JOURNAL OF COMPUTER AND SYSTEM SCIENCES 18, 194–211 (1979)

Propositional Dynamic Logic of Regular Programs*†

MICHAEL J. FISCHER AND RICHARD E. LADNER

Department of Computer Science, University of Washington, Seattle, Washington 98195

Received October 4, 1977; revised September 11, 1978

We introduce a fundamental propositional logical system based on modal logic for describing correctness, termination and equivalence of programs. We define a formal syntax and semantics for the *propositional dynamic logic of regular programs* and give several consequences of the definition. Principal conclusions are that deciding satisfiability of length n formulas requires time $d^{n/\log n}$ for some $d > 1$, and that satisfiability can be decided in nondeterministic time c^n for some c. We provide applications of the decision procedure to regular expressions, Ianov schemes, and classical systems of modal logic.

1. INTRODUCTION

Pratt [19] in conjunction with R. Moore has introduced a logical framework for programs based on modal logic. Their idea is to integrate programs into an assertion language by allowing programs to be modal operators. For instance, if a is a (possibly nondeterministic) program and p an assertion, then a new assertion, $\langle a \rangle p$ can be made. Informally, the meaning of "$\langle a \rangle p$" is "a can terminate with p holding on termination." In addition to modal operators, $\langle a \rangle$, for each program a, the usual Boolean operations and quantification are allowed. A dual modal operator $[a]$ is defined by $[a]p \equiv \sim\langle a \rangle \sim p$. The meaning of "$[a]p$" is "whenever a terminates p holds on termination." Following Harel, Meyer, and Pratt, we call such a system *dynamic logic* [8].

Dynamic logic provides a powerful language for describing programs, their correctness and termination. For example, the Hoare assertion "$p\{a\}q$" [10] can be expressed as "$p \supset [a]q$." The fact that a can terminate can be expressed by the assertion "$\langle a \rangle$ *true*." The determinacy of a program a can be expressed by the formula "$\langle a \rangle p \supset [a]p$," where p expresses the condition of determinacy.

One goal in developing a logic of programs is to provide a set of axioms and rules of inference for proving things about programs like "partial correctness," "termination," and "equivalence." One would expect that the things proved by the axioms and rules were at least "true." Hence, it is fundamental that there be a notion of "truth," that is,

* This research was supported in part by the National Science Foundation through Grant Nos. DCR74-12997-A01, GJ-43264, and MCS77-02474.

† An earlier version of this paper was presented at the Ninth ACM Symposium on Theory of Computing, Boulder, Colorado, May 2–4, 1977, under the title, "Propositional Model Logic of Programs."

0022-0000/79/020194-18$02.00/0

a semantics for the logic of programs. In the case of dynamic logic, a semantics must be provided for both the programs and for the formulas that talk about programs. The program semantics is derived from the relational semantics of programs (cf. Hoare and Lauer [11]) and the formula semantics is adopted from the relational semantics for modal logic introduced by Kripke [14].

Informally, each program a defines a relation $\rho(a)$ between program states: $(s, t) \in \rho(a)$ if and only if a executed in state s can terminate in state t. The truth of an assertion is determined relative to a program state, so we say "p is true in state s." The formula $\langle a \rangle p$ is true in state s if there is a state t such that $(s, t) \in \rho(a)$ and p is true in state t. The formula $p \vee q$ is true in state s if either p is true in state s or q is true in state s.

The system we introduce is an abstraction of the system introduced by Pratt. Pratt's basic programs are assignments and tests while our basic programs are uninterpreted labels. Pratt's formulas allow first order variables and quantification, while our formulas only allow propositional variables. Propositional dynamic logic of regular programs plays a role in the logic of programs analogous to the role the propositional calculus plays in the classical first-order logic.

The goal of this paper is to provide a mathematical definition of the syntax and semantics of propositional dynamic logic and to prove some fundamental consequences of this definition. In Section 2 we give the formal definitions and some examples. In Section 3 we show that satisfiability in propositional dynamic logic of regular programs is decidable in nondeterministic time c^n for some c. In Section 4 we show that deciding satisfiability requires deterministic time $d^{n/\log n}$ for some $d > 1$. In Section 5 we give applications to regular expressions, Ianov schemes, and classical modal logic.

2. The Formal System

We now define the syntax of the *propositional dynamic logic of regular programs*, PDL for short. There are two underlying sets of symbols: Φ_0, a set of *atomic formulas* which are propositional variables, and Σ_0, a set of *atomic programs* which can be thought of as indivisible statements in a programming language.

We inductively define the set of programs, Σ, and the set of formulas, Φ, by the following rules.

Programs:

(i) Atomic programs and θ are programs;

(ii) if a and b are programs and p is a formula, then $(a; b)$, $(a \cup b)$, a^*, and $p?$ are programs.

Formulas:

(i) Atomic formulas, *true* and *false*, are formulas;

(ii) if p and q are formulas and a is a program, then $(p \vee q)$, $\sim p$, and $\langle a \rangle p$ are formulas.

We normally reserve $P, Q, R, \ldots$ for members of Φ_0 ; $A, B, C, \ldots$ for members of Σ_0 . The letters $p, q, r, \ldots$ serve as metavariables for formulas and $a, b, c, \ldots$ serve as metavariables for programs. We call our programs *regular programs* because of their similarity to regular expressions. Regular programs can be thought of as abstractions of nondeterministic structured programs under the correspondence:

"θ" means "abort" or "blocked,"
"a; b" means "begin a; b end,"
"$a \cup b$" means "nondeterministically do a or do b,"
"a^*" means "repeat a a nondeterministically chosen number of times,"
"$p?$" means "test p and proceed only if true."

Note that $p?$ does not result in a state change if p is true, but the program is blocked if p is false. This meaning is similar to the semantics of Dijkstra's guarded commands [5].

