On Hoare Logic and Kleene Algebra with Tests

DEXTER KOZEN
Cornell University

We show that Kleene algebra with tests (KAT) subsumes propositional Hoare logic (PHL). Thus the specialized syntax and deductive apparatus of Hoare logic are inessential and can be replaced by simple equational reasoning. In addition, we show that all relationally valid inference rules are derivable in KAT and that deciding the relational validity of such rules is *PSPACE*-complete.

Categories and Subject Descriptors: D.2.2 [**Software Engineering**]: Design Tools and Techniques—*Structured programming*; D.2.4 [**Software Engineering**]: Software/Program Verification—*Correctness proofs*; D.3.3 [**Programming Languages**]: Language Constructs and Features—*Control structures*; F.3.1 [**Logics and Meanings of Programs**]: Specifying and Verifying and Reasoning about Programs—*Assertions*; *Invariants*; *Logics of programs*; *Mechanical verification*; *Pre- and post-conditions*; *Specification techniques*; F.3.2 [**Logics and Meanings of Programs**]: Semantics of Programming Languages—*Algebraic approaches to semantics*; F.3.3 [**Logics and Meanings of Programs**]: Studies of Program Constructs—*Control primitives*; I.1.1 [**Symbolic and Algebraic Manipulation**]: Expressions and Their Representation—*Simplification of expressions*; I.1.3 [**Symbolic and Algebraic Manipulation**]: Languages and Systems—*Special-purpose algebraic systems*; I.2.2 [**Artificial Intelligence**]: Automatic Programming—*Program modification*; *Program synthesis*; *Program transformation*; *Program verification*

General Terms: Design, Languages, Theory, Verification

Additional Key Words and Phrases: Dynamic logic, Hoare logic, Kleene algebra, Kleene algebra with tests, specification

1. INTRODUCTION

Hoare logic, introduced by C. A. R. Hoare in 1969 [Hoare 1969], was the first formal system for the specification and verification of well-structured

The support of the National Science Foundation under grant CCR-9708915 is gratefully acknowledged. This article is a revised and expanded version of a paper which appeared in the Proceedings of the Conference on Logic in Computer Science (LICS '99), pp. 167–172.
Author's address: Department of Computer Science, Cornell University, Ithaca, NY 14853-7501; email: kozen@cs.cornell.edu.

programs. This pioneering work initiated the field of program correctness and inspired dozens of technical articles [Cook 1978; Clarke et al. 1983; Cousot 1990]. For this achievement among others, Hoare received the Turing Award in 1980.

Hoare logic uses a specialized syntax involving *partial correctness assertions* (PCAs) of the form $\{b\}\ p\ \{c\}$ and a deductive apparatus consisting of a system of specialized rules of inference. Under certain conditions, these rules are relatively complete [Cook 1978]; essentially, the propositional fragment of the logic can be used to reduce partial correctness assertions to static assertions about the underlying domain of computation.

In this article we show that this propositional fragment, which we call *propositional Hoare logic* (PHL), is subsumed by *Kleene algebra with tests* (KAT), an equational algebraic system introduced in Kozen [1997]. The reduction transforms PCAs to ordinary equations and the specialized rules of inference to equational implications (universal Horn formulas). The transformed rules are all derivable in KAT by pure equational reasoning. More generally, we show that all Hoare-style inference rules of the form

$$\frac{\{b_1\}\ p_1\ \{c_1\}, \ldots, \{b_n\}\ p_n\ \{c_n\}}{\{b\}\ p\ \{c\}} \tag{1}$$

that are valid over relational models are derivable in KAT; this is trivially false for PHL. We also show that deciding the relational validity of such rules is *PSPACE*-complete.

A Kleene algebra with tests is defined simply as a Kleene algebra with an embedded Boolean subalgebra. Possible interpretations include the various standard relational and trace-based models used in program semantics, and KAT is complete for the equational theory of these models [Kozen and Smith 1996]. This work shows that the reasoning power represented by propositional Hoare logic is captured in a concise, purely equational system KAT that is complete over various natural classes of interpretations and whose exact complexity is known. Thus for all practical purposes KAT can be used in place of the Hoare rules in program correctness proofs.

1.1 Related Work

Equational logic possesses a rich theory and is the subject of numerous papers and texts [Taylor 1979]. Its power and versatility in program specification and verification are widely recognized [O'Donnell 1985; Goguen and Malcolm 1996].

The equational nature of Hoare logic has been observed previously. Manes and Arbib [1986] formulate Hoare logic in partially additive semirings and categories. The encoding of the PCA $\{b\}\ p\ \{c\}$ as the equation $bp\bar{c} = 0$ is observed there. They consider only relational models, and the treatment of iteration is infinitary. Bloom and Ésik [1991] reduce Hoare logic to the equational logic of iteration theories. They do not restrict their attention to **while** programs but capture all flowchart schemes, requiring extra notation for insertion, tupling, and projection. Their development is done in the framework of category theory. Semantic models consist of morphisms in algebraic theories, a particular kind of category. Other related work can be found in Bloom and Ésik [1992] and Main and Black [1990].

