# Kleene Algebra with Equations

Dexter Kozen and Konstantinos Mamouras

Computer Science Department, Cornell University, Ithaca, NY 14853-7501, USA
{kozen,mamouras}@cs.cornell.edu

**Abstract.** We identify sufficient conditions for the construction of free language models for systems of Kleene algebra with additional equations. The construction applies to a broad class of extensions of KA and provides a uniform approach to deductive completeness.

## 1 Introduction

Kleene algebra (KA) is the algebra of regular expressions. Introduced by Stephen Cole Kleene in 1956, it is fundamental and ubiquitous in computer science. It has proven useful in countless applications, from program specification and verification to the design and analysis of algorithms [1–8].

One can augment KA with Booleans in a seamless way to obtain Kleene algebra with tests (KAT). Unlike many other related logics for program verification, KAT is classically based, requiring no specialized syntax or deductive apparatus other than classical equational logic. In practice, statements in the logic are typically universal Horn formulas

$$s_1 = t_1 \to s_2 = t_2 \to \cdots \to s_n = t_n \to s = t,$$

where the conclusion $s = t$ is the main target task and the premises $s_i = t_i$ are the verification conditions needed to prove it. The conclusion $s = t$ may encode a partial correctness assertion, an equivalence between an optimized and an unoptimized version of a program, or an equivalence between a program annotated with static analysis information and the unannotated program. The verification conditions $s_i = t_i$ are typically simple properties of the underlying domain of computation that describe how atomic actions interact with atomic assertions. They may require first-order interpreted reasoning, but are proven once and for all, then abstracted to propositional form. The proof of the conclusion $s = t$ from the premises takes place at the propositional level in KAT. This methodology affords a clean separation of the theory of the domain of computation from the program restructuring operations. It is advantageous to separate the two levels of reasoning, because the full first-order theory of the domain of computation may be highly undecidable, even though we may only need small parts of it. By isolating those parts, we can often maintain decidability and deductive completeness.

A typical form of premise that arises frequently in practice is a *commutativity condition* $pb = bp$ for an action $p$ and a test $b$. This captures the idea that the action $p$ does not affect the truth of $b$. For example, the action $p$ might be an

J. Esparza et al. (Eds.): ICALP 2014, Part II, LNCS 8573, pp. 280–292, 2014.
© Springer-Verlag Berlin Heidelberg 2014

assignment $x := 3$ and $b$ might be a test $y = 4$, where $x$ and $y$ are distinct variables. It is clear that the truth value of $b$ is not affected by the action $p$, so it would be the same before as after. But once this is established, we no longer need to know what $p$ and $b$ are, but only that $pb = bp$. It follows by purely equational reasoning in KAT that $p_1 b = bp_1 \to \cdots \to p_n b = bp_n \to qb = bq$, where $q$ is any program built from atomic actions $p_1, \ldots, p_n$.

In some instances, Horn formulas with premises of a certain form can be reduced to the equational theory without loss of deductive completeness or decision efficiency using a technique known as *elimination of hypotheses* [3, 9, 10]. One important class of premises for which this is possible are those of the form $s = 0$. The universal Horn theory restricted to premises of this form is called the *Hoare theory*, because it subsumes Hoare logic: the partial correctness assertion $\{b\}p\{c\}$ can be encoded as the equation $bp\bar{c} = 0$. Other forms that arise frequently in practice are $bp = b$, which says that the action $p$ is not necessary if $b$ is true, useful in optimizations to eliminate redundant actions; and $pq = qp$, which says that the atomic actions $p$ and $q$ can occur in either order with the same effect, useful in reasoning about concurrency. Unfortunately, KAT with general commutativity assumptions $pq = qp$ is undecidable [11].

As a case in point, the NetKAT system [8] incorporates a number of such equational premises as part of the theory, which are taken as additional axioms besides those of KAT. Proofs of deductive completeness and complexity as given in [8] required extensive adaptation of the analogous proofs for KA and KAT. Indeed, this was already the case with KAT, which was an adaptation of KA to incorporate an embedded Boolean algebra.

Although each of these instances was studied separately, there are some striking similarities. It turns out that the key to progress in all of them is the identification of a suitable class of *language models* that characterize the equational theory of the system. A language model is a structure in which expressions are interpreted as sets of elements of some monoid. The language models should form the free models for the system at hand. For KA, a language model is the regular sets of strings over a finite alphabet, elements of a free monoid; for KAT, the regular sets of guarded strings; for NetKAT, the regular sets of strings of a certain reduced form. Once a suitable class of language models can be determined, this opens the door to a systematic treatment of deductive completeness. It is also clear from previous work [8, 12–15] that the existence of coalgebraic decision algorithms also depends strongly on the existence of language models (although we do not develop this connection in this paper). The question thus presents itself: Is there a general set of criteria that admit a uniform construction of language models and that would apply in a broad range of situations and subsume previous ad hoc constructions? That is the subject of this paper.

Alas, such a grand unifying framework is unlikely, given the negative results of [11] and of §2. However, we have identified a framework that goes quite far in this direction. It applies in the case in which the additional equational axioms are monoid equations or partial monoid equations (as is the case in all the examples mentioned above) and is based on a well-studied class of rewrite systems

called *inverse context-free systems* [16]. We give criteria in terms of these rewrite systems that imply the existence of free language models in a wide range of previously studied instances, as well as some new ones.

This paper is organized as follows. In §2 we present preliminary definitions and our negative result limiting the applicability of the method. In §3 we establish a connection between the classical theory of string rewriting and Kleene algebra. We recall from [16] the definition of *inverse context-free rewrite systems* and the key result that they preserve regularity. The original proof involved an automata-theoretic construction, but we show that it can be carried out axiomatically in KA. In §4 we give examples of partial and total monoid equations and give a general construction that establishes completeness in those cases. The construction is a special case of the more general results of §5, but we start with it as a conceptual first step to illustrate the ideas. However, we can already derive some interesting consequences in this special case. In §5, we establish completeness for typed monoid equations. This is the most general setting covered in this paper. We give the completeness proof along with several applications. In §6 we present conclusions, future work, and open problems.

Proofs are omitted for lack of space. A full version is available online [17].

