In this lecture and the next we develop a coalgebraic theory of KAT, which we call Kleene coalgebra with tests (KCT). Our treatment includes a definition of the Brzozowski derivative in the context of an automata-theoretic formulation of KAT involving automata on guarded strings [6]. The syntactic form of the Brzozowski derivative applies to all KAT expressions as defined in [5]. This treatment places KCT within the general coalgebraic framework described by Bonsangue, Rutten, and Silva [1, 2]. A somewhat different approach is given by Chen and Pucella [4].

We also give a complexity analysis of the coinductive proof principle. We show that an efficient implementation is tantamount to the construction of nondeterministic automata from the given expressions by a Kleene construction, determinizing the two automata by a standard subset construction, and constructing a bisimulation on states of the resulting deterministic automata. It follows that coinductive equivalence proofs can be generated automatically in PSPACE. Worthington [10] has given a similar bound for equational proofs.

1 Automata on Guarded Strings

Automata on guarded strings (AGS), also known as automata with tests, were introduced in [6]. They are a generalization of ordinary finite-state automata to include tests. An ordinary automaton with null transitions is an AGS over the two-element Boolean algebra.

1.1 Guarded Strings

Recall that a *guarded string* over P, B is an alternating sequence

$$\alpha_0 p_1 \alpha_1 p_2 \cdots \alpha_{n-1} p_n \alpha_n,$$

where $p_i \in \mathsf{P}$ and the α_i are atoms (minimal nonzero elements) of the free Boolean algebra B generated by B. The set of atoms is denoted At. Elements of B are called *tests*. Every test is equivalent to a sum of atoms. The elements of At can be regarded either as conjunctions of literals of B (elements of B or their negations) or as truth assignments to B. A *guarded string* is thus an element of $(\mathsf{At} \cdot \mathsf{P})^* \cdot \mathsf{At}$. The set of all guarded strings is denoted GS. Guarded strings represent the join-irreducible elements of the free KAT on generators P, B.

1.2 Nondeterministic Automata

A nondeterministic AGS consists of a labeled directed graph with two types of transitions, *action transitions* labeled with actions (elements of P) and *test transitions* labeled with tests (elements of B). There is a distinguished set START of *start states* and a distinguished set ACCEPT of *accept states.*

An input to an AGS is a guarded string $\alpha_0 p_1 \alpha_1 \cdots \alpha_{n-1} p_n \alpha_n$. Intuitively, it operates as follows. It starts with a pebble on a nondeterministically chosen start state and its input head scanning α_0. In the course of the computation, the pebble will be occupying some state s and the input head will be scanning some α_i in the input string. If $i < n$, it may read the next action symbol p_{i+1} from the input string and move the pebble to any nondeterministically chosen state t such that there is an action transition from s to t with label p_{i+1}. In that case, the input head is advanced past p_{i+1} in the input string and is now scanning α_{i+1}. Alternatively, when scanning α_i, it may slide the pebble along an enabled test transition at any time without advancing the input head. A test transition is *enabled* if $\alpha_i \leq b$, where b is the label of the transition. The automaton accepts if it is ever scanning α_n while the pebble is on an accept state. Thus the automaton accepts a guarded string x if there is a directed path π from START to ACCEPT such that $x \le e$, where e is the product of the labels of the edges along π.

Formally, a (nondeterministic) *automaton on guarded strings* (AGS) over P and B is a tuple

$$M \;=\; (Q,\,\Delta,\,\textsc{start},\,\textsc{accept}),$$

where Q is a set of *states*, $\textsc{start} \subseteq Q$ are the *start states*, $\textsc{accept} \subseteq Q$ are the *accept states*, and Δ is the *transition function*

$$\Delta : (\mathsf{P} + \mathsf{At}) \;\to\; Q \;\to\; 2^Q,$$

where $+$ denotes disjoint (marked) union.

We will define the operation of the automaton in terms of the *Kleisli composition* $(\,;\,)$ on maps $Q \to 2^Q$, defined as

$$(R\,;S)(s) \;\stackrel{\text{def}}{=}\; \bigcup_{t \in R(s)} S(t).$$

From this we can define powers with respect to Kleisli composition and the Kleisli asterate operation

$$(R^0)(s) \;\stackrel{\text{def}}{=}\; \{s\} \qquad\qquad R^{n+1} \;\stackrel{\text{def}}{=}\; R^n\,;R \qquad\qquad R^*(s) \;\stackrel{\text{def}}{=}\; \bigcup_{n\ge 0} R^n(s).$$

We could have equivalently defined it in terms of binary relations, as the structures

$$(Q \to 2^Q, \,;\,, s \mapsto \{s\}) \qquad\qquad (2^{Q\times Q}, \,;\,, \mathsf{id})$$

are isomorphic as monoids, where on the right, ; denotes relational composition and id is the identity relation on Q.

The map Δ generates a map

$$\widehat{\Delta} : (\mathsf{P} + \mathsf{At}) \;\to\; Q \;\to\; 2^Q$$

defined by

$$\widehat{\Delta}_\alpha \;\stackrel{\text{def}}{=}\; \Delta_\alpha^\dagger \qquad\qquad \widehat{\Delta}_p \;\stackrel{\text{def}}{=}\; \Delta_p.$$

Intuitively, $\widehat{\Delta}_\alpha(s)$ accumulates all states accessible from s by a sequence of test transitions enabled under α. The map $\widehat{\Delta}$ extends further to a monoid homomorphism

$$\widehat{\Delta} : (\mathsf{P} + \mathsf{At})^* \;\to\; Q \;\to\; 2^Q$$

from the free monoid $(\mathsf{P} + \mathsf{At})^*$ to the monoid $Q \to 2^Q$ under Kleisli composition. Thus

$$\widehat{\Delta}_\varepsilon \;\stackrel{\text{def}}{=}\; s \mapsto \{s\} \qquad\qquad \widehat{\Delta}_{xy} \;\stackrel{\text{def}}{=}\; \widehat{\Delta}_x\,;\widehat{\Delta}_y.$$

The automaton M *accepts* $x \in \mathsf{GS}$ if there exists $s \in \textsc{start}$ such that $\widehat{\Delta}_x(s) \cap \textsc{accept} \ne \varnothing$. The set of guarded strings accepted by M is denoted $\mathsf{GS}(M)$.

1.3 Deterministic Automata

The definition of deterministic AGS here differs slightly from [6] so as to conform to the coalgebraic structure to be introduced in §2, but the difference is inessential. In [6] the set of states of a deterministic AGS is separated into disjoint sets of *action states* and *test states*, whereas here we have not made that distinction.