The encoding of the **while** programming constructs using the regular operators and tests originated with propositional dynamic logic (PDL) [Fischer and Ladner 1979]. Although strictly less expressive than PDL, KAT has a number of advantages: (1) it isolates the equational part of PDL, allowing program equivalence proofs to be expressed in their natural form; (2) it conveniently overloads the operators $+$, $\cdot$, 0, 1, allowing concise and elegant algebraic proofs; (3) it is *PSPACE*-complete [Cohen et al. 1996], whereas PDL is *EXPTIME*-complete [Fischer and Ladner 1979]; (4) interpretations are not restricted to relational models, but may be any algebraic structure satisfying the axioms; and (5) it admits various general and useful algebraic constructions such as the formation of algebras of matrices over a KAT, which among other things allows a natural encoding of automata.

Halpern and Reif [1983] prove *SPACE*-completeness of strict deterministic PDL, but neither the upper nor the lower bound of our *PSPACE*-completeness result follows from theirs. Not only are PDL semantics restricted to relational models, but the arguments of Halpern and Reif [1983] depend on an additional nonalgebraic restriction: the relations interpreting atomic programs must be single-valued. Without this restriction, even if only **while** programs are allowed, PDL is exponential-time hard. In contrast, KAT imposes no such restrictions.

In Section 2 we review the definitions of Hoare logic and Kleene algebra with tests. In Section 3 we reduce PHL to KAT and derive the Hoare rules as theorems of KAT. In Section 4 we strengthen this result to show that KAT is complete for relationally valid rules of the form (1). In Section 5 we prove that the problem of deciding the relational validity of such rules is *PSPACE*-complete.

2. PRELIMINARY DEFINITIONS

2.1 Hoare Logic

Hoare logic is a system for reasoning inductively about well-structured programs. A comprehensive introduction can be found in Cousot [1990].

A common choice of programming language in Hoare logic is the language of **while** programs. The first-order version of this language contains a simple assignment $x := e$, conditional test **if** b **then** p **else** q, sequential composition $p\ ;\ q$, and a looping construct **while** b **do** p.

The basic assertion of Hoare logic is the *partial correctness assertion*

$$\{b\}\ p\ \{c\}, \tag{2}$$

where b and c are formulas and p is a program. Intuitively, this statement asserts that whenever b holds before the execution of the program p, then if and when p halts, c is guaranteed to hold of the output state. It does not assert that p must halt.

Semantically, programs p in Hoare logic and dynamic logic (DL) are usually interpreted as binary input/output relations $p^{\mathcal{M}}$ on a domain of computation $\mathcal{M}$, and assertions are interpreted as subsets of $\mathcal{M}$ [Cook 1978; Pratt 1978]. The definition of the relation $p^{\mathcal{M}}$ is inductive on the structure of p; for example, $(p\ ;\ q)^{\mathcal{M}} = p^{\mathcal{M}} \circ q^{\mathcal{M}}$, the ordinary relational composition of the relations corresponding to p and q. The meaning of the PCA (2) is the same as the meaning of the DL formula $b \to [p]c$, where $\to$ is ordinary propositional implication, and the modal construct $[p]c$ is interpreted in the model $\mathcal{M}$ as the set of states s such that for all $(s, t) \in p^{\mathcal{M}}$, the output state t satisfies c.

Hoare logic provides a system of specialized rules for deriving valid PCAs, one rule for each programming construct. The verification process is inductive on the structure of programs. The traditional Hoare inference rules are

Assignment Rule.

$$\{b[x/e]\}\ x := e\ \{b\} \tag{3}$$

Composition Rule.

$$\frac{\{b\}\ p\ \{c\},\ \{c\}\ q\ \{d\}}{\{b\}\ p\ ;\ q\ \{d\}} \tag{4}$$

Conditional Rule.

$$\frac{\{b \wedge c\}\ p\ \{d\},\ \{\neg b \wedge c\}\ q\ \{d\}}{\{c\}\ \mathbf{if}\ b\ \mathbf{then}\ p\ \mathbf{else}\ q\ \{d\}} \tag{5}$$

While Rule.

$$\frac{\{b \wedge c\}\ p\ \{c\}}{\{c\}\ \mathbf{while}\ b\ \mathbf{do}\ p\ \{\neg b \wedge c\}} \tag{6}$$

Weakening Rule.

$$\frac{b' \to b,\ \ \{b\}\ p\ \{c\},\ \ c \to c'}{\{b'\}\ p\ \{c'\}}. \tag{7}$$

Propositional Hoare logic consists of atomic proposition and program symbols, the usual propositional connectives, **while** program constructs, and PCAs built from these. Atomic programs are interpreted as binary relations on a set $\mathcal{M}$, and atomic propositions are interpreted as subsets of $\mathcal{M}$. The deduction system of PHL consists of the composition, conditional, while, and weakening rules (4)–(7) and propositional logic. The assignment rule (3) is omitted, since there is no first-order relational structure over which to interpret program variables; in practice, its role is played by PCAs over atomic programs that are postulated as assumptions.

In PHL, we are concerned with the problem of determining the validity of rules of the form

$$\frac{\{b_1\}\ p_1\ \{c_1\}, \ldots, \{b_n\}\ p_n\ \{c_n\}}{\{b\}\ p\ \{c\}} \tag{8}$$

over relational interpretations. The premises $\{b_i\}\ p_i\ \{c_i\}$ take the place of the assignment rule (3) and are an essential part of the formulation.

