The Complexity of Kleene Algebra with Tests

Ernie Cohen*
ernie@bellcore.com

Dexter Kozen†
kozen@cs.cornell.edu

Frederick Smith†
fms@cs.cornell.edu

July 19, 1996

Abstract

Kleene algebras with tests provide a natural framework for equational specification and verification. Kleene algebras with tests and related systems have been used successfully in basic safety analysis, source-to-source program transformation, and concurrency control. The equational theory of Kleene algebras with tests has been shown to be decidable in at most exponential time by an efficient reduction to Propositional Dynamic Logic. In this paper we prove that the theory is *PSPACE*-complete.

1 Introduction

Kleene algebra with tests (KAT) [15] is an algebraic system intermediate to Kleene algebra (KA) and Propositional Dynamic Logic (PDL) in expressive power. One can use KAT for a range of common verification tasks without resorting to the full power of PDL. KAT and related systems have been applied successfully to real problems in basic safety analysis, source-to-source

[1]Bell Communications Research Inc., 445 South St., Morristown, NJ 07960, USA
[2]Computer Science Department, Cornell University, Ithaca, NY 14853-7501, USA

program transformation, and concurrency control [3, 4, 5, 15], including a rigorous proof of a popular lazy caching protocol [4]. This paper is concerned with the extent to which this system can be automated.

Kleene algebra dates to a 1956 paper of Kleene [10] and was further developed in the 1971 monograph of Conway [6]. Kleene algebra has appeared in one form or another in relational algebra [19, 22], semantics and logics of programs [11, 20], automata and formal language theory [17, 18], and the design and analysis of algorithms [1, 9, 13]. Many authors have contributed over the years to the development of the algebraic theory; see [15] and references therein.

Propositional Dynamic Logic (PDL) [7] is a logical system that blends Kleene algebra with modal logic. Syntactically, PDL is a two-sorted logic consisting of *programs* and *propositions* defined by mutual induction. A basic operator in PDL is the *test operator* ?, by which a program $\varphi?$ can be formed from any proposition φ. Intuitively, $\varphi?$ acts as a guard that succeeds with no side effects in states satisfying φ and fails or aborts in states not satisfying φ. Tests are used to manipulate flow of control and are needed to model conventional programming constructs such as conditionals and **while** loops.

From a practical standpoint, many simple program manipulations such as loop unwinding and basic safety analysis do not require the full power of PDL, but can be carried out in a purely equational subsystem using the axioms of Kleene algebra. However, tests are an essential ingredient for modeling real programs, which motivates their inclusion in the system KAT.

It has been shown that Kleene algebra with extra conditions of the form $p = 0$ reduces efficiently to Kleene algebra without extra conditions, therefore remains decidable [3]; but that *-continuous Kleene algebra in the presence of extra commutativity conditions of the form $pq = qp$, even for primitive p and q, is undecidable [2]. In [15], it was shown how this undecidability proof can be used to establish that the universal Horn theory of *-continuous Kleene algebras is not finitely axiomatizable.

In [16] it was shown that the equational theories of Kleene algebras with tests and *-continuous Kleene algebras with tests coincide, and that these theories are complete over certain language-theoretic and relational models. The language-theoretic models involved regular sets of *guarded strings* over finite alphabets Σ and B of *actions* and *tests*, respectively. These sets play the same role in Kleene algebra with tests that the regular sets play in Kleene algebra. The completeness theorem of [16] is analogous to the completeness theorem of [14] in which the regular sets over Σ were shown to form the free Kleene algebra on generators Σ.

In [16] the proof of correctness of the reduction of [3] from Kleene algebra with conditions $p = 0$ to Kleene algebra without conditions was simplified and extended to handle Kleene algebras with tests. Since the extra commutativity conditions needed in [15] were all of this form (if b is a test, the commutativity condition $pb = bp$ is equivalent to the condition $bp\bar{b} + \bar{b}pb = 0$), the system used in [15] reduces efficiently to KAT without extra conditions, and is thus no more difficult to decide than KAT. The complexity of KAT is therefore of considerable practical interest.

It was shown in [16] that KAT is decidable in at most exponential time by an efficient reduction to PDL. Since KA is known to be *PSPACE*-complete [21], KAT is at least *PSPACE*-hard. It was conjectured in [15] that KAT is no more difficult to decide than KA.

In this paper we verify that conjecture. We give a new decidability proof that establishes that KAT is in *PSPACE*, therefore *PSPACE*-complete. In contrast, PDL is complete for exponential time [7], which indicates that some savings can be achieved by using KAT in applications where PDL would previously have been used.

The algorithm makes use of the free language-theoretic model involving sets of guarded strings introduced in [16] and matrices over Kleene algebras with tests.

2 Kleene Algebra with Tests

A *Kleene algebra with tests* [15] is a Kleene algebra with an embedded Boolean subalgebra. Formally, it is a two-sorted structure

$$(\mathcal{K},\ \mathcal{B},\ +,\ \cdot,\ ^*,\ ^-,\ 0,\ 1)$$

where $^-$ is a unary operator defined only on $\mathcal{B}$, such that

- $\mathcal{B} \subseteq \mathcal{K}$,
- $(\mathcal{K},\ +,\ \cdot,\ ^*,\ 0,\ 1)$ is a Kleene algebra, and
- $(\mathcal{B},\ +,\ \cdot,\ ^-,\ 0,\ 1)$ is a Boolean algebra.

The elements of $\mathcal{B}$ are called *tests*. We reserve the letters p, q, r, s for arbitrary elements of $\mathcal{K}$ and a, b, c for tests. In PDL, a test would be written $b?$, but since we are using different symbols for tests we can omit the ?.

This deceptively simple definition actually carries a lot of information. The operators $+$, $\cdot$, 0, and 1 play two roles: when applied to arbitrary elements of $\mathcal{K}$, they refer to choice, composition, fail, and skip, respectively; and when applied to tests, they take on the additional meaning of join, meet, falsity, and truth, respectively. The coexistence of these two usages admits considerable economy of expression.

As is customary, we will omit $\cdot$, writing pq instead of $p \cdot q$. The precedence of the operators is $^{-} > ^{*} > \cdot > +$. Thus $p + qr^*$ should be parsed $p + q(r^*)$.