A *deterministic automaton on guarded strings* (AGS) over P and B is a structure

$$M \;=\; (Q,\, \delta,\, \varepsilon,\, \textsc{start}),$$

where Q is a set of *states*, $\textsc{start} \in Q$ is the *start state*, and

$$\delta : \mathsf{At}\cdot\mathsf{P} \;\to\; Q \;\to\; Q \qquad\qquad \varepsilon : \mathsf{At} \;\to\; Q \;\to\; 2$$

with components

$$\delta_{\alpha p} : Q \;\to\; Q \qquad\qquad \varepsilon_\alpha : Q \;\to\; 2$$

for $\alpha \in \mathsf{At}$ and $p \in \mathsf{P}$. The components ε_α play the same role as the accept states in a nondeterministic automaton.

Define the function $L : Q \to \mathsf{GS} \to 2$ coinductively as follows:

$$L(u)(\alpha) \;\stackrel{\text{def}}{=}\; \varepsilon_\alpha(u) \qquad\qquad L(u)(\alpha p y) \;\stackrel{\text{def}}{=}\; L(\delta_{\alpha p}(u))(y), \tag{1}$$

where $y \in \mathsf{GS}$, $\alpha \in \mathsf{At}$, and $p \in \mathsf{P}$. The machine is said to *accept* $x \in \mathsf{GS}$ if $L(\textsc{start})(x) = 1$. The set of guarded strings accepted by M is denoted $\mathsf{GS}(M)$. Identifying a subset of GS with its characteristic function $\mathsf{GS} \to 2$, we can write $\mathsf{GS}(M) = L(\textsc{start})$.

1.4 Determinization

Nondeterministic automata on guarded strings can be determinized by a subset construction similar to that for ordinary automata. Given a nondeterministic AGS

$$N = (Q,\, \Delta,\, \textsc{start},\, \textsc{accept}),$$

there is an equivalent deterministic AGS

$$M = (2^Q,\, \delta,\, \varepsilon,\, \textsc{start}),$$

where for $A \subseteq Q$,

$$\varepsilon_\alpha(A) \stackrel{\text{def}}{=} \begin{cases} 1, & \text{if } \exists s \in A\ \widehat{\Delta}_\alpha(s) \cap \textsc{accept} \neq \varnothing, \\ 0, & \text{otherwise} \end{cases} \qquad\qquad \delta_{\alpha p}(A) \stackrel{\text{def}}{=} \bigcup_{s\in A} \widehat{\Delta}_{\alpha p}(s).$$

One can show by a straightforward induction on the length of $x \in \mathsf{GS}$ that for all $A \subseteq Q$,

$$L(A)(x) = \begin{cases} 1, & \text{if } \exists s \in A\ \widehat{\Delta}_x(s) \cap \textsc{accept} \neq \varnothing, \\ 0, & \text{otherwise;} \end{cases}$$

in particular,

$$L(\textsc{start})(x) = 1 \;\Leftrightarrow\; \exists s \in \textsc{start}\ \widehat{\Delta}_x(s) \cap \textsc{accept} \neq \varnothing.$$

These are exactly the acceptance criteria for M and N respectively, so $\mathsf{GS}(M) = \mathsf{GS}(N)$.

2 Kleene Coalgebra with Tests (KCT)

A Kleene coalgebra with tests (KCT) is very much like a Kleene coalgebra (KC) [8], but with the addition of Boolean tests. Formally, a *Kleene coalgebra with tests* (KCT) over P and B is a structure

$$M \;=\; (Q,\, \delta,\, \varepsilon),$$

where Q is a set of *states* and

$$\delta : \mathsf{At}\cdot\mathsf{P} \to Q \to Q \qquad\qquad \varepsilon : \mathsf{At} \to Q \to 2$$

for $\alpha \in \mathsf{At}$ and $p \in \mathsf{P}$, exactly as in deterministic automata on guarded strings. Thus we can view a KCT as simply a deterministic AGS without a designated start state. One could also say that a KCT is a coalgebra for the functor $FX = 2^{\mathsf{At}} \times X^{\mathsf{At}\cdot\mathsf{P}}$.

A KCT *morphism* $h : (Q, \delta, \varepsilon) \to (Q', \delta', \varepsilon')$ is a set map $h : Q \to Q'$ that commutes with δ, δ' and $\varepsilon, \varepsilon'$; that is,

$$\delta'_{\alpha p}(h(u)) = h(\delta_{\alpha p}(u)) \qquad\qquad \varepsilon'_{\alpha}(h(u)) = \varepsilon_{\alpha}(u).$$

We denote the category of KCTs and KCT morphisms over P and B also by KCT.

2.1 Brzozowski Derivative, Semantic Form

There is a natural KCT over P and B defined in terms of the Brzozowski derivative on sets of guarded strings. The traditional Brzozowski derivative [3] is a kind of residuation operator on sets of ordinary strings. The current form is quite similar, except that we extend the definition to accommodate tests.

We define two maps

$$D : \mathsf{At}\cdot\mathsf{P} \to 2^{\mathsf{GS}} \to 2^{\mathsf{GS}} \qquad\qquad E : \mathsf{At} \to 2^{\mathsf{GS}} \to 2,$$

where for $A \subseteq \mathsf{GS}$,

$$D_{\alpha p}(A) \overset{\text{def}}{=} \{x \in \mathsf{GS} \mid \alpha p x \in A\} \qquad\qquad E_{\alpha}(A) \overset{\text{def}}{=} \begin{cases} 1, & \text{if } \alpha \in A, \\ 0, & \text{if } \alpha \notin A. \end{cases}$$

It is clear that the structure

$$(2^{\mathsf{GS}}, D, E)$$

forms a KCT. Indeed, it is the final object in the category KCT: for any KCT (Q, δ, ε), the function $L : Q \to 2^{\mathsf{GS}}$ defined in (1) is the unique KCT morphism $L : (Q, \delta, \varepsilon) \to (2^{\mathsf{GS}}, D, E)$.