## 2 Preliminaries and a Negative Result

A *Kleene algebra* (KA) is an idempotent semiring $(K, +, \cdot, ^*, 0, 1)$ with an iteration operator $^*$ satisfying

$$1 + aa^* \leq a^* \qquad 1 + a^*a \leq a^* \qquad ax \leq x \Rightarrow a^*x \leq x \qquad xa \leq x \Rightarrow xa^* \leq x$$

where $\leq$ refers to the natural partial order on $K$: $a \leq b \Leftrightarrow a + b = b$. A *Kleene algebra with tests* (KAT) is a two-sorted structure $(K, B, +, \cdot, ^*, ^{\bar{}}, 0, 1)$ such that $(K, +, \cdot, ^*, 0, 1)$ is a KA, $(B, +, \cdot, ^{\bar{}}, 0, 1)$ is a Boolean algebra, and $(B, +, \cdot, 0, 1)$ is a subalgebra of $(K, +, \cdot, 0, 1)$ as an idempotent semiring.

Let $\Sigma$ be a finite alphabet of symbols. The *free monoid* $(\Sigma^*, \cdot, \epsilon)$ generated by $\Sigma$ is the set $\Sigma^*$ of words over $\Sigma$ together with the operation $\cdot$ of string concatenation and the empty string $\epsilon$ as identity. To generalize this construction, we consider a *finitely presented monoid* $M = \langle a, b, \ldots \mid u_1 \equiv u_2, v_1 \equiv v_2, \ldots \rangle$ with a finite set of generators $\Sigma = \{a, b, \ldots\}$ and a finite set of relations $R = \{(u_1, u_2), (v_1, v_2), \ldots\}$. We interchangeably write a relation as an equation $u \equiv u'$ or as a pair $(u, u')$. Let $\leftrightarrow_R^*$ be the smallest congruence on $\Sigma^*$ that contains $R$. The congruence class of a string $u$ is denoted by $[u]$. The finitely presented monoid $M = \langle \Sigma \mid R \rangle = \Sigma^*/R$ has the congruence classes $\{[u] \mid u \in \Sigma^*\}$ of $\leftrightarrow_R^*$ as its carrier. Multiplication is given by $[u] \cdot [v] \mapsto [uv]$, and the identity is $[\epsilon]$.

We define *regular expressions* over the alphabet $\Sigma$ to be the terms given by the grammar $e, e_1, e_2 ::= a \in \Sigma \mid 1 \mid 0 \mid e_1 + e_2 \mid e_1; e_2 \mid e^*$. We can interpret a regular expression as a subset of a finitely presented monoid $M = \langle \Sigma \mid R \rangle$ with multiplication $\cdot$ and identity $1_M = [\epsilon]$. The function $\mathcal{R}_M$, called *language interpretation* in $M$, sends a regular expression to a set of elements of $M$:

$$\begin{array}{ll} \mathcal{R}_M(a) = \{[a]\} & \mathcal{R}_M(e_1 + e_2) = \mathcal{R}_M(e_1) \cup \mathcal{R}_M(e_2) \\ \mathcal{R}_M(1) = \{1_M\} & \mathcal{R}_M(e_1; e_2) = \mathcal{R}_M(e_1) \cdot \mathcal{R}_M(e_2) \\ \mathcal{R}_M(0) = \emptyset & \mathcal{R}_M(e^*) = \bigcup_{n \geq 0} \mathcal{R}_M(e)^n \end{array}$$

where $\cdot$ on sets is given by $A \cdot B = \{u \cdot v \mid u \in A, v \in B\}$, and $A^n$ is defined inductively as $A^0 = \mathcal{R}_M(1)$ and $A^{n+1} = A^n \cdot A$. The image of the interpretation $\mathcal{R}_M$ together with the operations $\cup$, $\cdot$, $^*$, $\emptyset$, $\{1_M\}$ is the *algebra of regular sets* over $M$, denoted by $\mathsf{Reg}\, M$. If $M$ is the free monoid $\Sigma^*$, then $\mathcal{R}_M$ is the standard language interpretation of regular expressions.

It is known that the algebra of regular sets $\mathsf{Reg}\, \Sigma^*$ is the free Kleene algebra generated by $\Sigma$ [18]. This is equivalent to the completeness of the axioms of KA for the standard language interpretation $\mathcal{R}$ of regular expressions. That is, for any two regular expressions $e_1, e_2$ over $\Sigma$, if $\mathcal{R}(e_1) = \mathcal{R}(e_2)$ then $\mathsf{KA} \vdash e_1 \equiv e_2$. The question then arises if this result extends to the general case of $\mathsf{Reg}\, M$ for a finitely presented monoid $M = \langle \Sigma \mid R \rangle$. We ask the question of whether $\mathcal{R}_M(e_1) = \mathcal{R}_M(e_2)$ implies provability of $e_1 \equiv e_2$ in a system of KA augmented with (at least) the equations corresponding to the relations $R$.

In general, the answer to the question posed in the previous paragraph is negative. That is, there exists a finitely presented monoid $M = \langle \Sigma \mid R \rangle$ such that the equational theory of $\mathsf{Reg}\, M$ is not recursively enumerable, and therefore not recursively axiomatizable. The *equational theory* of the Kleene algebra $\mathsf{Reg}\, M$ is the set of equations between regular expressions that are true in $\mathsf{Reg}\, M$ under the interpretation $\mathcal{R}_M$, i.e., the set $\{e_1 \equiv e_2 \mid \mathcal{R}_M(e_1) = \mathcal{R}_M(e_2)\}$. We show this negative result using the ideas developed in [11]. The proof specifies a way to construct effectively the monoid whose existence we claim.

**Theorem 1.** There exists a finitely presented monoid $M$ such that the equational theory of $\mathsf{Reg}\, M$ is not recursively enumerable.

This negative result says that we can only hope to identify subclasses of finitely presented monoids $M$ such that the algebra $\mathsf{Reg}\, M$ of regular sets over $M$ is axiomatizable. The idea is to first restrict attention to those finite monoid presentations, where the equations can be oriented to give a confluent and terminating rewrite system. This allows one to consider as canonical representatives the irreducible strings of the congruence classes. Then, we focus on a subclass that allows two crucial algebraic constructions: a "descendants" automata-theoretic construction, and an "ancestors" construction, which is a homomorphism.

The proof of Theorem 1 is similar to that of [11, Theorem 4.1(ii)], but strictly speaking, neither theorem follows from the other. The theorem of [11] gives a uniform $\Pi^0_2$-lower bound when the monoid is considered part of the input, whereas Theorem 1 gives a $\Pi^0_1$-lower bound for a fixed monoid.

## 3 String Rewriting Systems

In this section we establish a connection between the classical theory of string rewriting systems and Kleene algebra. More specifically, we recall a result regarding the preservation of regularity: for every *inverse context-free* system $R$ and a regular set $L$, the set of the $R$-descendants of $L$ is also regular [16]. This result involves an automata-theoretic construction, which can be modeled in KA, because an automaton can be represented as an appropriate KA term [18].

