Chapter 15
On the Coalgebraic Theory of Kleene Algebra with Tests

Dexter Kozen

In honor of Rohit Parikh

Abstract We develop a coalgebraic theory of Kleene algebra with tests (KAT) along the lines of Rutten (1998) for Kleene algebra (KA) and Chen and Pucella (Electron Notes Theor Comput Sci 82(1), 2003) for a limited version of KAT, resolving some technical issues raised by Chen and Pucella. Our treatment includes a simple definition of the Brzozowski derivative for KAT expressions and an automata-theoretic interpretation involving automata on guarded strings. We also give a complexity analysis, showing that an efficient implementation of coinductive equivalence proofs in this setting is tantamount to a standard automata-theoretic construction. It follows that coinductive equivalence proofs can be generated automatically in *PSPACE*. This matches the bound of Worthington (2008) for the automatic generation of equational proofs in KAT.

Keywords Kleene algebra · Kleene algebra with tests · Coalgebra · Verification

15.1 Introduction

Kleene algebra (KA) is the algebra of regular expressions. The operations $+$, $\cdot$, and * of Kleene algebra can be used to model nondeterministic choice, sequential composition, and iteration, respectively, on a set of actions.

Kleene algebra with tests (KAT) is an extension of KA obtained by identifying a subset of *tests*, which must satisfy the axioms of Boolean algebra as well. The

D. Kozen (✉)
Computer Science Department, Cornell University, Ithaca, NY 14853-7501, USA
e-mail: kozen@cs.cornell.edu

© Springer International Publishing AG 2017

kozen@cs.cornell.edu

two sorts, actions and tests, interact seamlessly: on tests, nondeterministic choice becomes join and sequential composition becomes meet.

The presence of tests allows KAT to model basic programming and verification constructs such as conditional tests, while loops, and Hoare triples. Thus KAT gives a simple equational approach to partial correctness. By now KAT has a well-developed theory, including an extensive model theory and results on expressiveness, deductive completeness, and complexity. It has been applied successfully in a number of areas, including the verification of compiler optimizations and communication protocols and various other program analysis tasks.

Traditionally, KAT is axiomatized equationally (Kozen 1997). In Chen and Pucella (2003), Chen and Pucella develop a coalgebraic theory of KAT inspired by Rutten's coalgebraic theory of KA based on deterministic automata (Rutten 1998). Remarking that "the known automata-theoretic presentation of KAT (Kozen 2003) does not lend itself to a coalgebraic theory," and that "the notion of derivative, essential to the coinduction proof principle in this context, is not readily definable for KAT expressions as defined in Kozen (1997)," Chen and Pucella develop a new interpretation of KAT expressions and a corresponding automata theory differing from Kozen (2003) in several respects. They give a coinductive proof principle and show how it can be used to establish equivalence of expressions. This gives an alternative to equational proofs using the standard axiomatization (Kozen 1997) or by minimization of deterministic automata (Kozen 2003).

The ability to generate equivalence proofs automatically has important implications for proof-carrying code. Chen and Pucella argue that the coalgebraic approach makes this possible, since proofs can be generated purely mechanically via repeated application of the Brzozowski derivative, whereas classical equational logic "requires creativity" (Chen and Pucella 2003). This is not strictly true, as Worthington (Worthington 2008) has shown that equational proofs can also be generated automatically. However, it is fair to say that the coinductive approach does provide a more natural mechanism.

Still unresolved is the issue of proof complexity in the coinductive setting. Chen and Pucella claim that coinduction can give shorter proofs, but they give no supporting evidence. Worthington's technique is known to require *PSPACE* and to produce exponential-size proofs in the worst case. This worst-case bound is unlikely to be significantly improved, as the equational theory of KAT is *PSPACE*-complete (Cohen et al. 1996).

Chen and Pucella's treatment has a few technical shortcomings, as they themselves point out. In their words:

> The "path independence" of a mixed automaton gives any mixed automaton a certain form of redundancy. This redundancy persists in the definition of bisimulation…An open question is to cleanly eliminate this redundancy; a particular motivation for doing this would be to make proofs of expression equivalence as simple as possible. Along these lines, it would be of interest to develop other weaker notions of bisimulation that give rise to bisimulations; pseudo-bisimulations require a sort of "fixed variable ordering" that does not seem absolutely necessary…

kozen@cs.cornell.edu

Another issue for future work would be to give a class of expressions wider than our mixed expressions for which there are readily understandable and applicable rules for computing derivatives. In particular, a methodology for computing derivatives of the KAT expressions defined by Kozen (1997) would be nice to see. Intuitively, there seems to be a tradeoff between the expressiveness of the regular expression language and the simplicity of computing derivatives (in the context of KAT). Formal tools for understanding this tradeoff could potentially be quite useful (Chen and Pucella 2003).