2.2 Brzozowski Derivative, Syntactic Form

As with Brzozowski's original formulation [3], there is also a syntactic form of the Brzozowski derivative defined on KAT expressions. Let $\mathsf{Exp} = \mathsf{Exp}\,\mathsf{P}, \mathsf{B}$, the set of KAT expressions over P and B. We define a family of derivative operators

$$D : \mathsf{At}\cdot\mathsf{P} \to \mathsf{Exp} \to \mathsf{Exp} \qquad\qquad E : \mathsf{At} \to \mathsf{Exp} \to 2$$

consisting of components

$$D_{\alpha p} : \mathsf{Exp} \to \mathsf{Exp} \qquad\qquad E_{\alpha} : \mathsf{Exp} \to 2$$

defined inductively as follows. For $\alpha \in \mathsf{At}$, $p, q \in \mathsf{P}$, and $b \in B$,

$$\begin{aligned} D_{\alpha p}(e_1 + e_2) &\overset{\text{def}}{=} D_{\alpha p}(e_1) + D_{\alpha p}(e_2) \\ D_{\alpha p}(e_1 e_2) &\overset{\text{def}}{=} D_{\alpha p}(e_1)\, e_2 + E_{\alpha}(e_1)\, D_{\alpha p}(e_2) \\ D_{\alpha p}(e^*) &\overset{\text{def}}{=} D_{\alpha p}(e)\, e^* \end{aligned} \qquad \begin{aligned} D_{\alpha p}(q) &\overset{\text{def}}{=} \begin{cases} 1, & \text{if } p = q, \\ 0, & \text{otherwise,} \end{cases} \\ D_{\alpha p}(b) &\overset{\text{def}}{=} 0. \end{aligned}$$

$$E_\alpha(e_1+e_2) \overset{\text{def}}{=} E_\alpha(e_1)+E_\alpha(e_2) \qquad E_\alpha(b) \overset{\text{def}}{=} \begin{cases}1, & \text{if } \alpha \le b,\\ 0, & \text{otherwise,}\end{cases}$$
$$E_\alpha(e_1e_2) \overset{\text{def}}{=} E_\alpha(e_1)\,E_\alpha(e_2)$$
$$E_\alpha(e^*) \overset{\text{def}}{=} 1 \qquad E_\alpha(q) \overset{\text{def}}{=} 0.$$

These operators on KAT expressions are collectively called the *syntactic Brzozowski derivative.*

The map E_α is just the evaluation morphism that for any KAT expression substitutes 0 for any $p \in \mathsf{P}$, 1 for any $b \in \mathsf{B}$ such that $\alpha \le b$, and 0 for any $b \in \mathsf{B}$ such that $\alpha \le \bar{b}$, then simplifies the resulting expression over the two-element Kleene algebra $\mathbb{2}$. It is easily shown that for any KAT expression e,

$$E_\alpha(e) \;=\; \begin{cases}1, & \text{if } \alpha \le e,\\ 0, & \text{if } \alpha \not\le e\end{cases} \;=\; \begin{cases}1, & \text{if } \alpha \in \mathsf{GS}(e),\\ 0, & \text{if } \alpha \notin \mathsf{GS}(e).\end{cases}$$

The structure

$$(\mathsf{Exp},\, D,\, E)$$

is a KCT in the sense of §2, thus there is a unique KCT homomorphism $(\mathsf{Exp},\, D,\, E) \to (2^{\mathsf{GS}},\, D,\, E)$ to the final coalgebra defined in (1). We will show that this morphism is just G, the canonical interpretation of KAT expressions as sets of guarded strings. Thus G is both a KAT homomorphism and a KCT homomorphism.

Lemma 1. *For all $\alpha \in \mathsf{At}$, $p \in \mathsf{P}$, and $e, e' \in \mathsf{Exp}$,*

$$\alpha p e' \le e \;\Leftrightarrow\; e' \le D_{\alpha p}(e).$$

Proof. For the forward implication,

$$D_{\alpha p}(\alpha p e') \;=\; D_{\alpha p}(\alpha) p e' + E_\alpha(\alpha) D_{\alpha p}(p) e' + E_\alpha(\alpha) E_\alpha(p) D_{\alpha p}(e') \;=\; e'.$$

By monotonicity of $D_{\alpha p}$,

$$\alpha p e' \le e \;\Rightarrow\; e' = D_{\alpha p}(\alpha p e') \le D_{\alpha p}(e).$$

For the reverse implication, it suffices to show $\alpha p D_{\alpha p}(e) \le e$. We proceed by induction on the structure of e. For $p \in \mathsf{P}$,

$$\alpha p D_{\alpha p}(p) \;=\; \alpha p \;\le\; p.$$

For the case e_1e_2,

$$\begin{aligned}\alpha p D_{\alpha p}(e_1e_2) &= \alpha p D_{\alpha p}(e_1)e_2 + \alpha p E_\alpha(e_1) D_{\alpha p}(e_2)\\ &= \alpha p D_{\alpha p}(e_1)e_2 + \alpha E_\alpha(e_1) \alpha p D_{\alpha p}(e_2)\\ &\le e_1e_2.\end{aligned}$$

For the case e^*,

$$\alpha p D_{\alpha p}(e^*) \;=\; \alpha p D_{\alpha p}(e) e^* \;\le\; e e^* \;\le\; e^*.$$

All other cases are equally straightforward. □

Theorem 2. *For all KAT expressions e, taking e as the start state of the automaton $(\mathsf{Exp},\, D,\, E,\, e)$, the set accepted by this automaton is $G(e)$.*

Proof. We wish to show that for all $x \in \mathsf{GS}$, $x \in G(e)$ iff $L(e)(x) = 1$, where L is the map defined in §1.3. By the completeness theorem for KAT [7], we have $x \in G(e)$ iff $x \le e$, so it suffices to show that $x \le e$ iff $L(e)(x) = 1$. We proceed by induction on the length of x. The basis for x an atom α is immediate from the definition of E_α. For $x = \alpha p y$, by Lemma 1,

$$\alpha p y \le e \;\Leftrightarrow\; y \le D_{\alpha p}(e) \;\Leftrightarrow\; L(D_{\alpha p}(e))(y) = 1 \;\Leftrightarrow\; L(e)(apy) = 1.$$ □

3 Completeness

3.1 Bisimulation on KCTs

A *bisimulation* between two KCTs $M = (Q,\, \delta,\, \varepsilon)$ and $M' = (Q',\, \delta',\, \varepsilon')$ is a binary relation $\approx\, \subseteq Q \times Q'$ such that if $s \in Q$, $t \in Q'$, and $s \approx t$, then for all $\alpha \in \mathsf{At}$ and $p \in \mathsf{P}$,

(i) $\varepsilon_\alpha(s) = \varepsilon'_\alpha(t)$; and

(ii) $\delta_{\alpha p}(s) \approx \delta'_{\alpha p}(t)$.

Lemma 3. *For $s \in Q$ and $t \in Q'$,*

$$s \approxeq t \overset{def}{\Leftrightarrow} G(s) = G(t)$$

is the unique maximal bisimulation between M and M'.

Proof. It is easily shown that $\approxeq$ satisfies (i) and (ii). Moreover, if $\approx$ is any relation satisfying (i) and (ii), one can show by a straightforward inductive argument that $\approx$ refines $\approxeq$, thus $\approxeq$ is the unique maximal relation satisfying (i) and (ii). ☐

An *autobisimulation* is a bisimulation between M and itself. Bisimulations are closed under relational composition and arbitrary union, and the identity relation is an autobisimulation. Thus the reflexive, symmetric, and transitive closure of an autobisimulation is again an autobisimulation. An autobisimulation that is so closed is called a *bisimulation equivalence*. Bisimulation equivalences are exactly the kernels of KCT-morphisms.

