \equiv_\beta \omega_{|C|}$ and $C \equiv_\beta \Omega_{|C|}$. So we conclude by taking $|T| = |C| + 1$.

(ii) Without loss of generality (MLTT is normalizing) we can assume that W is

$$\begin{aligned}
\texttt{refl} : \forall X : \mathsf{Type}, x : X.x = x \qquad & \mathsf{sym} : \forall X : \mathsf{Type}, x : X.x = y \to y = x \\
\texttt{concat} : \forall X : \mathsf{Type}, x : X, y : X.x = y & \to \forall z : X.y = z \to x = z \\
\texttt{assoc} : \forall X : \mathsf{Type}, x : X, y : X, p : x = y, z : X, q & : y = z, t : X, r : z = t. \\
\texttt{concat}\, X\, x\, z\, (\texttt{concat}\, X\, x\, y\, p\, z\, q)\, t\, r & = \texttt{concat}\, X\, x\, y\, p\, t\, (\texttt{concat}\, X\, y\, z\, q\, t\, r) \\
\texttt{neutral} : \forall X : \mathsf{Type}, x : X, y : X, p : x = y.(\texttt{concat} & \, X\, x\, y\, p\, y\, (\texttt{refl}\, X\, y)) = p \\
\texttt{idem} : \forall X : \mathsf{Type}, x : X, y : X, p : x = y.\mathsf{sym} & \, X\, y\, x\, (\mathsf{sym}\, X\, x\, y\, p) = p \\
\texttt{horizontal} : \forall X : \mathsf{Type}, x : X, y : X, p : x = y, p' & : x = y.p = p' \to \\
\forall z : X, q : x = z, q' : x = z.q = q' \to \texttt{concat} & \, X\, x\, y\, p\, z\, q = \texttt{concat}\, X\, x\, y\, p'\, z\, q'
\end{aligned}$$

Fig. 3. Examples of groupoid laws with aliases for canonical inhabitants

in normal form. We proceed by a (nested) induction on the derivation $A : \mathsf{Type}, a : A \vdash_{\mathrm{id}} W : T$, we treat each possible case (IDENTITY and CONVERSION are obviously impossible since T cannot be Type_0):

- VARIABLE: In this case, we necessarily have $W \equiv a$ and $T \equiv A$. So we have $|T| = 0$ and $W \equiv \omega_0$.
- REFLEXIVITY: In this case, W and T are respectively of the shape $\mathbf{1}_M^C$ and $M =_C M$. By induction hypothesis, we have $M \equiv_\beta \omega_{|C|}$. Therefore $|T| = |C| + 1$ and $W \equiv_\beta \omega_{|C|+1}$.
- EXTENDED-PATH-INDUCTION: This is in fact an impossible case. We would have W of the shape $\mathbf{J}_{\forall x:C,y:M=x,\Delta.P}(B, U, V, \overrightarrow{Z})$ and by induction hypothesis, we would have $M \equiv_\beta \omega_{|C|} \equiv_\beta U$ and $V \equiv_\beta \omega_{|C|+1}$. Therefore V is a reflexivity and so W is not in normal form. □

The previous lemma allow us to find a canonic instantiation of any contractible context given by :

$$\begin{aligned}
&(A : \mathsf{Type}, a : A)^+ = (A, a) \\
&(\Gamma, x : A, y : M =_C x)^+ = (\Gamma^+, \omega_{|C[\Gamma^+/\Gamma]|}, \omega_{|C[\Gamma^+/\Gamma]|+1})
\end{aligned}$$

The following lemma state that this instantiation is correct :

Lemma 5.3

$$\begin{aligned}
\Gamma\ \mathbf{contr}\ \ &\textit{implies}\ \ A : \mathsf{Type}_0, a : A \vdash_{\mathrm{id}} \Gamma^+ : \Gamma && (1) \\
\Gamma \vdash_{\mathrm{id}} T : \mathsf{Type}_0\ \ &\textit{implies}\ \ T[\Gamma^+/\Gamma] \equiv_\beta \Omega_{|T[\Gamma^+/\Gamma]|} && (2) \\
\Gamma \vdash_{\mathrm{id}} T : \mathsf{Type}_0\ \textit{and}\ \Gamma \vdash_{\mathrm{id}} M : T\ \ &\textit{implies}\ \ M \equiv_\beta \omega_{|T[\Gamma^+/\Gamma]|} && (3)
\end{aligned}$$

Proof. We proceed by induction on size of derivations in MLID. We prove (1) by inspecting the last possible rule :

- INITIAL: Γ is of the shape $A : \mathsf{Type}_0, a : A$ and $\Gamma^+ = (A, a)$. By using two times VARIABLE, we check that : $(A : \mathsf{Type}_0, a : A) \vdash_{\mathrm{id}} (A, a) : (A : \mathsf{Type}_0, a : A)$.
- CONTRACTIBLE: Γ is of the shape $\Delta, x : B, p : M =_B x$ with $\Delta \vdash_{\mathrm{id}} B : \mathsf{Type}_0$ and $\Gamma \vdash_{\mathrm{id}} M : B$. By induction hypothesis, we have $A : \mathsf{Type}_0, a : A \vdash_{\mathrm{id}} \Delta^+ : \Delta$,

$B[\Delta^+/\Delta] \equiv_\beta \Omega_{|B[\Delta^+/\Delta]|}$ and $M[\Delta^+/\Delta] \equiv_\beta \omega_{|B[\Delta^+/\Delta]|}$. It is then easy to check using CONVERSION that we have $(A : \mathsf{Type}_0, a : A) \vdash_{\mathrm{id}} (\Delta^+, \omega_{|C[\Gamma^+/\Gamma]|}, \omega_{|c[\Gamma^+/\Gamma]|+1}) : (\Delta, x : B, p : M =_B x)$.

To prove (2) and (3), we notice that within the derivation of $\Gamma \vdash_{\mathrm{id}} T : \mathsf{Type}_0$ there is a strictly smaller derivation of Γ **contr** so we can use the induction hypothesis to obtain $A : \mathsf{Type}_0, a : A \vdash_{\mathrm{id}} \Gamma^+ : \Gamma$. Now by substitution, we derive that $A : \mathsf{Type}_0, a : A \vdash_{\mathrm{id}} T[\Gamma^+/\Gamma] : \mathsf{Type}_0$ and $A : \mathsf{Type}_0, a : A \vdash_{\mathrm{id}} M[\Gamma^+/\Gamma] : T[\Gamma^+/\Gamma]$. We conclude that $T[\Gamma^+/\Gamma] \equiv_\beta \Omega_{|T[\Gamma^+/\Gamma]|}$ and $M \equiv_\beta \omega_{|T[\Gamma^+/\Gamma]|}$ by previous lemma. □

Lemma 5.4 (All groupoid laws are inhabited) *If $\Gamma \vdash_{\mathrm{id}} T : \mathsf{Type}_0$, then there exists $\theta_{\Gamma.T}$ such that $\Gamma \vdash_{\mathrm{id}} \theta_{\Gamma.T} : T$.*

Proof. The contractible context Γ is of the shape

$$A : \mathsf{Type}_0, a : A, x_1 : C_1, y_1 : M_1 = x_1, \ldots, x_n : C_n, y_n : M_n = x_n$$

we construct $\theta_{\Gamma.c}$ by n successive extended path-inductions (from left to right). After all the inductions, it remains to find an inhabitant of $T[\Gamma^+/\Gamma]$ in the context $A : \mathsf{Type}_0, a : A$. But thanks to the previous lemma, we know that $T[\Gamma^+/\Gamma] \equiv_\beta \Omega_{|T[\Gamma^+/\Gamma]|}$, therefore using CONVERSION we can use $\omega_{|T[\Gamma^+/\Gamma]|}$. Spelled out, the term $\theta_{\Gamma.T}$ is :

$$\begin{aligned}\theta_{\Gamma.T} \equiv \mathbf{J}_{\forall x_1:C_1,y_1:M_1=x_1,\ldots,x_n:C_n,y_n:M_n=x_n.T}\big(&\\ \mathbf{J}_{\forall(x_2:C_2,y_2:M_2=x_2,\ldots,x_n:C_n,y_n:M_n=x_n.T)[\omega_{|C_1|}/x_1,\omega_{|C_1|+1}/y_1]}\big(&\\ \cdots \mathbf{J}_{\forall(x_n:C_n,y_n:M_n=x_n.T)[\Delta^+/\Delta]}\big(\omega_{|T[\Gamma^+/\Gamma]|}, x_n, y_n\big) \cdots, x_2, y_2\big), x_1, y_1\big)&\end{aligned}$$

where Δ is the context such that $\Gamma = \Delta, x_n : C_n, y_n : M_n = x_n$. □

As a corollary, we obtain the following theorem :

Theorem 5.5 (Canonicity for groupoid laws in MLID) *If $\Gamma \vdash_{\mathrm{id}} T : \mathsf{Type}$ and if we have two terms M and N such that $\Gamma \vdash_{\mathrm{id}} M : T$ and $\Gamma \vdash_{\mathrm{id}} N : T$, then there is a proof π such that $\Gamma \vdash \pi : M = N$.*

Proof. Simply notice that $\Gamma \vdash M = N : \mathsf{Type}_0$ and apply previous lemma. □

The reader should remark here that the proof π is also unique up to equality (by applying the theorem to $M = N$!). We are now ready to show that previous theorem may be generalized to the whole system MLTT by using parametricity theory.

Theorem 5.6 (Canonicity of inhabitants of groupoid laws in MLTT) *If $\Gamma \vdash_{\mathrm{id}} T : \mathsf{Type}$, $\vdash M : \forall\Gamma.T$ and $\vdash N : \forall\Gamma.T$, then there is a proof π such that $\overrightarrow{\gamma} : \Gamma \vdash \pi : (M\,\overrightarrow{\gamma}) = (N\,\overrightarrow{\gamma})$.*

Proof. By successive extended path-inductions (from left to right), we can derive $\overrightarrow{\gamma} : \Gamma \vdash (M\,\overrightarrow{\gamma}) = (N\,\overrightarrow{\gamma})$ from a derivation of $A : \mathsf{Type}_0, a : A \vdash (M\,\Gamma^+) = (N\,\Gamma^+)$. We notice that the type of $(M\,\Gamma^+)$ is $T[\Gamma^+/\Gamma]$ which is typable in MLID; by substitution we have $A : \mathsf{Type}_0, a : A \vdash_{\mathrm{id}} T[\Gamma^+/\Gamma] : \mathsf{Type}_0$. Therefore Lemma 5.3

gives us that $T[\Gamma^+/\Gamma] \equiv_\beta \Omega_n$ for some $n \in \mathbb{N}$. Using CONVERSION, we have $A : \mathsf{Type}_0, a : A \vdash (M\,\Gamma^+) : \Omega_n$. Therefore $(M\,\Gamma^+)$ is an inhabitant of a parametric loop space; therefore we can invoke Theorem 4.3 to obtain of proof that $(M\,\Gamma^+) = \omega_n$. Similarly we have a proof of $(N\,\Gamma^+) = \omega_n$ and by concatenating them we obtain a proof of $(M\,\overrightarrow{\gamma}) = (N\,\overrightarrow{\gamma})$. □

Finally, using the same arguments as at the end of Section 4 we can prove that π is unique up to propositional equality (and so on).

6 Discussions

6.1 The definition of groupoid laws

Our definition of MLID is inspired by an unpublished note written by Brunerie [3]. Our syntax for contractible contexts is a bit more general: in Brunerie's definition the starting point of paths occurring at odd positions are always variable (ie. M_k is always a variable) and there is no computation rules in his syntax. Brunerie defines ω-groupoids as models of its syntax, since the absence of computation rules makes his framework more free about how coherence issues are dealt with. However, the goal of our syntax is not give the general syntax for weak ω-groupoids but rather to study only the groupoid structure in the particular case of type theory where computation rules are the natural way to deal with coherence. Nevertheless, it would be an interesting future work to make a precise comparison between other definitions of ω-groupoids and models of MLID.

The semantical nature of Garner and van den Berg [9] makes it quite difficult to relate to our work, however we believe that Lumsdaine's construction [5] of a contractible globular operad may be described in our framework.

6.2 The n-ary case

Throughout this article, we only use the unary case of parametricity theory, but it could be easily generalized to the binary case by transporting along two paths : $[\![x = y]\!]_2 \equiv \lambda(p : x = y)(q : x' = y').p_*(q_*(x_R)) = y_R$. This translation is well-typed under the binary translation $[\![\alpha : \mathsf{Type}_i, x : \alpha, y : \alpha]\!]_2$ given by

$$\alpha\alpha' : \mathsf{Type}_i, \alpha_R : \alpha \to \alpha' \to \mathsf{Type}_i, x : \alpha, x' : \alpha', x_R : \alpha_R\,x\,x', y : \alpha, y' : \alpha', y_R : \alpha_R\,y\,y'$$

The translation $[\![\mathbf{1}^\alpha_x]\!]_2 : \mathbf{1}^\alpha_{x\,*}\big(\mathbf{1}^\alpha_{x'\,*}(x_R)\big) = x_R$ of $\mathbf{1}^\alpha_x$ is given by $[\![\mathbf{1}^\alpha_x]\!]_2 \equiv \mathbf{1}^{\alpha_R\,x\,x}_{x_R}$ which is defined under the translated context $[\![\alpha : \mathsf{Type}, x : \alpha]\!]_2$. Similarly, the translation of path-induction is obtained by nesting 2+1 path-induction:

$$[\![\mathbf{J}(B, U, V)]\!]_2 \equiv \mathbf{J}(\mathbf{J}(\mathbf{J}([\![B]\!]_2, U, V), U', V'), [\![U]\!]_2, [\![V]\!]_2)$$

It is a routine check to generalize the abstraction theorem of Section 3, in order to a have a binary version of parametricity. Likewise, the n-ary case, is obtained by transporting along n path and the translation of path-induction is obtained by nesting $n + 1$ path-inductions.

6.3 Encoding identity types with inductive families.

In dependent type systems that support inductive families, it is possible to encode identity types by an inductive predicate [4]. For instance, in the coq proof assistant:

```
Inductive paths (A : Type) (a : A) : A → Type := idpath : paths A a a.
```

As explained in [2] (Section 5.4), the parametricity translation extends well to inductive families. The idea is to translate an inductive type I by a new inductive I_R whose constructors are the translation of constructors of I. Likewise, the elimination scheme for I_R is the translation of the one of I.

```
Inductive paths_R (A : Type) (A_R : A → Type) (a : A) (a_R : A_R a) :
    forall x, A_R x → paths A a x → Type :=
  idpath_R : paths_R A A_R a a_R a a_R (idpath A a).
```

There is an equivalence of types (in the homotopy theory sense, see [8]) between this inductive type and our translation of identity. It is obtained by using the induction principle associated with $\mathtt{paths}_\mathtt{R}$ in one direction; and by nested path-inductions on p and on the proof of transport along p in the other direction. This indicates that they are morally the same; it may convince the reader that the results of the last two sections could have been carried out using the encoding instead of the primitive notion of identity types.

6.4 Dealing with axioms

While formalizing proofs that need axioms which are independent, it is a common practice to simply add them in the context. Then, if one want to use the parametricity translation, he also needs to provide a witness of the parametricity of the axiom. Therefore axioms that can prove their own parametricity are well-behaved with respect to the translation. More formally, we say that a closed type $P : \mathsf{Type}$ is *provably parametric* if the type $\forall h : P, h \in [\![P]\!]$ is inhabited. We will now give two examples of axioms using identity types which are provably parametric.

- *uniqueness of identity proofs* (UIP) : Let $\mathtt{uip}$ be the following type $\mathtt{uip} \equiv \forall X : \mathsf{Type}, x\, y : X, p\, q : x = y.p = q$. We want to find an inhabitant of $\forall f : \mathtt{uip}.f \in [\![\mathtt{uip}]\!]$. The statement $f \in [\![\mathtt{uip}]\!]$ unfolds into $\forall [\![X : \mathsf{Type}, x\, y : X, p\, q : x = y]\!].(f\, A\, x\, y\, p\, q)_*(p_R) = q_R$. The conclusion is an equality between paths, so it is provable using f.
- *function extensionality* : Let $\mathtt{funext}$ be the following type $\mathtt{funext} \equiv \forall A : \mathsf{Type}, B : A \to \mathsf{Type}, f\, g : \forall x : A.B x.(\forall x : A.f\, x = g\, x) \to f = g$. In his original development, Voedvoesky showed that $\mathtt{funext}$ is logically equivalent (there is a function in both directions) to the so-called *weak extensionality* (see [8]). It is defined by $\mathtt{weakext} \equiv \forall A : \mathsf{Type}, P : A \to \mathsf{Type}.(\forall x : A.\mathtt{Contr}\, P\, x) \to \mathtt{Contr}\, (\forall x : A.P\, x))$ where $\mathtt{Contr}\, A \equiv \exists x : A.\forall y : A.x = y$ is the predicate for *contractible types*; ie. types which are equivalent to the singleton type. We now sketch the proof that $\mathtt{weakext}$ is provably parametric (and thus so is $\mathtt{funext}$). The main idea is to notice that $M \in [\![\mathtt{Contr}\, A]\!]$ is logically equivalent to $\mathtt{Contr}\, ([\![A]\!]\, M_1)$

where M_1 is the first projection of M. The unfolding of $k \in \llbracket \mathtt{weakext} \rrbracket$ is:

$$\begin{array}{l}\forall A : \mathsf{Type}, A_R : A \to \mathsf{Type}, P : A \to \mathsf{Type}, P_R : (\forall x : A, x_R : A_R\, x.P\, x \to \mathsf{Type}),\\ \phi : (\forall x : A.\mathtt{Contr}\,(P\, x)), \phi_R : (\forall x : A, x_R : A_R\, x.(\phi\, x) \in \llbracket \mathtt{Contr}\, P\, x \rrbracket).\\ \qquad (k\, A\, P\, \phi) \in \llbracket \mathtt{Contr}\,(\forall x : A.P\, x) \rrbracket\end{array}$$

Therefore, using the logical equivalence in one direction we can deduce the conclusion from $\mathtt{Contr}\,(k\, A\, P\, \phi)_1 \in \llbracket \forall x : A.P\, x) \rrbracket)$. Then using $k : \mathtt{weakext}$ two times, it is enough to prove that $\forall x : A, x_R : A_R\, x.\mathtt{Contr}\,((k\, A\, P\, \phi)_1 x) \in \llbracket P\, x \rrbracket)$ (1). Notice that $(k\, A\, P\, \phi)_1\, x =_A (\phi\, x)_1$ because A is contractible. So we can transport along this path in (1) to obtain $\forall x : A, x_R : A_R\, x.\mathtt{Contr}\,((\phi x)_1 \in \llbracket P\, x \rrbracket)$ (2). Now, using the other direction of logical implication, (2) is implied by $\forall x : A, x_R : A_R\, x.(\phi x) \in \llbracket \mathtt{Contr}\,(P\, x) \rrbracket)$ which is exactly the type of ϕ_R.

7 Conclusion

This work shows that parametricity theory may be used to deduce properties about the algebraic structure of identity types. It allows to give formal arguments to prove canonicity results about definitions in a proof-relevant setting.

The most important question that remains open is whether or not we can extend the translation and the uniqueness property of groupoid laws to deal with Voevodsky's univalence axiom. One way to solve this question would be to prove that univalence axiom is provably parametric which would yield to a positive answer to the question of the compatibility of parametricity theory and univalence. Regardless of the answer to this problem, solving it would give a better understanding of polymorphic type quantifications in univalent universes.

References

[1] Henk Barendregt. Lambda Calculi with Types. In *Handbook of Logic in Computer Science*, pages 117–309. Oxford University Press, 1992.

[2] Jean-Philippe Bernardy, Patrik Jansson, and Ross Paterson. Proofs for free - parametricity for dependent types. *J. Funct. Program.*, 22(2):107–152, 2012.

[3] Guillaume Brunerie. Syntactic grothendieck weak ∞-groupoids. Available as http://uf-ias-2012.wikispaces.com/file/view/SyntacticInfinityGroupoidsRawDefinition.pdf.

[4] Peter Dybjer. Inductive families. *Formal Asp. Comput.*, 6(4):440–465, 1994.

[5] Peter LeFanu Lumsdaine. Weak omega-categories from intensional type theory. In Pierre-Louis Curien, editor, *TLCA*, volume 5608 of *Lecture Notes in Computer Science*, pages 172–187. Springer, 2009.

[6] Zhaohui Luo. Ecc, an extended calculus of constructions. In *LICS*, pages 386–395. IEEE Computer Society, 1989.

[7] John C. Reynolds. Types, Abstraction and Parametric Polymorphism. In *IFIP Congress*, pages 513–523, 1983.

[8] The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. http://homotopytypetheory.org/book, Institute for Advanced Study, 2013.

[9] Benno van den Berg and Richard Garner. Types are weak ω-groupoids. *Proceedings of the London Mathematical Society*, 102(2):370–394, 2011.

Relational Graph Models, Taylor Expansion and Extensionality

Giulio Manzonetto[1,2] Domenico Ruoppolo[1,3]

Laboratoire LIPN, CNRS UMR7030 Université Paris 13

Abstract

We define the class of relational graph models and study the induced order- and equational- theories. Using the Taylor expansion, we show that all λ-terms with the same Böhm tree are equated in any relational graph model. If the model is moreover extensional and satisfies a technical condition, then its order-theory coincides with Morris's observational pre-order. Finally, we introduce an extensional version of the Taylor expansion, then prove that two λ-terms have the same extensional Taylor expansion exactly when they are equivalent in Morris's sense.

Keywords: lambda calculus, linear logic, differential nets, extensional Böhm trees, Taylor expansion.

Introduction

An important problem in the theory of programming languages is to determine when two programs are equivalent. For λ-calculus, it has become standard to regard two programs M and N as equivalent when they are *contextually equivalent* with respect to some fixed set $\mathcal{O}$ of *observables*. This means that we can plug either M or N into any context $C(-)$, i.e. any program with a *hole*, without noticing any difference in the global behaviour: $C(M)$ reduces to an observable in $\mathcal{O}$ exactly when $C(N)$ does.

Two notable examples are $\equiv^{\text{hnf}}$ and Morris's equivalence $\equiv^{\text{nf}}$ [17] obtained by taking as observables the head normal forms and the β-normal forms, respectively. Working with these definitions is difficult because of the quantification over all possible contexts. However, researchers have found alternative characterisations of these program equivalences based on syntactic trees or denotational models.

For instance, two programs are equivalent with respect to $\equiv^{\text{hnf}}$ whenever they have the same Nakajima tree [18] or, equivalently, when their interpretations coin-

[1] This work is partly supported by ANR JCJC Project Coquas 12JS0200601.

[2] Email: giulio.manzonetto@lipn.univ-paris13.fr

[3] Email: domenico.ruoppolo@lipn.univ-paris13.fr

cide in Scott's model $\mathcal{D}_\infty$ [21]. Similarly, $\equiv^{\mathrm{nf}}$ is captured by extensional Böhm trees [14] and Coppo, Dezani and Zacchi's filter model $\mathcal{D}_{\mathrm{cdz}}$ [6].

The idea behind Böhm trees, and their extensional versions, is to extract the computational content of a program by representing its output as a possibly infinite tree — the continuity of this representation allows to infer properties of the whole tree by studying its finite approximants. For this reason Böhm-like trees and continuous models relied to them via approximation theorems constituted for over forty years the main tools to reason about the behaviour of a program. A limitation of these methods is that they abstract away from the execution process and overlook quantitative aspects such as the time, space, or energy consumed by a computation.

The present paper fits in a wider research programme whose aim is to rebuild the traditional theory of program approximations, by replacing it with a mathematical model of resource consumption. The starting point is [9], where Ehrhard and Regnier propose to analyse the behaviour of a program via its *Taylor expansion*, which is a generally infinite series of "resource approximants". Such approximants are terms of a *resource calculus* corresponding to a finitary fragment of the differential λ-calculus [7]. Each resource approximant t of a λ-term M captures a particular choice of the number of times M must call its sub-routines during its execution.

Both the differential λ-calculus and the Taylor expansion can be naturally interpreted in the relational semantics of linear logic [16]. The first author *et al.* built a relational model $\mathcal{D}_\omega$ living in such a semantics [5] and proved, using standard techniques, that the induced equality is exactly $\equiv^{\mathrm{hnf}}$ [15], just like for Scott's model $\mathcal{D}_\infty$ [12]. In this paper we provide syntactical and denotational methods based on Taylor expansion that allow to characterise Morris's equivalence $\equiv^{\mathrm{nf}}$.

First, we introduce the class of *relational graph models* (rgms) of λ-calculus, which are the relational analogous of graph models [3], and describe them as non-idempotent intersection type systems [19]. This class is general enough to encompass all relational models individually introduced in the literature [5,13], including $\mathcal{D}_\omega$ (while Scott's $\mathcal{D}_\infty$ cannot be a graph model since it is extensional). We then show that: (*i*) all rgms satisfy an approximation theorem for resource approximants (Theorem 3.10); (*ii*) in any rgm preserving the polarities of its "empty type" ω, β-normalisable λ-terms can be easily characterized (Lemma 4.3). As a consequence, we get that all extensional rgms preserving ω-polarities induce as order-theory Morris's observational pre-order, and hence $\equiv^{\mathrm{nf}}$ as equality (Corollary 4.6). As an instance, we provide the rgm $\mathcal{D}_\star$ generated by $\star \to \star \simeq \star$ where $\star$ is the only atom. It should be compared with the aforementioned filter model $\mathcal{D}_{\mathrm{cdz}}$, which has the same theory but is more complicated since it has two non-trivially ordered atoms $\varphi_\top \leq \varphi_\star$ and is generated by two equations $\varphi_\top \simeq \varphi_\star \to \varphi_\top$ and $\varphi_\star \simeq \varphi_\top \to \varphi_\star$.

Finally, we provide a notion of *extensional Taylor expansion* characterising, like extensional Böhm trees, Morris's equivalence while keeping the quantitative information. Intuitively, the extensional Taylor expansion of a λ-term is the η-normal form of its resource approximants. The definition is tricky because the η-reduction is meaningless on a *single* resource approximant — one should look at the whole series of approximants to decide whether an element should reduce or not. Our solution is to define a labeling as a global operation on the series of approximants, and then a local η-reduction on labeled terms. Two programs are then $\equiv^{\mathrm{nf}}$-equivalent exactly when they have the same extensional Taylor expansion (Theorem 5.17). We leave for future works a characterisation of Morris's preorder based on Taylor expansion.

Basic notations and conventions. We let $\mathbf{N}$ denote the set of natural numbers. Given a set A, $\mathcal{P}(A)$ (resp. $\mathcal{P}_{\mathrm{f}}(A)$) is the set of all (resp. finite) subsets of A and $\mathcal{M}_{\mathrm{f}}(A)$ is the set of all finite multisets over A. Finite multisets are represented as unordered lists $m = [\alpha_1, \dots, \alpha_n]$ with repetitions, $[]$ being the empty multiset.

Given a reduction $\to_{\mathtt{r}}$ we write $\twoheadrightarrow_{\mathtt{r}}$ ($=_{\mathtt{r}}$) for its transitive and reflexive (and symmetric) closure. A term t has an $\mathtt{r}$-normal form $\mathrm{nf}_{\mathtt{r}}(t)$, if $t \twoheadrightarrow_{\mathtt{r}} \mathrm{nf}_{\mathtt{r}}(t) \not\to_{\mathtt{r}}$.
N.B. Unless otherwise stated, throughout the paper we suppose that all operators $F : A \to B$ are extended to $\mathcal{P}(A)$ in the natural way: $F(a) = \{F(\alpha) \mid \alpha \in a\}$.

1 Lambda Calculus and Böhm Trees

We will generally use the notation of Barendregt's classic work [2] for λ-calculus. Let us fix an infinite set Var of variables. The set Λ of *λ-terms* is defined by:

$$\Lambda : \quad M, N, P ::= \ x \ \mid \ \lambda x.M \ \mid \ MN \qquad \text{for all } x \in \mathrm{Var}.$$

The set $\mathrm{fv}(M)$ of *free variables* of M and the α-conversion are defined as usual, see [2, Ch. 1§2]. A λ-term M is *closed* if $\mathrm{fv}(M) = \emptyset$. We denote by Λ^o the set of closed λ-terms. From now on, λ-terms will be considered up to α-conversion.

Given two λ-terms M, N we denote by $M\{N/x\}$ the capture-free substitution of N for all free occurrences of x in M. The β- and η-reductions are given for granted.

Concerning specific λ-terms, we fix the identity $\mathbf{I} = \lambda x.x$, its η-expansion $\mathbf{1} = \lambda xy.xy$, the paradigmatic looping term $\Omega = \Delta\Delta$ where $\Delta = \lambda x.xx$, Turing's fixpoint combinator $\Theta = \lambda f.\Theta_f\Theta_f$ where $\Theta_f = \lambda x.f(xx)$ and $\mathbf{J} = \Theta(\lambda zxy.x(zy))$ a term reducing to an infinite η-expansion of $\mathbf{I}$.

A λ-term M is called *solvable* if it has a *head normal form* (*hnf*, for short), that is if $M \twoheadrightarrow_\beta \lambda x_1 \dots x_n.yN_1 \cdots N_k$ (for $n, k \geq 0$); otherwise M is called *unsolvable*.

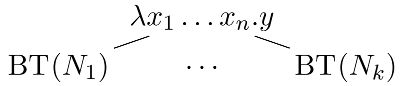
Given a *context* $C(-)$, i.e. a λ-term with a *hole* denoted by $(-)$, we write $C(M)$ for the λ-term obtained from C by substituting M for the hole possibly with capture of free variables in M. Given $\mathcal{O} \subseteq \Lambda$, the *$\mathcal{O}$-observational pre-order* is defined by:

$$M \sqsubseteq^{\mathcal{O}} N \iff \forall C(-)\ .\ C(M) \twoheadrightarrow_\beta M' \in \mathcal{O} \text{ entails } C(N) \twoheadrightarrow_\beta N' \in \mathcal{O}.$$

The induced *equivalence* $M \equiv^{\mathcal{O}} N$ is defined as $M \sqsubseteq^{\mathcal{O}} N$ and $N \sqsubseteq^{\mathcal{O}} M$. To obtain *Morris's pre-order* $\sqsubseteq^{\mathrm{nf}}$ and *equivalence* $\equiv^{\mathrm{nf}}$ just take as $\mathcal{O}$ the set of β-nfs [17].

The *Böhm tree* $\mathrm{BT}(M)$ of a λ-term M is defined coinductively: if M is unsolvable then $\mathrm{BT}(M) = \bot$; if M is solvable, then $M \twoheadrightarrow_\beta \lambda x_1 \dots x_n.yN_1 \cdots N_k$ and

$\mathrm{BT}(M) =$



Such a definition is sound in the sense that $M =_\beta N$ entails $\mathrm{BT}(M) = \mathrm{BT}(N)$. Examples of Böhm trees are: $\mathrm{BT}(\mathbf{I}) = \mathbf{I}$, $\mathrm{BT}(\mathbf{1}) = \mathbf{1}$, $\mathrm{BT}(\Delta) = \Delta$, $\mathrm{BT}(\Omega) = \bot$,

$$\mathrm{BT}(\lambda x.y\Omega) = \begin{array}{c}\lambda x.y\\ | \\ \bot\end{array} \qquad \mathrm{BT}(\mathbf{J}) = \begin{array}{c}\lambda xz_0.x\\ | \\ \lambda z_1.z_0\\ | \\ \lambda z_2.z_1\\ \vdots\end{array} \qquad \mathrm{BT}(\Theta) = \begin{array}{c}\lambda f.f\\ | \\ f\\ | \\ f\\ \vdots\end{array}$$

Given two Böhm trees T, T' we set $T \leq_\bot T'$ if and only if T results from T' by replacing some subtrees with $\bot$. The set $\mathcal{N}$ of *finite approximants* is the set of λ-terms possibly containing $\bot$ inductively defined as follows: $\bot \in \mathcal{N}$; if $a_i \in \mathcal{N}$ for $i = 1, \ldots, n$ then $\lambda\vec{x}.ya_1 \cdots a_n \in \mathcal{N}$. Hereafter we will confuse finite Böhm trees with normal approximants. Notice that the set of all finite approximants of a Böhm tree T, given by $T^* = \{a \in \mathcal{N} \mid a \leq_\bot T\}$, is an ideal with respect to $\leq_\bot$ [1, §2.3].

A λ-*theory* is any congruence on Λ containing $=_\beta$. A λ-theory is: *extensional* if it contains $=_\eta$; *sensible* if it equates all unsolvables. We denote by: $\lambda\beta\eta$ the least extensional λ-theory; $\mathcal{B}$ the λ-theory equating all λ-terms having the same Böhm tree; $\mathcal{B}\eta$ the least λ-theory containing $\mathcal{B}$ and $\lambda\beta\eta$; $\mathcal{H}^+$ (resp. $\mathcal{H}^*$) the λ-theory characterizing $\equiv^{\mathrm{nf}}$ (resp. $\equiv^{\mathrm{hnf}}$). From [2, Thm. 17.4.16] we get $\mathcal{B} \subsetneq \mathcal{B}\eta \subsetneq \mathcal{H}^+ \subsetneq \mathcal{H}^*$.

2 Resource Calculus and Taylor Expansion

We briefly recall Ehrhard's *resource calculus* [8], using the syntax proposed by Tranquilli in [22]. We are considering here the promotion-free fragment of [22].

Syntax. The set Λ^r of *resource terms* and the set Λ^b of *bags* are defined by:

$$\Lambda^r: \quad s, t ::= x \mid \lambda x.t \mid tb \qquad\qquad \Lambda^b: \quad b ::= [s_1, \ldots, s_n] \text{ where } n \geq 0. \tag{1}$$

Resource terms are in functional position, while bags are in argument position and represent unordered lists of resource terms. Intuitively, in a term of shape $t[s_1, \ldots, s_n]$ each s_i is a linear resource, that is t cannot duplicate nor erase it.

We will deal with bags as if they were multisets presented in multiplicative notation: 1 is the empty bag and $b_1 \cdot b_2$ is the multiset union of b_1 and b_2.

We use the power notation $[s^k]$ for the bag $[s, \ldots, s]$ containing k copies of s.

The α-equivalence and the set $\mathrm{fv}(t)$ of free variables of t are defined as for the ordinary λ-calculus. Resource terms and bags are considered up to α-equivalence.

As a syntactic sugar, we extend all the constructors of the grammar (1) as pointwise operations on (possibly infinite) sets of resource terms or bags. That is, given $\mathbb{T} \subseteq \Lambda^r$ and $\mathbb{B}, \mathbb{B}' \subseteq \Lambda^b$ we use the following notations: $\lambda x.\mathbb{T} = \{\lambda x.t \mid t \in \mathbb{T}\}$, $\mathbb{T}\mathbb{B} = \{tb \mid t \in \mathbb{T}, b \in \mathbb{B}\}$, $[\mathbb{T}] = \{[t] \mid t \in \mathbb{T}\}$ and $\mathbb{B} \cdot \mathbb{B}' = \{b \cdot b' \mid b \in \mathbb{B},\ b' \in \mathbb{B}'\}$.

Observe that, in the particular case of empty set, we get $\lambda x.\emptyset = \emptyset$, $t\emptyset = \emptyset$, $\emptyset b = \emptyset$, $[\emptyset] = \emptyset$ and $\emptyset \cdot b = \emptyset$. Hence, $\emptyset$ annihilates any resource term or bag.

This kind of meta-syntactic notation is discussed thoroughly in [9].

Reductions. Given a relation $\to_{\mathtt{r}} \subseteq \Lambda^r \times \mathcal{P}_{\mathrm{f}}(\Lambda^r)$ its *context closure* is the least relation in $\mathcal{P}_{\mathrm{f}}(\Lambda^r) \times \mathcal{P}_{\mathrm{f}}(\Lambda^r)$ such that, when $t \to_{\mathtt{r}} \mathbb{T}$, we have:

$$\lambda x.t \to_{\mathtt{r}} \lambda x.\mathbb{T}, \quad tb \to_{\mathtt{r}} \mathbb{T}b, \quad s([t]\cdot b) \to_{\mathtt{r}} s([\mathbb{T}]\cdot b), \quad \{t\}\cup\mathbb{S} \to_{\mathtt{r}} \mathbb{T}\cup\mathbb{S}.$$

We say that $t \in \Lambda^r$ *is in* $\mathtt{r}$*-normal form* if there is no $\mathbb{T}$ such that $t \to_{\mathtt{r}} \mathbb{T}$. When $\to_{\mathtt{r}}$ is confluent, $\mathrm{nf}_{\mathtt{r}}(t) \in \mathcal{P}_{\mathrm{f}}(\Lambda^r)$ denotes the unique $\mathtt{r}$-normal form of t, if it exists.

The *degree of* x *in* t, written $\deg_x(t)$, is the number of free occurrences of x in t. A β*-redex* is a resource term of the shape $(\lambda x.t)[s_1,\dots,s_k]$ and its *contractum* is a finite set of resource terms: when $\deg_x(t) = k$, it is the set of all possible resource terms obtained by linearly replacing each free occurrence of x in t by exactly one of the s_i's; otherwise, when $\deg_x(t) \neq k$, it is just $\emptyset$.

Formally, we define $\to_\beta$ as the context closure of:

$$(\lambda x.t)[s_1,\dots,s_k] \to_\beta \begin{cases} \bigcup_{p\in\mathfrak{S}_k} t\{s_{p(1)}/x_1,\dots,s_{p(k)}/x_k\} & \text{if } \deg_x(t) = k,\\ \emptyset & \text{otherwise.}\end{cases}$$

where $\mathfrak{S}_k$ is the group of permutations of $\{1,\dots,k\}$ and $x_1,\dots,x_n$ is an arbitrary enumeration of the free occurrences of x in t. Note that β-reduction is strongly normalizing (SN, for short) on $\mathcal{P}_{\mathrm{f}}(\Lambda^r)$, since whenever $t \to_\beta \mathbb{T}$ the size of t is strictly bigger than the size of each resource term in $\mathbb{T}$. Moreover, β-reduction is weakly confluent, and therefore confluent by Newman's lemma.

Theorem 2.1 *The* β*-reduction is strongly normalizing and confluent on* $\mathcal{P}_{\mathrm{f}}(\Lambda^r)$*.*

In the resource calculus there is no sensible notion of η-reduction on $\mathcal{P}_{\mathrm{f}}(\Lambda^r)$.

Taylor expansion. The *Taylor expansion* of a λ-term, as defined in [7,9], is a translation developing every λ-calculus application as an infinite series of resource applications with rational coefficients. For our purpose it is enough to consider a simplified version $\mathcal{T}(-) : \Lambda \to \mathcal{P}(\Lambda^r)$ corresponding to the support[4] of the actual Taylor expansion; that is, we consider possibly infinite sets of resource λ-terms.

Definition 2.2 The *Taylor expansion* $\mathcal{T}(M) \subseteq \Lambda^r$ of a λ-term M is defined by:

$$\mathcal{T}(x) = x, \qquad \mathcal{T}(\lambda x.M) = \lambda x.\mathcal{T}(M), \qquad \mathcal{T}(MN) = \mathcal{T}(M)\mathcal{M}_{\mathrm{f}}(\mathcal{T}(N)).$$

The Taylor expansion is extended to finite approximants in $\mathcal{N}$ by setting $\mathcal{T}(\bot) = \emptyset$, and to Böhm trees T by setting $\mathcal{T}(T) = \bigcup\{\mathcal{T}(a) \mid a \in T^*\}$.

Some examples of Taylor expansions of ordinary λ-terms are:

$$\mathcal{T}(\mathbf{I}) = \{\mathbf{I}\},\ \ \mathcal{T}(\Delta) = \{\lambda x.x[x^n] \mid n \geq 0\},\ \ \mathcal{T}(\lambda y.xyy) = \{\lambda y.x[y^n][y^k] \mid n,k \geq 0\},$$

[4] I.e., the set of those resource terms appearing in the series with a non-zero coefficient.

$$\begin{aligned}\mathcal{T}(\Omega) &= \{(\lambda x.x[x^{n_0}])[\lambda x.x[x^{n_1}],\ldots,\lambda x.x[x^{n_k}]] \mid k,n_0,\ldots,n_k \geq 0\},\\ \mathcal{T}(\Theta) &= \{\lambda f.(\lambda x.f[x[x^{n_1}],\ldots,x[x^{n_k}]])[\lambda x.f[x[x^{n_{1,1}}],\ldots,x[x^{n_{1,k_1}}]],\ldots,\\ &\qquad\qquad \lambda x.f[x[x^{n_{h,1}}],\ldots,x[x^{n_{h,k_h}}]]] \mid k,n_i,h,n_{i,j} \geq 0\},\\ \mathcal{T}(\mathbf{J}) &= \{t[\lambda zxy.x[z[y^{n_{1,1}}],\ldots,z[y^{n_{1,k_1}}]],\ldots,\\ &\qquad \lambda zxy.x[z[y^{n_{h,1}}],\ldots,z[y^{n_{h,k_h}}]]] \mid t \in \mathcal{T}(\Theta),\ h,k_i,n_{i,j} \geq 0\}.\end{aligned}$$

From the examples above it is clear that if a λ-term M has a β-redex, then there are resource terms $t \in \mathcal{T}(M)$ having β-redexes too. However, by Theorem 2.1, each t has a unique β-nf and we can always compute $\mathrm{nf}_\beta(\mathcal{T}(M)) = \bigcup\{\mathrm{nf}_\beta(t) \mid t \in \mathcal{T}(M)\}$. For instance: $\mathcal{T}(\mathbf{I})$, $\mathcal{T}(\Delta)$ and $\mathcal{T}(\lambda y.xyy)$ are already β-normal, while $\mathrm{nf}_\beta(\mathcal{T}(\Omega)) = \emptyset$.

Lemma 2.3 *Let $a \in \mathcal{N}$ and $M \in \Lambda$, then $\mathcal{T}(a) \subseteq \mathcal{T}(\mathrm{BT}(M))$ entails $a \in \mathrm{BT}(M)^*$.*

The following results proved in [8] show the strong relationship between the Böhm tree of a λ-term, and its Taylor expansion.

Theorem 2.4 *For every λ-term M, $\mathrm{nf}_\beta(\mathcal{T}(M)) = \mathcal{T}(\mathrm{BT}(M))$.*

Corollary 2.5 *For all $M,N \in \Lambda$, $\mathrm{BT}(M) = \mathrm{BT}(N)$ iff $\mathrm{nf}_\beta(\mathcal{T}(M)) = \mathrm{nf}_\beta(\mathcal{T}(N))$.*

Using Theorem 2.4, we can easily calculate further examples:
$\mathrm{nf}_\beta(\mathcal{T}(\Theta)) = \{\lambda f.f1, \lambda f.f[(f1)^n], \lambda f.f[f[(f1)^{n_1}],\ldots,f[(f1)^{n_k}]],\ldots\}$,
$\mathrm{nf}_\beta(\mathcal{T}(\mathbf{J})) = \{\lambda xz_0.x1, \lambda xz_0.x[(\lambda z_1.z_0 1)^n],\ldots\}$.

3 Relational Graph Models and Intersection Types

In this section we introduce the class of *relational graph models* (rgm, for short); some examples of such models were individually studied in [13].

3.1 Relational Graph Models

We call rgms *relational* because they are (linear) reflexive objects in the ccc **MRel** [5], the Kleisli category of **Rel** with respect to the comonad $\mathcal{M}_{\mathrm{f}}(-)$. In **MRel** the objects are all the sets, a morphism $f \in \mathbf{MRel}(A,B)$ is any relation between $\mathcal{M}_{\mathrm{f}}(A)$ and B, and the exponential object $A \Rightarrow B$ is given by $\mathcal{M}_{\mathrm{f}}(A) \times B$. Any function $f : A \to B$ can be sent to $f^\dagger \in \mathbf{MRel}(A,B)$ by setting $f^\dagger = \{([a], f(a)) \mid a \in A\}$.