2.1 Kleene Algebra

There have been many competing axiomatizations of Kleene algebra. The formulation we adopt here (KA) is from [14]. Succinctly put, a *Kleene algebra* is an (additively) idempotent, (multiplicatively) commutative semiring under $+, \cdot, 0, 1$, satisfying the additional properties

$$\begin{aligned}
1 + pp^* &= p^* && (1)\\
1 + p^*p &= p^* && (2)\\
q + pr \leq r &\rightarrow p^*q \leq r && (3)\\
q + rp \leq r &\rightarrow qp^* \leq r && (4)
\end{aligned}$$

where $\leq$ refers to the natural partial order on $\mathcal{K}$:

$$p \leq q \quad \leftrightarrow \quad p + q = q \ .$$

The operation $+$ gives the supremum with respect to the natural order $\leq$. Instead of (3) and (4), we might take the equivalent axioms

$$\begin{aligned}
pr \leq r &\rightarrow p^*r \leq r && (5)\\
rp \leq r &\rightarrow rp^* \leq r \ . && (6)
\end{aligned}$$

These axioms say essentially that * behaves like the Kleene asterate operator of formal language theory or the reflexive transitive closure operator of relational algebra.

Typical 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.

Some useful identities of Kleene algebra are

$$(p^*q)^*p^* = (p+q)^* \tag{7}$$

$$p(qp)^* = (pq)^*p \tag{8}$$

$$p^* = (pp)^*(1+p)\ . \tag{9}$$

All the operators are monotone with respect to $\leq$. In other words, if $p \leq q$, then $pr \leq qr$, $p+r \leq q+r$, and $p^* \leq q^*$ for any r.

A Kleene algebra is said to be **-continuous* if it satisfies the infinitary condition

$$pq^*r = \sup_{n \geq 0} pq^n r \tag{10}$$

where

$$q^0 \stackrel{\text{def}}{=} 1$$

$$q^{n+1} \stackrel{\text{def}}{=} qq^n$$

and the supremum is with respect to the natural order $\leq$. We can think of (10) as a conjunction of the infinitely many equational axioms

$$pq^n r \leq pq^*r, \quad n \geq 0 \tag{11}$$

and the infinitary Horn formula

$$\bigwedge_{n \geq 0} pq^n r \leq s \ \rightarrow \ pq^*r \leq s\ . \tag{12}$$

In the presence of the other axioms, the *-continuity condition (10) implies (3–6), and is strictly stronger in the sense that there exist Kleene algebras that are not *-continuous [12].

The main result of [14] is that all true identities between regular expressions, interpreted as regular sets of strings, are derivable from the axioms of Kleene algebra [14], and only such identities are derivable. In other words, the algebra of regular sets of strings over the finite alphabet Σ is the free Kleene algebra on generators Σ. It is also the free *-continuous Kleene algebra on generators Σ; *i.e.*, the equational theory of the Kleene algebras and the *-continuous Kleene algebras coincide.

See [14] for a more thorough introduction.

2.2 The Boolean Subalgebra

The Boolean subalgebra $\mathcal{B}$ admits a Boolean negation operator $^{-}$. The sequential composition operator $\cdot$ acts as conjunction when applied to elements of $\mathcal{B}$, and the choice operator $+$ acts as disjunction. Intuitively, a test bc succeeds iff both b and c succeed, and $b + c$ succeeds iff either b or c succeeds.

Since $b \leq 1$ for all $b \in \mathcal{B}$, it is tempting to define tests in an arbitrary Kleene algebra $\mathcal{K}$ to be the set $\{p \in \mathcal{K} \mid p \leq 1\}$. Although this makes sense in algebras of binary relations, in general the set $\{p \in \mathcal{K} \mid p \leq 1\}$ may not extend to a Boolean algebra. For example, $p \leq 1$ for all p in the (min,+) Kleene algebra of the theory of algorithms (see [13]), but the multiplicative idempotence law $pp = p$ fails.

Even over algebras of binary relations, we would like to admit models with programs whose input/output relations are subsets of the identity (*i.e.*, have no side effects) but whose complements are nevertheless uncomputable. We intend tests b to be viewed as simple predicates that are easily recognizable as such, and that are immediately decidable in a given state (and whose complements are therefore also immediately decidable).

We remark that under the present definition, every Kleene algebra extends trivially to a Kleene algebra with tests by taking $\mathcal{B}$ to be the two-element Boolean algebra consisting of 0 and 1.

2.3 The Language of Kleene Algebra with Tests

Let Σ and B be disjoint finite sets of symbols. Elements of Σ are called *primitive actions* and elements of B are called *primitive tests*. *Terms* and *Boolean terms* are defined inductively:

- any primitive action p is a term
- any primitive test b is a Boolean term
- 0 and 1 are Boolean terms
- if p and q are terms, then so are $p + q$, pq, and p^* (suitably parenthesized)
- if b and c are Boolean terms, then so are $b + c$, bc, and $\overline{b}$ (suitably parenthesized)

- any Boolean term is a term.

The set of all terms over Σ and B is denoted $T_{\Sigma,\mathsf{B}}$. The set of all Boolean terms over B is denoted T_{B}.

A term $p \in T_{\Sigma,\mathsf{B}}$ is called a *regular expression* if all occurrences of $^{-}$ are applied to primitive tests only; *i.e.*, if p is a regular expression in the usual sense over the alphabet $\Sigma \cup \mathsf{B} \cup \overline{\mathsf{B}}$, where $\overline{\mathsf{B}} = \{\overline{b} \mid b \in \mathsf{B}\}$. The set of all regular expressions over $\Sigma \cup \mathsf{B} \cup \overline{\mathsf{B}}$ is denoted $R_{\Sigma,\mathsf{B}}$. Any term in $T_{\Sigma,\mathsf{B}}$ can be efficiently converted to an equivalent regular expression in $R_{\Sigma,\mathsf{B}}$ of comparable size using the De Morgan laws and the law $\overline{\overline{b}} = b$ of Boolean algebra.