2.2 Kleene Algebra

Kleene algebra (KA) is the algebra of regular expressions [Kleene 1956; Conway 1971]. The axiomatization used here is from Kozen [1994]. A *Kleene algebra* is an algebraic structure $(\mathcal{K}, +, \cdot, {}^*, 0, 1)$ that is an idempotent semiring under $+, \cdot, 0, 1$ satisfying

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

$$1 + p^*p = p^* \tag{10}$$

$$q + pr \le r \rightarrow p^*q \le r \tag{11}$$

$$q + rp \le r \rightarrow qp^* \le r \tag{12}$$

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

$$p \le q \overset{\text{def}}{\Leftrightarrow} p + q = q.$$

The operation $+$ gives the supremum with respect to the natural order $\le$. Instead of (11) and (12), we might take the equivalent axioms

$$pr \le r \rightarrow p^*r \le r \tag{13}$$

$$rp \le r \rightarrow rp^* \le r. \tag{14}$$

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

Kleene algebra is a versatile system with many useful interpretations. 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 used in shortest-path algorithms and models consisting of convex polyhedra used in computational geometry [Iwano and Steiglitz 1990].

The following are some typical identities that hold in all Kleene algebras:

$$(p^*q)^*p^* = (p + q)^* \tag{15}$$

$$p(qp)^* = (pq)^*p \tag{16}$$

$$(pq)^* = 1 + p(qp)^*q \tag{17}$$

$$p^* = (pp)^*(1 + p). \tag{18}$$

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

The completeness result of Kozen [1994] says that all true identities between regular expressions interpreted as regular sets of strings are derivable from the axioms of Kleene algebra. In other words, the algebra of regular sets of strings over the finite alphabet Σ is the free Kleene algebra on generators Σ. The axioms are also complete for the equational theory of relational models.

See Kozen [1994] for a more thorough introduction.

2.3 Kleene Algebra with Tests (KAT)

Kleene algebras with tests were introduced in Kozen [1997] and the theory was further developed in Kozen and Smith [1996] and Cohen et al. [1996]. A Kleene algebra with tests is just a Kleene algebra with an embedded Boolean subalgebra. That is, it is a two-sorted structure

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

such that

—$(\mathcal{K}, +, \cdot, {}^*, 0, 1)$ is a Kleene algebra,

—$(\mathcal{B}, +, \cdot, \bar{}\,, 0, 1)$ is a Boolean algebra, and

—$\mathcal{B} \subseteq \mathcal{K}$.

The Boolean complementation operator $\bar{}$ is defined only on $\mathcal{B}$. Elements of $\mathcal{B}$ are called *tests*. The letters p, q, r, s denote arbitrary elements of $\mathcal{K}$, and a, b, c denote tests.

This deceptively simple definition actually carries a lot of information in a concise package. The operators $+$, $\cdot$, 0, 1 each play two roles: applied to arbitrary elements of $\mathcal{K}$, they refer to nondeterministic choice, composition, fail, and skip, respectively; and applied to tests, they take on the additional meaning of Boolean disjunction, conjunction, falsity, and truth, respectively. These two usages do not conflict—for example, sequential testing of b and c is the same as testing their conjunction—and their coexistence admits considerable economy of expression.

The encoding of the **while** program constructs is as in PDL [Fischer and Ladner 1979]:

$$p\ ;q \overset{\text{def}}{=} pq \tag{19}$$

$$\textbf{if } b \textbf{ then } p \textbf{ else } q \overset{\text{def}}{=} bp + \bar{b}q \tag{20}$$

$$\textbf{while } b \textbf{ do } p \overset{\text{def}}{=} (bp)^*\bar{b}. \tag{21}$$

For applications in program verification, the standard interpretation would be a Kleene algebra of binary relations on a set and the Boolean algebra of subsets of the identity relation. One could also consider trace models, in which the Kleene elements are sets of traces (sequences of states) and in which the Boolean elements are sets of states (traces of length 0). As with KA, one can form the algebra Mat($\mathcal{K}$, $\mathcal{B}$, n) of $n \times n$ matrices over a KAT ($\mathcal{K}$, $\mathcal{B}$); the Boolean elements of this structure are the diagonal matrices over $\mathcal{B}$. There is also a language-theoretic model that plays the same role in KAT that the regular sets of strings over a finite alphabet play in KA, namely the family of regular sets of *guarded strings* over a finite alphabet Σ with guards from a set **B**. This is the free KAT on generators Σ, **B**; that is, the equational theory of this structure is exactly the set of all equational consequences of the KAT axioms. Moreover, KAT is complete for the equational theory of relational models [Kozen and Smith 1996].

3. KAT AND HOARE LOGIC

In this section we encode Hoare logic in KAT and derive the Hoare composition, conditional, while, and weakening rules as theorems of KAT. We will strengthen this result in Section 4 by showing that KAT can derive all relationally valid rules of the form (8).

The PCA $\{b\}\ p\ \{c\}$ is encoded in KAT by the equation

$$bp\bar{c} = 0. \tag{22}$$

Intuitively, this says that the program p with preguard b and postguard $\bar{c}$ has no halting execution. An equivalent formulation is

$$bp = bpc, \tag{23}$$

which says intuitively that testing c after executing bp is always redundant.