A KCT is bisimilar to its quotient by any bisimulation equivalence under the map $\{(s,\, [s]) \mid s \in Q\}$, where $[s]$ is the bisimulation equivalence class of s. The quotient by the unique maximal autobisimulation is a sub-coalgebra of the final coalgebra.

3.2 Bisimulation on Deterministic Automata

For deterministic automata on guarded strings $M = (Q,\, \delta,\, \varepsilon,\, \text{START})$ and $M' = (Q',\, \delta',\, \varepsilon',\, \text{START}')$, we say that they are *bisimilar* if there is a bisimulation $\approx$ between the underlying KCTs $(Q,\, \delta,\, \varepsilon)$ and $(Q',\, \delta',\, \varepsilon')$ such that $\text{START} \approx \text{START}'$.

Lemma 4. *M and M' are bisimilar iff $\mathsf{GS}(M) = \mathsf{GS}(M')$.*

Proof. Let $\approxeq$ be the maximum bisimulation relation defined in the proof of Lemma 3. If $\mathsf{GS}(M) = \mathsf{GS}(M')$, then $L(\text{START}) = L(\text{START}')$ by the definition of acceptance, therefore $\text{START} \approxeq \text{START}'$. Then M and M' are bisimilar under $\approxeq$.

Conversely, if there exists a bisimulation $\approx$ between M and M', then $\text{START} \approx \text{START}'$, and by Lemma 3, $\approx$ refines $\approxeq$, therefore $\text{START} \approxeq \text{START}'$. Thus $\approxeq$ is a bisimulation of automata. ☐

The quotient of an automaton by its unique maximal autobisimulation gives the unique minimal equivalent automaton (ignoring inaccessible states).

Theorem 5 (Completeness)**.** *The following are equivalent:*

(i) $(\mathsf{Exp},\, D,\, E,\, e) \approxeq (\mathsf{Exp},\, D,\, E,\, e')$;

(ii) $L(e) = L(e')$;

(iii) $G(e) = G(e')$;

(iv) $\mathsf{KAT} \vdash e = e'$.

Proof. The equivalence of (i)–(iii) follows from Theorem 2 and Lemma 4. The equivalence of (iii) and (iv) are just the soundness and completeness of KAT for the guarded string model [7]. $\square$

4 Complexity

Let $\mathsf{Exp} = \mathsf{Exp}\,\mathsf{P},\mathsf{B}$ and let $e \in \mathsf{Exp}$. Let $(\mathsf{Exp},\, D,\, E,\, e)$ be the syntactic automaton with start state e, where D and E are the syntactic Brzozowski derivative. Let $(\mathsf{Exp}_e,\, D,\, E,\, e)$ denote the subautomaton of $(\mathsf{Exp},\, D,\, E,\, e)$ consisting of those expressions that are accessible from e; that is, those expressions of the form $D_x(e)$ for some $x \in (\mathsf{At}\cdot\mathsf{P})^*$. Theorem 5 by itself is not very useful as a deductive system or decision procedure for equivalence, because Exp_e is infinite in general. However, equivalent finite systems exist. In particular, by Theorem 5, KAT equivalence is the maximal autobisimulation on Exp. The quotient with respect to this relation, ignoring inaccessible states, gives the minimal deterministic AGS accepting $G(e)$, which is finite since $G(e)$ is regular.

To construct this automaton directly, we would need an independent algorithm to decide KAT equivalence. Fortunately, however, we can obtain finite automata with finer congruences that are easier to decide than full KAT equivalence. Chen and Pucella [4] use equivalence modulo additive associativity, commutativity, and idempotence (ACI-equivalence). Here we consider equivalence modulo the axioms of idempotent commutative monoids for $+, 0$ and the axioms

$$1 \cdot x = x \qquad\qquad 0 \cdot x = 0 \qquad\qquad (x+y)\cdot z = xz + yz. \tag{2}$$

Multiplicative associativity is not assumed, nor is left distributivity. We might call structures satisfying these axioms *right presemirings*. We denote by $\approx$ the congruence on terms generated by these axioms. We will show that $\mathsf{Exp}_e/{\approx}$ has finitely many accessible classes. It is a coarser relation than ACI-equivalence, therefore has fewer classes, but is still easy to decide, as there are normal forms up to additive commutativity. Of course, it makes the most sense to use the coarsest relation possible that is easily decidable, because coarser relations give smaller automata.

Because there are only finitely many $\approx$-classes accessible from e, the quotient automaton $\mathsf{Exp}_e/{\approx}$ is finite, and we can use it to obtain finite coinductive equivalence proofs. More interestingly, we will also show that $\mathsf{Exp}_e/{\approx}$ is a homomorphic image of a deterministic automaton M_e obtained by creating a nondeterministic AGS N_e from the expression e by a Kleene construction, then determinizing N_e by a subset construction as described in §1.4. This characterization gives a bound on the size of $\mathsf{Exp}_e/{\approx}$, which we can then use to argue that coinductive equivalence proofs can be generated automatically in PSPACE.

Lemma 6. *The relation $\approx$ is a bisimulation equivalence on* Exp.

Proof. We must show that if $e \approx e'$, then $E_\alpha(e) = E_\alpha(e')$ and $D_{\alpha p}(e) \approx D_{\alpha p}(e')$. The first conclusion follows from Theorem 5 and the fact that $\approx$ refines KAT-equivalence.

For the additive axioms of idempotent commutative monoids, the second conclusion follows from the additivity of $D_{\alpha p}$.

For the axioms (2),

$$\begin{aligned}
D_{\alpha p}(1x) &= D_{\alpha p}(1)x + E_\alpha(1)D_{\alpha p}(x) \;=\; 0x + 1D_{\alpha p}(x) \;\approx\; D_{\alpha p}(x)\\
D_{\alpha p}(0x) &= D_{\alpha p}(0)x + E_\alpha(0)D_{\alpha p}(x) \;=\; 0x + 0D_{\alpha p}(x) \;\approx\; D_{\alpha p}(0)\\
D_{\alpha p}((x+y)z) &= (D_{\alpha p}(x) + D_{\alpha p}(y))z + (E_\alpha(x) + E_\alpha(y))D_{\alpha p}(z)\\
&\approx D_{\alpha p}(x)z + E_\alpha(x)D_{\alpha p}(z) + D_{\alpha p}(y)z + E_\alpha(y)D_{\alpha p}(z)\\
&= D_{\alpha p}(xz + yz).
\end{aligned}$$