An *interpretation* over a Kleene algebra with tests $\mathcal{K}$ is a homomorphism (function commuting with the distinguished operations and constants) defined on $T_{\Sigma,\mathsf{B}}$ and taking values in $\mathcal{K}$ such that the Boolean terms are mapped to elements of the distinguished Boolean algebra of $\mathcal{K}$. If $\mathcal{K}$ is a Kleene algebra with tests and I is an interpretation over $\mathcal{K}$, we write $\mathcal{K}, I \vDash \varphi$ if the formula φ holds in $\mathcal{K}$ under the interpretation I according to the usual semantics of first-order logic. In this paper the only formulas we consider are equations or equational implications.

We write $\mathsf{KAT} \vDash \varphi$ if the formula φ is a logical consequence of KAT, *i.e.* if φ holds under all interpretations over Kleene algebras with tests. We write $\mathsf{KAT}^* \vDash \varphi$ if φ holds under all interpretations over *-continuous Kleene algebras with tests.

3 A Language-Theoretic Model

The following language-theoretic model $\mathcal{G}$ and standard interpretation G were introduced in [16]. We repeat the definitions here for completeness.

Let Σ and $\mathsf{B} = \{b_1, \ldots, b_k\}$ be disjoint finite sets of symbols. An *atom* of is a string of literals $c_1 c_2 \cdots c_k$, where each $c_i \in \{b_i, \overline{b}_i\}$. This assumes an arbitrary but fixed order $b_1, b_2, \ldots, b_k$ on B; for technical reasons, we require the literals in an atom to occur in this order. An atom is thus a Boolean expression representing an atom (minimal nonzero element) of the free Boolean algebra on generators B. There are exactly 2^k atoms. We denote atoms of B by $\alpha, \beta, \alpha_0, \ldots$ The set of all atoms of B is denoted $1_{\mathcal{G}}$.

This notation is chosen because $1_{\mathcal{G}}$ will turn out to be the multiplicative identity of our language-theoretic model $\mathcal{G}$.

If $b \in \mathsf{B}$ and α is an atom of B, we write $\alpha \leq b$ if b occurs positively in α and $\alpha \leq \overline{b}$ if b occurs negatively in α. This notation is consistent with the natural order in the free Boolean algebra generated by B.

Intuitively, the symbols of Σ can be thought of as single instructions in a computation and atoms as guards or conditions that must be satisfied for the computation to proceed. If $\alpha \leq c_i$, then α asserts that c_i holds (and $\overline{c}_i$ fails) at that point in the computation.

Definition 1 A *guarded string* over Σ and B is any element of $(1_{\mathcal{G}}\Sigma)^*1_{\mathcal{G}}$, *i.e.*, any string of the form

$$\alpha_0 p_1 \alpha_1 \cdots \alpha_{n-1} p_n \alpha_n \ , \quad n \geq 0 \ ,$$

where each α_i is an atom of B and each $p_i \in \Sigma$. Note that a guarded string begins and ends with an atom. In the case $n = 0$, a guarded string is just a single atom.

The set of all guarded strings over Σ and B is denoted $\mathrm{GS}_{\Sigma,\mathsf{B}}$, or just GS when Σ and B are understood. □

We denote strings in $(\Sigma \cup \mathsf{B} \cup \overline{\mathsf{B}})^*$, including guarded strings, by the letters $x, y, z, x_1, \ldots$.

The analog of concatenation for guarded strings is *coalesced product* ($\diamond$).

Definition 2 The *coalesced product* operation $\diamond$ is a *partial* binary operation on GS defined as follows:

$$x\alpha \diamond \beta y \ \stackrel{\mathrm{def}}{=} \ \begin{cases} x\alpha y \ , & \text{if } \alpha = \beta \\ \text{undefined} \ , & \text{otherwise.} \end{cases}$$

In other words, if the terminal atom of the first string is the same as the initial atom of the second string, then the two strings can be *coalesced*. This is like concatenation, except that the common intermediate atom is only written once.

If $A, B \subseteq \mathrm{GS}$, then

$$A \diamond B \ \stackrel{\mathrm{def}}{=} \ \{x \diamond y \mid x \in A, \ y \in B\} \ .$$

Thus $A \diamond B$ consists of all existing coalesced products of guarded strings in A with guarded strings in B. □

Example 3 Let $\mathcal{B} = \{b, c\}$ and $\Sigma = \{p, q, r, s\}$. Let

$$\begin{aligned} A &= \{bcp\overline{b}\overline{c}, \overline{b}\overline{c}, bcqb\overline{c}\} \\ B &= \{\overline{b}\overline{c}rbc, \overline{b}\overline{c}, b\overline{c}sbc\} . \end{aligned}$$

Then

$$A \diamond B = \{bcp\overline{b}\overline{c}rbc, bcp\overline{b}\overline{c}, \overline{b}\overline{c}rbc, \overline{b}\overline{c}, bcqb\overline{c}sbc\} .$$

□

Note that the identity for coalesced product is $1_{\mathcal{G}}$.

Whereas the operation $\diamond$ is partial when applied to guarded strings, it is total when applied to *sets* of guarded strings. If there are no existing coalesced products of strings from A and B, then $A \diamond B = \varnothing$. It is not difficult to show that $\diamond$ is associative and that coalesced product distributes over union.

For $A \subseteq$ GS, define inductively

$$\begin{aligned} A^0 &\stackrel{\text{def}}{=} 1_{\mathcal{G}} \\ A^{n+1} &\stackrel{\text{def}}{=} A \diamond A^n . \end{aligned}$$

The asterate operation for sets of guarded strings is defined by

$$A^* \stackrel{\text{def}}{=} \bigcup_{n \geq 0} A^n .$$

Let $^-$ denote set complementation in $1_{\mathcal{G}}$. That is, if $A \subseteq 1_{\mathcal{G}}$, then $\overline{A} = 1_{\mathcal{G}} - A$. Consider the structure

$$\mathcal{P}_{\Sigma,\mathsf{B}} = (2^{\mathrm{GS}},\ 2^{1_{\mathcal{G}}},\ \cup,\ \diamond,\ ^*,\ ^-,\ \varnothing,\ 1_{\mathcal{G}}) .$$

We write $\mathcal{P}$ for $\mathcal{P}_{\Sigma,\mathsf{B}}$ when Σ and B are understood. It is quite straightforward to verify that $\mathcal{P}$ is a *-continuous Kleene algebra with tests, *i.e.* is a model of KAT^*. The Boolean algebra axioms hold for $2^{1_{\mathcal{G}}}$ because it is a set-theoretic Boolean algebra; on subsets of $1_{\mathcal{G}}$, $\diamond$ is set intersection.