The equivalence of (22) and (23) can be argued easily in KAT. This equivalence was previously observed by Manes and Arbib [1986]. Assuming (22),

$$\begin{aligned} bp &= bp(c+\bar{c}) && \text{by the axiom } a1 = a \text{ and Boolean algebra}\\ &= bpc + bp\bar{c} && \text{by distributivity}\\ &= bpc && \text{by (22) and the axiom } a + 0 = a. \end{aligned}$$

Conversely, assuming (23),

$$\begin{aligned} bp\bar{c} &= bpc\bar{c} && \text{by (23)}\\ &= bp0 && \text{by associativity and Boolean algebra}\\ &= 0 && \text{by the axiom } a0 = 0. \end{aligned}$$

Equation (23) is equivalent to the inequality $bp \le bpc$, since the reverse inequality is a thorem of KAT; it follows immediately from the axiom $c \le 1$ of Boolean algebra and monotonicity of multiplication.

Using (19)–(21) and (23), the Hoare rules (4)–(7) take the following form:

Composition Rule.

$$bp = bpc \wedge cq = cqd \rightarrow bpq = bpqd \tag{24}$$

Conditional Rule.

$$bcp = bcpd \wedge \bar{b}cq = \bar{b}cqd \rightarrow c(bp + \bar{b}q) = c(bp + \bar{b}q)d \tag{25}$$

While Rule.

$$bcp = bcpc \rightarrow c(bp)^*\bar{b} = c(bp)^*\bar{b}\bar{b}c \tag{26}$$

Weakening Rule.

$$b' \le b \wedge bp = bpc \wedge c \le c' \rightarrow b'p = b'pc' \tag{27}$$

These implications are to be interpreted as universal Horn formulas; that is, the variables are implicitly universally quantified. To establish the adequacy of the translation, we show that (24)–(27) encoding the Hoare rules (4)–(7) are theorems of KAT.

THEOREM 1. *The universal Horn formulas (24)–(27) are theorems of KAT.*

PROOF. First we derive (24). Assuming the premises

$$bp = bpc \tag{28}$$

$$cq = cqd, \tag{29}$$

we have

$$\begin{aligned} bpq &= bpcq && \text{by (28)} \\ &= bpcqd && \text{by (29)} \\ &= bpqd && \text{by (28).} \end{aligned}$$

Thus the implication (24) holds.

For (25), assume the premises

$$bcp = bcpd \tag{30}$$

$$\bar{b}cq = \bar{b}cqd. \tag{31}$$

Then

$$\begin{aligned} c(bp + \bar{b}q) &= cbp + c\bar{b}q && \text{by distributivity} \\ &= bcp + \bar{b}cq && \text{by commutativity of tests} \\ &= bcpd + \bar{b}cqd && \text{by (30) and (31)} \\ &= cbpd + c\bar{b}qd && \text{by commutativity of tests} \\ &= c(bp + \bar{b}q)d && \text{by distributivity.} \end{aligned}$$

For (26), by trivial simplifications it suffices to show

$$cbp \le cbpc \rightarrow c(bp)^* \le c(bp)^*c.$$

Assume

$$cbp \le cbpc. \tag{32}$$

By (12) we need only show

$$c + c(bp)^*cbp \le c(bp)^*c.$$

But

$$\begin{aligned} c + c(bp)^*cbp &\le c + c(bp)^*cbpc && \text{by (32) and monotonicity} \\ &\le c1c + c(bp)^*cbpc && \text{by Boolean algebra} \\ &\le c(1 + (bp)^*cbp)c && \text{by distributivity} \\ &\le c(1 + (bp)^*bp)c && \text{by monotonicity} \\ &\le c(bp)^*c && \text{by (10).} \end{aligned}$$

Finally, for (27), we can rewrite the rule as

$$b' \le b \wedge bp\bar{c} = 0 \wedge \bar{c}' \le \bar{c} \rightarrow b'p\bar{c}' = 0,$$

which follows immediately from the monotonicity of multiplication. □

4. A COMPLETENESS THEOREM

Theorem 3.1 says that for any proof rule of PHL, or more generally, for any rule of the form

$$\frac{\{b_1\}\, p_1\, \{c_1\}, \ldots, \{b_n\}\, p_n\, \{c_n\}}{\{b\}\, p\, \{c\}}$$

derivable in PHL, the corresponding equational implication (universal Horn formula)

$$b_1 p_1 \bar{c}_1 = 0 \wedge \cdots \wedge b_n p_n \bar{c}_n = 0 \rightarrow bp\bar{c} = 0 \tag{33}$$

is a theorem of KAT. In this section we strengthen this result to show (Corollary 4.2) that *all* universal Horn formulas of the form

$$r_1 = 0 \wedge \cdots \wedge r_n = 0 \rightarrow p = q \tag{34}$$

that are relationally valid (true in all relational models) are theorems of KAT; in other words, KAT is complete for universal Horn formulas of the form (34) over relational interpretations. This result subsumes Theorem 3.1, since the Hoare rules are relationally valid. Corollary 4.2 is trivially false for PHL; for example, the rule

$$\frac{\{c\}\ \textbf{if}\ b\ \textbf{then}\ p\ \textbf{else}\ p\ \{c\}}{\{c\}\ p\ \{c\}}$$

is not derivable, since the Hoare rules only increase the length of programs.