We define the Boolean connectives $\wedge$, $\supset$, $\equiv$ in the usual way from $\vee$ and $\sim$. The dual operator $[a]$ is an abbreviation for $\sim\langle a\rangle\sim$. λ is an abbreviation for θ^*, the null program. We also define standard block structured programming constructs:

"*if* p *then* a *else* b" means "$p?; a \cup \sim p?; b$"
"*while* p *do* a" means "$(p?; a)^*; \sim p?$"

Although the formal syntax is fully parenthetical, we will commonly drop parentheses for readability. Thinking of $\langle a\rangle$ and $[a]$ as unary operators on formulas, the precedence of operators from highest to lowest is $\langle a\rangle$, $[a]$, $\sim$, $\wedge$, $\vee$, $\supset$, $\equiv$, $?$, *, $;$, $\cup$.

We now define the semantics of the propositional dynamic logic of regular programs. A *structure* (or *model*) $\mathscr{A}$ is a triple $(W^{\mathscr{A}}, \pi^{\mathscr{A}}, \rho^{\mathscr{A}})$, where

$$\pi^{\mathscr{A}}: \Phi_0 \to 2^{W^{\mathscr{A}}}$$
$$\rho^{\mathscr{A}}: \Sigma_0 \to 2^{W^{\mathscr{A}} \times W^{\mathscr{A}}}$$

Informally, $W^{\mathscr{A}}$ is a set of program states. The function $\pi^{\mathscr{A}}$ provides an interpretation for the atomic formulas: "$w \in \pi^{\mathscr{A}}(P)$" means "$P$ is true in the state w." The function $\rho^{\mathscr{A}}$ provides an interpretation for the atomic programs: "$(u, v) \in \rho^{\mathscr{A}}(A)$" means "there is an execution of A which begins in state u and ends in state v."

We extend $\rho^{\mathscr{A}}$ to all programs and $\pi^{\mathscr{A}}$ to all formulas inductively (we drop the superscript when there is no ambiguity):

$\rho(\theta) = \phi$,
$\rho(a; b) = \rho(a) \circ \rho(b)$ (composition of relations),
$\rho(a \cup b) = \rho(a) \cup \rho(b)$ (union of relations),
$\rho(a^*) = \rho(a)^*$ (reflexive and transitive closure of a relation),
$\rho(p?) = \{(w, w): w \in \pi(p)\}$,
$\pi(\textit{true}) = W$,
$\pi(\textit{false}) = \phi$,
$\pi(p \vee q) = \pi(p) \cup \pi(q)$,
$\pi(\sim p) = W - \pi(p)$,
$\pi(\langle a\rangle p) = \{w \in W: \exists v\ ((w, v) \in \rho(a) \text{ and } v \in \pi(p))\}$.

By the definition of [a] we can compute

$$\pi([a]p) = \{w \in W\colon \forall v\ ((w, v) \in \rho(a) \text{ implies } v \in \pi(p))\}.$$

These extensions are natural so that "$(u, v) \in \rho(a)$" means "the program a can take state u to state v" and "$w \in \pi(p)$" means "p is true in state w."

Using more standard semantic notation, we write $\mathscr{A}, w \models p$ just in case $w \in \pi^{\mathscr{A}}(p)$. We say p is *valid* if for all $\mathscr{A}$ and $w \in W^{\mathscr{A}}$, $\mathscr{A}, w \models p$. Further p is *satisfiable* if there is a structure $\mathscr{A}$ and a $w \in W^{\mathscr{A}}$ such that $\mathscr{A}, w \models p$. Clearly p is valid if and only if $\sim p$ is not satisfiable.

We give two examples of structures, the first fairly complex and the second simple.

EXAMPLE 1. Let $\mathbb{N} = \{0, 1, 2, \ldots\}$ and let V be a set of first-order variables.

$$\Phi_0 = \{x = y,\ x = 0,\ x = y + 1,\ x = y \dot{-} 1,\ x = y + z\colon x, y, z \in V\},$$
$$\Sigma_0 = \{x \leftarrow \text{random},\ x \leftarrow y + 1,\ x \leftarrow y \dot{-} 1\colon x, y \in V\}.$$

Consider the following structure $\mathscr{A} = (W, \pi, \rho)$:

$W = V \to \mathbb{N}$ (the set of assignments of the variables in $\mathbb{N}$),
$\pi(x = y) = \{s\colon s(x) = s(y)\}$,
$\pi(x = 0) = \{s\colon s(x) = 0\}$,
$\pi(x = y + 1) = \{s\colon s(x) = s(y) + 1\}$,
$\pi(x = y \dot{-} 1) = \{s\colon s(x) = s(y) \dot{-} 1\}$ ($0 \dot{-} 1 = 0$ by def),
$\pi(x = y + z) = \{s\colon s(x) = s(y) + s(z)\}$,
$\rho(x \leftarrow \text{random}) = \{(s, t)\colon s(y) = t(y) \text{ for all } y \neq x\}$,
$\rho(x \leftarrow y + 1) = \{(s, t)\colon t(x) = s(y) + 1 \text{ and } t(z) = s(z) \text{ for all } z \neq x\}$,
$\rho(x \leftarrow y \dot{-} 1) = \{(s, t)\colon t(x) = s(y) \dot{-} 1 \text{ and } t(z) = s(z) \text{ for all } z \neq x\}$.

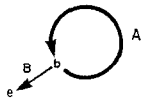
The meaning of the program "$x \leftarrow$ random" is "nondeterministically set x to any value." Hence, the meaning of "$\langle x \leftarrow \text{random}\rangle$" is "$\exists x$."

Let $a =^{\text{df}} (\sim w = 0\,?; (w \leftarrow w \dot{-} 1); (z \leftarrow z + 1))^*; (w = 0\,?)$. Let $p =^{\text{df}} (w = x \wedge z = y) \supset [a](z = x + y)$. We leave it to the reader to verify that for any $s \in W$, $\mathscr{A}, s \models p$. Informally, p states that a is a program that correctly computes addition.