The *-continuity condition follows immediately from the definition of * and the distributivity of coalesced product over infinite union. Since

$$B^* = \bigcup_{n \geq 0} B^n ,$$

we have that

$$A \diamond B^* \diamond C \;=\; A \diamond (\bigcup_{n \geq 0} B^n) \diamond C \;=\; \bigcup_{n \geq 0} A \diamond B^n \diamond C \; .$$

Both of these expressions denote the set

$$\{x \diamond y \diamond z \mid x \in A,\; z \in C,\; \exists n \; y \in B^n\} \; .$$

3.1 Standard Interpretation

The *standard interpretation* G is defined to be the unique homomorphism $G : T_{\Sigma,\mathsf{B}} \rightarrow \mathcal{P}_{\Sigma,\mathsf{B}}$ whose values on primitive actions and primitive tests is given by

$$\begin{aligned} G(p) &\stackrel{\text{def}}{=} \{\alpha p \beta \mid \alpha, \beta \in 1_{\mathcal{G}}\} \\ G(b) &\stackrel{\text{def}}{=} \{\alpha \in 1_{\mathcal{G}} \mid \alpha \leq b\} \; . \end{aligned}$$

The structure $\mathcal{G} = \mathcal{G}_{\Sigma,\mathsf{B}}$ is defined to be the image of $T_{\Sigma,\mathsf{B}}$ under this map; *i.e.*, the subalgebra of $\mathcal{P}_{\Sigma,\mathsf{B}}$ generated by the elements $G(p)$ and $G(b)$ for $p \in \Sigma$ and $b \in \mathsf{B}$. Elements of $\mathcal{G}$ are called *regular sets*.

The map G thus associates a regular set of guarded strings with each term in $T_{\Sigma,\mathsf{B}}$:

$$\begin{aligned} G(p+q) &= G(p) \cup G(q) \\ G(pq) &= G(p) \diamond G(q) \\ G(1) &= 1_{\mathcal{G}} \\ G(0) &= \varnothing \\ G(\overline{b}) &= 1_{\mathcal{G}} - G(b) \\ G(p^*) &= G(p)^* \; . \end{aligned}$$

Let R denote the standard interpretation of regular expressions as regular sets:

$$\begin{aligned} R(p) &= \{p\} \\ R(p+q) &= R(p) \cup R(q) \\ R(pq) &= \{xy \mid x \in R(p),\; y \in R(q)\} \\ R(1) &= \epsilon \\ R(0) &= \varnothing \\ R(p^*) &= R(p)^* \end{aligned}$$

where xy denotes the concatenation of x and y and where ϵ denotes the empty string.

The following result of [16] is analogous to the corresponding result for Kleene algebra and R as proved in [14].

Theorem 4 ([16])

$$\mathsf{KAT} \models p = q \iff G(p) = G(q) \ .$$

3.2 Relating G and R

For regular expressions $p \in R_{\Sigma,\mathsf{B}}$, the standard interpretation R applies, giving a regular subset $R(p) \subseteq (\Sigma \cup \mathsf{B} \cup \overline{\mathsf{B}})^*$.

By a slight abuse, we can regard strings in $(\Sigma \cup \mathsf{B} \cup \overline{\mathsf{B}})^*$ as regular expressions in $R_{\Sigma,\mathsf{B}}$, interpreting the null string ϵ as the term 1. As such, we can apply G or R to them. Note that if x is a string, then $R(x) = \{x\}$, and if x is a guarded string, then $G(x) = \{x\}$.

For $A \subseteq T_{\Sigma,\mathsf{B}}$, define

$$G(A) \stackrel{\text{def}}{=} \bigcup_{t \in A} G(t) \ . \tag{13}$$

It follows immediately that if A is a set of guarded strings, then

$$G(A) = A \ . \tag{14}$$

Lemma 5 *If $p \in R_{\Sigma,\mathsf{B}}$, then $G(p) = G(R(p))$.*

Proof. The proof proceeds by a straightforward structural induction.

For the basis, if $p \in \Sigma \cup \mathsf{B} \cup \overline{\mathsf{B}}$, then

$$\begin{aligned} R(p) &= \{p\} \\ G(R(p)) &= \bigcup_{t \in R(p)} G(t) \\ &= G(p) \ . \end{aligned}$$

This also takes care of the case of $^-$, so we do not need to treat this operator separately. Also,

$$
\begin{aligned}
R(1) &= \{\epsilon\} \\
G(R(1)) &= \bigcup_{t \in R(1)} G(t) \\
&= G(1) \\
R(0) &= \varnothing \\
G(R(0)) &= \bigcup_{t \in R(0)} G(t) \\
&= \varnothing \\
&= G(0) \ .
\end{aligned}
$$

For the induction step, we have

$$
\begin{aligned}
G(R(p+q)) &= G(R(p) \cup R(q)) \\
&= G(R(p)) \cup G(R(q)) \\
&= G(p) \cup G(q) && \text{(induction hypothesis)} \\
&= G(p+q) \ ;
\end{aligned}
$$

$$
\begin{aligned}
G(R(pq)) &= \bigcup_{t \in R(pq)} G(t) \\
&= \bigcup_{\substack{u \,\in\, R(p) \\ v \,\in\, R(q)}} G(uv) \\
&= \bigcup_{\substack{u \,\in\, R(p) \\ v \,\in\, R(q)}} G(u) \diamond G(v) \\
&= (\bigcup_{u \in R(p)} G(u)) \diamond (\bigcup_{v \in R(q)} G(v)) && \text{(distributivity)} \\
&= G(R(p)) \diamond G(R(q)) \\
&= G(p) \diamond G(q) && \text{(induction hypothesis)} \\
&= G(pq) \ ;
\end{aligned}
$$

$$
\begin{aligned}
G(R(p^*)) &= G(\bigcup_{n \geq 0} R(p^n)) \\
&= \bigcup_{n \geq 0} G(R(p^n))
\end{aligned}
$$

$$
\begin{aligned}
&= \bigcup_{n\geq 0} G(p^n) && \text{(induction hypothesis)}\\
&= G(p^*)\ .
\end{aligned}
$$