This paper addresses these issues. We develop a coalgebraic theory of KAT, which we call KCT, along the lines of Chen and Pucella (2003); Rutten (1998). Our treatment includes a new definition of the Brzozowski derivative, but in the context of the original automata-theoretic formulation of KAT involving automata on guarded strings (Kozen 2003). The syntactic form of the Brzozowski derivative applies to all KAT expressions as defined in Kozen (1997). The somewhat artificial concepts of path independence, fixed variable ordering, and pseudo-bisimulation do not arise in this setting. This treatment places KCT within the general coalgebraic framework described by Bonsangue et al. (2007, 2009).

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*. This matches Worthington's bound (Worthington 2008) for equational proofs.

15.2 KA and KAT

15.2.1 Kleene Algebra

Kleene algebra (KA) is the algebra of regular expressions (Conway 1971; Kleene 1956). The axiomatization used here is from Kozen (1994). A *Kleene algebra* is a structure $(K, +, \cdot, ^*, 0, 1)$ such that K is an idempotent semiring under $+$, $\cdot$, 0, and 1 and satisfies the axioms

$$\begin{aligned} 1 + pp^* &\le p^* & \qquad q + pr \le r &\Rightarrow p^*q \le r \\ 1 + p^*p &\le p^* & \qquad q + rp \le r &\Rightarrow qp^* \le r \end{aligned}$$

for *. There is a natural partial order $p \le q \overset{\text{def}}{\Longleftrightarrow} p + q = q$.

Standard models include the family of regular sets over a finite alphabet, the family of binary relations on a set, and the family of $n \times n$ matrices over another Kleene algebra. Other more unusual interpretations include the min,+ algebra, also known as the *tropical semiring*, used in shortest path algorithms, and models consisting of convex polyhedra used in computational geometry.

kozen@cs.cornell.edu

The completeness result of Kozen (1994) says that the algebra of regular sets of strings over a finite alphabet Σ is the free Kleene algebra on generators Σ. The axioms are also complete for the equational theory of relational models.

15.2.2 Kleene Algebra with Tests

A *Kleene algebra with tests* (KAT) (Kozen 1997) consists of a Kleene algebra K with an embedded Boolean algebra B such that the semiring structure on B is a subalgebra of the semiring structure on K. Elements of B are called *tests*. The Boolean negation operator is defined only on tests.

Like KA, KAT has language and relational models and is deductively complete over these interpretations (Kozen and Smith 1996). The chief language-theoretic models are the regular sets of *guarded strings* over alphabets Σ and T of primitive action and test symbols, respectively (see Sect. 15.3.1). This is the free KAT on generators Σ, T. The set of guarded strings represented by a KAT expression e is denoted $\mathsf{GS}(e)$.

KAT can code elementary programming constructs and Hoare partial correctness assertions and subsumes propositional Hoare logic (PHL). It is deductively complete over relational models, whereas PHL is not. Moreover, KAT is no more difficult to decide, as PHL, KA, and KAT are all *PSPACE*-complete.

For KAT expressions e_1, e_2, we write $e_1 \leq e_2$ if this inequality holds in the free KAT on generators Σ, T; that is, if it is a consequence of the axioms of KAT.

See Kozen (1994, 1997, 2000) for a more detailed introduction.

15.3 Automata on Guarded Strings

Automata on guarded strings (AGS), also known as automata with tests, were introduced in Kozen (2003). 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.

15.3.1 Guarded Strings

Guarded strings were first introduced in Kaplan (1969). A *guarded string* over Σ, T is an alternating sequence $\alpha_0 p_1 \alpha_1 p_2 \alpha_2 \cdots p_n \alpha_n$, $n \geq 0$, where $p_i \in \Sigma$ and the α_i are atoms (minimal nonzero elements) of the free Boolean algebra B generated by T. The set of atoms is denoted At. The elements of At can be regarded either as conjunctions of literals of T (elements of T or their negations) or as truth assignments to T. A *guarded string* is thus an element of $\mathsf{At} \cdot (\Sigma \cdot \mathsf{At})^*$. The set of all guarded strings is

kozen@cs.cornell.edu

denoted GS. Guarded strings represent the join-irreducible elements of the free KAT on generators Σ, T.

15.3.2 Nondeterministic Automata

A nondeterministic AGS consists of a labeled directed graph with two types of transitions, *action transitions* labeled with elements of Σ and *test transitions* labeled with elements of B. There is a set start of *start states* and a set accept of *accept states*.

