Automata on Guarded Strings and Applications

Dexter Kozen*
Cornell University

January 25, 2001

Abstract

Guarded strings are like ordinary strings over a finite alphabet P, except that atoms of the free Boolean algebra on a set of atomic tests B alternate with the symbols of P. The regular sets of guarded strings play the same role in Kleene algebra with tests as the regular sets of ordinary strings do in Kleene algebra.

In this paper we develop the elementary theory of finite automata on guarded strings, a generalization of the theory of finite automata on ordinary strings. We give several basic constructions, including determinization, state minimization, and an analog of Kleene's theorem.

We then use these results to verify a conjecture on the complexity of a complete Gentzen-style sequent calculus for partial correctness. We also show that a basic result of the theory of Boolean decision diagrams (BDDs), namely that minimal ordered BDDs are unique, is a special case of the Myhill-Nerode theorem for a class of automata on guarded strings.

1 Introduction

The regular sets of *guarded strings* were introduced in [6] as a language-theoretic model for Kleene algebra with tests (KAT). Guarded strings are like ordinary strings over a finite alphabet P, except that atoms of the free Boolean algebra on a set of atomic tests B alternate with the symbols of P. The regular sets of guarded strings play the same role in KAT as the regular sets of ordinary strings do in Kleene algebra; specifically, they form the free Kleene algebra with tests on generators P, B [6].

Recently we have found guarded strings useful in other contexts. In [7], we developed a complete Gentzen-style sequent calculus S for partial correctness. Guarded strings

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

played a central role in the completeness proof. We also conjectured that the decision problem for S was *PSPACE*-complete.

In this paper we verify that conjecture. The proof requires the development the elementary theory of finite automata on guarded strings, a generalization of the theory of finite automata on ordinary strings. We give several basic constructions, including determinization, state minimization, and an analog of Kleene's theorem. We also point out a connection to the complexity of BDDs (*binary* or *Boolean decision diagrams*), a well-studied structure in model checking. In particular, we observe that a basic result of the theory of BDDs, namely that minimal ordered BDDs are unique, is a special case of the Myhill–Nerode theorem for a class of deterministic automata on guarded strings.

2 Guarded Strings

We refer the reader to [4, 5] for an introduction to Kleene algebra with tests (KAT).

Guarded strings over P, B were introduced in [6]. Let $\mathsf{B} = \{b_1, \ldots, b_k\}$ and $\mathsf{P} = \{p_1, \ldots, p_m\}$ be fixed finite sets of atomic tests and atomic programs, respectively, and let $\overline{\mathsf{B}} \stackrel{\text{def}}{=} \{\overline{b} \mid b \in \mathsf{B}\}$. A *test* is a Boolean expression over B and a *program* is a term of KAT over P, B. An *atom* of B is a program $c_1 \cdots c_k$ such that $c_i \in \{b_i, \overline{b}_i\}$, $1 \leq i \leq k$. An atom represents a minimal nonzero element of the free Boolean algebra on B. We denote by $\mathcal{A}_\mathsf{B}$ the set of all atoms of B and use the symbols $\alpha, \beta, \ldots$ exclusively for atoms. For an atom α and a test b, we write $\alpha \leq b$ if $\alpha \to b$ is a propositional tautology.

A *guarded string* is a sequence $x = \alpha_0 q_1 \alpha_1 \cdots \alpha_{n-1} q_n \alpha_n$, where $n \geq 0$ and each $\alpha_i \in \mathcal{A}_\mathsf{B}$ and $q_i \in \mathsf{P}$. We define $\mathbf{first}(x) = \alpha_0$ and $\mathbf{last}(x) = \alpha_n$. The set of all guarded strings over P, B is denoted $\mathbf{GS}$. If $\mathbf{last}(x) = \mathbf{first}(y)$, we can form the *fusion product* xy by concatenating x and y, omitting the extra copy of the common atom. For example, if $x = \alpha p \beta$ and $y = \beta q \gamma$, then $xy = \alpha p \beta q \gamma$. If $\mathbf{last}(x) \neq \mathbf{first}(y)$, then xy does not exist.

For sets X, Y of guarded strings, define $X \diamond Y$ to be the set of all existing fusion products xy with $x \in X$ and $y \in Y$, and define X^n to be the product of n copies of X with respect to this operation. Each program p of KAT denotes a set $G(p)$ of guarded

strings as follows:

$$\begin{aligned}
G(p) &\stackrel{\text{def}}{=} \{\alpha p \beta \mid \alpha, \beta \in \mathcal{A}_{\mathsf{B}}\} && p \text{ an atomic program} \\
G(b) &\stackrel{\text{def}}{=} \{\alpha \in \mathcal{A}_{\mathsf{B}} \mid \alpha \leq b\} && b \text{ a test} \\
G(p+q) &\stackrel{\text{def}}{=} G(p) \cup G(q) \\
G(pq) &\stackrel{\text{def}}{=} G(p) \circ G(q) \\
G(p^*) &\stackrel{\text{def}}{=} \bigcup_{n \geq 0} G(p)^n.
\end{aligned}$$

A guarded string x is itself a program, and $G(x) = \{x\}$.

A set of guarded strings over P, B is *regular* if it is $G(p)$ for some program p. The family of regular sets of guarded strings over P, B is denoted $\mathsf{Reg}_{\mathsf{P},\mathsf{B}}$. It forms the free Kleene algebra with tests on generators P, B [6]; in other words, $G(p) = G(q)$ iff $p = q$ is a theorem of KAT. A key lemma used in that proof is the following:

Lemma 2.1 ([6]) *For any program p, there exists a program $\widehat{p}$ such that $p = \widehat{p}$ is a theorem of KAT and $G(\widehat{p}) = R(\widehat{p})$.*

Moreover, if $R(q) \subseteq \mathbf{GS}$, then $R(q) = G(q)$; this is because $G(q) = \bigcup\{ G(x) \mid x \in R(q)\}$ and $G(x) = \{x\}$ for $x \in \mathbf{GS}$. It was observed in [7] that this result implies that $\mathsf{Reg}_{\mathsf{P},\mathsf{B}}$ is closed under the Boolean operations. Our automata-theoretic characterization of $\mathsf{Reg}_{\mathsf{P},\mathsf{B}}$ will give an alternative proof of this result.

Expressions of KAT can also be interpreted as sets of traces or sets of binary relations in Kripke frames. A *Kripke frame* over P, B is a structure $(K,\ \mathfrak{m}_K)$, where K is a set of *states* and

$$\mathfrak{m}_K : \mathsf{P} \to 2^{K \times K} \qquad \mathfrak{m}_K : \mathsf{B} \to 2^K.$$

A *trace* in K is a sequence σ of the form $s_0 q_1 s_1 \cdots s_{n-1} q_n s_n$, where $n \geq 0$, $s_i \in K$, $q_i \in \mathsf{P}$, and $(s_i, s_{i+1}) \in \mathfrak{m}_K(q_{i+1})$ for $0 \leq i \leq n-1$. We define $\mathbf{first}(\sigma) = s_0$, $\mathbf{last}(\sigma) = s_n$, and $\mathbf{label}(\sigma) = q_1 \cdots q_n$. If $\mathbf{last}(\sigma) = \mathbf{first}(\tau)$, the trace $\sigma\tau$ is the trace consisting of σ followed by τ. If $\mathbf{last}(\sigma) \neq \mathbf{first}(\tau)$, then $\sigma\tau$ does not exist. A trace $s_0 q_1 s_1 \cdots s_{n-1} q_n s_n$ is *linear* if the s_i are distinct.