□

4 Matrix Algebras

Let $\mathcal{K}$ be a Kleene algebra with tests $\mathcal{B}$. The family $\mathcal{M}(n,\mathcal{K})$ of $n \times n$ matrices over $\mathcal{K}$ again forms a KAT. The operations $+$ and $\cdot$ are the usual operations of matrix addition and multiplication, respectively, 0_n is the $n \times n$ zero matrix, and I_n the $n \times n$ identity matrix. The operation * on matrices is defined inductively:

$$
\begin{aligned}
&\left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right]^* \\
&\quad = \left[\begin{array}{c|c} (A+BD^*C)^* & (A+BD^*C)^*BD^* \\ \hline D^*C(A+BD^*C)^* & D^*+D^*C(A+BD^*C)^*BD^* \end{array}\right] . \qquad (15)
\end{aligned}
$$

The distinguished Boolean subalgebra is the set $\Delta(n,\mathcal{B})$ of $n \times n$ diagonal matrices with entries from the distinguished Boolean algebra $\mathcal{B}$. The operation $^-$ on $\Delta(n,\mathcal{B})$ just complements the diagonal elements, leaving the off-diagonal elements 0.

Theorem 6 *The structure*

$$(\mathcal{M}(n,\mathcal{K}),\ \Delta(n,\mathcal{B}),\ +,\ \cdot,\ ^*,\ ^-,\ 0_n,\ I_n)$$

is a Kleene algebra with tests.

Proof. As shown in [14], the structure

$$(\mathcal{M}(n,\mathcal{K}),\ +,\ \cdot,\ ^*,\ 0_n,\ I_n)$$

forms a Kleene algebra, and it is easy to verify that $\Delta(n,\mathcal{B})$ forms a Boolean algebra. □

Definition 7 Let $\mathcal{K}$, $\mathcal{K}'$ be Kleene algebras with tests $\mathcal{B}$, $\mathcal{B}'$, respectively. A KAT-*homomorphism* is a Kleene algebra homomorphism $h : \mathcal{K} \to \mathcal{K}'$ whose restriction to $\mathcal{B}$ is a Boolean algebra homomorphism $h : \mathcal{B} \to \mathcal{B}'$. $\square$

Lemma 8 *Let $h : \mathcal{K} \to \mathcal{K}'$ be a* KAT*-homomorphism, and let $H : \mathcal{M}(n, \mathcal{K}) \to \mathcal{M}(n, \mathcal{K}')$ be its componentwise extension to matrices. Then H is a* KAT*-homomorphism.*

Proof. By definition, $H(E)_{ij} = h(E_{ij})$. It is immediate that $H(E + F) = H(E) + H(F)$, $H(EF) = H(E)H(F)$, $H(0) = 0$, and $H(I) = I$. For *, we can use the inductive definition (15) to give a straightforward inductive proof that $H(E^*) = H(E)^*$. Finally, for $E \in \Delta(n, \mathcal{B})$,

$$\begin{aligned}
H(\overline{E})_{ij} &= h(\overline{E}_{ij}) \\
&= \begin{cases} h(\overline{E_{ij}}) \ , & \text{if } i = j, \\ h(0) \ , & \text{if } i \neq j \end{cases} \\
&= \begin{cases} \overline{h(E_{ij})} \ , & \text{if } i = j, \\ 0 \ , & \text{if } i \neq j \end{cases} \\
&= \begin{cases} \overline{H(E)_{ij}} \ , & \text{if } i = j, \\ 0 \ , & \text{if } i \neq j \end{cases} \\
&= \overline{H(E)}_{ij} \ ,
\end{aligned}$$

so $H(\overline{E}) = \overline{H(E)}$. $\square$

A Kleene algebra or Kleene algebra with tests is called *finitary* if for all $a \in \mathcal{K}$ there exists an $m \geq 0$ such that $a^* = (1 + a)^m$. Any finite algebra is finitary, and any finitary algebra is *-continuous.

Lemma 9 *If $\mathcal{K}$ is finitary, then so is $\mathcal{M}(n, \mathcal{K})$.*

Proof. We proceed by induction on n. For the basis, the algebras $\mathcal{K}$ and $\mathcal{M}(1, \mathcal{K})$ are isomorphic, so there is nothing to prove. Now suppose $n \geq 2$. Break up $E \in \mathcal{M}(n, \mathcal{K})$ arbitrarily into submatrices

$$E = \left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right]$$

where A and D are square. Using (7),

$$\begin{aligned}
\left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right]^* &= \left(\left[\begin{array}{c|c} A & 0 \\ \hline 0 & D \end{array}\right] + \left[\begin{array}{c|c} 0 & B \\ \hline C & 0 \end{array}\right]\right)^* \\
&= \left(\left[\begin{array}{c|c} A & 0 \\ \hline 0 & D \end{array}\right]^* \left[\begin{array}{c|c} 0 & B \\ \hline C & 0 \end{array}\right]\right)^* \left[\begin{array}{c|c} A & 0 \\ \hline 0 & D \end{array}\right]^* \\
&= \left(\left[\begin{array}{c|c} A^* & 0 \\ \hline 0 & D^* \end{array}\right] \left[\begin{array}{c|c} 0 & B \\ \hline C & 0 \end{array}\right]\right)^* \left[\begin{array}{c|c} A^* & 0 \\ \hline 0 & D^* \end{array}\right] \\
&= \left[\begin{array}{c|c} 0 & A^*B \\ \hline D^*C & 0 \end{array}\right]^* \left[\begin{array}{c|c} A^* & 0 \\ \hline 0 & D^* \end{array}\right] .
\end{aligned}$$