The combinatorial arguments of the construction can then be replaced by equational reasoning in KA. As it turns out, this connection will allow us to obtain powerful completeness metatheorems in later sections.

A *string rewriting system* $R$ over a finite alphabet $\Sigma$ consists of rules $\ell \to r$, where $\ell$ and $r$ are finite strings over $\Sigma$. This extends to the *one-step rewrite relation* $\to_R$, given by $x\ell y \to_R xry$, for strings $x, y$ and rule $\ell \to r$ of $R$. If $x \to_R y$ then we say that $y$ is an *$R$-successor* of $x$, and $x$ is an *$R$-predecessor* of $y$. We write $\to_R^*$ for the reflexive-transitive closure of $\to_R$, which is called the *rewrite relation* for $R$. If $u, v$ are strings for which $u \to_R^* v$ we say that $v$ is an *$R$-descendant* of $u$, and that $u$ is an *$R$-ancestor* of $v$. For a set of strings $L$:

$$\mathsf{Desc}_R(L) = \{v \mid \exists u \in L.\ u \to_R^* v\} \quad \mathsf{Ance}_R(L) = \{u \mid \exists v \in L.\ u \to_R^* v\}$$

So, $\mathsf{Desc}_R(L)$ is the set of all the $R$-descendants of the strings in $L$, and similarly $\mathsf{Ance}_R(L)$ is the set of all $R$-ancestors of the strings in $L$. The *inverse system* $R^{-1}$ of $R$ is the system that results by taking a rule $r \to \ell$ for every rule $\ell \to r$ of $R$. If $u$ is an $R$-ancestor of a string $v$, then $u$ is an $R^{-1}$-descendant of $v$. Define $\leftrightarrow_R^*$ to be the smallest congruence on $\Sigma^*$ that contains $\{(u, v) \mid u \to v \text{ is } R\text{-rule}\}$. The congruence class of a string $u$ is denoted by $[u]$.

Let $R$ be a rewrite system. We say that $R$ is *terminating* if there is no infinite rewrite chain $x_0 \to_R x_1 \to_R x_2 \to_R \cdots$. If $R$ has rules of the form $\ell \to r$ with $|r| < |\ell|$ then it is terminating, because every rule application strictly reduces the length of the string. A string $x$ is called *$R$-irreducible* if no rule of $R$ applies to it, that is, there is no $y$ with $x \to_R y$. We say that $R$ is *confluent* if $u \to_R^* x$ and $u \to_R^* y$ imply that there exists $z$ with $x \to_R^* z$ and $y \to_R^* z$. It is said that $R$ has the *Church-Rosser property* (we also say that "$R$ is Church-Rosser") if for all strings $x, y$ with $x \leftrightarrow_R^* y$ there exists $z$ such that $x \to_R^* z$ and $y \to_R^* z$. It is a standard result that confluence and the Church-Rosser property are equivalent [16]. A system $R$ is said to be *locally (or weakly) confluent* if for all strings $u, x, y$ with $u \to_R x$ and $u \to_R y$, there exists a string $z$ such that $x \to_R^* z$ and $y \to_R^* z$. If $R$ is both locally confluent and terminating, then $R$ is confluent [16, 19].

Suppose that $R$ is confluent and terminating. We map each string $u$ to the unique $R$-irreducible string $\mathsf{nf}_R(u)$ that results from rewriting $u$ as much as possible. For strings $u, v$, it holds that $u \leftrightarrow_R^* v$ iff $\mathsf{nf}_R(u) = \mathsf{nf}_R(v)$. So, two strings are congruent iff they can be rewritten to the same $R$-irreducible. For every congruence class $[u]$ of $\leftrightarrow_R^*$, we choose as *canonical representative* (normal form) the $R$-irreducible string $\mathsf{nf}_R(u)$.

**Definition 1 (Total Coalesced Product).** Assume that $R$ is confluent and terminating, and let $I_R$ be the set of $R$-irreducible strings. Define the binary operation $\diamond$ on $I_R$, called *coalesced product*, by $u \diamond v = \mathsf{nf}_R(uv)$. We lift the operation to sets of $R$-irreducible strings as $A \diamond B = \{u \diamond v \mid u \in A,\ v \in B\}$.

**Definition 2.** Let $R$ be an arbitrary string rewrite system. For a language $L \subseteq \Sigma^*$, we define $\mathscr{C}_R(L) = \bigcup_{u \in L} [u] = \{v \mid \exists u \in L.\ v \leftrightarrow_R^* u\}$. Assume additionally that $R$ is confluent and terminating, so that the function $\mathsf{nf}_R$ is well-defined. For $L \subseteq \Sigma^*$, we define $\mathscr{G}_R(L) = \{\mathsf{nf}_R(u) \mid u \in L\}$.

**Lemma 1.** Let $R$ be a confluent and terminating rewrite system over $\Sigma$.

1. $\mathscr{C}_R(L) = \bigcup\{[u] \mid u \in \mathscr{G}_R(L)\}$, for a language $L \subseteq \Sigma^*$.
2. $\mathscr{G}_R(L_1) = \mathscr{G}_R(L_2)$ iff $\mathscr{C}_R(L_1) = \mathscr{C}_R(L_2)$, for languages $L_1, L_2 \subseteq \Sigma^*$.
3. $\mathscr{C}_R(L) = \mathsf{Ance}_R(\mathsf{Desc}_R(L))$, for a language $L \subseteq \Sigma^*$.

A rewrite system $R$ is said to *preserve regularity* if for every regular language $L$, the $R$-descendants $\mathsf{Desc}_R(L)$ form a regular set. A system $R$ is called *inverse context-free* if it only contains rules of the form $\ell \to r$, where $|r| \leq 1$. That is, every right-hand side of a rule is either a single letter or the empty string. A classical result of the theory of string rewriting is that inverse context-free systems preserve regularity (see Chapter 4 of [16] for a detailed proof). The proof of this fact uses a construction on finite automata, which we briefly present here. We will be referring to it as the *descendants construction*. Suppose that $L$ is a regular language, recognized by an automaton $\mathcal{A}$. The automaton is possibly nondeterministic and it may have epsilon transitions. We will describe a sequence of transformations on $\mathcal{A}$. When the sequence reaches a fixpoint, we obtain an automaton (nondeterministic with epsilon transitions) that recognizes $\mathsf{Desc}_R(L)$.