Definition 3.1 A *relational graph model* $\mathcal{D} = (D,i)$ is given by an infinite set D and a total injection $i : \mathcal{M}_{\mathrm{f}}(D) \times D \to D$. $\mathcal{D}$ is *extensional* when i is bijective.

Every rgm $\mathcal{D} = (D,i)$ induces a reflexive object $(D, i^\dagger, (i^{-1})^\dagger)$, i.e. $D \Rightarrow D \lhd D$ since $i^\dagger; (i^{-1})^\dagger = \mathrm{Id}_{D \Rightarrow D}$. When $\mathcal{D}$ is moreover extensional we also have $(i^{-1})^\dagger ; i^\dagger = \mathrm{id}_D$. These reflexive objects are all *linear* in the sense of [16] and live in a differential ccc, they are therefore sound models of the resource calculus as well (Theorem 3.8).

Rgms, just like the regular ones [3], can be built by performing the free completion of a partial pair. A *partial pair* $\mathcal{A}$ is a pair (A,j) where A is a non-empty set of elements (called *atoms*) and $j : \mathcal{M}_{\mathrm{f}}(A) \times A \to A$ is a partial injection. We say that $\mathcal{A}$ is *extensional* when j is a bijection between $\mathrm{dom}(j)$ and A. Wlog., we will only consider partial pairs $\mathcal{A}$ whose underlying set A does not contain any pair.

Definition 3.2 The *completion* $\overline{\mathcal{A}}$ of a partial pair $\mathcal{A}$ is the pair $(\overline{A}, \overline{j})$ defined as: $\overline{A} = \bigcup_{n\in\mathbf{N}} A_n$, where $A_0 = A$ and $A_{n+1} = ((\mathcal{M}_{\mathrm{f}}(A_n) \times A_n) - \mathrm{dom}(j)) \cup A$; the function $\overline{j}$ is given by $\overline{j}(a,\alpha) = j(a,\alpha)$ if $(a,\alpha) \in \mathrm{dom}(j)$, $\overline{j}(a,\alpha) = (a,\alpha)$ otherwise.

Note that, for every rgm $\mathcal{D}$ we have $\overline{\mathcal{D}} = \mathcal{D}$ (up to isomorphism).

Proposition 3.3 *If $\mathcal{A}$ is a partial pair, then $\overline{\mathcal{A}}$ is an* rgm*. When $\mathcal{A}$ is extensional, also $\overline{\mathcal{A}}$ is extensional.*

Proof The proof of the fact that $\overline{\mathcal{A}}$ is an rgm is analogous to the one for regular graph models [3]. It is easy to check that when j is bijective, also $\overline{j}$ is. □

Example 3.4 We define the relational analogues of:

- Engeler's model [10]: $\mathcal{E} = \overline{(\mathbf{N}, \emptyset)}$, first defined in [13],
- Scott's model [21]: $\mathcal{D}_\omega = \overline{(\{\varepsilon\}, \{([], \varepsilon) \mapsto \varepsilon\})}$, first defined (up to iso) in [5],
- Coppo, Dezani and Zacchi's model [6]: $\mathcal{D}_\star = \overline{(\{\star\}, \{([\star], \star) \mapsto \star\})}$.

Notice that $\mathcal{D}_\omega$ and $\mathcal{D}_\star$ are extensional, while $\mathcal{E}$ is not.

3.2 Non-Idempotent Intersection Type Systems

As discussed thoroughly in [19], the choice of presenting a relational model as a reflexive object or as a non-idempotent intersection type system is more a matter of taste rather than a technical decision. Here we provide the latter presentation.

Let $\mathcal{A}$ be a partial pair and $\mathcal{D}$ be its completion. The set $\mathsf{T}_\mathcal{D}$ of *types* and the set $\mathsf{I}_\mathcal{D}$ of *non-idempotent intersections* are defined by mutual induction (for $\alpha \in A$):

$$\mathsf{T}_\mathcal{D} : \quad \sigma, \tau \ ::= \ \alpha \ \mid \ \mu \to \sigma \qquad\qquad \mathsf{I}_\mathcal{D} : \quad \mu, \nu \ ::= \ \omega \ \mid \ \sigma \ \mid \ \sigma \wedge \mu$$

Note that types are (unary) intersections while the converse does not hold; indeed intersections may only appear at the left-hand side of an arrow. Thus ω is not a type, it denotes the empty intersection and is therefore its neutral element ($\mu \wedge \omega = \mu$). Accordingly, we write $\wedge_{i=1}^n \sigma_n$ for $\sigma_1 \wedge \cdots \wedge \sigma_n$ when $n \geq 1$, and for ω when $n = 0$. Types will be considered up to associativity and commutativity of $\wedge$ and neutrality of ω, while we assume that the intersection is *not* idempotent, that is $\sigma \wedge \sigma \neq \sigma$.

Every $\sigma \in \mathsf{T}_\mathcal{D}$ ($\mu \in \mathsf{I}_\mathcal{D}$) corresponds to an element $\sigma^\bullet$ of D ($\mu^\bullet$ of $\mathcal{M}_{\mathrm{f}}(D)$) defined as $\alpha^\bullet = \alpha$, $(\mu \to \tau)^\bullet = i(\mu^\bullet, \tau^\bullet)$ and $(\sigma_1 \wedge \cdots \wedge \sigma_n)^\bullet = [\sigma_1^\bullet, ..., \sigma_n^\bullet]$. Hence, the model $\mathcal{D}$ induces a congruence on the intersection types: $\sigma \simeq^\mathcal{D} \tau$ if and only if $\sigma^\bullet = \tau^\bullet$.

An *environment* is a map $\Gamma : \mathrm{Var} \to \mathsf{I}_\mathcal{D}$ such that $\mathrm{dom}(\Gamma) = \{x \mid \Gamma(x) \neq \omega\}$ is finite. We write $x_1 : \mu_1, \ldots, x_n : \mu_n$ for the environment Γ such that $\Gamma(x_i) = \mu_i$ and $\Gamma(y) = \omega$ for all $y \notin \vec{x}$. The environment mapping all variables to ω is denoted by $\emptyset$, or just omitted as in Example 3.6. The intersection $\Gamma_1 \wedge \Gamma_2$ and the equivalence $\Gamma_1 \simeq^\mathcal{D} \Gamma_2$ of two environments are defined pointwise; note that $\Gamma \wedge \emptyset = \Gamma$.

Definition 3.5 The interpretation of $M \in \Lambda$ (or $M \in \mathcal{N}$) in $\mathcal{D}$ is defined as:

$$\frac{}{x : \sigma \vdash^{\mathcal{D}} x : \sigma}\ \texttt{var} \qquad \frac{\Gamma, x : \mu \vdash^{\mathcal{D}} M : \sigma}{\Gamma \vdash^{\mathcal{D}} \lambda x.M : \mu \to \sigma}\ \texttt{lam} \qquad \frac{\Gamma \vdash^{\mathcal{D}} M : \tau \quad \sigma \simeq^{\mathcal{D}} \tau}{\Gamma \vdash^{\mathcal{D}} M : \sigma}\ \texttt{eq}$$

$$\frac{\Gamma_0 \vdash^{\mathcal{D}} M : \wedge_{i=1}^{n} \sigma_i \to \tau \quad \Gamma_i \vdash^{\mathcal{D}} N : \sigma_i \quad \text{for } i = 1, \dots, n}{\Gamma_0 \wedge (\wedge_{i=1}^{n} \Gamma_i) \vdash^{\mathcal{D}} MN : \tau}\ \texttt{app}$$

(a) Non-idempotent intersection type system for Λ and $\mathcal{N}$.

$$\frac{\Gamma_0 \vdash^{\mathcal{D}} t : \wedge_{i=1}^{n} \sigma_i \to \tau \quad \Gamma_i \vdash^{\mathcal{D}} s_i : \sigma_i \quad \text{for } i = 1, \dots, n}{\Gamma_0 \wedge (\wedge_{i=1}^{n} \Gamma_i) \vdash^{\mathcal{D}} t[s_1, \dots, s_n] : \tau}\ \texttt{app}'$$

(b) Non-idempotent intersection type system for Λ^r.

Figure 1: The intersection type systems for Λ, $\mathcal{N}$ and Λ^r. The other rules for typing Λ^r are analogous to (`var`), (`lam`), (`eq`) of Figure 1(a) and are omitted.

$$[\![M]\!]^{\mathcal{D}} = \{(\Gamma, \sigma) \mid \Gamma \vdash^{\mathcal{D}} M : \sigma\}, \text{ where the type system } \vdash^{\mathcal{D}} \text{ is given in Fig. 1(a).}$$

The definition of $[\![t]\!]^{\mathcal{D}}$ for $t \in \Lambda^r$ is analogous, using the rules of Fig. 1(b). Note that $\vdash^{\mathcal{D}}$ also works for terms in $\mathcal{N}$: $\bot$ is not typable, but e.g. $\vdash^{\mathcal{D}} \lambda x.x\bot : (\omega \to \tau) \to \tau$.

Example 3.6 Let $\mathcal{D}$ be any rgm. Then we have: $[\![\mathbf{I}]\!]^{\mathcal{D}} = \{\sigma \mid \sigma \simeq \tau \to \tau,\ \tau \in \mathsf{T}_{\mathcal{D}}\}$, $[\![\mathbf{1}]\!]^{\mathcal{D}} = \{\sigma \mid \sigma \simeq (\mu \to \tau) \to \mu \to \tau,\ \tau \in \mathsf{T}_{\mathcal{D}}, \mu \in \mathsf{I}_{\mathcal{D}}\}$, $[\![\mathbf{J}]\!]^{\mathcal{D}} = \{\sigma \mid \sigma \simeq (\omega \to \tau) \to \omega \to \tau,\ \tau \in \mathsf{T}_{\mathcal{D}}\}$, $[\![\lambda x.x\Omega]\!]^{\mathcal{D}} = \{\sigma \mid \sigma \simeq (\omega \to \tau) \to \tau,\ \tau \in \mathsf{T}_{\mathcal{D}}\}$, $[\![\Omega]\!]^{\mathcal{D}} = \emptyset$. It follows that $[\![\mathbf{I}]\!] = [\![\mathbf{1}]\!]$ in both $\mathcal{D}_\omega$ and $\mathcal{D}_\star$, but $[\![\mathbf{I}]\!]^{\mathcal{D}_\omega} = [\![\mathbf{J}]\!]^{\mathcal{D}_\omega}$, while $\star \in [\![\mathbf{I}]\!]^{\mathcal{D}_\star} - [\![\mathbf{J}]\!]^{\mathcal{D}_\star}$.

When $\mathcal{D}$ is clear from the context we simply write $\simeq$, $\vdash$ and $[\![-]\!]$. Note that $\Gamma \vdash M : \sigma$ implies $\mathrm{dom}(\Gamma) \subseteq \mathrm{fv}(M)$ and $\Gamma' \vdash M : \sigma'$ for $\Gamma \simeq \Gamma'$ and $\sigma \simeq \sigma'$ [19].

Theorem 3.7 (Inversion Lemma, cf. [19]) *Let $\mathcal{D}$ be an* rgm*.*

(i) $\Gamma \vdash x : \sigma$ *entails* $\Gamma = x : \tau$ *for* $\tau \simeq \sigma$*,*

(ii) $\Gamma \vdash \lambda x.M : \sigma$ *if and only if* $\Gamma, x : \mu \vdash M : \tau$ *for some* $\mu \to \tau \simeq \sigma$*,*

(iii) $\Gamma \vdash MN : \sigma$ *entails that* $\Gamma = \Gamma_0 \wedge (\wedge_{i=1}^{n} \Gamma_i)$ *for some* $n \geq 0$*,* $\Gamma_0 \vdash M : \wedge_{i=1}^{n} \sigma_i \to \sigma$ *and* $\Gamma_i \vdash N : \sigma_i$*.*

For resource λ-terms an analogous statement holds, where (iii) is replaced with:

(iii') $\Gamma \vdash t[s_1, \dots, s_n] : \sigma$ *entails* $\Gamma = \Gamma_0 \wedge (\wedge_{i=1}^{n} \Gamma_i)$*,* $\Gamma_0 \vdash t : \wedge_{i=1}^{n} \sigma_i \to \sigma$ *and* $\Gamma_i \vdash s_i : \sigma_i$*.*

Theorem 3.8 *Let $\mathcal{D}$ be an* rgm*, then for Λ and Λ^r:*

(i) Substitution lemma, subject reduction and subject expansion hold in $\vdash^{\mathcal{D}}$*.*

(ii) The interpretation $[\![-]\!]^{\mathcal{D}}$ *is* sound *with respect to* $=_\beta$*.*

Proof (i) is proved in [19] for Λ and in [16] for relational models of Λ^r.
(ii) follows from (i). □

The *λ-theory* and the *order theory* induced by $\mathcal{D}$ are given by $\mathrm{Th}(\mathcal{D}) = \{(M, N) \mid [\![M]\!] = [\![N]\!]\}$ and $\mathrm{Th}_{\leq}(\mathcal{D}) = \{(M, N) \mid [\![M]\!] \subseteq [\![N]\!]\}$, respectively. We write $\mathcal{D} \models M = N$ if $(M, N) \in \mathrm{Th}(\mathcal{D})$, and $\mathcal{D} \models M \leq N$ if $(M, N) \in \mathrm{Th}_{\leq}(\mathcal{D})$.

A model $\mathcal{D}$ is *$\mathcal{O}$-inequationally fully abstract* when $\mathcal{D} \models M \leq N$ if and only if $M \sqsubseteq^{\mathcal{O}} N$, and *$\mathcal{O}$-fully abstract* when $\mathcal{D} \models M = N$ if and only if $M \equiv^{\mathcal{O}} N$.

Lemma 3.9 *If $\mathcal{D}$ is an extensional* rgm*, then $\lambda\beta\eta \subseteq \mathrm{Th}(\mathcal{D})$.*

Proof The equivalence between $\Gamma \vdash M : \sigma$ and $\Gamma \vdash \lambda x.Mx : \sigma$ when $x \notin \mathrm{fv}(M)$ follows by induction on σ using the fact that $\alpha \simeq \mu \to \tau$ for every atomic type α.□

As a consequence, the λ-theories induced by rgms and by regular graph models are different, since no graph model is extensional. For instance, the λ-theory of $\mathcal{D}_\omega$, the relational analogue of Scott's $\mathcal{D}_\infty$, is $\mathcal{H}^\star$ [15]. That is $\mathcal{D}_\omega$ is hnf-fully abstract.

While approximation theorems for Böhm trees and idempotent intersection type systems are usually proved through reducibility techniques, the following one for Taylor expansion and rgms can be proved by induction on the type derivation using the subject reduction (Theorem 3.8) and the SN of Λ^r (Theorem 2.1).

Theorem 3.10 (Approximation Theorem) *Let M be a λ-term. Then*

$$\Gamma \vdash M : \sigma \text{ if and only if there exists } t \in \mathcal{T}(M) \text{ such that } \Gamma \vdash t : \sigma.$$

Therefore $[\![M]\!] = [\![\mathcal{T}(M)]\!]$.

Corollary 3.11 *For all* rgms *$\mathcal{D}$ we have that $\mathcal{B} \subseteq \mathrm{Th}(\mathcal{D})$. In particular $\mathrm{Th}(\mathcal{D})$ is sensible and $[\![M]\!]^{\mathcal{D}} = \emptyset$ for all unsolvable λ-terms M.*

Proof From Theorem 3.10 we have $[\![M]\!] = [\![\mathcal{T}(M)]\!] = \bigcup_{t \in \mathcal{T}(M)} [\![t]\!]$. By subject reduction for Λ^r (Theorem 3.8) this is equal to $\bigcup_{t \in \mathcal{T}(M)} [\![\mathrm{nf}_\beta(t)]\!]$, which is equal to $\bigcup_{t \in \mathcal{T}(\mathrm{BT}(M))} [\![t]\!] = [\![\mathcal{T}(\mathrm{BT}(M))]\!]$, by Theorem 2.4. Therefore, whenever $\mathrm{BT}(M) = \mathrm{BT}(N)$ we get $[\![M]\!] = [\![\mathcal{T}(\mathrm{BT}(M))]\!] = [\![\mathcal{T}(\mathrm{BT}(N))]\!] = [\![N]\!]$. □

4 Full Abstraction for Morris's Observational Preorder

This section is devoted to show that every extensional rgm $\mathcal{D}$ satisfying the condition of Definition 4.1 — in particular $\mathcal{D}_\star$ — is (inequationally) fully abstract with respect to Morris's pre-order $\sqsubseteq^{\mathrm{nf}}$. Rather than working directly with $\sqsubseteq^{\mathrm{nf}}$, and building separating contexts, we use Levy's notion of *extensional Böhm tree*

$$\mathrm{BT}^e(M) = \{\mathrm{nf}_\eta(a) \mid a \in \mathrm{BT}(M')^*,\ M' \twoheadrightarrow_\eta M\}.$$

Indeed, it is well known that $M \sqsubseteq^{\mathrm{nf}} N$ exactly when $\mathrm{BT}^e(M) \subseteq \mathrm{BT}^e(N)$ [11] and that two λ-terms have the same extensional Böhm tree when their Böhm trees are equal up to (possibly infinitely many) η-expansions of *finite depth.* These trees are therefore different from Nakajima trees: for instance $\mathbf{I} \in \mathrm{BT}^e(\mathbf{I}) - \mathrm{BT}^e(\mathbf{J})$.

Examples of extensional Böhm trees are: $\mathrm{BT}^e(\mathbf{1}) = \mathrm{BT}^e(\mathbf{I})$,
$\mathrm{BT}^e(\mathbf{I}) = \{\bot, \mathbf{I}, \lambda xz_0.x\bot, \lambda xz_0.x(\lambda z_1.z_0(\lambda z_2.z_1\bot)), \dots\}$, $\mathrm{BT}^e(\mathbf{J}) = \mathrm{BT}^e(\mathbf{I}) - \{\mathbf{I}\}$,
$\mathrm{BT}^e(\lambda y.xyy) = \{\bot, x\bot, \lambda y.xyy, \lambda y.xy\bot, \dots\}, \mathrm{BT}^e(x\Omega) = \mathrm{BT}^e(\lambda y.xyy) - \{\lambda y.xyy\}$.

Given a polarity $p \in \{+, -\}$, we define inductively for all types σ the relations $\omega \in^p \sigma$ and $\omega \in^{\neg p} \sigma$, where $\neg p$ is the opposite polarity, as: (i) $\omega \in^- \mu \to \tau$ if

$\mu = \omega$; (ii) if $\omega \in^p \tau$ then $\omega \in^p \mu \to \tau$; (iii) if $\omega \in^{\neg p} \tau$ then $\omega \in^p \tau \wedge \mu \to \tau'$. When $\omega \in^+ \sigma$ $(\omega \in^- \sigma)$ we say that ω *occurs positively* (*negatively*) in σ. We write $\omega \notin^+ \sigma$ $(\omega \notin^- \sigma)$ if ω does not occur positively (negatively) in σ. These notions extend to intersections in the obvious way, for instance $\omega \in^p \sigma_1 \wedge \cdots \wedge \sigma_n$ if $\omega \in^p \sigma_i$ for some i.

Definition 4.1 An rgm $\mathcal{D}$ *preserves ω-polarities* whenever $\omega \in^p \sigma$ and $\sigma \simeq \tau$ entail $\omega \in^p \tau$, for all $\sigma, \tau \in \mathsf{T}_\mathcal{D}$ and $p \in \{+, -\}$.

For instance $\mathcal{E}$ and $\mathcal{D}_\star$ preserve ω-polarities, while $\mathcal{D}_\omega$ does not because $\omega \in^+ (\omega \to \varepsilon) \to \varepsilon \simeq \varepsilon \to \varepsilon$ but $\omega \notin^+ \varepsilon \to \varepsilon$. Note that, if an rgm $\mathcal{D}$ preserve ω-polarities, then we also have that $\omega \notin^p \sigma$ and $\sigma \simeq \tau$ entail $\omega \notin^p \tau$ (where $p \in \{+, -\}$).

Proposition 4.2 *Let $\mathcal{A}$ be a partial pair such that, for all $m \in \mathcal{M}_\mathrm{f}(A)$ and $\alpha \in A$, $(m, \alpha) \in \mathrm{dom}(j)$ entails that $m \neq []$. Then $\overline{\mathcal{A}}$ preserves ω-polarities.*

Lemma 4.3 *Let $M \in \Lambda$. The following are equivalent:*

(i) *M has a normal form,*

(ii) *there is $a \in \mathrm{BT}(M)^*$ that does not contain $\bot$,*

(iii) *there is $t \in \mathrm{nf}_\beta(\mathcal{T}(M))$ that does not contain the empty bag 1,*

(iv) *in every rgm $\mathcal{D}$ preserving ω-polarities, $\Gamma \vdash^\mathcal{D} M : \sigma$ for some environment Γ and type σ such that $\omega \notin^+ \sigma$ and $\omega \notin^- \Gamma$ (that is $\omega \notin^- \Gamma(x)$ for all $x \in \mathrm{Var}$).*

Proof [Sketch] ($i \iff ii$) is trivial and ($ii \iff iii$) follows from Theorem 2.4.

($iii \Rightarrow iv$) One proves by induction on the β-normal t that $\Gamma \vdash t : \sigma$ holds for some Γ, σ such that $\omega \notin^- \Gamma$ and $\omega \notin^+ \sigma$. Then one concludes by subject expansion for Λ^r and the approximation theorem (Theorem 3.10).

($iv \Rightarrow iii$) By the approximation theorem and subject reduction for Λ^r there is $t \in \mathrm{nf}_\beta \mathcal{T}(M)$ such that $\Gamma \vdash t : \sigma$ is derivable for some Γ, σ satisfying $\omega \notin^- \Gamma$ and $\omega \notin^+ \sigma$. Then, using Theorem 3.7 and the preservation of ω-polarities, one proves by induction on the structure of normal form of t that it does not contain 1. □

Notice that in the model $\mathcal{D}_\omega$, which does not preserve ω-polarities, the above lemma does not hold. For instance, $\omega \notin^+ \varepsilon \to \varepsilon \in [\![\mathbf{J}]\!]^{\mathcal{D}_\omega}$, but $\mathbf{J}$ is not normalizing.

In Coppo, Dezani and Zacchi's model $\mathcal{D}_\mathrm{cdz}$ presented in [6], there is an atomic type $\varphi_\star$ (resp. $\varphi_\top$) characterizing the terms having a β-nf (resp. persistent β-nf).

In the model $\mathcal{D}_\star$ the type $\star$ captures those λ-terms $M \in \Lambda^o$ having a normal form that is "linear". A λ-term M is called *linear* whenever: (i) every $y \in \mathrm{fv}(M)$ occurs once in M; (ii) every subterm $\lambda x.N$ of M is such that x occurs once in N.

Lemma 4.4 *Let $M \in \Lambda$ and $\Gamma = x_1 : \star, \ldots, x_n : \star$. Then $\Gamma \vdash^{\mathcal{D}_\star} M : \star$ if and only if M has a linear β-normal form and $\mathrm{fv}(\mathrm{nf}_\beta(M)) = \mathrm{dom}(\Gamma)$.*

We now prove the main results of the section.

Theorem 4.5 *Let $\mathcal{D}$ be an extensional rgm preserving ω-polarities. The following are equivalent (for $M, N \in \Lambda^o$):*

(i) *$\mathcal{D} \models M \leq N$,*

(ii) $M \sqsubseteq^{\mathrm{nf}} N$,

(iii) $\mathrm{BT}^e(M) \subseteq \mathrm{BT}^e(N)$.

Proof ($i \Rightarrow ii$) Suppose $[\![M]\!] \subseteq [\![N]\!]$ and consider a context $C(-)$ such that $C(M)$ has a normal form. By Lemma 4.3 there is $\sigma \in [\![C(M)]\!]$ such that $\omega \notin^+ \sigma$. Since $[\![-]\!]$ is contextual we have $[\![C(M)]\!] \subseteq [\![C(N)]\!]$, therefore $\sigma \in [\![C(N)]\!]$ and, by applying Lemma 4.3 again, we conclude that $C(N)$ has a normal form.

($ii \iff iii$) See Hyland's original paper [11], or [20] for a cleaner proof.

($iii \Rightarrow i$) We have:

$$\begin{array}{rll} [\![M]\!] & = \cup_{M' \twoheadrightarrow_\eta M}[\![M']\!] & \text{by Lemma 3.9} \\ & = \cup_{M' \twoheadrightarrow_\eta M}[\![\mathcal{T}(M')]\!] & \text{by Theorem 3.10} \\ & = \cup_{M' \twoheadrightarrow_\eta M}[\![\mathrm{nf}_\beta \mathcal{T}(M')]\!] & \text{by Theorem 3.8}(ii) \text{ for } \Lambda^r \\ & = \cup_{M' \twoheadrightarrow_\eta M}[\![\mathrm{BT}(M')^*]\!] & \text{by Theorem 2.4} \\ & = \cup_{M' \twoheadrightarrow_\eta M}[\![\mathrm{nf}_\eta \mathrm{BT}(M')^*]\!] & \text{by Lemma 3.9} \\ & = [\![\mathrm{BT}^e(M)]\!] & \text{by definition of } \mathrm{BT}^e(M). \end{array}$$

Thus $\mathrm{BT}^e(M) \subseteq \mathrm{BT}^e(N)$ entails $[\![M]\!] = [\![\mathrm{BT}^e(M)]\!] \subseteq [\![\mathrm{BT}^e(N)]\!] = [\![N]\!]$. □

Corollary 4.6 (Full abstraction) *Every extensional* rgm $\mathcal{D}$ *respecting ω-polarities has order-theory* $\mathrm{Th}_\leq(\mathcal{D}) = \{(M,N) \mid M \sqsubseteq^{\mathrm{nf}} N\}$ *and λ-theory* $\mathrm{Th}(\mathcal{D}) = \mathcal{H}^+$.

5 Extensional Taylor Expansion and η-Trees

We introduce the notion of *extensional Taylor expansion $\mathcal{T}^\eta(M)$ of a λ-term M* and prove that it is equal to the Taylor expansion of the extensional Böhm tree of M (Theorem 5.15). This result is the analogue of Theorem 2.4. As a byproduct, we obtain a new syntactical characterization of $\equiv^{\mathrm{nf}}$ (Corollary 5.17).

For technical reasons, we work with an alternative notion of extensional Böhm tree of M, that will be denoted by $\mathrm{BT}^\eta(M)$. Rather than producing a set of η-normal approximants, $\mathrm{BT}^\eta(-)$ gives an actual (possibly infinite) η-normal tree. The *η-normal form $\eta(T)$ of a Böhm tree T* is defined coinductively: $\eta(\bot) = \bot$ and

$$\eta\left(\begin{array}{ccc} & \lambda x_1 \ldots x_n.y & \\ / & & \backslash \\ T_1 & \cdots & T_m \end{array}\right) = \begin{cases} \eta\left(\begin{array}{ccc} & \lambda x_1 \ldots x_{n-1}.y & \\ / & & \backslash \\ T_1 & \cdots & T_{m-1} \end{array}\right) & \begin{array}{l}\text{If } x_n \notin \mathrm{fv}(yT_1 \cdots T_{m-1}) \\ T_m \in \mathcal{N}, \text{ i.e. it is finite} \\ \text{and } T_m \twoheadrightarrow_\eta x_n,\end{array} \\ \begin{array}{ccc} & \lambda x_1 \ldots x_n.y & \\ / & & \backslash \\ \eta(T_1) & \cdots & \eta(T_m) \end{array} & \text{otherwise.} \end{cases}$$

Therefore, we define the *Böhm η-tree* $\mathrm{BT}^\eta(M)$ *of a λ-term M* as $\eta(\mathrm{BT}(M))$.

Examples of Böhm η-trees are: $\mathrm{BT}^\eta(\mathbf{J}) = \mathrm{BT}(\mathbf{J})$, $\mathrm{BT}^\eta(\lambda y.xyy) = \lambda y.xyy$, $\mathrm{BT}^\eta(\lambda xy_1y_2.x(\lambda z_1.y_1(\lambda z_2.z_1(\lambda z_3.z_2z_3))y_2) = \mathrm{BT}^\eta(\mathbf{I}) = \mathbf{I}$, and $\mathrm{BT}^\eta(\lambda y.x\bot y) = x\bot$.

The notions of $\mathrm{BT}^\eta(-)$ and $\mathrm{BT}^e(-)$ are equivalent in the sense that, for all $M, N \in \Lambda$, $\mathrm{BT}^e(M) = \mathrm{BT}^e(N)$ if and only if $\mathrm{BT}^\eta(M) = \mathrm{BT}^\eta(N)$ [23,14]. On the other hand, $\mathrm{BT}^e(M) \subseteq \mathrm{BT}^e(N)$ is not equivalent to $\mathrm{BT}^\eta(M) \leq_\bot \mathrm{BT}^\eta(N)$. E.g. $\mathrm{BT}^e(x\bot) \subseteq \mathrm{BT}^e(\lambda y.xyy)$ but $\mathrm{BT}^\eta(x\bot) = x\bot \not\leq_\bot \lambda y.xyy = \mathrm{BT}^\eta(\lambda y.xyy)$.

5.1 Extensional Taylor Expansion

In order to obtain the analogue of Ehrhard and Regnier's Theorem 2.4 in the extensional setting, the extensional Taylor expansion of M should be the η-normal form of $\mathrm{nf}_\beta\mathcal{T}(M)$, just like $\mathrm{BT}^\eta(M)$ is the η-normal form of $\mathrm{BT}(M)$.

The problem is that defining an η-reduction on $\mathcal{P}(\mathrm{nf}_\beta(\Lambda^r))$ is no easy task. Consider for instance the *naïve* definition $\to_\eta = \cup_{k\geq 0}(\to_{\eta\mathrm{k}})$ where $\lambda x.t[x^k] \to_{\eta\mathrm{k}} t$ if $x \notin \mathrm{fv}(t)$. This correctly reduces $\mathcal{T}(\lambda y.xy) = \{\lambda y.x[y^k] \mid k \geq 0\}$ to $\{x\}$, but the fact that $\lambda y.x1 \to_{\eta_0} x$ is a problem, since $\lambda y.x1$ also belongs to $\mathcal{T}(\lambda y.x\Omega)$, whereas $x \notin \mathcal{T}(\mathrm{nf}_\eta(\lambda y.x\Omega)) = \{\lambda y.x1\}$. Similarly, $\lambda y.x1[y]$ as an element of $\mathcal{T}(\lambda y.xzy)$ is supposed to η-reduce to $x1$, while as an element of $\mathcal{T}(\lambda y.xyy)$ should be η-normal.

These examples reveal that, while the β-reduction of $\mathcal{T}(M)$ can be performed *locally* by reducing each term individually, the η-reduction of $\mathrm{nf}_\beta\mathcal{T}(M)$ must be a *global* operation, that considers the whole set of terms before deciding whether a term should reduce or not. Rather than defining an infinitary rewriting system handling countably many terms, we prefer to divide the problem of computing the η-normal form of $\mathcal{T}(M)$ into two phases:

(i) we first define a labeling $\mathcal{L}(-)$ on the terms $t \in \mathcal{T}(M)$ as a global operation annotating on the empty bags 1 occurring in t:

- whether they "come from" a finite η-expansion of some variable y; for instance $\lambda y.x1 \in \mathcal{T}(\lambda y.x(\lambda z.yz))$ should be labeled as $\lambda y.x1_{\eta(y)}$,
- the set of free variables that were forgotten by taking 1 in the Taylor expansion; for instance $\lambda y.x1[y] \in \mathcal{T}(\lambda y.xyy)$ should be labeled as $\lambda y.x1^y[y]$.

(ii) We then define a local reduction $\to_{\eta^\ell}$ on $\mathcal{L}(\mathrm{nf}_\beta\mathcal{T}(M))$ that exploits this extra-information annotated to perform the η-reduction only when it is safe.

The definition of the labeling $\mathcal{L}$ (Definition 5.1) relies on a certain homogeneity exhibited by the structure of the resource terms in $\mathrm{nf}_\beta\mathcal{T}(M)$. As shown in [4], this homogeneity relies on a *definedness relation* $\preceq$ between normal resource terms:

$$\lambda x_1 \dots x_n.y \preceq \lambda x_1 \dots x_n.y \qquad \frac{t \preceq t' \quad b \preceq b'}{tb \preceq t'b'} \qquad 1 \preceq b \qquad \frac{\exists t' \in b'\ \forall t \in b\,,\ t \preceq t'}{b \preceq b'}$$

The relation $\preceq$ is not a preorder since it is transitive, but not reflexive. For instance, $x[y1[y], y[y]1] \not\preceq x[y1[y], y[y]1]$, since $y1[y] \not\preceq y[y]1$ and $y[y]1 \not\preceq y1[y]$. See the discussion after Definition 9 in [4] for more properties of this relation, and examples. Notice that all singletons $\{\lambda x_1 \dots x_n.y\}$ (for $n \geq 0$) are ideals with respect to $\preceq$.

By Lemma 12 in [4], every ideal $\mathbb{S}$ has one of the following shapes: $\{x\}$, $\lambda x.\mathbb{T}$, $\mathbb{TB}$ for some ideals $\mathbb{T}$ and $\mathbb{B}$. Therefore, the following definition is sound.

Definition 5.1 Let $\mathbb{S} \subseteq \mathrm{nf}_\beta(\Lambda^r)$ be an ideal with respect to $\preceq$ and $t \in \mathbb{S}$. The labeled term $\mathcal{L}(t, \mathbb{S})$ is defined as follows:

$$\mathcal{L}(x, \{x\}) = x, \quad \mathcal{L}(\lambda x.t, \lambda x.\mathbb{T}) = \lambda x.\mathcal{L}(t, \mathbb{T}), \quad \mathcal{L}(tb, \mathbb{T}\mathbb{B}) = \mathcal{L}(t, \mathbb{T})\mathcal{L}(b, \mathbb{B}),$$
$$\mathcal{L}([t_1, \dots, t_k], \mathbb{B}) = [\mathcal{L}(t_1, \bigcup\mathbb{B}), \dots, \mathcal{L}(t_k, \bigcup\mathbb{B})], \text{ for } k > 0$$
$$\mathcal{L}(1, \mathbb{B}) = \begin{cases} 1^x_{\eta(x)} & \text{if there exists } t' \in \bigcup\mathbb{B} \text{ such that } t' \twoheadrightarrow_{\eta'} x, \\ 1^{\mathrm{fv}(\mathbb{B})} & \text{otherwise.} \end{cases} \qquad (\bullet)$$

where $\to_{\eta'}$ is $\lambda x.t[x^{k+1}] \to_{\eta'} t$ when $x \notin \mathrm{fv}(t)$. We set $\mathcal{L}(\mathbb{S}) = \{\mathcal{L}(t, \mathbb{S}) \mid t \in \mathbb{S}\}$. Given a labelled term t, we write $\ulcorner t \urcorner$ for the term obtained by erasing all its labels.

The labeling $\mathcal{L}(-)$ can be always applied to $\mathrm{nf}_\beta\mathcal{T}(M)$ thanks to the following.

Proposition 5.2 *[4, Lemma 23] Let $M \in \Lambda$. Then $\mathrm{nf}_\beta\mathcal{T}(M)$ is an ideal w.r.t. $\preceq$.*

Remark 5.3 The definition of $\mathcal{L}(t, \mathbb{S})$ will be only used when $\mathbb{S}$ is the β-normal of a Taylor expansion. Under this hypothesis, the case $\mathcal{L}(1, \mathbb{B})$ is applied when $\bigcup\mathbb{B} = \mathcal{T}(M)$ for some β-normal $M \in \Lambda$ and Condition $(\bullet)$ becomes "there is $t \in \mathcal{T}(M)$ such that $t \twoheadrightarrow_{\eta'} x$" which holds exactly when $M \twoheadrightarrow_\eta x$.

For example, for $t = \lambda y.x11$ and $\mathbb{S} = \mathrm{nf}_\beta\mathcal{T}(\lambda y.x\Omega y) = \{\lambda y.x1[y^n] \mid n \geq 0\}$ we have $\mathcal{L}(t, \mathbb{S}) = \lambda y.\mathcal{L}(x, \{x\})\mathcal{L}(1, \{1\})\mathcal{L}(1, \{[y^k] \mid k \geq 0\}) = \lambda y.x1^\emptyset 1^y_{\eta(y)}$. While $\mathcal{L}(\lambda y.x11, \mathrm{nf}_\beta\mathcal{T}(\lambda y.xyy)) = \lambda y.x1^y_{\eta(y)}1^y_{\eta(y)}$. Thus $\mathcal{L}(\mathcal{T}(\lambda y.xyy)) = \{\lambda y.x1^y_{\eta(y)}1^y_{\eta(y)}\}$ $\cup\{\lambda y.x1^y_{\eta(y)}[y^{n+1}] \mid n \geq 0\} \cup \{\lambda y.x[y^{k+1}]1^y_{\eta(y)} \mid k \geq 0\} \cup \{\lambda y.x[y^{k+1}][y^{n+1}] \mid n, k \geq 0\}$.

The definition of the set $\widetilde{\mathrm{fv}}(t)$ of *free variables of a labeled term* t is analogous to the one of $\mathrm{fv}(t)$, except for the clauses $\widetilde{\mathrm{fv}}(1^x_{\eta(x)}) = \{x\}$ and $\widetilde{\mathrm{fv}}(1^{\vec{x}}) = \{\vec{x}\}$.

Remark 5.4 Given $T = \mathrm{BT}(M)$, $x \in \mathrm{fv}(T)$ iff $x \in \widetilde{\mathrm{fv}}(t)$ for every $t \in \mathcal{L}(\mathcal{T}(T))$.

Definition 5.5 The reduction $\to_{\eta^\ell}$ on labelled β-normal resource terms, is the contextual closure of $\cup_{n\in\mathbb{N}}(\to_{\eta^\ell_\mathrm{n}})$ where $\to_{\eta^\ell_\mathrm{n}}$ is defined as follows:

$$(\eta^\ell_0)\ \lambda x.t1^x_{\eta(x)} \to_{\eta^\ell_0} t, \text{ if } x \notin \widetilde{\mathrm{fv}}(t), \qquad (\eta^\ell_{n+1})\ \lambda x.t[x^{n+1}] \to_{\eta^\ell_{\mathrm{n+1}}} t, \text{ if } x \notin \widetilde{\mathrm{fv}}(t).$$

For example, we have $\mathcal{L}(\lambda y.x1[y], \mathrm{nf}_\beta\mathcal{T}(\lambda y.xzy)) = \lambda y.x1^z_{\eta(z)}[y] \to_{\eta^\ell} x1^z_{\eta(z)}$, while $\mathcal{L}(\lambda y.x1[y], \mathrm{nf}_\beta\mathcal{T}(\lambda y.xyy)) = \lambda y.x1^y_{\eta(y)}y$, which is already η^ℓ-normal.

Lemma 5.6 *The reduction $\to_{\eta^\ell}$ is SN and confluent.*

Proof The reduction $\to_{\eta^\ell}$ is SN since the size of the term decreases. It is moreover weakly confluent, and therefore confluent by Newman's lemma. □

Definition 5.7 The *extensional Taylor expansion* of a λ-term M is given by:

$$\mathcal{T}^\eta(M) = \ulcorner \mathrm{nf}_{\eta^\ell}\mathcal{L}(\mathrm{nf}_\beta\mathcal{T}(M)) \urcorner$$

In the definition above, β- and η^ℓ-reductions are separated because the reduction $\beta \cup \eta^\ell$ is not confluent: for instance $\lambda x.\mathbf{I}[x, x] \to_{\eta^\ell} \mathbf{I}$ while $\lambda x.\mathbf{I}[x, x] \to_\beta \emptyset$.

5.2 Eta-Reduction on Böhm Approximants

We now provide the technical tools that will be used to prove Theorem 5.15. By Theorem 2.4, it is enough to prove that $\mathcal{T}(\mathrm{BT}^\eta(M))$ is equal to $\ulcorner \mathrm{nf}_{\eta^\ell}\mathcal{L}(\mathcal{T}(\mathrm{BT}(M)))\urcorner$. The difficulty lies in that $\mathrm{BT}^\eta(M)$, which is the η-normal form of $\mathrm{BT}(M)$, is defined coinductively on $\mathrm{BT}(M)$, while the η^ℓ-reduction of $\mathcal{T}(\mathrm{BT}(M))$ works on a set of (labeled) resource terms coming from the finite approximants in $\mathrm{BT}(M)^*$. Therefore, as an intermediate step, we define the η-normal form of the set $\mathrm{BT}(M)^*$ mimicking what we did in Subsection 5.1 for sets of resource terms. In particular, even in this framework the η-reduction must be a global operation; therefore, we introduce a labeling on finite approximants in the spirit of Definition 5.1.

Given $\mathbb{M} \subseteq \mathcal{N}$, $\mathbb{M}\!\downarrow$ denotes its downward closure $\{a \in \mathcal{N} \mid \exists b \in \mathbb{M}, a \leq_\bot b\}$. When $\mathbb{M}$ is an ideal, we have that $\mathbb{M} = \mathbb{M}\!\downarrow$ and all its elements have a similar syntactic structure, except for $\bot$. We adopt for sets $\mathbb{M}$ of approximants the same syntactic sugar we used for $\mathcal{P}(\Lambda^r)$, by extending all the constructors of the grammar of $\mathcal{N}$ as pointwise operations on $\mathcal{P}(\mathcal{N})$. For instance the ideal $\mathrm{BT}(\mathbf{J}x)^*$ can be written as $\{\lambda z_0.x(\mathrm{BT}(\mathbf{J}z_0)^*)\}\!\downarrow = \lambda z_0.x(\mathrm{BT}(\mathbf{J}z_0)^*) \cup \{\bot\}$.

Definition 5.8 Let $\mathbb{M} \subseteq \mathcal{N}$ be an ideal w.r.t. $\leq_\bot$ and $a \in \mathbb{M}$. Define $\mathcal{E}(a, \mathbb{M})$ as:

$$\mathcal{E}(x, \{x\}\!\downarrow) = x, \qquad\qquad \mathcal{E}(\lambda x.a, (\lambda x.\mathbb{M})\!\downarrow) = \lambda x.\mathcal{E}(a, \mathbb{M}\!\downarrow),$$

$$\mathcal{E}(ac, (\mathbb{M}\mathbb{N})\!\downarrow) = \mathcal{E}(a, \mathbb{M}\!\downarrow)\mathcal{E}(c, \mathbb{N}),$$

$$\mathcal{E}(\bot, \mathbb{M}) = \begin{cases} \bot^x_{\eta(x)} & \text{if there exists a } \bot\text{-free } a \in \mathbb{M} \text{ such that } a \twoheadrightarrow_\eta x, \quad (\circ) \\ \bot^{\mathrm{fv}(\mathbb{M})} & \text{otherwise.} \end{cases}$$

We extend the definition to $\mathbb{M}$ by setting $\mathcal{E}(\mathbb{M}) = \{\mathcal{E}(a, \mathbb{M}) \mid a \in \mathbb{M}\}$.

Notice that in the case $(\mathbb{M}\mathbb{N})\!\downarrow$ above, the set $\mathbb{N}$ is already downward closed. As $\mathrm{BT}(M)^*$ is an ideal for every $M \in \Lambda$, we can always compute $\mathcal{L}(\mathrm{BT}(M)^*)$. Condition $(\circ)$ is then equivalent to check that $\mathbb{M} = \mathrm{BT}(M')^*$ for some $M' \twoheadrightarrow_\eta x$.