By the induction hypothesis, there exists an $m \geq 0$ such that

$$\begin{aligned}
\left[\begin{array}{c|c} A^* & 0 \\ \hline 0 & D^* \end{array}\right] &= \left[\begin{array}{c|c} (I+A)^m & 0 \\ \hline 0 & (I+D)^m \end{array}\right] \\
&= \left[\begin{array}{c|c} I+A & 0 \\ \hline 0 & I+D \end{array}\right]^m \\
&\leq (I+E)^m .
\end{aligned}$$

Also, using (9),

$$\begin{aligned}
\left[\begin{array}{c|c} 0 & A^*B \\ \hline D^*C & 0 \end{array}\right]^* &= \left[\begin{array}{c|c} A^*BD^*C & 0 \\ \hline 0 & D^*CA^*B \end{array}\right]^* \left[\begin{array}{c|c} I & A^*B \\ \hline D^*C & I \end{array}\right] \\
&= \left[\begin{array}{c|c} (A^*BD^*C)^* & 0 \\ \hline 0 & (D^*CA^*B)^* \end{array}\right] \left[\begin{array}{c|c} I & A^*B \\ \hline D^*C & I \end{array}\right] .
\end{aligned}$$

By the induction hypothesis, there exists a $k \geq 0$ such that

$$\begin{aligned}
&\left[\begin{array}{c|c} (A^*BD^*C)^* & 0 \\ \hline 0 & (D^*CA^*B)^* \end{array}\right] \\
&= \left[\begin{array}{c|c} (I+A^*BD^*C)^k & 0 \\ \hline 0 & (I+D^*CA^*B)^k \end{array}\right] \\
&= \left[\begin{array}{c|c} I+A^*BD^*C & 0 \\ \hline 0 & I+D^*CA^*B \end{array}\right]^k
\end{aligned}$$

$$
\begin{aligned}
&= \left(I + \left[\begin{array}{c|c} A^* & 0 \\ \hline 0 & D^* \end{array}\right]\left[\begin{array}{c|c} 0 & B \\ \hline C & 0 \end{array}\right]\left[\begin{array}{c|c} A^* & 0 \\ \hline 0 & D^* \end{array}\right]\left[\begin{array}{c|c} 0 & B \\ \hline C & 0 \end{array}\right]\right)^k \\
&\leq (I + (I+E)^m E (I+E)^m E)^k \\
&\leq (I+E)^{2k(m+1)} .
\end{aligned}
$$

Similarly,

$$
\begin{aligned}
\left[\begin{array}{c|c} I & A^*B \\ \hline D^*C & I \end{array}\right] &= I + \left[\begin{array}{c|c} A^* & 0 \\ \hline 0 & D^* \end{array}\right]\left[\begin{array}{c|c} 0 & B \\ \hline C & 0 \end{array}\right] \\
&\leq I + (I+E)^m E \\
&\leq (I+E)^{m+1} .
\end{aligned}
$$

Putting these all together, we have

$$
E^* \leq (I+E)^{2km+2k+2m+1} .
$$

□

4.1 Matrices over a Boolean Algebra

In this section we establish some special properties of matrices over a Boolean algebra that will prove useful in the subsequent development.

If $\mathcal{B}$ is the distinguished Boolean algebra of a Kleene algebra with tests $\mathcal{K}$, then the algebra $\mathcal{M}(n, \mathcal{B})$ is a subalgebra of $\mathcal{M}(n, \mathcal{K})$. (Note that it is not the distinguished Boolean algebra of $\mathcal{M}(n, \mathcal{K})$; in fact, it is not even a Boolean algebra in general). The algebra $\Delta(n, \mathcal{B})$ of diagonal matrices over $\mathcal{B}$ is the distinguished Boolean algebra of both $\mathcal{M}(n, \mathcal{K})$ and $\mathcal{M}(n, \mathcal{B})$.

Since $b^* = 1$ for any $b \in \mathcal{B}$, it follows immediately from Lemma 9 that $\mathcal{M}(n, \mathcal{B})$ is finitary. In fact, it can be established by combinatorial means that if $A \in \mathcal{M}(n, \mathcal{B})$, then $A^* = (I+A)^{n-1}$, but we will not need this tighter bound.

Let $\mathcal{B}$ denote the free Boolean algebra on generators B. Given a matrix $J \in \mathcal{M}(n, \mathcal{B})$ and an atom α, let J_α be the 0-1 matrix

$$
(J_\alpha)_{ij} = \begin{cases} 1, & \text{if } \alpha \leq J_{ij} \\ 0, & \text{otherwise.} \end{cases}
$$

Lemma 10

$$\alpha \leq (J^*)_{ij} \iff (J_\alpha^*)_{ij} = 1 \ .$$

In particular, one can determine whether $\alpha \leq (J^*)_{ij}$ *in linear time.*

Proof. The first statement is a direct application of Lemma 8, using the Boolean homomorphism $h_\alpha : \mathcal{B} \to \{0,1\}$ defined by

$$h_\alpha(b) \quad = \quad \begin{cases} 1 \ , & \text{if } \alpha \leq b \\ 0 \ , & \text{otherwise.} \end{cases}$$

Then $J_\alpha = H_\alpha(J)$, where H_α is the componentwise extension of h_α to matrices. The condition to be proved is equivalent to the statement $H_\alpha(J^*) = H_\alpha(J)^*$.

The entries of J_α can be determined by testing whether $\alpha \leq b$, which essentially amounts to evaluating a Boolean expression on a given truth assignment. The matrix J_α is a 0-1 matrix, so $(J_\alpha^*)_{ij}$ can be determined in linear time by depth first search on the corresponding directed graph. The entire matrix J_α^* can be computed efficiently using any standard transitive closure algorithm. □

4.2 Matrix Representation of Terms

We eventually want to give an algorithm for deciding whether $\mathsf{KAT} \models p = q$. By Theorem 4, it suffices to decide whether $G(p) = G(q)$.

One possible approach, exploited in [16], is to construct from $p \in T_{\Sigma,\mathsf{B}}$ a regular expression $\widehat{p} \in R_{\Sigma,\mathsf{B}}$ such that

$$G(p) \quad = \quad R(\widehat{p}) \ . \tag{16}$$

Then deciding whether $G(p) = G(q)$ reduces to deciding whether $R(\widehat{p}) = R(\widehat{q})$, which we know how to do in *PSPACE*.