Expressions are interpreted in K as sets of traces according to the following inductive

definition:

$$\begin{aligned}
[\![p]\!]_K &\stackrel{\text{def}}{=} \{spt \mid (s,t) \in \mathfrak{m}_K(p)\}, \quad p \text{ an atomic program} \\
[\![b]\!]_K &\stackrel{\text{def}}{=} \mathfrak{m}_K(b), \quad b \text{ an atomic test} \\
[\![\overline{b}]\!]_K &\stackrel{\text{def}}{=} K - \mathfrak{m}_K(b) \\
[\![0]\!]_K &\stackrel{\text{def}}{=} \varnothing \\
[\![p+q]\!]_K &\stackrel{\text{def}}{=} [\![p]\!]_K \cup [\![q]\!]_K \\
[\![pq]\!]_K &\stackrel{\text{def}}{=} [\![p]\!]_K \circ [\![q]\!]_K \\
[\![p^*]\!]_K &\stackrel{\text{def}}{=} \bigcup_{n\geq 0} [\![p]\!]_K^n,
\end{aligned}$$

where $X \circ Y \stackrel{\text{def}}{=} \{\sigma\tau \mid \sigma \in X,\ \tau \in Y,\ \sigma\tau \text{ exists}\}$ and $X^0 \stackrel{\text{def}}{=} K$, $X^{n+1} \stackrel{\text{def}}{=} X \circ X^n$.

Every trace σ has an associated guarded string $\mathrm{gs}(\sigma)$ defined by

$$\mathrm{gs}(s_0 q_1 s_1 \cdots s_{n-1} q_n s_n) \stackrel{\text{def}}{=} \alpha_0 q_1 \alpha_1 \cdots \alpha_{n-1} q_n \alpha_n,$$

where α_i is the unique atom of B such that $s_i \in [\![\alpha_i]\!]_K$, and $\mathrm{gs}(\sigma)$ is the unique guarded string over P, B such that $\sigma \in [\![\mathrm{gs}(\sigma)]\!]_K$. The relationship between trace semantics and guarded strings is given by the following lemma.

Lemma 2.2 ([7]) *In any trace model K, for any program p and trace τ, $\tau \in [\![p]\!]_K$ iff $\mathrm{gs}(\tau) \in G(p)$. In other words, $[\![p]\!]_K = \mathrm{gs}^{-1}(G(p))$. The map $X \mapsto \mathrm{gs}^{-1}(X)$ is a* KAT *homomorphism from the algebra of regular sets of guarded strings to the algebra of regular sets of traces over K.*

3 Automata on Guarded Strings

A *finite automaton on guarded strings* (AGS) over atomic programs P and atomic tests B is just an ordinary finite automaton with transition labels $\mathsf{P} \cup \mathcal{B}$, where $\mathcal{B}$ is the set of tests built from atomic tests B, except that acceptance is defined differently. Strictly speaking, $\mathcal{B}$ is infinite; however, it is finite up to propositional equivalence, and the semantics of acceptance does not distinguish propositionally equivalent tests. Transitions labeled with atomic programs are called *action transitions* and those labeled with tests are called *test transitions*.

Ordinary finite automata with ε-transitions can be regarded as the special case in which $\mathsf{B} = \varnothing$, giving the two-element Boolean algebra $\{0,1\}$. An ε-transition is just a test transition with Boolean label 1. For nonempty B, tests can be more complicated.

Intuitively, nondeterministic automata on guarded strings work as follows. An input to the automaton is a guarded string over P and B. We start with a pebble on an input

state with the input pointer reading the first atom of the input string. At any point in the computation, the pebble is occupying a state, and the input pointer is pointing to an atom somewhere in the input string. If there is an action transition from the current state labeled with $p \in \mathsf{P}$, and the next program symbol in the input string is p, then we may nondeterministically choose to move the pebble along that transition and advance the input pointer beyond p. If there is a test transition from the current state labeled with a test $b \in \mathcal{B}$, and if that transition is enabled, then we may nondeterministically choose to move the pebble along that transition, but we do not advance the input pointer. The transition is enabled if the current atom α in the input string satisfies b, where we regard α as a truth assignment to B. The input is accepted if the pebble occupies an accept state while the input pointer is pointing to the last atom in the input string.

Formally, an automaton on guarded strings over P, B is a Kripke frame $M = (Q, \mathfrak{m}_M)$ over atomic programs $\mathsf{P} \cup \mathcal{B}$ and atomic tests $\varnothing$, along with a distinguished set $S \subseteq Q$ of *start states* and a distinguished set $F \subseteq Q$ of *final* or *accept states*. We write $u \xrightarrow[M]{d} v$ if $(u, v) \in \mathfrak{m}_M(d)$, $d \in \mathsf{P} \cup \mathcal{B}$, or just $u \xrightarrow{d} v$ if M is understood.

A guarded string y over P, B is said to be *accepted* by M if $y \in G(x)$ for some $x \in R(M)$, where $R(M)$ is the set of strings in $(\mathsf{P} \cup \mathcal{B})^*$ accepted by M under the ordinary definition of finite automaton acceptance. The set of all guarded strings over P, B accepted by M is denoted $G(M)$. Formally,

$$\begin{aligned} R(M) &\stackrel{\text{def}}{=} \{\mathbf{label}(\sigma) \mid \mathbf{first}(\sigma) \in S,\ \mathbf{last}(\sigma) \in F\} \\ G(M) &\stackrel{\text{def}}{=} H(R(M)), \end{aligned}$$

where σ represents a trace in M and H is the map

$$\begin{aligned} H &: 2^{(\mathsf{P} \cup \mathcal{B})^*} \to 2^{\mathbf{GS}} \\ H(A) &\stackrel{\text{def}}{=} \textstyle\bigcup_{x \in A} G(x). \end{aligned}$$

3.1 Kleene's Theorem

The following is the analog of Kleene's theorem for automata on guarded strings. We need the second clause for our complexity result in Section 4.1.

Theorem 3.1 *Automata on guarded strings over* P, B *accept all and only regular sets. Moreover, the size of the equivalent automaton* M *constructed from a given program* p *is linear in the size of* p*.*

Proof. Given a program p over P, B, consider it as a regular expression over the alphabet $\mathsf{P} \cup \mathcal{B}$ with the classical interpretation, and construct an equivalent finite automaton M with input alphabet $\mathsf{P} \cup \mathcal{B}$ as in the usual proof of Kleene's theorem (see e.g. [3]).

The construction is linear. Conversely, given a finite automaton M with input alphabet $\mathsf{P} \cup \mathcal{B}$, construct an equivalent regular expression p. In either direction, let $R(p)$ denote the regular subset of $(\mathsf{P} \cup \mathcal{B})^*$ denoted by p under the classical interpretation of regular expressions, and let $R(M)$ denote the subset of $(\mathsf{P} \cup \mathcal{B})^*$ accepted by M under the classical semantics of finite automata. By Kleene's theorem, $R(p) = R(M)$.

We claim that in both constructions, $G(p) = G(M)$ as well. To show this, it suffices to show that

$$G(M) = H(R(M)) \tag{1}$$
$$G(p) = H(R(p)). \tag{2}$$

The equation (1) is just the definition of acceptance for automata on guarded strings. For (2), it is easily shown that the map H is a homomorphism with respect to the operators $\cup$, $\circ$, and *. Moreover, the maps G and $H \circ R$ agree on the generators P and $\mathcal{B}$, since $H(R(d)) = H(\{d\}) = G(d)$ for $d \in \mathsf{P} \cup \mathcal{B}$, and $H(R(0)) = G(0) = \varnothing$. It follows by induction that G and $H \circ R$ agree on all regular expressions over $\mathsf{P} \cup \mathcal{B}$. ∎

3.2 Determinization

In this section we show how to construct a deterministic automaton on guarded strings equivalent to a given nondeterministic one. This is the basis of our *PSPACE* algorithm of Section 4.1. The construction is analogous to the standard subset construction for automata on ordinary strings (see e.g. [3]).