An input to an AGS is a guarded string $\alpha_0 p_1 \alpha_1 p_2 \alpha_2 \cdots p_n \alpha_n$. 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, say the pebble is occupying state s and the input head is scanning α_i. 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}. The input head is advanced past p_{i+1} in the input string and is now scanning α_{i+1}. Also while 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 $\alpha_i \le b$, 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 Σ and T is a tuple

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

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

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

where $+$ denotes disjoint (marked) union.

The definition of acceptance involves the Kleisli composition $\bullet$ and asterate † operations on maps $Q \to 2^Q$ defined by:

$$R \bullet S \overset{\text{def}}{=} s \mapsto \bigcup_{t \in S(s)} R(t) \qquad\qquad R^0 \overset{\text{def}}{=} s \mapsto \{s\}$$

$$R^\dagger \overset{\text{def}}{=} \bigcup_{n \ge 0} R^n \qquad\qquad R^{n+1} \overset{\text{def}}{=} R \bullet R^n.$$

The map Δ generates a map

$$\hat{\Delta} : (\Sigma + \mathsf{At}) \to Q \to 2^Q \qquad \hat{\Delta}_\alpha \overset{\text{def}}{=} \Delta_\alpha^\dagger \qquad \hat{\Delta}_p \overset{\text{def}}{=} \Delta_p.$$

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

$$\hat{\Delta} : (\Sigma + \mathsf{At})^* \to Q \to 2^Q \qquad \hat{\Delta}_\varepsilon \overset{\text{def}}{=} s \mapsto \{s\} \qquad \hat{\Delta}_{xy} \overset{\text{def}}{=} \hat{\Delta}_y \bullet \hat{\Delta}_x$$

from the free monoid $(\Sigma + \mathsf{At})^*$ to the monoid $Q \to 2^Q$ under Kleisli composition. The guarded strings $\mathsf{GS} = \mathsf{At} \cdot (\Sigma \cdot \mathsf{At})^*$ form a submonoid of $(\Sigma + \mathsf{At})^*$. The automaton M *accepts* $x \in \mathsf{GS}$ if there exist $s \in \mathsf{start}$ and $t \in \mathsf{accept}$ such that $t \in \hat{\Delta}_x(s)$. The set of guarded strings accepted by M is denoted $\mathsf{GS}(M)$.

15.3.3 Deterministic Automata

The definition of deterministic AGS here differs from that of Kozen (2003) so as to conform to the coalgebraic structure to be introduced in Sect. 15.4, but the difference is inessential. In Kozen (2003) the set of states of a deterministic AGS is separated into disjoint sets of *action states* and *test states*, whereas here we have elided the action states.

A *deterministic automaton on guarded strings* (AGS) over Σ and T is a structure

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

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

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

with components

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

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

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

$$L(u)(\alpha) \overset{\text{def}}{=} \varepsilon_\alpha(u) \qquad L(u)(\alpha p y) \overset{\text{def}}{=} L(\delta_{\alpha p}(u))(y), \tag{15.1}$$

where $y \in \mathsf{GS}$, $\alpha \in \mathsf{At}$, and $p \in \Sigma$. The machine is said to *accept* $x \in \mathsf{GS}$ if $L(\mathsf{start})(x) = 1$. 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 $\mathsf{GS}(M) = L(\mathsf{start})$.

kozen@cs.cornell.edu

The map δ extends to a monoid homomorphism $\hat{\delta} : (\mathsf{At} \cdot \Sigma)^* \to Q \to Q$ from the free monoid $(\mathsf{At} \cdot \Sigma)^*$ to the monoid $Q \to Q$ under functional composition.

15.3.4 *Determinization*

Determinization is effected by a subset construction similar to that for ordinary automata. Given a nondeterministic AGS

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

there is an equivalent deterministic AGS

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

where for $A \subseteq Q$,

$$\varepsilon_\alpha(A) \overset{\text{def}}{=} \begin{cases} 1, & \text{if } \exists s \in A\ \exists t \in \mathsf{accept}\ t \in \hat{\Delta}_\alpha(s), \\ 0, & \text{otherwise} \end{cases} \qquad \delta_{\alpha p}(A) \overset{\text{def}}{=} \bigcup_{s \in A} \hat{\Delta}_{\alpha p}(s).$$

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

$$L(A)(x) = \begin{cases} 1, & \text{if } \exists s \in A\ \exists t \in \mathsf{accept}\ t \in \hat{\Delta}_x(s), \\ 0, & \text{otherwise;} \end{cases}$$

in particular,

$$L(\mathsf{start})(x) = 1 \Leftrightarrow\ \exists s \in \mathsf{start}\ \exists t \in \mathsf{accept}\ t \in \hat{\Delta}_x(s).$$

As these are exactly the acceptance criteria for M and N respectively, the two machines accept the same set of guarded strings.