Finally, we must show that if $e_1 \approx e_2$, then $D_{\alpha p}(e_1+e_3) \approx D_{\alpha p}(e_2+e_3)$, $D_{\alpha p}(e_1e_3) \approx D_{\alpha p}(e_2e_3)$, $D_{\alpha p}(e_3e_1) \approx D_{\alpha p}(e_3e_2)$, and $D_{\alpha p}(e_1^*) \approx D_{\alpha p}(e_2^*)$. These arguments are all quite easy. For example,

$$\begin{aligned}
D_{\alpha p}(e_1e_3) &= D_{\alpha p}(e_1)e_3 + E_\alpha(e_1)D_{\alpha p}(e_3) \;\approx\; D_{\alpha p}(e_2)e_3 + E_\alpha(e_2)D_{\alpha p}(e_3) \;=\; D_{\alpha p}(e_2e_3)\\
D_{\alpha p}(e_1^*) &= D_{\alpha p}(e_1)e_1^* \;\approx\; D_{\alpha p}(e_2)e_2^* \;=\; D_{\alpha p}(e_2^*).
\end{aligned}$$

□

4.1 Closure

To establish the finiteness of the quotient automaton $\mathsf{Exp}_e/\!\approx$ and explain its relationship to the Kleene construction, we derive a formal relationship between the set of accessible $\approx$-classes of derivatives $\{[\![D_x(e)]\!] \mid x \in (\mathsf{At}\cdot\mathsf{P})^*\}$ and certain sets of terms derived from e.

For KAT term e, we define the *closure* of e, denoted $\mathrm{cl}(e)$, to be the smallest set of terms containing e and 1 and closed under the following rules:

$$\begin{array}{ccc}
\dfrac{e \in \mathrm{cl}(e_1)}{e \in \mathrm{cl}(e_1+e_2)} & \dfrac{e \in \mathrm{cl}(e_1)}{ee_2 \in \mathrm{cl}(e_1e_2)} & \dfrac{e \in \mathrm{cl}(e_1)}{ee_1^* \in \mathrm{cl}(e_1^*)}\\[2ex]
\dfrac{e \in \mathrm{cl}(e_2)}{e \in \mathrm{cl}(e_1+e_2)} & \dfrac{e \in \mathrm{cl}(e_2)}{e \in \mathrm{cl}(e_1e_2)} & \dfrac{e \in \mathrm{cl}(b)}{e \in \mathrm{cl}(\bar b)}
\end{array} \tag{3}$$

Lemma 7. *The set* $\mathrm{cl}(e)$ *contains at most* $|e|+1$ *elements, where* $|e|$ *is the number of subterms of* e.

Proof. We show by induction on e that $\mathrm{cl}'(e)$ contains at most $|e|$ elements, where $\mathrm{cl}'(e) = \mathrm{cl}(e) \setminus \{1\}$. For $e \in \mathsf{P} \cup \mathsf{B}$, $\mathrm{cl}'(e) = \{e\}$. For the other operators, from the rules (3) we have

$$\begin{aligned}
\mathrm{cl}'(\bar b) &= \{\bar b\} \cup \mathrm{cl}'(b),\\
\mathrm{cl}'(e_1+e_2) &= \{e_1+e_2\} \cup \mathrm{cl}'(e_1) \cup \mathrm{cl}'(e_2),\\
\mathrm{cl}'(e_1e_2) &= \{e_1e_2\} \cup \{ee_2 \mid e \in \mathrm{cl}'(e_1)\} \cup \mathrm{cl}'(e_2),\\
\mathrm{cl}'(e_1^*) &= \{e_1^*\} \cup \{ee_1^* \mid e \in \mathrm{cl}'(e_1)\}.
\end{aligned}$$

The result follows. □

4.2 Set Representation of Derivatives

We now construct a nondeterministic transition function Δ on the set of states $\mathsf{Exp} + (\mathsf{At} \times \mathsf{Exp})$ as follows. The elements of Exp are called *test states* and the elements of $\mathsf{At} \times \mathsf{Exp}$ are called *action states*. The test transitions go only from test states to action states, and the action transitions go only from action states to test states. Thus for $\alpha \in \mathsf{At}$ and $p \in \mathsf{P}$,

$$\Delta_\alpha : \mathsf{Exp} \to 2^{\mathsf{At}\times\mathsf{Exp}} \qquad\qquad \Delta_p : \mathsf{At}\times\mathsf{Exp} \to 2^{\mathsf{Exp}}.$$

The test transitions are deterministic: $\Delta_\alpha(e) \overset{\text{def}}{=} \{(\alpha, e)\}$. The action transitions are defined inductively:

$$\begin{aligned}
\Delta_p(\alpha, q) &\overset{\text{def}}{=} \begin{cases} \{1\}, & \text{if } q \in \mathsf{P} \text{ and } q = p, \\ \varnothing, & \text{if } q \in \mathsf{P} \text{ and } q \neq p \text{ or } q \in B, \end{cases} \\
\Delta_p(\alpha, e_1 + e_2) &\overset{\text{def}}{=} \Delta_p(\alpha, e_1) \cup \Delta_p(\alpha, e_2), \\
\Delta_p(\alpha, e_1 e_2) &\overset{\text{def}}{=} \begin{cases} \{e e_2 \mid e \in \Delta_p(\alpha, e_1)\} \cup \Delta_p(\alpha, e_2), & \text{if } E_\alpha(e_1) = 1, \\ \{e e_2 \mid e \in \Delta_p(\alpha, e_1)\}, & \text{if } E_\alpha(e_1) = 0, \end{cases} \\
\Delta_p(\alpha, e_1^*) &\overset{\text{def}}{=} \{e e_1^* \mid e \in \Delta_p(\alpha, e_1)\}.
\end{aligned}$$

Due to the bipartite structure of the states, we have $\widehat{\Delta}_{\alpha p} = \Delta_\alpha \,;\, \Delta_p$, where $\widehat{\Delta}$ is the extension of Δ defined in §1.2. Then

$$\widehat{\Delta}_{\alpha p}(e) \;=\; (\Delta_\alpha \,;\, \Delta_p)(e) \;=\; \bigcup \{\Delta_p(\alpha, e)\} \;=\; \Delta_p(\alpha, e). \tag{4}$$

We thus have

$$\begin{aligned}
\widehat{\Delta}_{\alpha p}(q) &\overset{\text{def}}{=} \begin{cases} \{1\}, & \text{if } q \in \mathsf{P} \text{ and } q = p, \\ \varnothing, & \text{if } q \in \mathsf{P} \text{ and } q \neq p \text{ or } q \in B, \end{cases} \\
\widehat{\Delta}_{\alpha p}(e_1 + e_2) &\overset{\text{def}}{=} \widehat{\Delta}_{\alpha p}(e_1) \cup \widehat{\Delta}_{\alpha p}(e_2), \\
\widehat{\Delta}_{\alpha p}(e_1 e_2) &\overset{\text{def}}{=} \begin{cases} \{e e_2 \mid e \in \widehat{\Delta}_{\alpha p}(e_1)\} \cup \widehat{\Delta}_{\alpha p}(e_2), & \text{if } E_\alpha(e_1) = 1, \\ \{e e_2 \mid e \in \widehat{\Delta}_{\alpha p}(e_1)\}, & \text{if } E_\alpha(e_1) = 0, \end{cases} \\
\widehat{\Delta}_{\alpha p}(e_1^*) &\overset{\text{def}}{=} \{e e_1^* \mid e \in \widehat{\Delta}_{\alpha p}(e_1)\}.
\end{aligned}$$