An automaton M on guarded strings is *deterministic* if it satisfies the following properties.

(i) There is exactly one start state.

(ii) Each state may have either exiting action transitions or exiting test transitions, but not both. A state is called an *action state* or a *test state* in these two circumstances, respectively. Every state is either an action state or a test state.

(iii) Every action state has exactly one exiting action transition for each element of P.

(iv) The labels of the exiting test transitions of a test state are pairwise exclusive and exhaustive. By this we mean that if the labels are $c_1, \ldots, c_n$, then $\overline{c}_i + \overline{c}_j$ for $i \neq j$ and $c_1 + \cdots + c_n$ are propositional tautologies.

(v) Every cycle contains at least one action transition.

(vi) All final states are action states.

Note that M is not a deterministic automaton in the classical sense. Conditions (i) and (iii) are standard for deterministic automata. Condition (ii) ensures that there is no ambiguity in whether to continue to test Boolean inputs or whether to read the next atomic program. Condition (iv) ensures that at any test state, exactly one exiting transition is enabled. Condition (v) ensures that there can be no endless loop of tests. Condition (vi) forces all pending tests to be resolved before deciding whether to accept the input.

Lemma 3.2 *For any $x \in \mathbf{GS}$ and state u of a deterministic AGS M, there is a unique maximal trace $\sigma_M(u,x)$ of M such that* $\mathbf{first}(\sigma_M(u,x)) = u$ *and* $x \in G(\mathbf{label}(\sigma_M(u,x)))$. *Moreover,* $\mathbf{last}(\sigma_M(u,x))$ *is an action state.*

Proof. This follows from the conditions of determinacy by induction on the length of x. ∎

We can convert a given nondeterministic automaton N to an equivalent deterministic automaton M by a subset construction. Suppose N has states Q, transition relation $\mathfrak{m}_N \subseteq Q \times (\mathsf{P} \cup \mathcal{B}) \times Q$, start states $S \subseteq Q$, and final states $F \subseteq Q$. Define M with states $Q' = 2^Q \times \{\mathbf{a}, \mathbf{t}\}$ and deterministic transition relation $\mathfrak{m}_M \subseteq Q' \times (\mathsf{P} \cup \mathcal{A}_\mathsf{B}) \times Q'$ as follows. The tags $\mathbf{a}, \mathbf{t}$ determine whether the state is an action or a test state, respectively.

$$\begin{aligned}(U,\mathbf{a}) \xrightarrow[M]{p} (V,\mathbf{t}) \quad &\stackrel{\text{def}}{\iff} \quad V = \{v \mid \exists u \in U\ u \xrightarrow[N]{p} v\} \\ &\qquad = \{\mathbf{last}(\tau) \mid \mathbf{first}(\tau) \in U,\ p = \mathbf{label}(\tau)\}\end{aligned}$$

$$\begin{aligned}(U,\mathbf{t}) \xrightarrow[M]{\alpha} (V,\mathbf{a}) \quad &\stackrel{\text{def}}{\iff} \quad V = \{\mathbf{last}(\pi) \mid \mathbf{first}(\pi) \in U,\ \alpha \le \mathbf{label}(\pi)\} \\ &\qquad = \{\mathbf{last}(\pi) \mid \mathbf{first}(\pi) \in U,\ \alpha \in G(\mathbf{label}(\pi))\},\end{aligned}$$

where τ and π represent traces of N, $p \in \mathsf{P}$, and $\alpha \in \mathcal{A}_\mathsf{B}$. The unique start state of M is $(S,\mathbf{t})$ and the final states are $\{(E,\mathbf{a}) \mid E \cap F \neq \varnothing\}$.

Thus $(U,\mathbf{a}) \xrightarrow[M]{p} (V,\mathbf{t})$ iff V is the set of states of N reachable from a state in U via a single transition with label p, and $(U,\mathbf{t}) \xrightarrow[M]{\alpha} (V,\mathbf{a})$ iff V is the set of states of N reachable from a state in U via a trace whose label is a sequence of tests, all of which are satisfied by α.

The automaton M constructed above is evidently deterministic. Moreover, since the graph of M is bipartite between action and test states, and since all test labels are atoms, the label of any trace consists of alternating atomic programs and atoms of B. Thus if $\mathbf{first}(\tau) = (U,\mathbf{t})$ and $\mathbf{last}(\tau) = (V,\mathbf{a})$, then $\mathbf{label}(\tau)$ begins and ends with an atom, so it is a guarded string. Since the start state is of the form $(S,\mathbf{t})$ and the final states are all of the form $(E,\mathbf{a})$, any string accepted by M is a guarded string, therefore $R(M) \subseteq \mathbf{GS}$.

It follows from these remarks and Lemma 3.2 that for all $x \in \mathbf{GS}$ and $U \subseteq Q$, the unique maximal trace $\sigma_M((U,\mathbf{t}),x)$ of M determined by $(U,\mathbf{t})$ and x not only has $x \in G(\mathbf{label}(\sigma_M((U,\mathbf{t}),x)))$, but actually $x = \mathbf{label}(\sigma_M((U,\mathbf{t}),x))$. Moreover, $\mathbf{last}(\sigma_M((U,\mathbf{t}),x))$ is of the form $(V,\mathbf{a})$. Let us denote by $\Delta(U,x)$ the set V uniquely determined by U and x in this way.

Lemma 3.3 *For all $x \in \mathbf{GS}$ and $U \subseteq Q$,*

$$\Delta(U,x) \quad = \quad \{\mathbf{last}(\sigma) \mid \mathbf{first}(\sigma) \in U,\ x \in G(\mathbf{label}(\sigma))\},$$

where σ ranges over traces of N.

Proof. We proceed by induction on the length of x. The basis $x = \alpha \in \mathcal{A}_\mathsf{B}$ is just the definition of $\xrightarrow[M]{\alpha}$. For x of the form $yp\alpha$, by the definition of $\xrightarrow[M]{\alpha}$ and $\xrightarrow[M]{p}$ and the induction hypothesis, we have

$$\begin{aligned}
&\Delta(U,x)\\
&= \{\mathbf{last}(\pi) \mid \mathbf{first}(\pi) \in \{\mathbf{last}(\tau) \mid \mathbf{first}(\tau) \in \Delta(U,y),\ p = \mathbf{label}(\tau)\},\\
&\qquad \alpha \in G(\mathbf{label}(\pi))\}\\
&= \{\mathbf{last}(\pi) \mid \mathbf{first}(\pi) \in \{\mathbf{last}(\tau) \mid \mathbf{first}(\tau) \in \{\mathbf{last}(\sigma) \mid \mathbf{first}(\sigma) \in U,\\
&\qquad y \in G(\mathbf{label}(\sigma))\},\ p = \mathbf{label}(\tau)\},\ \alpha \in G(\mathbf{label}(\pi))\}\\
&= \{s \mid \exists\sigma\ \exists\tau\ \exists\pi\ \ s = \mathbf{last}(\pi),\ \mathbf{first}(\pi) = \mathbf{last}(\tau),\ \mathbf{first}(\tau) = \mathbf{last}(\sigma),\\
&\qquad \mathbf{first}(\sigma) \in U,\ y \in G(\mathbf{label}(\sigma)),\ p = \mathbf{label}(\tau),\ \alpha \in G(\mathbf{label}(\pi))\}\\
&= \{s \mid \exists\xi\ \exists\sigma\ \exists\tau\ \exists\pi\ \ \xi = \sigma\tau\pi,\ s = \mathbf{last}(\xi),\ \mathbf{first}(\xi) \in U,\\
&\qquad y \in G(\mathbf{label}(\sigma)),\ p = \mathbf{label}(\tau),\ \alpha \in G(\mathbf{label}(\pi))\}\\
&= \{\mathbf{last}(\xi) \mid \mathbf{first}(\xi) \in U,\ \exists\sigma\ \exists\tau\ \exists\pi\ \ \xi = \sigma\tau\pi,\\
&\qquad y \in G(\mathbf{label}(\sigma)),\ p = \mathbf{label}(\tau),\ \alpha \in G(\mathbf{label}(\pi))\}\\
&= \{\mathbf{last}(\xi) \mid \mathbf{first}(\xi) \in U,\ yp\alpha \in G(\mathbf{label}(\xi))\}\\
&= \{\mathbf{last}(\xi) \mid \mathbf{first}(\xi) \in U,\ x \in G(\mathbf{label}(\xi))\}.
\end{aligned}$$

∎

Theorem 3.4 $G(M) = G(N)$.