15.4 Kleene Coalgebra with Tests (KCT)

A Kleene coalgebra with tests (KCT) is very much like Kleene coalgebra (KC) (Rutten 1998), but with the addition of Boolean tests. Formally, a *Kleene coalgebra with tests* (KCT) over Σ and T is a structure

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

kozen@cs.cornell.edu

where Q is a set of *states* and

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

for $\alpha \in \mathsf{At}$ and $p \in \Sigma$, exactly as in deterministic automata on guarded strings. Thus we can view a KCT as simply a deterministic AGS without a designated start state.

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 Σ and T also by KCT.

15.4.1 *The Brzozowski Derivative, Semantic Form*

There is a natural KCT over Σ and T defined in terms of the Brzozowski derivative on sets of guarded strings. The traditional Brzozowski derivative (Brzozowski 1964) 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

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

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

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

It is clear that the structure

$$\mathsf{Brz} \overset{\text{def}}{=} (2^{\mathsf{GS}}, D, E)$$

forms a KCT. Indeed, it is the final object in the category KCT: for any KCT $M = (Q, \delta, \varepsilon)$, the function $L : Q \to 2^{\mathsf{GS}}$ defined in (15.1) is the unique KCT morphism $L : M \to \mathsf{Brz}$.

15.4.2 *The Brzozowski Derivative, Syntactic Form*

As with Brzozowski's original formulation (Brzozowski 1964), there is also a syntactic form of the Brzozowski derivative defined on KAT expressions. Let Exp denote

kozen@cs.cornell.edu

the set of KAT expressions over Σ and T. We define a family of derivative operators

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

consisting of components

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

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

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

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

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 $p \in \Sigma$, 1 for any $b \in T$ such that $\alpha \le b$, and 0 for any $b \in T$ such that $\alpha \le \bar{b}$, then simplifies the resulting expression over the two-element Kleene algebra $\mathsf{2}$. 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 (Exp, D, E) is a KCT in the sense of Sect. 15.4, thus there is a unique KCT morphism $L : \mathsf{Exp} \to \mathsf{Brz}$ to the final coalgebra Brz defined in (15.1). We will show that $L(e) = \mathsf{GS}(e)$, where GS is the traditional interpretation of KAT expressions mentioned in Sect. 15.2.2.

Lemma 15.4.1 *For all $\alpha \in \mathsf{At}$, $p \in \Sigma$, 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 $D_{\alpha p}$,

kozen@cs.cornell.edu

$$\alpha pe' \leq e \Rightarrow\ e' = D_{\alpha p}(\alpha pe') \leq D_{\alpha p}(e).$$

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

$$\alpha p D_{\alpha p}(p) = \alpha p \leq p.$$

For the case e_1e_2,

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

For the case e^*,

$$\alpha p D_{\alpha p}(e^*) = \alpha p D_{\alpha p}(e)e^* \leq ee^* \leq e^*.$$

All other cases are equally straightforward. □

Theorem 15.4.2 *For all* KAT *expressions e,* $\mathsf{GS}(e) = L(e)$. *Thus the set accepted by the automaton* (Exp, D, E, e) *is* $\mathsf{GS}(e)$.

Proof We wish to show that for all $x \in \mathsf{GS}$, $x \in \mathsf{GS}(e)$ iff $L(e)(x) = 1$. By the completeness theorem for KAT (Kozen and Smith 1996), we have $x \in \mathsf{GS}(e)$ iff $x \leq e$, so it suffices to show that $x \leq e$ 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 $x = \alpha py$, by Lemma 15.4.1,

$$\alpha py \leq e \Leftrightarrow\ y \leq D_{\alpha p}(e)\ \Leftrightarrow\ L(D_{\alpha p}(e))(y) = 1\ \Leftrightarrow\ L(e)(apy) = 1.$$

□

15.5 Completeness

15.5.1 Bisimulation on KCT*s*

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

(i) $\varepsilon_\alpha(s) = \varepsilon'_\alpha(t)$; and
(ii) $\delta_{\alpha p}(s) \equiv \delta'_{\alpha p}(t)$.

kozen@cs.cornell.edu

Lemma 15.5.1 *The relation*

$$s \mathrel{\hat{\equiv}} t \overset{\text{def}}{\Longleftrightarrow} L(s) = L(t)$$

is the unique maximal bisimulation between M and M'.

Proof It is easily shown that $\hat{\equiv}$ satisfies (i) and (ii). Moreover, if $\equiv$ is any relation satisfying (i) and (ii), one can show by a straightforward inductive argument that $\equiv$ refines $\hat{\equiv}$, thus $\hat{\equiv}$ 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 KCT-*congruence*. KCT-congruences are exactly the kernels of KCT-morphisms.