EXAMPLE 2. Consider the structure $\mathscr{B}$:

$W = \{b, e\}$,
$\pi = \phi$,
$\rho(A) = \{(b, b)\}$, $\quad \rho(B) = \{(b, e)\}$.

Graphically:



We have $\mathscr{B}, b \models [A^*]$ ($\langle A\rangle$ *true* $\wedge$ $\langle B\rangle[A \cup B]$ *false*).

We next give some sample validities of PDL.

EXAMPLE 3.

(1) $\langle a; b\rangle p \equiv \langle a\rangle\langle b\rangle p$,

(2) $\langle a \cup b\rangle p \equiv \langle a\rangle p \vee \langle b\rangle p$,

(3) $\langle a^*\rangle p \equiv p \vee \langle a\rangle\langle a^*\rangle p$,

(4) $\langle p?\rangle q \equiv p \wedge q$,

(5) $\langle a\rangle(p \vee q) \equiv \langle a\rangle p \vee \langle a\rangle q$,

(6) $\langle a^*\rangle \sim p \equiv \langle$while p do $a\rangle$ *true*.

(This says that "while p do a" can terminate iff it is possible, by repeated executions of a, to reach a state in which $\sim p$ holds.)

(7) $\langle a\rangle p \supset \langle$if $\langle a\rangle p$ then a else $b\rangle p$.

3. UPPER BOUNDS ON THE COMPLEXITY OF PDL

Propositional dynamic logic is the basic logical framework for program correctness. Validities in PDL represent universal or logical truths. They may be thought of as "contentless" assertions since they do not depend on the meanings of the basic assertions or the basic programs.

In this section, we show that the complexity of the validity problem for PDL is in co-NTIME(c^n) for some c, where n is the size of the formula being tested. (That is, the complement of the validity problem for PDL is recognizable by a nondeterministic Turing machine in time $\leqslant c^n$.) This compares to classical propositional logic whose validity problem is in co-NP. In Section 4, we show that the validity problem is not in DTIME($c^{n/\log n}$) for some $c > 1$.

Let *size*(p) denote the length of p regarded as a string over $\Phi_0 \cup \Sigma_0 \cup \{\langle\ ,\ \rangle, \cup, ;, *, ?, \sim, \vee, \theta, (\ ,\ ),$ *true*, *false*$\}$. Define *size*($\mathscr{A}$) to be $|\ W^{\mathscr{A}}\ |$. If Φ_0 or Σ_0 is infinite then the size of p is the length of a string in an infinite alphabet. In order to realize formulas in a finite alphabet we assume $\Phi_0 \subseteq P \cdot \{0, 1\}^*$ and $\Sigma_0 \subseteq A \cdot \{0, 1\}^*$. When we speak of the *length* of a formula p we mean its length over the alphabet $\{P, A, 0, 1, \langle\ ,\ \rangle, \cup, ;, *, ?, \sim, \vee, \theta, (\ ,\ ),$ *true*, *false*$\}$. We denote the length of p by $l(p)$, in fact, if x is any word in a finite alphabet we let $l(x)$ denote its length.

For technical reasons, it is more convenient to treat the satisfiability problem for PDL, that is, given a formula p, to determine if p is satisfiable in some world w of some structure $\mathscr{A}$. We will show:

THEOREM 3.1. *The satisfiability problem for* PDL *is in* NTIME(c^n) *for some constant* c, *where* n *is the size of the formula.*

The result for validity then follows from the fact that p is valid iff $\sim p$ is not satisfiable.

COROLLARY. *The validity problem for* PDL *is in* co-NTIME(c^n) *for some constant* c, *where* n *is the size of the formula.*

(Note that the theorem and its corollary also hold if n is interpreted as the length of the formula instead of the size since size(p) $\leqslant l(p)$).

The proof of Theorem 3.1 depends on two key lemmas. (1) If p is satisfiable, then p is satisfiable in a model of exponential size. (2) The problem of testing whether a formula p is true at a state w in a structure $\mathscr{A}$ can be decided in time polynomial in the sizes of p and $\mathscr{A}$ (given suitable encodings). A nondeterministic algorithm for testing satisfiability is then simply:

ALGORITHM S. To test p of size n for satisfiability:

(1) Guess a structure $\mathscr{A}$ of size at most c^n.

(2) Guess a world $w \in W^{\mathscr{A}}$.

(3) Test if p holds at w in $\mathscr{A}$. If so, answer "yes."

The time for algorithm S is polynomial in c^n and hence is bounded by $(c')^n$ for some new constant c'.

THEOREM 3.2 (Small Model Theorem). *Let* p *be a satisfiable formula. Then there exists a structure* $\mathscr{A}$ *and a world* $w \in W^{\mathscr{A}}$ *such that* $\mathscr{A}, w \models p$ *and* size($\mathscr{A}$) $\leqslant 2^{\text{size}(p)}$.

Proof. Assume $\mathscr{A}_0, w_0 \models p_0$, that is, p_0 is a formula which is satisfied at w_0 in $\mathscr{A}_0$. The structure $\mathscr{A}_0$ may be finite or infinite. There are two phases in the construction of a small structure satisfying p_0. In the first phase we generate from p_0 a set of formulas S. Some of the formulas of S may contain new atomic formulas which we call Q-variables. From $\mathscr{A}_0$ we define an expanded structure $\mathscr{A}$ which has the same states as $\mathscr{A}_0$ but has added meanings for the new Q-variables. In the second phase we use S to define an equivalence relation between the states of $\mathscr{A}$. We then define a "quotient" model $\bar{\mathscr{A}}$ whose states are the equivalence classes of the states of $\mathscr{A}$. We show that $\bar{\mathscr{A}}$ has small size and $\bar{\mathscr{A}}, \bar{w}_0 \models p_0$ where $\bar{w}_0$ is the equivalence class of w_0.