As we did for resource terms, we speak of *labeled approximants* a, we define the set $\widetilde{\mathrm{fv}}(a)$ by adding the clauses $\widetilde{\mathrm{fv}}(\bot^x_{\eta(x)}) = \{x\}$ and $\widetilde{\mathrm{fv}}(\bot^{\vec{x}}) = \{\vec{x}\}$, and we write $\ulcorner a \urcorner$ for the term obtained from a by erasing all its labels.

Remark 5.9 Given $T = \mathrm{BT}(M)$, $x \in \mathrm{fv}(T)$ iff $x \in \widetilde{\mathrm{fv}}(t)$ for every $t \in \mathcal{E}(T^*)$.

Definition 5.10 The reduction $\to_{\eta^{\mathsf{e}}}$ on labeled approximants is defined as:

$$\lambda x.a\bot^x_{\eta(x)} \to_{\eta^{\mathsf{e}}} a, \text{ if } x \notin \widetilde{\mathrm{fv}}(a), \qquad \lambda x.ax \to_{\eta^{\mathsf{e}}} a, \text{ if } x \notin \widetilde{\mathrm{fv}}(a).$$

It is easy to check that also $\to_{\eta^{\mathsf{e}}}$ is strongly normalizing and confluent.

After a technical lemma, we show that the η^e-reduction on $\mathcal{E}(\mathrm{BT}(M))$ computes exactly the finite approximants of the co-inductively defined tree $\mathrm{BT}^\eta(M)$. Given two sets of terms $\mathbb{X}, \mathbb{Y}$ and a reduction $\to_{\mathtt{r}}$ we write $\mathbb{X} \Rightarrow_{\mathtt{r}} \mathbb{Y}$ if for all $t_1 \in \mathbb{X}$ there is $t_2 \in \mathbb{Y}$ such that $t_1 \twoheadrightarrow_{\mathtt{r}} t_2$ and for all $t_2 \in \mathbb{Y}$ there is $t_1 \in \mathbb{X}$ such that $t_1 \twoheadrightarrow_{\mathtt{r}} t_2$.

Lemma 5.11 *Let* $T = \lambda\vec{x}y.zT_1 \cdots T_{k+1}$ *be a Böhm tree such that* T_{k+1} *is finite,* $T_{k+1} \twoheadrightarrow_\eta y$ *and* $y \notin \mathrm{fv}(zT_1 \cdots T_k)$*. Then* $\mathcal{E}(T^*) \Rightarrow_{\eta^{\mathsf{e}}} \mathcal{E}((\lambda\vec{x}.zT_1^* \cdots T_k^*){\downarrow})$*.*

Proposition 5.12 *For all* $M \in \Lambda$*, we have* $\mathrm{BT}^\eta(M)^* = \ulcorner \mathrm{nf}_{\eta^{\mathsf{e}}}\mathcal{E}(\mathrm{BT}(M)^*)\urcorner$*.*

Proof [Sketch] One proceeds by co-induction on $\mathrm{BT}(M)$ using Lemma 5.11. □

5.3 A Taylor-Based Characterization of Morris's Equivalence

Now that the technical tools for proving the main result of the section are finally in place, we are able to prove that the extensional Taylor expansion of a λ-term M, actually captures the Taylor expansion of $\mathrm{BT}^\eta(M)$.

We first need the following technical results, then we show a sort of commutation between the η^ℓ-normalization and the Taylor expansion.

Lemma 5.13 *Let* $T = \lambda\vec{x}y.zT_1 \cdots T_{k+1}$ *be a Böhm tree such that* T_{k+1} *is finite,* $T_{k+1} \twoheadrightarrow_\eta y$ *and* $y \notin \mathrm{fv}(zT_1 \cdots T_k)$*. Then* $\mathcal{L}(\mathcal{T}(T)) \Rightarrow_{\eta^\ell} \mathcal{L}(\mathcal{T}(\lambda\vec{x}.zT_1 \cdots T_k))$*.*

Proposition 5.14 *For all* $M \in \Lambda$*,* $\mathcal{T}(\ulcorner \mathrm{nf}_{\eta^{\mathsf{e}}}\mathcal{E}(\mathrm{BT}(M)^*)\urcorner) = \ulcorner \mathrm{nf}_{\eta^\ell}\mathcal{L}(\mathcal{T}(\mathrm{BT}(M)))\urcorner$*.*

Proof [Sketch] By coinduction on $\mathrm{BT}(M)$, applying Lemma 5.13. □

We can finally prove the main result of the section.

Theorem 5.15 *For every* λ*-term* M*,* $\mathcal{T}^\eta(M) = \mathcal{T}(\mathrm{BT}^\eta(M))$*.*

Proof Collecting the results above, we have the following chain of equalities:

$$\begin{aligned}
\mathcal{T}^\eta(M) &= \ulcorner \mathrm{nf}_{\eta^\ell}\mathcal{L}(\mathrm{nf}_\beta\mathcal{T}(M))\urcorner && \text{by Definition 5.7}\\
&= \ulcorner \mathrm{nf}_{\eta^\ell}\mathcal{L}(\mathcal{T}(\mathrm{BT}(M)))\urcorner && \text{by Theorem 2.4}\\
&= \mathcal{T}(\ulcorner \mathrm{nf}_{\eta^{\mathsf{e}}}\mathcal{E}(\mathrm{BT}(M)^*)\urcorner) && \text{by Prop. 5.14}\\
&= \mathcal{T}(\mathrm{BT}^\eta(M)^*) && \text{by Prop. 5.12}
\end{aligned}$$

□

Corollary 5.16 *For all* $M, N \in \Lambda$*, we have* $\mathrm{BT}^\eta(M)^* \subseteq \mathrm{BT}^\eta(N)^*$ *if and only if* $\mathcal{T}^\eta(M) \subseteq \mathcal{T}^\eta(N)$*.*

Proof ($\Rightarrow$) Let $t \in \mathcal{T}^\eta(M)$. Then there is $a \in \mathrm{BT}^\eta(M)^*$ such that $t \in \mathcal{T}(a)$. Since $\mathrm{BT}^\eta(M)^* \subseteq \mathrm{BT}^\eta(N)^*$, we have that $a \in \mathrm{BT}^\eta(N)^*$. So $t \in \mathcal{T}(\mathrm{BT}^\eta(N))$ and we get from Theorem 5.15 that $t \in \mathcal{T}^\eta(N)$.

($\Leftarrow$) Let $a \in \mathrm{BT}^\eta(M)^*$. Then by Theorem 5.15 $\mathcal{T}(a) \subseteq \mathcal{T}(\mathrm{BT}^\eta(M)) = \mathcal{T}^\eta(M) \subseteq \mathcal{T}^\eta(N)$. Since $\mathcal{T}^\eta(N) = \mathcal{T}(\mathrm{BT}^\eta(N))$ holds still by Theorem 5.15, we have that $\mathcal{T}(a) \subseteq \mathcal{T}(\mathrm{BT}^\eta(N))$. From Lemma 2.3 we conclude that $a \in \mathrm{BT}^\eta(N)^*$. □

A further corollary is that the notion of extensional Taylor expansion provides an alternative characterization of Morris's equivalence.

Corollary 5.17 *For* $M, N \in \Lambda$*, we have* $M \equiv^{\mathrm{nf}} N$ *if and only if* $\mathcal{T}^\eta(M) = \mathcal{T}^\eta(N)$*.*

Proof We have the following chain of equivalences: By [23] $M \equiv^{\mathrm{nf}} N$ if and only if $\mathrm{BT}^{\eta}(M) = \mathrm{BT}^{\eta}(N)$, that is $\mathrm{BT}^{\eta}(M)^{*} = \mathrm{BT}^{\eta}(N)^{*}$. By Corollary 5.16 this holds if and only if $\mathcal{T}^{\eta}(M) = \mathcal{T}^{\eta}(N)$ does. □

Thanks to Henk Barendregt, Mariangiola Dezani, Thomas Ehrhard, Stefano Guerrini, Michele Pagani for stimulating discussions and the reviewers for the comments.

References

[1] Amadio, R. and P.-L. Curien, "Domains and Lambda Calculi," Cambridge tracts in theoretical computer science, Cambridge University Press, 1998.

[2] Barendregt, H., "The lambda-calculus, its syntax and semantics," Number 103 in Stud. Logic Found. Math., North-Holland, 1984, second edition.

[3] Berline, C., *From computation to foundations via functions and application: The λ-calculus and its webbed models*, Theor. Comput. Sci. **249** (2000), pp. 81–161.

[4] Boudes, P., F. He and M. Pagani, *A characterization of the taylor expansion of lambda-terms*, in: S. Ronchi Della Rocca, editor, *CSL'13*, LIPIcs **23** (2013), pp. 101–115.

[5] Bucciarelli, A., T. Ehrhard and G. Manzonetto, *Not enough points is enough*, in: *CSL'07*, LNCS **4646** (2007), pp. 298–312.

[6] Coppo, M., M. Dezani and M. Zacchi, *Type theories, normal forms and D_{∞}-lambda-models*, Inf. Comput. **72** (1987), pp. 85–116.

[7] Ehrhard, T. and L. Regnier, *The differential lambda-calculus*, Theor. Comput. Sci. **309** (2003), pp. 1–41.

[8] Ehrhard, T. and L. Regnier, *Böhm trees, Krivine's machine and the Taylor expansion of lambda-terms*, in: *CiE*, LNCS **3988**, 2006, pp. 186–197.

[9] Ehrhard, T. and L. Regnier, *Uniformity and the Taylor expansion of ordinary lambda-terms*, Theor. Comput. Sci. **403** (2008), pp. 347–372.

[10] Engeler, E., *Algebras and combinators*, Algebra Universalis **13** (1981), pp. 389–392.

[11] Hyland, J., *A survey of some useful partial order relations on terms of the lambda calculus*, in: C. Böhm, editor, *Lambda-Calculus and Computer Science Theory*, LNCS **37** (1975), pp. 83–95.

[12] Hyland, J., *A syntactic characterization of the equality in some models for the λ-calculus*, J. London Math. Soc. (2) **12(3)** (1975/76), pp. 361–370.

[13] Hyland, M., M. Nagayama, J. Power and G. Rosolini, *A category theoretic formulation for Engeler-style models of the untyped λ-calculus*, Electronic Notes in Theor. Comp. Sci. **161** (2006), pp. 43–57.

[14] Levy, J.-J., *Le lambda calcul - notes du cours* (2005), in French, http://pauillac.inria.fr/~levy/courses/X/M1/lambda/dea-spp/jjl.pdf.

[15] Manzonetto, G., *A general class of models of $\mathcal{H}^{\star}$*, in: *MFCS 2009*, LNCS **5734** (2009), pp. 574–586.

[16] Manzonetto, G., *What is a categorical model of the differential and the resource λ-calculi?*, Mathematical Structures in Computer Science **22** (2012), pp. 451–520.

[17] Morris, J., "Lambda calculus models of programming languages," Ph.D. thesis, MIT (1968).

[18] Nakajima, R., *Infinite normal forms for the lambda calculus*, in: *Lambda-Calculus and Computer Science Theory*, Lecture Notes in Computer Science **37** (1975), pp. 62–82.

[19] Paolini, L., M. Piccolo and S. Ronchi Della Rocca, *Logical relational λ-models* (2014), draft available at http://www.di.unito.it/~paolini/papers/logicalRelational.pdf.

[20] Ronchi Della Rocca, S. and L. Paolini, "The Parametric λ-Calculus: a Metamodel for Computation," Texts in TCS: An EATCS Series, Springer-Verlag, Berlin, 2004.

[21] Scott, D., *Continuous lattices*, in: Lawvere, editor, *Toposes, Algebraic Geometry and Logic*, Lecture Notes in Math. **274** (1972), pp. 97–136.

[22] Tranquilli, P., *Intuitionistic differential nets and λ-calculus*, Th. Comp. Sci. **412** (2011), pp. 1979–1997.

[23] van Bakel, S., F. Barbanera, M. Dezani-Ciancaglini and F.-J. de Vries, *Intersection types for lambda-trees*, Theor. Comput. Sci. **272** (2002), pp. 3–40.

A Technical Appendix

This technical appendix is devoted to provide some proofs that were omitted or just sketched in the article.

A.1 Omitted proofs of Section 2

2.3 *Let $a \in \mathcal{N}$ and $M \in \Lambda$, then $\mathcal{T}(a) \subseteq \mathcal{T}(\mathrm{BT}(M))$ entails $a \in \mathrm{BT}(M)^*$.*

Proof By structural induction on a.

Case $a = \bot$ and $\mathcal{T}(a) = \emptyset \subseteq \mathcal{T}(\mathrm{BT}(M))$. Then it is trivial since $\bot \in \mathrm{BT}(M)$.

Case $a = \lambda\vec{x}.ya_1 \cdots a_k$ and $\mathcal{T}(a) = \bigcup_{n_1,\dots,n_k \geq 0} \lambda\vec{x}.y[\mathcal{T}(a_1)^{n_1}] \cdots [\mathcal{T}(a_k)^{n_k}] \subseteq \mathcal{T}(\mathrm{BT}(M))$. Then $M \twoheadrightarrow_\beta \lambda\vec{x}.yN_1 \cdots N_k$ for some $N_1, \dots, N_k \in \Lambda$ such that $\mathcal{T}(a_i) \subseteq \mathcal{T}(\mathrm{BT}(N_i))$. By induction hypothesis $a_i \in \mathrm{BT}(N_i)^*$ for all $1 \leq i \leq k$ and we conclude that $\lambda\vec{x}.ya_1 \cdots a_k \in \mathrm{BT}(M)^*$. □

A.2 Omitted proofs of Section 3

3.10 (Approximation Theorem) *Let M be a λ-term. Then $\Gamma \vdash M : \sigma$ if and only if there exists $t \in \mathcal{T}(M)$ such that $\Gamma \vdash t : \sigma$.*

Proof ($\Rightarrow$) The proof is by induction on a derivation of $\Gamma \vdash M : \sigma$. We proceed by case analysis on the last rule applied in the derivation.

Case `var`**.** We have $x : \sigma \vdash x : \sigma$ using the rule (`var`). This case is trivial since $\mathcal{T}(x) = \{x\}$.

Case `lam`**.** We have $\Gamma \vdash \lambda x.N : \sigma$ using the rule (`lam`). By Theorem 3.7(ii), we have that $\Gamma, x : \mu \vdash N : \tau$ for some $\mu \to \tau \simeq \sigma$. By IH, there exists $t' \in \mathcal{T}(N)$ such that $\Gamma, x : \mu \vdash t' : \tau$. Therefore, $\lambda x.t' \in \mathcal{T}(\lambda x.N)$ and

$$\dfrac{\dfrac{\Gamma, x : \mu \vdash t' : \tau}{\Gamma \vdash \lambda x.t' : \mu \to \tau}\,(\mathtt{lam}) \qquad \mu \to \tau \simeq \sigma}{\Gamma \vdash \lambda x.t' : \sigma}\,(\mathtt{eq})$$

Case `app`**.** We have $\Gamma \vdash NP : \sigma$ using the rule (`app`). By Theorem 3.7(iii), there is a decomposition $\Gamma = \Gamma_0 \wedge (\wedge_{i=1}^n \Gamma_i)$ for some $n \geq 0$, such that $\Gamma_0 \vdash N : \wedge_{i=1}^n \sigma_i \to \sigma$ and $\Gamma_i \vdash P : \sigma_i$. By IH, there exists $s \in \mathcal{T}(N)$ such that $\Gamma_0 \vdash s : \wedge_{i=1}^n \sigma_i \to \sigma$, and there exist $t_1, \dots, t_n \in \mathcal{T}(P)$ such that $\Gamma_i \vdash t_i : \sigma_i$.

Therefore we have that $s[t_1, \dots, t_n] \in \mathcal{T}(NP)$ and:

$$\dfrac{\Gamma_0 \vdash s : \wedge_{i=1}^n \sigma_i \to \sigma \qquad \Gamma_i \vdash t_i : \sigma_i \quad \forall i \in \{1, \dots, n\}}{\Gamma \vdash s[t_1, \dots, t_n] : \sigma}$$

Case `eq`**.** Let $\Gamma \vdash M : \sigma$ using the rule (`eq`). Then $\Gamma \vdash M : \tau$ for some $\tau \simeq \sigma$. By IH there exists $t \in \mathcal{T}(M)$ such that $\Gamma \vdash t : \tau$. By applying (`eq`) we derive $\Gamma \vdash t : \sigma$.

This concludes the left-to-right implication.

($\Leftarrow$) Let $t \in \mathcal{T}(M)$ such that $\Gamma \vdash t : \sigma$. We proceed by induction on the derivation for such a type assignment.

Case var. We have $x : \sigma \vdash x : \sigma$ and $x \in \mathcal{T}(M)$ which entails $M = x$ by definition of the Taylor expansion. This case is therefore trivial.

Case app. We have $s[t_1, \dots, t_n] \in \mathcal{T}(M)$ such that $\Gamma \vdash s[t_1, \dots, t_n] : \sigma$. By Theorem 3.7(iii)', we get the decomposition $\Gamma = \Gamma_0 \wedge (\wedge_{i=1}^n \Gamma_i)$ and the typing assignments $\Gamma_0 \vdash t : \wedge_{i=1}^n \sigma_i \to \sigma$ and $\Gamma_i \vdash s_i : \sigma_i$. By definition of Taylor expansion, if $s[t_1, \dots, t_n] \in \mathcal{T}(M)$ then $M = NP$ for some $N, P \in \Lambda$ such that $s \in \mathcal{T}(N)$ and $t_1, \dots, t_n \in \mathcal{T}(P)$. By IH, $\Gamma_0 \vdash N : \wedge_{i=1}^n \sigma_i \to \sigma$ and $\Gamma_i \vdash P : \sigma_i$ for all $i \in \{1, \dots, n\}$. Therefore we derive:

$$\dfrac{\Gamma_0 \vdash N : \wedge_{i=1}^n \sigma_i \to \sigma \quad \Gamma_i \vdash P : \sigma_i \quad \forall i \in \{1, \dots, n\}}{\Gamma \vdash NP : \sigma}\ (\mathtt{app})$$

Case lam. We have $\lambda x.t \in \mathcal{T}(M)$ such that $\Gamma \vdash \lambda x.t : \sigma$. By definition of Taylor expansion, $\lambda x.t \in \mathcal{T}(M)$ entails $M = \lambda x.N$ for some $N \in \Lambda$ such that $t \in \mathcal{T}(N)$. By Theorem 3.7(ii), one gets $\Gamma, x : \mu \vdash t : \tau$ for some $\mu \to \tau \simeq \sigma$. By IH, we have $\Gamma, x : \mu \vdash N : \tau$. Therefore, we can derive

$$\dfrac{\dfrac{\Gamma, x : \mu \vdash N : \tau}{\Gamma \vdash \lambda x.N : \mu \to \tau}\ (\mathtt{lam}) \qquad \mu \to \tau \simeq \sigma}{\Gamma \vdash \lambda x.N : \sigma}\ (\mathtt{eq})$$

Case eq. Let $t \in \mathcal{T}(M)$ and suppose $\Gamma \vdash t : \sigma$ comes from $\Gamma \vdash t : \tau$ by (eq). By IH, we have $\Gamma \vdash M : \tau$. By applying (eq) we derive $\Gamma \vdash N : \sigma$. □

A.3 Omitted proofs of Section 4

We recall the definition of "ω occurs positively/negatively in a type σ".

Definition A.1 The relations $\omega \in^p \sigma$ for $p \in \{+, -\}$ are defined as follows:

(i) $\omega \in^- \mu \to \sigma$ for any type σ and intersection μ such that $\mu = \omega$;

(ii) if $\omega \in^p \sigma$ then $\omega \in^p \mu \to \sigma$ for any intersection μ;

(iii) if $\omega \in^p \sigma$ then $\omega \in^{\neg p} \sigma \wedge \mu \to \tau$ for any types σ, τ and intersection μ.

Remark that the condition $\mu = \omega$ in (i) is non-trivial, since equality $=$ between types includes the neutrality of ω. For instance $\omega \in^- \omega \wedge \omega \to \sigma$ as $\omega \wedge \omega = \omega$.

4.2 *Let $\mathcal{A}$ be a partial pair such that, for all $m \in \mathcal{M}_{\mathrm{f}}(A)$ and $\alpha \in A$, $(m, \alpha) \in$* dom(j) *entails that $m \neq []$. Then $\overline{\mathcal{A}}$ preserves ω-polarities.*

Proof We perform an induction loading and prove that, for all type $\sigma, \tau \in \mathsf{T}_{\overline{\mathcal{A}}}$ and $p \in \{+, -\}$: if $\omega \in^p \sigma$ and $\tau \simeq^{\overline{\mathcal{A}}} \sigma$ then $\tau^\bullet \notin A$ and $\omega \in^p \tau$. In the rest of the proof we will just write $\simeq$ for $\simeq^{\overline{\mathcal{A}}}$.

We proceed by induction on the definition of $\omega \in^p \sigma$.

Case (i). Suppose that $\omega \in^p \sigma$ because $p = -$ and $\sigma = \omega \to \gamma$, then we need to prove that $\omega \in^- \tau$, for any τ such that $\tau \simeq \sigma$, that is such that $\tau^\bullet = \sigma^\bullet$. By definition, we have:

$$\sigma^\bullet = (\omega \to \gamma)^\bullet = \overline{j}([], \gamma^\bullet) = ([], \gamma^\bullet)$$

where the last equality follows from Definition 3.2 and the hypothesis that $([], \gamma^\bullet) \notin \mathrm{dom}(j)$. From $\tau^\bullet = ([], \gamma^\bullet)$ we get that $\tau^\bullet \notin A$ since A does not contain any pair, and this entails that also τ cannot be atomic.

Suppose therefore $\tau = \mu \to \delta$, then we have $\tau^\bullet = (\mu \to \delta)^\bullet = \overline{j}(\mu^\bullet, \delta^\bullet) = \overline{j}([], \gamma^\bullet) = \sigma^\bullet$. From the injectivity of $\overline{j}$, we get that $\mu^\bullet = []$ and $\delta^\bullet = \gamma^\bullet$, so $\tau = \omega \to \delta$ and $\omega \in^- \tau$.

Case (ii). Suppose that $\omega \in^p \sigma$ because $\sigma = \mu \to \gamma$ and $\omega \in^p \gamma$. Then

$$\sigma^\bullet = (\mu \to \gamma)^\bullet = \overline{j}(\mu^\bullet, \gamma^\bullet) = \tau^\bullet.$$

From $\omega \in^p \gamma$, $\gamma \simeq \gamma$ and the induction hypothesis, we get that $\gamma^\bullet \notin A$ and therefore $(\mu^\bullet, \gamma^\bullet) \notin \mathrm{dom}(j)$. By Definition 3.2, we have that $\overline{j}(\mu^\bullet, \gamma^\bullet) = (\mu^\bullet, \gamma^\bullet)$, and since this is equal to $\tau^\bullet$, we get $\tau = \nu \to \delta$ for some ν, δ. From $\overline{j}(\mu^\bullet, \gamma^\bullet) = \overline{j}(\nu^\bullet, \delta^\bullet)$ and the injectivity of $\overline{j}$ we get that $\mu^\bullet = \nu^\bullet$ and $\gamma^\bullet = \delta^\bullet$.

From $\omega \in^p \gamma$ and $\delta \simeq \gamma$ we get, by induction hypothesis, that $\omega \in^p \delta$ and therefore $\omega \in^p \nu \to \delta = \tau$.

Case (iii). Suppose that $\omega \in^p \sigma$ because $\sigma = \gamma_1 \wedge \mu \to \gamma_2$ and $\omega \in^{\neg p} \gamma_1$. From $\gamma_1 \wedge \mu \to \gamma_2 \simeq \tau$, we get

$$(\gamma_1 \wedge \mu \to \gamma_2)^\bullet = \overline{j}([\gamma_1^\bullet] + \mu^\bullet, \gamma_2^\bullet) = \tau^\bullet.$$

Suppose, by the way of contradiction, that τ is an atomic type α. Then, we have $\overline{j}([\gamma_1^\bullet] + \mu^\bullet, \gamma_2^\bullet) = \alpha$ which implies, by Definition 3.2, that $([\gamma_1^\bullet] + \mu^\bullet, \gamma_2^\bullet) \in \mathrm{dom}(j) \subseteq \mathcal{M}_\mathrm{f}(A) \times A$. In particular, we get $\gamma_1^\bullet \in A$, which is impossible since $\omega \in^{\neg p} \gamma_1$ and $\gamma_1 \simeq \gamma_1$, so by the induction hypothesis we conclude that $\gamma_1^\bullet$ is not atomic.

So, $\tau = \nu \to \delta_2$, and $(\nu \to \delta_2)^\bullet = \overline{j}(\nu^\bullet, \delta_2^\bullet) = \overline{j}([\gamma_1^\bullet] + \mu^\bullet, \gamma_2^\bullet)$. Since $\overline{j}$ is injective, $\nu^\bullet = [\gamma_1^\bullet] + \mu^\bullet$ and $\delta_2^\bullet = \gamma_2^\bullet$. Therefore, $\nu = \delta_1 \wedge \nu'$ such that $\delta_1^\bullet = \gamma_1^\bullet$ and $\nu'^\bullet = \mu^\bullet$. Since $\omega \in^{\neg p} \gamma_1$ and $\gamma_1 \simeq \delta_1$, by IH we get $\omega \in^{\neg p} \gamma_1$ and we conclude that $\omega \in^p \tau$. □

For convenience, we present Lemma 4.3 with an additional equivalent sentence (iii-bis), which is an intermediate step between (iii) and (iv).

4.3 *Let $M \in \Lambda$. The following are equivalent:*

(i) *M has a normal form,*

(ii) *there is $a \in \mathrm{BT}(M)^*$ that does not contain $\bot$,*

(iii) *there is $t \in \mathrm{nf}_\beta(\mathcal{T}(M))$ that does not contain the empty bag 1,*

(iii-bis) *in every rgm $\mathcal{D}$ preserving ω-polarities, $\Gamma \vdash^{\mathcal{D}} t : \sigma$ for some $t \in \mathrm{nf}_\beta\mathcal{T}(M)$, environment Γ and type σ such that $\omega \notin^+ \sigma$ and $\omega \notin^- \Gamma$, that is $\omega \notin^- \Gamma(x)$ for all $x \in \mathrm{Var}$.*

(iv) *in every rgm $\mathcal{D}$ preserving ω-polarities, $\Gamma \vdash^{\mathcal{D}} M : \sigma$ for some environment Γ and type σ such that $\omega \notin^+ \sigma$ and $\omega \notin^- \Gamma$, that is $\omega \notin^- \Gamma(x)$ for all $x \in \mathrm{Var}$.*

Proof ($i \iff ii$) is trivial.

(*ii* $\iff$ *iii*) follows from Theorem 2.4.

(iii $\Rightarrow$ *iii-bis)* We prove that this implication holds more generally for any β-normal form t that does not contain 1 (regardless the fact that t belongs to a Taylor expansion). We proceed by structural induction on t.

Case $t = \lambda x.t'$ where t' is β-normal. By induction hypothesis, $\Gamma' \vdash t' : \tau$ holds for some context Γ' and type τ such that $\omega \notin^- \Gamma'$ and $\omega \notin^+ \tau$. Note that Γ' can be written as $\Gamma, x : \mu$ for some Γ and μ, therefore we can derive:

$$\frac{\Gamma, x : \mu \vdash t' : \tau}{\Gamma \vdash \lambda x.t' : \mu \to \tau}\ (\texttt{lam})$$

From $\omega \notin^- \Gamma'$ we get that $\omega \notin^- \Gamma$ and $\omega \notin^- \mu$, which entails $\omega \notin^+ \mu \to \tau$.

Case $t = yb_1 \cdots b_k$, for some $k \geq 0$, and each $b_i = [s_{i,1}, \dots, s_{i,n_i}]$ (for $n_i \geq 0$) only contains β-normal terms. By induction hypothesis, there are environments Γ_{ij}, and types τ_{ij}, such that $\omega \notin^- \Gamma_{ij}$ and $\omega \notin^+ \tau_{ij}$ and $\Gamma_{ij} \vdash s_{ij} : \tau_{ij}$ holds. Then we can derive:

$$\frac{\Gamma_0 \vdash y : \mu_1 \to \cdots \to \mu_k \to \alpha \quad \Gamma_{ij} \vdash s_{ij} : \tau_{ij} \quad i \in \{1, \dots, k\},\ j \in \{1, \dots, n_i\}}{\Gamma \vdash yb_1 \cdots b_k : \alpha}$$

where $\mu_i = \wedge_{j=1}^{n_i} \tau_{ij}$, $\Gamma_0 = y : \mu_1 \to \cdots \to \mu_k \to \alpha$ and $\Gamma = \Gamma_0 \wedge (\wedge_{i=1}^{k} \wedge_{j=1}^{n_i} \Gamma_{ij})$. As $\omega \notin^+ \tau_{ij}$ we also have $\omega \notin^- \mu_i$ and therefore $\omega \notin^- \Gamma_0$. From this, and the hypotheses that $\omega \notin^- \Gamma_{ij}$ we get that $\omega \notin^- \Gamma$. Of course $\omega \notin^- \alpha$ because α is an atom.

(iii-bis $\Rightarrow$ *iii)* Consider $t \in \mathrm{nf}_\beta \mathcal{T}(M)$ such that $\Gamma \vdash t : \sigma$ where Γ and σ satisfy the hypotheses of (iii-bis). We proceed by induction on the structure of the β-normal t.

Case $t = \lambda x.t'$ where t' is β-normal. By applying Theorem 3.7(ii) we have that $\Gamma, x : \mu \vdash t' : \tau$ holds for $\mu \in \mathsf{I}_\mathcal{D}$ and $\tau \in \mathsf{T}_\mathcal{D}$ such that $\sigma \simeq \mu \to \tau$. Since $\mathcal{D}$ preserves ω-polarities, $\omega \notin^+ \sigma$ entails $\omega \notin^+ \mu \to \tau$. As neither Γ nor μ has negative occurrences of ω, we have $\omega \notin^- (\Gamma, x : \mu)$ and $\omega \notin^+ \tau$, so, by the induction hypothesis, we get that t' does not have occurrences of 1. Therefore 1 does not occur in $\lambda x.t'$ either.

Case $t = yb_1 \cdots b_k$, for some $k \geq 0$, and each $b_i = [s_{i,1}, \dots, s_{i,n_i}]$ (for $n_i \geq 0$) only contains β-normal terms. If $k = 0$ we are done, as y does not contain 1. Consider then the case $k > 0$. By iterating Theorem 3.7(iii') we know that there is a decomposition $\Gamma = \Gamma_0 \wedge (\wedge_{i=1}^{k} \wedge_{j=1}^{n_i} \Gamma_{ij})$ such that (setting $\mu_i = \wedge_{j=1}^{n_i} \tau_{ij}$):

$$\frac{\Gamma_0 \vdash y : \mu_1 \to \cdots \to \mu_k \to \sigma \quad \Gamma_{ij} \vdash s_{ij} : \tau_{ij} \quad \text{for } i = 1, \dots, k \quad j = 1, \dots, n_i}{\Gamma \vdash yb_1 \cdots b_k : \sigma}$$

By Theorem 3.7(i), we get that $\Gamma_0 = y : \tau$ for some $\tau \simeq \mu_1 \to \cdots \to \mu_k \to \sigma$. From this, it follows that $\Gamma(y) = \tau \wedge \mu$ for some μ, so $\omega \notin^- \Gamma$ entails that $\omega \notin^- \tau$ and, as $\mathcal{D}$ preserves ω-polarities, we get that $\omega \notin^- \mu_1 \to \cdots \to \mu_k \to \sigma$. From this, on the one side we get that each μ_i is different from ω (that is, $n_i > 0$, so $b_i \neq 1$) and on the other side that $\omega \notin^+ \tau_{ij}$ holds for $1 \leq i \leq k$ and $1 \leq j \leq n_i$. We can therefore apply the induction hypothesis to each derivation $\Gamma_{ij} \vdash s_{ij} : \tau_{ij}$ and conclude that the terms s_{ij} do not contain 1, so neither does the term $yb_1 \cdots b_k$.

(*iii-bis* $\iff$ *iv*) Let us suppose (iv). By Theorem 3.10, we have that $\Gamma \vdash M : \sigma$ holds if and only if there exists $s \in \mathcal{T}(M)$ such that $\Gamma \vdash s : \sigma$. By Theorem 2.1 (strong normalization of Λ^r) and Theorem 3.8(i) (subject reduction), the latter is equivalent to the existence of $t \in \mathrm{nf}_\beta \mathcal{T}(M)$ such that $\Gamma \vdash t : \sigma$. Therefore, (iv) is equivalent to (iii-bis). □

For proving Lemma 4.4, we need the following remark and technical lemma.

Remark A.2 In the model $\mathcal{D}_\star$, we have that $\sigma \simeq \star$ holds if and only if σ is generated by the following grammar:

$$\gamma ::= \star \mid \gamma \to \gamma$$

In particular, $\mu \to \sigma \simeq \star$ entails that $\mu = \tau$ for some $\tau \simeq \star$ and $\sigma \simeq \star$.

Lemma A.3 *Let $N \in \Lambda$ be a β-normal form. If $\Gamma \vdash N : \sigma$, for some Γ and σ such that $\Gamma(x) \simeq \star$ for all $x \in \mathrm{dom}(\Gamma)$ and $\sigma \simeq \star$, then N is linear and* $\mathrm{dom}(\Gamma) = \mathrm{fv}(N)$.

Proof We proceed by structural induction on N.

Case $N = \lambda x.N'$ where N' is β-normal. From $\Gamma \vdash \lambda x.N' : \sigma$ we get, by Theorem 3.7(ii), that $\Gamma, x : \mu \vdash N' : \tau$ for some μ, τ such that $\mu \to \tau \simeq \sigma$ and, by transitivity of $\simeq$, we get that $\mu \to \tau \simeq \star$ holds. By Remark A.2 this entails $\mu = \gamma$ for some $\gamma \simeq \star$ and $\tau \simeq \star$, therefore we can apply the induction hypothesis and get that N' is linear and $\mathrm{dom}(\Gamma, x : \gamma) = \mathrm{fv}(N')$. Thus, $\lambda x.N'$ is also linear and $\mathrm{dom}(\Gamma) = \mathrm{fv}(N') - \{x\} = \mathrm{fv}(\lambda x.N')$ which is what we are meant to prove.

Case $N = yN_1 \cdots N_k$ such that $N_1, \dots, N_k$ are β-normal. By Theorem 3.7(iii) and there is a decomposition $\Gamma = \Gamma_0 \wedge (\wedge_{i=1}^k \wedge_{j=1}^{n_i} \Gamma_{ij})$ such that $\Gamma_0 \vdash y : \mu_1 \to \cdots \to \mu_k \to \sigma$ holds for some $\mu_i = \tau_{i1} \wedge \cdots \wedge \tau_{in_i}$ and $\Gamma_{ij} \vdash N_i : \tau_{ij}$ is derivable for all $1 \leq i \leq k$ and $1 \leq j \leq n_i$. By Theorem 3.7(i), $\Gamma_0 = y : \gamma$, for a type $\gamma \simeq \mu_1 \to \cdots \to \mu_k \to \sigma$. As $\Gamma_0(y) = \Gamma(y) = \gamma \simeq \star$ we also have by transitivity of $\simeq$ that $\mu_1 \to \cdots \to \mu_k \to \sigma \simeq \star$ which entails by Remark A.2 that $\mu_i = \tau_i$ (i.e. $n_i = 1$) and $\tau_i \simeq \star$. Therefore we have $\Gamma = \Gamma_0 \wedge (\wedge_{i=1}^k \Gamma_i)$ and $\Gamma_i \vdash N_i : \tau_i$ for some Γ_i such that $\Gamma_i(x) \simeq \star$ for all $x \in \mathrm{dom}(\Gamma_i)$ and $\tau_i \simeq \star$.

By the induction hypothesis we get that each N_i is linear and $\mathrm{dom}(\Gamma_i) = \mathrm{fv}(N_i)$. We conclude that $yN_1 \cdots N_k$ is linear and $\mathrm{dom}(\Gamma) = \mathrm{dom}(\Gamma_0) \cup (\bigcup_{i=1}^k \mathrm{dom}(\Gamma_i)) = \mathrm{fv}(yN_1 \cdots N_k)$. □

4.4 *Let $M \in \Lambda$ and $\Gamma = x_1 : \star, \dots, x_n : \star$. Then $\Gamma \vdash^{\mathcal{D}_\star} M : \star$ if and only if M has a linear β-normal form and* $\mathrm{dom}(\Gamma) = \mathrm{fv}(\mathrm{nf}_\beta(M))$.

Proof ($\Rightarrow$) By Theorem 4.2, the rgm $\mathcal{D}_\star$ preserves ω-polarities. As ω does not occur positively nor negatively in $\star$, we can deduce by Lemma 4.3 that M has a β-normal form. By subject reduction, we derive $\Gamma \vdash \mathrm{nf}_\beta(M) : \star$ and, by Lemma A.3, we conclude that $\mathrm{nf}_\beta(M)$ is linear.

($\Leftarrow$) Suppose that $M \in \Lambda$ has a linear β-normal form and that the environment $\Gamma = x_1 : \star, \dots, x_n : \star$ is such that $\mathrm{dom}(\Gamma) = \mathrm{fv}(\mathrm{nf}_\beta(M))$. It is enough to prove that $\Gamma \vdash \mathrm{nf}_\beta(M) : \star$ is derivable, then one concludes by subject expansion (Theorem 3.8(i)) that $\Gamma \vdash M : \star$ holds. We proceed by induction on $\mathrm{nf}_\beta(M)$.

Case $\mathrm{nf}_\beta(M) = \lambda x.N'$ where N' is β-normal. Obviously, N' is linear and $\mathrm{dom}(\Gamma, x : \star) = \mathrm{fv}(N')$, so by the induction hypothesis

$$\dfrac{\dfrac{\Gamma, x : \star \vdash N' : \star}{\Gamma \vdash \lambda x.N' : \star \to \star}\,(\texttt{lam}) \qquad \star \simeq \star \to \star}{\Gamma \vdash \lambda x.N' : \star}\,(\texttt{eq})$$

is also derivable.

Case $\mathrm{nf}_\beta(M) = yN_1 \cdots N_k$ such that $N_1, \dots, N_k$ are β-normal. We let Γ_i to be the environment such that $\Gamma_i(x) = \star$ if $x \in \mathrm{fv}(N_i)$ and $\Gamma_i(x) = \omega$ otherwise. As the N_i's are linear, we derive $\Gamma_i \vdash N_i : \star$ by the induction hypothesis. Then we can derive (for $\Gamma_0 = y : \star$):

$$\dfrac{\dfrac{\dfrac{}{\Gamma_0 \vdash y : \star}\,(\texttt{var}) \qquad \star \simeq \star \to \cdots \to \star \to \star}{\Gamma_0 \vdash y : \star \to \cdots \to \star \to \star}\,(\texttt{eq}) \qquad \Gamma_i \vdash N_i : \star \quad 1 \leq i \leq k}{\Gamma_0 \wedge (\wedge_{i=1}^{k} \Gamma_i) \vdash yN_1 \cdots N_k : \star}\,(\texttt{lam})$$

To conclude, it is enough to check that $\Gamma = \Gamma_0 \wedge (\wedge_{i=1}^{k} \Gamma_i)$. □

A.4 Omitted proofs of Section 5

Lemma A.4 *Let $M \in \Lambda$ be a β-normal form such that $M \twoheadrightarrow_\eta x$. For all $a \in M^*$, we have that either $\mathcal{E}(a, M^*) = \bot^x_{\eta(x)}$ or $\mathcal{E}(a, M^*) \twoheadrightarrow_{\eta^\mathsf{e}} x$.*

Proof Since M is β-normal, it has the shape $\lambda x_1 \dots x_n.xN_1 \cdots N_k$. As $M \twoheadrightarrow_\eta x$ we get that $n = m$, $x \neq x_i$ and $N_i \twoheadrightarrow_\beta x_i$ for all $i \in \{1, \dots, n\}$.

We proceed by induction on a.

Case $a = \bot$. Then $\mathcal{E}(a, M^*) = \bot^x_{\eta(x)}$ by Definition 5.8.

Case $a = \lambda x_1 \dots x_n.xa_1 \cdots a_n$ with $a_i \in N_i^*$ for all $i \in \{1, \dots, n\}$. By induction hypothesis, either $\mathcal{E}(a_i, N_i^*) \twoheadrightarrow_{\eta^\mathsf{e}} x_i$ or $\mathcal{E}(a_i, N_i^*) \twoheadrightarrow_{\eta^\mathsf{e}} \bot^{x_i}_{\eta(x_i)}$ so $\mathcal{E}(a, M^*) = \lambda x_1 \dots x_n.x\mathcal{E}(a_1, N_1^*) \cdots \mathcal{E}(a_n, N_n^*) \twoheadrightarrow_{\eta^\mathsf{e}} x$. □

5.11 *Let $T = \lambda \vec{x}y.zT_1 \cdots T_{k+1}$ be a Böhm tree such that T_{k+1} is finite, $T_{k+1} \twoheadrightarrow_\eta y$ and $y \notin \mathrm{fv}(zT_1 \cdots T_k)$. Then $\mathcal{E}(T^*) \Rightarrow_{\eta^\mathsf{e}} \mathcal{E}((\lambda \vec{x}.zT_1^* \cdots T_k^*)\downarrow)$.*

Proof We first prove that, given $a \in T^*$, there exists $a' \in (\lambda \vec{x}.zT_1^* \cdots T_k^*)\downarrow$ such that $\mathcal{E}(a, T^*) \twoheadrightarrow_{\eta^\mathsf{e}} \mathcal{E}(a', (\lambda \vec{x}.zT_1^* \cdots T_k^*)\downarrow)$. We split into cases depending on a.

Case $a = \bot$. Then $\mathcal{E}(a, T^*) = \mathcal{E}(\bot, T^*) = \mathcal{E}(\bot, (\lambda \vec{x}y.zT_1^* \cdots T_k^* T_{k+1}^*)\downarrow)$. From the fact that T_{k+1} is finite, we get that $T_{k+1} \in \mathcal{N}$ and since $T_{k+1} \twoheadrightarrow_\eta y$ we have that T_{k+1} is $\bot$-free. As $y \notin \mathrm{fv}(zT_1 \cdots T_k)$, there is a $\bot$-free $c_1 \in T^*$ such that $c_1 \twoheadrightarrow_\eta z$ if and only if there exists a $\bot$-free $c_2 \in (\lambda \vec{x}.zT_1^* \cdots T_k^*)\downarrow$ such that $c_2 \twoheadrightarrow_\eta z$. Therefore $\mathcal{E}(\bot, (\lambda \vec{x}y.zT_1^* \cdots T_k^* T_{k+1}^*)\downarrow) = \mathcal{E}(\bot, (\lambda \vec{x}.zT_1^* \cdots T_k^*)\downarrow)$, so $a' = \bot$.