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

15.5.2 Bisimulation on Deterministic Automata

For deterministic automata, we add an extra condition. A *bisimulation* between two deterministic AGS $M = (Q, \delta, \varepsilon, \mathsf{start})$ and $M' = (Q', \delta', \varepsilon', \mathsf{start}')$ is a bisimulation $\equiv$ between the underlying KCTs (Q, δ, ε) and $(Q', \delta', \varepsilon')$ such that $\mathsf{start} \equiv \mathsf{start}'$. Two automata are *bisimilar* if there exists a bisimulation between them.

Lemma 15.5.2 *M and M' are bisimilar iff $\mathsf{GS}(M) = \mathsf{GS}(M')$.*

Proof Let $\hat{\equiv}$ be the relation defined in the proof of Lemma 15.5.1. If $\mathsf{GS}(M) = \mathsf{GS}(M')$, then $L(\mathsf{start}) = L(\mathsf{start}')$ by the definition of acceptance, therefore $\mathsf{start} \mathrel{\hat{\equiv}} \mathsf{start}'$. Then M and M' are bisimilar under $\hat{\equiv}$.

Conversely, if there exists a bisimulation $\equiv$ between M and M', then $\mathsf{start} \equiv \mathsf{start}'$, and by Lemma 15.5.1, $\equiv$ refines $\hat{\equiv}$, therefore $\mathsf{start} \mathrel{\hat{\equiv}} \mathsf{start}'$. Thus $\hat{\equiv}$ 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 15.5.3 (Completeness) *The following are equivalent:*

(i) the automata (Exp, D, E, e) and (Exp, D, E, e') are bisimilar;
(ii) $L(e) = L(e')$;
(iii) $\mathsf{GS}(e) = \mathsf{GS}(e')$;

kozen@cs.cornell.edu

(iv) e and e' are equivalent modulo the axioms of KAT.

Proof The equivalence of (i)–(iii) follows from Theorem 15.4.2 and Lemma 15.5.2. The equivalence of (iii) and (iv) are just the soundness and completeness of KAT for the guarded string model (Kozen and Smith 1996). □

15.6 Complexity

Let Exp_e denote the subautomaton of (Exp, D, E, e) consisting of those expressions that are accessible from e; that is, those expressions of the form $\hat{D}_x(e)$ for some $x \in (\mathsf{At} \cdot \Sigma)^*$. Theorem 15.5.3 by itself is not very useful as a deductive system or decision procedure for equivalence, because Exp_e is not a finite system in general. However, equivalent finite systems exist. In particular, by Theorem 15.5.3, KAT equivalence is the maximal congruence on Exp. The quotient with respect to this relation, ignoring inaccessible states, gives the minimal deterministic AGS accepting $\mathsf{GS}(e)$, which is finite since $\mathsf{GS}(e)$ is regular.

Unfortunately, to construct this automaton directly, we would need an independent algorithm to decide KAT equivalence. However, we can obtain finite automata with finer congruences that are easier to decide than full KAT equivalence. Chen and Pucella (2003) 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

$$1 \cdot x = x \qquad 0 \cdot x = 0 \qquad (x + y) \cdot z = xz + yz. \tag{15.2}$$

Multiplicative associativity is not assumed, nor is left distributivity. We might call structures satisfying these axioms *right presemirings*. We denote by $\approx$ the KAT-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 Sect. 15.3.4. 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 15.6.1 *The relation $\approx$ is a* KCT*-congruence on* Exp.

kozen@cs.cornell.edu

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 15.5.3 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 (15.2),

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

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,

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

and

$$D_{\alpha p}(e_1^*) = D_{\alpha p}(e_1)e_1^* \approx D_{\alpha p}(e_2)e_2^* = D_{\alpha p}(e_2^*).$$

□

15.6.1 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 $\{\hat{D}_x(e)/{\approx} \mid x \in (\mathsf{At}\cdot\Sigma)^*\}$ 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:

$$\begin{array}{ccc}
\dfrac{e \in \mathrm{cl}(e_1)}{e \in \mathrm{cl}(e_1+e_2)} & \dfrac{e \in \mathrm{cl}(e_1)}{ee_2 \in \mathrm{cl}(e_1e_2)} & \dfrac{e \in \mathrm{cl}(e_1)}{ee_1^* \in \mathrm{cl}(e_1^*)}\\[2ex]
\dfrac{e \in \mathrm{cl}(e_2)}{e \in \mathrm{cl}(e_1+e_2)} & \dfrac{e \in \mathrm{cl}(e_2)}{e \in \mathrm{cl}(e_1e_2)} & \dfrac{e \in \mathrm{cl}(b)}{e \in \mathrm{cl}(\bar{b})}
\end{array} \qquad (15.3)$$

kozen@cs.cornell.edu

Lemma 15.6.2 *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 $\mathrm{cl}'(e) = \mathrm{cl}(e) - \{1\}$. For $e \in \Sigma \cup T$, $\mathrm{cl}'(e) = \{e\}$. For the other operators, from the rules (15.3) 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. $\square$