Proof. We have argued that $R(M) \subseteq \mathbf{GS}$. Since $G(x) = \{x\}$ for guarded strings x, H is the identity on subsets of $\mathbf{GS}$, therefore

$$G(M) \quad = \quad H(R(M)) \quad = \quad R(M).$$

Now using Lemma 3.3,

$$\begin{aligned}
R(M) &= \{x \mid \Delta(S,x) \cap F \neq \varnothing\}\\
&= \{x \mid \{\mathbf{last}(\sigma) \mid \mathbf{first}(\sigma) \in S,\ x \in G(\mathbf{label}(\sigma))\} \cap F \neq \varnothing\}\\
&= \{x \mid \exists\sigma\ \mathbf{first}(\sigma) \in S,\ x \in G(\mathbf{label}(\sigma)),\ \mathbf{last}(\sigma) \in F\}\\
&= G(N).
\end{aligned}$$

∎

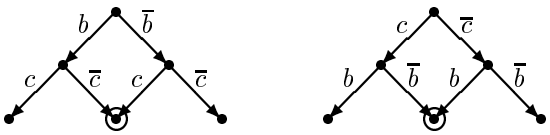
3.3 State Minimization

It turns out that the existence of unique minimal deterministic AGSs depends on the choice of input alphabet and restictions on how inputs can be read. We show in Section 3.3.2 that if any test in $\mathcal{B}$ is allowed as an input symbol, unique minimal deterministic AGSs exist. The test symbols of the minimal AGS can be taken to be the atoms of B. However, although the number of states is small, the specification of transitions may be exponential in the size of B.

A more reasonable choice of input alphabet for tests is $\mathsf{B} \cup \overline{\mathsf{B}}$. There is no loss of generality in this restriction, since all regular sets of guarded strings can still be represented, but the number of states may increase. Unfortunately, uniqueness of minimal automata is no longer guaranteed. However, if the automata are constrained to read their Boolean inputs in a given fixed order—such automata are called *ordered*—then minimal automata are unique. We show in Section 4.2 that the Canonicity Lemma for reduced ordered Boolean decision diagrams (ROBDDs) (see [1]) is a special case of this result.

3.3.1 Ordered AGSs

If we restrict the input alphabet to $\mathsf{P} \cup \mathsf{B} \cup \overline{\mathsf{B}}$, uniqueness of minimal deterministic automata is not guaranteed. For example, the automata



over $\mathsf{P} = \varnothing$ and $\mathsf{B} = \{b, c\}$ represent the same set of guarded strings $\{b\overline{c}, \overline{b}c\}$. However, uniqueness can be guaranteed provided we constrain the automata to test Boolean inputs in a particular fixed order, say $b_1, b_2, \ldots, b_k$. In such automata, each test state is assigned a *level* between 0 and $k-1$, inclusive. A test state at level i has one exiting transition labeled b_{i+1} and one labeled $\overline{b}_{i+1}$, and the transitions must go either to an action state or to a higher-level test state. Such an AGS is called *ordered*. We show that there is a unique minimal deterministic ordered AGS with respect to the given linear order on B. The construction is a generalization of the Myhill–Nerode construction of classical finite automata theory (see [3]).

A *prefix* of a guarded string is a string x such that $xy \in \mathbf{GS}$ for some string y. A prefix of a guarded string is just like a guarded string, except that $\mathbf{last}(x)$ may be a prefix of an atom; that is, a string of the form $c_1 \cdots c_j$, where $0 \leq j \leq k$ and $c_i \in \{b_i, \overline{b}_i\}$, $1 \leq i \leq j$. The set of all prefixes of guarded strings is denoted $\mathbf{PGS}$.

We define a binary operation $\triangleleft$ on $\mathbf{PGS}$ as follows. If $c_1 \cdots c_m$ and $d_1 \cdots d_n$ are prefixes of atoms, define

$$c_1 \cdots c_m \triangleleft d_1 \cdots d_n \stackrel{\text{def}}{=} \begin{cases} c_1 \cdots c_m, & \text{if } n \leq m \\ c_1 \cdots c_m d_{m+1} \cdots d_n, & \text{otherwise.} \end{cases}$$

Intuitively, we overlay $c_1 \cdots c_m$ on $d_1 \cdots d_n$, resolving disagreements in favor of the c_i. For $x, y \in \mathbf{PGS}$, we define $x \triangleleft y$ similarly to the fusion product xy, except that we perform $\triangleleft$ at the interface instead of fusing $\mathbf{last}(x)$ and $\mathbf{first}(y)$ as in the fusion product. Unlike fusion product, $\triangleleft$ is a total operation. For example,

$$\begin{aligned} b\,\overline{c}\,d\,p\,\overline{b}\,c \triangleleft b\,c\,\overline{d}\,q\,\overline{b}\,\overline{c} &= b\,\overline{c}\,d\,p\,\overline{b}\,c\,\overline{d}\,q\,\overline{b}\,\overline{c} \\ b\,\overline{c}\,d\,p\,\overline{b}\,\overline{c}\,d \triangleleft b\,c &= b\,\overline{c}\,d\,p\,\overline{b}\,\overline{c}\,d. \end{aligned}$$

It is easily shown that $\triangleleft$ is associative.

Now let $A \subseteq \mathbf{GS}$. For $x, y \in \mathbf{PGS}$, define the Myhill–Nerode relation

$$x \equiv y \iff \forall z \in \mathbf{GS}\ (x \triangleleft z \in A \Leftrightarrow y \triangleleft z \in A).$$

Lemma 3.5 *If $x, y \in \mathbf{PGS}$ and $x \equiv y$, then $x \triangleleft z \equiv y \triangleleft z$ for any $z \in \mathbf{PGS}$.*

Proof. Using the associativity of $\triangleleft$, for any $w \in \mathbf{GS}$, $(x \triangleleft z) \triangleleft w \in A$ iff $x \triangleleft (z \triangleleft w) \in A$ iff $y \triangleleft (z \triangleleft w) \in A$ iff $(y \triangleleft z) \triangleleft w \in A$. ∎

Lemma 3.6 *If $x, y \in \mathbf{GS}$ and $x \equiv y$, then $x \in A$ iff $y \in A$.*

Proof. For any atom α, $x \in A$ iff $x \triangleleft \alpha \in A$ iff $y \triangleleft \alpha \in A$ iff $y \in A$. ∎

Define $\mathbf{level}(x)$ to be the maximum value of $|\mathbf{last}(y)|$ over all $y \equiv x$, where $|\omega|$ is the length of ω. Then $0 \leq \mathbf{level}(x) \leq k$, and $\mathbf{level}(x) = k$ iff x is $\equiv$-equivalent to a guarded string.