Let F be a set of formulas and $\mathscr{B}_0$ be a structure, both over Φ_0 and Σ_0. We simultaneously define the *closure of* F, cl(F), and the *closure* $\mathscr{B}$ *of* $\mathscr{B}_0$ *with respect to* F inductively using the rules:

1. $F \subseteq \text{cl}(F)$, $W^{\mathscr{B}} = W^{\mathscr{B}_0}$, $\pi^{\mathscr{B}}(P) = \pi^{\mathscr{B}_0}(P)$ for all $P \in \Phi_0$, $\rho^{\mathscr{B}}(A) = \rho^{\mathscr{B}_0}(A)$ for all $A \in \Sigma_0$,

2. (a) $p \vee q \in \text{cl}(F) \Rightarrow p, q \in \text{cl}(F)$,
   (b) $\sim p \in \text{cl}(F) \Rightarrow p \in \text{cl}(F)$,
   (c) $\langle A \rangle p \in \text{cl}(F) \Rightarrow p \in \text{cl}(F)$ for all $A \in \Sigma_0 \cup \{\theta\}$,
   (d) $\langle q? \rangle p \in \text{cl}(F) \Rightarrow p, q \in \text{cl}(F)$,
   (e) $\langle a; b \rangle p \in \text{cl}(F) \Rightarrow \langle a \rangle Q^{\langle b \rangle p}, \langle b \rangle p \in \text{cl}(F)$, and $\pi^{\mathscr{B}}(Q^{\langle b \rangle p}) = \pi^{\mathscr{B}}(\langle b \rangle p)$,
   (f) $\langle a \cup b \rangle p \in \text{cl}(F) \Rightarrow \langle a \rangle Q^p, \langle b \rangle Q^p, p \in \text{cl}(F)$ and $\pi^{\mathscr{B}}(Q^p) = \pi^{\mathscr{B}}(p)$,
   (g) $\langle a^* \rangle p \in \text{cl}(F) \Rightarrow p, \langle a \rangle Q^{\langle a^* \rangle p} \in \text{cl}(F)$ and $\pi^{\mathscr{B}}(Q^{\langle a^* \rangle p}) = \pi^{\mathscr{B}}(\langle a^* \rangle p)$.

The new Q-variables are given meaning in $\mathscr{B}$ when they are introduced into the closure. The definition of $\pi^{\mathscr{B}}(Q^r)$ depends only on $\pi^{\mathscr{B}}(r)$ which has been already well defined previously. The new structure $\mathscr{B}$ is over Φ_0' and Σ_0, where $\Delta_0 = \Phi_0' - \Phi_0$ is the set of Q-variables introduced in taking the closure of F.

At this point it is helpful to explain the role of the Q-variables in the proof. First, their presence will allow us to argue that the cardinality of the closure of p_0 is linear in the size of p_0. Second, they aid in our induction proof of Claim 2 by providing a base for the induction. This simplifies an earlier version of the proof which had two separate inductions (and which did not handle "?").

Each rule (a)–(g) has as premise a single formula in $\mathrm{cl}(F)$ so $\mathrm{cl}(F_1 \cup F_2) = \mathrm{cl}(F_1) \cup \mathrm{cl}(F_2)$ for any sets F_1 and F_2. It follows that $\mathrm{cl}(F) = \bigcup_{p \in F} \mathrm{cl}(\{p\})$. Hence $|\mathrm{cl}(F)| \leqslant \sum_{p \in F} |\mathrm{cl}(\{p\})|$.

To analyze $|\mathrm{cl}(\{p\})|$, let $\gamma(p)$ be the number of occurences in p of symbols in $\{\vee, \sim, ?, ;, \cup, {}^*\} \cup \Phi_0 \cup \Sigma_0 \cup \{\theta, \mathit{true}, \mathit{false}\}$. Note that γ does not count the new Q-variables. With this definition it happens that each rule (a)–(g) is of the form

$$p \in \mathrm{cl}(F) \Rightarrow p_1, \ldots, p_k \in \mathrm{cl}(F), \tag{1}$$

where $\gamma(p) = 1 + \gamma(p_1) + \cdots + \gamma(p_k)$. Further, if a rule (1) is applicable to a formula p, it is the only such rule so that

$$\mathrm{cl}(\{p\}) = \{p\} \cup \bigcup_{i=1}^{k} \mathrm{cl}(\{p_i\}),$$

and if no rule (1) is applicable, then $\mathrm{cl}(\{p\}) = \{p\}$. It can be easily verified by induction on $\gamma(p)$ that

$$|\mathrm{cl}(\{p\}) - \Delta_0| \leqslant \gamma(p). \tag{2}$$

Two other useful facts can be easily verified:

$$\langle a \rangle p \in \mathrm{cl}(F) \Rightarrow p \in \mathrm{cl}(F), \tag{3}$$

$$Q^p \in \mathrm{cl}(F) \Rightarrow p \in \mathrm{cl}(F) \quad \text{and} \quad \pi^{\mathscr{B}}(Q^p) = \pi^{\mathscr{B}}(p). \tag{4}$$

Let $S = \mathrm{cl}(\{p_0\})$ and let $\mathscr{A}$ be the closure of $\mathscr{A}_0$ with respect to $\{p_0\}$. Define the equivalence relation $\equiv$ on $W^{\mathscr{A}}$ by $u \equiv v$ iff $\forall p \in S[\mathscr{A}, u \models p \Leftrightarrow \mathscr{A}, v \models p]$. Define the quotient structure $\bar{\mathscr{A}}$ as follows.

$$\begin{aligned}
\bar{w} &= \{v : w \equiv v\} \quad \text{for} \quad w \in W^{\mathscr{A}}, \\
W^{\bar{\mathscr{A}}} &= \{\bar{w} : w \in W^{\mathscr{A}}\}, \\
\pi^{\bar{\mathscr{A}}}(P) &= \{\bar{w} : w \in \pi^{\mathscr{A}}(P)\} \quad \text{for} \quad P \in \Phi_0', \\
\rho^{\bar{\mathscr{A}}}(A) &= \{(\bar{w}, \bar{v}) : (w, v) \in \rho^{\mathscr{A}}(A)\} \quad \text{for} \quad A \in \Sigma_0.
\end{aligned}$$

CLAIM 1. *The index of* $\equiv$ *is bounded by* $2^{\mathrm{size}(p_0)}$.