15.6.2 *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 $p \in \Sigma$,

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

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

$$\begin{aligned}
\Delta_p(\alpha, q) &\overset{\text{def}}{=} \begin{cases} \{1\}, & \text{if } q \in \Sigma \text{ and } q = p,\\ \emptyset, & \text{if } q \in \Sigma \text{ and } q \neq p \text{ or } q \in B,\end{cases}\\
\Delta_p(\alpha, e_1+e_2) &\overset{\text{def}}{=} \Delta_p(\alpha, e_1) \cup \Delta_p(\alpha, e_2),\\
\Delta_p(\alpha, e_1e_2) &\overset{\text{def}}{=} \begin{cases} \{ee_2 \mid e \in \Delta_p(\alpha, e_1)\} \cup \Delta_p(\alpha, e_2), & \text{if } E_\alpha(e_1) = 1,\\ \{ee_2 \mid e \in \Delta_p(\alpha, e_1)\}, & \text{if } E_\alpha(e_1) = 0,\end{cases}\\
\Delta_p(\alpha, e_1^*) &\overset{\text{def}}{=} \{ee_1^* \mid e \in \Delta_p(\alpha, e_1)\}.
\end{aligned}$$

Due to the bipartite structure of the states, we have $\hat{\Delta}_{\alpha p} = \Delta_p \bullet \Delta_\alpha$, where $\hat{\Delta}$ is the extension of Δ defined in Sect. 15.3.2. Then

$$\hat{\Delta}_{\alpha p}(e) = (\Delta_p \bullet \Delta_\alpha)(e) = \bigcup \{\Delta_p(\alpha, e)\} = \Delta_p(\alpha, e). \tag{15.4}$$

kozen@cs.cornell.edu

We thus have

$$\hat{\Delta}_{\alpha p}(q) \overset{\text{def}}{=} \begin{cases} \{1\}, & \text{if } q \in \Sigma \text{ and } q = p, \\ \emptyset, & \text{if } q \in \Sigma \text{ and } q \neq p \text{ or } q \in B, \end{cases}$$

$$\hat{\Delta}_{\alpha p}(e_1 + e_2) \overset{\text{def}}{=} \hat{\Delta}_{\alpha p}(e_1) \cup \hat{\Delta}_{\alpha p}(e_2),$$

$$\hat{\Delta}_{\alpha p}(e_1 e_2) \overset{\text{def}}{=} \begin{cases} \{ee_2 \mid e \in \hat{\Delta}_{\alpha p}(e_1)\} \cup \hat{\Delta}_{\alpha p}(e_2), & \text{if } E_\alpha(e_1) = 1, \\ \{ee_2 \mid e \in \hat{\Delta}_{\alpha p}(e_1)\}, & \text{if } E_\alpha(e_1) = 0, \end{cases}$$

$$\hat{\Delta}_{\alpha p}(e_1^*) \overset{\text{def}}{=} \{ee_1^* \mid e \in \hat{\Delta}_{\alpha p}(e_1)\}.$$

Lemma 15.6.3 *For all* KAT *terms e and* $x \in (\mathsf{At} \cdot \Sigma)^*$, $\hat{\Delta}_x(e) \subseteq \mathrm{cl}(e)$.

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

$$\hat{\Delta}_{\alpha p}(e_1 + e_2) = \hat{\Delta}_{\alpha p}(e_1) \cup \hat{\Delta}_{\alpha p}(e_2) \subseteq \mathrm{cl}(e_1) \cup \mathrm{cl}(e_2) \subseteq \mathrm{cl}(e_1 + e_2)$$

$$\begin{aligned} \hat{\Delta}_{\alpha p}(e_1 e_2) &= \begin{cases} \{ee_2 \mid e \in \hat{\Delta}_{\alpha p}(e_1)\} \cup \hat{\Delta}_{\alpha p}(e_2), & \text{if } E_\alpha(e_1) = 1 \\ \{ee_2 \mid e \in \hat{\Delta}_{\alpha p}(e_1)\}, & \text{if } E_\alpha(e_1) = 0 \end{cases} \\ &\subseteq \{ee_2 \mid e \in \mathrm{cl}(e_1)\} \cup \mathrm{cl}(e_2) \subseteq \mathrm{cl}(e_1 e_2) \end{aligned}$$

$$\hat{\Delta}_{\alpha p}(e_1^*) = \{ee_1^* \mid e \in \hat{\Delta}_{\alpha p}(e_1)\} \subseteq \{ee_1^* \mid e \in \mathrm{cl}(e_1)\} \subseteq \mathrm{cl}(e_1^*).$$