Case $a = \lambda \vec{x}y.za_1 \cdots a_{k+1}$, with $a_i \in T_i^*$ for $1 \leq i \leq k+1$. By definition, we have $\mathcal{E}(a, T^*) = \lambda \vec{x}y.z\mathcal{E}(a_1, T_1^*) \cdots \mathcal{E}(a_k, T_k^*)\mathcal{E}(a_{k+1}, T_{k+1}^*)$. By hypothesis, T_{k+1} is actually a λ-term (i.e., finite and $\bot$-free) such that $T_{k+1} \twoheadrightarrow_\eta y$ so, by Lemma 5.11, either $\mathcal{E}(a_{k+1}, T_{k+1}) \twoheadrightarrow_{\eta^{\mathsf{e}}} \bot^{y}_{\eta(y)}$ or $\mathcal{E}(a_{k+1}, T_{k+1}) \twoheadrightarrow_{\eta^{\mathsf{e}}} y$. By Remark 5.9 $y \notin \mathrm{fv}(zT_1 \cdots T_k)$ entails $y \notin \widetilde{\mathrm{fv}}(z\mathcal{E}(a_1, T_1^*) \cdots \mathcal{E}(a_k, T_k^*))$, hence in both cases we get $\mathcal{E}(a, T^*) \twoheadrightarrow_{\eta^{\mathsf{e}}} \lambda\vec{x}.z\mathcal{E}(a_1, T_1^*) \cdots \mathcal{E}(a_k, T_k^*) \in \mathcal{E}((\lambda\vec{x}.zT_1^* \cdots T_k^*)\!\downarrow)$. Therefore the a' we were looking for is just $\lambda\vec{x}.za_1 \cdots a_k$.

Second, we prove that for every $a' \in (\lambda\vec{x}.zT_1^* \cdots T_k^*)\!\downarrow$ there is $a \in T^*$ such that $\mathcal{E}(a, T^*) \twoheadrightarrow_{\eta^{\mathsf{e}}} \mathcal{E}(a', (\lambda\vec{x}.zT_1^* \cdots T_k^*)\!\downarrow)$. Again, we split into cases depending on a'.

Case $a' = \bot$. It is enough to take $a' = \bot$ and reason as above.

Case $a' = \lambda\vec{x}.za'_1 \cdots a'_k$ with $a'_i \in T_i^*$ for all $1 \leq i \leq k$. Clearly, $\bot \in T_{k+1}^*$ and $\mathcal{E}(\bot, T_{k+1}^*) = \bot^{y}_{\eta(y)}$, since by hypothesis T_{k+1} is finite and $T_{k+1} \twoheadrightarrow_\eta y$. Therefore, for $a = \lambda\vec{x}y.za'_1 \cdots a'_k\bot \in T^*$ we have

$$\begin{array}{rcll}
\mathcal{E}(a, T^*) & = & \lambda\vec{x}y.z\mathcal{E}(a'_1, T_1^*) \cdots \mathcal{E}(a'_k, T_k^*)\mathcal{E}(a'_{k+1}, T_{k+1}^*) & \\
& = & \lambda\vec{x}y.z\mathcal{E}(a'_1, T_1^*) \cdots \mathcal{E}(a'_k, T_k^*)\bot^{y}_{\eta(y)} & \\
& \to_{\eta^{\mathsf{e}}} & \lambda\vec{x}.z\mathcal{E}(a'_1, T_1^*) \cdots \mathcal{E}(a'_k, T_k^*) & \text{using Remark 5.9} \\
& = & \mathcal{E}(a', \lambda\vec{x}.zT_1^* \cdots T_k^*). &
\end{array}$$

We conclude as $\mathcal{E}(a, T^*) \in \mathcal{E}(T^*)$. □

Lemma A.5 *For all Böhm trees T, we have $\eta(T)^* = \ulcorner \mathrm{nf}_{\eta^{\mathsf{e}}}(\mathcal{E}(T^*)) \urcorner$.*

Proof We proceed by co-induction on T.

If $T = \bot$, then $\eta(T)^* = \{\bot\} = \{\ulcorner \bot^{\emptyset} \urcorner\} = \{\ulcorner \mathcal{E}(\bot, \bot) \urcorner\} = \ulcorner \mathrm{nf}_{\eta^{\mathsf{e}}}(\mathcal{E}(T^*)) \urcorner$.

Otherwise, the Böhm tree T can be written in a unique way as $T = \lambda x_1 \ldots x_n y_1 \ldots y_m.zT_1 \cdots T_k T'_1 \cdots T'_m$ (for some $n, m, k \geq 0$) such that:

- $y_i \notin \mathrm{fv}(zT_1 \cdots T_k)$, T'_i is finite and $T'_i \twoheadrightarrow_\eta y_i$ for all $i \in \{1, \ldots, m\}$,
- $x_n \in \mathrm{fv}(zT_1 \cdots T_k)$ or T_k is infinite, or T_k is finite but does not η-reduce to x_n.

The following equalities hold:

$$
\begin{aligned}
\eta(T)^* &= \lambda\vec{x}.z\eta(T_1)^*\cdots\eta(T_k)^* \cup \{\bot\} && \text{by def. of } \eta(-)\\
&= \lambda\vec{x}.z\ulcorner \mathrm{nf}_{\eta^{\mathsf{e}}}(\mathcal{E}(T_1^*))\urcorner\cdots\ulcorner \mathrm{nf}_{\eta^{\mathsf{e}}}(\mathcal{E}(T_k^*))\urcorner \cup \{\bot\} && \text{by co-IH}\\
&= \ulcorner \lambda\vec{x}.z\mathrm{nf}_{\eta^{\mathsf{e}}}(\mathcal{E}(T_1^*))\cdots\mathrm{nf}_{\eta^{\mathsf{e}}}(\mathcal{E}(T_k^*))\urcorner\\
&\quad \cup \ulcorner\{\mathcal{E}(\bot,(\lambda\vec{x}.zT_1^*\cdots T_k^*)\downarrow)\}\urcorner && \text{by def. of } \ulcorner\cdot\urcorner\\
&= \ulcorner \lambda\vec{x}.z\mathrm{nf}_{\eta^{\mathsf{e}}}(\mathcal{E}(T_1^*))\cdots\mathrm{nf}_{\eta^{\mathsf{e}}}(\mathcal{E}(T_k^*)) && \text{by def. of } \ulcorner\cdot\urcorner\\
&\quad \cup \{\mathrm{nf}_{\eta^{\mathsf{e}}}(\mathcal{E}(\bot,(\lambda\vec{x}.zT_1^*\cdots T_k^*)\downarrow)\})\urcorner && \text{and of } \mathrm{nf}_{\eta}(-)\\
&= \ulcorner \mathrm{nf}_{\eta^{\mathsf{e}}}\big(\lambda\vec{x}.z\mathcal{E}(T_1^*)\cdots\mathcal{E}(T_n^*))\\
&\quad \cup \{\mathcal{E}(\bot,(\lambda\vec{x}.zT_1^*\cdots T_k^*)\downarrow)\}\big)\urcorner && \text{by def. of } \mathrm{nf}_{\eta}(-)\\
&= \ulcorner \mathrm{nf}_{\eta^{\mathsf{e}}}(\mathcal{E}(\lambda\vec{x}.zT_1^*\cdots T_k^*)\downarrow)\urcorner && \text{by def. of } \mathcal{E}(-)\\
&= \ulcorner \mathrm{nf}_{\eta^{\mathsf{e}}}(\mathcal{E}(T^*))\urcorner && \text{by Lemma 5.11.} \quad \Box
\end{aligned}
$$

5.12 *For all $M \in \Lambda$, we have* $\mathrm{BT}^{\eta}(M)^* = \ulcorner \mathrm{nf}_{\eta^{\mathsf{e}}}\mathcal{E}(\mathrm{BT}(M)^*)\urcorner$.

Proof Since $\mathrm{BT}^{\eta}(M) = \eta(\mathrm{BT}(M))$, the result follows directly by Lemma A.5. □

Lemma A.6 *Let $M \in \Lambda$ be a β-normal form such that $M \twoheadrightarrow_{\eta} x$. Then for all $t \in \mathcal{T}(M)$, we have $\mathcal{L}(t, \mathcal{T}(M)) \twoheadrightarrow_{\eta^{\ell}} x$.*

Proof By hypothesis, M has the shape $\lambda x_1 \dots x_n.xM_1 \cdots M_n$ (for some $n \geq 0$) such that, for all $i \in \{1, \dots, n\}$, $x \neq x_i$ and M_i is a β-normal form such that $M_i \twoheadrightarrow_{\eta} x_i$. We proceed by induction on t. Since $t \in \mathcal{T}(M)$, we have $t = \lambda x_1 \dots x_n.xb_1 \cdots b_n$ such that $b_i \in \mathcal{M}_{\mathrm{f}}(\mathcal{T}(M_i))$ for every $1 \leq i \leq n$. If $n = 0$ we are done. Otherwise, by Definition 5.1 we have $\mathcal{L}(t, \mathcal{T}(M)) = \lambda x_1 \dots x_n.x\mathcal{L}(b_1, \mathcal{M}_{\mathrm{f}}(\mathcal{T}(M_1))) \cdots \mathcal{L}(b_n, \mathcal{M}_{\mathrm{f}}(\mathcal{T}(M_n)))$.

Suppose $b_n = [t_1, \dots, t_k]$ with $t_j \in \mathcal{T}(M_n)$ for all $j \in \{1, \dots, k\}$.

If $k = 0$ then, by Definition 5.1, $\mathcal{L}(b_n, \mathcal{M}_{\mathrm{f}}(\mathcal{T}(M_n))) = 1^{x_n}_{\eta(x_n)}$ because $M_n \twoheadrightarrow_{\eta} x_n$ entails that there is $s \in \bigcup \mathcal{M}_{\mathrm{f}}(\mathcal{T}(M_n)) = \mathcal{T}(M_n)$ such that $s \twoheadrightarrow_{\eta'} x_n$. Therefore:

$$
\begin{aligned}
\mathcal{L}(t, \mathcal{T}(M_n)) &\twoheadrightarrow_{\eta^{\ell}} \lambda x_1 \dots x_n.x\mathcal{L}(b_1, \mathcal{M}_{\mathrm{f}}(\mathcal{T}(M_1))) \cdots \mathcal{L}(b_{n-1}, \mathcal{M}_{\mathrm{f}}(\mathcal{T}(M_{n-1})))1^{x_n}_{\eta(x_n)}\\
&\twoheadrightarrow_{\eta^{\ell}} \lambda x_1 \dots x_{n-1}.x\mathcal{L}(b_1, \mathcal{M}_{\mathrm{f}}(\mathcal{T}(M_1))) \cdots \mathcal{L}(b_{n-1}, \mathcal{M}_{\mathrm{f}}(\mathcal{T}(M_{n-1})))
\end{aligned}
$$

If $k > 0$, then by induction hypothesis $\mathcal{L}(t_{nj}, \mathcal{T}(M_n)) \twoheadrightarrow_{\eta^{\ell}} x_n$. Therefore,

$$
\begin{aligned}
\mathcal{L}(t, \mathcal{T}(M_n)) &\twoheadrightarrow_{\eta^{\ell}} \lambda x_1 \dots x_n.x\mathcal{L}(b_1, \mathcal{M}_{\mathrm{f}}(\mathcal{T}(M_1))) \cdots \mathcal{L}(b_{n-1}, \mathcal{M}_{\mathrm{f}}(\mathcal{T}(M_{n-1})))[x_n^k]\\
&\twoheadrightarrow_{\eta^{\ell}} \lambda x_1 \dots x_{n-1}.x\mathcal{L}(b_1, \mathcal{M}_{\mathrm{f}}(\mathcal{T}(M_1))) \cdots \mathcal{L}(b_{n-1}, \mathcal{M}_{\mathrm{f}}(\mathcal{T}(M_{n-1})))
\end{aligned}
$$

By iterating this reasoning on $b_1, \dots, b_{n-1}$ we conclude that $\mathcal{L}(t, \mathcal{T}(M)) \twoheadrightarrow_{\eta^{\ell}} x$. □

5.13 *Let $T = \lambda\vec{x}y.zT_1\cdots T_{k+1}$ be a Böhm tree such that T_{k+1} is finite, $T_{k+1} \twoheadrightarrow_\eta y$ and $y \notin \mathrm{fv}(zT_1\cdots T_k)$. Then $\mathcal{L}(\mathcal{T}(T)) \Rightarrow_{\eta^\ell} \mathcal{L}(\mathcal{T}(\lambda\vec{x}.zT_1\cdots T_k))$.*

Proof We first take $t \in \mathcal{T}(T)$, that is $t = \lambda\vec{x}y.zb_1\cdots b_{k+1}$ with $b_i \in \mathcal{M}_{\mathrm{f}}(\mathcal{T}(T_i))$, and show that $\mathcal{L}(t, \mathcal{T}(T)) \twoheadrightarrow_{\eta^\ell} \mathcal{L}(t', \mathcal{T}(\lambda\vec{x}.zT_1\cdots T_k))$ holds for $t' = \lambda\vec{x}.zb_1\cdots b_k \in \mathcal{L}(\mathcal{T}(\lambda\vec{x}.zT_1\cdots T_k))$. By definition of the labeling $\mathcal{L}(-)$, we have $\mathcal{L}(t, \mathcal{T}(T)) = \lambda\vec{x}y.z\mathcal{L}(b_1, \mathcal{M}_{\mathrm{f}}(\mathcal{T}(T_1)))\cdots\mathcal{L}(b_{k+1}, \mathcal{M}_{\mathrm{f}}(\mathcal{T}(T_{k+1})))$. By Remark 5.4 we have that $y \notin \mathrm{fv}(zT_1\cdots T_k)$ implies $y \notin \widetilde{\mathrm{fv}}(z\mathcal{L}(b_1, \mathcal{M}_{\mathrm{f}}(\mathcal{T}(T_1)))\cdots\mathcal{L}(b_k, \mathcal{M}_{\mathrm{f}}(\mathcal{T}(T_k)))$.

Suppose that $b_{k+1} = [t_1, \dots, t_n]$, we split into cases depending on n.

Case $n = 0$. As $T_{k+1} \twoheadrightarrow_\eta y$, then T_{k+1} is $\bot$-free finite tree, and therefore there exists an $s \in \mathcal{T}(T_{k+1})$ without empty bags such that $s \twoheadrightarrow_{\eta'} y$. Hence $\mathcal{L}(b_{k+1}, \mathcal{M}_{\mathrm{f}}(\mathcal{T}(T_{k+1}))) = \mathcal{L}(1, \mathcal{M}_{\mathrm{f}}(\mathcal{T}(T_{k+1}))) = 1^y_{\eta(y)}$ since $s \in \bigcup\mathcal{M}_{\mathrm{f}}(\mathcal{T}(T_{k+1})) = \mathcal{T}(T_{k+1})$. Therefore we have:

$$\begin{aligned}
\mathcal{L}(t, \mathcal{T}(T)) &= \lambda\vec{x}y.z\mathcal{L}(b_1, \mathcal{M}_{\mathrm{f}}(\mathcal{T}(T_1)))\cdots\mathcal{L}(b_{k+1}, \mathcal{M}_{\mathrm{f}}(\mathcal{T}(T_{k+1}))) \\
&= \lambda\vec{x}y.z\mathcal{L}(b_1, \mathcal{M}_{\mathrm{f}}(\mathcal{T}(T_1)))\cdots\mathcal{L}(b_k, \mathcal{M}_{\mathrm{f}}(\mathcal{T}(T_k)))1^y_{\eta(y)} \\
&\to_{\eta^\ell} \lambda\vec{x}y.z\mathcal{L}(b_1, \mathcal{M}_{\mathrm{f}}(\mathcal{T}(T_1)))\cdots\mathcal{L}(b_k, \mathcal{M}_{\mathrm{f}}(\mathcal{T}(T_k))) \\
&= \mathcal{L}(\lambda\vec{x}y.zb_1\cdots b_k, \mathcal{T}(\lambda\vec{x}y.zT_1\cdots T_k)).
\end{aligned}$$

Case $n > 0$**.** Then $t_i \in \mathcal{T}(T_{k+1})$ for $1 \leq i \leq n$, and $\mathcal{L}(b_{k+1}, \mathcal{M}_{\mathrm{f}}(\mathcal{T}(T_{k+1}))) = [\mathcal{L}(t_1, \mathcal{T}(T_{k+1})), \dots, \mathcal{L}(t_n, \mathcal{T}(T_{k+1}))]$. Since $T_{k+1} \twoheadrightarrow_\eta y$, then T_{k+1} is a $\bot$-free finite tree (that is a β-normal λ-term), so by Lemma A.6 we have $\mathcal{L}(t_i, \mathcal{T}(T_{k+1})) \twoheadrightarrow_{\eta^\ell} y$ for every $1 \leq i \leq n$. Therefore:

$$\begin{aligned}
\mathcal{L}(t, \mathcal{T}(T)) &= \lambda\vec{x}y.z\mathcal{L}(b_1, \mathcal{M}_{\mathrm{f}}(\mathcal{T}(T_1)))\cdots\mathcal{L}(b_{k+1}, \mathcal{M}_{\mathrm{f}}(\mathcal{T}(T_{k+1}))) \\
&\twoheadrightarrow_{\eta^\ell} \lambda\vec{x}y.z\mathcal{L}(b_1, \mathcal{M}_{\mathrm{f}}(\mathcal{T}(T_1)))\cdots\mathcal{L}(b_k, \mathcal{M}_{\mathrm{f}}(\mathcal{T}(T_k)))[y^n] \\
&\to_{\eta^\ell} \lambda\vec{x}y.z\mathcal{L}(b_1, \mathcal{M}_{\mathrm{f}}(\mathcal{T}(T_1)))\cdots\mathcal{L}(b_k, \mathcal{M}_{\mathrm{f}}(\mathcal{T}(T_k)))
\end{aligned}$$

Second, we take $s \in \mathcal{T}(\lambda\vec{x}.zT_1\cdots T_k)$, i.e. $s = \lambda\vec{x}.zb_1\cdots b_k$ with $b_i \in \mathcal{M}_{\mathrm{f}}(\mathcal{T}(T_i))$, and show that $\mathcal{L}(t, \mathcal{T}(T)) \twoheadrightarrow_{\eta^\ell} \mathcal{L}(s, \mathcal{T}(\lambda\vec{x}.zT_1\cdots T_k))$ for $t = \lambda\vec{x}y.zb_1\cdots b_k1 \in \mathcal{T}(T)$.

As $T_{k+1} \twoheadrightarrow_\eta y$, then T_{k+1} is $\bot$-free finite tree, and therefore there exists an $s \in \mathcal{T}(T_{k+1})$ without empty bags such that $s \twoheadrightarrow_{\eta'} y$. Thus $\mathcal{L}(1, \mathcal{M}_{\mathrm{f}}(\mathcal{T}(T_{k+1}))) = 1^y_{\eta(y)}$ since $s \in \bigcup\mathcal{M}_{\mathrm{f}}(\mathcal{T}(T_{k+1})) = \mathcal{T}(T_{k+1})$. Hence, we have:

$$\begin{aligned}
\mathcal{L}(t, \mathcal{T}(T)) &= \lambda\vec{x}y.z\mathcal{L}(b_1, \mathcal{M}_{\mathrm{f}}(\mathcal{T}(T_1)))\cdots\mathcal{L}(b'_k, \mathcal{M}_{\mathrm{f}}(\mathcal{T}(T_k)))\mathcal{L}(1, \mathcal{M}_{\mathrm{f}}(\mathcal{T}(T_{k+1}))) \\
&= \lambda\vec{x}y.z\mathcal{L}(b_1, \mathcal{M}_{\mathrm{f}}(\mathcal{T}(T_1)))\cdots\mathcal{L}(b_k, \mathcal{M}_{\mathrm{f}}(\mathcal{T}(T_k)))1^y_{\eta(y)} \\
&\to_{\eta^\ell} \lambda\vec{x}.z\mathcal{L}(b_1, \mathcal{M}_{\mathrm{f}}(\mathcal{T}(T_1)))\cdots\mathcal{L}(b_k, \mathcal{M}_{\mathrm{f}}(\mathcal{T}(T_k))) \\
&= \mathcal{L}(s, \mathcal{T}(\lambda\vec{x}.zT_1\cdots T_k)).
\end{aligned}$$

This completes the proof. □

Lemma A.7 *For all Böhm tree T the following equality holds:*

$$\mathcal{T}(\ulcorner \mathrm{nf}_{\eta^{\mathsf{e}}}\mathcal{E}(T^*)\urcorner) = \ulcorner \mathrm{nf}_{\eta^{\ell}}\mathcal{L}(\mathcal{T}(T))\urcorner$$

Proof We proceed by co-induction on T.

If $T = \bot$, then the equality follows because $\mathcal{T}(\bot) = \emptyset$.

Otherwise, the Böhm tree T can be written in a unique way as $T = \lambda x_1 \dots x_n y_1 \dots y_m . z T_1 \cdots T_k T'_1 \cdots T'_m$ (for some $n, m, k \geq 0$) such that:

- $y_i \notin \mathrm{fv}(zT_1 \cdots T_k)$, T'_i is finite and $T'_i \twoheadrightarrow_\eta y_i$ for all $i \in \{1, \dots, m\}$,
- $x_n \in \mathrm{fv}(zT_1 \cdots T_k)$ or T_k is infinite, or T_k is finite but does not η-reduce to x_n.

Therefore, the following equalities hold:

$$\begin{aligned}
&\mathcal{T}(\ulcorner \mathrm{nf}_{\eta^{\mathsf{e}}}\mathcal{E}(T^*)\urcorner) = \mathcal{T}(\ulcorner \mathrm{nf}_{\eta^{\mathsf{e}}}\mathcal{E}((\lambda x_1 \dots x_n . z T_1 \cdots T_k)\!\downarrow)\urcorner) && \text{by Lemma 5.11}\\
&= \mathcal{T}(\ulcorner \lambda \vec{x}.z \mathrm{nf}_{\eta^{\mathsf{e}}}(\mathcal{E}(T_1^*)) \cdots \mathrm{nf}_{\eta^{\mathsf{e}}}(\mathcal{E}(T_k^*))\urcorner \cup \{\ulcorner \mathcal{E}(\bot, \lambda \vec{x}.z T_1^* \cdots T_k^*)\urcorner\}) && \text{by def. of } \mathcal{E}(-)\\
&= \mathcal{T}(\ulcorner \lambda \vec{x}.z \mathrm{nf}_{\eta^{\mathsf{e}}}(\mathcal{E}(T_1^*)) \cdots \mathrm{nf}_{\eta^{\mathsf{e}}}(\mathcal{E}(T_k^*))\urcorner) \cup \mathcal{T}(\ulcorner \mathcal{E}(\bot, \lambda \vec{x}.z T_1^* \cdots T_k^*)\urcorner) && \text{by def. of } \mathcal{T}(-)\\
&= \lambda \vec{x}.z \mathcal{M}_{\mathrm{f}}(\mathcal{T}(\ulcorner \mathrm{nf}_{\eta^{\mathsf{e}}}(\mathcal{E}(T_1^*))\urcorner)) \cdots \mathcal{M}_{\mathrm{f}}(\mathcal{T}(\ulcorner \mathrm{nf}_{\eta^{\mathsf{e}}}(\mathcal{E}(T_k^*))\urcorner)) \cup \mathcal{T}(\bot) && \text{by def. of } \mathcal{T}(-)\\
&= \lambda \vec{x}.z \mathcal{M}_{\mathrm{f}}(\mathcal{T}(\ulcorner \mathrm{nf}_{\eta^{\mathsf{e}}}(\mathcal{E}(T_1^*))\urcorner)) \cdots \mathcal{M}_{\mathrm{f}}(\mathcal{T}(\ulcorner \mathrm{nf}_{\eta^{\mathsf{e}}}(\mathcal{E}(T_k^*))\urcorner)) && \text{since } \mathcal{T}(\bot) = \emptyset\\
&= \lambda \vec{x}.z \mathcal{M}_{\mathrm{f}}(\ulcorner \mathrm{nf}_{\eta^{\ell}}(\mathcal{L}(\mathcal{T}(T_1)))\urcorner) \cdots \mathcal{M}_{\mathrm{f}}(\ulcorner \mathrm{nf}_{\eta^{\ell}}(\mathcal{L}(\mathcal{T}(T_k)))\urcorner) && \text{by co-IH}\\
&= \ulcorner \mathrm{nf}_{\eta^{\ell}}(\lambda \vec{x}.z \mathcal{M}_{\mathrm{f}}(\mathcal{L}(\mathcal{T}(T_1))) \cdots \mathcal{M}_{\mathrm{f}}(\mathcal{L}(\mathcal{T}(T_k))))\urcorner && \text{by def. of } \mathrm{nf}_{\eta^{\ell}}\\
&= \ulcorner \mathrm{nf}_{\eta^{\ell}}\mathcal{L}(\lambda \vec{x}.z \mathcal{M}_{\mathrm{f}}(\mathcal{T}(T_1)) \cdots \mathcal{M}_{\mathrm{f}}(\mathcal{T}(T_k)))\urcorner && \text{by def. of } \mathcal{L}(-)\\
&= \ulcorner \mathrm{nf}_{\eta^{\ell}}\mathcal{L}(\mathcal{T}(\lambda \vec{x}.z T_1 \cdots T_k))\urcorner && \text{by def. of } \mathcal{T}(-)\\
&= \ulcorner \mathrm{nf}_{\eta^{\ell}}\mathcal{L}(\mathcal{T}(T))\urcorner && \text{by Lemma 5.13.}\square
\end{aligned}$$

5.14 *For all $M \in \Lambda$, $\mathcal{T}(\ulcorner \mathrm{nf}_{\eta^{\mathsf{e}}}\mathcal{E}(\mathrm{BT}(M)^*)\urcorner) = \ulcorner \mathrm{nf}_{\eta^{\ell}}\mathcal{L}(\mathcal{T}(\mathrm{BT}(M)))\urcorner$.*

Proof It follows directly from Lemma A.7. □

The Coinductive Resumption Monad

Maciej Piróg[1] and Jeremy Gibbons[2]

Department of Computer Science
University of Oxford

Abstract

Resumptions appear in many forms as a convenient abstraction, such as in semantics of concurrency and as a programming pattern. In this paper we introduce generalised resumptions in a category-theoretic, coalgebraic context and show their basic properties: they form a monad, they come equipped with a corecursion scheme in the sense of Adámek *et al.*'s notion of completely iterative monads (cims), and they enjoy a certain universal property, which specialises to the coproduct with a free cim in the category of cims.

Keywords: resumptions, completely iterative monads, coalgebra

1 Introduction

1.1 Resumptions

Resumptions were introduced by Milner [32] to denote the external behaviour of a communicating agent in concurrency theory. In categorical terms, as given by Abramsky [1], a resumption is an element of the carrier of the final coalgebra νR of the functor $RX = (X \times O)^I$ on $\textsc{Set}$, where I and O are the sets of possible inputs and outputs respectively. Informally, a resumption is a function that consumes some input and returns some output paired with another resumption, and finality implies that the process of consuming and producing can be iterated indefinitely. There are a lot of possible generalisations, for example to the final coalgebra of the functor $\mathcal{P}^{\text{fin}}((\text{-}) \times O)^I$ for agents with finitely-branching nondeterministic behaviour, or, depending on the computational effect realised by the agent, any monad in place of $\mathcal{P}^{\text{fin}}$, as studied by Hasuo and Jacobs [24].

The idea of 'resumable' computations appeared also in the context of computational effects and monadic programming. Cenciarelli and Moggi [13] defined what

[1] maciej.pirog@cs.ox.ac.uk

[2] jeremy.gibbons@cs.ox.ac.uk

they called the generalised resumption monad transformer as $TA = \mu X.M(\Sigma X + A)$, where M is a monad, Σ is an endofunctor, and A is an object of variables, which allows to sequentially compose resumptions. The resumption monad can be alternatively given by $M(\Sigma M)^*$, a composition of M and the free monad generated by the composition of Σ and M. It is canonical in the sense that it is the coproduct of M and Σ^* in the category of monads and monad morphisms, as shown by Hyland, Plotkin, and Power [26].

The resumption monad is given by an initial algebra, so it is not exactly a generalisation of the resumptions studied by Milner and others. Intuitively, it models resumptions that can be iterated only finitely many times. Thus, it is natural to ask about final coalgebras of functors of the shape $M(\Sigma(\text{-}) + A)$. Indeed, Goncharov and Schröder [21] used monads of the shape $TA = \nu X.M(X + A)$ to give semantics to concurrent processes with generic effects, while the monad $TA = \nu X.M(\Sigma X + A)$ was discussed by the present authors [39] under the name "coinductive resumption monad". In this paper, we further generalise the latter construction and take a closer look at its properties.

1.2 Coinduction

Usually, an effect-free data structure is called coinductive if it is given by the carrier of a final coalgebra. Informally, finality means that a coalgebra $c : X \to FX$ describes one step, which, repeated indefinitely, builds a structure of the type νF layer by layer. In the monadic world, however, the steps are often meant to interact – if we imagine that monadic values are computations, all the steps should be composed (monadically speaking, multiplied out) into one, big computation; if we view monads as algebraic theories, we should take into account that operations in one layer have their arguments in the next layer. Obviously, not every monad is coinductive in this sense, because the notion of multiplication of infinitely many layers is not always well-defined. Thus, to capture the notion of coinduction in the monadic context, we adopt a property called *complete iterativity*, introduced by Elgot *et al.* [16] and later studied by Adámek *et al.* [4,30]. A monad M is *completely iterative* (is a 'cim') if it is equipped with an additional coinductive structure: certain ('guarded') morphisms $e : X \to M(X + A)$, where X represents variables (seeds of the corecursion) and A is an object of parameters (final values), have unique solutions $e^\dagger : X \to MA$ coherent with the monadic structure of M.

Not too surprisingly, the usual notion of coinduction is captured by free completely iterative monads (informally: layers do not interact). The free completely iterative monad generated by an endofunctor F is given as $F^\infty A = \nu X.FX + A$ with the monadic structure given by substitution in A. An ordinary coalgebra $X \to FX$ can be seen as a guarded morphism $X \to F^\infty(X + 0)$, where 0 is the initial object of the base category, with the unique solution $X \to F^\infty 0 \cong \nu F$.

1.3 Contributions

The monad $TA = \nu X.M(\Sigma X + A)$ can alternatively be given as $M(\Sigma M)^\infty$. In Section 3, we generalise this construction to MS^∞, where S is any right module of M (all the necessary definitions and notations are given in Section 2). We give it a monadic structure and prove that it is completely iterative. Moreover, if M is also a cim, MS^∞ is a cim both 'vertically' (informally, we build new levels of the free structure) and 'horizontally' (we unfold more M structure) simultaneously.

In Section 4, we turn our attention back to the instance $M(\Sigma M)^\infty$. We show that it enjoys a certain universal property, which entails that it is the coproduct of Σ^∞ and M in the category of cims. In Section 5, we discuss corollaries and potential applications of our construction in semantics and programming.

Because of space limitations we present only short outlines of some proofs. For the full proofs, consult the associated technical appendix available online at `http://www.cs.ox.ac.uk/people/maciej.pirog/crm-appendix.pdf`.

2 Idealised and completely iterative monads

In the rest of the paper, we work in a base category $\mathbb{B}$ with finite coproducts. For brevity, we assume strict associativity of the coproduct bifunctor. The left and right injections are called inl and inr respectively. For an endofunctor $F : \mathbb{B} \to \mathbb{B}$, we denote the carrier of the initial F-algebra as μF, and the carrier the final F-coalgebra as νF. The unique morphism from a coalgebra $\langle A, g : A \to FA\rangle$ to the final coalgebra $\langle \nu F, \xi : \nu F \to F\nu F\rangle$ is written as $[\![g]\!]$. We use the letters M, N, T for monads. Their monadic structure is always denoted as η (unit) and μ (multiplication), possibly with superscripts to assign the natural transformations to the appropriate monad. The category of monads and monad morphism is denoted as $\textsc{Mnd}$, while the category of Eilenberg-Moore algebras of a monad M is denoted as M-$\textsc{Malg}$.

Definition 2.1 *Let M be a monad. An endofunctor $\overline{M}$ together with a natural transformation (an* action*) $\overline{\mu} : \overline{M}M \to \overline{M}$ is called a* (right) M-module *if $\overline{\mu} \cdot \overline{M}\eta = \mathsf{id} : \overline{M} \to \overline{M}$ and $\overline{\mu} \cdot \overline{\mu}M = \overline{\mu} \cdot \overline{M}\mu : \overline{M}M^2 \to \overline{M}$.*

We define a morphism between two M-modules $\langle \overline{M}, \overline{\mu}\rangle$ and $\langle \widetilde{M}, \widetilde{\mu}\rangle$ as a natural transformation $f : \overline{M} \to \widetilde{M}$ such that $\widetilde{\mu} \cdot fM = f \cdot \overline{\mu} : \overline{M}M \to \widetilde{M}$.

Slightly abusing notation, we often denote a module $\langle \overline{M}, \overline{\mu}\rangle$ simply as $\overline{M}$.

Example 2.2 The following are examples of right modules:

(i) For a monad M, the pair $\langle M, \mu^M\rangle$ is an M-module.

(ii) Let $m : M \to T$ be a monad morphism. Then, the pair $\langle T,\ \mu^T \cdot Tm\rangle$ is an M-module.

(iii) Let $\langle \overline{M}, \overline{\mu}^M\rangle$ be an M-module and F be an endofunctor. Then, $\langle F\overline{M}, F\overline{\mu}^M\rangle$ is an M-module.

(iv) With the definitions as above, let $\lambda : TM \to MT$ be a distributive law between monads. The pair $\langle \overline{M}T,\ (\overline{\mu}^M * \mu^T)\cdot \overline{M}\lambda T\rangle$ is a module of the induced monad MT.

(v) If $\langle \overline{M}, \overline{\mu}^M\rangle$ and $\langle \widetilde{M}, \widetilde{\mu}^M\rangle$ are two M-modules, the pair $\langle \overline{M}+\widetilde{M},\ \overline{\mu}^M+\widetilde{\mu}^M\rangle$ is also an M-module.

(vi) Let F be an endofunctor with a right adjoint U. Then, F is an UF-module with the action given by $\varepsilon F : FUF \to F$, where ε is the counit of the adjunction.

Definition 2.3 *An* idealised monad *is a triple consisting of a monad M, an M-module $\langle \overline{M}, \overline{\mu}^M\rangle$, and a module homomorphism $\sigma : \langle \overline{M}, \overline{\mu}^M\rangle \to \langle M, \mu^M\rangle$. We say that M is* idealised *with $\langle \overline{M}, \overline{\mu}^M\rangle$. If $M = \overline{M} + \mathsf{Id}$, we say that M is* ideal. *For an endofunctor F, a natural transformation $k : F \to M$ is* ideal *if it factors through σ.*

If not stated otherwise, for an idealised monad M, by $\overline{\mu}^M$ we always denote the action of the associated module $\overline{M}$, and by σ^M the associated module homomorphism.

Example 2.4 Extending Example 2.2 (iv) and (v), it holds that:

(i) Let M be idealised with $\overline{M}$ and $\lambda : TM \to MT$ be a distributive law between monads. The induced monad MT is idealised with $\overline{M}T$. The associated module morphism is given by $\sigma^M T : \overline{M}T \to MT$.

(ii) Let M be idealised with $\overline{M}$ as well as with $\widetilde{M}$. Then, M is idealised with $\overline{M}+\widetilde{M}$. The associated module morphism is given by $[\sigma, \sigma'] : \overline{M}+\widetilde{M} \to M$, where σ and σ' are the respective associated morphisms of the two modules.

Definition 2.5 *Let M be a monad idealised with $\overline{M}$. A morphism $e : X \to M(X+A)$ is called a* guarded equation morphism *if it factors as follows:*

$$\begin{tikzcd} X \arrow[r, dashed] & \overline{M}(X+A)+A \arrow[r, "{[\sigma_{X+A},\ \eta_{X+A}\cdot \mathsf{inr}_{X,A}]}"] & M(X+A) \end{tikzcd}$$

We call a morphism $e^\dagger : X \to MA$ a solution *of e if the following diagram commutes:*

$$\begin{tikzcd} X \arrow[r, "e^\dagger"] \arrow[d, "e"] & MA \\ M(X+A) \arrow[r, "{M[e^\dagger, \eta_A]}"] & M^2A \arrow[u, "\mu_A"'] \end{tikzcd}$$

An idealised monad M is completely iterative *(is a 'cim') if every guarded equation morphism has a unique solution.*

A monad morphism $m : M \to T$, where T is idealised with $\langle \overline{T}, \overline{\mu}^T\rangle$, is said to preserve solutions *if there exists a natural transformation $\overline{m} : \overline{M} \to \overline{T}$, such that $m \cdot \sigma^M = \sigma^T \cdot \overline{m} : \overline{M} \to T$.*

We denote the category of all cims and solution-preserving monad morphisms as Cim.

Note that we use a different notion of morphisms between cims than Adámek *et al.* [5], whose morphisms – called *idealised monad morphisms* – preserve also the structure of modules. The name 'solution-preserving' comes from the fact that for an equation morphism e and m as in the definition above, it holds that $m_{X+A} \cdot e$ is guarded and that $(m_{X+A} \cdot e)^\dagger = m_A \cdot e^\dagger$ (see the proof of a theorem by Milius [30, Prop. 5.9]).

An important example of a cim is the free cim Σ^∞ generated by an endofunctor Σ. Intuitively, it captures the monad of non-well-founded Σ-terms. Given an endofunctor Σ (a signature), if the final coalgebra $\langle \nu X.\Sigma X + A, \xi_A \rangle$ exists for all objects A, then we define $\Sigma^\infty A = \nu X.\Sigma X + A$. One can show that it is functorial in A, with the obvious action on morphisms, and that it has a monadic structure given by substitution in A, which we denote as η^∞ and μ^∞. The monad Σ^∞ is ideal and completely iterative. We define a natural transformation $\mathsf{emb} : \Sigma \to \Sigma^\infty$ as:

$$\mathsf{emb}_A = \left(\Sigma A \xrightarrow{\Sigma\eta^\infty} \Sigma\Sigma^\infty A \xrightarrow{\mathsf{inl}} \Sigma\Sigma^\infty A + A \xrightarrow{\xi^{-1}} \Sigma^\infty A \right)$$

As discussed by Aczel *et al.* [4], Σ^∞ is the free cim generated by Σ. Intuitively, this means that every ideal interpretation of Σ in a cim M extends in a unique way to an interpretation of the entire structure in M. Formally:

Theorem 2.6 *For a cim M and an ideal natural transformation $k : \Sigma \to M$, there exists a unique monad morphism $\iota(k) : \Sigma^\infty \to M$ such that $k = \iota(k) \cdot \mathsf{emb}$. Moreover, the morphism $\iota(k)$ preserves solutions.*

We also need the following cancellation property:

Lemma 2.7 *For a cim M and a solution-preserving monad morphism $m : \Sigma^\infty \to M$, the composition $m \cdot \mathsf{emb}$ is an ideal natural transformation, and $\iota(m \cdot \mathsf{emb}) = m$.*

3 Monadic structure and complete iterativity

Let $\langle S, \overline{\mu}^S \rangle$ be a right M-module such that S^∞ exists. We give a monadic structure to MS^∞ via a distributive law [11]. This construction is an adaptation of Hyland, Plotkin, and Power's proof [26] that the inductive resumptions $M(\Sigma M)^*$ form a monad. We use the fact due to Adámek, Milius, and Velebil [7] that the category of complete Elgot algebras is strictly monadic over the base category $\mathbb{B}$. Note that we cannot employ Uustalu's construction on parametrised monads [41] (successfully used by Goncharov and Schröder [21] in the special case of MM^∞), since MS^∞ is not in general given by the carrier of a final coalgebra, and also we make extensive use of the distributive law later in this paper. We start by recalling the definition of Elgot algebras [7].

Definition 3.1 *Let H be an endofunctor. For two objects A and X, we call a morphism $e : X \to HX + A$ a* flat equation morphism. *We call a morphism $e^\dagger : X \to A$ a* solution *in an H-algebra $a : HA \to A$ if $e^\dagger = [a \cdot He^\dagger, \mathsf{id}] \cdot e$.*

Just like in the case of cims, we denote the solutions in Elgot algebras by $(\text{-})^\dagger$ or $(\text{-})^\ddagger$. This overloading should not impose any confusion, since we are always clear about the types.

Definition 3.2 *For flat equation morphisms $e : X \to HX + Y$ and $f : Y \to HY + A$, and a morphism $h : Y \to Z$, we define two operations:*

$$h \triangleleft e = \Big(X \xrightarrow{e} HX + Y \xrightarrow{\mathsf{id}+h} HX + Z\Big)$$

$$f \ominus e = \Big(X + Y \xrightarrow{[e,\mathsf{inr}]} HX + Y \xrightarrow{\mathsf{id}+f} HX + HY + A \xrightarrow{[H\mathsf{inl},H\mathsf{inr}]+\mathsf{id}} H(X + Y) + A\Big)$$

Definition 3.3 *For an endofunctor H, a* complete Elgot H-algebra *is a triple $\langle A, a : HA \to A, (\text{-})^\dagger\rangle$, where $(\text{-})^\dagger$ assigns to every flat equation morphism $e : X \to HX + A$ a solution $e^\dagger : X \to A$ such that the following two conditions hold:*

- *(Functoriality) For two equation morphisms $e : X \to HX+A$ and $f : Y \to HY+A$ understood as $H(\text{-})+A$ coalgebras, let $h : X \to Y$ be a coalgebra homomorphism, that is $f \cdot h = (Hh + \mathsf{id}_A) \cdot e$. Then, $e^\dagger = f^\dagger \cdot h$.*
- *(Compositionality) For two equation morphisms $e : X \to HX + Y$ and $f : Y \to HY + A$ it holds that $(f^\dagger \triangleleft e)^\dagger = (f \ominus e)^\dagger \cdot \mathsf{inl}$.*

Definition 3.4 *For two complete Elgot H-algebras $\langle A, a, (\text{-})^\dagger\rangle$ and $\langle B, b, (\text{-})^\ddagger\rangle$, a morphism $h : A \to B$ is said to* preserve solutions *if for every flat equation morphism $e : X \to HX + A$ it holds that $(h \triangleleft e)^\ddagger = h \cdot e^\dagger$. Complete Elgot H-algebras and solution-preserving morphisms form a category, which we denote as H-*Elgot*.*

Theorem 3.5 (Adámek, Milius, Velebil [7]) *The obvious forgetful functor $U_{\text{Elg}} : H\text{-}\textsc{Elgot} \to \mathbb{B}$ is strictly H^∞-monadic (or simply 'monadic' in Mac Lane's terminology [29, Ch. 6]), which entails $H\text{-}\textsc{Elgot} \cong H^\infty\text{-}\textsc{Malg}$.*

Construction 3.6 *Recall that $\langle S, \overline{\mu}^S\rangle$ is a right M-module. For a complete Elgot algebra $\langle A, a : SA \to A, (\text{-})^\dagger\rangle$ we define an algebra $\langle MA,\ a' : SMA \to MA,\ (\text{-})^\ddagger\rangle$ as follows:*

$$a' = \Big(SMA \xrightarrow{\overline{\mu}^S} SA \xrightarrow{a} A \xrightarrow{\eta^M} MA\Big)$$

Let $e : X \to SX + MA$ be a flat equation morphism. We define an auxiliary morphism $|e|$ and a solution $e^\ddagger$:

$$|e| = \Big(SX + A \xrightarrow{Se+\mathsf{id}} S(SX + MA) + A \xrightarrow{\mathsf{flat}+\mathsf{id}} S(SX + A) + A\Big)$$

$$e^\ddagger = \Big(X \xrightarrow{e} SX + MA \xrightarrow{\mathsf{inl}+\mathsf{id}} SX + A + MA \xrightarrow{|e|^\dagger+\mathsf{id}} A + MA \xrightarrow{[\eta^M,\mathsf{id}]} MA\Big)$$

where $\mathsf{flat}_{A,B}$ *is given as:*

$$S(A + MB) \xrightarrow{S(\eta^M+\mathsf{id})} S(MA + MB) \xrightarrow{S[M\mathsf{inl},M\mathsf{inr}]} SM(A + B) \xrightarrow{\overline{\mu}^S} S(A + B)$$

Lemma 3.7 *The triple $\langle MA, a', (\text{-})^\ddagger\rangle$ from Construction 3.6 is a complete Elgot algebra. Moreover, the assignment $\langle A, a, (\text{-})^\dagger\rangle \mapsto \langle MA, a', (\text{-})^\ddagger\rangle$ on objects and $f \mapsto$*

*Mf on morphisms is an endofunctor on S-*Elgot *with a monadic structure given by the monadic structure of M.*

Theorem 3.8 *The composition MS^∞ is a monad.*

Proof. The assignment from Lemma 3.7 is a monad, so it is a lifting of M to S-Elgot with respect to U_{Elg}. Thus, by Theorem 3.5, it is a lifting of M to S^∞-Malg. This induces a distributive law between monads $\lambda : S^\infty M \to MS^\infty$, which gives a monadic structure to MS^∞. □

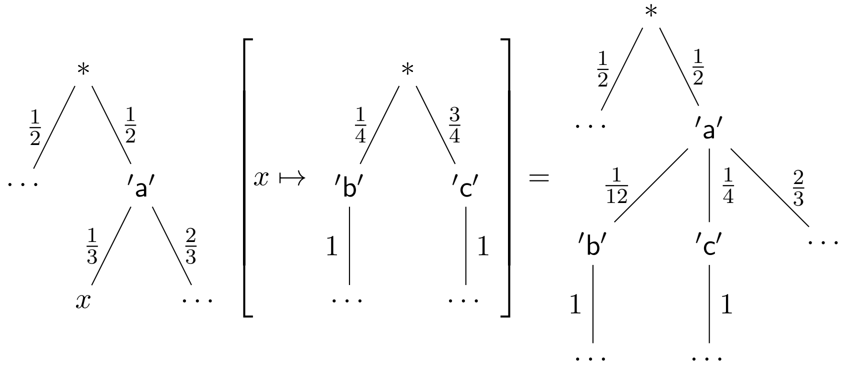


Fig. 1. Example of a substitution in the $\mathcal{D}(O \times \mathcal{D}(\text{-}))^\infty$ monad.