Lemma 8. *For all* KAT *terms e and $x \in (\mathsf{At} \cdot \mathsf{P})^*$, $\widehat{\Delta}_x(e) \subseteq \mathrm{cl}(e)$.*

Proof. We first show that for $\alpha \in \mathsf{At}$ and $p \in \mathsf{P}$, $\widehat{\Delta}_{\alpha p}(e) \subseteq \mathrm{cl}(e)$ by induction on the structure of e. The cases $e \in \mathsf{P}$ or $e \in B$ are easy. For the other operators,

$$\begin{aligned}
\widehat{\Delta}_{\alpha p}(e_1 + e_2) &= \widehat{\Delta}_{\alpha p}(e_1) \cup \widehat{\Delta}_{\alpha p}(e_2) \;\subseteq\; \mathrm{cl}(e_1) \cup \mathrm{cl}(e_2) \;\subseteq\; \mathrm{cl}(e_1 + e_2) \\
\widehat{\Delta}_{\alpha p}(e_1 e_2) &= \begin{cases} \{e e_2 \mid e \in \widehat{\Delta}_{\alpha p}(e_1)\} \cup \widehat{\Delta}_{\alpha p}(e_2), & \text{if } E_\alpha(e_1) = 1 \\ \{e e_2 \mid e \in \widehat{\Delta}_{\alpha p}(e_1)\}, & \text{if } E_\alpha(e_1) = 0 \end{cases} \\
&\subseteq \{e e_2 \mid e \in \mathrm{cl}(e_1)\} \cup \mathrm{cl}(e_2) \;\subseteq\; \mathrm{cl}(e_1 e_2) \\
\widehat{\Delta}_{\alpha p}(e_1^*) &= \{e e_1^* \mid e \in \widehat{\Delta}_{\alpha p}(e_1)\} \;\subseteq\; \{e e_1^* \mid e \in \mathrm{cl}(e_1)\} \;\subseteq\; \mathrm{cl}(e_1^*).
\end{aligned}$$

For arbitrary $x \in (\mathsf{At} \cdot \mathsf{P})^*$, we proceed by induction on the length of x. The base case $x = \varepsilon$ is easy and the case $x = \alpha p$ is given by the previous argument. For $x \neq \varepsilon$ and $y \neq \varepsilon$,

$$\widehat{\Delta}_{xy}(e) \;=\; (\widehat{\Delta}_x \,;\, \widehat{\Delta}_y)(e) \;=\; \bigcup \{\widehat{\Delta}_y(d) \mid d \in \widehat{\Delta}_x(e)\} \;\subseteq\; \bigcup \{\mathrm{cl}(d) \mid d \in \mathrm{cl}(e)\} \;=\; \mathrm{cl}(e). \qquad \square$$

Lemma 9. *For all* KAT *terms e and $x \in (\mathsf{At} \cdot \mathsf{P})^*$, $D_x(e) \approx \sum \widehat{\Delta}_x(e)$.*

Proof. We first show that for $\alpha \in \mathsf{At}$ and $p \in \mathsf{P}$, $D_{\alpha p}(e) \approx \sum \widehat{\Delta}_{\alpha p}(e)$ by induction on the structure of e. For $q \in \mathsf{P}$, we have

$$D_{\alpha p}(q) \;=\; \begin{cases} 1, & \text{if } p = q \\ 0, & \text{if } p \neq q \end{cases} \;=\; \begin{cases} \sum \{1\}, & \text{if } p = q \\ \sum \varnothing, & \text{if } p \neq q \end{cases} \;=\; \sum \widehat{\Delta}_{\alpha p}(q).$$

For $b \in B$,

$$D_{\alpha p}(b) \;=\; 0 \;=\; \sum \varnothing \;=\; \sum \widehat{\Delta}_{\alpha p}(b).$$

For the other operators,

$$\begin{aligned}
D_{\alpha p}(e_1 + e_2) \;&=\; D_{\alpha p}(e_1) + D_{\alpha p}(e_2) \;\approx\; \sum \widehat{\Delta}_{\alpha p}(e_1) + \sum \widehat{\Delta}_{\alpha p}(e_2) \\
&\approx\; \sum (\widehat{\Delta}_{\alpha p}(e_1) \cup \widehat{\Delta}_{\alpha p}(e_2)) \;=\; \sum \widehat{\Delta}_{\alpha p}(e_1 + e_2),
\end{aligned}$$

$$\begin{aligned}
D_{\alpha p}(e_1 e_2) \;&=\; D_{\alpha p}(e_1)e_2 + E_\alpha(e_1)D_{\alpha p}(e_2) \\
&\approx\; (\sum \widehat{\Delta}_{\alpha p}(e_1))e_2 + E_\alpha(e_1) \sum \widehat{\Delta}_{\alpha p}(e_2) \\
&\approx\; \begin{cases} \sum\{ee_2 \mid e \in \widehat{\Delta}_{\alpha p}(e_1)\} + \sum \widehat{\Delta}_{\alpha p}(e_2), & \text{if } E_\alpha(e_1) = 1 \\ \sum\{ee_2 \mid e \in \widehat{\Delta}_{\alpha p}(e_1)\}, & \text{if } E_\alpha(e_1) = 0 \end{cases} \\
&\approx\; \begin{cases} \sum(\{ee_2 \mid e \in \widehat{\Delta}_{\alpha p}(e_1)\} \cup \widehat{\Delta}_{\alpha p}(e_2)), & \text{if } E_\alpha(e_1) = 1 \\ \sum\{ee_2 \mid e \in \widehat{\Delta}_{\alpha p}(e_1)\}, & \text{if } E_\alpha(e_1) = 0 \end{cases} \\
&=\; \sum \widehat{\Delta}_{\alpha p}(e_1 e_2),
\end{aligned}$$

$$\begin{aligned}
D_{\alpha p}(e_1^*) \;&=\; D_{\alpha p}(e_1)e_1^* \;\;\approx\;\; (\sum \widehat{\Delta}_{\alpha p}(e_1))e_1^* \\
&\approx\; \sum\{ee_1^* \mid e \in \widehat{\Delta}_{\alpha p}(e_1)\} \;=\; \sum \widehat{\Delta}_{\alpha p}(e_1^*).
\end{aligned}$$