– Suppose that the system $R$ has a rule $\ell \to a$, where $a$ is a single letter, and $\ell = \ell_1\ell_2\cdots\ell_m$ is a string of length $m$. We assume that there is an $\ell$-path from the state $q_0$ to the state $q_n$ of the automaton. That is, a sequence

$$q_0 \xrightarrow{x_1} q_1 \xrightarrow{x_2} q_2 \xrightarrow{x_3} \cdots \xrightarrow{x_{n-1}} q_{n-1} \xrightarrow{x_n} q_n,$$

where each $x_i$ is a letter or $\epsilon$, $x_1 \cdot x_2 \cdot \ldots \cdot x_n = \ell$, and each $q_{i-1} \xrightarrow{x_i} q_i$ is a transition of the automaton. We add the transition $q_0 \xrightarrow{a} q_n$. The idea is that if the automaton accepts $x\ell y$, then it should also accept the $R$-descendant $xay$.

– Similarly, suppose that the system $R$ has a rule $\ell \to \epsilon$, where $\epsilon$ is the empty string, and that there is an $\ell$-path from the state $q_0$ to the state $q_n$. Then, we add the epsilon transition $q_0 \xrightarrow{\epsilon} q_n$ to the transition table of the automaton.

This process is iterated until no new transitions are added. The resulting automaton accepts exactly the set of $R$-descendants $\mathsf{Desc}_R(L)$.

**Theorem 2.** Let $R$ be an inverse context-free rewrite system and $e$ a regular expression whose interpretation is $L = \mathcal{R}(e)$. We can construct effectively a new regular expression $\hat{e}$ such that $\mathsf{KA}_R \vdash e \equiv \hat{e}$ and $\mathcal{R}(\hat{e}) = \mathsf{Desc}_R(L)$. $\mathsf{KA}_R$ is the system $\mathsf{KA}$ augmented with an equation $\ell \equiv r$ for every rewrite rule $\ell \to r$ of $R$.

Theorem 2 says that the descendants construction, which is combinatorial, can be modeled algebraically in the system of KA with some extra equations. This is a central technical result that we will use for our later theorems.

## 4 Completeness: (Partial) Monoid Equations

In this section we present our first completeness metatheorems, from which we can prove the existence of free language models for systems of KA with extra monoid and partial monoid equations. Our metatheorems are not only a conceptual first step towards the more general typed monoid case, which we investigate in §5, but they also allow us to obtain previously unknown completeness results. As a concrete novel application, think of the assignment statement $x := c$,

where $c$ is a constant. The action $x := c$ is idempotent, meaning that the effect of $x := c; x := c$ is the same as the effect of $x := c$. We express this fact with the monoid equation $aa \equiv a$, where $a$ is a single letter abstraction of the assignment. KA can be augmented with any number of such idempotence equations, and our metatheorem implies the existence of a free language model (see Example 1).

**Definition 3 (Language Interpretation).** Let $R$ be a confluent and terminating rewrite system. The corresponding coalesced product is $\diamond$. We define the function $\mathcal{G}_R$ that sends a regular expression to a set of $R$-irreducibles:

$$\begin{array}{rlrl} \mathcal{G}_R(a) = \{\mathsf{nf}_R(a)\} & & \mathcal{G}_R(e_1 + e_2) = \mathcal{G}_R(e_1) \cup \mathcal{G}_R(e_2) \\ \mathcal{G}_R(0) = \emptyset & & \mathcal{G}_R(e_1; e_2) = \mathcal{G}_R(e_1) \diamond \mathcal{G}_R(e_2) \\ \mathcal{G}_R(1) = \{\mathsf{nf}_R(\epsilon)\} & & \mathcal{G}_R(e^*) = \bigcup_{n \geq 0} \mathcal{G}_R(e)^{\langle n \rangle} \end{array}$$

where, for a set $A$ of $R$-irreducibles, $A^{\langle n \rangle}$ is defined by $A^{\langle 0 \rangle} = \mathcal{G}_R(1)$ and $A^{\langle n+1 \rangle} = A^{\langle n \rangle} \diamond A$. We also define the interpretation $\mathcal{C}_R(e) = \mathscr{C}_R(\mathcal{R}(e)) = \bigcup_{u \in \mathcal{R}(e)} [u]$.

Let $R$ be a confluent and terminating system over $\Sigma$, and $M = \langle \Sigma \mid R \rangle$ be the corresponding monoid. For a regular expression $e$, we have that $\mathcal{R}_M(e) = \{[u] \mid u \in \mathcal{G}_R(e)\}$. The algebra $\mathsf{Reg}\, M$ is isomorphic to the algebra that is the image of $\mathcal{G}_R$. This implies that $\mathcal{R}_M(e_1) = \mathcal{R}_M(e_2)$ iff $\mathcal{G}_R(e_1) = \mathcal{G}_R(e_2)$. So, our investigations of completeness can be w.r.t. the interpretation $\mathcal{G}_R$.

**Lemma 2.** Let $R$ be a confluent and terminating string rewrite system.

1. $\mathcal{G}_R(e) = \{\mathsf{nf}_R(u) \mid u \in \mathcal{R}(e)\} = \mathscr{G}_R(\mathcal{R}(e))$, for an expression $e$.
2. $\mathcal{C}_R(e) = \bigcup\{[v] \mid v \in \mathcal{G}_R(e)\}$, for an expression $e$.
3. $\mathcal{G}_R(e_1) = \mathcal{G}_R(e_2)$ iff $\mathcal{C}_R(e_1) = \mathcal{C}_R(e_2)$, for expressions $e_1$, $e_2$.

**Definition 4 (Well-Behaved Rewrite System).** Let $R$ be a rewrite system over $\Sigma$. We say that $R$ is *well-behaved* if it consists of finitely many rules $\ell \to r$ with $|r| = 1$ and $|\ell| > 1$, and it additionally satisfies confluence and the following property: For every letter $a$ of the alphabet, the $R$-ancestors of $a$ form a regular set $\mathcal{R}(e_a)$ for some expression $e_a$, so that $\mathsf{KA}_R \vdash e_a \equiv a$. Recall that $\mathsf{KA}_R$ is the system of KA extended with equations corresponding to the rules of $R$.

Intuitively, we say that $R$ is well-behaved if it allows two important algebraic constructions. First, the special form of the rules allows the automata-theoretic descendants construction (described in §3), which can be modeled in KA, because automata can be encoded as matrices. Then, the regularity requirement for the sets of $R$-ancestors of single letters implies that we can apply a homomorphism to obtain all the ancestors of a regular set. We can thus "close" a regular expression under the congruence induced by $R$.