Example 3.9 (a) Let $\mathbb{B}$ be Set, $\mathcal{D}$ be the monad of discrete probability distributions, $O = \{\mathsf{a}, \mathsf{b}, \ldots\}$ be a set, and $\Sigma X = O \times X$ be an endofunctor. A value of the monad $\mathcal{D}(\Sigma\mathcal{D})^\infty X$ is a countably branching, possibly infinite decision tree with elements of the set O in nodes (except for the root), edges labelled with probabilities, and elements of X in leaves. For illustration purposes, it is important that there is no label in the root – we take a random step before generating a label and a random step before reaching a leaf. This way, when we substitute a tree for a variable, we take two random steps before generating a label or reaching a leaf. The monadic structure of $\mathcal{D}(\Sigma\mathcal{D})^\infty X$ takes care of flattening these to one random step by multiplying out the adjacent distributions; see Figure 1 for an example.

(b) In a cartesian closed category, we can define a version of the state monad that keeps track of all (possibly infinitely many) intermediate states. It is similar but not identical to 'states' given in [39]. Fix an object of states A and consider the reader monad $\mathcal{R}X = X^A$. By Example 2.2 (vi), the writer $\mathcal{W}X = X \times A$ is an $\mathcal{R}$-module. More directly, the action can be given as $\langle \mathsf{ev}^{\mathsf{A}}_{\mathsf{X}}, \mathsf{outr}\rangle : \mathcal{W}\mathcal{R}X = X^A \times A \to X \times A = \mathcal{W}X$, where ev is the evaluation morphism of the exponential object, outr is the right projection, and $\langle \text{-}, \text{-} \rangle$ is the product mediator. Intuitively, for an initial state, the monad $\mathcal{R}\mathcal{W}^\infty = ((\text{-} \times A)^\infty)^A$ produces a (possibly infinite) stream of intermediate states $\mathcal{W}^\infty$. If the stream is finite, it is terminated with a final value of the computation. The monad $\mathcal{R}\mathcal{W}^\infty$ is an instance of the resumption monad that in general is not given by a final coalgebra. (Compare also Ahman and Uustalu's update monads [8].)

Now, assume that M is completely iterative with respect to a module $\overline{M}$. Note that an 'ordinary' monad is always a cim with respect to the module C_0 (the constant functor returning the initial object), so this assumption does not narrow down the choice of M. We prove MS^∞ to be a cim with respect to the module $\overline{M}S^\infty + MSS^\infty$. This means that each coinductive step can unfold the structure 'horizontally' (by unfolding more structure of M), 'vertically' (unfolding the free structure), or both.

Definition 3.10 *For a monad T, a* distributive law *of a T-module $\overline{T}$ over a monad M is a distributive law between monads $\lambda : TM \to MT$ together with a natural transformation $\overline{\lambda} : \overline{T}M \to M\overline{T}$ such that the obvious analogues of the diagrams for distributive laws between monads commute (except for the diagram for η^T, since in general $\overline{T}$ is not a monad). Moreover, if T is idealised with $\overline{T}$, we say that the tuple $\langle \lambda, \overline{\lambda} \rangle$* preserves modules *if $M\sigma^T \cdot \overline{\lambda} = \lambda \cdot \sigma^T M : \overline{T}M \to MT$.*

Lemma 3.11 *Let T be an idealised monad and let $\langle \lambda, \overline{\lambda} \rangle$ be a module-preserving distributive law of $\overline{T}$ over M. Then, the induced monad MT is idealised with $M\overline{T}$.*

One can show that $\lambda : S^\infty M \to MS^\infty$ from the proof of Theorem 3.8 can be extended to a module-preserving distributive law of SS^∞ over M. Hence, Lemma 3.11 together with Example 2.4 lead us to the following corollary:

Theorem 3.12 *The monad MS^∞ is idealised with respect to $\overline{M}S^\infty + MSS^\infty$.*

We describe a solution in MS^∞ as a two-step process. Intuitively, given an equation morphism $e : X \to MS^\infty(X + A)$, we first unfold 'horizontally' via the completely iterative structure of M. We are left with what can be seen as an equation morphism in a monad induced by the distributive law λ on the Kleisli category of M. This morphism (the 'vertical' unfolding) has a unique solution, too, which is the desired solution in MS^∞.

Lemma 3.13 *Let M be a cim. Let $e : X \to M(X + A + B)$ factor as follows:*

$$\begin{tikzcd} X \arrow[r, dashed] & \overline{M}(X+A+B) + MA + B \arrow[r, "{[\sigma,\ M(\mathsf{inl}\cdot\mathsf{inr}),\ \eta\cdot\mathsf{inr}]}"] & M(X+A+B) \end{tikzcd}$$

The morphism e has a unique solution $e^\dagger : X \to M(A + B)$.

Lemma 3.14 *Let $e : X \to MS^\infty(X + A)$ be a morphism that factors as follows:*

$$\begin{tikzcd} X \arrow[r, dashed] & M(SS^\infty(X+A) + A) \arrow[r, "{M[\sigma^\infty,\ \eta^\infty\cdot\mathsf{inr}]}"] & MS^\infty(X+A) \end{tikzcd}$$

Then, e has a unique solution $e^\ddagger$ in MS^∞.

Proof. Consider the Kleisli category of M, denoted as $\mathcal{K}\ell(M)$. For a morphism $f : A \to MB$ in $\mathcal{K}\ell(M)$, we define a $\mathbb{B}$-morphism $\widehat{f} = \overline{\mu}^S_B \cdot Sf : SA \to SB$. We define an endofunctor G on $\mathcal{K}\ell(M)$ given as $GA = SA$ on objects and $G(f : A \to MB) = \eta^M_B \cdot S\widehat{f} : SA \to MSB$ on morphisms. The distributive law λ induces a monad $\langle\!\langle S^\infty \rangle\!\rangle$ on $\mathcal{K}\ell(M)$ given by $\langle\!\langle S^\infty \rangle\!\rangle A = S^\infty A$ and $\langle\!\langle S^\infty \rangle\!\rangle(f : A \to MB) = \lambda_B \cdot S^\infty f : S^\infty A \to MS^\infty B$. One can show that $\langle\!\langle S^\infty \rangle\!\rangle$ is the free cim generated by G in $\mathcal{K}\ell(M)$. The morphism e is a guarded equation morphism in $\langle\!\langle S^\infty \rangle\!\rangle$, so it has a unique solution $e^{\ddagger} : X \to M\langle\!\langle S^{\infty}\rangle\!\rangle A$. One can verify that it is the desired morphism and that it is unique. □

Now, we can prove the main theorem:

Theorem 3.15 *The monad MS^{∞} is completely iterative with respect to the module $\overline{M}S^{\infty} + MSS^{\infty}$.*

Proof. In this proof, for brevity, we denote S^{∞} as T and its module SS^{∞} as $\overline{T}$. Let $e : X \to MT(X+A)$ be a guarded equation morphism. This means that it factors as $X \dashrightarrow \overline{M}T(X+A)+M\overline{T}(X+A)+A \xrightarrow{[\sigma^M, M\sigma^T, \eta\cdot\mathsf{inr}]} MT(X+A)$. Since T is an ideal monad, there exist isomorphisms $T(X+A) \cong \overline{T}(X+A)+X+A \cong X+\overline{T}(X+A)+A$. Thus, e factors as $X \dashrightarrow \overline{M}(X+\overline{T}(X+A)+A)+M\overline{T}(X+A)+A \xrightarrow{[\sigma^M, M(\mathsf{inl}\cdot\mathsf{inr}), \eta\cdot\mathsf{inr}]} M(X+\overline{T}(X+A)+A)$, and it is a guarded equation morphism in the monad M in the sense of Lemma 3.13. Therefore, there exists a unique solution $e^{\dagger} : X \to M(\overline{T}(X+A)+A)$. The morphism $M[\sigma^T, \eta^T \cdot \mathsf{inr}] \cdot e^{\dagger} : X \to MT(X+A)$ is a guarded equation morphism in the sense of Lemma 3.14, so it has a unique solution $(M[\sigma^T, \eta^T \cdot \mathsf{inr}] \cdot e^{\dagger})^{\ddagger} : X \to MTA$. One can verify that it is a unique solution of e in MT. □

Example 3.16 Consider the monad $\mathcal{D}(\Sigma\mathcal{D})^{\infty}$ from Example 3.9 (a). It is a cim, which gives us a semantics for discrete Markov processes with sequential composition. An equation morphism $e : X \to \mathcal{D}(\Sigma\mathcal{D})^{\infty}(X+A)$ is a transition system, where X is the state space. The solution $e^{\dagger} : X \to \mathcal{D}(\Sigma\mathcal{D})^{\infty}A$ is a denotational semantics: for an initial state, it gives us the decision tree – the intermediate (internal) states are forgotten. The solution diagram amounts to adequacy.

For an example of the horizontal and vertical complete iterativity, consider the monad $\mathcal{D}'X = \mathcal{D}(1 + X)$ for the terminal object 1, which denotes failure. It is a cim with respect to the module that consists of those values of $\mathcal{D}'$ that fail with the probability at least 0.5. Each transition of a process denoted by $\mathcal{D}'(\Sigma\mathcal{D}')^{\infty}$ can either produce an output from the set O or perform a silent step, but with the probability of failure at least 0.5.

4 Universal property and coproduct

A particularly interesting instance of MS^{∞} is $M(\Sigma M)^{\infty}$. In this section, we investigate its universal property: for a cim N, a monad morphism $M \to N$, and an ideal natural transformation $\Sigma \to N$, there exists a unique coherent monad morphism $M(\Sigma M)^{\infty} \to N$. Informally, morphisms that 'interpret' every level of the structure of a resumption in another cim uniquely extend to an interpretation of the whole structure. First, we define three 'injections':

$$\mathsf{liftl} = \left(\Sigma^{\infty} \xrightarrow{\iota(\mathsf{emb}\cdot\Sigma\eta^M)} (\Sigma M)^{\infty} \xrightarrow{\eta^M(\Sigma M)^{\infty}} M(\Sigma M)^{\infty}\right)$$

$$\mathsf{liftr} = \left(M \xrightarrow{M\eta^{\infty}} M(\Sigma M)^{\infty}\right) \qquad \mathsf{liftf} = \left(\Sigma \xrightarrow{\mathsf{emb}} \Sigma^{\infty} \xrightarrow{\mathsf{liftl}} M(\Sigma M)^{\infty}\right)$$

It is easy to see that liftl and liftr are monad morphisms (liftl is a composition of two monad morphisms). They are also solution-preserving, the former via $M\Sigma M(\Sigma M)^\infty$, the latter via $\overline{M}(\Sigma M)^\infty$.

Definition 4.1 *By $\Sigma/\textsc{Cim}$ we denote the following category: objects are ideal natural transformations $(\Sigma \xrightarrow{f} T)$, where T is a cim; morphisms are (not necessarily solution-preserving) monad morphisms $k : T \to N$ such that the following diagram commutes:*

$$\begin{tikzcd}
& T \arrow[dd, "k"] \\
\Sigma \arrow[ur, "f"] \arrow[dr, "g"'] & \\
& N
\end{tikzcd}$$

We define a forgetful functor $U : \Sigma/\textsc{Cim} \to \textsc{Mnd}$ as $U(\Sigma \xrightarrow{f} M) = M$ on objects and $Uk = k$ on morphisms.

Definition 4.2 *A monad M is called Σ-resumable if $(\Sigma M)^\infty$ exists. By Σ-$\textsc{Res}$ we denote the full subcategory of $\textsc{Mnd}$ with Σ-resumable cims as objects. We denote the inclusion functor by $I : \Sigma\text{-}\textsc{Res} \to \textsc{Mnd}$.*

We establish the universal property in terms of an I-relative adjunction. For a discussion on relative adjoints, see, for example, Altenkirch *et al.* [9].

Theorem 4.3 *The functor U has an I-relative left adjoint $F : \Sigma\text{-}\textsc{Res} \to \Sigma/\textsc{Cim}$ given by $FM = (\Sigma \xrightarrow{\mathsf{liftf}} M(\Sigma M)^\infty)$ on objects and $Fm = m * [\![(\Sigma m(\Sigma M)^\infty + \mathsf{id}) \cdot \xi]\!] = m * \iota(\mathsf{emb} \cdot \Sigma m)$ on morphisms.*

Proof. One can show that the morphism Fm is a monad morphism, so F is indeed a functor. Let M be a Σ-resumable monad, N be a cim, and $g : \Sigma \to N$ be an ideal natural transformation. We define the isomorphism

$$\lfloor\text{-}\rfloor : \Sigma/\textsc{Cim}[FM, (\Sigma \xrightarrow{g} N)] \cong \textsc{Mnd}[IM, U(\Sigma \xrightarrow{g} N)] : \lceil\text{-}\rceil$$

by

$$\begin{array}{ll}
k : (\Sigma \xrightarrow{\mathsf{liftf}} M(\Sigma M)^\infty) \to (\Sigma \xrightarrow{f} N) & \quad m : M \to U(\Sigma \xrightarrow{f} N) \\
\lfloor k \rfloor : M \to U(\Sigma \xrightarrow{f} N) & \quad \lceil m \rceil : (\Sigma \xrightarrow{\mathsf{liftf}} M(\Sigma M)^\infty) \to (\Sigma \xrightarrow{f} N) \\
\lfloor k \rfloor = k \cdot M\eta^{(\Sigma M)^\infty} & \quad \lceil m \rceil = \mu^N \cdot (m * \iota(\mu^N \cdot (f * m)))
\end{array}$$

For $\lceil m \rceil$ to be a well-defined morphism in $\Sigma/\textsc{Cim}$, we note that $\lceil m \rceil \cdot \mathsf{liftf} = f$. Using the properties of $M(\Sigma M)^\infty$ and free cims, one can verify that $\lfloor(\text{-})\rfloor$ is natural in M and g, and that $\lfloor(\text{-})\rfloor$ and $\lceil(\text{-})\rceil$ are mutual inverses. □

An alternative reading of Theorem 4.3 is that $(\Sigma \xrightarrow{\mathsf{liftf}} M(\Sigma M)^\infty)$ is the free object in $\Sigma/\textsc{Cim}$ generated by a Σ-resumable cim M. This means that for a cim N, an ideal natural transformation $g : \Sigma \to N$, and a monad morphism $m : M \to N$, there exists a unique monad morphism $j : M(\Sigma M)^\infty \to N$ such that the following diagram commutes (one can easily see that liftr is the unit of the relative adjunction):

$$\begin{tikzcd}
\Sigma \arrow[r, "\mathsf{liftf}"] \arrow[dr, "g"'] & M(\Sigma M)^{\infty} \arrow[d, dashed, "j"] & M \arrow[l, "\mathsf{liftr}"'] \arrow[dl, "m"] \\
& N &
\end{tikzcd}$$

Note that if $M = \mathsf{Id}$, the diagram above instantiates to the fact that Σ^{∞} is indeed the free cim generated by Σ. More precisely, the identity monad Id is initial in Mnd (the unique monad morphism $! : \mathsf{Id} \to N$ is given by the unit of N), so the right-hand side of the diagram above becomes trivial, and what is left is exactly the diagram from Theorem 2.6.

Also, for a solution-preserving monad morphism $n : \Sigma^{\infty} \to N$, the composition $n \cdot \mathsf{emb} : \Sigma \to N$ is an ideal natural transformation. For a solution-preserving monad morphism $m : M \to N$, there exists a unique monad morphism $j : M(\Sigma M)^{\infty} \to N$ such that $m = j \cdot \mathsf{liftr}$ and $n \cdot \mathsf{emb} = j \cdot \mathsf{liftf} = j \cdot \mathsf{liftl} \cdot \mathsf{emb}$. This means that $\iota(n \cdot \mathsf{emb}) = \iota(j \cdot \mathsf{liftl} \cdot \mathsf{emb})$, hence, by Lemma 2.7, we get $n = j \cdot \mathsf{liftl}$. Therefore, the morphism j uniquely makes the following diagram commute:

$$\begin{tikzcd}
\Sigma^{\infty} \arrow[r, "\mathsf{liftl}"] \arrow[dr, "n"'] & M(\Sigma M)^{\infty} \arrow[d, dashed, "j"] & M \arrow[l, "\mathsf{liftr}"'] \arrow[dl, "m"] \\
& N &
\end{tikzcd} \tag{1}$$

The morphism $j = U\lceil m \rceil_{M, n \cdot \mathsf{emb}}$ does not necessarily preserve solutions, even though liftl, liftr, m, and n do. Informally, the action of j on the right component of $M(\Sigma M)^{\infty}$'s module $\overline{M}(\Sigma M)^{\infty} + M\Sigma M(\Sigma M)^{\infty}$ is not guaranteed to take the leading M into $\overline{N}$. Nevertheless, we can amend the situation if we know that N is a cim with respect to a two-sided module:

Definition 4.4 *A two-sided module of a monad N is its right module $\langle \overline{N}, \overline{\mu}^{\mathsf{R}} \rangle$ together with a natural transformation $\overline{\mu}^{\mathsf{L}} : N\overline{N} \to \overline{N}$ such that the obvious diagrams mirroring those of Definition 2.1 commute and $\overline{\mu}^{\mathsf{R}} \cdot \overline{\mu}^{\mathsf{L}} N = \overline{\mu}^{\mathsf{L}} \cdot N\overline{\mu}^{\mathsf{R}} : N\overline{N}N \to \overline{N}$. Similarly, we adjust the definition of homomorphisms between modules and idealised monads. We denote the category of monads that are completely iterative with respect to two-sided ideals and solution-preserving monad morphisms as* TCim.

In the context of the properties studied in this paper, we can switch from Cim to TCim at no cost:

Theorem 4.5 *The category* TCim *is a full reflective subcategory of* Cim. *In other words, the obvious embedding functor $U_{\mathrm{TC}} :$ TCim $\to$ Cim has a left adjoint F_{TC}.*

In practise, this means that for a monad N completely iterative with respect to a right ideal $\overline{N}$, there exists a two-sided ideal $\widetilde{N}$ (given by $N\overline{N}$) and a module homomorphism $\overline{N} \to \widetilde{N}$ (that is, every equation morphism guarded via $\overline{N}$ is also guarded via $\widetilde{N}$), such that N is completely iterative with respect to $\widetilde{N}$. In such a case, since j from the diagram (1) is equal to $\mu^{N} \cdot (m * \iota(\mu^{N} \cdot ((n \cdot \mathsf{emb}) * m)))$ and the right-hand side argument of the left-most $*$ preserves solutions (Theorem 2.6), the morphism j is solution-preserving too, which can be verified by a simple diagram chase. In general, the module of Σ^∞ is two-sided, but the module of $M(\Sigma M)^\infty$ is not. Thus, taking the reflection of the diagram (1) in TCIM, we obtain the following corollary, which is a 'completely iterative' counterpart of Hyland, Plotkin and Power's result that $M(\Sigma M)^*$ is a coproduct of Σ^* and M in MND [26]:

Corollary 4.6 *For an endofunctor Σ and a monad M completely iterative with respect to a two-sided ideal, the F_{TC}-image of $M(\Sigma M)^\infty$ (that is, $M(\Sigma M)^\infty$ idealised with $M(\Sigma M)^\infty(\overline{M}(\Sigma M)^\infty + M\Sigma M(\Sigma M)^\infty)$) is the coproduct of Σ^∞ and M in* TCIM*.*

5 Discussion

5.1 Complete iterativity

The results presented in this paper are in general contributions to the study of completely iterative monads [4]. We give new examples of cims and describe coproducts with free cims. Note that if M is ideal, then so is MS^∞, which means that our constructions scale to the category of ideal cims, if one prefers to work in such a setting.

Our results suggest a form of 'least- vs greatest fixed points' duality between ordinary monads and cims: free objects are given by $F^*A = \mu X.FX + A$ and $F^\infty A = \nu X.FX + A$ respectively, and coproducts with free objects by $M(\Sigma M)^*$ and $M(\Sigma M)^\infty$. There are other constructions on monads that involve initial algebras, for example the coproduct of two ideal monads, as shown by Ghani and Uustalu [20]. We conjecture that a similar construction with ν in place of μ yields the coproduct in the category of ideal cims.

Adámek *et al.* [5,6] consider also a finitary case: they define an *iterative monad* (without 'completely') as a finitary monad on a locally finitely presentable category, such that all guarded equation morphisms with finitely presentable object of variables have unique solutions. Similarly, there exists a finitary version of complete Elgot algebras (namely, *Elgot algebras*) together with an analogue of Theorem 3.5. This suggests that the presented constructions should scale to the finitary case, but we have not yet worked out the details.

Elgot's original results were set in the context of algebraic theories rather than general category theory. As a future direction of research, it would be interesting to look at structures based on resumptions from the point of view of algebraic specification (operations and equations), especially those that could be used in semantics and programming, like the logging monad from Example 3.9 (b).

5.2 Semantics and programming

As suggested by the present authors in a previous paper [39], models in the style of Moggi [34] of effects generating observable behaviour (such as I/O) require a form of complete iterativity, given that programs need not terminate. Thus, by understanding the category of cims, one hopes for a better understanding of such effects. For example, if one pursues modularity, the coproduct in Mnd of two cims is not in general completely iterative. So, a much better notion of the 'smallest' amalgam of such effects would be their coproduct in TCim. This has a practical aspect: the Haskell programming language is equipped with a single 'all-inclusive' *IO* monad, incorporating effects as diverse as textual interaction, file handling, system calls, the foreign function interface, random number generation, and concurrency; one would hope for more fine-grained components that indicate the actual impure effects in use (see Peyton Jones [27] for discussion).

Resumptions-like structures are used as denotations of processes, that is programs that exhibit observable behaviour in the course of execution (see Abramsky [1]). A monadic structure captures the notion of composition, which allows the resumption monad to serve as a basis of a programming calculus in the style of Moggi [34]. For example, Goncharov and Schröder [21] give a simple semantics for asynchronous side-effecting concurrent processes using the monad MM^∞. We hope that the richer structure of MS^∞, can be used to describe more advanced behaviours and synchronisation of processes along the lines of Abramsky's interaction categories [2].

In pure functional programming, monads are often used as an embedded domain-specific language (EDSL) used to represent computational effects, built from atomic actions using functoriality and monadic structure. Very often such monadic values describe not necessarily terminating (thus, in a sense, coinductive) programs with non-trivial, parallel resource management, like lazy I/O; see, for example, Kiselyov's iteratees [28]. Such programs are often represented by data structures similar to monadic resumptions, which are in a close relationship with the outer, 'imperative' monad, such as Haskell's *IO*. The relationship between the two can be formalised by the operations discussed in this paper: *IO* actions can be lifted to the level of resumptions (liftr), while the whole EDSL program can be executed, that is flattened back into *IO*, which can be modelled by the universal property.

5.3 Related work

The notion of complete iterativity was introduced by Elgot [16], and later studied by Aczel *et al.* [4,30]. The properties of the monad Σ^∞ were studied also by Moss [35] and Ghani *et al.* [19]. Milius and Moss used Elgot algebras [7] to describe solutions to recursive program schemes [31]. The composition $M(\Sigma M)^\infty$ has been previously known to be a monad and a cim in the vertical manner (that is, with respect to the module $M\Sigma M(\Sigma M)^\infty$) [21,39]. In contrast to those results, our proofs do not depend on the resumption monad being given by a final coalgebra.

Resumptions were used in semantics of concurrency in many different shapes and under different names. A domain-theoretic approach to resumptions was taken by Milner [32], Plotkin [40], and Papaspyrou [37]. Interaction trees, that is final coalgebras of functors of the shape $\mathcal{P}((\text{-}) \times O)^I$ on Set, where $\mathcal{P}$ is the powerset functor, were extensively used in Abramsky's interaction categories [2] (the existence of such final coalgebras was assured by employing Aczel's non-well-founded set theory [3]).

In programming, different forms of resumptions were independently rediscovered dozens of times, and used for flexible structuring of programs, though usually without much formal treatment. In the Lisp community, resumptions were dubbed 'engines' (Haynes and Friedman [25], Dybvig and Hieb [15]) or 'trampolined' programs (Ganz, Friedman, and Wand [18]), while in the Haskell libraries they can be found under the name 'free monad transformer' (since liftr from Section 4 is a monad morphism natural in M, the functor $M \mapsto M(\Sigma M)^\infty$ is a monad transformer in the sense of Moggi [33]). Different programming patterns that involve resumptions were discussed by Claessen [14], Kiselyov [28], Harrison [23], and the present authors [38]. Interleaving of data and effects in the algebraic context was also studied by Filinski and Støvring [17], and Atkey *et al.* [10].

In type theory, similar constructions were used to model interactive programs (Hancock and Setzer [22]), general recursion via guarded corecursion (Capretta [12]), or semantics of imperative languages (Nakata and Uustalu [36]).

Acknowledgements

We would like to thank Stefan Milius for his remarks on an early draft of this paper, and the anonymous reviewers for their detailed comments and helpful suggestions. The work reported here was partially supported by UK EPSRC-funded project *Reusability and Dependent Types* (EP/G034516/1).

References

[1] Samson Abramsky. Retracing some paths in process algebra. In Ugo Montanari and Vladimiro Sassone, editors, *CONCUR*, volume 1119 of *LNCS*, pages 1–17. Springer, 1996.

[2] Samson Abramsky, Simon Gay, and Rajagopal Nagarajan. Interaction categories and the foundations of typed concurrent programming. In *Proceedings of the 1994 Marktoberdorf Summer School on Deductive Program Design*, pages 35–113. Springer-Verlag, 1996.

[3] Peter Aczel. *Non-well-founded Sets*. Number 14 in Lecture Notes. Center for the Study of Language and Information, Stanford University, 1988.

[4] Peter Aczel, Jiří Adámek, Stefan Milius, and Jiří Velebil. Infinite trees and completely iterative theories: a coalgebraic view. *Theor. Comput. Sci.*, 300(1-3):1–45, 2003.

[5] Jiří Adámek, Stefan Milius, and Jiří Velebil. On rational monads and free iterative theories. *Electr. Notes Theor. Comput. Sci.*, 69:23–46, 2002.

[6] Jiří Adámek, Stefan Milius, and Jiří Velebil. Free iterative theories: A coalgebraic view. *Math. Struct. Comp. Sci.*, 13(2):259–320, 2003.

[7] Jiří Adámek, Stefan Milius, and Jiří Velebil. Elgot algebras. *Log. Meth. Comput. Sci.*, 2(5), 2006.

[8] Danel Ahman and Tarmo Uustalu. Update monads: Cointerpreting directed containers. In *Book of abstracts of the 19th Meeting of "Types for Proofs and Programs", TYPES 2013 (Toulouse, April 2013)*, pages 16–17, 2013.

[9] Thorsten Altenkirch, James Chapman, and Tarmo Uustalu. Monads need not be endofunctors. In *FOSSACS*, volume 6014 of *LNCS*, pages 297–311. Springer, 2010.

[10] Robert Atkey, Neil Ghani, Bart Jacobs, and Patricia Johann. Fibrational induction meets effects. In *FOSSACS*, volume 7213 of *LNCS*, pages 42–57. Springer, 2012.

[11] Jon Beck. Distributive laws. In *Seminar on Triples and Categorical Homology Theory*, volume 80 of *Lecture Notes in Mathematics*, pages 119–140. Springer Berlin / Heidelberg, 1969. 10.1007/BFb0083084.

[12] Venanzio Capretta. General recursion via coinductive types. *Log. Meth. Comput. Sci.*, 1(2), 2005.

[13] Pietro Cenciarelli and Eugenio Moggi. A syntactic approach to modularity in denotational semantics. In *Category Theory and Computer Science*, Amsterdam, The Netherlands, 1993.

[14] Koen Claessen. A poor man's concurrency monad. *J. Funct. Program.*, 9(3):313–323, 1999.

[15] R. Kent Dybvig and Robert Hieb. Engines from continuations. *Comput. Lang.*, 14(2):109–123, 1989.

[16] Calvin C. Elgot, Stephen L. Bloom, and Ralph Tindell. On the algebraic structure of rooted trees. *J. Comput. Syst. Sci.*, 16:362–399, 1978.

[17] Andrzej Filinski and Kristian Støvring. Inductive reasoning about effectful data types. In *ICFP*, pages 97–110. ACM, 2007.

[18] Steven E. Ganz, Daniel P. Friedman, and Mitchell Wand. Trampolined style. In *ICFP*, pages 18–27. ACM, 1999.

[19] Neil Ghani, Christoph Lüth, Federico De Marchi, and John Power. Algebras, coalgebras, monads and comonads. *Electr. Notes Theor. Comput. Sci.*, 44(1):128–145, 2001.

[20] Neil Ghani and Tarmo Uustalu. Coproducts of ideal monads. *Theoretical Informatics and Applications*, 38(4):321–342, 2004.

[21] Sergey Goncharov and Lutz Schröder. A coinductive calculus for asynchronous side-effecting processes. In *FCT*, volume 6914 of *LNCS*, pages 276–287. Springer, 2011.

[22] Peter Hancock and Anton Setzer. Interactive programs in dependent type theory. In *CSL*, volume 1862 of *LNCS*, pages 317–331. Springer, 2000.

[23] William L. Harrison. The essence of multitasking. In *AMAST*, volume 4019 of *LNCS*, pages 158–172. Springer, 2006.

[24] Ichiro Hasuo and Bart Jacobs. Traces for coalgebraic components. *Math. Struct. Comp. Sci.*, 21(2):267–320, 2011.

[25] Christopher T. Haynes and Daniel P. Friedman. Engines build process abstractions. In *LISP and Functional Programming*, pages 18–24, 1984.

[26] Martin Hyland, Gordon D. Plotkin, and John Power. Combining effects: Sum and tensor. *Theor. Comput. Sci.*, 357(1-3):70–99, 2006.

[27] Simon Peyton Jones. Tackling the awkward squad: Monadic input/output, concurrency, exceptions, and foreign-language calls in Haskell. In *Engineering theories of software construction*, pages 47–96. IOS Press, 2002.

[28] Oleg Kiselyov. Iteratees. In *FLOPS*, volume 7294 of *LNCS*, pages 166–181. Springer, 2012.

[29] Saunders Mac Lane. *Categories for the Working Mathematician.* Graduate Texts in Mathematics. Springer, 1998.

[30] Stefan Milius. Completely iterative algebras and completely iterative monads. *Inf. Comput.*, 196:1–41, 2005.

[31] Stefan Milius and Lawrence S. Moss. The category-theoretic solution of recursive program schemes. *Theor. Comput. Sci.*, 366(1-2):3–59, 2006.

[32] Robin Milner. Processes: A mathematical model of computing agents. In *Logic Colloquium '73*, Studies in logic and the foundations of mathematics, pages 157–173. North-Holland Pub. Co., 1975.

[33] Eugenio Moggi. An Abstract View of Programming Languages. Technical report, Edinburgh University, 1989.

[34] Eugenio Moggi. Notions of computation and monads. *Inf. Comput.*, 93(1):55–92, 1991.

[35] Lawrence S. Moss. Parametric corecursion. *Theor. Comput. Sci.*, 260(1-2):139–163, 2001.

[36] Keiko Nakata and Tarmo Uustalu. Resumptions, weak bisimilarity and big-step semantics for While with interactive I/O: An exercise in mixed induction-coinduction. In *SOS*, volume 32 of *EPTCS*, pages 57–75, 2010.

[37] Nikolaos S. Papaspyrou. A resumption monad transformer and its applications in the semantics of concurrency. In *Proceedings of the 3rd Panhellenic Logic Symposium, Anogia*, 2001.

[38] Maciej Piróg and Jeremy Gibbons. Tracing monadic computations and representing effects. In *MSFP*, volume 76 of *EPTCS*, pages 90–111, 2012.

[39] Maciej Piróg and Jeremy Gibbons. Monads for behaviour. *Electr. Notes Theor. Comput. Sci.*, 298:309–324, 2013. Mathematical Foundations of Programming Semantics.

[40] Gordon D. Plotkin. A powerdomain construction. *SIAM J. Comput.*, 5(3):452–487, 1976.

[41] Tarmo Uustalu. Generalizing substitution. *RAIRO—Theoretical Informatics and Applications*, 37:315–336, 10 2003.

Towards a Quantum Domain Theory: Order-enrichment and Fixpoints in W*-algebras

Mathys Rennela[1]

Institute for Computing and Information Sciences (iCIS)
Radboud Universiteit Nijmegen
Nijmegen, The Netherlands

Abstract

We discuss how the theory of operator algebras, also called operator theory, can be applied in quantum computer science. From a computer scientist point of view, we explain some fundamental results of operator theory and their relevance in the context of domain theory. In particular, we consider the category $\mathbf{W^*}$ of W*-algebras together with normal sub-unital maps, provide an order-enrichment for this category and exhibit a class of its endofunctors with a canonical fixpoint.

Keywords: W*-algebras, operator theory, domain theory, fixpoint theorem, quantum computation

Introduction

Our aim here is to use the theory of operator algebras to study the differences and similarities between probabilistic and quantum computations, by unveiling their domain-theoretic and topological structure. To our knowledge, the deep connection between the theory of operator algebras and domain theory was not fully exploited before. This might be due to the fact that the theory of operator algebras, mostly unknown to computer scientists, was developed way before the theory of domains.

Our main contribution is a connection between two different communities: the community of theoretical computer scientists, who use domain theory to study program language semantics (and logic), and the community of mathematicians and theoretical physicists, who use a special class of algebras called W*-algebras to study quantum mechanics.

[1] Email: mathys.rennela@gmail.com

Our main purpose was to pave the way to a study of quantum programming language semantics, where types are denoted by W*-algebras, terms are denoted by normal completely positive maps, recursive terms and weakest preconditions are denoted by fixed points (via $\mathbf{Dcpo}_{\perp !}$-enrichment), and recursive types are denoted by fixed points of endofunctors (via algebraical compactness).

We list here the main points of the paper, that will be detailed and explained later on:

- Positive maps can be ordered by a generalized version of the Löwner order [Low34], considered in the finite-dimensional case by Selinger in [Sel04]: for two positive maps f and g, $f \sqsubseteq g$ iff $(g - f)$ is positive [Definition 2.1]. We also consider completely positive maps in Section 4.
- Hom-sets of normal (positive) sub-unital maps between W*-algebras are directed-complete with this so-called Löwner order [Theorem 2.2].
- The category of $\mathbf{W}^*$ of W*-algebras together with normal sub-unital maps is order-enriched [Theorem 2.9].
- The notion of von Neumann functor is introduced to denote the locally continuous endofunctors on $\mathbf{W}^*$ which preserve multiplicative maps [Definition 3.1].
- For such functors, we provide a canonical way to construct a fixpoint, by showing that the category $\mathbf{W}^*$ is algebraic compact (for the class of von Neumann functors) [Theorem 3.6]. Our proof are in the lines of Smyth-Plotkin [SP82].

1 A short introduction to operator theory

In this section, we will introduce two structures, known as C*-algebras and W*-algebras. We refer the interested reader to [Tak02] for more details.

1.1 C*-algebras

A Banach space is a normed vector space where every Cauchy sequence converges. A Banach algebra is a linear associative algebra A over the complex numbers $\mathbb{C}$ with a norm $\|\cdot\|$ such that its norm $\|\cdot\|$ is submultiplicative (i.e. $\forall x, y \in A, \|xy\| \leq \|x\| \, \|y\|$) and turns A into a Banach space. A Banach algebra A is unital if it has a unit, i.e. if it has an element 1 such that $a1 = 1a = a$ holds for every $a \in A$ and $\|1\| = 1$.

A *-algebra is a linear associative algebra A over $\mathbb{C}$ with an operation $(-)^* : A \to A$ such that for all $x, y \in A$, the following equations holds: $(x^*)^* = x$, $(x + y)^* = (x^* + y^*)$, $(xy)^* = y^* x^*$ and $(\lambda x)^* = \overline{\lambda} x^* \quad (\lambda \in \mathbb{C})$.

A C*-algebra is a Banach *-algebra A such that $\|x^* x\| = \|x\|^2$ for all $x \in A$. This identity is sometimes called the C*-identity, and implies that every element x of a C*-algebra is such that $\|x\| = \|x^*\|$.

Consider now a unital C*-algebra A. An element $x \in A$ is self-adjoint if $x = x^*$. An element $x \in A$ is positive if it can be written in the form $x = y^* y$, where $y \in A$.

We write $A_{\mathrm{sa}} \hookrightarrow A$ (resp. $A^+ \hookrightarrow A$) for the subset of self-adjoint (resp. positive) elements of A.

For every C*-algebra, the subset of positive elements is a convex cone and thus induces a partial order structure on self-adjoint elements, see [Tak02, Definition 6.12]. That is to say, one can define a partial order on self-adjoint elements of a C*-algebra A as follows: $x \leq y$ if and only if $y - x \in A^+$.

From now on, we will consider the following kind of maps of C*-algebras. Let $f : A \to B$ be a linear map of C*-algebras:

P The map f is *positive* if it preserves positive elements and therefore restricts to a function $A^+ \to B^+$. A positive map $A \to \mathbb{C}$ will be called a state on A. It should be noted that positive maps of C*-algebras preserve the order on self-adjoint elements.

M The map f is *multiplicative* if $\forall x, y \in A, f(xy) = f(x)f(y)$;

I The map f is *involutive* if $\forall x \in A, f(x^*) = f(x)^*$;

U The map f is *unital* if it preserves the unit;

sU The map f is *sub-unital* if the inequality $0 \leq f(1) \leq 1$ holds;

cP For every C*-algebra A, one can easily define pointwise a C*-algebra $\mathcal{M}_n(A)$ from the set of n-by-n matrices whose entries are elements of A. The map f is *completely positive* if for every $n \in \mathbb{N}$, $\mathcal{M}_n(f) : \mathcal{M}_n(A) \to \mathcal{M}_n(B)$ defined for every matrix $[x_{i,j}]_{i,j\leq n} \in \mathcal{M}_n(A)$ by $\mathcal{M}_n(f)([x_{i,j}]_{i,j\leq n}) = [f(x_{i,j})]_{i,j\leq n}$ is positive.

For every Hilbert space H, the Banach space $\mathcal{B}(H)$ of bounded linear maps on H is a C*-algebra. The space $\mathcal{C}_0(X)$ of complex-valued continuous functions, that vanish at infinity, on a locally compact Hausdorff space X is a common example of commutative C*-algebra.

Self-adjoint and positive elements of $\mathcal{B}(H)$ can be defined alternatively through the inner product of H, as in the following standard theorem (see [Con07, II.2.12,VIII.3.8]):

Theorem 1.1 *Let H be a Hilbert space and $T \in \mathcal{B}(H)$. Then:*

(i) *T is self-adjoint if and only if $\forall x \in H, \langle Tx|x\rangle \in \mathbb{R}$;*

(ii) *T is positive if and only if T is self-adjoint and $\forall x \in H, \langle Tx|x\rangle \geq 0$.*

1.2 W-algebras*

We will denote by $\mathcal{B}(H)$ (resp. $\mathrm{Ef}(H)$) the collection of all bounded operators (resp. positive bounded operators below the unit) on a Hilbert space H. There are several standard topologies that one can define on a collection $\mathcal{B}(H)$ (see [Tak02] for an overview).

Definition 1.2 The operator norm $\|T\|$ is defined for every bounded operator T in $\mathcal{B}(H)$ by: $\|T\| = \sup\{\|T(x)\| \mid x \in H, \|x\| \leq 1\}$. The norm topology is the topology induced by the operator norm on $\mathcal{B}(H)$. A sequence of bounded operators (T_n) converges to a bounded operator T in this topology if and only if $\|T_n - T\| \underset{n\to\infty}{\longrightarrow} 0$.

The strong operator topology on $\mathcal{B}(H)$ is the topology of pointwise convergence in the norm of H: a net of bounded operators $(T_\lambda)_{\lambda\in\Lambda}$ converges to a bounded operator T in this topology if and only if $\|(T_\lambda - T)x\| \longrightarrow 0$ for each $x \in H$. In that case, T is said to be strongly continuous.

The weak operator topology on $\mathcal{B}(H)$ is the topology of pointwise weak convergence in the norm of H: a net of bounded operators $(T_\lambda)_{\lambda\in\Lambda}$ converges to a bounded operator T in this topology if and only if $\langle (T_\lambda - T)x|y\rangle \longrightarrow 0$ for $x, y \in H$. In that case, T is said to be weakly continuous.

It is known that, for an arbitrary Hilbert space H, the weak operator topology on $\mathcal{B}(H)$ is weaker than the strong operator topology on $\mathcal{B}(H)$, which is weaker than the norm topology on $\mathcal{B}(H)$. However, when H is finite-dimensional, the weak topology, the strong topology and the norm topology coincide. Moreover, for the strong and the weak operator topologies, the use of nets instead of sequences should not be considered trivial: it is known that, for an arbitrary Hilbert space H, the norm topology is first-countable whereas the other topologies are not necessarily first-countable, see [Tak02, Chapter II.2].

The commutant of $A \subset \mathcal{B}(H)$ is the set A' of all bounded operators that commutes with those of A: $A' = \{T \in \mathcal{B}(H) \mid \forall S \in A, TS = ST\}$. The bicommutant of A is the commutant of A' and will be denoted by A''.

The following theorem is a fundamental result in operator theory as it remarkably relates a topological property (being closed in two operator topologies) to an algebraic property (being its own bicommutant).

Theorem 1.3 (von Neumann bicommutant theorem) *Let A be a unital *-subalgebra of $\mathcal{B}(H)$ for some Hilbert space H. The following conditions are equivalent:*

(i) *$A = A''$.*

(ii) *A is closed in the weak topology of $\mathcal{B}(H)$.*

(iii) *A is closed in the strong topology of $\mathcal{B}(H)$.*

A W*-algebra (or von Neumann algebra) is a C*-algebra which satifies one (hence all) of the conditions of the von Neumann bicommutant theorem. The collections of bounded operators on Hilbert spaces are the most trivial examples of W*-algebras. The function space $L^\infty(X)$ for some standard measure space X and the space $\ell^\infty(\mathbb{N})$ of bounded sequences are common examples of commutative W*-algebras.