In Kozen and Smith [1996], based on a technique of Cohen [1994] for KA, we showed that a formula of KAT of the form (34) is valid over all models if and only if it is valid over *-continuous models; moreover, its validity over either class of models is equivalent to the validity of a pure equation. We strengthen this result by showing that this equivalence still holds when models are further restricted to relational models. The deductive completeness of KAT over relationally valid formulas of the form (34) follows as a corollary.

Let $T_{\Sigma,\mathbf{B}}$ denote the set of terms of the language of KAT over primitive propositions $\Sigma = \{a_1, \ldots, a_m\}$ and primitive tests $\mathbf{B} = \{b_1, \ldots, b_k\}$. Let $r_1, \ldots, r_n, p, q \in T_{\Sigma,\mathbf{B}}$. Let $u = (a_1 + \cdots + a_m)^*$, and let $r = \Sigma_i r_i$. The formula (34) is equivalent to $r = 0 \rightarrow p = q$. Consider the four conditions

$$\text{KAT} \models \mathrm{r} = 0 \rightarrow \mathrm{p} = \mathrm{q}, \tag{35}$$

$$\text{KAT*} \models r = 0 \rightarrow p = q, \tag{36}$$

$$\text{REL} \models r = 0 \rightarrow p = q, \tag{37}$$

$$\models p + uru = q + uru. \tag{38}$$

It does not matter whether (38) is preceded by KAT, KAT*, or REL, since the equational theories of these classes coincide [Kozen and Smith 1996]. It was shown in Kozen and Smith [1996] that the metastatements (35), (36), and (38) are equivalent. We wish to add (37) to this list.

The algebra $\mathcal{G}_{\Sigma, \mathbf{B}}$ of regular sets of guarded strings over Σ, $\mathbf{B}$ and the standard interpretation $G : T_{\Sigma, \mathbf{B}} \rightarrow \mathcal{G}_{\Sigma, \mathbf{B}}$ were defined in Kozen and Smith [1996]. We briefly review the definitions here. An *atom* of $\mathbf{B}$ is a term of the form $c_1 c_2 \cdot \cdot \cdot c_k$, where c_i is either b_i or $\bar{b}_i$. An atom represents an atom of the free Boolean algebra generated by $\mathbf{B}$. Atoms are denoted α, β, ... A *guarded string over* Σ, $\mathbf{B}$ is a term of the form

$$\beta_0 p_0 \beta_1 p_1 \beta_2 \cdots \beta_{n-1} p_{n-1} \beta_n,$$

where each $p_i \in \Sigma$ and each β_i is an atom. This includes the case $n = 0$, so atoms are guarded strings. If $x\alpha$, βy are guarded strings and $\alpha = \beta$, then their product is $x\alpha y$. If $\alpha \neq \beta$, then the product does not exist. We can form the Kleene algebra of all sets of guarded strings with operations

$$A + B \overset{\text{def}}{=} A \cup B$$

$$AB \overset{\text{def}}{=} \{x\beta y \mid x\beta \in A, \beta y \in B\}$$

$$A^* \overset{\text{def}}{=} \bigcup_n A^n$$

$$0 \overset{\text{def}}{=} \emptyset$$

$$1 \overset{\text{def}}{=} \{\text{atoms of } \mathbf{B}\}.$$

This becomes a KAT by taking the Boolean algebra of tests to be the powerset of the set 1. The map G is defined to be the unique homomorphic map on $T_{\Sigma, \mathbf{B}}$ extending

$$G(a) \overset{\text{def}}{=} \{\alpha a \beta \mid \alpha, \beta \text{ are atoms of } \mathbf{B}\}, a \in \Sigma$$

$$G(b) \overset{\text{def}}{=} \{\beta \mid \beta \leq b\}, b \in \mathbf{B}$$

where $\beta \leq b$ denotes that b occurs positively in β. The algebra $\mathcal{G}_{\Sigma, \mathbf{B}}$ is defined to be the image of $T_{\Sigma, \mathbf{B}}$ under the map G. It was shown in Kozen and Smith [1996] that $\mathcal{G}_{\Sigma, \mathbf{B}}$ is the free KAT on generators Σ, $\mathbf{B}$ in the sense that for any terms $s, t \in T_{\Sigma, \mathbf{B}}$

$$\vDash s = t \Leftrightarrow G(s) = G(t). \tag{39}$$

Note that $G(u)$ is the set of all guarded strings over Σ, $\mathbf{B}$.