We now build a deterministic ordered AGS N over $\mathsf{P} \cup \mathsf{B} \cup \overline{\mathsf{B}}$ from the equivalence classes of $\equiv$. Let

$$[\,x\,] \stackrel{\text{def}}{=} \{y \in \mathbf{PGS} \mid y \equiv x\}.$$

The states of N are $\{[\,x\,] \mid x \in \mathbf{PGS}\}$. A state $[\,x\,]$ is a test state if $0 \leq \mathbf{level}(x) \leq k-1$ and an action state if $\mathbf{level}(x) = k$. The transitions are

$$\left.\begin{array}{l} [\,x\,] \xrightarrow[N]{b_i} [\,xb_i\,] \\ [\,x\,] \xrightarrow[N]{\overline{b}_i} [\,x\overline{b}_i\,] \end{array}\right\} \quad \text{if } \mathbf{level}(x) = |\mathbf{last}(x)| = i - 1 < k$$

$$[\,x\,] \xrightarrow[N]{p} [\,xp\,] \qquad \text{if } \mathbf{level}(x) = |\mathbf{last}(x)| = k.$$

The start state of N is $[\varepsilon]$ and the final states are $\{[x] \mid x \in A\}$. The transitions are well defined by Lemma 3.5, and $x \in A$ iff $[x]$ is a final state by Lemma 3.6.

By Lemma 3.2, for any $x \in \mathbf{GS}$, there exists a unique maximal trace $\sigma_N([\varepsilon], x)$ such that $\mathbf{first}(\sigma_N([\varepsilon], x)) = [\varepsilon]$ and $x \in G(\mathbf{label}(\sigma_N([\varepsilon], x)))$. We will show in Lemma 3.10 that $\mathbf{last}(\sigma_N([\varepsilon], x)) = [x]$.

For any $x, y \in \mathbf{PGS}$, let $\sigma_N([x], y)$ be the greatest common prefix of the traces $\sigma_N([x], y \lhd \alpha)$ for all atoms α. We denote this by $\mathbf{lcp}_\alpha\ \sigma_N([x], y \lhd \alpha)$.

Lemma 3.7 *Let $x \in \mathbf{PGS}$ and let ω be a prefix of an atom. If $x \lhd \eta \equiv x \lhd \omega$ for all η such that $|\eta| = |\omega|$, then $x \equiv x \lhd \omega$.*

Proof. For any $z \in \mathbf{GS}$, let η_z be the prefix of $\mathbf{first}(z)$ of length $|\omega|$. Then

$$\begin{aligned} x \lhd z \in A &\Leftrightarrow x \lhd (\eta_z \lhd z) \in A \\ &\Leftrightarrow (x \lhd \eta_z) \lhd z \in A \\ &\Leftrightarrow (x \lhd \omega) \lhd z \in A. \end{aligned}$$

Since z was arbitrary, $x \equiv x \lhd \omega$. ∎

Lemma 3.8 *The two successors of any test state in N are distinct.*

Proof. Let $[x]$ be a test state. Assume without loss of generality that $|\mathbf{last}(x)| = \mathbf{level}([x]) = i - 1$. The exiting transitions are $[x] \xrightarrow{b_i} [xb_i]$ and $[x] \xrightarrow{\overline{b}_i} [x\overline{b}_i]$, and we must show that $xb_i \not\equiv x\overline{b}_i$. But if $xb_i \equiv x\overline{b}_i$, then by Lemma 3.7 we would have $x \equiv xb_i$, which would contradict the assumption that $|\mathbf{last}(x)| = \mathbf{level}([x])$. ∎

Lemma 3.9 *For all $x \in \mathbf{PGS}$, $\sigma_N([x], \varepsilon) = [x]$.*

Proof. We wish to show that $\mathbf{lcp}_\alpha\ \sigma_N([x], \alpha) = [x]$. If $[x]$ is an action state, then for all atoms α, $\sigma_N([x], \alpha) = [x]$, and we are done. If $[x]$ is a test state, then its two successors are distinct by Lemma 3.8. We can pick atoms α and β with opposite values for the $b \in \mathsf{B}$ tested at $[x]$, so the longest common prefix of $\sigma_N([x], \alpha)$ and $\sigma_N([x], \beta)$ is $[x]$. Thus $\sigma_N([x], \varepsilon) = \mathbf{lcp}_\alpha\ \sigma_N([x], \alpha) = [x]$. ∎

Lemma 3.10 *For all $x, y \in \mathbf{PGS}$, $\mathbf{last}(\sigma_N([x], y)) = [x \lhd y]$. In particular, $\mathbf{last}(\sigma_N([\varepsilon], x)) = [x]$.*

Proof. Assume without loss of generality that $|\mathbf{last}(x)| = \mathbf{level}(\texttt{[}x\texttt{]})$. First we show the result for $y = \omega$, a prefix of an atom. If $|\omega| \leq \mathbf{level}(\texttt{[}x\texttt{]})$, then

$$\begin{aligned}
\sigma_N(\texttt{[}x\texttt{]}, \omega) &= \operatorname*{\mathbf{lcp}}_{\alpha} \sigma_N(\texttt{[}x\texttt{]}, \omega \triangleleft \alpha) \\
&= \operatorname*{\mathbf{lcp}}_{\alpha} \sigma_N(\texttt{[}x\texttt{]}, \alpha) \\
&= \sigma_N(\texttt{[}x\texttt{]}, \varepsilon) \\
&= \texttt{[}x\texttt{]} \quad \text{by Lemma 3.9} \\
&= \texttt{[}x \triangleleft \omega\texttt{]}.
\end{aligned}$$

If $|\omega| > \mathbf{level}(\texttt{[}x\texttt{]})$, let $i = \mathbf{level}(\texttt{[}x\texttt{]}) + 1 \leq |\omega|$ and let $c \in \{b_i, \overline{b}_i\}$ such that $\omega \leq c$. For all atoms α,

$$\sigma_N(\texttt{[}x\texttt{]}, \omega \triangleleft \alpha) = (\texttt{[}x\texttt{]} \xrightarrow{c} \texttt{[}xc\texttt{]}) \cdot \sigma_N(\texttt{[}xc\texttt{]}, \omega \triangleleft \alpha),$$

thus

$$\begin{aligned}
\sigma_N(\texttt{[}x\texttt{]}, \omega) &= \operatorname*{\mathbf{lcp}}_{\alpha} \sigma_N(\texttt{[}x\texttt{]}, \omega \triangleleft \alpha) \\
&= (\texttt{[}x\texttt{]} \xrightarrow{c} \texttt{[}xc\texttt{]}) \cdot \operatorname*{\mathbf{lcp}}_{\alpha} \sigma_N(\texttt{[}xc\texttt{]}, \omega \triangleleft \alpha) \\
&= (\texttt{[}x\texttt{]} \xrightarrow{c} \texttt{[}xc\texttt{]}) \cdot \sigma_N(\texttt{[}xc\texttt{]}, \omega),
\end{aligned}$$

therefore

$$\begin{aligned}
\mathbf{last}(\sigma_N(\texttt{[}x\texttt{]}, \omega)) &= \mathbf{last}((\texttt{[}x\texttt{]} \xrightarrow{c} \texttt{[}xc\texttt{]}) \cdot \sigma_N(\texttt{[}xc\texttt{]}, \omega)) \\
&= \mathbf{last}(\sigma_N(\texttt{[}xc\texttt{]}, \omega)) \\
&= \texttt{[}xc \triangleleft \omega\texttt{]} \quad \text{by the induction hypothesis} \\
&= \texttt{[}x \triangleleft \omega\texttt{]}.
\end{aligned}$$