Now we show the result for arbitrary $x \in (\mathsf{At} \cdot \mathsf{P})^*$ by induction on the length of x. The case $x = \varepsilon$ is trivial, and the case $x = \alpha p$ is given by the previous argument. Finally, for $x \neq \varepsilon$ and $y \neq \varepsilon$,

$$\begin{aligned}
D_{xy}(e) \;&=\; D_y(D_x(e)) \\
&\approx\; D_y(\sum \widehat{\Delta}_x(e)) && \text{by Lemma 6} \\
&=\; \sum\{D_y(d) \mid d \in \widehat{\Delta}_x(e)\} \\
&\approx\; \sum\{\sum \widehat{\Delta}_y(d) \mid d \in \widehat{\Delta}_x(e)\} \;\approx\; \sum \bigcup\{\widehat{\Delta}_y(d) \mid d \in \widehat{\Delta}_x(e)\} \\
&=\; \sum(\widehat{\Delta}_x \,;\, \widehat{\Delta}_y)(e) \;=\; \sum \widehat{\Delta}_{xy}(e).
\end{aligned}$$

□

Theorem 10. *The automaton* $\mathsf{Exp}_e/\!\approx$ *has at most* $2^{|e|+1}$ *accessible states.*

Proof. The accessible states of $\mathsf{Exp}_e/\!\approx$ are $\{[D_x(e)] \mid x \in (\mathsf{At} \cdot \mathsf{P})^*\}$, where $[d]$ is the congruence class of d modulo $\approx$. The stated bound follows from Lemmas 7, 8, and 9. □

4.3 Brzozowski Meets Kleene

It is possible to obtain $\mathsf{Exp}_e/\!\approx$ by a Kleene construction to obtain a nondeterministic AGS N_e with finitely many states, then apply the construction of §1.4 to obtain a deterministic automaton M_e with at most $2^{|e|+1}$ states. The automaton $\mathsf{Exp}_e/\!\approx$ is a homomorphic image of M_e. A version of Kleene's theorem for KAT terms and automata on guarded strings has been described previously in [6], but the current treatment parallels more closely Brzozowski's original treatment for ordinary regular expressions [3] and aligns with the general coalgebraic structure of [1, 2].

Define the nondeterministic automaton

$$N_e \stackrel{\text{def}}{=} (Q,\ \Delta,\ \textsc{start},\ \textsc{accept}),$$

where the set of states Q is the disjoint union $\mathrm{cl}(e) + (\mathsf{At} \times \mathrm{cl}(e))$, the transition function Δ is that defined in §4.2, and the start and accept states are

$$\textsc{start} \stackrel{\text{def}}{=} \{e\} \qquad\qquad \textsc{accept} \stackrel{\text{def}}{=} \{(\alpha, d) \mid E_\alpha(d) = 1\}.$$

That Δ_α maps $\mathrm{cl}(e)$ to $2^{\mathsf{At}\times\mathrm{cl}(e)}$ is immediate from the definition of Δ_α, and that Δ_p maps $\mathsf{At}\times\mathrm{cl}(e)$ to $2^{\mathrm{cl}(e)}$ is guaranteed by (4) and Lemma 8.

Now let

$$M_e \stackrel{\text{def}}{=} (2^{\mathrm{cl}(e)},\ \delta,\ \varepsilon,\ \textsc{start})$$

be the deterministic automaton obtained from N_e by the subset construction as described in §1.4. The start state of M_e is $\{e\}$, and δ and ε are given by

$$\delta_{\alpha p}(A) \ = \ \bigcup_{d\in A} \widehat{\Delta}_{\alpha p}(d) \qquad\qquad \varepsilon_\alpha(A) \ = \ \begin{cases} 1, & \text{if } \exists d \in A\ E_\alpha(d) = 1,\\ 0, & \text{otherwise.}\end{cases}$$

Note that the accessible states are all of the form $A \subseteq \mathrm{cl}(e)$, thus by Lemma 7, M_e has at most $2^{|e|+1}$ accessible states.

Theorem 11. *For $A \subseteq \mathsf{Exp}$, the map $A \mapsto (\sum A)/{\approx}$ is a* KCT *homomorphism. Ignoring inaccessible states, the quotient automaton $\mathsf{Exp}_e/{\approx}$ is the image of M_e under this map.*

Proof. We must show that the function $A \mapsto \sum A$ maps the start state of M_e to the start state of Exp_e, and that this function is a bisimulation modulo $\approx$. For δ,

$$\begin{aligned}
\sum \delta_{\alpha p}(A) &= \sum \bigcup \{\widehat{\Delta}_{\alpha p}(d) \mid d \in A\}\\
&\approx \sum \{\sum \widehat{\Delta}_{\alpha p}(d) \mid d \in A\}\\
&\approx \sum \{D_{\alpha p}(d) \mid d \in A\} && \text{by Lemma 9}\\
&= D_{\alpha p}(\sum A),
\end{aligned}$$

therefore

$$(\sum \delta_{\alpha p}(A))/{\approx} \ = \ (D_{\alpha p}(\sum A))/{\approx} \ = \ D_{\alpha p}((\sum A)/{\approx}).$$

For ε,

$$\varepsilon_\alpha(A) \ = \ \begin{cases} 1, & \text{if } \exists d \in A\ E_\alpha(d) = 1\\ 0, & \text{otherwise}\end{cases} \ = \ E_\alpha(\sum A) \ = \ E_\alpha((\sum A)/{\approx}).$$

The map also preserves start states:

$$\{e\} \ \mapsto \ (\sum \{e\})/{\approx} \ = \ e/{\approx}.$$

Thus the map $A \mapsto (\sum A)/{\approx}$ is a KCT-morphism mapping M_e to $\mathsf{Exp}_e/{\approx}$. □

4.4 Automatic Proof Generation in PSPACE

The results of Sections 4.2 and 4.3 give rise to a nondeterministic linear-space algorithm for deciding the equivalence of two given KAT terms. By Savitch's theorem [9], there is a deterministic quadratic-space algorithm. The deterministic algorithm can be used to create bisimulation proofs of equivalence or inequivalence automatically.

To obtain the linear space bound, we first show that each element of $\mathrm{cl}(e)$ corresponds to an occurrence of a subterm of e. This lets us use the occurrences of subterms of e as representatives for the elements of $\mathrm{cl}(e)$. To define the correspondence, we view terms as labeled trees; that is, as partial functions

$$e : \omega^* \rightharpoonup \mathsf{P} \cup \mathsf{B} \cup \{+, \cdot, {}^*, {}^-, 0, 1\}$$

with domain of definition $\mathsf{dom}\, e \subseteq \omega^*$ such that

- $\mathsf{dom}\, e$ is finite, nonempty, and prefix-closed;
- if $\sigma \in \mathsf{dom}\, e$ and $e(\sigma)$ is of arity n, then $\sigma i \in \mathsf{dom}\, e$ iff $i < n$. The arities of elements of P and B are 0 and those of $+, \cdot, {}^*, {}^-, 0, 1$ are 2, 2, 1, 1, 0, 0, respectively.