For every C*-algebra A, we denote by A' the dual space of A, i.e. the set of all linear maps $\phi : A \to \mathbb{C}$. It is known that a C*-algebra A is a W*-algebra if and only if there is a Banach space A_*, called pre-dual of A, such that $(A_*)' = A$, see [Sak71, Definition 1.1.2].

A positive map $\phi : A \to B$ between two C*-algebras is normal if every increasing net $(x_\lambda)_{\lambda\in\Lambda}$ in A^+ with least upper bound $\bigvee x_\lambda \in A^+$ is such that the net $(\phi(x_\lambda))_{\lambda\in\Lambda}$ is an increasing net in B^+ with least upper bound $\bigvee \phi(x_\lambda) = \phi(\bigvee x_\lambda)$.

1.3 Direct sums and tensors of W-algebras*

The direct sum of a family of C*-algebras $\{A_i\}_{i\in I}$ is defined as the C*-algebra

$$\bigoplus_{i\in I} A_i = \left\{ (a_i)_i \in \prod_{i\in I} A_i \mid \sup_{i\in I} \|a_i\| < \infty \right\}$$

where the operations are defined component-wise and with a norm defined by $\|(a_i)_{i\in I}\|_\infty = \sup \|a_i\|$.

The direct sum of a family of W*-algebras $\{A_i\}_{i\in I}$ is the W*-algebra $\bigoplus_{i\in I} A_i$ defined as the dual of the C*-algebras $\bigoplus_{i\in I} A_{i*}$, such that A_i is the dual of A_{i*}, seen as a C*-algebra.

The spatial tensor product $A\overline{\otimes}B$ of two W*-algebras A with universal normal representations $\pi_A : A \to \mathcal{B}(H)$ and $\pi_B : B \to \mathcal{B}(K)$ can be defined as the subalgebra of $\mathcal{B}(H\otimes K)$ generated by the operators $m\otimes n \in \mathcal{B}(H\otimes K)$ where $(m, n) \in A\times B$.

Proposition 1.4 *For a W*-algebra A, one has the following properties:*

- $A \oplus 0 = A = 0 \oplus A$*;*
- $A\overline{\otimes}0 = 0 = 0\overline{\otimes}A$*;*
- $A\overline{\otimes}\mathbb{C} = A = \mathbb{C}\overline{\otimes}A$*;*
- $A\overline{\otimes}(\bigoplus_{i\in I} B_i) = \bigoplus_{i\in I}(A\overline{\otimes}B_i)$ *for every family of W*-algebras* $\{A_i\}_{i\in I}$.

2 Rediscovering the domain-theoretic structure of W*-algebras

The W*-algebras together with the normal sub-unital maps (or NsU-maps), i.e. positive and Scott-continuous maps, give rise to a category $\mathbf{W}^*$, which is a subcategory of the category $\mathbf{C}^*$ of C*-algebras together with positive sub-unital maps.

In this section, after recalling some standard notion of domain theory, we will show that positive sub-unital maps can be ordered in such a way that the category $\mathbf{W}^*$ will be $\mathbf{Dcpo}_{\perp!}$-enriched. The decision of considering normal sub-unital maps instead of normal completely positive maps will be discussed in Section 4.1.

2.1 A short introduction to domain theory

A non-empty subset Δ of a poset P is called directed if every pair of elements of Δ has an upper bound in Δ. We denote it by $\Delta \subseteq_{dir} P$. A poset P is a directed-complete partial order (dcpo) if each directed subset has a least upper bound. A function $\phi : P \to Q$ between two posets P and Q is strict if $\phi(\perp_P$ $) =\perp_Q$, is monotonic if it preserves the order and Scott-continuous if it preserves directed joins. We denote by $\mathbf{Dcpo}_\perp$ (resp. $\mathbf{Dcpo}_{\perp!}$) the category with dcpos with bottoms as objects and Scott-continuous maps (resp. strict Scott-continuous maps) as morphisms.

2.2 A Löwner order on positive maps

Since positive elements are self-adjoint, one can define the following order on positive maps of C*-algebras.

Definition 2.1 [Löwner partial order] For positive maps $f, g : A \to B$ between C*-algebras A and B, we define pointwise the following partial order $\sqsubseteq$, which turns out to be an infinite-dimensional generalization of the Löwner partial order [Low34] for positive maps: $f \sqsubseteq g$ if and only if $\forall x \in A^+, f(x) \leq g(x)$ if and only if $\forall x \in A^+, (g - f)(x) \in B^+$ (i.e. $g - f$ is positive).

One might ask if, for arbitrary C*-algebras A and B, the poset $(\mathbf{C}^*(A, B), \sqsubseteq)$ is directed-complete. The answer turns out to be no, as shown by our following counter-example:

Let us consider the C*-algebra $C([0,1]) := \{f : [0,1] \to \mathbb{C} \mid f \text{ continuous}\}$.

The hom-set $\mathbf{C}^*(\mathbb{C}, C([0,1]))$ is isomorphic to $C([0,1])$ if one considers the functions $F : \mathbf{C}^*(\mathbb{C}, C([0,1])) \to C([0,1])$ and $G : C([0,1]) \to \mathbf{C}^*(\mathbb{C}, C([0,1]))$ respectively defined by $F(f) = f(1)$ and $G(g) = \lambda\alpha \in \mathbb{C}.\alpha \cdot g$.

We define an increasing chain $(f_n)_{n \geq 0}$ of $C([0,1])$ define for every $n \in \mathbb{N}$ by

$$f_n(x) = \begin{cases} 0 & \text{if } 0 \leq x < \frac{1}{2} \\ (x - \frac{1}{2})2^{n+1} & \text{if } \frac{1}{2} \leq x \leq \frac{1}{2} + 2^{-(n+1)} \\ 1 & \text{if } \frac{1}{2} + 2^{-(n+1)} < x \leq 1 \end{cases}$$

Suppose that there is a least upper bound ϕ in $C([0,1])$ for this chain. Then, $\phi(x) = 0$ if $x < \frac{1}{2}$. Moreover, $\lim_{n\to\infty} \left(\frac{1}{2} + 2^{-(n+1)}\right) = \frac{1}{2}$ implies that $\phi(x) = 1$ if $x > \frac{1}{2}$. It follows that $\phi(x) \in \{0,1\}$ if $x \neq \frac{1}{2}$.

By the Intermediate Value Theorem, the continuity of the function ϕ on the interval $[0,1]$ implies that there is a $c \in [0,1]$ such that $\phi(c) = \frac{1}{2}$. From $\phi(c) \notin \{0,1\}$, we obtain that $c = \frac{1}{2}$. That is to say $\phi(\frac{1}{2}) = \frac{1}{2}$, which is absurd since $f_n(\frac{1}{2}) = 0$ for every $n \in \mathbb{N}$.

It follows that there is no least upper bound for this chain in $C([0,1])$ and therefore $C([0,1])$ is not chain-complete. However:

Theorem 2.2 *For W*-algebras A and B, the poset $(\mathbf{W}^*(A, B), \sqsubseteq)$ is directed-complete.*

The proof of this theorem will be postponed until after the following lemmas.

Lemma 2.3 ([RD66], Corollary 1) *Let $f \in \mathbf{C}^*(A, B)$ and $x \in A^+$. Then, $f(x) \leq \|x\| \cdot 1$. Therefore, $\|f(x)\| \leq \|x\|$.*

The following result is known in physics as Vigier's theorem [Vig46]. A weaker version of this theorem can be found in [Sel04]. It is important in this context because it establishes the link between limits in topology and joins in order theory.

Lemma 2.4 *Let H be a Hilbert space. Let $(T_\lambda)_{\lambda\in\Lambda}$ be an increasing net of* $\mathrm{Ef}(H)$. *Then the least upper bound $\bigvee T_\lambda$ exists in* $\mathrm{Ef}(H)$ *and is the limit of the net $(T_\lambda)_{\lambda\in\Lambda}$ in the strong topology.*

Proof. For any operator $U \in \mathcal{B}(H)$, the inner product $\langle Ux|x\rangle$ is real if and only if U is self-adjoint (by Theorem 1.1). Thus, for each $x \in H$, the net $(\langle T_\lambda x|x\rangle)_{\lambda\in\Lambda}$ of real numbers is increasing, bounded by $\|x\|^2$ and thus convergent to a limit $\lim_\lambda \langle T_\lambda x|x\rangle$ since $\mathbb{R}$ is bounded-complete.

By polarization on norms, $\langle T_\lambda x|y\rangle = \frac{1}{2}(\langle T_\lambda(x+y)|(x+y)\rangle - \langle T_\lambda x|x\rangle - \langle T_\lambda y|y\rangle)$ for any $\lambda \in \Lambda$. Then, for all $x, y \in H$, the limit $\lim_\lambda \langle T_\lambda x|y\rangle$ exists and thus we can define pointwise an operator $T \in \mathrm{Ef}(H)$ by $\langle Tx|y\rangle = \lim_\lambda \langle T_\lambda x|y\rangle$ for $x, y \in H$.

Indeed, T is the limit of the net $(T_\lambda)_{\lambda\in\Lambda}$ in the weak topology, and therefore in the strong topology since a bounded net of positive operators converges strongly whenever it converges weakly (see [Bla06, I.3.2.8]).

Moreover, T is an upper bound for the net $(T_\lambda)_{\lambda\in\Lambda}$ since $T_\lambda \leq T$ for every $\lambda \in \Lambda$. By Theorem 1.1, if there is a self-adjoint operator $S \in B(H)$ such that $T_\lambda \leq S$ for every $\lambda \in \Lambda$, then $\langle T_\lambda x|x\rangle \leq \langle Sx|x\rangle$ for every $\lambda \in \Lambda$. Thus, $\langle Tx|x\rangle = \lim_\lambda \langle T_\lambda x|x\rangle \leq \langle Sx|x\rangle$. Then, $\langle (S-T)x|x\rangle \geq 0$ for every $x \in H$. By Theorem 1.1, $S - T$ positive and thus $T \leq S$. It follows that T is the least upper bound of $(T_\lambda)_{\lambda\in\Lambda}$. □

Corollary 2.5 *For every W*-algebra A, the poset $[0,1]_A$ is directed-complete.*

Proof. Let A be a W*-algebra. By definition, A is a strongly closed subalgebra of $\mathcal{B}(H)$, for some Hilbert space H. Then, let $(T_\lambda)_{\lambda\in\Lambda}$ be an increasing net in $[0,1]_A \subseteq \mathrm{Ef}(H)$. By Lemma 2.4, $(T_\lambda)_{\lambda\in\Lambda}$ converges strongly to $\bigvee T_\lambda \in \mathrm{Ef}(H)$. It follows that $\bigvee T_\lambda \in [0,1]_A$ because $[0,1]_A$ is strongly closed. Thus, $[0,1]_A$ is directed-complete. □

This corollary constitutes a crucial step in the proof of Theorem 2.2, as it unveils a link between the topological properties and the order-theoretic properties of W*-algebras.

Lemma 2.6 *Any positive map $f : A \to B$ between C*-algebras is completely determined and defined by its action on* $[0,1]_A$.

Proof. A positive map of C*-algebras $f : A \to B$ restrict by definition to a map $f : A^+ \to B^+$. Since f preserves the order $\leq$ on positive elements, it restricts to $[0,1]_A \to [0,1]_B$:

Let $x \in A^+ \setminus \{0\}$. From $x \leq \|x\| 1$, we can see that $\frac{1}{\|x\|}x \in [0,1]_A$ and thus $f(\frac{1}{\|x\|}x) \in [0,1]_B$. Moreover, $f(x) = \|x\| f(\frac{1}{\|x\|}x)$. This statement can be extended to every element in A since each $y \in A$ is a linear combination of four positive elements (see [Bla06, II.3.1.2]), determining $f(y) \in B$. □

We can now show that the poset $(\mathbf{W}^*(A,B), \sqsubseteq)$ is directed-complete for every pair A and B of W*-algebras.

Proof. [Proof of Theorem 2.2] Let A and B be two W*-algebras. By Corollary 2.5, $[0,1]_A$ and $[0,1]_B$ are directed-complete.

We now consider an increasing net $(f_\lambda)_{\lambda\in\Lambda}$ of NsU-maps from A to B, increasing in the Löwner order. Then, for every $x \in A^+$, there is an increasing net $(f_\lambda(x))_{\lambda\in\Lambda}$ bounded by $\|x\| \cdot 1$ (by Lemma 2.3).

Moreover, for every non-zero element $x \in A^+$, from the fact that $[0,1]_B$ is directed-complete, we obtain that the increasing net $(f_\lambda(\frac{x}{\|x\|}))_{\lambda\in\Lambda}$ has a join $\bigvee f_\lambda(\frac{x}{\|x\|})$ in $[0,1]_B$ and thus we can define pointwise the following upper bound $f : [0,1]_A \to [0,1]_B$ for the increasing net $(f'_\lambda)_{\lambda\in\Lambda}$ of NsU-maps from $[0,1]_A$ to $[0,1]_B$ such that, for every $\lambda \in \Lambda$, $f'_\lambda(x) = f_\lambda(\frac{x}{\|x\|})(x \neq 0)$: $f(\frac{x}{\|x\|}) = \bigvee f_\lambda(\frac{x}{\|x\|}) \quad (x \in A^+ \setminus \{0\})$

This upper bound f is a positive sub-unital map by construction and can be extended to an upper bound $f : A \to B$ for the increasing net $(f_\lambda)_{\lambda\in\Lambda}$: for every nonzero $x \in A^+$, the increasing sequence $(f_\lambda(x))_{\lambda\in\Lambda} = (\|x\|\, f_\lambda(\frac{x}{\|x\|}))_{\lambda\in\Lambda}$ has a join $\bigvee f_\lambda(x) = \|x\| \bigvee f_\lambda(\frac{x}{\|x\|})$ in B^+ and thus one can define pointwise an upper bound $f : A \to B$ for $(f_\lambda)_{\lambda\in\Lambda}$ by $f(x) = \bigvee f_\lambda(x)$ for every $x \in A^+$.

We now need to prove that the map f is normal, by exchange of joins.
Let $(x_\gamma)_{\gamma\in\Gamma}$ be an increasing bounded net in A^+ with join $\bigvee_\gamma x_\gamma$. For every $\gamma' \in \Gamma$, we observe that $x_{\gamma'} \leq \bigvee_\gamma x_\gamma$ and thus, $f(x_\gamma) \leq f(\bigvee_\gamma x_\gamma)$ (recall that f preserves the order). As seen earlier, since $[0,1]_B$ is directed-complete, the increasing net $(f(x_\gamma))_{\gamma\in\Gamma}$, which is equal by definition to the increasing net $(\bigvee_\lambda(f_\lambda(x_\gamma)))_{\gamma\in\Gamma}$, has a join in B^+ defined by $\bigvee_\gamma f(x_\gamma) = \bigvee_{\gamma\in\Gamma, x_\gamma\neq 0} \|x_\gamma\|\, f(\frac{1}{\|x_\gamma\|} x_\gamma)$ if there is a $\gamma'' \in \Gamma$ such that $x_{\gamma''} \neq 0$ and by $\bigvee_\gamma f(x_\gamma) = 0$ otherwise. It follows that $\bigvee_\gamma f(x_\gamma) \leq f(\bigvee_\gamma x_\gamma)$.

We have to prove now that $f(\bigvee_\gamma x_\gamma) \leq \bigvee_\gamma f(x_\gamma)$. Since each map f_λ $(\lambda \in \Lambda)$ is normal, we obtain that $f(\bigvee_\gamma x_\gamma) = \bigvee_\lambda(f_\lambda(\bigvee_\gamma x_\gamma)) = \bigvee_\lambda(\bigvee_\gamma(f_\lambda(x_\gamma)))$. Moreover, for $\gamma' \in \Gamma$ and $\lambda' \in \Lambda$, $f_{\lambda'}(x_{\gamma'}) \leq \bigvee_\lambda f_\lambda(x_{\gamma'}) \leq \bigvee_\gamma(\bigvee_\lambda f_\lambda(x_\gamma))$. Then, $\bigvee_\gamma f_{\lambda'}(x_\gamma) \leq \bigvee_\gamma(\bigvee_\lambda f_\lambda(x_\gamma))$ and thus $\bigvee_\lambda(\bigvee_\gamma f_\lambda(x_\gamma)) \leq \bigvee_\gamma(\bigvee_\lambda f_\lambda(x_\gamma))$. It follows that $f(\bigvee_\gamma x_\gamma) = \bigvee_\lambda(\bigvee_\gamma f_\lambda(x_\gamma)) \leq \bigvee_\gamma(\bigvee_\lambda f_\lambda(x_\gamma)) = \bigvee_\gamma f(x_\gamma)$.

Let $g \in \mathbf{W}^*(A,B)$ be an upper bound for the increasing net $(f_\lambda)_{\lambda\in\Lambda}$. For $\lambda' \in \Lambda$ and $x \in A^+$, $f_{\lambda'}(x) \leq g(x)$. Then, $\forall x \in A^+, f(x) = \bigvee f_\lambda(x) \leq g(x)$, i.e. $f \sqsubseteq g$. It follows that f is the join of $(f_\lambda)_{\lambda\in\Lambda}$. □

This theorem generalizes the fact that the effects of a W*-algebra A (i.e. the positive unital maps from $\mathbb{C}^2$ to A) form a directed-complete poset [Tak02, III.3.13-16]. Moreover, it turns out that Theorem 2.2 can be slightly generalized to the following theorem, with a similar proof.

Theorem 2.7 *Let A and B be two C*-algebras.*
If $[0,1]_B$ is directed-complete, then the poset $(\mathbf{C}^*(A,B), \sqsubseteq)$ is directed-complete.

2.3 $\mathbf{Dcpo}_{\perp!}$*-enrichment for W*-algebras*

In this section, we will provide a $\mathbf{Dcpo}_{\perp!}$-enrichment for the category $\mathbf{W}^*$ and discuss the domain-theoretic properties of C*-algebras.

Definition 2.8 Let $\mathbf{C}$ be a category for which every hom-set is equipped with the structure of a poset. $\mathbf{C}$ is said to be $\mathbf{Dcpo}_{\perp!}$-enriched if its hom-sets are dcpos with bottom and if the composition of homomorphisms is strict and Scott-continuous, i.e. the pre-composition $(-) \circ f : \mathbf{C}(B, C) \to \mathbf{C}(A, C)$ and the post-composition $h \circ (-) : \mathbf{C}(A, B) \to \mathbf{C}(A, C)$ are strict and Scott-continuous for homomorphisms $f : A \to B$ and $h : B \to C$.

Theorem 2.9 *The category $\mathbf{W}^*$ is a $\mathbf{Dcpo}_{\perp!}$-enriched category.*

Proof.

For every pair (A, B) of W*-algebras, $\mathbf{W}^*(A, B)$ together with the Löwner order is a dcpo with zero map as bottom, and therefore $\mathbf{W}^*(A, B) \in \mathbf{Dcpo}_{\perp!}$.

In particular, for every W*-algebra A, $\mathbf{W}^*(A, A) \in \mathbf{Dcpo}_{\perp!}$. We consider now for every W*-algebra A a map $I_A : 1_A = \{\perp_A\} \to \mathbf{W}^*(A, A)$ such that $I_A(\perp) \in \mathbf{W}^*(A, A)$ is the identity map on A. The map I_A is clearly strict Scott-continuous for every W*-algebra A.

Then, what need to be proved is that, given three W*-algebras A, B, C, the composition $\circ_{A,B,C} : \mathbf{W}^*(B, C) \times \mathbf{W}^*(A, B) \to \mathbf{W}^*(A, C)$ is Scott-continuous (the strictness of the composition can be easily verified).

We now consider a NsU-map $f : A \to B$ and the increasing net $(g_\lambda)_{\lambda \in \Lambda}$ in $\mathbf{W}^*(B, C)$, with join $\bigvee_\lambda g_\lambda \in \mathbf{W}^*(B, C)$. One can define an upper bound pointwise by $u(x) = ((\bigvee_\lambda g_\lambda) \circ f)(x)$ for the increasing net $(g_\lambda \circ f)_{\lambda \in \Lambda}$ in $\mathbf{W}^*(A, C)$. It is easy to check that u is a join for the increasing net $(g_\lambda \circ f)_{\lambda \in \Lambda}$: for every upper bound $v \in \mathbf{W}^*(A, C)$ of the increasing net $(g_\lambda \circ f)_{\lambda \in \Lambda}$, we have that $\forall \lambda \in \Lambda, g_\lambda \circ f \sqsubseteq v$, i.e. $\forall \lambda \in \Lambda, \forall x \in A^+, g_\lambda(f(x)) \leq v(x)$ and thus $\forall x \in A^+, u(x) = ((\bigvee_\lambda g_\lambda) \circ f)(x) = (\bigvee_\lambda g_\lambda)(f(x)) \leq v(x)$, which implies that $u \sqsubseteq v$. It follows that the pre-composition is Scott-continuous and, similarly, the post-composition is Scott-continuous. □

In operator theory, a C*-algebra is monotone-complete (or monotone-closed) if it is directed-complete for bounded increasing nets of positive elements. The notion of monotone-completeness goes back at least to Dixmier [Dix51] and Kadison [Kad55] but, to our knowledge, it is the first time that the notion of monotone-completeness is explicitly related to the notion of directed-completeness. The interested reader will find in Appendix A a more detailed correspondence between operator theory and order theory.

It is natural to ask if all monotone-complete C*-algebras are W*-algebras. Dixmier proved that every W*-algebra is a monotone-complete C*-algebra and that the converse is not true [Dix51]. For an example of a subclass of monotone-complete C*-algebras which are not W*-algebras, we refer the reader to a recent work by Saitô and Wright [SW12].

3 A fixpoint theorem for endofunctors on W*-algebras

In this section, we will show that it is possible to exhibit a fixpoint for a specific class of endofunctors on W*-algebras, that we will define later.

For this purpose, we first observe that the one-element W*-algebra $\mathbf{0} = \{0\}$ is the zero object, i.e. initial and terminal object, of the category $\mathbf{W}^*$: a NsU-map

$f : A \to \mathbf{0}$ must be defined by $f(x) = 0$; a NsU-map $g : \mathbf{0} \to A$ is linear and thus $g(0) = 0_A$ must holds.

We will now consider the following class of functors and then show that they admit a canonical fixpoint.

Definition 3.1 A von Neumann functor is a locally continuous endofunctor on $\mathbf{W}^*$ which preserves multiplicative maps.

Example 3.2 The identity functor and the constant functors on $\mathbf{W}^*$ are locally continuous and so does any (co)product of locally continuous functors. It is also clear that all those functors preserve multiplicative maps.

Secondly, our proofs and structures will use the notion of embedding-projection pairs, that we will define as follows.

Definition 3.3 An embedding-projection pair is a pair of arrows $\langle e, p\rangle \in \mathbf{C}(X, Y) \times \mathbf{C}(Y, X)$ in a $\mathbf{Dcpo}_{\perp}$-enriched category $\mathbf{C}$ such that $p \circ e = \mathrm{id}_X$ and $e \circ p \leq \mathrm{id}_Y$.

For two pairs $\langle e_1, p_1\rangle$, $\langle e_2, p_2\rangle$, it can be shown that $e_1 \leq e_2$ iff $p_2 \leq p_1$, which means that one component of the pair can uniquely determine the other one. We denote by e^P the projection corresponding to a given embedding e and p^E the embedding corresponding to a given projection p. It should be noted that $(e \circ f)^P = f^P \circ e^P$, $(p \circ q)^E = q^E \circ p^E$ and $\mathrm{id}^P = \mathrm{id}^E = \mathrm{id}$.

The category $\mathbf{C}^E$ of embeddings of a $\mathbf{Dcpo}_{\perp!}$-enriched category $\mathbf{C}$ is the subcategory of $\mathbf{C}$ that has objects of $\mathbf{C}$ has objects and embeddings as arrows. It should be noted that this category is itself a $\mathbf{Dcpo}_{\perp!}$-enriched category. Dually, one can define the category $\mathbf{C}^P = (\mathbf{C}^E)^{op}$ of projections of a $\mathbf{Dcpo}_{\perp!}$-enriched category $\mathbf{C}$.

An endofunctor F on a $\mathbf{Dcpo}_{\perp!}$-enriched category $\mathbf{C}$ is locally continuous (resp. locally monotone) if $F_{X,Y} : \mathbf{C}(X, Y) \to \mathbf{C}(FX, FY)$ is Scott-continuous (resp. monotone).

We can now consider the following setting. Let $F : \mathbf{W}^* \to \mathbf{W}^*$ be a von Neumann functor. Consider the ω-chain $\Delta = (D_n, \alpha_n)_n$ for which $D_0 = \mathbf{0}$, the embedding $\alpha_0 : D_0 \to FD_0$ is the unique NsU-map from D_0 to FD_0, and the equalities $\alpha_{n+1} = F\alpha_n$ and $D_{n+1} = FD_n$ hold for every $n \geq 0$.

Since the endofunctor F is locally monotone, if for some $n \in \mathbb{N}$ there is an embedding-projection pair $\langle \alpha_n^E, \alpha_n^P\rangle$, the pair $\langle \alpha_{n+1}^E, \alpha_{n+1}^P\rangle = \langle F\alpha_n^E, F\alpha_n^P\rangle$ is also an embedding-projection pair. It follows that the ω-chain Δ is well-defined.

Definition 3.4 Consider the collection $D = \{(x_n)_n \in \bigoplus_n D_n \mid \forall n \geq 0, \alpha_n^p(x_{n+1}) = x_n\}$. It forms a poset together with the order $\leq_D$ defined by $(x_n)_n \leq_D (y_n)_n \equiv \forall n \geq 0, x_n \leq_{D_n} y_n$. Moreover, the collection D can be seen as a *-algebra:

From the fact that the projection $\alpha_0^P : D_1 \to D_0 = \mathbf{0}$ is trivially a multiplicative map (which maps everything to the unique element of $\mathbf{0}$) and that the functor F preserves multiplicative maps, we can conclude that for every $n \geq 0$, the projection $\alpha_{n+1}^P = F\alpha_n^P = \cdots = F^{n+1}\alpha_0^P$ is a NMIsU-map. In fact, it can be shown that the embeddings $\alpha_n : D_n \to D_{n+1}$ are NMIsU-maps as well, by the same reasoning. Moreover, it should also be noted that embeddings and projections are strict, i.e.

preserves 0.

Considering these facts, one can verify that the collection D forms a *-algebra with operations defined component-wise on the family of W*-algebras $\{D_n\}_{n\geq 0}$.

- The unit is defined by $(1_n)_n = (1_{D_n})_n$. From the fact that the embeddings α_n^E are NsU-maps, i.e. $\alpha_n^E(1) \leq 1$ for every $n \in \mathbb{N}$, we deduce that $1 = (\alpha_n^P \circ \alpha_n^E)(1) \leq \alpha_n^P(1)$ for every $n \in \mathbb{N}$ (recall that the projection α_n^P is an order-preserving map). Hence, every projection α_n^P is a unital map and thus $\alpha_n^P(1_{n+1}) = \alpha_n^P(1_n)$ holds for every $n \in \mathbb{N}$.
- The addition is defined by $(x_n)_n +_D (y_n)_n = (x_n +_{D_n} y_n)_n$ for all $(x_n)_n, (y_n)_n \in D$. This operation is well-defined: since the projections α_n^P are linear maps, one can observe that $\alpha_n^P(0_{n+1}) = 0_n$ and $\alpha_n^P(x_{n+1} + y_{n+1}) = \alpha_n^P(x_{n+1}) + \alpha_n^P(y_{n+1}) = x_n + y_n$ for every $n \in \mathbb{N}$. Moreover, by the triangle inequality, $\sup_n \|x_n + y_n\| \leq \sup_n \|x_n\| + \sup_n \|y_n\| < \infty$.
- The scalar multiplication is defined by $\lambda(x_n)_n = (\lambda x_n)_n$ for every $\lambda \in \mathbb{C}$ and and every $(x_n)_n \in D$. It is easy to verify that this operation is well-defined: $\alpha_n^P(\lambda x_{n+1}) = \lambda\alpha_n^P(x_{n+1}) = \lambda x_n$ for every $n \in \mathbb{N}$ (by linearity of α_n^P) and $\|(\lambda x_n)_n\|_\infty = \lambda \|(x_n)_n\|_\infty < \infty$.
- The multiplication is defined by $(x_n)_n \cdot_D (y_n)_n = (x_n \cdot_{D_n} y_n)_n$ for all $(x_n)_n, (y_n)_n \in D$. This operation is well-defined since the projections α_n^P are multiplicatives: $\alpha_n^P(x_{n+1} \cdot y_{n+1}) = \alpha_n^P(x_{n+1}) \cdot \alpha_n^P(y_{n+1}) = x_n \cdot y_n$.

  Moreover, the Banach spaces D_n are submultiplicatives and thus the following inequality holds: $\sup_n \|x_n \cdot y_n\| \leq (\sup_n \|x_n\|)(\sup_n \|y_n\|) < \infty$.
- The involution is defined by $((x_n)_n)^* = (x_n^*)_n$ for every $(x_n)_n \in D$.
  This operation is well-defined since the projections α_n^P are involutives: $\alpha_n^P(x_{n+1}^*) = \alpha_n^P(x_{n+1})^* = x_n^*$. Moreover, as a direct consequence of the C*-identity of the C*-algebras D_n, the following equality holds: $\sup_n \|x_n^*\| = \sup_n \|x_n\| < \infty$.

Proposition 3.5 *The *-algebra D forms a C*-algebra.*

Proof. Since every D_n is a C*-algebra, the C*-identity holds for D as well: $\|(x_n)_n^*(x_n)_n\|_\infty = \|(x_n^* x_n)_n\|_\infty = \sup_n \|x_n^* x_n\| = \sup_n \|x_n\|^2 = (\|(x_n)_n\|_\infty)^2$

Consider a Cauchy sequence $((x_{m,n})_n)_m \in D$. It follows that for every $n' \in \mathbb{N}$, the following proposition holds: $\forall \varepsilon > 0, \exists M \in \mathbb{N}, \forall m, m' \geq M, \left\|x_{m,n'} - x_{m',n'}\right\| \leq \sup_n \left\|x_{m,n} - x_{m',n}\right\| < \varepsilon$.

We are required to prove that the Cauchy sequence $((x_{m,n})_n)_m \in D$ converges, by constructing its limit that we will denote by $(l_n)_n$. We can first deduce that, for every $n \in \mathbb{N}$, the sequence $(x_{m,n})_m$ in D_n is a Cauchy sequence, which converges to a limit $l_n \in D_n$ since D_n is a W*-algebra (and therefore a Banach space). Then, we obtain a sequence $(l_n)_n \in \prod_{n\in\mathbb{N}} D_n$. Since PsU-maps (and therefore NMIsU-maps) are contractive, we can conclude that $(l_n)_n \in D$ by the following arguments.

Let $n \in \mathbb{N}$ and $\varepsilon > 0$. From the fact that the inequality $\|x_{m,n+1} - l_{n+1}\| < \varepsilon$ holds, we deduce that $\left\|\alpha_n^P(x_{m,n+1}) - \alpha_n^P(l_{n+1})\right\| = \left\|\alpha_n^P(x_{m,n+1} - l_{n+1})\right\| \leq \|x_{m,n+1} - l_{n+1}\| < \varepsilon$ and thus, $l_n = \lim_{m\to\infty} x_{m,n} = \lim_{m\to\infty} \alpha_n^P(x_{m,n+1}) =$

$\alpha_n^P(l_{n+1})$. Moreover, $\|(l_n)_n\|_\infty = \sup_n \|l_n\| < \infty$ since l_n is the limit of a Cauchy sequence (recall that every Cauchy sequence is bounded). □

This leads us to the following new result.

Theorem 3.6 *The category* $\mathbf{W}^*$ *is algebraically compact for the class of von Neumann functors, i.e. every von Neumann functor* F *admits a canonical fixpoint and there is an isomorphism between the initial* F*-algebra and the inverse of the final* F*-coalgebra.*

The proof of this theorem involves the following notions.

Definition 3.7 An ω-chain in a category $\mathbf{C}$ is a sequence of the form $\Delta = D_0 \xrightarrow{\alpha_0} D_1 \xrightarrow{\alpha_1} \cdots$

Given an object D in a category $\mathbf{C}$, a cocone $\mu : \Delta \to D$ for the ω-chain Δ is a sequence of arrows $\mu_n : D_n \to D$ such that the equality $\mu_n = \mu_{n+1} \circ \alpha_n$ holds for every $n \geq 0$.

A colimit (or colimiting cocone) of the ω-chain Δ is an initial cocone from Δ to D, i.e. it has the following universal property: for every cocone $\mu' : \Delta \to D'$, there exists a unique map $f : D \to D'$ such that the equality $f \circ \mu_n = \mu'_n$ holds for every $n \geq 0$.

Dually, we will consider ω^{op}-chains $\Delta^{\mathrm{op}} = D_0 \xleftarrow{\beta_0} D_1 \leftarrow \cdots$ in a category, cones $\gamma : \Delta^{\mathrm{op}} \leftarrow D$ and limits (or limiting cones) for an ω^{op}-chain Δ^{op}.

Proof. [Proof of Theorem 3.6] Together with its previously defined order, the C*-algebra D is monotone-complete, since all the W*-algebras D_n are monotone-complete. Moreover, a separating set of normal states can be defined for D, on the separating set of normal states of the W*-algebras D_n. We can then conclude by Theorem A.4 that D forms a W*-algebra and we are now required to prove that it can be turn into a colimit for the diagram Δ.

We now define a cocone $\Delta \to D$ which arrows are embeddings $\mu_n : D_n \to D(n \geq 0)$ defined by:

- $\mu_n(x) = ((\alpha_0^P \circ \cdots \circ \alpha_{n-1}^P)(x), (\alpha_1^P \circ \cdots \circ \alpha_{n-1}^P)(x), \ldots, \alpha_{n-1}^P(x), x, \alpha_n^E(x), (\alpha_{n+1}^E \circ \alpha_n^E)(x), \ldots)$ for every $x \in D_n$. It is easy to check that for every $m \in \mathbb{N}$ if we define a sequence $(y_n)_n = \mu_m(x)$, then the equation $\alpha_n^P(y_{n+1}) = y_n$ holds for every $n \in \mathbb{N}$. Moreover, as a positive map, the embedding μ_n is contractive and thus $\|\mu_n(x)\|_\infty \leq \sup_n \|x\| = \|x\|_\infty < \infty$.
- $\mu_n^P((x_n)_n) = x_n$ for every $(x_n)_n \in D$.

Indeed, it is easy to check that those projections μ_n^P are NMIU-maps by construction and that the corresponding embeddings μ_n^E are NMIsU-maps by construction.

Then, one can see that $\mu_n^P(\mu_n^E(x)) = x$ for every $x \in D_n$ and that $\mu_n^E(\mu_n^P((x_n)_n)) \leq (x_n)_n$ for every $(x_n)_n \in D$ since:

(i) For every $m \in \mathbb{N}$ such that $0 \leq m < n$, $(\alpha_m^P \circ \cdots \circ \alpha_{n-1}^P)^E(x_m) = (\alpha_{n-1}^E \circ \cdots \circ \alpha_m^E)(x_m) = x_n$, which implies that $x_m = (\alpha_m^P \circ \cdots \circ \alpha_{n-1}^P)((\alpha_m^P \circ \cdots \circ \alpha_{n-1}^P)^E(x_m)) = (\alpha_m^P \circ \cdots \circ \alpha_{n-1}^P)(x_n)$ and thus $((\mu_n^E \circ \mu_n^P)((x_n)_n))_m = x_m$ for every $m \leq n$;

(ii) From the fact that $\alpha_n^E \circ \alpha_n^P \leq \mathrm{id}_{D_{n+1}}$ for every $n \in \mathbb{N}$, we obtain that $\alpha_n^E(x_n) = \alpha_n^E(\alpha_n^P(x_{n+1})) \leq x_{n+1}$ for every $n \in \mathbb{N}$ and thus by induction, $\alpha_m^E \circ \cdots \circ \alpha_{n+1}^E \circ \alpha_n^E(x_n) \leq \alpha_m^E \circ \cdots \circ \alpha_{n+1}^E(x_{n+1}) \leq \cdots \leq x_m$ for every $m \geq n$. Thus, $((\mu_n^E \circ \mu_n^P)((x_n)_n))_m \leq x_m$ for every $m \geq n$.

Moreover, for every $n \geq 0$, we observe that $\mu_n = \mu_{n+1} \circ \alpha_n$ since $\alpha_n^P(\mu_{n+1}^P((x_n)_n)) = \alpha_n^P(x_{n+1}) = x_n = \mu_n^P((x_n)_n)$ and thus $\mu_n^P = \alpha_n^P \circ \mu_{n+1}^P = (\mu_{n+1} \circ \alpha_n)^P$.

As stated in [RT92], the fact that F is locally continuous implies that $\bigvee_n(\mu_n \circ \mu_n^P) = \mathrm{id}_D$, and thus $\mu : \Delta \to D$ is a colimiting cocone for Δ by [SP82, Theorem 2,Proposition A]. Dually, one can show that $\mu^P : D \to \Delta^P$ is a limiting cone for Δ^P, the cone of projections $D_0 \xleftarrow{\alpha_0^P} D_1 \xleftarrow{\alpha_1^P} \cdots$.

Since F is locally continuous, it is therefore locally monotone. It follows that :

- For every $n \in \mathbb{N}$, $\langle F\mu_n, F\mu_n^P \rangle$ is an embedding-projection pair;
- The chain $\{F\mu_n \circ F\mu_n^P\}_n$ is increasing with join $\bigvee_n(F\mu_n \circ F\mu_n^P) = \mathrm{id}_{FD}$.

From [SP82, Theorem 2] again, we conclude that $F\mu : F\Delta \to FD$ is a colimiting cocone (and dually $F\mu^P : FD \to F\Delta^P$ is a limiting cone). Then, we observe that $F\Delta$ is obtained by removing the first arrow from Δ (recall that $F\alpha_n = \alpha_{n+1}$). Finally, the fact that two colimiting cocone with the same vertices are isomorphic implies that D and FD share the same limit and the same colimit and that there is an isomorphism $\phi : D \to FD$, i.e. the functor F admits a fixpoint. □

We will now consider as example the construction of the natural numbers.

Example 3.8 The functor defined by $FX = X \oplus \mathbb{C}$ gives the chain of embeddings $\mathbf{0} \to \mathbb{C} \to \mathbb{C}^2 \to \mathbb{C}^3 \to \cdots$, where $\mathbb{C}^n$ is the direct sum of n copies of $\mathbb{C}$. The relation $\alpha_{n+1}^E = \alpha_n^E \oplus \mathrm{id}_{\mathbb{C}}$ holds for every $n \in \mathbb{N}$ and thus by induction, $\alpha_n^E = \alpha_0^E \oplus \mathrm{id}_{\mathbb{C}^n}$. Hence, for every $n \in \mathbb{N}$, $\alpha_n^E : \mathbb{C}^n = \mathbf{0} \oplus \mathbb{C}^n \to \mathbb{C}^{n+1}$ is defined by $\alpha_n^E(c_1, \ldots, c_n) = (0, c_1, \ldots, c_n)$.

Similarly, for every $n \in \mathbb{N}$, $\alpha_n^P = \alpha_0^P \oplus \mathrm{id}_{\mathbb{C}^n} : \mathbb{C}^{n+1} \to \mathbb{C}^n$ is defined by $\alpha_n^P(c_1, c_2, \ldots, c_n) = (c_2, \ldots, c_n)$ and thus the property $\alpha_n^P(x_{n+1}) = x_n$ holds for every $(x_n)_n \in \bigoplus_{i \geq 1} \mathbb{C}^i = \bigoplus_{i \geq 0} \mathbb{C}$.

It follows that $D = \bigoplus_{i \geq 0} \mathbb{C} = \ell^\infty(\mathbb{N})$ for this functor. More generally, if one consider a functor $FX = X \oplus A$ where A is a W*-algebra, then $D = \bigoplus_{i \geq 0} A$.

4 Streams of qubits

We will now consider the functor $FX = (X \overline{\otimes} A) \oplus \mathbb{C}$ to represent the construction of a list of unbounded length whose elements are in a W*-algebra A. It should be noted that in this setting, the functor $FX = (X \overline{\otimes} M_2) \oplus \mathbb{C}$ represent the construction of a list of unbounded length whose elements are qubits.

The functor defined by $FX = (X \overline{\otimes} A) \oplus \mathbb{C}$ gives the chain of embeddings $\mathbf{0} \to \mathbb{C} \to A \oplus \mathbb{C} \to 2 \cdot A \oplus A \oplus \mathbb{C} \to \cdots$ where $n \cdot A$ denotes the spatial tensor of n

copies of A. Assume that this functor has a canonical fixpoint D (this point will be discussed in the next subsection).

From the relation $\alpha^E_{n+1} = (\alpha^E_n \otimes \mathrm{id}_A) \oplus \mathrm{id}_{\mathbb{C}}(n \in \mathbb{N})$, we obtain by induction that $\alpha^E_n = (\alpha^E_0 \otimes \mathrm{id}_{n \cdot A}) \oplus \mathrm{id}_{(n-1)\cdot A} \oplus \cdots \oplus \mathrm{id}_A \oplus \mathrm{id}_{\mathbb{C}}$ for every $n \in \mathbb{N}$. It follows that an embedding $\alpha^E_n : n \cdot A \oplus \cdots \oplus A \oplus \mathbb{C} \to (n+1) \cdot A \oplus \cdots \oplus A \oplus \mathbb{C}$ $(n \in \mathbb{N})$ is defined by $\alpha^E_n(\langle a^n_1, \ldots, a^n_n\rangle, \ldots, \langle a^1_1\rangle, x) = (\langle 0, a^{n-1}_1, \ldots, a^{n-1}_n\rangle, \langle a^{n-1}_1, \ldots, a^{n-1}_n\rangle, \ldots, \langle a^1_1\rangle, x)$.

It is clear that the corresponding projection is $\pi_{2,(n+1)\cdot A \oplus \cdots \oplus A \oplus \mathbb{C}}$ and thus $D = \bigoplus_{i \geq 0} i \cdot A$ (where, by convention, we denote $\mathbb{C}$ by $0 \cdot A$).

Remark 4.1 It is well known that $\overline{\bigotimes}_{i\geq 1} A$ is the colimit of the (trivial) diagram $A \xrightarrow{-\overline{\otimes} A} A \overline{\otimes} A \to \cdots$ in $\mathbf{W^*}_{\mathrm{NMIU}}$, the category of W*-algebras together with NMIU-maps. However in our framework, the functor $F = -\overline{\otimes} A$ is associated to the diagram $\mathbf{0} \to \mathbf{0}\overline{\otimes} A = \mathbf{0} \to \mathbf{0}\overline{\otimes} A = \mathbf{0} \to \cdots$.

4.1 Remarks about complete positivity

Unfortunately, the functor $FX = (X \overline{\otimes} A) \oplus \mathbb{C}$ does not preserve NsU-maps. However, one might consider restricting to NcPsU-maps to consider such functor. There is a $\mathbf{Dcpo}_{\perp !}$-enrichment for the category $\mathbf{W^*}_{\mathrm{cP}}$ of W*-algebras together with NcPsU-maps, investigated independently by Cho [Cho14], who proposed the following variation of the Löwner order :

$f \sqsubseteq_{cP} g$ if and only if $g - f$ is completely positive, i.e. $\forall n. \forall x. \mathcal{M}_n(f)(x) \leq \mathcal{M}_n(g)(x)$.

In fact, the following proposition shows that our domain-theoretic structure do not change if one restricts to completely-positive maps.