**Theorem 3 (Completeness).** *Let $R$ be a well-behaved rewrite system over $\Sigma$. For any expressions $e_1$ and $e_2$, $\mathcal{G}_R(e_1) = \mathcal{G}_R(e_2)$ implies that $\mathsf{KA}_R \vdash e_1 \equiv e_2$.*

**Example 1 (Idempotence Hypotheses).** We will see how the general completeness metatheorem we have shown (Theorem 3) can be used to obtain a completeness result for the regular algebra of a simple finitely presented monoid. Consider the monoid $M = \langle a, b \mid aa \equiv a \rangle$. The rewrite system $R$ contains only the rule $aa \to a$. In order to invoke Theorem 3 we verify that $R$ is well-behaved:

- For the only rule $\ell = aa \to a = r$ of $R$, we have that $|r| = 1$ and $|\ell| > 1$.
- To show confluence of $R$, it is sufficient to show local confluence, since $R$ is terminating. This is known as Newman's Lemma (see [16, 19]). We have the following critical-pair lemma: Suppose that $u \to x$ and $u \to y$. If $x = y$, we are done. If $x \neq y$, then $u, x, y$ must be of the following forms: $u = v_1 a^{m+1} v_2 a^{n+1} v_3$, $x = v_1 a^m v_2 a^{n+1} v_3$, and $y = v_1 a^{m+1} v_2 a^n v_3$. Notice now that $x, y \to v_1 a^m v_2 a^n v_3$, which establishes local confluence.
- For the $R$-ancestors of the letters $a$ and $b$, we see that $\mathsf{Ance}_R(b) = \{b\}$, and $\mathsf{Ance}_R(a) = \{a^i \mid i \geq 1\} = \mathcal{R}(a^+)$, where $a^+ = a; a^*$. We put $e_b = b$ and $e_a = a^+$. Clearly, $\mathsf{KA}_R \vdash e_b \equiv b$. Reasoning in $\mathsf{KA}_R$: $a \leq a^+$ and $a^+ = a; a^* \leq a \Longleftarrow a; a \leq a \Longleftarrow a; a \equiv a$. We have thus shown that $\mathsf{KA}_R \vdash e_a \equiv a$.

Since the rewrite system $R$ satisfies the conditions of Theorem 3, we get completeness of $\mathsf{KA}$ together with the equation $a; a \equiv a$ for the interpretation $\mathcal{R}_M$.

We would like to generalize our result in a way that allows us to designate certain strings as being *non-well-formed* or *undefined*. Any string with a non-well-formed substring has to be discarded from the interpretation. For a string $a_1 \cdots a_k$ over the alphabet, we declare it to be non-well-formed using the equation $a_1 \cdots a_k \equiv \bot$, where $\bot$ is a special "undefined" symbol not in the alphabet.

We define a *partial monoid* to be an algebraic structure $(M, \cdot, 1_M, \bot_M)$ satisfying the monoid axioms, as well as the equations $x \cdot \bot_M = \bot_M$ and $\bot_M \cdot x = \bot_M$. The identity is $1_M$, and $\bot_M$ is called the *undefined element* of $M$. In a presentation of a partial monoid $M_\bot = \langle \Sigma \mid x_1 \equiv y_1, x_2 \equiv y_2, \ldots, z_1 \equiv \bot, z_2 \equiv \bot, \ldots \rangle$ we allow equations $x \equiv y$ between strings over $\Sigma$ (call the collection of these $R$), as well as equations of the form $z \equiv \bot$, where $z$ is a string over $\Sigma$ ($\bot$ is not in $\Sigma$). In order to give a concrete description of the partial monoid, we consider the strings over the extended alphabet $\Sigma \cup \{\bot\}$, and the equations $R_\bot$:

$$x_i \equiv y_i \qquad z_i \equiv \bot \qquad a\bot \equiv \bot,\ \bot a \equiv \bot \quad (a \in \Sigma) \qquad \bot\bot \equiv \bot$$

Let $\sim$ be the smallest congruence on $(\Sigma \cup \{\bot\})^*$ that contains the relations $R_\bot$. The partial monoid $M_\bot$ is the set of strings $(\Sigma \cup \{\bot\})^*$ quotiented by the congruence $\sim$, and hence equal to $\langle \Sigma \cup \{\bot\} \mid R_\bot \rangle$. The identity is the $\sim$-congruence class $[\epsilon]$, and the undefined element is the class of $[\bot]$.

**Assumption 1.** We collect a list of assumptions for $(\Sigma, R, R_\bot)$. First, assume that $R$ is a confluent and terminating rewrite system over the alphabet $\Sigma$. The rewrite system $R_\bot$ extends $R$ with rules of the form $z \to \bot$, where $z \in \Sigma^*$ and $|z| \geq 2$. Moreover, $R_\bot$ contains the rule $\bot\bot \to \bot$, as well as all the rules $a\bot \to \bot$ and $\bot a \to \bot$ for every letter $a \in \Sigma$. We further assume that $R_\bot$ is terminating, and that the *seamlessness property* is satisfied: If $xzy$ is a string with $z \to \bot$ in $R_\bot$, then any $R$-successor of $xzy$ is of the form $x'z'y'$, where $z' \to \bot$ is in $R_\bot$. Intuitively, seamlessness says that if a string contains a non-well-formed substring, then no $R$-rewriting can make it well-formed.

**Definition 5 (Partial Coalesced Product).** Let $(\Sigma, R, R_\bot)$ satisfy Assumption 1. Define the partial *coalesced product* $\diamond$ on $R_\bot$-irreducibles in $\Sigma^*$:

$$u \diamond v = \mathsf{nf}_R(uv), \text{ if } uv \not\sim \bot; \qquad u \diamond v = \text{undefined}, \text{ if } uv \sim \bot.$$

The condition $uv \not\to \bot$ is equivalent to $\mathsf{nf}_R(uv)$ not having a substring $z$ with $z \to \bot$. We lift the coalesced product into a total operation on sets of $R_\bot$-irreducibles: $A \diamond B = \{u \diamond v \mid u \diamond v \text{ exists}, u \in A, v \in B\}$.