Proof. We have by (2) $|S - \Delta_0| \leqslant \gamma(p_0)$ and clearly $\gamma(p_0) \leqslant \text{size}(p_0)$. By (4) if $Q^p \in S$ then $\pi^{\mathscr{A}}(Q^p) = \pi^{\mathscr{A}}(p)$ and $p \in S$ so that members of Δ_0 play no essential role in distinguishing the states of $\mathscr{A}$. Thus, the inequivalent pairs of states are just those that can be distinguished by some member of $S - \Delta_0$. Hence the index of $\equiv$ is $\leqslant 2^{\text{size}(p_0)}$.

CLAIM 2. *For all* $p \in S$

(i) *if* $p = \langle a \rangle r$ *then* $\forall u, v \in W^{\mathscr{A}}\ [(u, v) \in \rho^{\mathscr{A}}(a) \Rightarrow (\bar{u}, \bar{v}) \in \rho^{\bar{\mathscr{A}}}(a)]$,

(ii) $\forall u[\mathscr{A}, u \models p \Leftrightarrow \bar{\mathscr{A}}, \bar{u} \models p]$.

Proof. The proof is by induction on $\gamma(p)$. Since $\gamma(p) \geqslant 0$ for all p, the claim holds vacuously for all values of $\gamma(p)$ less than 0.

Now assume $p' \in S$ and Claim 2 holds for all $p \in S$ for which $\gamma(p) < \gamma(p')$. We proceed by cases on the structure of p' to show that (i) and (ii) hold for p'.

$p' \in \Phi_0'$: (i) is true vacuously for p'. To verify (ii), if $\mathscr{A}, u \models p'$ then by the definition of $\pi^{\bar{\mathscr{A}}}$, $\bar{\mathscr{A}}, \bar{u} \models p'$, and if $\bar{\mathscr{A}}, \bar{u} \models p'$ then there exists $u' \equiv u$ such that $\mathscr{A}, u' \models p'$. Since $p' \in S$ then $\mathscr{A}, u \models p'$.

The cases $p \vee q$ and $\sim p$ are immediate from the definitions.

$\langle A \rangle p$: By (3), $p \in S$, so by induction, (i) and (ii) hold for p.

That (i) holds for $\langle A \rangle p$ follows immediately from the definition of $\rho^{\bar{\mathscr{A}}}$.

If $\mathscr{A}, u \models \langle A \rangle p$ then there exists v such that $\mathscr{A}, v \models p$ and $(u, v) \in \rho^{\mathscr{A}}(A)$. By (i), $(\bar{u}, \bar{v}) \in \rho^{\bar{\mathscr{A}}}(A)$. By the induction hypothesis $\bar{\mathscr{A}}, \bar{v} \models p$. Hence $\bar{\mathscr{A}}, \bar{u} \models \langle A \rangle p$.

Conversely, if $\bar{\mathscr{A}}, \bar{u} \models \langle A \rangle p$ then there exists $\bar{v} \in w^{\bar{\mathscr{A}}}$ such that $\bar{\mathscr{A}}, \bar{v} \models p$ and $(\bar{u}, \bar{v}) \in \rho^{\bar{\mathscr{A}}}(A)$. By the definition of $\rho^{\bar{\mathscr{A}}}(A)$ there exists $u', v' \in W^{\mathscr{A}}$ such that $(u', v') \in \rho^{\mathscr{A}}(A)$, $u' \equiv u$, and $v' \equiv v$. Then

$\mathscr{A}, v \models p$	by the induction hypothesis,
$\mathscr{A}, v' \models p$	because $p \in S$ and $v \equiv v'$,
$\mathscr{A}, u' \models \langle A \rangle p$	because $(u', v') \in \rho^{\mathscr{A}}(A)$,
$\mathscr{A}, u \models \langle A \rangle p$	because $\langle A \rangle p \in S$ and $u \equiv u'$.

Hence, (ii) holds for $\langle A \rangle p$.

$\langle q? \rangle p$: We first verify (i). If $(u, v) \in \rho^{\mathscr{A}}(q?)$ then $u = v$ and $\mathscr{A}, u \models q$. Now, $q \in S$ so that by the induction hypothesis $\bar{\mathscr{A}}, \bar{u} \models q$. Thus $(\bar{u}, \bar{v}) \in \rho^{\bar{\mathscr{A}}}(q?)$.

It is straightforward to verify (ii).

$\langle a; b \rangle p$: To verify (i) assume $(u, v) \in \rho^{\mathscr{A}}(a; b)$. There is w such that $(u, w) \in \rho^{\mathscr{A}}(a)$ and $(w, v) \in \rho^{\mathscr{A}}(b)$. Now, both $\langle a \rangle Q^{\langle b \rangle p}$ and $\langle b \rangle p$ are in S and are "smaller" than $\langle a; b \rangle p$ so by the induction hypothesis $(\bar{u}, \bar{w}) \in \rho^{\bar{\mathscr{A}}}(a)$ and $(\bar{w}, \bar{v}) \in \rho^{\bar{\mathscr{A}}}(b)$. Thus $(\bar{u}, \bar{v}) \in \rho^{\bar{\mathscr{A}}}(a; b)$.