THEOREM 4.1 *The metastatements (35)–(38) are equivalent.*

PROOF. Since REL $\subseteq$ KAT* $\subseteq$ KAT, the implications (35) $\rightarrow$ (36) $\rightarrow$ (37) hold trivially. Also, it is clear that

$$\text{KAT} \vDash p + uru = q + uru \rightarrow (r = 0 \rightarrow p = q).$$

Therefore (39) $\rightarrow$ (35) as well. It thus remains to show that (37) $\rightarrow$ (38). Writing equations as pairs of inequalities, it suffices to show

$$\text{REL } \vDash r = 0 \rightarrow p \leq q \Rightarrow \vDash p \leq q + uru. \tag{40}$$

To show (40), we construct a relational model $\mathcal{R}$ on states $G(u) - G(uru)$. Note that if $x, y, z \in G(u)$ such that $xyz \in G(u) - G(uru)$, then $y \in G(u) - G(uru)$. If $G(u) \subseteq G(uru)$ then we are done, since in that case $G(p) \subseteq G(u) \subseteq G(uru)$, and the right-hand side of (40) follows immediately from (39). Similarly, if $G(1) \subseteq G(uru)$ then $G(u) \subseteq G(uuru) \subseteq G(uru)$ and the same argument applies. We can therefore assume without loss of generality that both $G(u) - G(uru)$ and $G(1) - G(uru)$ are nonempty.

The atomic symbols are interpreted in $\mathcal{R}$ as follows:

$$R(a) \overset{\text{def}}{=} \{(x, xa\beta) \mid xa\beta \in G(u) - G(uru)\}, a \in \Sigma$$

$$R(a) \overset{\text{def}}{=} \{(x, x) \mid x = x\beta \in G(u) - G(uru), \beta \leq b\}, b \in \mathbf{B}$$

The interpretations of compound expressions are defined inductively in the standard way for relational models.

We now show that for any $t \in T_{\Sigma, \mathbf{B}}$,

$$R(t) = \{(x, xy) \mid xy \in G(u) - G(uru), y \in G(t)\} \tag{41}$$

by induction on the structure of t. For primitive programs a and tests b,

$$\begin{aligned} R(a) &= \{(x, xa\beta) \mid xa\beta \in G(u) - G(uru)\} \\ &= \{(x, x\alpha a\beta) \mid x\alpha a\beta \in G(u) - G(uru)\} \\ &= \{(x, xy) \mid xy \in G(u) - G(uru), y \in G(a)\}, \end{aligned}$$

$$\begin{aligned} R(b) &= \{(x, x) \mid x = x\beta \in G(u) - G(uru),\ \beta \in G(b)\} \\ &= \{(x, x\beta) \mid x\beta \in G(u) - G(uru),\ \beta \in G(b)\} \\ &= \{(x, xy) \mid xy \in G(u) - G(uru), y \in G(b)\}. \end{aligned}$$

For the constants 0 and 1, we have

$$\begin{aligned} R(0) &= \emptyset \\ &= \{(x, xy) \mid xy \in G(u) - G(uru), y \in G(0)\}, \\ R(1) &= \{(x, x) \mid x \in G(u) - G(uru)\} \\ &= \{(x, xy) \mid xy \in G(u) - G(uru), y \in G(1)\}. \end{aligned}$$

For compound expressions, we have

$$\begin{aligned} R(s + t) &= R(s) \cup R(t) \\ &= \{(x, xy) \mid xy \in G(u) - G(uru), y \in G(s)\} \\ &\quad \cup \{(x, xy) \mid xy \in G(u) - G(uru), y \in G(t)\} \\ &= \{(x, xy) \mid xy \in G(u) - G(uru), y \in G(s) \cup G(t)\} \\ &= \{(x, xy) \mid xy \in G(u) - G(uru), y \in G(s + t)\}, \\ R(st) &= R(s) \circ R(t) \\ &= \{(x, xz) \mid xz \in G(u) - G(uru), z \in G(s)\} \\ &\quad \circ \{(y, yw) \mid yw \in G(u) - G(uru), w \in G(t)\} \\ &= \{(x, xzw) \mid xzw \in G(u) - G(uru), z \in G(s), w \in G(t)\} \\ &= \{(x, xy) \mid xy \in G(u) - G(uru), y \in G(st)\}, \\ R(t^*) &= \bigcup_n R(t^n) \\ &= \bigcup_n \{(x, xy) \mid xy \in G(u) - G(uru), y \in G(t^n)\} \\ &= \{(x, xy) \mid xy \in G(u) - G(uru), y \in \bigcup_n G(t^n)\} \\ &= \{(x, xy) \mid xy \in G(u) - G(uru), y \in G(t^*)\} \end{aligned}$$

We now show (40). Suppose the left-hand side holds. By (41),

$$R(r) \overset{\text{def}}{=} \{(x, xy) \mid xy \in G(u) - G(uru), y \in G(r)\} = \emptyset.$$

By the left-hand side of (40), $R(p) \subseteq R(q)$. In particular, for any $x \in G(p) - G(uru)$, $(\epsilon, x) \in R(p)$; therefore $(\epsilon, x) \in R(q)$ as well; thus $x \in G(q) - G(uru)$. But this says $G(p) - G(uru) = G(q) - G(uru)$; thus $G(p) \subseteq G(q) \cup G(uru) = G(q + uru)$. It follows from (39) that the right-hand side of (40) holds. □

Corollary 4.2 KAT is deductively complete for formulas of the form (34) over relational models.