Finally, for $yp\omega \in \mathbf{PGS}$ where $y \in \mathbf{GS}$, $p \in \mathsf{P}$, and ω is a prefix of an atom, by the induction hypothesis we have $\mathbf{last}(\sigma_N(\texttt{[}x\texttt{]}, y)) = \texttt{[}x \triangleleft y\texttt{]}$. Then

$$\begin{aligned}
&\sigma_N(\texttt{[}x\texttt{]}, yp\omega) \\
&\quad = \operatorname*{\mathbf{lcp}}_{\alpha} \sigma_N(\texttt{[}x\texttt{]}, yp\omega \triangleleft \alpha) \\
&\quad = \operatorname*{\mathbf{lcp}}_{\alpha} (\sigma_N(\texttt{[}x\texttt{]}, y) \cdot (\texttt{[}x \triangleleft y\texttt{]} \xrightarrow{p} \texttt{[}x \triangleleft yp\texttt{]}) \cdot \sigma_N(\texttt{[}x \triangleleft yp\texttt{]}, \omega \triangleleft \alpha)) \\
&\quad = \sigma_N(\texttt{[}x\texttt{]}, y) \cdot (\texttt{[}x \triangleleft y\texttt{]} \xrightarrow{p} \texttt{[}x \triangleleft yp\texttt{]}) \cdot \operatorname*{\mathbf{lcp}}_{\alpha} \sigma_N(\texttt{[}x \triangleleft yp\texttt{]}, \omega \triangleleft \alpha) \\
&\quad = \sigma_N(\texttt{[}x\texttt{]}, y) \cdot (\texttt{[}x \triangleleft y\texttt{]} \xrightarrow{p} \texttt{[}x \triangleleft yp\texttt{]}) \cdot \sigma_N(\texttt{[}x \triangleleft yp\texttt{]}, \omega),
\end{aligned}$$

thus

$$\begin{aligned}
&\mathbf{last}(\sigma_N([x], yp\omega)) \\
&\quad = \mathbf{last}(\sigma_N([x], y) \cdot ([x \triangleleft y] \xrightarrow{p} [x \triangleleft yp]) \cdot \sigma_N([x \triangleleft yp], \omega)) \\
&\quad = \mathbf{last}(\sigma_N([x \triangleleft yp], \omega)) \\
&\quad = [(x \triangleleft yp) \triangleleft \omega] \qquad \text{by an earlier argument} \\
&\quad = [x \triangleleft yp\omega].
\end{aligned}$$

∎

Theorem 3.11 *Up to isomorphism, the automaton N constructed above is the unique minimal deterministic ordered AGS for A. Thus there are finitely many $\equiv$-classes iff A is regular.*

Proof. To show that $G(N) = A$, for any $x \in \mathbf{GS}$, $x \in G(N)$ iff $\mathbf{last}(\sigma_N([\varepsilon], x))$ is a final state of N. By Lemmas 3.6 and 3.10, this occurs iff $x \in A$.

For any other deterministic ordered AGS M for A, there is a surjective structure-preserving map from the accessible states of M to the states of N. For $x, y \in \mathbf{PGS}$, define $x \sim y$ if $\mathbf{last}(\sigma_M(s, x)) = \mathbf{last}(\sigma_M(s, y))$, where s is the start state of M. There is a one-to-one correspondence between the accessible states of M and the $\sim$-equivalence classes. Moreover, if $x \sim y$, then $x \triangleleft z \in A$ iff $y \triangleleft z \in A$ for any z, therefore $x \equiv y$. Thus $\sim$ refines $\equiv$. The desired map is $\mathbf{last}(\sigma_M(s, x)) \mapsto [x] = \mathbf{last}(\sigma_N([\varepsilon], x))$. ∎

3.3.2 Unrestricted Tests

If any test in $\mathcal{B}$ is allowed as an input symbol, we can adapt the construction of the previous section to give unique minimal deterministic automata.

Define $\mathbf{PGS}^i$ to be the set of prefixes x of guarded strings such that $|\mathbf{last}(x)| = i$. Then $\mathbf{PGS}^0 = \{xp \mid x \in \mathbf{GS},\ p \in \mathsf{P}\} \cup \{\varepsilon\}$ and $\mathbf{PGS}^k = \mathbf{GS}$. We define $\triangleleft$ and $\equiv$ as in Section 3.3.1 and take $\{[x] \mid x \in \mathbf{PGS}^0 \cup \mathbf{PGS}^k\}$ as states of our automaton. The equivalence classes $[x]$ for $x \in \mathbf{GS}$ are the action states. The remaining states are test states. The transitions are

$$[y] \xrightarrow{\alpha} [y\alpha] \qquad [x] \xrightarrow{p} [xp]$$

for $\alpha \in \mathcal{A}_{\mathsf{B}}$, $p \in \mathsf{P}$, $x \in \mathbf{GS}$, and $y \in \mathbf{PGS}^0$ such that $y \not\equiv z$ for any $z \in \mathbf{GS}$. The start state is $[\varepsilon]$ and the final states are $\{[x] \mid x \in A\}$. A direct adaptation of the arguments of Section 3.3.1 shows that there are finitely many $\equiv$-classes iff A is regular, and that this construction gives the minimal deterministic AGS for A over the alphabet $\mathsf{P} \cup \mathcal{B}$.

4 Applications

4.1 The Complexity of System S

System S, introduced in [7], is a Gentzen-style sequent calculus for partial correctness that subsumes propositional Hoare Logic. The system is a noncommutative intuitionistic Linear Logic. It was shown in [7] that the system is sound and complete over relational and trace-based models.

The syntax of system S is given in the following grammar. Here we revert to the notation of [7], in which $+$ is written as $\sqcup$, $\cdot$ as $\otimes$, and 0 as $\bot$. Also, the positive iteration operator $^+$ is taken as primitive, and * is defined by $p^* \stackrel{\text{def}}{=} 1 \sqcup p^+$. Note that there are two kinds of propositions, *tests* and *formulas*.

tests	$b, c, d \ldots$	$b ::= \langle\text{atomic tests}\rangle \mid \bot \mid b \to c$
programs	$p, q, r, \ldots$	$p ::= \langle\text{atomic programs}\rangle \mid b \mid p \sqcup q \mid p \otimes q \mid p^+$
formulas	$\varphi, \psi, \ldots$	$\varphi ::= b \mid p \to \varphi$
environments	$\Gamma, \Delta, \ldots$	$\Gamma ::= \varepsilon \mid \Gamma, p \mid \Gamma, \varphi$
sequents	$\Gamma \vdash \varphi$	

We abbreviate $b \to \bot$ by $\overline{b}$, $\overline{\bot}$ by 1, and $p \otimes q$ by pq.

A *formula* is either a test or an expression $p \to \varphi$, read "after p, φ," where p is a program and φ is a formula. Intuitively, the meaning is similar to the modal construct $[p]\varphi$ of Dynamic Logic (DL) (see [2]). The operator $\to$ associates to the right. The empty environment is denoted ε. Intuitively, an environment describes a previous computation that has led to the current state. *Sequents* are of the form $\Gamma \vdash \varphi$, where Γ is an environment and φ is a formula. We write $\vdash \varphi$ for $\varepsilon \vdash \varphi$. Intuitively, the meaning of $\Gamma \vdash \varphi$ is similar to the DL assertion $[\Gamma]\varphi$, where we think of the environment $\Gamma = \ldots, p, \ldots, \psi, \ldots$ as the rich-test program $\cdots ; p; \cdots ; \psi?; \cdots$ of DL.

It is shown in [7] how to encode propositional Hoare Logic (PHL). It follows from the completeness theorem of [7] that all relationally valid Hoare rules are derivable; this is false for PHL [5, 8].