Assume $\mathscr{A}, u \models \langle a; b \rangle p$. Then there exists v such that $\mathscr{A}, v \models p$ and $(u, v) \in \rho^{\mathscr{A}}(a; b)$. By (i) and the induction hypothesis $\bar{\mathscr{A}}, \bar{v} \models p$ and $(\bar{u}, \bar{v}) \in \rho^{\bar{\mathscr{A}}}(a; b)$. Thus $\bar{\mathscr{A}}, \bar{u} \models \langle a; b \rangle p$.

Conversely, let $\mathscr{A}, \bar{u} \models \langle a; b\rangle p$. There exists $\bar{v}$ such that $(\bar{u}, \bar{v}) \in \rho^{\mathscr{A}}(a)$ and $\mathscr{A}, \bar{v} \models \langle b\rangle p$. Also, $\langle b\rangle p$, $Q^{\langle b\rangle p}$ and $\langle a\rangle Q^{\langle b\rangle p} \in S$. Then

$$\begin{aligned}
&\mathscr{A}, v \models \langle b\rangle p && \text{by the induction hypothesis,}\\
&\mathscr{A}, v \models Q^{\langle b\rangle p} && \text{since } \pi^{\mathscr{A}}(Q^{\langle b\rangle p}) = \pi^{\mathscr{A}}(\langle b\rangle p),\\
&\mathscr{A}, \bar{v} \models Q^{\langle b\rangle p} && \text{by the induction hypothesis,}\\
&\mathscr{A}, \bar{u} \models \langle a\rangle Q^{\langle b\rangle p} && \text{since } (\bar{u}, \bar{v}) \in \rho^{\mathscr{A}}(a),\\
&\mathscr{A}, u \models \langle a\rangle Q^{\langle b\rangle p} && \text{by the induction hypothesis,}\\
&\mathscr{A}, u \models \langle a\rangle\langle b\rangle p && \text{since } \pi^{\mathscr{A}}(Q^{\langle b\rangle p}) = \pi^{\mathscr{A}}(\langle b\rangle p),\\
&\mathscr{A}, u \models \langle a; b\rangle p && \text{by semantic equivalence.}
\end{aligned}$$

$\langle a \cup b\rangle p$: If $(u, v) \in \rho^{\mathscr{A}}(a \cup b)$, then either $(u, v) \in \rho^{\mathscr{A}}(a)$ or $(u, v) \in \rho^{\mathscr{A}}(b)$. Both $\langle a\rangle Q^p$ and $\langle b\rangle Q^p$ are in S, so by the induction hypothesis, either $(\bar{u}, \bar{v}) \in \rho^{\bar{\mathscr{A}}}(a)$ or $(\bar{u}, \bar{v}) \in \rho^{\bar{\mathscr{A}}}(b)$. Hence, $(\bar{u}, \bar{v}) \in \rho^{\bar{\mathscr{A}}}(a \cup b)$, proving (i).

Assume $\mathscr{A}, u \models \langle a \cup b\rangle p$. Then there exists v such that $\mathscr{A}, v \models p$ and $(u, v) \in \rho^{\mathscr{A}}(a \cup b)$. By (i) and the induction hypothesis, $\bar{\mathscr{A}}, \bar{v} \models p$ and $(\bar{u}, \bar{v}) \in \rho^{\bar{\mathscr{A}}}(a \cup b)$. Thus, $\bar{\mathscr{A}}, \bar{u} \models \langle a \cup b\rangle p$.

Conversely, let $\bar{\mathscr{A}}, \bar{u} \models \langle a \cup b\rangle p$. There exists $\bar{v}$ such that $(\bar{u}, \bar{v}) \in \rho^{\bar{\mathscr{A}}}(a \cup b)$ and $\bar{\mathscr{A}}, \bar{v} \models p$. Either $(\bar{u}, \bar{v}) \in \rho^{\bar{\mathscr{A}}}(a)$ or $(\bar{u}, \bar{v}) \in \rho^{\bar{\mathscr{A}}}(b)$. If $(\bar{u}, \bar{v}) \in \rho^{\bar{\mathscr{A}}}(a)$, then

$$\begin{aligned}
&\mathscr{A}, v \models p && \text{by the induction hypothesis,}\\
&\mathscr{A}, v \models Q^p && \text{since } \pi^{\mathscr{A}}(Q^p) = \pi^{\mathscr{A}}(p),\\
&\bar{\mathscr{A}}, \bar{v} \models Q^p && \text{by the induction hypothesis,}\\
&\bar{\mathscr{A}}, \bar{u} \models \langle a\rangle Q^p && \text{by assumption that } (\bar{u}, \bar{v}) \in \rho^{\bar{\mathscr{A}}}(a),\\
&\mathscr{A}, u \models \langle a\rangle Q^p && \text{by the induction hypothesis,}\\
&\mathscr{A}, u \models \langle a\rangle p && \text{since } \pi^{\mathscr{A}}(Q^p) = \pi^{\mathscr{A}}(p).
\end{aligned}$$

Similarly, if $(\bar{u}, \bar{v}) \in \rho^{\bar{\mathscr{A}}}(b)$, then $\mathscr{A}, u \models \langle b\rangle p$. Hence, $\mathscr{A}, u \models \langle a\rangle p$ or $\mathscr{A}, u \models \langle b\rangle p$, so $\mathscr{A}, u \models \langle a \cup b\rangle p$ by semantic equivalence.

$\langle a^*\rangle p$: If $(u, v) \in \rho^{\mathscr{A}}(a^*)$ then there exists $u_0, u_1, \ldots, u_n$ such that $u = u_0$, $v = u_n$, and $(u_i, u_{i+1}) \in \rho^{\mathscr{A}}(a)$, $0 \leqslant i < n$. Since $\langle a\rangle Q^{\langle a^*\rangle p} \in S$ and is "smaller" than $\langle a^*\rangle p$ then $(\bar{u}_i, \bar{u}_{i+1}) \in \rho^{\bar{\mathscr{A}}}(a)$. Thus $(\bar{u}, \bar{v}) \in \rho^{\bar{\mathscr{A}}}(a^*)$.