PROOF. If the formula (34) is valid over relational models, then by Theorem 4.1, (38) holds. Since KAT is complete for valid equations,

$$\text{KAT} \vdash p + uru = q + uru.$$

But clearly

$$\text{KAT} \vdash p + uru = q + uru \wedge r = 0 \rightarrow p = q,$$

therefore

$$\text{KAT} \vdash r = 0 \rightarrow p = q.$$

□

5. COMPLEXITY

As defined in Section 2.1, the decision problem for PHL is to determine whether a given rule of the form (8) is valid over all relational interpretations. Note that *PSPACE*-hardness does not follow immediately from the *PSPACE*-hardness of the equational theory, since the conclusion $\{b\}\, p\, \{c\}$ is of a restricted form $q = 0$. Indeed, E. Cohen has shown[1] that the complexity of valid equations of the form $q = 0$ in KAT is *co-NP*-complete.

THEOREM 5.1 *The decision problem for PHL is PSPACE-complete.*

PROOF. The reduction of Sections 3 and 4 transforms the decision problem for PHL to the problem of the universal validity of Horn formulas of the form (34). As shown in Section 4, this can be reduced to testing the validity of a single equation without premises. The equational theory of KAT is decidable in *PSPACE* [Cohen et al. 1996]; thus the decision problem for PHL is in *PSPACE*.

We now show that the problem is *PSPACE*-hard. This holds even if the premises $\{b_i\}\, p\, \{c_i\}$ are restricted to refer only to atomic programs, and even if they are restricted to refer only to a single atomic program p. We give a direct encoding of the computation of a polynomial space-bounded one-tape deterministic Turing machine in an instance of the decision problem for PHL. Our approach is similar to Halpern and Reif [1983], using the premises $\{b_i\}\, p\, \{c_i\}$ to circumvent the determinacy assumption. E. Cohen (personal communication) has given an alternative hardness proof using the universality problem for regular expressions.

[1]Cohen, E. 1999. Personal communication.

Consider the computation of a polynomially space-bounded one-tape deterministic Turing machine M on some input x of length n. Let N be a polynomial bound on the amount of space used by M on input x. Let Q be the set of states of M; let Γ be its tape alphabet; let s be its start state; and let t be its unique halt state. We use polynomially many atomic propositional symbols with the following intuitive meanings:

$T_{i,a}$	"the ith tape cell currently contains symbol a," $0 \leq i \leq N$, $a \in \Gamma$,
H_i	"the tape head is currently scanning the ith tape cell," $0 \leq i \leq N$,
S_q	"the machine is currently in state q," $q \in Q$.

Let p be an atomic program. Intuitively, p represents the action of one step of M. We will devise a set of assumptions $\varphi_1, \ldots, \varphi_m$ that will say that p faithfully models the action of M. The PCA ψ will say that if started in state s on input x, the program

while the current state is not t **do** p

fails. The PCA ψ will be a logical consequence of $\varphi_1, \ldots, \varphi_m$ iff M does not halt on input x.

The start configuration of M on x consists of a left endmarker $\vdash$ written on tape cell 0, the input $x = a_1 \cdots a_n$ written on cells 1 through n, and the remainder of the tape filled with the blank symbol $\sqcup$ out to the Nth cell. The machine starts in state s scanning the left endmarker. This situation is captured by the propositional formula

$$\text{START} \overset{\text{def}}{=} T_{0,\vdash} \wedge \bigwedge_{1 \leq i \leq n} T_{i,a_i} \wedge \bigwedge_{n+1 \leq i \leq N} T_{i,\sqcup} \wedge S_s \wedge H_0.$$

We will need a formula to ensure that M is in at most one state, that it is scanning at most one tape cell, and that there is at most one symbol written on each tape cell:

$$\text{FORMAT} \overset{\text{def}}{=} \bigwedge_{0 \leq i \leq N} \bigwedge_{a \neq b} \neg(T_{i,a} \wedge T_{i,b}) \wedge \bigwedge_{p \neq q} \neg(S_p \wedge S_q)$$
$$\wedge \bigwedge_{0 \leq i < j \leq N} \neg(H_i \wedge H_j)$$

We include the PCA

$$\{\text{FORMAT}\}\, p\, \{\text{FORMAT}\}$$

as one of the assumptions φ_i to ensure that FORMAT is an invariant of p and therefore preserved throughout the simulation of M.

Suppose the transition function of M says that when scanning a cell containing symbol a in state p, M prints the symbol b on that cell, moves right, and enters state q. We capture this constraint by the family of PCAs

$$\{T_{i,a} \wedge H_i \wedge S_p\}\; p\; \{T_{i,b} \wedge H_{i+1} \wedge S_q\},\;\; 0 \le i \le N-1.$$

All these PCAs are included for each possible transition of the machine; there are only polynomially many in all.

We must also ensure that the symbols on tape cells not currently being scanned do not change; this is accomplished by the family of PCAs

$$\{T_{i,a} \wedge \neg H_i\}\; p\; \{T_{i,a}\},\;\; 0 \le i \le N,\;\; a \in \Gamma.$$

These are the assumptions $\varphi_1, \ldots, \varphi_m$ in our instance of the decision problem. It is apparent that under any interpretation of p satisfying these PCAs, successive executions of p starting from any state satisfying START $\wedge$ FORMAT move only to states whose values for the atomic propositions S_q, $T_{i,a}$, and H_i model valid configurations of M, and the values change in such a way as to model the computation of M. Thus there is a reachable state satisfying S_t if and only if M halts on x.