Programs and tests are interpreted over Kripke frames K as described in Section 2. Additionally, we interpret formulas, environments, and sequents as follows:

$$\begin{aligned}
[\![p \to \varphi]\!]_K &\stackrel{\text{def}}{=} \{ s \mid \forall \tau \ \mathbf{first}(\tau) = s \text{ and } \tau \in [\![p]\!]_K \Rightarrow \mathbf{last}(\tau) \in [\![\varphi]\!]_K \} \\
[\![\varepsilon]\!]_K &\stackrel{\text{def}}{=} K \\
[\![\Gamma, \Delta]\!]_K &\stackrel{\text{def}}{=} [\![\Gamma]\!]_K \circ [\![\Delta]\!]_K .
\end{aligned}$$

The sequent $\Gamma \vdash \varphi$ is *valid* in the trace model K if for all traces $\sigma \in [\![\Gamma]\!]_K$, $\mathbf{last}(\sigma) \in [\![\varphi]\!]_K$; equivalently, if $[\![\Gamma]\!]_K \subseteq [\![\Gamma, \varphi]\!]_K$.

The rules of S are given in Fig. 1. It was shown in [7] that this system is sound and complete over trace models; that is, the sequent $\Gamma \vdash \varphi$ is valid in all trace models iff it is derivable in this deductive system.

A rule is *admissible* if for any substitution instance for which the premises are provable, the conclusion is also provable.

Lemma 4.1 *The operator $\otimes$ and rules* **(I $\otimes$)** *and* **(E $\otimes$)** *can be extended to pairs of formulas in the following sense: there exists a map $\varphi, \psi \mapsto \varphi \otimes \psi$ such that the rules*

$$\frac{\Gamma, \varphi, \psi, \Delta \vdash \rho}{\Gamma, \varphi \otimes \psi, \Delta \vdash \rho} \qquad\qquad \frac{\Gamma, \varphi \otimes \psi, \Delta \vdash \rho}{\Gamma, \varphi, \psi, \Delta \vdash \rho}$$

are admissible. We use **(I $\otimes$)** *and* **(E $\otimes$)***, respectively, to refer to these extended rules as well.*

Proof. If $\varphi = p_1 \to \cdots \to p_m \to b$ and $\psi = q_1 \to \cdots \to q_n \to c$, define

$$\varphi \otimes \psi \quad \stackrel{\text{def}}{=} \quad (p_1 \cdots p_m \overline{b} \sqcup q_1 \cdots q_n \overline{c}) \to \bot.$$

Using **(R $\to$)**, **(MP)**, and **(I $\otimes$)** and **(E $\otimes$)** on programs, it can be shown that

$$\begin{aligned} p_1 \to \cdots \to p_m \to b \quad &\vdash \quad p_1 \cdots p_m \overline{b} \to \bot \\ p_1 \cdots p_m \overline{b} \to \bot \quad &\vdash \quad p_1 \to \cdots \to p_m \to b. \end{aligned}$$

We also have

$$p \to \bot, q \to \bot \vdash p \sqcup q \to \bot \qquad p \sqcup q \to \bot \vdash p \to \bot \qquad p \sqcup q \to \bot \vdash q \to \bot$$

by the following arguments:

$$\dfrac{\dfrac{\dfrac{p \to \bot \vdash p \to \bot}{p \to \bot, q \to \bot, p \vdash \bot}\ \textbf{(MP), (W}\ \psi\textbf{)} \qquad \dfrac{q \to \bot \vdash q \to \bot}{p \to \bot, q \to \bot, q \vdash \bot}\ \textbf{(MP), (W}\ \psi\textbf{)}}{p \to \bot, q \to \bot, p \sqcup q \vdash \bot}\ \textbf{(I}\ \sqcup\textbf{)}}{p \to \bot, q \to \bot \vdash p \sqcup q \to \bot}\ \textbf{(R}\ \to\textbf{)}$$

$$\dfrac{\dfrac{\dfrac{p \sqcup q \to \bot \vdash p \sqcup q \to \bot}{p \sqcup q \to \bot, p \sqcup q \vdash \bot}\ \textbf{(MP)}}{\dfrac{p \sqcup q \to \bot, p \vdash \bot}{p \sqcup q \to \bot \vdash p \to \bot}\ \textbf{(R}\ \to\textbf{)} \qquad \dfrac{p \sqcup q \to \bot, q \vdash \bot}{p \sqcup q \to \bot \vdash q \to \bot}\ \textbf{(R}\ \to\textbf{)}}\ \textbf{(E1}\ \sqcup\textbf{), (E2}\ \sqcup\textbf{)}$$

It follows from **(cut)** that

$$\varphi, \psi \vdash \varphi \otimes \psi \qquad \varphi \otimes \psi \vdash \varphi \qquad \varphi \otimes \psi \vdash \psi.$$

The admissibility of the extended **(I $\otimes$)** and **(E $\otimes$)** then follows from **(cut)**. ∎

Axiom: $b \vdash c$ where $b \to c$ is a classical propositional tautology

Arrow Rules:

(R $\to$) $\dfrac{\Gamma, p \vdash \varphi}{\Gamma \vdash p \to \varphi}$ **(I $\to$)** $\dfrac{\Gamma, p, \psi, \Delta \vdash \varphi}{\Gamma, p \to \psi, p, \Delta \vdash \varphi}$

Introduction Rules:

(I $\otimes$) $\dfrac{\Gamma, p, q, \Delta \vdash \varphi}{\Gamma, p \otimes q, \Delta \vdash \varphi}$ **(I $\sqcup$)** $\dfrac{\Gamma, p, \Delta \vdash \varphi \qquad \Gamma, q, \Delta \vdash \varphi}{\Gamma, p \sqcup q, \Delta \vdash \varphi}$

(I $\bot$) $\Gamma, \bot, \Delta \vdash \varphi$ **(I $^+$)** $\dfrac{q \to \varphi, p \vdash \varphi \qquad q \to \varphi, q, q \vdash \varphi}{q \to \varphi, p^+ \vdash \varphi}$

Elimination Rules:

(E $\otimes$) $\dfrac{\Gamma, p \otimes q, \Delta \vdash \varphi}{\Gamma, p, q, \Delta \vdash \varphi}$ **(E $^+$)** $\dfrac{\Gamma, p^+, \Delta \vdash \varphi}{\Gamma, p, \Delta \vdash \varphi}$

(E1 $\sqcup$) $\dfrac{\Gamma, p \sqcup q, \Delta \vdash \varphi}{\Gamma, p, \Delta \vdash \varphi}$ **(E2 $\sqcup$)** $\dfrac{\Gamma, p \sqcup q, \Delta \vdash \varphi}{\Gamma, q, \Delta \vdash \varphi}$

Weakening Rules:

(W ψ) $\dfrac{\Gamma, \Delta \vdash \varphi}{\Gamma, \psi, \Delta \vdash \varphi}$ **(W p)** $\dfrac{\Gamma \vdash \varphi}{p, \Gamma \vdash \varphi}$ **(W $^+$)** $\dfrac{\Gamma, p^+, \Delta \vdash \varphi}{\Gamma, p^+, p^+, \Delta \vdash \varphi}$

Cut Rule:

(cut) $\dfrac{\Gamma \vdash \psi \qquad \Gamma, \psi, \Delta \vdash \varphi}{\Gamma, \Delta \vdash \varphi}$

Figure 1: Rules of System S [7]

Lemma 4.2 *Let* $w = (\bigsqcup \mathsf{P})^*$ *be an expression denoting all guarded strings. The sequent*

$$q_1 \to \bot, p_1, q_2 \to \bot, p_2, \ldots, q_n \to \bot, p_n \quad \vdash \quad \bot \tag{3}$$

is valid if and only if there do not exist guarded strings $x_1, \ldots, x_n$ *such that* $x_1 \cdots x_n$ *exists,* $x_i \in G(p_i)$*,* $1 \le i \le n$*, and* $x_i x_{i+1} \cdots x_n \notin G(q_1 w)$*,* $1 \le i \le n$*.*

Proof. Suppose such $x_1, \ldots, x_n$ exist. Let Γ be the environment on the left-hand side of (3). Construct a trace model K consisting of a single linear trace σ such that $\mathrm{gs}(\sigma) = x_1 \cdots x_n$. The model is uniquely determined by this specification. Let σ_i, $1 \le i \le n$, be the unique subtraces of σ such that $\mathrm{gs}(\sigma_i) = x_i$ and $\sigma = \sigma_1 \cdots \sigma_n$. By Lemma 2.2, $\sigma_i \in [\![p_i]\!]_K$. Since $x_i x_{i+1} \cdots x_n \notin G(q_i w)$, no prefix of $x_i x_{i+1} \cdots x_n$ is in $G(q_i)$, so by Lemma 2.2 no prefix of $\sigma_i \sigma_{i+1} \cdots \sigma_n$ is in $[\![q_i]\!]_K$. Since these are the only traces in K with initial state $\mathbf{first}(\sigma_i)$, we have $\mathbf{first}(\sigma_i) \in [\![q_i \to \bot]\!]_K$, therefore $\sigma_i \in [\![q_i \to \bot, p_i]\!]_K$. It follows that $\sigma \in [\![\Gamma]\!]_K$. Since $[\![\Gamma]\!]_K$ is nonempty, (3) is not valid.