For arbitrary $x \in (\mathsf{At} \cdot \Sigma)^*$, 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 $y \neq \varepsilon$,

$$\begin{aligned} \hat{\Delta}_{xy}(e) &= (\hat{\Delta}_y \bullet \hat{\Delta}_x)(e) = \bigcup \{\hat{\Delta}_y(d) \mid d \in \hat{\Delta}_x(e)\} \\ &\subseteq \bigcup \{\mathrm{cl}(d) \mid d \in \mathrm{cl}(e)\} = \mathrm{cl}(e). \end{aligned}$$ □

Lemma 15.6.4 *For all* KAT *terms e and* $x \in (\mathsf{At} \cdot \Sigma)^*$, $\hat{D}_x(e) \approx \sum \hat{\Delta}_x(e)$.

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

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

For $b \in B$,

$$D_{\alpha p}(b) = 0 = \sum \emptyset = \sum \hat{\Delta}_{\alpha p}(b).$$

kozen@cs.cornell.edu

For the other operators,

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

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

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

Now we show the result for arbitrary $x \in (\mathsf{At} \cdot \Sigma)^*$ 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 $y \neq \varepsilon$,

$$\begin{aligned} \hat{D}_{xy}(e) &= \hat{D}_y(\hat{D}_x(e)) \\ &\approx \hat{D}_y\left(\sum \hat{\Delta}_x(e)\right) \qquad \text{by Lemma 6.1} \\ &= \sum\{\hat{D}_y(d) \mid d \in \hat{\Delta}_x(e)\} \\ &\approx \sum\left\{\sum \hat{\Delta}_y(d) \middle| \in \hat{\Delta}_x(e)\right\} \approx \sum \bigcup \{\hat{\Delta}_y(d) \mid d \in \hat{\Delta}_x(e)\} \\ &= \sum(\hat{\Delta}_y \bullet \hat{\Delta}_x)(e) = \sum \hat{\Delta}_{xy}(e). \end{aligned}$$

□

Theorem 15.6.5 *The automaton* $\mathsf{Exp}_e/{\approx}$ *has at most* $2^{|e|+1}$ *accessible states.*

Proof The accessible states of $\mathsf{Exp}_e/{\approx}$ are $\{\hat{D}_x(e)/{\approx} \mid x \in (\mathsf{At} \cdot \Sigma)^*\}$, where $d/{\approx}$ is the congruence class of d modulo $\approx$. The stated bound follows from Lemmas 15.6.2, 15.6.3, and 15.6.4. □

15.6.3 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 Sect. 15.3.4 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 Kozen (2003), but the current treatment parallels more closely Brzozowski's original treatment for ordinary regular expressions (Brzozowski 1964) and aligns with the general coalgebraic structure of Bonsangue et al. (2007, 2009).

Define the nondeterministic automaton

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

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 Sect. 15.6.2, and the start and accept states are

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

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 (15.4) and Lemma 15.6.3.

Now let

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

be the deterministic automaton obtained from N_e by the subset construction as described in Sect. 15.3.4. The start state of M_e is $\{e\}$, and δ and ε are given by

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

Note that the accessible states are all of the form $A \subseteq \mathrm{cl}(e)$, thus by Lemma 15.6.2, M_e has at most $2^{|e|+1}$ accessible states.

Theorem 15.6.6 *For $A \subseteq \mathsf{Exp}$, the map $A \mapsto (\sum A)/{\approx}$ is a KCT-morphism. 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,

kozen@cs.cornell.edu

$$\begin{aligned}\sum \delta_{\alpha p}(A) &= \sum \bigcup \{\hat{\Delta}_{\alpha p}(d) \mid d \in A\} \\ &\approx \sum \left\{ \sum \hat{\Delta}_{\alpha p}(d) \middle| d \in A \right\} \\ &\approx \sum \{D_{\alpha p}(d) \mid d \in A\} && \text{by Lemma 6.4} \\ &= D_{\alpha p}\left(\sum A\right),\end{aligned}$$

therefore

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

For ε,

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

The map also preserves start states:

$$\{e\} \mapsto \left(\sum \{e\}\right) \Big/ {\approx} \; = \; e/{\approx}.$$

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

15.6.4 Automatic Proof Generation in PSPACE

The results of Sects. 15.6.2 and 15.6.3 give rise to a nondeterministic linear-space algorithm for deciding the equivalence of two given KAT terms. By Savitch's theorem (Savitch 1970), 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 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 cl(e). To define the correspondence, we view terms as labeled trees; that is, as partial functions