Using what we have just shown it is easy to verify that $\mathscr{A}, u \models \langle a^*\rangle p$ implies $\bar{\mathscr{A}}, \bar{u} \models \langle a^*\rangle p$. To show that $\bar{\mathscr{A}}, \bar{u} \models \langle a^*\rangle p$ implies $\mathscr{A}, u \models \langle a^*\rangle p$ we show by a subinduction on n that $\bar{\mathscr{A}}, \bar{u} \models \langle a^n\rangle p$ implies $\mathscr{A}, u \models \langle a^*\rangle p$. By definition, $a^0 = \lambda = \theta^*$.

If $\bar{\mathscr{A}}, \bar{u} \models \langle a^0\rangle p$ then $\bar{\mathscr{A}}, \bar{u} \models p$. By the main induction hypothesis $\mathscr{A}, u \models p$ which implies $\mathscr{A}, u \models \langle a^*\rangle p$. Now assume $\bar{\mathscr{A}}, \bar{u} \models \langle a^{n-1}\rangle p$ implies $\mathscr{A}, u \models \langle a^*\rangle p$. Let $\bar{\mathscr{A}}, \bar{u} \models \langle a^n\rangle p$. There is $\bar{v}$ such that $\bar{\mathscr{A}}, \bar{v} \models \langle a^{n-1}\rangle p$ and $(\bar{u}, \bar{v}) \in \rho^{\bar{\mathscr{A}}}(a)$.

$\mathscr{A}, v \models \langle a^* \rangle p$ by the subinduction on n,
$\mathscr{A}, v \models Q^{\langle a^* \rangle p}$ by the fact that $\pi^{\mathscr{A}}(Q^{\langle a^* \rangle p}) = \pi^{\mathscr{A}}(\langle a^* \rangle p)$,
$\bar{\mathscr{A}}, \bar{v} \models Q^{\langle a^* \rangle p}$ by the main induction hypothesis,
$\bar{\mathscr{A}}, \bar{u} \models \langle a \rangle Q^{\langle a^* \rangle p}$ since $(\bar{u}, \bar{v}) \in \rho^{\bar{\mathscr{A}}}(a)$,
$\mathscr{A}, u \models \langle a \rangle Q^{\langle a^* \rangle p}$ by main induction hypothesis,
$\mathscr{A}, u \models \langle a \rangle \langle a^* \rangle p$ by the fact that $\pi^{\mathscr{A}}(Q^{\langle a^* \rangle p}) = \pi^{\mathscr{A}}(\langle a^* \rangle p)$,
$\mathscr{A}, u \models \langle a^* \rangle p$ by semantic implication.

This completes the proof of Claim 2.

To finally verify Theorem 3.2 we note first that $\text{size}(\bar{\mathscr{A}}) \leqslant 2^{\text{size}(p_0)}$ by Claim 1 and since $\mathscr{A}, w_0 \models p_0$ and $p_0 \in S$ then by Claim 2 $\bar{\mathscr{A}}, \bar{w}_0 \models p_0$. ∎

To complete the proof of Theorem 3.1, we show how to decide $\mathscr{A}, w \models p$ in time polynomial in the sizes of $\mathscr{A}$ and p. Of course, $\mathscr{A}$, w, and p must be encoded as strings to be presentable to a Turing machine. The only properties we need of such encodings, however, is that they can be decoded in time polynomial in the sizes of $\mathscr{A}$ and p.

THEOREM 3.3. *There exists a deterministic procedure which, given the code of a structure $\mathscr{A}$, a state w, and a formula p, determines in time polynomial in* $\text{size}(\mathscr{A}) + \text{size}(p)$ *whether or not* $\mathscr{A}, w \models p$.

Proof. The idea is to use the inductive definitions of $\rho^{\mathscr{A}}$ and $\pi^{\mathscr{A}}$ as procedures to compute $\rho^{\mathscr{A}}(a)$ and $\pi^{\mathscr{A}}(p)$ for programs a and formulas p. For instance, to compute $\pi^{\mathscr{A}}(\langle a \rangle p)$ first compute $\pi^{\mathscr{A}}(p)$, then $\rho^{\mathscr{A}}(a)$, then form the set $\{w: \exists u((w, u) \in \rho^{\mathscr{A}}(a)$ and $u \in \pi^{\mathscr{A}}(p))\}$. As another example, to compute $\rho^{\mathscr{A}}(a^*)$ simply compute the transitive closure of $\rho^{\mathscr{A}}(a)$. Each of the equations in the inductive definitions of $\rho^{\mathscr{A}}$ and $\pi^{\mathscr{A}}$ can be computed in polynomial time. ∎

Our upper bounds also extend to the propositional dynamic logic augmented with the program operator *converse* (or *transpose*). If a is a program then a^- also is and means "run a in reverse." Formally, $\rho(a^-) = \rho(a)^t = \{(u, v): (v, u) \in \rho(a)\}$. To extend the Small Model Theorem, we first push the converse operator to the atomic programs using the equivalences of programs: $(a; b)^- \leftrightarrow b^-; a^-$, $(a \cup b)^- \leftrightarrow a^- \cup b^-$, $(a^*)^- \leftrightarrow (a^-)^*$, and $(p?)^- \leftrightarrow p?$. This does not increase the length of the formula by more than a constant factor. Now we apply the quotient model construction treating A^- as just another atomic program symbol. It is easily verified that if the symbol A^- is given the "correct" interpretation in the original structure $\mathscr{A}_0$, i.e., $\rho^{\mathscr{A}_0}(A^-) = (\rho^{\mathscr{A}_0}(A))^t$, then A^- also has the correct interpretation in the quotient structure $\bar{\mathscr{A}}$. Thus, if p_0 is satisfied in $\mathscr{A}_0$, then it is also satisfied in $\bar{\mathscr{A}}$.