The construction of $\widehat{p}$ from p as given in [16] involves an exponential blowup, which the following example shows to be unavoidable. Suppose $k = 2m$. Consider the expression

$$p \quad = \quad (b_1 b_{m+1} + \overline{b}_1 \overline{b}_{m+1})(b_2 b_{m+2} + \overline{b}_2 \overline{b}_{m+2}) \cdots (b_m b_k + \overline{b}_m \overline{b}_k)$$

This expression represents the set of atoms in which the i^{th} and $m+i^{\text{th}}$ literal have the same parity. Any nondeterministic finite automaton accepting $G(p)$ must store in its state the first half of the string so that it can verify that the second half is correct. Therefore the automaton must have at least 2^m states. Since the translation between regular expressions and nondeterministic automata is linear, any regular expression $\widehat{p}$ such that $R(\widehat{p}) = G(p)$ must be exponentially longer than p.

To circumvent this exponential blowup, we work with a matrix representation of expressions. We first produce a matrix $P \in \mathcal{M}(n, \mathcal{F})$ with small entries and 0-1 vectors u, v of length n such that

$$R(p) \;=\; R(u^T P^* v) \;, \tag{17}$$

where n is approximately the size of p and $\mathcal{F}$ is the free Kleene algebra with tests on generators Σ and B. The construction of P is by induction on the structure of p, and corresponds to the combinatorial construction of an automaton from a regular expression as found for example in [8]. The matrix P is the transition matrix of the automaton equivalent to the regular expression p over the input alphabet $\Sigma \cup \mathsf{B} \cup \overline{\mathsf{B}}$. The vectors u and v specify the start and final states of the automaton, respectively. The elements of P are 0, 1, and sums of primitive symbols. This construction is given in its entirety in [14], so we do not repeat it here.

Since the entries of P are sums of primitive symbols, we can write $P = J + A$, where the entries of J are sums of elements of $\mathsf{B} \cup \overline{\mathsf{B}}$ and the entries of A are sums of elements of Σ. Using (7), we can then write

$$P^* \;=\; (J^*A)^*J^* \;.$$

This form is particularly well suited to the treatment of guarded strings $\alpha_0 p_1 \alpha_1 \cdots \alpha_{m-1} p_m \alpha_m$, the guards α_i being handled by J^* and the symbols p_i by A.

We extend the definition of J_α above to general matrices. For $p \in \Sigma$, define the 0-1 matrix

$$(A_p)_{ij} \;=\; \begin{cases} 1\,, & \text{if } p \le A_{ij} \\ 0\,, & \text{otherwise.} \end{cases}$$

Lemma 11 *$\alpha_0 p_1 \alpha_1 \cdots \alpha_{m-1} p_m \alpha_m \in G(u^T(J^*A)^*J^*v)$ if and only if*

$$u^T J^*_{\alpha_0} A_{p_1} J^*_{\alpha_1} \cdots J^*_{\alpha_{m-1}} A_{p_m} J^*_{\alpha_m} v \;=\; 1 \;.$$

Proof. Because of the restricted form of the entries in J^* and A, all guarded strings in $G(u^T(J^*A)^kJ^*v)$ are of the form $\alpha_0 p_1 \alpha_1 \cdots \alpha_{k-1} p_k \alpha_k$. Since

$$G(u^T(J^*A)^*J^*v) \;=\; \bigcup_{k\geq 0} G(u^T(J^*A)^kJ^*v) \;,$$

we have that

$$\begin{gathered} \alpha_0 p_1 \alpha_1 \cdots \alpha_{m-1} p_m \alpha_m \in G(u^T(J^*A)^*J^*v) \\ \Longleftrightarrow \\ \alpha_0 p_1 \alpha_1 \cdots \alpha_{m-1} p_m \alpha_m \in G(u^T(J^*A)^mJ^*v) \;. \end{gathered}$$

Furthermore, by the definition of matrix multiplication, this occurs iff there exist s_0, t_0, s_1, t_1, $\ldots$, s_m, t_m such that

- $u_{s_0} = 1$
- $\alpha_i \leq (J^*)_{s_it_i}$, $0 \leq i \leq m$
- $p_i \leq A_{t_{i-1}s_i}$, $1 \leq i \leq m$
- $v_{t_m} = 1$.

By Lemma 10 and the definition of A_{p_i}, this occurs iff there exist s_0, t_0, s_1, t_1, $\ldots$, s_m, t_m such that

- $u_{s_0} = 1$
- $(J^*_{\alpha_i})_{s_it_i} = 1$, $0 \leq i \leq m$
- $(A_{p_i})_{t_{i-1}s_i} = 1$, $1 \leq i \leq m$
- $v_{t_m} = 1$.

By the definition of Boolean matrix multiplication, this occurs iff

$$u^T J^*_{\alpha_0} A_{p_1} J^*_{\alpha_1} \cdots J^*_{\alpha_{m-1}} A_{p_m} J^*_{\alpha_m} v \;=\; 1 \;.$$

□

5 A *PSPACE* Algorithm

In this section we give a *PSPACE* algorithm for deciding whether $\mathsf{KAT} \models p \leq q$, or equivalently by Theorem 4, whether $G(p) \subseteq G(q)$.

The algorithm will nondeterministically guess a guarded string

$$\alpha_0 p_1 \alpha_1 \cdots \alpha_{m-1} p_m \alpha_m \quad \in \quad G(p) - G(q) \; .$$

We first produce the matrices u, P, v and y, Q, z such that

$$\begin{aligned} R(p) &= R(u^T P^* v) \\ R(q) &= R(y^T Q^* z) \end{aligned}$$

as shown in §4.2. By Lemma 5, we also have

$$\begin{aligned} G(p) &= G(u^T P^* v) \\ G(q) &= G(y^T Q^* z) \; . \end{aligned}$$

Writing $P = J + A$ and $Q = K + B$ where the entries of J and K are sums of elements of $\mathsf{B} \cup \overline{\mathsf{B}}$ and the entries of A and B are sums of elements of Σ, we have

$$\begin{aligned} G(p) &= G(u^T (J^* A) J^* v) \\ G(q) &= G(y^T (K^* B) K^* z) \; . \end{aligned}$$