We take as our conclusion ψ the PCA

$$\{\text{START} \wedge \text{FORMAT}\}\; \textbf{while}\; \neg S_t\; \textbf{do}\; p\; \{\text{FALSE}\},$$

which says intuitively that when started in the start configuration, repeatedly executing p will never cause M to enter state t. The PCA ψ is therefore a logical consequence of $\varphi_1, \ldots, \varphi_m$ if and only if M does not halt on x. □

ACKNOWLEDGMENTS

I thank Krzysztof Apt, Steve Bloom, Ernie Cohen, Zoltán Ésik, Joe Halpern, Greg Morrisett, Moshe Vardi, and Thomas Yan for valuable comments on an earlier version of this article.

REFERENCES

Bloom, S. L. and Ésik, Z. 1991. Floyd-Hoare logic in iteration theories. *J. ACM 38*, 4 (Oct. 1991), 887–934.

Bloom, S. L. and Ésik, Z. 1992. Program correctness and matricial iteration theories. In *Proceedings of the 7th International Conference on Mathematical Foundations of Programming Semantics* (MFPS '92), Springer Lecture Notes in Computer Science, vol. 598. Springer-Verlag, New York, 457–476.

Clarke, E. M., German, S. M., and Halpern, J. Y. 1983. Effective axiomatizations of Hoare logics. *J. ACM 30*, 3 (July), 612–636.

Cohen, E. 1994. Hypotheses in Kleene algebra. Available as ftp://ftp.telcordia.com/pub/ernie/research/homepage.html.

Cohen, E., Kozen, D., and Smith, F. 1996. The complexity of Kleene algebra with tests. Tech. Rep. 96-1598. Department of Computer Science, Cornell University, Ithaca, NY.

CONWAY, J. H. 1971. *Regular Algebra and Finite Machines*. Chapman and Hall, Ltd., London, UK.

COOK, S. A. 1978. Soundness and completeness of an axiom system for program verification. *SIAM J. Comput. 7*, 1 (Feb.), 70–90.

COUSOT, P. 1990. Methods and logics for proving programs. In *Handbook of Theoretical Computer Science, Vol. B: Formal Models and Semantics*, J. van Leeuwen, Ed. MIT Press, Cambridge, MA, 841–993.

FISCHER, M. J. AND LADNER, R. E. 1979. Propositional dynamic logic of regular programs. *J. Comput. Syst. Sci. 18*, 2, 194–211.

GOGUEN, J. A. AND MALCOLM, G. 1996. *Algebraic Semantics of Imperative Programs*. Foundations of Computing. MIT Press, Cambridge, MA.

HALPERN, J. Y. AND REIF, J. H. 1983. The propositional logic of deterministic, well-structured programs. *Theor. Comput. Sci. 27*, 127–165.

HOARE, C. 1969. An axiomatic basis for computer programming. *Commun. ACM 12*, 10, 576–583.

IWANO, K. AND STEIGLITZ, K. 1990. A semiring on convex polygons and zero-sum cycle problems. *SIAM J. Comput. 19*, 5 (Oct. 1990), 883–901.

KLEENE, S. C. 1956. Representation of events in nerve nets and finite automata. In *Automata Studies*, C. E. Shannon and J. McCarthy, Eds. Princeton University Press, Princeton, NJ, 3–41.

KOZEN, D. 1994. A completeness theorem for Kleene algebras and the algebra of regular events. *Inf. Comput. 110*, 2 (May 1, 1994), 366–390.

KOZEN, D. 1997. Kleene algebra with tests. *ACM Trans. Program. Lang. Syst. 19*, 3, 427–443.

KOZEN, D. AND SMITH, F. 1996. Kleene algebra with tests: Completeness and decidability. In *Proceedings of the 10th International Workshop on Computer Science Logic* (CSL '96, Utrecht, The Netherlands, Sept.), D. van Dalen and M. Bezem, Eds. Springer Lecture Notes in Computer Science Springer-Verlag, New York, 244–259.

MAIN, M. G. AND BLACK, D. L. 1990. Semantic models for total correctness and fairness. In *Proceedings of the fifth international conference on Mathematical Foundations of Programming Semantics* (Tulane Univ., New Orleans, LA, Mar. 29–Apr. 1, 1989), M. G. Main, A. C. Melton, and M. W. Mislove, Eds. Springer Lecture Notes in Computer Science, vol. 442. Springer-Verlag, New York, NY, 247–270.

MANES, E. G. AND ARBIB, M. A. 1986. *Algebraic Approaches to Program Semantics*. AKM series in theoretical computer science. Springer-Verlag, New York, NY.

O'DONNELL, M. J. 1985. *Equational Logic as a Programming Language*. MIT Press series in the foundations of computing. Massachusetts Institute of Technology, Cambridge, MA.

PRATT, V. R. 1978. A practical decision method for propositional dynamic logic. In *Proceedings of the 10th Symposium on Theory of Computing*, ACM Press, New York, NY, 326–337.

TAYLOR, W. 1979. Equational logic. *Houston J. Math.*, i–83. Survey.

Received: July 1999; revised: January 2000; accepted: January 2000