Proposition 4.2 *Let f and g be two NsU-maps from a W*-algebra A to a W*-algebra B. If f and g are completely positive maps, then the relation $f \sqsubseteq_{cP} g$ holds if and only if the relation $f \sqsubseteq g$ (Definition 2.1) holds.*

Proof. If $g - f$ is completely positive with f and g completely positive (i.e. $f \sqsubseteq_{cP} g$), it is therefore positive and thus it is clear that $f \sqsubseteq_{cP} g$ implies $f \sqsubseteq g$.

Conversely, we will now show that $f \sqsubseteq g$ implies $f \sqsubseteq_{cP} g$ when f and g are completely positive maps. It is equivalent to show that if f and $f + g$ are completely positive maps, then g is positive implies that g is completely positive.

By the Hahn-Banach theorem, if we consider $P = \mathbf{W^*}(A, B)$ as a normed vector space (defined pointwise) and $cP = \mathbf{W^*}_{\mathrm{cP}}(A, B)$ as a linear subspace of P and if we consider an element $z \notin P \setminus \mathrm{span}(cP)$, then there is a (continuous) linear map $\varphi : P \to \mathbb{R}$ with $\varphi(x) = 0$ for every $x \in cP$ and $\varphi(z) = 1$.

We will now apply this fact. If the map g is just positive and not completely positive, we obtain that $\varphi(g) = 1$ and therefore $\varphi(f + g) = \varphi(f) + \varphi(g) = 0 + 1 = 1$ by linearity. But this is absurd since, by assumption, the map $f + g$ is completely positive, and thus $\varphi(f + g) = 0$. It follows that g is completely positive. □

Then, it is easy to see that every directed join of completely positive maps is completely positive map as well (using the fact that $\mathcal{M}(\bigvee_i f_i) = \bigvee_i \mathcal{M}(f_i)$ for every direct set of completely positive maps $\{f_i\}_i$).

Moreover, as shown in [Kor12], the direct sums and the spatial tensor products of W*-algebras can be turned into endofunctors $-\oplus- : \mathbf{W^*}_{\mathrm{cP}} \times \mathbf{W^*}_{\mathrm{cP}} \to \mathbf{W^*}_{\mathrm{cP}}$ and $-\overline{\otimes}- : \mathbf{W^*}_{\mathrm{cP}} \times \mathbf{W^*}_{\mathrm{cP}} \to \mathbf{W^*}_{\mathrm{cP}}$, which are von Neumann functors.

Concluding remarks

The theorem 2.2 provides a $\mathbf{Dcpo}_{\perp!}$-enrichment for the category $\mathbf{W^*}$ of W*-algebras with NsU-maps, while the theorem 3.6 gives a canonical fixpoint for every multiplicative map-preservering locally continuous endofunctor on $\mathbf{W^*}$. We believe that these two theorems are encouraging enough to consider further investigations of the semantics of quantum computation, using W*-algebras.

Acknowledgment

The author would like to thank Bart Jacobs, Sam Staton, Robert Furber, Bas Westerbaan, and Sander Uijlen for helpful discussions. This research has been financially supported by the European Research Council (ERC) under the QCLS grant (Quantum Computation, Logic & Security).

References

[Bla06] B. Blackadar, *Operator Algebras : Theory of C*-algebras and von Neumann Algebras*, Springer, 2006.

[Cho14] K. Cho, "Semantics for a Quantum Programming Language by Operator Algebras". Masters thesis, The University of Tokyo. Available at http://www-mmm.is.s.u-tokyo.ac.jp/~ckn/papers/master-thesis.pdf

[Con07] J.B. Conway, *A course in functional analysis*, Graduate texts in mathematics 96, 2007.

[Dix51] J. Dixmier, "Sur certains espaces considrs par M.H. Stone", Summa Brasil. Math. 2, pp.151-181, 1951.

[Kad55] R. V. Kadison, "Operator Algebras with a Faithful Weakly-Closed Representation", Annals of mathematics, Second Series, Vol. 64, No. 1 (Jul., 1956), pp.175-181.

[KR83] R. V. Kadison, J. R. Ringrose, *Fundamentals of the theory of operator algebras. Volume 1 : Elementary theory*, 1983.

[Kor12] A. Kornell, "Quantum collections", arXiv:1202.2994, 2012.

[Low34] K. Löwner, "Über monotone Matrixfunktionen". Math. Z. 38, pp.177-216, 1934.

[RT92] J.J.M.M. Rutten, D. Turi, "On the Foundation of Final Semantics: Non-Standard Sets, Metric Spaces, Partial Orders". REX Workshop, pp. 477-530, 1992.

[RD66] B. Russo, H.A. Dye, "A note on unitary operators in C*-algebras", Duke Math. J., 33:413416, 1966.

[Sak71] S. Sakai, *C*-algebras and W*-algebras*, Springer-Verlag, 1971.

[SW12] K. Saitô, J.D. Maitland Wright, "On classifying monotone complete algebras of operators", Richerche di Mathematica 56(2), 2007.

[Sel04] P. Selinger, "Towards a quantum programming language", MSCS 14(4):527-586, 2004.

[SP82] M.B. Smyth and G.D. Plotkin. “The category-theoretic solution of recursive domain equations”. SIAM J. Comput., 11, pp. 761-783, 1982.

[Tak02] M. Takesaki, *Theory of operator algebras Vol. 1*, Springer, 2002.

[Vig46] J.-P. Vigier, “Infinite sequences of hermetian operators”, Ph.D. thesis, Geneva University, 1946.

A Correspondence between operator theory and order theory

In this section, we will provide the following correspondence table between operator theory and order theory, where A and B are C*-algebras.

Operator Theory	**Order theory**	**Reference**
A monotone-closed	$[0,1]_A$ directed-complete	A.2
$f : A \to B$ NsU-map	$f : [0,1]_A \to [0,1]_B$ Scott-continuous PsU-map	A.3
A W*-algebra	$[0,1]_A$ dcpo with a separating set of normal states	A.4

In the standard litterature [Bla06,Tak02], monotone-closed C*-algebras and normal maps are defined as follows.

Definition A.1 A C*-algebra A is monotone-closed (or monotone-complete) if every bounded increasing net of positive elements of A has a join in A^+.

A positive map $\phi : A \to B$ between C*-algebras is normal (or a N-map) if every increasing net $(x_\lambda)_{\lambda\in\Lambda}$ in A^+ with a join $\bigvee x_\lambda \in A^+$ is such that the net $(\phi(x_\lambda))_{\lambda\in\Lambda}$ is an increasing net in B^+ with join $\bigvee \phi(x_\lambda) = \phi(\bigvee x_\lambda)$.

In the standard definition of the notion of monotone-closedness, the increasing nets are not required to be bounded by the unit, like in the definitions we used in this thesis. We will now show that we can assume that the upper bound is the unit, without loss of generality.

Proposition A.2 *A C*-algebra A is monotone-closed if and only if the poset $([0,1]_A, \leq)$ is directed-complete.*

Proof. Let A be a C*-algebra.

If A is monotone-closed, then, by definition every increasing net of positive elements bounded by 1 has a join in $[0,1]_A$ and therefore, the poset $([0,1]_A, \leq)$ is directed-complete.

Conversely, suppose that $[0,1]_A$ is directed-complete. We now consider an increasing net of positive elements $(a_\lambda)_{\lambda\in\Lambda}$ in A^+, bounded by a nonzero positive element $b \in A^+$. Then, it restricts to an increasing net $(\frac{a_\lambda}{\|b\|})_{\lambda\in\Lambda}$ in $[0,1]_A$ since $b \leq \|b\| \cdot 1$. By assumption, the increasing net $(\frac{a_\lambda}{\|b\|})_{\lambda\in\Lambda}$ has a join $\bigvee_\lambda \frac{a_\lambda}{\|b\|} \in [0,1]_A$ and thus $\|b\| \bigvee_\lambda \frac{a_\lambda}{\|b\|}$ is an upper bound for $(a_\lambda)_{\lambda\in\Lambda}$.

Let $c \in A^+$ be an upper bound for the increasing net $(a_\lambda)_{\lambda\in\Lambda}$ such that $c \leq b$. For every $\lambda' \in \Lambda$, $a_{\lambda'} \leq c \leq b \leq \|b\| \cdot 1$ and thus $\frac{c}{\|b\|}$ is an upper bound for the increasing net $(\frac{a_\lambda}{\|b\|})_{\lambda\in\Lambda}$. It follows that $\bigvee_\lambda \frac{a_\lambda}{\|b\|} \leq \frac{c}{\|b\|}$ and therefore, $\|b\| \bigvee_\lambda \frac{a_\lambda}{\|b\|} \leq c$. Thus, $\|b\| \bigvee_\lambda \frac{a_\lambda}{\|b\|}$ is the join of the increasing net $(a_\lambda)_{\lambda\in\Lambda}$ bounded by b and we can conclude that A is monotone-closed. □

In this thesis, we have chosen to use the standard definition of normal maps. However, one can say that a PsU-map is normal if its restriction $f : [0,1]_A \to [0,1]_B$ is Scott-continuous.

Proposition A.3 *A PsU-map $f : A \to B$ between C*-algebras is normal if and only if its restriction $f : [0,1]_A \to [0,1]_B$ is Scott-continuous.*

Proof. Let $f : A \to B$ be a positive map between two C*-algebras A and B.

If f is normal, then by definition every increasing net $(x_\lambda)_{\lambda\in\Lambda}$ in $[0,1]_A \subseteq A^+$ with join $\bigvee x_\lambda \in [0,1]_A$ is such that the net $(f(x_\lambda))_{\lambda\in\Lambda}$ is an increasing net in $[0,1]_B \subseteq \mathcal{B}^+$ with join $\bigvee f(x_\lambda) = f(\bigvee x_\lambda) \in [0,1]_B$. That is to say, the restriction $f : [0,1]_A \to [0,1]_B$ is Scott-continuous.

Conversely, suppose that the restriction $f : [0,1]_A \to [0,1]_B$ is Scott-continuous. Let $(x_\lambda)_{\lambda\in\Lambda}$ be an increasing net in A^+ with a nonzero join $y \in A^+$. Since $y \leq \|y\| \cdot 1$, it restricts to an increasing net $(\frac{x_\lambda}{\|y\|})_{\lambda\in\Lambda}$ in $[0,1]_A$ with a join $\frac{y}{\|y\|}$. From the Scott-continuity of $f : [0,1]_A \to [0,1]_B$, we deduce that the net $(f(\frac{x_\lambda}{\|y\|}))_{\lambda\in\Lambda}$ is an increasing net in $[0,1]_B$ with join $\bigvee f(\frac{x_\lambda}{\|y\|}) = f(\frac{y}{\|y\|}) \in [0,1]_B$. It follows that the net $(f(x_\lambda))_{\lambda\in\Lambda}$, which is equal to $(\|y\|\, f(\frac{x_\lambda}{\|y\|}))_{\lambda\in\Lambda}$ by linearity, is an increasing net in B^+ with an upper bound $\|y\| \bigvee f(\frac{x_\lambda}{\|y\|}) = f(\|y\|\, \frac{y}{\|y\|}) = f(y) \in B^+$.

Suppose that $z \in B^+$ is an upper bound for the increasing net $(f(x_\lambda))_{\lambda\in\Lambda}$. From the fact that $f(x_{\lambda'}) \leq z$ and therefore $f(\frac{x_{\lambda'}}{\|y\|}) = \frac{f(x_{\lambda'})}{\|y\|} \leq \frac{z}{\|y\|}$ for every $\lambda' \in \Lambda$, we obtain that $f(\frac{y}{\|y\|}) \leq \frac{z}{\|y\|}$ and thus $f(y) \leq z$. It follows that $f(y)$ is the join of the increasing net $(f(x_\lambda))_{\lambda\in\Lambda}$. Hence, we can conclude that the map f is normal. □

It is known that a C*-algebra A is a W*-algebra if and only if it is monotone-complete and admits sufficiently many normal states, i.e. the set of normal states of A separates the points of A, see [Tak02, Theorem 3.16]. By combining this fact and Proposition A.2, one can provide an order-theoretic characterization of W*-algebras, as in the following theorem.

Theorem A.4 *Let A be a C*-algebra.*

Then A is a W-algebra if and only if its set of effects $[0,1]_A$ is directed-complete with a separating set of normal states (i.e. $\forall x \in A, \exists f \in \mathbf{W}^*(A,[0,1]_\mathbb{C}), f(x) \neq 0$).*

The proof will be postponed until after the following theorem, which can be found in [Tak02], and the following lemma.

Theorem A.5 *Every C*-algebra A admits a faithful (i.e. injective) representation, i.e. an injective *-homomorphism $\pi : A \to \mathcal{B}(H)$ for some Hilbert space H. A C*-algebra A is a W*-algebra if and only if there is a faithful representation $\pi : A \to \mathcal{B}(H)$, for some Hilbert space H, such that $\pi(A)$ is a strongly-closed subalgebra of $\mathcal{B}(H)$.*

Lemma A.6 *For each W*-algebra A, there is an isomorphism $A \simeq \mathrm{span}(\mathcal{NS}(A))'$, where $\mathcal{NS}(A) = \mathbf{W}^*(A, [0,1])$ is the collection of normal states of A.*

Proof. We now consider the map $\zeta_X : X \to X''$ defined by $\zeta_X(x)(\phi) = \phi(x)$ for $x \in X$ and $\phi \in X'$. Let A be a W*-algebra. We observe that $\zeta_{A_*} : A_* \to A'$ is a "canonical embedding" of A_* into A' and it can be proved that A_* is a linear subspace of A' generated by the normal states of A, i.e. $\zeta_{A_*}(A_*) = \mathrm{span}(\mathcal{NS}(A))$, see the proof of [Sak71, Theorem 1.13.2]. Then, we can now consider the induced surjection $\zeta_{A_*} : A_* \to \mathrm{span}(\mathcal{NS}(A))$, which turns out to be injective (and thus bijective): for every pair $(x, y) \in A_* \times A_*$ such that $x \neq y$, there is a $f \in \mathcal{NS}(A)$ such that $\zeta_{A_*}(x)(f) = f(x) \neq f(y) = \zeta_{A_*}(y)(f)$, which implies that $\zeta_{A_*}(x) \neq \zeta_{A_*}(y)$.

Then for every W*-algebra, from $A_* \simeq \mathrm{span}(\mathcal{NS}(A))$ for every W*-algebra A, we obtain that $A = (A_*)' \simeq \mathrm{span}(\mathcal{NS}(A))'$.

□

Proof. [Proof of Theorem A.4] Let A be a C*-algebra.

Suppose that A is a W*-algebra. Then, by Corollary 2.5, $[0,1]_A$ is a dcpo and thus A is monotone-complete by Proposition A.2. Moreover, we know by Lemma A.6 that there is an isomorphism $\zeta_A : A \to \mathrm{span}(\mathcal{NS}(A))'$ defined by $\zeta_A(a)(\varphi) = \varphi(a)$ for $a \in A$ and $\varphi \in A'$. Therefore, ζ_A is injective and thus for every pair (x, y) of distinct elements of A, $\zeta_A(x) \neq \zeta_A(y)$, which means that there is a $\varphi \in \mathcal{NS}(A)$ such that $\varphi(x) = \zeta_A(x)(\varphi) \neq \zeta_A(y)(\varphi) = \varphi(y)$. It follows that the set $\mathcal{NS}(A)$ is a separating set for A.

Conversely, suppose that A is monotone-closed and admits its normal states as a separating set.

There is a representation $\pi : A \to B(H)$, for some Hilbert space H, induced by the normal states on A, by the Gelfand-Naimark-Segal (GNS) construction [Tak02, Theorem I.9.14, Definition I.9.15]:

- Every normal state ω on A induces a representation $\pi_\omega : A \to \mathcal{B}(H_\omega)$ such that there is a vector ξ_ω such that $\omega(x) = \langle \pi_\omega(x)\xi_\omega | \xi_\omega \rangle$ for every $x \in A$
- We define a Hilbert space H, which is the direct sum of the Hilbert spaces H_ω, where ω is a normal state on A.
- The representation $\pi : A \to \mathcal{B}(H)$ is defined pointwise for every $x \in A$: $\pi(x)$ is the bounded operator on H defined as the direct sum of the bounded operators $\pi_\omega(x)$ on H_ω, where ω is a normal state on A.

By assumption, the set of normal states of A is a separating set for A and thus, for every pair of distincts elements x, y in A, there is a state ρ on A such that $\langle \pi_\rho(x)\xi_\rho | \xi_\rho \rangle = \rho(x) \neq \rho(y) = \langle \pi_\rho(y)\xi_\rho | \xi_\rho \rangle$ and thus $\pi_\rho(x) \neq \pi_\rho(y)$ for some state ρ on A. It follows that $\pi(x) \neq \pi(y)$ and hence, the representation π is faithful.

Let ρ be a normal state on A. Since A is monotone-closed, every directed set $(\rho(x_\lambda))_{\lambda \in \Lambda}$ in $\mathcal{B}(H)$ has a join $\bigvee_\lambda \rho(x_\lambda) = \rho(\bigvee_\lambda x_\lambda)$. According to the definition we gave earlier of π_ρ, this imply that $\pi_\rho(x_\lambda)$ converges weakly to $\bigvee_\lambda \pi_\rho(x_\lambda)$. Since a bounded net of positive operators converges strongly whenever it converges weakly (see [Bla06, I.3.2.8]), it turns out that $\bigvee_\lambda \pi_\rho(x_\lambda)$ is the strong limit of $(\pi_\rho(x_\lambda))_{\lambda \in \Lambda}$ in

$\mathcal{B}(H_\rho)$. Hence, the strong limit of $(\pi(x_\lambda))_{\lambda\in\Lambda}$ in $\mathcal{B}(H)$ exists in $\mathcal{B}(H)$ and is defined as the direct sum of the strong limit of the nets $(\pi_\omega(x_\lambda))_{\lambda\in\Lambda}$ where ω is a normal state on A. Thus, $\pi(A)$ is strongly closed in $\mathcal{B}(H)$ and thus A is a W*-algebra (by Theorem A.5). □

It is important to note that, in one of the very first articles about W*-algebras [Kad55], Kadison defined W*-algebras as monotone-closed C*-algebras which separates the points. However, to our knowledge, this definition never became standard.

Identifying All Preorders on the Subdistribution Monad

Tetsuya Sato[1]

Research Institute for Mathematical Sciences
Kyoto University
Kyoto, Japan

Abstract

The countable valuation monad, the countable distribution monad, and the countable subdistribution monad are often used in the coalgebraic treatment of discrete probabilistic transition systems. We identify preorders on them using a technique based on the preorder $\top\top$-lifting and elementary facts about preorders on real intervals preserved by convex combinations. We show that there are exactly 15, 5, and 41 preorders on the countable valuation monad, the countable distribution monad, and the countable subdistribution monad respectively. We also give concrete definitions of these preorders. By applying Hughes and Jacobs's construction to some preorder on the countable subdistribution monad, we obtain probabilistic bisimulation between Markov chains ignoring states with deadlocks.

Keywords: coalgebras, preorders, monads, probabilistic transition systems, probabilistic bisimulation

1 Introduction

We completely identify preorders on the countable valuation monad $\mathcal{V}$, the countable distribution monad $\mathcal{D}_{=1}$, and the countable subdistribution monad $\mathcal{D}$ on **Set** respectively. We list the main results of this paper:

- There are exactly 15 preorders on the monad $\mathcal{V}$, and they are generated from 4 preorders $\sqsubseteq^0$, $\sqsubseteq^1$, $\sqsubseteq^2$, and $\sqsubseteq^3$ (Section 4).
- There are exactly 5 preorders on the monad $\mathcal{D}_{=1}$, and they are generated from the equality $\mathrm{Eq}^{\mathcal{D}_{=1}}$ and the support-inclusion $\sqsubseteq^s$ (Section 5).
- There are exactly 41 preorders on the monad $\mathcal{D}$, and they are generated from 5 preorders $\sqsubseteq^r$, $\sqsubseteq^s$, $\sqsubseteq^d$, $\sqsubseteq^m$, and $\sqsubseteq^M$ (Section 6).
- To identify preorders on $\mathcal{V}$, it is enough to analyse preorders at the singleton type. To identify preorders on $\mathcal{D}$ and $\mathcal{D}_{=1}$, it is enough to analyse preorders at

[1] Email:satoutet@kurims.kyoto-u.ac.jp

the Boolean type.

Our task is identifying the class $\mathbf{Pre}(T)$ of preorders on a monad T ($T = \mathcal{V}, \mathcal{D}_{=1}, \mathcal{D}$). We focus on the component $\sqsubseteq_I$ of each $\sqsubseteq \in \mathbf{Pre}(T)$ at a set I. The component $\sqsubseteq_I$ is a preorder on TI that satisfies *congruence* and *substitutivity*. We denote by $\mathbf{CSPre}(T, I)$ the set of such preorders on TI. We introduce the mapping $(-)_I \colon \mathbf{Pre}(T) \to \mathbf{CSPre}(T, I)$ that extracts components at I from preorders on T. We calculate preorders on T from $\mathbf{CSPre}(T, I)$ by the left adjoint $\langle - \rangle^I$ and the right adjoint $[-]^I$ of the mapping $(-)_I$, and we analyse the sandwiching situation $\langle \preceq \rangle^I \trianglelefteq \sqsubseteq \trianglelefteq [\preceq]^I$ for each $\preceq \in \mathbf{CSPre}(T, I)$, where $\trianglelefteq$ is the component-wise inclusion order for preorders on T.

We identify $\mathbf{Pre}(\mathcal{V})$, $\mathbf{Pre}(\mathcal{D}_{=1})$, and $\mathbf{Pre}(\mathcal{D})$ as the following steps:

(i) We identify the sets $\mathbf{CSPre}(\mathcal{V}, 1)$, $\mathbf{CSPre}(\mathcal{D}_{=1}, 2)$, and $\mathbf{CSPre}(\mathcal{D}, 1)$. Then, the class $\mathbf{Pre}(\mathcal{V})$ is identified by applying [9, Lemma 7].

(ii) We calculate the mappings $\langle - \rangle^I$ and $[-]^I$ for $(T, I) = (\mathcal{D}_{=1}, 2)$ and $(T, I) = (\mathcal{D}, 1)$. We then identify $\mathbf{Pre}(\mathcal{D}_{=1})$ by proving $\langle - \rangle^2 = [-]^2$. To finish identifying $\mathbf{Pre}(\mathcal{D})$, we analyse the remaining preorders $\sqsubseteq \in \mathbf{Pre}(\mathcal{D})$ such that $\langle \sqsubseteq_1 \rangle^1 \ntrianglelefteq \sqsubseteq \ntrianglelefteq [\sqsubseteq_1]^1$ by using preorders on $\mathcal{D}_{=1}$.

In [9], Katsumata and the author developed a method to identity preorders on monads, but it is not applied well to the monads $\mathcal{V}$, $\mathcal{D}_{=1}$, and $\mathcal{D}$. In this paper, we introduce the following new ideas to identify $\mathbf{Pre}(\mathcal{V})$, $\mathbf{Pre}(\mathcal{D}_{=1})$, and $\mathbf{Pre}(\mathcal{D})$: in (i) of the above steps, we use Lemma 1.1 to identify congruent and substitutive preorders on the *infinite* sets $\mathcal{V}1$, $\mathcal{D}_{=1}2$, and $\mathcal{D}1$. In (ii), we introduce the left adjoint $\langle - \rangle^I$ of the mapping $(-)_I$, and we use the sandwiching situation $\langle \preceq \rangle^I \trianglelefteq \sqsubseteq \trianglelefteq [\preceq]^I$ to identify $\mathbf{Pre}(\mathcal{D}_{=1})$ and $\mathbf{Pre}(\mathcal{D})$.

This work is motivated by a mathematical interest. The author has not found interesting applications of the main results of this work yet, but at least, we have the following contribution: By applying preorders on $\mathcal{D}$ to methods in [6,8,9], we discuss coalgebraic simulations between probabilstic transition systems, and obtain probabilistic bisimulations *ignoring states with deadlocks* between Markov chains (Section 7). From preorders on the monad $\mathcal{D}_{=1}$ (resp. $\mathcal{D}$), we obtain all precongruences on each typed language given from probabilistic choice $\sum_{i \in I} p_i(-_i)$ (resp. and deadlocks) and several axioms.

1.1 Background

Preorders on monads are equivalent to pointwise *preorder enrichments on their Kleisli categories*. A suitable partial order on a monad gives a coalgebraic trace semantics [5] and forward/backward simulations between coalgebras [4]. In the studies [6,8], simulations between coalgebras are given from preorders on coalgebra functors systematically. Many of them involve preorders on monads (e.g. the inclusion order $\mathcal{P}(A \times -)$).

In the study [11], precongruences on a typed language with nondeterminism (`or`) and a divergent term are determined completely, and they are almost equivalent to preorders on the composite monad $\mathcal{PL}$ of the powerset monad $\mathcal{P}$ and the monad $\mathcal{L}$ given by $\mathcal{L} = 1 + \mathrm{Id}$ [9]. From this point of view, in other words, our work is seen as the variant of [11] for probabilistic systems.

1.2 Preliminaries

Throughout this paper, we work on the category **Set** of sets and functions. For a monad (T, η, μ) on **Set** and a function $f\colon X \to TY$, the *Kleisli Lifting* $f^\sharp\colon TX \to TY$ of f is the composition $f^\sharp = \mu \circ T(f)$.

For each set X, we denote by $\top_X$ the trivial relation $X \times X$ on X, and denote by Eq_X the equality/diagonal relation on X. We denote by R^{op} the opposite relation of R.

We will use the *complete semiring* $([0,\infty], +, \cdot, 0, 1)$ for the countable valuation monad; it has arbitrary summations, and an infinite sum is the least upper bound with respect to the standard order $\leq$ of all finite partial sums [3, Volume A, pp. 124–125, denoted by $\mathscr{R}_+$].

The following lemma is crucial to analyse preorders.

Lemma 1.1 *Let $0 < N < \infty$. If $\preceq$ is a preorder on the interval $[0, N]$ that is preserved by* convex combinations*; in other words, the preorder $\preceq$ satisfies*

$$(p_1 \preceq q_1 \wedge p_2 \preceq q_2 \wedge t \in [0,1]) \implies tp_1 + (1-t)p_2 \preceq tq_1 + (1-t)q_2$$

then $p \preceq q$ for some $0 < p < q < N$ implies $r \preceq s$ for each $0 < r < s < N$.

Proof (sketch). Let $a_n = p^{n+1}/q^n$ and $b_n = \beta^n p + (1 - \beta^n)N$ for each $n \in \mathbb{N}$, where $\beta = (N-q)/(N-p)$. We prove $a_n \preceq b_n$ $(n \in \mathbb{N})$, $\lim_{n\to\infty} a_n = 0$, and $\lim_{n\to\infty} b_n = N$ from $p \preceq q$ and $0 < p < q < N$. We then prove $r \preceq s$ for each $0 < r < s < N$ by using $a_m < r < s < b_m$ and $a_m \preceq b_m$ for some $m \in \mathbb{N}$. □

2 Monads for Probabilistic Branching

We first introduce some notations: the *sum* $d[U]$ of $d\colon X \to [0,\infty]$ over $U \subseteq X$ is defined by $\sum_{x \in U} d(x)$. The *support* of $d\colon X \to [0,\infty]$ is defined by $\mathrm{supp}(d) = \{\, x \in X \mid d(x) \neq 0 \,\}$. The *zero distribution* $\mathtt{0}$ is defined by $\mathtt{0}(x) = 0$. The *Dirac distribution* δ_x is defined by $\delta_x(x) = 1$ and $\delta_x(y) = 0$ $(x \neq y)$.

Next, we define the three monads $\mathcal{V}$, $\mathcal{D}$, and $\mathcal{D}_{=1}$ on **Set** as follows:

Definition 2.1
- We denote by $(\mathcal{V}, \eta^{\mathcal{V}}, \mu^{\mathcal{V}})$ the *countable valuation monad* that is defined as follows: the functor part $\mathcal{V}$ is defined by for each set X, $\mathcal{V}X = \{\, d\colon X \to [0,\infty] \mid \omega \geq |\mathrm{supp}(d)| \,\}$ and $\mathcal{V}f(d)(y) = \sum_{x \in f^{-1}(y)} d(x)$ for each $f\colon X \to Y$ and $y \in Y$. The unit and multiplication are defined by $\eta^{\mathcal{V}}_X(x) = \delta_x$ and $(\mu^{\mathcal{V}}_X(\xi))(x) = \sum_{d \in \mathcal{V}X} \xi(d) \cdot d(x)$ $(x \in X)$.
- The countable *subdistribution monad* $(\mathcal{D}, \eta^{\mathcal{D}}, \mu^{\mathcal{D}})$ is defined as follows: for each set X, $\mathcal{D}X = \{\, d\colon X \to [0,1] \mid d[X] \leq 1 \,\}$, and the unit and the multiplication are inherited from the countable valuation monad.

- The countable *distribution monad* $(\mathcal{D}_{=1}, \eta^{\mathcal{D}_{=1}}, \mu^{\mathcal{D}_{=1}})$ is defined as follows: for each set X, $\mathcal{D}_{=1}X = \{\, d \colon X \to [0,1] \mid d[X] = 1 \,\}$, and the unit and the multiplication are inherited from the subdistribution monad.

We remark that the condition $\omega \geq |\mathrm{supp}(d)|$ is automatically obtained from $d[X] = 1$ ($d[X] \leq 1$) in the definitions of the (sub)distribution monad.

The probabilistic branching is characterised coalgebraically by $\mathcal{D}$:

- A Markov chain is characterised as $\xi_1 \colon X \to \mathcal{D}X$.
- A probabilistic transition system is characterised as $\xi_2 \colon X \to \mathcal{D}(A \times X)$.
- A Segala automaton [13] is characterised as $\xi_3 \colon X \to \mathcal{P}\mathcal{D}(1 + A \times X)$.

Since $\mathcal{D}X \cong \mathcal{D}_{=1}(1 + X)$, we obtain the notion of *deadlocks* in the probabilistic branching. For example, a Markov chain $\xi \colon X \to \mathcal{D}X$ has a deadlock at a state $x \in X$ when $\xi(x)[X] < 1$. For further examples, see [14].

3 The Class of Preorders on a Monad

We introduce some results of [9], which we use to identify preorders on monads. We fix a monad (T, η, μ) on **Set**. We denote it by T for simplicity.

We define the congruence and substitutivity of preorders on TI and preorders on the monad T, the latter of which correspond bijectively to pointwise preorder enrichments of the Kleisli category $\mathbf{Set}_T$ of T.

Definition 3.1 Let I be a set, and let $\preceq$ be a preorder on TI. (i) We call $\preceq$ *congruent* if $(\forall j \in J. f(j) \preceq g(j)) \implies (\forall x \in TJ. f^\sharp(x) \preceq g^\sharp(x))$ for each set J and functions $f, g \colon J \to TI$. (ii) We call $\preceq$ *substitutive* if $f^\sharp$ is a monotone function on $(TI, \preceq)$ for each $f \colon I \to TI$.

We write $(\mathbf{CSPre}(T, I), \subseteq)$ for the set of congruent and substitutive preorders on TI, ordered by inclusions. It is closed under opposites and intersections, and it has the greatest and least preorders $\top_{TI}$ and Eq_{TI} respectively.

Definition 3.2 ([9, Definition 3]) A *preorder* $\sqsubseteq$ on a monad T is an assignment of a preorder $\sqsubseteq_I$ on TI to each set I such that (i) each $\sqsubseteq_I$ is congruent, and (ii) for each $f \colon J \to TI$, $f^\sharp$ is a monotone function from $(TJ, \sqsubseteq_J)$ to $(TI, \sqsubseteq_I)$ (we also call this property *substitutivity*).

For example, the assignment $\sqsubseteq$ that is defined by $A \sqsubseteq_X B \iff A \subseteq B$ is indeed a preorder on the powerset monad $\mathcal{P}$.

We write $(\mathbf{Pre}(T), \trianglelefteq)$ for the class of preorders on T, ordered by the partial order $\trianglelefteq$ defined by $\sqsubseteq \trianglelefteq \sqsubseteq' \overset{\mathrm{def}}{\iff} \forall I. \sqsubseteq_I \subseteq \sqsubseteq'_I$. It is closed under these opposites and intersections, which are defined by $(\sqsubseteq^{\mathrm{op}})_X = (\sqsubseteq_X)^{\mathrm{op}}$ and $(\bigcap_{\lambda \in \Lambda} \sqsubseteq^\lambda)_X = \bigcap_{\lambda \in \Lambda} \sqsubseteq^\lambda_X$, and it has the least and greatest preorders: the equality Eq^T defined by $\mathrm{Eq}^T_X = \mathrm{Eq}_{TX}$ and the trivial preorder $\top^T$ defined by $\top^T_X = \top_{TX}$.

For each preorder $\sqsubseteq$ on T, we call $\sqsubseteq_I$ the *evaluation* at I of $\sqsubseteq$.

$$\begin{tikzcd}
(\mathbf{CSPre}(T,I),\subseteq) \arrow[d, shift right=8, "\langle - \rangle^I"] \arrow[d, shift left=8, "{[-]^I}"] \arrow[d, phantom, "\dashv \qquad \dashv"] \\
(\mathbf{Pre}(T),\trianglelefteq) \arrow[u, "(-)_I"']
\end{tikzcd}$$

The evaluation mapping $(-)_I \colon \sqsubseteq \mapsto \sqsubseteq_I$ is a monotone mapping from $(\mathbf{Pre}(T),\trianglelefteq)$ to $(\mathbf{CSPre}(T,I),\subseteq)$. It has both the right and left adjoints. The right adjoint $[-]^I$ of the evaluation mapping $(-)_I$ is defined by

$$x\,[\preceq]^I_X\, y \iff \forall f\colon X \to TI. f^\sharp(x) \preceq f^\sharp(y).$$

The mapping $[-]^I$ is monotone, and it preserves opposites and intersections.

Proposition 3.3 ([9, Theorem 3]) *For each I, $(-)_I \dashv [-]^I$ and $[-]^I_I = \mathrm{Id}$.*

Hence, the preorder $[\preceq]^I$ on T is the *greatest* one whose evaluation at I equals $\preceq$ for each $\preceq\, \in \mathbf{CSPre}(T,I)$.

The left adjoint $\langle - \rangle^I$ of the evaluation mapping $(-)_I$ is defined by

$$\langle \preceq \rangle^I = \bigcap \{ \sqsubseteq \in \mathbf{Pre}(T) \mid \sqsubseteq_I = \preceq \}.$$

The preorder $\langle \preceq \rangle^I$ on T is the *least* one whose evaluation at I equals $\preceq$ for each $\preceq\, \in \mathbf{CSPre}(T,I)$ since $\mathbf{Pre}(T)$ is closed under intersections, and $[\preceq]^I_I = \preceq$ holds. By using this, we easily obtain that the mapping $\langle - \rangle^I$ is monotone, that it preserves opposites, and that the adjunction $\langle - \rangle^I \dashv (-)_I$ holds.

Lemma 3.4 *Let $\preceq\, \in \mathbf{CSPre}(T,I)$. If $[\preceq]^I = \langle \preceq \rangle^I$ then the preorder $[\preceq]^I$ the* unique *preorder whose evaluation at I equals $\preceq$.*

We here introduce the opposite-intersection operators on $\mathbf{Pre}(T)$ and $\mathbf{CSPre}(T,I)$. The one on $\mathbf{CSPre}(T,I)$ is given as follows:

$$\mathcal{C}^{\mathbf{CSPre}(T,I)}_{\cap,\mathrm{op}}(\mathbb{K}) = \left\{ \bigcap \mathbb{L} \cap \left(\bigcap \mathbb{M} \right)^{\mathrm{op}} \;\middle|\; \mathbb{L},\mathbb{M} \subseteq \mathbb{K} \right\} \text{ where } \mathbb{K} \subseteq \mathbf{CSPre}(T,I)$$

The opposite-intersection closure operator on $\mathbf{Pre}(T)$ is given in a similar way as the above (we denote it by $\mathcal{C}^{\mathbf{Pre}(T)}_{\cap,\mathrm{op}}$). We often write $\mathcal{C}_{\cap,\mathrm{op}}$ for simplicity.

3.1 Main Results

Theorem 3.5 *Preorders on $\mathcal{V}$, $\mathcal{D}_{=1}$, and $\mathcal{D}$ are identified as follows:*

(i) $\mathbf{Pre}(\mathcal{V}) = \mathcal{C}_{\cap,\mathrm{op}}\{\sqsubseteq^0,\sqsubseteq^1,\sqsubseteq^2,\sqsubseteq^3\} \cong \mathbf{CSPre}(\mathcal{V},1) \cong 15$ *where*

$$\begin{aligned}
d_1 \sqsubseteq^0_X d_2 &\overset{\text{def}}{\iff} \mathrm{supp}(d_1) \subseteq \mathrm{supp}(d_2)\\
d_1 \sqsubseteq^1_X d_2 &\overset{\text{def}}{\iff} \forall x \in X. (d_1(x) \leq d_2(x))\\
d_1 \sqsubseteq^2_X d_2 &\overset{\text{def}}{\iff} \forall x \in X. (d_1(x) = d_2(x) \vee d_2(x) = \infty)\\
d_1 \sqsubseteq^3_X d_2 &\overset{\text{def}}{\iff} \forall x \in X. (d_1(x) \leq d_2(x) \wedge (d_1(x) = 0 \implies d_2(x) \in \{\infty, 0\})).
\end{aligned}$$

(ii) $\mathbf{Pre}(\mathcal{D}_{=1}) = \mathcal{C}_{\cap,\mathrm{op}}\{\sqsubseteq^s, \mathrm{Eq}^{\mathcal{D}_{=1}}\} \cong \mathbf{CSPre}(\mathcal{D}_{=1},2) \cong 5$ *where*

$$d_1 \sqsubseteq^s_X d_2 \overset{\text{def}}{\iff} \mathrm{supp}(d_1) \subseteq \mathrm{supp}(d_2).$$

(iii) $\mathbf{Pre}(\mathcal{D}) = \mathcal{C}_{\cap,\text{ꟼP}}\{\sqsubseteq^r, \sqsubseteq^s, \sqsubseteq^d, \sqsubseteq^m, \sqsubseteq^M\} \cong \mathbf{CSPre}(\mathcal{D}, 2) \cong 41$ *where*

$$\begin{aligned}
d_1 \sqsubseteq^r_X d_2 &\overset{\text{def}}{\iff} \forall x \in X. d_1(x) \le d_2(x),\\
d_1 \sqsubseteq^s_X d_2 &\overset{\text{def}}{\iff} \mathrm{supp}(d_1) \subseteq \mathrm{supp}(d_2),\\
d_1 \sqsubseteq^d_X d_2 &\overset{\text{def}}{\iff} (d_1[X] = 1 \implies d_2[\mathrm{supp}(d_1)] = 1),\\
d_1 \sqsubseteq^m_X d_2 &\overset{\text{def}}{\iff} (d_1[X] = 1 \implies d_2 = d_1),\\
d_1 \sqsubseteq^M_X d_2 &\overset{\text{def}}{\iff} (d_1[X] = 1 \implies (d_2[X] = 1 \wedge \mathrm{supp}(d_1) = \mathrm{supp}(d_2))).
\end{aligned}$$

We prove (i), (ii), and (iii) of Theorem 3.5 in Section 4, 5, and 6.

4 Preorders on the Countable Valuation Monad

Preorders on a semiring-valued *finite* multiset monad are pointwise [9, Lemma 7 and Theorem 8]. The following lemma holds by applying this fact to the monad $\mathcal{V}$ with a slight change of cardinality of supports to *countable*.

Lemma 4.1 *Each* $\sqsubseteq \in \mathbf{Pre}(\mathcal{V})$ *satisfies* $d_1 \sqsubseteq_X d_2 \iff \forall x \in X. d_1(x) \sqsubseteq_1 d_2(x)$, *where* $1 = \{*\}$*. Moreover,* $\mathbf{CSPre}(\mathcal{V}, 1) \cong \mathbf{Pre}(\mathcal{V})$.

Hence, it suffices to identify $\mathbf{CSPre}(\mathcal{V}, 1)$ to identify $\mathbf{Pre}(\mathcal{V})$. We regard $\mathcal{V}1 \cong [0, \infty]$ by the correspondence of each $d \in \mathcal{V}1$ with the value $d(*) \in [0, \infty]$. For each $\preceq \in \mathbf{CSPre}(\mathcal{V}, 1)$, the substitutivity of $\preceq$ is equivalent to

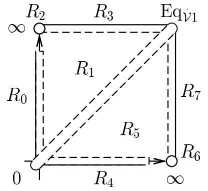
$$(p \preceq q \wedge t \in [0, \infty]) \implies tp \preceq tq,$$

and the congruence of $\preceq$ is equivalent to

$$\forall i \in I. (p_i \preceq q_i \wedge t_i \in [0, \infty]) \implies \textstyle\sum_{i \in I} p_i t_i \preceq \sum_{i \in I} q_i t_i.$$

Hence, each $\preceq \in \mathbf{CSPre}(\mathcal{V}, 1)$ is preserved by convex combinations.

We partition the set $\mathcal{V}1 \times \mathcal{V}1 \cong [0, \infty] \times [0, \infty]$ into $R_0 = \{ (0, q) \mid q \in (0, \infty) \}$, $R_1 = \{ (p, q) \mid 0 < p < q < \infty \}$, $R_2 = \{(0, \infty)\}$, $R_3 = \{ (p, \infty) \mid p \in (0, \infty) \}$, $R_4 = R_0{}^{\text{ꟼP}}$, $R_5 = R_1{}^{\text{ꟼP}}$, $R_6 = R_2{}^{\text{ꟼP}}$, $R_7 = R_3{}^{\text{ꟼP}}$, and $\mathrm{Eq}_{\mathcal{V}1}$.



By using Lemma 1.1, we obtain Lemma 4.2 and 4.3.

Lemma 4.2 *Let* $\preceq \in \mathbf{CSPre}(\mathcal{V}, 1)$*. We obtain the following properties:*

(i) $p \preceq \infty$ *for some* $0 < p < \infty$ *if and only if* $r \preceq \infty$ *for all* $0 < r \le \infty$.
This is equivalent to $R_3 \cap \preceq \neq \emptyset \implies R_3 \subseteq \preceq$.

(ii) $0 \preceq \infty$ *if and only if* $r \preceq s$ *for all* $0 \le r \le \infty$.
This is equivalent to $R_2 \cap \preceq \neq \emptyset \implies R_2 \cup R_3 \subseteq \preceq$.

(iii) $p \preceq q$ *for some* $0 < p < q < \infty$ *if and only if* $r \preceq s$ *for all* $0 < r < s \le \infty$.
This is equivalent to $R_1 \cap \preceq \neq \emptyset \implies R_1 \cup R_3 \subseteq \preceq$.

(iv) $0 \preceq q$ *for some* $0 < q < \infty$ *if and only if* $r \preceq s$ *for all* $0 \le r < s \le \infty$.
This is equivalent to $R_0 \cap \preceq \neq \emptyset \implies R_0 \cup R_1 \cup R_2 \cup R_3 \subseteq \preceq$.

Lemma 4.3 *Let* $\preceq \in \mathbf{CSPre}(\mathcal{V}, 1)$*. We obtain* $\preceq = \mathrm{Eq}_{\mathcal{V}1} \cup \bigcup_{i \in I} R_i$ *where* $I = \{ i \in \{0, 1, \dots, 7\} \mid R_i \cap \preceq \neq \emptyset \}$.