An occurrence of a subterm of e is identified by its position $\sigma \in \mathsf{dom}\, e$. The subterm at position σ is $\lambda\tau.e(\sigma\tau)$, and its domain is $\{\tau \mid \sigma\tau \in \mathsf{dom}\, e\}$.

Define a partial function $R : \omega^* \times \mathsf{Exp} \to \mathsf{Exp}$ inductively by

$$\begin{aligned} R(0\sigma, e_1 + e_2) &\overset{\text{def}}{=} R(\sigma, e_1) & \qquad R(0\sigma, e_1e_2) &\overset{\text{def}}{=} R(\sigma, e_1) \cdot e_2 \\ R(1\sigma, e_1 + e_2) &\overset{\text{def}}{=} R(\sigma, e_2) & R(1\sigma, e_1e_2) &\overset{\text{def}}{=} R(\sigma, e_2) \\ R(0\sigma, e^*) &\overset{\text{def}}{=} R(\sigma, e) \cdot e^* & R(\varepsilon, e) &\overset{\text{def}}{=} e. \end{aligned}$$

One can show by induction that $R(\sigma, e)$ is defined iff $\sigma \in \mathsf{dom}\, e$, and that a term is in $\mathrm{cl}(e)$ iff it is either 1 or $R(\sigma, e)$ for some $\sigma \in \mathsf{dom}\, e$.

Now we show how to construct coinductive equivalence and inequivalence proofs for two given terms e_1 and e_2. Construct the two nondeterministic AGS N_{e_1} and N_{e_2} as described in §4.3, representing the states by $\mathsf{dom}\, e_1$ and $\mathsf{dom}\, e_2$, respectively (assume without loss of generality that $1 = R(\sigma, e_1) = R(\tau, e_2)$ for some σ and τ). If we like, we can also reduce terms modulo $\approx$, so that if $R(\sigma, e_1) \approx R(\tau, e_1)$, we only need one of σ, τ.

Place pebbles on the start states of the two automata. Nondeterministically guess a string $y \in (\mathsf{At} \cdot \mathsf{P})^*$ and move the pebbles to all accessible states according to the transition functions of the two machines. Halt and declare e_1 and e_2 inequivalent if there exists $\alpha \in \mathsf{At}$ such that

$$E_\alpha(\sum_{\tau \in A} R(\tau, e_1)) \;\neq\; E_\alpha(\sum_{\rho \in B} R(\rho, e_2)),$$

where A and B are the sets of states of N_{e_1} and N_{e_2}, respectively, currently occupied by pebbles; we have found a guarded string $x = y\alpha$ accepted by one but not by the other, since

$$\begin{aligned} L(e_1)(x) &= E_\alpha(D_y(e_1)) = E_\alpha(\sum_{\tau \in A} R(\tau, e_1)) \\ L(e_2)(x) &= E_\alpha(D_y(e_2)) = E_\alpha(\sum_{\rho \in B} R(\rho, e_2)), \end{aligned}$$

therefore $L(e_1)(x) \neq L(e_2)(x)$.

Once we can decide equivalence in quadratic space, we can produce a bisimulation proof of equivalence in the same amount of space. We first produce the deterministic automata M_{e_1} and M_{e_2} equivalent to N_{e_1} and N_{e_2}. The states of M_{e_1} and M_{e_2} are represented by the powersets of $\mathsf{dom}\, e_1$ and $\mathsf{dom}\, e_2$, respectively. These sets are of exponential size, but they can be generated sequentially in linear space. The transition function is the action on subsets as defined in §1.4, and this can also be generated in linear space.

Now we attempt to construct the maximal bisimulation between the two deterministic automata. We iterate through all pairs of states, testing equivalence of each pair as described above. If the states are equivalent, we output the pair as bisimilar. The set of pairs that are ever output is the maximal bisimulation.

In case e_1 and e_2 are not equivalent, a witness for inequivalence can also be produced in PSPACE. A witness for inequivalence is a guarded string x accepted by one automaton but not the other. The shortest such string can be exponentially long in the worst case, but can be produced in the same way that one would produce an exponential-length accepting computation of a nondeterministic linear-space Turing machine, by a straightforward modification of the proof of Savitch's theorem [9].

References

[1] Marcello M. Bonsangue, Jan J. M. M. Rutten, and Alexandra M. Silva. Regular expressions for polynomial coalgebras. Technical Report SEN-E0703, Centrum voor Wiskunde en Informatica, Amsterdam, December 2007.

[2] Marcello M. Bonsangue, Jan J. M. M. Rutten, and Alexandra M. Silva. A Kleene theorem for polynomial coalgebras. Manuscript, May 2008.

[3] Janusz A. Brzozowski. Derivatives of regular expressions. *J. Assoc. Comput. Mach.*, 11(4):481–494, October 1964.

[4] Hubie Chen and Riccardo Pucella. A coalgebraic approach to Kleene algebra with tests. *Electronic Notes in Theoretical Computer Science*, 82(1), 2003.

[5] Dexter Kozen. Kleene algebra with tests. *ACM Trans. Programming Languages and Systems (TOPLAS'97)*, 19(3):427–443, May 1997.

[6] Dexter Kozen. Automata on guarded strings and applications. *Matématica Contemporânea*, 24:117–139, 2003.

[7] Dexter Kozen and Frederick Smith. Kleene algebra with tests: Completeness and decidability. In D. van Dalen and M. Bezem, editors, *Proc. 10th Int. Workshop Computer Science Logic (CSL'96)*, volume 1258 of *Lecture Notes in Computer Science*, pages 244–259, Utrecht, The Netherlands, September 1996. Springer-Verlag.

[8] J. Rutten. Universal coalgebra: a theory of systems. *Theor. Comput. Sci.*, 249:3–80, 2000.

[9] W. Savitch. Relationship between nondeterministic and deterministic tape complexities. *J. Comput. Syst. Sci.*, 4(2):177–192, 1970.

[10] James Worthington. Automatic proof generation in Kleene algebra. In R. Berghammer, B. Möller, and G. Struth, editors, *10th Int. Conf. Relational Methods in Computer Science (RelMiCS10) and 5th Int. Conf. Applications of Kleene Algebra (AKA5)*, volume 4988 of *Lect. Notes in Computer Science*, pages 382–396. Springer-Verlag, 2008.