**Definition 6 (Language Interpretation).** Let $(\Sigma, R, R_\bot)$ satisfy Assumption 1. For a string $u$, define $[u]_\Sigma = \Sigma^* \cap [u]$. For a language $L \subseteq \Sigma^*$, put:

$$\mathscr{G}_{R_\bot}(L) = \{\mathsf{nf}_R(u) \mid u \in L\} \setminus [\bot]_\Sigma \qquad \mathscr{C}_{R_\bot}(L) = [\bot]_\Sigma \cup \bigcup_{u \in L} [u]_\Sigma$$

Now, $\mathcal{G}_{R_\bot}$ sends a regular expression to a set of $R_\bot$-irreducibles of $\Sigma^*$:

$$\begin{aligned} \mathcal{G}_{R_\bot}(a) &= \{\mathsf{nf}_R(a)\} \setminus [\bot]_\Sigma & \mathcal{G}_{R_\bot}(e_1 + e_2) &= \mathcal{G}_{R_\bot}(e_1) \cup \mathcal{G}_{R_\bot}(e_2) \\ \mathcal{G}_{R_\bot}(0) &= \emptyset & \mathcal{G}_{R_\bot}(e_1; e_2) &= \mathcal{G}_{R_\bot}(e_1) \diamond \mathcal{G}_{R_\bot}(e_2) \\ \mathcal{G}_{R_\bot}(1) &= \{\mathsf{nf}_R(\epsilon)\} \setminus [\bot]_\Sigma & \mathcal{G}_{R_\bot}(e^*) &= \textstyle\bigcup_{n \geq 0} \mathcal{G}_{R_\bot}(e)^{\langle n \rangle} \end{aligned}$$

where $A^{\langle 0 \rangle} = \mathcal{G}_{R_\bot}(1)$ and $A^{\langle n+1 \rangle} = A^{\langle n \rangle} \diamond A$. Define $\mathcal{C}_{R_\bot}(e) = \mathscr{C}_{R_\bot}(\mathcal{R}(e))$. The interpretation $\mathcal{G}_{R_\bot}$ discards the undefined strings, but $\mathcal{C}_{R_\bot}$ adds them all in.

**Definition 7 (Well-Behaved).** We suppose that $(\Sigma, R, R_\bot)$ satisfies Assumption 1. We say that it is *well-behaved* if $R_\bot$ consists of finitely many rules, every rule $\ell \to r$ of $R$ satisfies $|r| = 1$ and $|\ell| > 1$, and it satisfies the property: For every letter $a$ of the alphabet, the $R$-ancestors of $a$ form a regular set $\mathcal{R}(e_a)$ for some regular expression $e_a$, so that $\mathsf{KA}_R \vdash e_a \equiv a$. The empty string and the single-letter strings are $R_\bot$-irreducible. We write $\mathsf{KA}_{R_\bot}$ for the system $\mathsf{KA}_R$ extended with an equation $a_1; \cdots; a_k \equiv 0$ for every rule $a_1 \cdots a_k \to \bot$ of $R_\bot$.

**Theorem 4 (Completeness).** *Suppose that $(\Sigma, R, R_\bot)$ is well-behaved. Then, $\mathcal{G}_{R_\bot}(e_1) = \mathcal{G}_{R_\bot}(e_2)$ implies that $\mathsf{KA}_{R_\bot} \vdash e_1 \equiv e_2$.*

## 5 Completeness: Typed Monoid Equations

We further generalize the partial monoid setting by assuming more structure on the strings and the rewrite system. One major difference from the partial monoid case is the introduction of a new category of primitive symbols, the subidentities, which allow the encoding of Booleans. We show how to cover several examples: plain KAT, KAT with simple Hoare hypotheses $b; p; c \equiv 0$, KAT with hypotheses $c; p \equiv c$, and NetKAT. There are even more applications which for lack of space we do not present here: commutativity equations $b; p \equiv p; b$ (test $b$, atomic action $p$), Boolean equations $b \equiv c$ (tests $b, c$), and so on. These examples attest to the generality and wide applicability of our technique.

**Assumption 2.** We collect a list of assumptions for $(P, Id, R, R_\bot)$. Let $\Sigma = P \cup Id$ be a finite alphabet, whose symbols are partitioned into a set $P$ of *action symbols* and a set $Id$ of *subidentities*. We write $p, q, r, \ldots$ to vary over actions symbols, $\alpha, \beta, \gamma, \ldots$ to vary over subidentities, and $a, b, c, \ldots$ to vary over arbitrary symbols of $\Sigma$. Let $S$ be the subset of $\Sigma^*$ consisting of all strings in which an action symbol $p$ always appears surrounded by subidentities, as in $\alpha p \beta$. The set $S$ is regular, and the corresponding regular expression is $e_S = Id \cdot (Id^* \cdot P \cdot Id)^* \cdot Id^*$. Let $R$ be a rewrite system over $\Sigma$ that includes at least the rules $\alpha\alpha \to \alpha$ for every subidentity $\alpha \in Id$, and additionally it satisfies:

(1) $S$ is closed under $\to_R$: if $x \in S$ and $x \to_R y$ then $y \in S$. Moreover, $S$ is closed under the inverse of $\to_R$: if $y \in S$ and $x \to_R y$ then $x \in S$. (2) For every rule $\ell \to r$ of $R$ we have that $|\ell| > |r|$. (3) $R$ is confluent on $S$: For $u, x, y \in S$, $u \to_R^* x$ and $u \to_R^* y$ imply that $x \to_R^* z$ and $y \to_R^* z$ for some $z \in S$. Now, suppose that $R_\perp$ extends $R$ with the rules $\alpha\beta \to \perp$ for all subidentities $\alpha \neq \beta$, and possibly more rules of the form $z \to \perp$, where $z \in S$ and $|z| \geq 2$. Moreover, $R_\perp$ contains all the rules $a\perp \to \perp$, $\perp a \to \perp$ (for each $a \in \Sigma$), as well as the rule $\perp\perp \to \perp$. We assume that $R_\perp$ satisfies additionally the *seamlessness property*: For $xzy \in S$ with $z \to \perp$ in $R_\perp$, any $R$-successor of $xzy$ is of the form $x'z'y'$ for some rule $z' \to \perp$ of $R_\perp$. We will use the term *irreducible* (unqualified) to mean $R_\perp$-irreducible of $S$. Finally, define the function $\mathsf{cp}$ to send every letter $a$ of $\Sigma$ to a finite subset $\mathsf{cp}(a)$ of $S$, called the *components* of $a$. For a subidentity $\alpha \in Id$, we put $\mathsf{cp}(\alpha) = \{\alpha\}$. For an action symbol $p \in P$, we put $\mathsf{cp}(p) = \{\alpha p \beta \mid \alpha, \beta \in Id\}$.