We prepare the following congruent substitutive preorders on $\mathcal{V}1$:

- $p \preceq^0 q \overset{\text{def}}{\iff} (p > 0 \implies q > 0)$
- $p \preceq^1 q \overset{\text{def}}{\iff} (p \leq q)$
- $p \preceq^2 q \overset{\text{def}}{\iff} (p = q) \vee (q = \infty)$
- $p \preceq^3 q \overset{\text{def}}{\iff} (p \leq q) \wedge (p = 0 \implies q \in \{\infty, 0\})$

Proposition 4.4 *We obtain* $\mathbf{CSPre}(\mathcal{V}, 1) = \mathcal{C}_{\cap,\mathrm{op}} \left\{\preceq^0, \preceq^1, \preceq^2, \preceq^3\right\} \cong 15$.

Proof (Sketch). Let $\preceq \in \mathbf{CSPre}(\mathcal{V}, 1)$. We define $R(p_0, p_1, \ldots, p_7) = \mathrm{Eq}_{\mathcal{V}1} \cup \bigcup \{ R_i \mid p_i = \text{true} \}$. By Lemma 4.3, we obtain $\preceq = R(p_0, p_1, \ldots, p_7)$ where $p_i \iff R_i \cap \preceq \neq \emptyset$ $(i \in \{0, 1, \ldots, 7\})$. From Lemma 4.2 and the transitivity of $\preceq$, the octuple $(p_0, p_1, \ldots, p_7)$ should satisfy the following formula:

$$\begin{aligned} P = \; & (p_0 \implies p_1 \wedge p_2) \wedge (p_1 \vee p_2 \implies p_3) \\ & \wedge (p_3 \wedge p_7 \implies p_1 \wedge p_5) \wedge (p_2 \wedge p_7 \implies p_0) \wedge (p_3 \wedge p_6 \implies p_4) \\ & \wedge (p_4 \implies p_5 \wedge p_6) \wedge (p_5 \vee p_6 \implies p_7). \end{aligned}$$

We remark that the last 2 clauses of P are given by applying $\preceq^{\mathrm{op}}$ to Lemma 4.2. There are exactly 15 satisfying assignments of P. We then prove by hand,

$$15 \cong \{ R(p_0, p_1, \ldots, p_7) \mid (p_0, p_1, \ldots, p_7) \text{ satisfies } P \} \subseteq \mathcal{C}_{\cap,\mathrm{op}}\{\preceq^0, \preceq^1, \preceq^2, \preceq^3\}.$$

Since $\mathbf{CSPre}(\mathcal{V}, 1) \subseteq \{ R(p_0, p_1, \ldots, p_7) \mid (p_0, p_1, \ldots, p_7) \text{ satisfies } P \}$ and $\preceq^0, \preceq^1, \preceq^2, \preceq^3 \in \mathbf{CSPre}(\mathcal{V}, 1)$, we conclude this proposition. □

Theorem 4.5 (Theorem 3.5(i)) *Let $\sqsubseteq^i$ be the pointwise ordering generated from $\preceq^i$ $(i \in \{0, 1, 2, 3\})$. We obtain* $\mathbf{Pre}(\mathcal{V}) = \mathcal{C}_{\cap,\mathrm{op}} \left\{\sqsubseteq^0, \sqsubseteq^1, \sqsubseteq^2, \sqsubseteq^3\right\} \cong 15$.

Proof. It is proved immediately from Lemma 4.1 and Proposition 4.4. □

5 Preorders on the Distribution Monad

First, we identify $\mathbf{CSPre}(\mathcal{D}_{=1}, 2)$ where $2 = \{\mathbf{0}, \mathbf{1}\}$. We regard $\mathcal{D}_{=1}2 \cong [0, 1]$ by the correspondence of each $d = d(\mathbf{0})\delta_{\mathbf{0}} + (1 - d(\mathbf{0}))\delta_{\mathbf{1}} \in \mathcal{D}_{=1}2$ with $d(\mathbf{0}) \in [0, 1]$. For each $\preceq \in \mathbf{CSPre}(\mathcal{D}_{=1}, 2)$, the substitutivity of $\preceq$ is equivalent to

$$p \preceq q \implies \forall t, u \in [0, 1].((t - u)p + u \preceq (t - u)q + u),$$

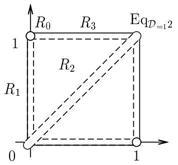
and the congruence of $\preceq$ is equivalent to

$$(\forall i \in I.(p_i \preceq q_i) \wedge \textstyle\sum_{i \in I} t_i = 1) \implies \textstyle\sum_{i \in I} p_i t_i \preceq \sum_{i \in I} q_i t_i.$$

Hence, each $\preceq \in \mathbf{CSPre}(\mathcal{V}, 1)$ is preserved by convex combinations.

We partition the set $\mathcal{D}_{=1}2 \times \mathcal{D}_{=1}2 \cong [0, 1] \times [0, 1]$ into $R_0 = \{ (0, 1), (1, 0) \}$, $R_1 = \{ (p, q) \mid p \in \{0, 1\}, 0 < q < 1 \}$, $R_2 = \{ (p, q) \mid p, q \in (0, 1), p \neq q \}$, $R_3 = {R_1}^{\mathrm{op}}$, and $\mathrm{Eq}_{\mathcal{D}_{=1}2}$.

By using Lemma 1.1, we obtain Lemma 5.1 and 5.2.



Lemma 5.1 *Let $\preceq \in \mathbf{CSPre}(\mathcal{D}_{=1}, 2)$. We obtain the following properties:*

(i) *$p \preceq q$ for some $0 < p < q < 1$ if and only if $r \preceq s$ for all $r, s \in (0, 1)$. This is equivalent to $R_2 \cap \preceq \neq \emptyset \implies R_2 \subseteq \preceq$.*

(ii) *$0 \preceq q$ for some $0 < q < 1$ if and only if $r \preceq s$ for all $(r, s) \in [0, 1] \times (0, 1)$. This is equivalent to $R_1 \cap \preceq \neq \emptyset \implies R_1 \cup R_2 \subseteq \preceq$.*

(iii) *$0 \preceq 1$ if and only if $r \preceq s$ for all $r, s \in [0, 1]$. This is equivalent to $R_0 \cap \preceq \neq \emptyset \implies R_0 \cup R_1 \cup R_2 \cup R_3 \subseteq \preceq$.*

Lemma 5.2 *Let $\preceq \in \mathbf{CSPre}(\mathcal{D}_{=1}, 2)$. We obtain $\preceq = \mathrm{Eq}_{\mathcal{D}_{=1}2} \cup \bigcup_{i \in I} R_i$ where $I = \{\, i \in \{0, 1, 2, 3\} \mid R_i \cap \preceq \neq \emptyset \,\}$.*

Proposition 5.3 . *We have the following identification:*

$$\mathbf{CSPre}(\mathcal{D}_{=1}, 2) = \mathcal{C}_{\cap,\mathrm{op}}\{\preceq^s, \mathrm{Eq}_{\mathcal{D}_{=1}2}\} = \{\top_{\mathcal{D}_{=1}2}, \mathrm{Eq}_{\mathcal{D}_{=1}2}, \preceq^s, \preceq^{s\mathrm{op}}, \preceq^s \cap \preceq^{s\mathrm{op}}\} \cong 5,$$

where $p \preceq^s q \overset{\mathrm{def}}{\iff} (p \neq q) \implies (0 < q < 1)$.

Proof (Sketch). Analogous to Lemma 4.4 with $R(p_0, p_1, p_2, p_3) = \mathrm{Eq}_{\mathcal{D}_{=1}2} \cup \bigcup_{i \in I} \{\, R_i \mid p_i = \mathrm{true} \,\}$ and the following formula:

$$P = (p_0 \implies p_1 \wedge p_2 \wedge p_3) \wedge (p_1 \implies p_2) \wedge (p_1 \wedge p_3 \implies p_0) \wedge (p_3 \implies p_2)$$

that is obtained from Lemma 5.1 and 5.2 and the transitivity of $\preceq$. There are exactly 5 satisfying assignments (p_0, p_1, p_2, p_3) of P. We then prove by hand,

$$5 \cong \{\, R(p_0, p_1, p_2, p_3) \mid (p_0, p_1, p_2, p_3) \text{ satisfies } P \,\} \subseteq \mathcal{C}_{\cap,\mathrm{op}}\{\preceq^s, \mathrm{Eq}_{\mathcal{D}_{=1}2}\}.$$

Since $\mathbf{CSPre}(\mathcal{D}_{=1}, 2) \subseteq \{\, R(p_0, p_1, p_2, p_3) \mid (p_0, p_1, p_2, p_3) \text{ satisfies } P \,\}$ and $\preceq^s, \mathrm{Eq}_{\mathcal{D}_{=1}2} \in \mathbf{CSPre}(\mathcal{D}_{=1}, 2)$, we conclude this proposition. □

Next, we calculate the mapping $[-]^2 \colon \mathbf{CSPre}(\mathcal{D}_{=1}, 2) \to \mathbf{Pre}(\mathcal{D}_{=1})$. Since it preserves intersections and opposites, and $\mathbf{CSPre}(\mathcal{D}_{=1}, 2) = \mathcal{C}_{\cap,\mathrm{op}}\{\preceq^s, \mathrm{Eq}_{\mathcal{D}_{=1}2}\}$, it suffices to identify the preorders $[\mathrm{Eq}_{\mathcal{D}_{=1}2}]^2$ and $[\preceq^s]^2$ (we denote it by $\sqsubseteq^s$).

Proposition 5.4 *The preorders $[\mathrm{Eq}_{\mathcal{D}_{=1}2}]^2$ and $\sqsubseteq^s$ are identified as follows:*

(i) *$d_1 \, [\mathrm{Eq}_{\mathcal{D}_{=1}2}]^2_X \, d_2 \iff d_1 = d_2$.*

(ii) *$d_1 \sqsubseteq^s_X d_2 \iff \mathrm{supp}(d_1) \subseteq \mathrm{supp}(d_2)$.*

Next, we calculate the mapping $\langle - \rangle^2 : \mathbf{CSPre}(\mathcal{D}_{=1}, 2) \to \mathbf{Pre}(\mathcal{D}_{=1})$.

Lemma 5.5 *Let $\preceq \in \mathbf{CSPre}(\mathcal{D}_{=1}, 2)$ and $\alpha \in [0, 1]$. If $d_1, d_2 \in \mathcal{D}_{=1}X$ satisfy the following condition: for each $y \in X$ such that $d_1(y) > d_2(y)$,*

$$\left(\alpha + (1 - \alpha)\frac{d_2(y)}{d_1(y)}\right)\delta_{\mathbf{0}} + (1 - \alpha)\left(1 - \frac{d_2(y)}{d_1(y)}\right)\delta_{\mathbf{1}} \preceq \frac{d_2(y)}{d_1(y)}\delta_{\mathbf{0}} + \left(1 - \frac{d_2(y)}{d_1(y)}\right)\delta_{\mathbf{1}}$$

then $(\alpha d_1 + (1 - \alpha) d_2) \, \langle \preceq \rangle^2_X \, d_2$ holds.

Proposition 5.6 *The mapping $\langle - \rangle^2$ equals the mapping $[-]^2$.*

Proof (Sketch). We prove the case $\preceq \,=\, \preceq^s \cap \preceq^{s\mathrm{op}}$, and omit the other cases. (Case: $\preceq \,=\, \preceq^s \cap \preceq^{s\mathrm{op}}$) Suppose $d_1 [\preceq]^2_X d_2$. By Lemma 5.4, it is equivalent to $\mathrm{supp}(d_1) = \mathrm{supp}(d_2)$. This implies for each $y \in X$ such that $d_1(y) > d_2(y)$,

$$\left(\frac{d_1(y) + d_2(y)}{2d_1(y)} \delta_{\mathbf{0}} + \frac{d_1(y) - d_2(y)}{2d_1(y)} \delta_{\mathbf{1}} \right) \preceq \left(\frac{d_2(y)}{d_1(y)} \delta_{\mathbf{0}} + \frac{d_1(y) - d_2(y)}{d_1(y)} \delta_{\mathbf{1}} \right).$$

By Lemma 5.5 with $\alpha = 1/2$, we obtain $(d_1 + d_2)/2 \, \langle \preceq \rangle^2_X \, d_2$. Similarly, we also have $d_1 \, \langle \preceq \rangle^2_X \, (d_1 + d_2)/2$. Thus, $d_1 \, \langle \preceq \rangle^2_X \, d_2$. Therefore, $[\preceq]^2 = \langle \preceq \rangle^2$ holds. □

Theorem 5.7 (Theorem 3.5(ii)) *We obtain the following identification:*

$$\mathbf{Pre}(\mathcal{D}_{=1}) = \mathcal{C}_{\cap,\mathrm{op}}\{\sqsubseteq^s, \mathrm{Eq}^{\mathcal{D}_{=1}}\} = \{\top^{\mathcal{D}_{=1}}, \mathrm{Eq}^{\mathcal{D}_{=1}}, \sqsubseteq^s, \sqsubseteq^{s\mathrm{op}}, \sqsubseteq^s \cap \sqsubseteq^{s\mathrm{op}}\} \cong 5.$$

Proof. It is proved from Lemma 3.4, Proposition 5.4, 5.3, and 5.6. □

6 Preorders on the Subdistribution Monad

First, we idenfity $\mathbf{CSPre}(\mathcal{D}, 1)$. We regard $\mathcal{D}1 \cong [0,1]$ by the correspondence of each $d \in \mathcal{D}1$ with the value $d(*) \in [0,1]$. For each $\preceq \in \mathbf{CSPre}(\mathcal{D}, 1)$, the substitutivity of $\preceq$ is equivalent to

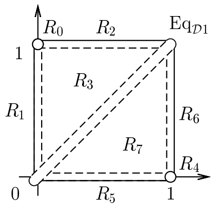
$$(p \preceq q \wedge t \in [0,1]) \implies tp \preceq tq,$$

and the congruence of $\preceq$ is equivalent to

$$\forall i \in I.(p_i \preceq q_i) \wedge \textstyle\sum_{i \in I} t_i \leq 1 \implies \sum_{i \in I} p_i t_i \preceq \sum_{i \in I} q_i t_i.$$

Hence, each $\preceq \in \mathbf{CSPre}(\mathcal{D}, 1)$ is preserved by convex combinations.

We partition the set $\mathcal{D}1 \times \mathcal{D}1 \cong [0,1] \times [0,1]$ into $R_0 = \{(0,1)\}$, $R_1 = \{\, (0,q) \mid 0 < q < 1 \,\}$, $R_2 = \{\, (p,1) \mid 0 < p < 1 \,\}$, $R_3 = \{\, (p,q) \mid 0 < p < q < 1 \,\}$, $R_4 = {R_0}^{\mathrm{op}}$, $R_5 = {R_1}^{\mathrm{op}}$, $R_6 = {R_2}^{\mathrm{op}}$, $R_7 = {R_3}^{\mathrm{op}}$, and $\mathrm{Eq}_{\mathcal{D}1}$.



By using Lemma 1.1, we obtain Lemma 6.1 and 6.2.

Lemma 6.1 *Let $\preceq \in \mathbf{CSPre}(\mathcal{D}, 1)$. We obtain the following properties:*

(i) *$p \preceq q$ for some $0 < p < q < 1$ if and only if $r \preceq s$ for all $0 < r < s < 1$.*
This is equivalent to $R_3 \cap \preceq \neq \emptyset \implies R_3 \subseteq \preceq$.

(ii) *$0 \preceq q$ for some $0 < q < 1$ if and only if $r \preceq s$ for all $0 \leq r < s < 1$.*
This is equivalent to $R_1 \cap \preceq \neq \emptyset \implies R_1 \cup R_3 \subseteq \preceq$.

(iii) *$p \preceq 1$ for some $0 < p < 1$ if and only if $r \preceq s$ for all $0 < r < s \leq 1$.*
This is equivalent to $R_2 \cap \preceq \neq \emptyset \implies R_2 \cup R_3 \subseteq \preceq$.

(iv) *$0 \preceq 1$ if and only if $r \preceq s$ for all $0 \leq r < s \leq 1$.*
This is equivalent to $R_0 \cap \preceq \neq \emptyset \implies R_0 \cup R_1 \cup R_2 \cup R_3 \subseteq \preceq$.

Lemma 6.2 *Let $\preceq \in \mathbf{CSPre}(\mathcal{D}, 1)$. We obtain $\preceq \,=\, \mathrm{Eq}_{\mathcal{D}1} \cup \bigcup_{i \in I} R_i$ where $I = \{\, i \in \{0, 1, \ldots, 7\} \mid R_i \cap \preceq \neq \emptyset \,\}$.*

We prepare the following congruent substitutive preorders on $\mathcal{D}1$:

- $p \preceq^r q \overset{\text{def}}{\iff} p \leq q$
- $p \preceq^s q \overset{\text{def}}{\iff} p > 0 \implies q > 0$
- $p \preceq^d q \overset{\text{def}}{\iff} p = 1 \implies q = 1$

The superscripts r, s, and d stand for real values, supports, and deadlocks of distributions respectively. We let $\preceq^{sd} = \preceq^s \cap \preceq^d$ for simplicity.

Proposition 6.3 *We obtain* $\mathbf{CSPre}(\mathcal{D}, 1) = \mathcal{C}_{\cap,\mathrm{op}}\{\preceq^r, \preceq^s, \preceq^d\} \cong 25$.

Proof (Sketch). Analogous to Lemma 4.4 with $R(p_0, p_1, \dots, p_7) = \mathrm{Eq}_{\mathcal{D}1} \cup \bigcup_{i \in I} \{\, R_i \mid p_i = \text{true} \,\}$ and the following formula: $P = P' \wedge P''$ where

$$\begin{aligned} P' &= (p_0 \iff (p_1 \wedge p_2)) \wedge ((p_1 \vee p_2) \implies p_3), \\ P'' &= (p_4 \iff (p_5 \wedge p_6)) \wedge ((p_5 \vee p_6) \implies p_7). \end{aligned}$$

There are 25 satisfying assignments $(p_0, p_1, \dots, p_7)$ of P. We prove by hand,

$$25 \cong \{\, R(p_0, p_1, \dots, p_7) \mid (p_0, p_1, \dots, p_7) \text{ satisfies } P \,\} \subseteq \mathcal{C}_{\cap,\mathrm{op}}\{\preceq^r, \preceq^s, \preceq^d\}.$$

Since $\mathbf{CSPre}(\mathcal{D}, 1) \subseteq \{\, R(p_0, p_1, \dots, p_7) \mid (p_0, p_1, \dots, p_7) \text{ satisfies } P \,\}$ and $\preceq^r, \preceq^s, \preceq^d \in \mathbf{CSPre}(\mathcal{D}, 1)$, we conclude this proposition. □

Next, we calculate the mapping $[-]^1 \colon \mathbf{CSPre}(\mathcal{D}, 1) \to \mathbf{Pre}(\mathcal{D})$. Since it preserves intersections and opposites, and $\mathbf{CSPre}(\mathcal{D}, 1) = \mathcal{C}_{\cap,\mathrm{op}}\{\preceq^r, \preceq^s, \preceq^d\}$ holds, it suffices to identify the preorders $[\preceq^r]^1$, $[\preceq^s]^1$, and $[\preceq^d]^1$ (e.g. $[\preceq^d \cap \preceq^{s\mathrm{op}}]^1 = [\preceq^d]^1 \cap [\preceq^s]^{1\mathrm{op}}$). Let $\sqsubseteq^r = [\preceq^r]^1$, $\sqsubseteq^s = [\preceq^s]^1$, and $\sqsubseteq^d = [\preceq^d]^1$.

Proposition 6.4 *The preorders* $\sqsubseteq^r$, $\sqsubseteq^s$, *and* $\sqsubseteq^d$ *are identified as follows:*

(i) $d_1 \sqsubseteq^r_X d_2 \iff \forall x \in X. d_1(x) \leq d_2(x)$.

(ii) $d_1 \sqsubseteq^s_X d_2 \iff \mathrm{supp}(d_1) \subseteq \mathrm{supp}(d_2)$.

(iii) $d_1 \sqsubseteq^d_X d_2 \iff (d_1[X] = 1 \implies d_2[\mathrm{supp}(d_1)] = 1)$.

Next, we calculate the mapping $\langle - \rangle^1 : \mathbf{CSPre}(\mathcal{D}, 1) \to \mathbf{Pre}(\mathcal{D})$. Generally speaking, $\langle - \rangle^I : \mathbf{CSPre}(T, I) \to \mathbf{Pre}(T)$ needs not preserve intersections, but the mapping $\langle - \rangle^1 : \mathbf{CSPre}(\mathcal{D}, 1) \to \mathbf{Pre}(\mathcal{D})$ preserves intersections.

Proposition 6.5 *The mapping* $\langle - \rangle^1$ *satisfies the following:*

- *The mapping* $\langle - \rangle^1$ *preserves intersections and opposites.*
- $\langle \preceq^r \rangle^1 = \sqsubseteq^r$, $\langle \preceq^s \rangle^1 = \sqsubseteq^s$, *and* $\left\langle \preceq^d \right\rangle^1 = \sqsubseteq^m$ *where* $\sqsubseteq^m$ *is defined by*

$$d_1 \sqsubseteq^m_X d_2 \overset{\text{def}}{\iff} (d_1[X] = 1 \implies d_1 = d_2).$$

By Proposition 6.3 and 6.5, we obtain that the preorder $\langle \preceq \rangle^1$ is identified completely for each $\preceq \in \mathbf{CSPre}(\mathcal{D}, 1)$ (e.g. $\left\langle \preceq^d \cap \preceq^{s\mathrm{op}} \right\rangle^1 = \sqsubseteq^m \cap \sqsubseteq^{s\mathrm{op}}$).

The following lemma and is crucial to identify the mapping $\langle - \rangle^1$.

Lemma 6.6 *Let* $\preceq \in \mathbf{CSPre}(\mathcal{D}, 1)$*. If* $d_1, d_2 \in \mathcal{D}X$ *satisfy the condition:*

$$\forall x \in \mathrm{supp}(d_1).\left(\frac{1 + d_1[X]}{2} \preceq \frac{(1 + d_1[X])}{2}\frac{\min(d_1, d_2)(x)}{d_1(x)}\right) \qquad (1)$$

then we obtain $d_1 \langle\preceq\rangle^1_X \min(d_1, d_2)$*.*

Here, $\min(d_1, d_2) \in \mathcal{D}X$ is defined by $\min(d_1, d_2)(x) = \min(d_1(x), d_2(x))$.

Proof of Proposition 6.5 (Sketch). First, we prove $\sqsubseteq^m \in \mathbf{Pre}(\mathcal{D})$. Since $\sqsubseteq^m_1 = \preceq^d$, the image of the mapping $(-)_1$ under $\mathcal{C}_{\cap,\mathrm{op}}\{\sqsubseteq^r, \sqsubseteq^s, \sqsubseteq^m\}$ is $\mathbf{CSPre}(\mathcal{D}, 1)$. Next, we prove $d_1 \langle\sqsubseteq_1\rangle^1_X \min(d_1, d_2) \langle\sqsubseteq_1\rangle^1_X d_2$ for each $d_1 \sqsubseteq_X d_2$ by applying Lemma 6.6 for each $\sqsubseteq \in \mathcal{C}_{\cap,\mathrm{op}}\{\sqsubseteq^r, \sqsubseteq^s, \sqsubseteq^m\}$. □

To finish identifying the class $\mathbf{Pre}(\mathcal{D})$, we search for a preorder $\sqsubseteq$ on $\mathcal{D}$ such that $\langle\preceq\rangle^1 \ntrianglelefteq \sqsubseteq \ntrianglelefteq [\preceq]^1$ for each $\preceq \in \mathbf{CSPre}(\mathcal{D}, 1)$. Here, $\langle\preceq\rangle^1$ and $[\preceq]^1$ are the least and greatest preorder on $\mathcal{D}$ whose evaluations at 1 equal $\preceq$.

Proposition 6.7 *Let* $\preceq \in \mathbf{CSPre}(\mathcal{D}, 1)$*. If* $\preceq$ *is one of* $\preceq^d$*,* $\preceq^{d\mathrm{op}}$*,* $\preceq^d \cap \preceq^{s\mathrm{op}}$*, and* $\preceq^{d\mathrm{op}} \cap \preceq^s$*, a preorder* $\sqsubseteq \in \mathbf{Pre}(\mathcal{D})$ *such that* $\langle\preceq\rangle^1 \ntrianglelefteq \sqsubseteq \ntrianglelefteq [\preceq]^1$ *is determined* uniquely *as follows:*

$$\begin{aligned} \preceq = \preceq^d &\implies \sqsubseteq = \sqsubseteq^M, & \preceq = \preceq^{d\mathrm{op}} &\implies \sqsubseteq = \sqsubseteq^{M\mathrm{op}}, \\ \preceq = \preceq^d \cap \preceq^{s\mathrm{op}} &\implies \sqsubseteq = \sqsubseteq^M \cap \sqsubseteq^{s\mathrm{op}}, & \preceq = \preceq^{d\mathrm{op}} \cap \preceq^s &\implies \sqsubseteq = \sqsubseteq^{M\mathrm{op}} \cap \sqsubseteq^s, \end{aligned}$$

where, the preorder $\sqsubseteq^M \in \mathbf{Pre}(\mathcal{D})$ *is defined by*

$$d_1 \sqsubseteq_X d_2 \overset{\text{def}}{\iff} (d_1[X] = 1 \implies (d_2[X] = 1 \wedge \mathrm{supp}(d_1) = \mathrm{supp}(d_2))).$$

Otherwise, a preorder $\sqsubseteq \in \mathbf{Pre}(\mathcal{D})$ *such that* $\langle\preceq\rangle^1 \ntrianglelefteq \sqsubseteq \ntrianglelefteq [\preceq]^1$ *does not exist.*

Let $\tau \colon \mathcal{D}_{=1} \Rightarrow \mathcal{D}$ be the natural transformation defined by $\tau_X(d) = d$ for each $d \in \mathcal{D}_{=1}X$. For each $\sqsubseteq \in \mathbf{Pre}(\mathcal{D})$, we define the restriction $C(\sqsubseteq)$ of $\sqsubseteq$ by

$$C(\sqsubseteq)_X = \{\, (d_1, d_2) \mid \tau_X(d_1) \sqsubseteq_X \tau_X(d_2) \,\} \subseteq \mathcal{D}_{=1}X \times \mathcal{D}_{=1}X.$$

The following lemma shows that the restriction $C(-)$ is a monotone mapping from $(\mathbf{Pre}(\mathcal{D}), \trianglelefteq)$ to $(\mathbf{Pre}(\mathcal{D}_{=1}), \trianglelefteq)$ since the monotonicity of C is obvious.

Lemma 6.8 *For each* $\sqsubseteq \in \mathbf{Pre}(\mathcal{D})$*,* $C(\sqsubseteq)$ *is indeed a preorder on* $\mathcal{D}_{=1}$*.*

Lemma 6.9 *Let* $\preceq \in \mathbf{CSPre}(\mathcal{D}, 1)$ *and* $\sqsubseteq \in \mathbf{Pre}(\mathcal{D})$ *with* $\langle\preceq\rangle^1 \ntrianglelefteq \sqsubseteq \ntrianglelefteq [\preceq]^1$*.*

(i) $(d_1[X] < 1 \vee d_2[X] < 1) \implies (d_1\, [\preceq]^1_X\, d_2 \iff d_1 \sqsubseteq_X d_2 \iff d_1 \langle\preceq\rangle^1_X d_2)$

(ii) $C(\langle\preceq\rangle^1) \ntrianglelefteq C(\sqsubseteq) \ntrianglelefteq C([\preceq]^1)$

Hence, each preorder $\sqsubseteq \in \mathbf{Pre}(\mathcal{D})$ such that $\langle\preceq\rangle^1 \ntrianglelefteq \sqsubseteq \ntrianglelefteq [\preceq]^1$ is determined by preorders on $\mathcal{D}_{=1}$ between $C(\langle\preceq\rangle^1)$ and $C([\preceq]^1)$ and the preorder $[\preceq]^1$.

Proof of Proposition 6.7 (Sketch). In the first 4 cases, $C(\langle\preceq\rangle^1) = \mathrm{Eq}^{\mathcal{D}_{=1}}$ and $C([\preceq]^1) \in \{\sqsubseteq^s, \sqsubseteq^{s\mathrm{op}}\}$. Thus, $C(\sqsubseteq) = \sqsubseteq^s \cap \sqsubseteq^{s\mathrm{op}}$ by Lemma 6.9 (ii). Hence, the preorder $\sqsubseteq$ is determined uniquely by Lemma 6.9 (i). Otherwise, $\langle\preceq\rangle^1 \underset{\neq}{\vartriangleleft} \sqsubseteq \underset{\neq}{\vartriangleleft} [\preceq]^1$ contradicts Lemma 6.9 (ii) since $C([\preceq]^1) = \sqsubseteq^s \cap \sqsubseteq^{s\mathrm{op}}$. □

We have finished identifying $\mathbf{Pre}(\mathcal{D})$.

Theorem 6.10 (Theorem 3.5(iii)) *The class* $\mathbf{Pre}(\mathcal{D})$ *is identified as Table 1 below. Moreover, we obtain* $\mathbf{Pre}(\mathcal{D}) = \mathcal{C}_{\cap,\mathrm{op}}\{\sqsubseteq^r, \sqsubseteq^s, \sqsubseteq^d, \sqsubseteq^m, \sqsubseteq^M\} \cong 41$.

$\preceq\, \in \mathbf{CSPre}(\mathcal{D}, 1)$	$\sqsubseteq\, \in \mathbf{Pre}(\mathcal{D})$ such that $\sqsubseteq_1 = \preceq$
$\top_{\mathcal{D}1}$	$\top^{\mathcal{D}}$
$\mathrm{Eq}_{\mathcal{D}1}$	$\mathrm{Eq}^{\mathcal{D}}$
$\preceq^r$	$\sqsubseteq^r$
$\preceq^r \cap \preceq^{s\mathrm{op}}$	$\sqsubseteq^r \cap \sqsubseteq^{s\mathrm{op}}$
$\preceq^r \cap \preceq^{d\mathrm{op}}$	$\sqsubseteq^r \cap \sqsubseteq^{d\mathrm{op}}$
$\preceq^r \cap \preceq^{s\mathrm{op}} \cap \preceq^{d\mathrm{op}}$	$\sqsubseteq^r \cap \sqsubseteq^{s\mathrm{op}} \cap \sqsubseteq^{d\mathrm{op}}$
$\preceq^s$	$\sqsubseteq^s$
$\preceq^s \cap \preceq^{s\mathrm{op}}$	$\sqsubseteq^s \cap \sqsubseteq^{s\mathrm{op}}$
$\preceq^d \cap \preceq^s$	$\sqsubseteq^m \cap \sqsubseteq^s,\ \sqsubseteq^d \cap \sqsubseteq^s$
$\preceq^d \cap \preceq^{d\mathrm{op}}$	$\sqsubseteq^m \cap \sqsubseteq^{m\mathrm{op}},\ \sqsubseteq^d \cap \sqsubseteq^{d\mathrm{op}}$
$\preceq^d \cap \preceq^{d\mathrm{op}} \cap \preceq^s$	$\sqsubseteq^m \cap \sqsubseteq^{m\mathrm{op}} \cap \sqsubseteq^s,\ \sqsubseteq^d \cap \sqsubseteq^{d\mathrm{op}} \cap \sqsubseteq^s$
$\preceq^d \cap \preceq^s \cap \preceq^{s\mathrm{op}}$	$\sqsubseteq^m \cap \sqsubseteq^s \cap \sqsubseteq^{s\mathrm{op}},\ \sqsubseteq^d \cap \sqsubseteq^s \cap \sqsubseteq^{s\mathrm{op}}$
$\preceq^d \cap \preceq^{d\mathrm{op}} \cap \preceq^s \cap \preceq^{s\mathrm{op}}$	$\sqsubseteq^m \cap \sqsubseteq^{m\mathrm{op}} \cap \sqsubseteq^s \cap \sqsubseteq^{s\mathrm{op}},\ \sqsubseteq^d \cap \sqsubseteq^{d\mathrm{op}} \cap \sqsubseteq^s \cap \sqsubseteq^{s\mathrm{op}}$
$\preceq^d$	$\sqsubseteq^m,\ \sqsubseteq^M,\ \sqsubseteq^d$
$\preceq^d \cap \preceq^{s\mathrm{op}}$	$\sqsubseteq^m \cap \sqsubseteq^{s\mathrm{op}},\ \sqsubseteq^M \cap \sqsubseteq^{s\mathrm{op}},\ \sqsubseteq^d \cap \sqsubseteq^{s\mathrm{op}}$

Table 1
The table of $\mathbf{CSPre}(\mathcal{D}, 1)$ and $\mathbf{Pre}(\mathcal{D})$ (we omit opposite preorders)

Proof. It is proved immediately from Proposition 6.3, 6.4, 6.5, and 6.7. □

The next lemma tells that $\mathbf{CSPre}(\mathcal{D}, 2)$ is enough to identify $\mathbf{Pre}(\mathcal{D})$.

Theorem 6.11 *We obtain* $\mathbf{Pre}(\mathcal{D}) \cong \mathbf{CSPre}(\mathcal{D}, 2)$.

Proof (Sketch). By Lemma 3.3, it suffices to prove $\sqsubseteq_2 \neq \sqsubseteq'_2$ whenever $\sqsubseteq \neq \sqsubseteq'$. By [9, Lemma 3], it suffices to compare $\sqsubseteq$ and $\sqsubseteq'$ such that $\sqsubseteq_1 = \sqsubseteq'_1$. □

7 Coalgebraic Simulations between Markov Chains

Simulations between coalgebras are defined coalgebraically by using *relational liftings* of coalgebra functors. In this section, we focus on simulations between Markov chains (i.e. $\mathcal{D}$-coalgebras). We now discuss relational liftings of $\mathcal{D}$ that are constructed from preorders on $\mathcal{D}$ by the following two methods.

The first method is given in [6,8]. For a given preorder $\sqsubseteq\, \in \mathbf{Pre}(\mathcal{D})$, we construct the relational lifting $\overline{\mathcal{D}}^{(\sqsubseteq)}$ of $\mathcal{D}$ by

$$\overline{\mathcal{D}}^{(\sqsubseteq)}(R) = \sqsubseteq_X \circ \{\, (\mathcal{D}\pi_1(d), \mathcal{D}\pi_2(d)) \in \mathcal{D}X \times \mathcal{D}Y \mid d \in \mathcal{D}(R) \,\} \circ \sqsubseteq_Y$$

where $\pi_1 \colon R \to X$ and $\pi_2 \colon R \to Y$ are projections from a relation $R \subseteq X \times Y$.

The second method called the preorder $\top\top$-lifting is defined in [9]. For a given $\preceq \in \mathbf{CSPre}(\mathcal{D}, I)$, we define the relational lifting $\mathcal{D}^{\top\top(\preceq)}$ of $\mathcal{D}$ by

$$\begin{aligned} \mathcal{D}^{\top(\preceq)}(R) &= \{\, (f\colon X \to \mathcal{D}I, g\colon Y \to \mathcal{D}I) \mid \forall (x,y) \in R. f(x) \preceq g(y) \,\} \\ \mathcal{D}^{\top\top(\preceq)}(R) &= \left\{\, (d_1, d_2) \in \mathcal{D}X \times \mathcal{D}Y \;\middle|\; \forall (f,g) \in \mathcal{D}^{\top(\preceq)}(R). f^\sharp(d_1) \preceq g^\sharp(d_2) \,\right\}. \end{aligned}$$

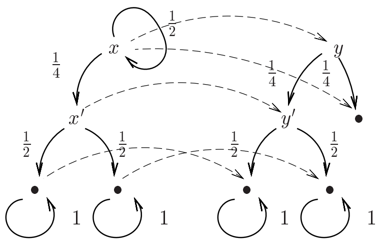
We relate earlier studies of (bi)simulations between probabilistic transition systems to the coalgebraic simulations obtained by the above methods:

- $\mathcal{D}^{\top\top(\mathrm{Eq}_{\mathcal{D}1})}$-simulation is equivalent to the generalised definition of probabilistic bisimulation [10] given at [2, Definition 4.2]. At [2, Theorem 4.5], assuming z-closedness [2, Section 2], the generalised definition is equivalent to $\overline{\mathcal{D}}^{(\mathrm{Eq}^{\mathcal{D}})}$-simulation (i.e. $\mathcal{D}$-bisimulation [2, Section 3]).
- We obtain $\mathcal{D}^{\top\top(\preceq^r)}(R) = \{\, (d_1, d_2) \mid \forall U \subseteq X. d_1[U] \leq d_2[R(U)] \,\}$. Hence, the probabilistic bisimulation defined in [16] is equivalent to $\mathcal{D}^{\top\top(\preceq^r)}(-) \cap (\mathcal{D}^{\top\top(\preceq^r)}(-^{\mathrm{op}}))^{\mathrm{op}}$-simulation [12]. Moreover, the relational lifting for probabilistic simulations defined at [1, Definition 3.6] equals $(\mathcal{D}^{\top\top(\preceq^r)}(-^{\mathrm{op}}))^{\mathrm{op}}$.
- Jonsson-Larsen simulations over Markov chains are $\overline{\mathcal{D}}^{(\sqsubseteq^r)}$-simulations [4].
- $\overline{\mathcal{D}}^{(\sqsubseteq^s)}$-simulations, $\mathcal{D}^{\top\top(\preceq^s)}$-simulations, and *support simulation* that are defined as follows are equivalent: R is a support (bi)simulation between Markov chains (X, ξ) and (Y, ξ') if R is a (bi)simulation between two $\mathcal{P}$-colagebras $(X, \mathrm{supp} \circ \xi)$ and $(Y, \mathrm{supp} \circ \xi')$ in the standard sense.
- $\overline{\mathcal{D}}^{(\sqsubseteq^s \cap \sqsubseteq^{s\,\mathrm{op}})}$-simulations and $\mathcal{D}^{\top\top(\preceq^s \cap \preceq^{s\,\mathrm{op}})}$-simulations are equivalent to support bisimulations and support bisimulations up to z-closedness respectively.

We apply the preorders $\sqsubseteq^m$, $\sqsubseteq^M$, and $\sqsubseteq^d$ on $\mathcal{D}$ to the construction $\overline{\mathcal{D}}^{(-)}$. For two Markov chains (X, ξ) and (Y, ξ'), a binary relation $R \subseteq X \times Y$ is:

- a $\overline{\mathcal{D}}^{(\sqsubseteq^m)}$-simulation if and only if
$$(x,y) \in R \wedge \xi(x)[X] = 1 \implies (\xi(x), \xi'(y)) \in \overline{\mathcal{D}}^{(\mathrm{Eq}^{\mathcal{D}})}(R).$$
This is seen as a probabilistic bisimulation *ignoring states with deadlocks.*
- a $\overline{\mathcal{D}}^{(\sqsubseteq^M)}$-simulation if and only if
$$(x,y) \in R \wedge \xi(x)[X] = 1 \implies (\xi(x), \xi'(y)) \in \overline{\mathcal{D}}^{(\sqsubseteq^s \cap \sqsubseteq^{s\,\mathrm{op}})}(R).$$
This is seen as a support-bisimulation ignoring states with deadlocks.
- a $\overline{\mathcal{D}}^{(\sqsubseteq^d)}$-simulation if and only if
$$(x,y) \in R \wedge \xi(x)[X] = 1 \implies (\xi(x), \xi'(y)) \in \overline{\mathcal{D}}^{(\sqsubseteq^{s\,\mathrm{op}})}(R).$$
This is seen as a reverse support simulation ignoring states with deadlocks.

We give an example of $\overline{\mathcal{D}}^{(\sqsubseteq^m)}$-simulation. We consider two Markov chains (X, ξ) and (Y, ξ') and their start states $x \in X$ and $y \in Y$ as in the right figure. The dashed arrows are a $\overline{\mathcal{D}}^{(\sqsubseteq^m)}$-simulation R between x and y. First, since the state x has a deadlock, the states x and y are assumed to be probabilistic bisimilar unconditionally. Next, since transitions started from the state x' has no deadlock, the state y' must be probabilistic bisimilar to the state x' in the sense of $\overline{\mathcal{D}}^{(\mathrm{Eq}^{\mathcal{D}})}$-(bi)similarity.



8 Future Work

We have the following future work at this time:

- We expect to analyse preorders on other monads. For example, the convex module monad $\mathcal{CM}$ [7,15] that captures discrete probabilistic branching *combined with nondeterminism.*
- We expect to obtain preorders on the composite monad ST of monads S and T by using a distributive law $\delta\colon TS \Rightarrow ST$ from preorders on S and T.

Acknowledgement

The author thanks to Shin-ya Katsumata for valuable comments and stimulating discussions, Paul Blain Levy for constructive suggestions, and Naohiko Hoshino, Ichiro Hasuo, and Corina Cîrstea for helpful comments.

References

[1] Corina Cîrstea. On logics for coalgebraic simulation. *Electron. Notes Theor. Comput. Sci.*, 106:63–90, December 2004.

[2] E.P de Vink and J.J.M.M Rutten. Bisimulation for probabilistic transition systems: a coalgebraic approach. *Theoretical Computer Science*, 221(1 - 2):271 – 293, 1999.

[3] Samuel Eilenberg. *Automata, languages, and machines.* Pure and Applied Mathematics. Elsevier Science, 1974.

[4] Ichiro Hasuo. Generic forward and backward simulations ii: Probabilistic simulation. In Paul Gastin and François Laroussinie, editors, *CONCUR 2010 - Concurrency Theory*, volume 6269 of *Lecture Notes in Computer Science*, pages 447–461. Springer Berlin Heidelberg, 2010.

[5] Ichiro Hasuo, Bart Jacobs, and Ana Sokolova. Generic trace semantics via coinduction. *Logical Methods in Computer Science*, 3(4), 2007.

[6] Wim H. Hesselink and Albert Thijs. Fixpoint semantics and simulation. *Theor. Comp. Sci*, 238:200–0, 2000.

[7] Bart Jacobs. Coalgebraic trace semantics for combined possibilitistic and probabilistic systems, 2008. Proceedings of the Ninth Workshop on Coalgebraic Methods in Computer Science (CMCS 2008).

[8] Bart Jacobs and Jesse Hughes. Simulations in coalgebra. *Electronic Notes in Theoretical Computer Science*, 82(1):128–149, 2003. CMCS'03, Coalgebraic Methods in Computer Science (Satellite Event for ETAPS 2003).

[9] Shin-ya Katsumata and Tetsuya Sato. Preorders on monads and coalgebraic simulations. In Frank Pfenning, editor, *Foundations of Software Science and Computation Structures*, volume 7794 of *Lecture Notes in Computer Science*, pages 145–160. Springer Berlin Heidelberg, 2013.

[10] Kim Guldstrand Larsen and Arne Skou. Bisimulation through probabilistic testing. *Information and Computation*, 94(1):1–28, 1991.

[11] Paul Blain Levy. Boolean precongruences, 2009. Manuscript.

[12] Paul Blain Levy. Similarity quotients as final coalgebras. In Martin Hofmann, editor, *Foundations of Software Science and Computational Structures*, volume 6604 of *Lecture Notes in Computer Science*, pages 27–41. Springer Berlin Heidelberg, 2011.

[13] Roberto Segala. A compositional trace-based semantics for probabilistic automata. In Insup Lee and Scott Smolka, editors, *CONCUR '95: Concurrency Theory*, volume 962 of *Lecture Notes in Computer Science*, pages 234 – 248. Springer Berlin / Heidelberg, 1995. 10.1007/3-540-60218-6_17.

[14] Ana Sokolova. Probabilistic systems coalgebraically: A survey. *Theor. Comput. Sci.*, 412(38):5095–5110, 2011.

[15] Daniele Varacca and Glynn Winskel. Distributing probability over non-determinism. *Mathematical Structures in Computer Science*, 16(1):87–113, 2006.

[16] James Worrell. Coinduction for recursive data types: partial orders, metric spaces and omega-categories. *Electr. Notes Theor. Comput. Sci.*, 33:337–356, 2000.