Conversely, suppose (3) is not valid. Let K be a trace model and σ a trace in K such that $\sigma \in [\![\Gamma]\!]_K$. There exist subtraces σ_i in K, $1 \le i \le n$, such that $\sigma = \sigma_1 \cdots \sigma_n$ and $\sigma_i \in [\![q_i \to \bot, p_i]\!]_K$. Then $\mathbf{first}(\sigma_i) \in [\![q_i \to \bot]\!]_K$, so no prefix of $\sigma_i \sigma_{i+1} \cdots \sigma_n$ is in $[\![q_i]\!]_K$, and $\sigma_i \in [\![p_i]\!]_K$. Let $x_i = \mathrm{gs}(\sigma_i)$. By Lemma 2.2, no prefix of $x_i x_{i+1} \cdots x_n$ is in $G(q_i)$, therefore $x_i x_{i+1} \cdots x_n \notin G(q_i w)$, and $x_i \in G(p_i)$, $1 \le i \le n$. ∎

Theorem 4.3 *The problem of deciding whether a given sequent of System* S *is valid is PSPACE-complete.*

Proof. As observed in [7], the problem encodes the equivalence problem for regular expressions, a well-known *PSPACE*-complete problem [9], therefore is *PSPACE*-hard. It thus remains to show that the problem is in *PSPACE*.

Suppose we are given a sequent of the form

$$q_1 \to \bot, p_1, q_2 \to \bot, p_2, \ldots, q_n \to \bot, p_n \quad \vdash \quad \bot.$$

Using the extended **(I** $\otimes$**)** and **(E** $\otimes$**)** of Lemma 4.1 along with **(MP)** and **(R** $\to$**)**, we can transform any given sequent to one of this form with no significant increase in size, so the assumption is without loss of generality.

Now build nondeterministic automata M_i from the p_i and N_i from the $q_i w$ as in Theorem 3.1, where $w = (\bigsqcup \mathsf{P})^*$ is an expression representing all guarded strings. Our *PSPACE* algorithm will guess guarded strings $x_1, \ldots, x_n$ symbol by symbol in that order, scanning the automata to check the positive and negative conditions of Lemma 4.2. We must check that the automata M_i accept the x_i and the automata N_i reject

$x_i x_{i+1} \cdots x_n$. This is done by simulating the subset construction of Section 3.2 with pebbles occupying the states of the M_i and N_i. After every guessed atomic program or atom, the pebbles are moved according to the transitions of the deterministic automata constructed in Section 3.2. Guessing an atom amounts to guessing a truth assignment to B; then to determine whether a test transition is enabled, we just evaluate the label on that truth assignment. We also guess the boundaries between the x_i and x_{i+1} and make sure that $\mathbf{last}(x_i) = \mathbf{first}(x_{i+1})$. The entire simulation can be done in *PSPACE*. It does not matter how long the strings x_i are; the simulation continues to guess symbols until it succeeds (or not). This gives a nondeterministic *PSPACE* algorithm, which can be made deterministic using Savitch's theorem. ∎

4.2 Boolean Decision Diagrams

We refer the reader to Andersen's lecture notes [1] for an introduction to BDDs. An BDD is *ordered* (OBDD) if the order of the tests along any path is consistent with a given linear order on B. An OBDD is *reduced* (ROBDD) if (i) no two nodes that test the same Boolean variable have the same $\mathbf{true}$ successors and the same $\mathbf{false}$ successors, and (ii) the $\mathbf{true}$ and $\mathbf{false}$ successors of any node are distinct. The Canonicity Lemma ([1, p. 13]) says that any Boolean function has a unique ROBDD for a given linear order on B. The next theorem shows that the Canonicity Lemma is essentially Theorem 3.11 in the special case $\mathsf{P} = \varnothing$.

Theorem 4.4 *Let $\mathsf{P} = \varnothing$. For any $A \subseteq \mathcal{A}_{\mathsf{B}}$, the minimal ordered AGS for A for a given order on B constructed in Section 3.3.1 is the canonical ROBDD for $\bigvee A$ with respect to that order.*

Proof. The AGS is apparently an OBDD for $\bigvee A$. It therefore remains to check conditions (i) and (ii).

For condition (i), suppose $\mathbf{level}([x]) = \mathbf{level}([y]) = i-1$. Assume without loss of generality that $|\mathbf{last}(x)| = |\mathbf{last}(y)| = i-1$. If $[xb_i] = [yb_i]$ and $[x\overline{b}_i] = [y\overline{b}_i]$, then $xb_i \equiv yb_i$ and $x\overline{b}_i \equiv y\overline{b}_i$. Let $z \in \mathbf{GS}$ be arbitrary. If b_i occurs positively in $\mathbf{first}(z)$, then $x \lhd z \in A$ iff $xb_i \lhd z \in A$ iff $yb_i \lhd z \in A$ iff $y \lhd z \in A$. Similarly, if b_i occurs negatively in $\mathbf{first}(z)$, then $x \lhd z \in A$ iff $x\overline{b}_i \lhd z \in A$ iff $y\overline{b}_i \lhd z \in A$ iff $y \lhd z \in A$. Thus $x \equiv y$ and $[x] = [y]$.

Condition (ii) is just Lemma 3.8. ∎

Acknowledgements

I am indebted to Jerzy Tiuryn for many valuable ideas and engaging discussions. This work grew out of joint work with him [8, 7]. The support of the National Science

Foundation under grant CCR-9708915 is gratefully acknowledged.

References

[1] Henrik Reif Andersen. An introduction to binary decision diagrams. Lecture notes, Department of Information Technology, Technical University of Denmark, Copenhagen. http://www.itu.dk/people/hra/notes-index.html, April 1998.

[2] David Harel, Dexter Kozen, and Jerzy Tiuryn. *Dynamic Logic*. MIT Press, Cambridge, MA, 2000.

[3] Dexter Kozen. *Automata and Computability*. Springer-Verlag, New York, 1997.

[4] Dexter Kozen. Kleene algebra with tests. *Transactions on Programming Languages and Systems*, 19(3):427–443, May 1997.

[5] Dexter Kozen. On Hoare logic and Kleene algebra with tests. *Trans. Computational Logic*, 1(1):60–76, July 2000.

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

[7] Dexter Kozen and Jerzy Tiuryn. Intuitionistic linear logic and partial correctness. Technical Report 2000-1830, Computer Science Department, Cornell University, December 2000.

[8] Dexter Kozen and Jerzy Tiuryn. On the completeness of propositional Hoare logic. In J. Desharnais, editor, *Proc. 5th Int. Seminar Relational Methods in Computer Science (RelMiCS 2000)*, pages 195–202, January 2000.

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