**Definition 8 (Language Interpretation).** Let $(P, Id, R, R_\perp)$ satisfy Assumption 2. For a string $u$, we put $[u]_S = S \cap [u]$. For a language $L \subseteq S$, we define:

$$\mathscr{G}_{R_\perp}(L) = \{\mathsf{nf}_R(u) \mid u \in L\} \setminus [\perp]_S \qquad \mathscr{C}_{R_\perp}(L) = [\perp]_S \cup \bigcup_{u \in L} [u]_S$$

The *coalesced product* of irreducibles, written $\diamond$, is defined as in Definition 5. The interpretation $\mathcal{G}_{R_\perp}$ sends a regular expression to a set of irreducibles:

$$\begin{array}{ll} \mathcal{G}_{R_\perp}(a) = \mathsf{nf}_R(\mathsf{cp}(a)) \setminus [\perp]_S & \mathcal{G}_{R_\perp}(e_1 + e_2) = \mathcal{G}_{R_\perp}(e_1) \cup \mathcal{G}_{R_\perp}(e_2) \\ \mathcal{G}_{R_\perp}(0) = \emptyset & \mathcal{G}_{R_\perp}(e_1; e_2) = \mathcal{G}_{R_\perp}(e_1) \diamond \mathcal{G}_{R_\perp}(e_2) \\ \mathcal{G}_{R_\perp}(1) = Id & \mathcal{G}_{R_\perp}(e^*) = \bigcup_{n \geq 0} \mathcal{G}_{R_\perp}(e)^{\langle n \rangle} \end{array}$$

Define $\mathcal{C}_{R_\perp}(e) = \mathscr{C}_{R_\perp}(\mathcal{R}(e))$, for expressions $e$ with $\mathcal{R}(e) \subseteq S$.

**Definition 9 (Well-Behaved).** Let $(P, Id, R, R_\perp)$ be a tuple satisfying Assumption 2. We say that the tuple is *well-behaved* if $R_\perp$ consists of finitely many rules, every rule $\ell \to r$ of $R$ satisfies $|r| = 1$ and $|\ell| > 1$, and it satisfies the following property: For every letter $a$ of the alphabet, the $R$-ancestors of $a$ form a regular set $\mathcal{R}(e_a)$ for some regular expression $e_a$, so that $\mathsf{KA}_R \vdash e_a \equiv a$.

We define the finite collection $E$ of *equations associated* with the well-behaved tuple $(P, Id, R, R_\perp)$ to contain: (1) an equation $x \equiv y$ for every rule $x \to y$ of $R$, (2) an equation $z \equiv 0$ for every rule $z \to \perp$ of $R_\perp$, as well as (3) the equation $\sum_{\alpha \in Id} \alpha \equiv 1$. We write $\mathsf{KA}_E$ for the system of KA augmented with the equations $E$. It is easy to prove in $\mathsf{KA}_E$ the equation $\sum_{x \in \mathsf{cp}(a)} x \equiv a$ for every letter $a$.

**Theorem 5 (Completeness).** *Let $(P, Id, R, R_\perp)$ be well-behaved, and $E$ be the associated equations. Then, $\mathcal{G}_{R_\perp}(e_1) = \mathcal{G}_{R_\perp}(e_2)$ implies that $\mathsf{KA}_E \vdash e_1 \equiv e_2$.*

**Applications.** Theorem 5 gives us four completeness results as corollaries. First, we show that KAT is complete for the standard interpretation of KAT terms as sets of guarded strings. We then consider the case of KAT extended with simple Hoare hypotheses $b; p; c \equiv 0$ (tests $b, c$, atomic action $p$), and with hypotheses $c; p \equiv c$. We conclude with a completeness proof for NetKAT.

**Theorem 6.** Let $\mathcal{G}_{\mathsf{KAT}}$ be the standard interpretation of KAT expressions. For any $e_1$ and $e_2$, it holds that $\mathcal{G}_{\mathsf{KAT}}(e_1) = \mathcal{G}_{\mathsf{KAT}}(e_2)$ implies $\mathsf{KAT} \vdash e_1 \equiv e_2$.

A *simple Hoare assertion* is an expression $\{b\}p\{c\}$, where $b, c$ are tests and $p$ is an atomic action. It can be encoded in KAT with the equation $b; p; \neg c \equiv 0$. This equation is equivalent to the conjunction of the equations $\beta; p; \gamma \equiv 0$, where $\beta, \gamma$ are atoms with $\beta \leq b$ and $\gamma \leq \neg c$. So, w.l.o.g. we restrict attention to assertions of the form $\beta; p; \gamma \equiv 0$, where $\beta, \gamma$ are atoms and $p$ is an atomic action.

**Theorem 7.** Let $Z_h$ be a finite collection of strings of the form $\gamma p \delta$, where $\gamma, \delta$ are atoms and $p$ is an atomic action symbol. Let $W$ be the set of strings containing some $\gamma p \delta$ in $Z_h$, and $H$ be the collection of equations $\gamma; p; \delta \equiv 0$ for every $\gamma p \delta$ in $Z_h$. Define the interpretation $\mathcal{G}_h$ by $\mathcal{G}_h(e) = \mathcal{G}_{\mathsf{KAT}}(e) \setminus W$. Then, $\mathcal{G}_h(e_1) = \mathcal{G}_h(e_2)$ implies $\mathsf{KAT} + H \vdash e_1 \equiv e_2$.

We consider now another class of equations of the form $c; p \equiv c$, where $c$ is a test and $p$ is an atomic action. We see that $c; p \equiv c$ is equivalent to the conjunction of $\gamma; p \equiv \gamma$ for $\gamma \leq c$. So, we can restrict our attention to equations of the form $\gamma; p \equiv \gamma$, where $\gamma$ is an atom, and $p$ is an atomic action.

**Theorem 8.** Let $X$ be a finite set of strings of the form $\gamma p$, where $\gamma$ is an atom and $p$ is an atomic action symbol, and $H$ be the set of equations $\gamma; p \equiv \gamma$ for every $\gamma p$ in $X$. For an atomic action symbol $p$, define the set of atoms $A(p) = \{\gamma \mid \gamma p \in X\}$. Let $\mathcal{G}_h$ be the interpretation that differs from $\mathcal{G}_{\mathsf{KAT}}$ only for the base case of atomic action symbols: $\mathcal{G}_h(p) = A(p) \cup \{\gamma p \delta \mid \gamma \notin A(p)\}$. Then, $\mathcal{G}_h(e_1) = \mathcal{G}_h(e_2)$ implies $\mathsf{KAT}{+}H \vdash e_1 \equiv e_2$, for any KAT expressions $e_1$, $e_2$.