By Lemma 11, it suffices to guess $\alpha_0 p_1 \alpha_1 \cdots \alpha_{m-1} p_m \alpha_m$ such that

$$\begin{aligned} u^T J^*_{\alpha_0} A_{p_1} J^*_{\alpha_1} \cdots J^*_{\alpha_{m-1}} A_{p_m} J^*_{\alpha_m} v &= 1 \quad \text{and} \\ y^T K^*_{\alpha_0} B_{p_1} K^*_{\alpha_1} \cdots K^*_{\alpha_{m-1}} B_{p_m} K^*_{\alpha_m} z &= 0 \; . \end{aligned}$$

Let $u_0 = u$ and $y_0 = y$. We guess α_0, p_1, α_1, p_2, α_2, $\ldots$ in that order. After guessing α_i, $i \geq 0$, we calculate J_{α_i} and K_{α_i} and their reflexive transitive closures $J^*_{\alpha_i}$ and $K^*_{\alpha_i}$, then calculate the 0-1 column vectors w_i and x_i such that

$$\begin{aligned} w_i^T &= u_i^T J^*_{\alpha_i} \\ x_i^T &= y_i^T K^*_{\alpha_i} \; . \end{aligned}$$

After guessing p_i, $i \geq 1$, we calculate A_{p_i} and B_{p_i}, then calculate the 0-1 column vectors u_i and y_i such that

$$\begin{aligned} u_i^T &= w_{i-1}^T A_{p_i} \\ y_i^T &= x_{i-1}^T B_{p_i} \ . \end{aligned}$$

It follows inductively that

$$\begin{aligned} w_i^T &= u_0^T J_{\alpha_0}^* A_{p_1} J_{\alpha_1}^* \cdots J_{\alpha_{i-1}}^* A_{p_i} J_{\alpha_i}^* \\ x_i^T &= y_0^T K_{\alpha_0}^* B_{p_1} K_{\alpha_1}^* \cdots K_{\alpha_{i-1}}^* B_{p_i} K_{\alpha_i}^* \ . \end{aligned}$$

We halt and accept if at any point $w_i^T v = 1$ and $x_i^T z = 0$.

The correctness of this algorithm follows from Lemma 11. It uses at most polynomial space, since in each stage of the computation only the vectors w_i and x_i need be remembered.

The algorithm can be made deterministic using Savitch's Theorem (see [8]). The problem is known to be *PSPACE*-hard [21]. We have thus shown

Theorem 12 *The equational theory of* KAT *is PSPACE-complete.*

Acknowledgements

The support of the National Science Foundation under grant CCR-9317320 is gratefully acknowledged. The third author is supported on a National Science Foundation Graduate Fellowship.

References

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, 1975.

[2] E. Cohen, February 1994. Personal communication.

[3] ——, *Hypotheses in Kleene algebra.* `ftp://ftp.bellcore.com/pub/ernie/research/homepage.html`, April 1994.

[4] ——, *Lazy caching.* `ftp://ftp.bellcore.com/pub/ernie/research/homepage.html`, 1994.

[5] ——, *Using Kleene algebra to reason about concurrency control.* `ftp://ftp.bellcore.com/pub/ernie/research/homepage.html`, 1994.

[6] J. H. Conway, *Regular Algebra and Finite Machines*, Chapman and Hall, 1971.

[7] M. J. Fischer and R. E. Ladner, *Propositional dynamic logic of regular programs*, J. Comput. Syst. Sci., 18 (1979), pp. 194–211.

[8] J. E. Hopcroft and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, 1979.

[9] K. Iwano and K. Steiglitz, *A semiring on convex polygons and zero-sum cycle problems*, SIAM J. Comput., 19 (1990), pp. 883–901.

[10] S. C. Kleene, *Representation of events in nerve nets and finite automata*, in Automata Studies, Shannon and McCarthy, eds., Princeton University Press, 1956, pp. 3–41.

[11] D. Kozen, *On induction vs. *-continuity*, in Proc. Workshop on Logic of Programs, Kozen, ed., vol. 131 of Lect. Notes in Comput. Sci., Springer, 1981, pp. 167–176.

[12] ——, *On Kleene algebras and closed semirings*, in Proc. Math. Found. Comput. Sci., Rovan, ed., vol. 452 of Lect. Notes in Comput. Sci., Springer, 1990, pp. 26–47.

[13] ——, *The Design and Analysis of Algorithms*, Springer-Verlag, 1991. ISBN 0-387-97687-6. 322 pages.

[14] ——, *A completeness theorem for Kleene algebras and the algebra of regular events*, Infor. and Comput., 110 (1994), pp. 366–390.

[15] ——, *Kleene algebra with tests and commutativity conditions*, in Proc. Second Int. Workshop Tools and Algorithms for the Construction and Analysis of Systems (TACAS'96), T. Margaria and B. Steffen, eds., vol. 1055 of Lect. Notes in Comput. Sci., Springer, March 1996, pp. 14–33.

[16] D. Kozen and F. Smith, *Kleene algebra with tests: completeness and decidability*, Tech. Rep. TR96-1582, Cornell University, April 1996.

[17] W. Kuich, *The Kleene and Parikh theorem in complete semirings*, in Proc. 14th Colloq. Automata, Languages, and Programming, Ottmann, ed., vol. 267 of Lect. Notes in Comput. Sci., EATCS, Springer, 1987, pp. 212–225.

[18] W. Kuich and A. Salomaa, *Semirings, Automata, and Languages*, Springer, 1986.

[19] K. C. Ng, *Relation Algebras with Transitive Closure*, PhD thesis, University of California, Berkeley, 1984.

[20] V. Pratt, *Dynamic algebras as a well-behaved fragment of relation algebras*, in Proc. Conf. on Algebra and Computer Science, D. Pigozzi, ed., vol. 425 of Lect. Notes in Comput. Sci., Springer, June 1988, pp. 77–110.

[21] L. J. Stockmeyer and A. R. Meyer, *Word problems requiring exponential time*, in Proc. 5th Symp. Theory of Computing, ACM, 1973, pp. 1–9.

[22] A. Tarski, *On the calculus of relations*, J. Symb. Logic, 6 (1941), pp. 65–106.