$$e : \omega^* \to \Sigma \cup T \cup \{+, \cdot, {}^*, \bar{}, 0, 1\}$$

with domain of definition dom $e \subseteq \omega^*$ such that

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

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

kozen@cs.cornell.edu

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

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

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

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 Sect. 15.6.3, representing the states by $\operatorname{dom} e_1$ and $\operatorname{dom} e_2$, 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 $y \in (\mathsf{At} \cdot \Sigma)^*$ 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

$$E_\alpha\left(\sum_{\tau \in A_1} R(\tau, e_1)\right) \neq E_\alpha\left(\sum_{\tau \in A_2} R(\rho, e_2)\right),$$

where A_1 and A_2 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

$$L(e_i)(x) = E_\alpha(\hat{D}_y(e_i)) = E_\alpha\left(\sum_{\tau \in A_i} R(\tau, e_i)\right)$$

for $i \in \{1, 2\}$, 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 $\operatorname{dom} e_1$ and $\operatorname{dom} e_2$, 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 Sect. 15.3.4, 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

kozen@cs.cornell.edu

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 (Savitch 1970).

Acknowledgements Many thanks to Jiří Adámek, Marcello Bonsangue, Helle Hvid Hansen, Raul Leal, Jan Rutten, Mehrnoosh Sadrzadeh, Luigi Santocanale, Alexandra Silva, Yde Venema, and James Worthington. An earlier version of this work (Kozen 2008) was presented at CMCS 2008 and was supported by NSF Grant CCF-0635028. The preparation of this revised version was supported by NSF grants CCF-1535952 and CCF-1637532 and by the National Security Agency.

References

Bonsangue, M. M., Rutten, J. J. M. M., & Silva, A. M. (2007). *Regular expressions for polynomial coalgebras*. Technical Report SEN-E0703, Centrum voor Wiskunde en Informatica, Amsterdam.

Bonsangue, M. M., Rutten, J. J. M. M., & Silva, A. M. (2009). A Kleene theorem for polynomial coalgebras. In L. de Alfaro (Ed.), *Proceedings of the 12th international conference foundations of software science and computation structures (FoSSaCS 2009)* (Vol. 5504, pp. 122–136)., of lecture notes in computer science New York: Springer.

Brzozowski, J. A. (1964). Derivatives of regular expressions. *Journal of the Association for Computing Machinery*, *11*, 481–494.

Chen, H., & Pucella, R. (2003). A coalgebraic approach to Kleene algebra with tests. *Electronic Notes in Theoretical Computer Science*, *82*(1),

Cohen, E., Kozen, D., & Smith, F. (1996). *The complexity of Kleene algebra with tests*. Technical Report TR96-1598, Computer Science Department, Cornell University.

Conway, J. H. (1971). *Regular algebra and finite machines*. London: Chapman and Hall.

Kaplan, D. M. (1969). Regular expressions and the equivalence of programs. *Journal of Computer and System Sciences*, *3*, 361–386.

Kleene, S. C. (1956). Representation of events in nerve nets and finite automata. In C. E. Shannon & J. McCarthy (Eds.), *Automata studies* (pp. 3–41). Princeton, NJ: Princeton University Press.

Kozen, D. (1994). A completeness theorem for Kleene algebras and the algebra of regular events. *Computing and Information*, *110*(2), 366–390.

Kozen, D. (1997). Kleene algebra with tests. *Transactions on Programming Languages and Systems*, *19*(3), 427–443.

Kozen, D. (2000). On Hoare logic and Kleene algebra with tests. *Transactions on Computational Logic*, *1*(1), 60–76.

Kozen, D. (2003). Automata on guarded strings and applications. *Matématica Contemporânea*, *24*, 117–139.

Kozen, D. (2008). *On the coalgebraic theory of Kleene algebra with tests*. Technical Report.? http://hdl.handle.net/1813/10173, Computing and Information Science, Cornell University.

Kozen, D., & Smith, F. (1996). Kleene algebra with tests: Completeness and decidability. In D. van Dalen & M. Bezem (Eds.), *Proceedings of the 10th international workshop computer science logic (CSL'96)* (Vol. 1258, pp. 244–259)., of lecture notes in computer science Utrecht: Springer.

Rutten, J. J. M. M. (1998). Automata and coinduction (an exercise in coalgebra). *Proceedings of CONCUR'98* (Vol. 1466, pp. 193–217)., lecture notes in computer science Berlin: Springer.

Savitch, W. (1970). Relationship between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, *4*(2), 177–192.

Worthington, J. (2008). Automatic proof generation in Kleene algebra. In R. Berghammer, B. Möller, & G. Struth (Eds.), *10th International conference relational methods in computer science (RelMiCS10) and 5th international conference applications of Kleene algebra (AKA5)* (Vol. 4988, pp. 382–396)., of lecturer notes in computer science Berlin: Springer.

kozen@cs.cornell.edu