We turn to the case of NetKAT. Fix an alphabet $At$ of atoms. For $\alpha \in At$ we introduce an action symbol $p_\alpha$, and we put $P = \{p_\alpha \mid \alpha \in At\}$. Let $\mathsf{dup}$ be a new action symbol, and set $\Sigma = P \cup \{\mathsf{dup}\} \cup At$. $\mathsf{NetKAT}$ extends $\mathsf{KA}$ with:

$$\begin{array}{ccc} \sum_{\alpha \in At} \alpha \equiv 1 & \alpha; \mathsf{dup} \equiv \mathsf{dup}; \alpha & p_\alpha \equiv p_\alpha; \alpha \\ \alpha; \beta \equiv 0 \ (\alpha \neq \beta) & p_\alpha; p_\beta \equiv p_\beta & \alpha \equiv \alpha; p_\alpha \end{array}$$

The axioms imply $\alpha; \alpha \equiv \alpha; p_\alpha; \alpha \equiv \alpha; p_\alpha \equiv \alpha$, for every atom $\alpha$. So, $\mathsf{NetKAT}$ can also be defined as an extension of $\mathsf{KAT}$. The following axioms

$$\begin{array}{cccc} \sum_{\alpha \in At} \alpha \equiv 1 & \alpha; \alpha \equiv \alpha & a; p_\alpha; \alpha \equiv \alpha & \alpha; \mathsf{dup}; \beta \equiv 0 \ (\alpha \neq \beta) \\ & \alpha; \beta \equiv 0 \ (\alpha \neq \beta) & p_\alpha; \alpha; p_\beta \equiv p_\beta & \alpha; p_\beta; \gamma \equiv 0 \ (\beta \neq \gamma) \end{array}$$

give an equivalent axiomatization of $\mathsf{NetKAT}$.

**Theorem 9.** Let $At$ be the subidentities (atoms), and $P' = P \cup \{\mathsf{dup}\}$ be the alphabet of action symbols, where $P = \{p_\alpha \mid \alpha \in At\}$. Define $R$ and $R_\perp$ as:

$$\begin{array}{ccc} \alpha\alpha \to \alpha \ (\alpha \in At) & \alpha p_\alpha \alpha \to \alpha \ (\alpha \in At) & p_\alpha \alpha p_\beta \to p_\beta \ (\alpha, \beta \in At) \\ \alpha\beta \to \bot \ (\alpha \neq \beta) & \alpha \mathsf{dup} \beta \to \bot \ (\alpha \neq \beta) & \alpha p_\beta \gamma \to \bot \ (\beta \neq \gamma) \end{array}$$

$(P', At, R, R_\perp)$ is well-behaved, and $\mathsf{NetKAT}$ is complete for $\mathcal{G}_{R_\perp}$.

## 6 Conclusion

We have identified sufficient conditions for the construction of free language models for systems of Kleene algebra with additional equations. The construction provides a uniform approach to deductive completeness and coalgebraic decision procedures. The criteria are given in terms of *inverse context-free rewrite*

*systems* [16]. They imply the existence of free language models in a wide range of previously studied instances, including KAT [6] and NetKAT [8], as well as some new ones. We have also given a negative result that establishes a limit to the applicability of the technique.

For the future, we would like to investigate the possibility of developing a uniform approach to coalgebraic bisimulation-based decision procedures [8, 12–15]. Such decision procedures typically involve some variant of Brzozowski derivatives and are highly dependent on the existence of language models.

**Acknowledgments.** We thank Bjørn Grathwohl, Stathis Zachos, and the anonymous reviewers for helpful suggestions. This work was supported by the National Security Agency under award #H98230-14-C-0140.

## References

1. Angus, A., Kozen, D.: Kleene algebra with tests and program schematology. Technical Report TR2001-1844, CS Department, Cornell University (July 2001)
2. Barth, A., Kozen, D.: Equational verification of cache blocking in LU decomposition using Kleene algebra with tests. Technical Report TR2002-1865, Computer Science Department, Cornell University (June 2002)
3. Cohen, E.: Hypotheses in Kleene algebra. Technical report, Bellcore (1993)
4. Cohen, E.: Lazy caching in Kleene algebra (1994)
5. Cohen, E.: Using Kleene algebra to reason about concurrency control. Technical report, Telcordia, Morristown, N.J (1994)
6. Kozen, D.: Kleene algebra with tests. Transactions on Programming Languages and Systems 19(3), 427–443 (1997)
7. Kozen, D., Patron, M.C.: Certification of compiler optimizations using Kleene algebra with tests. In: Proc. 1st Int. Conf. Comput. Logic (CL 2000), pp. 568–582 (2000)
8. Anderson, C.J., Foster, N., Guha, A., Jeannin, J.-B., Kozen, D., Schlesinger, C., Walker, D.: NetKAT: Semantic foundations for networks. In: Proceedings of POPL 2014, San Diego, California, USA, pp. 113–126. ACM (January 2014)
9. Kozen, D., Smith, F.: Kleene algebra with tests: Completeness and decidability. In: van Dalen, D., Bezem, M. (eds.) CSL 1996. LNCS, vol. 1258, pp. 244–259. Springer, Heidelberg (1997)
10. Hardin, C., Kozen, D.: On the elimination of hypotheses in Kleene algebra with tests. Technical Report TR2002-1879, CS Department, Cornell University (2002)
11. Kozen, D.: On the complexity of reasoning in Kleene algebra. Information and Computation 179, 152–162 (2002)
12. Foster, N., Kozen, D., Milano, M., Silva, A., Thompson, L.: A coalgebraic decision procedure for NetKAT. Technical Report, Computing and Information Science, Cornell University (2014), `http://hdl.handle.net/1813/36255`
13. Grathwohl, N.B.B., Kozen, D., Mamouras, K.: KAT + B! Technical Report, CIS, Cornell University (January 2014), `http://hdl.handle.net/1813/34898`
14. Rot, J., Bonsangue, M., Rutten, J.: Coalgebraic bisimulation-up-to. In: van Emde Boas, P., Groen, F.C.A., Italiano, G.F., Nawrocki, J., Sack, H. (eds.) SOFSEM 2013. LNCS, vol. 7741, pp. 369–381. Springer, Heidelberg (2013)

15. Bonchi, F., Pous, D.: Checking NFA equivalence with bisimulations up to congruence. In: Proceedings of POPL 2013, pp. 457–468. ACM (2013)
16. Book, R.V., Otto, F.: String-Rewriting Systems. Springer (1993)
17. Kozen, D., Mamouras, K.: Kleene algebra with equations. Technical Report CIS, Cornell University (February 2014), http://hdl.handle.net/1813/36202
18. Kozen, D.: A completeness theorem for Kleene algebras and the algebra of regular events. Infor. and Comput. 110(2), 366–390 (1994)
19. Baader, F., Nipkow, T.: Term Rewriting and All That. CUP (1998)