4. THE LOWER BOUND

The goal of this section is to show that there is a $c > 1$ such that the satisfiability problem for PDL (equivalently the validity problem for PDL) is not a member of

$\text{DTIME}(c^{n/\log n})$, where n is the length of the formula. The method we use is similar to that of Chandra and Stockmeyer [3] where they show certain game strategy problems require "exponential time." The fundamental observation is that a formula of PDL can efficiently describe the computation of an *alternating Turing machine*. Using the fact that

$$\text{ASPACE}(s(n)) = \bigcup_{c>1} \text{DTIME}(c^{s(n)}),$$

(Chandra and Stockmeyer [3] and Kozen [12]) we obtain the result.

For completeness we give a formal but simplified definition of an alternating Turing machine. A one-tape alternating Turing machine is a seven-tuple $M = (Q, \Delta, \Gamma, b, \delta, q_0, U)$, where

Q is the set of states,

Δ is the input alphabet,

Γ is the tape alphabet,

$b \in \Gamma - \Delta$ is the blank symbol,

$\delta \subseteq (Q \times \Gamma) \times (Q \times \Gamma \times \{L, R\})$ is the next move relation,

$U \subseteq Q$ is the set of universal states,

$Q - U$ is the set of existential states.

A *configuration* is a member of $\Gamma^*Q\Gamma^+$ and represents a complete state of the Turing machine. A *universal configuration* is a member of $\Gamma^*U\Gamma^+$ while an *existential configuration* is a member of $\Gamma^*(Q - U)\Gamma^+$.

Let $\alpha = xq\sigma y$ be a configuration, where $\sigma \in \Gamma$, $x, y \in \Gamma^*$, and $q \in Q$. We define $tape(\alpha) = x\sigma y$, $pos(\alpha) = l(x) + 1$, and $state(\alpha) = q$. ($l(x)$ is the length of x.) Let $\beta = x'q'\sigma'y'$ be a configuration, where $\sigma' \in \Gamma$, $x', y' \in \Gamma^*$, and $q' \in Q$. β is a *next* configuration of α if for some $\tau \in \Gamma$, either

(1) $(q, \sigma, q', \tau, L) \in \delta$, $x'\sigma' = x$, and $y' = \tau y$,

or

(2) $(q, \sigma, q', \tau, R) \in \delta$, $x' = x\tau$, and $\sigma'y' = y$ or $(y = y' = \lambda$ and $\sigma' = b)$.

A *computation sequence* is a sequence of configurations $\alpha_1, \ldots, \alpha_k$ for which α_{i+1} is a next configuration of α_i, $1 \leqslant i < k$.

A *trace* of M is a set C of pairs (α, t), where α is a configuration and $t \in \mathbb{N}$, such that

(i) if $(\alpha, t) \in C$ and α is a universal configuration, then for every next configuration β of α, there is a $t' < t$ for which $(\beta, t') \in C$;

and

(ii) if $(\alpha, t) \in C$ and α is an existential configuration, then there exists a next configuration β of α and a $t' < t$ for which $(\beta, t') \in C$.

The *set accepted by* M is

$$L(M) = \{x \in \Delta^*: \text{there exists } t \in \mathbb{N} \text{ and a trace } C \text{ of } M \text{ such that } (q_0x, t) \in C\}.$$

A trace C *uses space at most* s if for every $(\alpha, t) \in C$, α uses at most s tape cells. An alternating machine M *operates in space* $s(n)$ if for every $x \in L(M)$ of length n, there exists $t \in \mathbb{N}$ and a trace C of M such that $(q_0x, t) \in C$ and C uses space at most $s(n)$. Finally, ASPACE($s(n)$) is the class of sets accepted by alternating Turing machines which operate in space $s(n)$.

LEMMA 4.1. *If* $s(n) \geqslant n$, $K \subseteq \Delta^*$, *and* $K \in$ ASPACE($s(n)$) *then there exists a mapping* f *of* Δ^* *into formulas of* PDL *with the properties*

(i) $x \in K$ *iff* $f(x)$ *is satisfiable,*

(ii) *if* $n = l(x)$ *then* size($f(x)$) $= O(s(n))$,

(iii) *if the function* s *is suitably "honest" (i.e., computable in time polynomial in* s*), then* f *is computable in time polynomial in* s.

Proof. Let M be an alternating Turing machine that accepts K and operates in space $s(n)$. The following shows we may assume that M never repeats a configuration. There is an integer $m \geqslant 2$ such that $m^{s(n)}$ bounds the number of possible distinct configurations with no more than $s(n)$ tape cells. We may add a track to the tape of M which maintains a count (in m-ary) of how many "moves" have been made so far. The new machine accepts the same language as the old machine and also operates in space $s(n)$.

By thus eliminating the possibility of looping, the formulas we construct need only "simulate" the first component of the trace. Formally, a *simplified trace* is a finite set D of configurations such that

(i) if $\alpha \in D$ and α is a universal configuration, then every next configuration β of α is in D;

and

(ii) if $\alpha \in D$ and α is an existential configuration, then there is a next configuration β of α in D.

A simplified trace D *accepts* $x \in \Sigma^*$ if the initial configuration $q_0x \in D$.

For machines without looping, there is a natural correspondence between traces and simplified traces. A trace is mapped into a simplified trace by simply dropping the second component of each pair. To go from a simplified trace to a trace, the maximum length computation sequence beginning from a configuration α in the simplified trace will serve as the second component for the pair beginning with α in the trace. Since the machine never repeats a configuration, the maximum always exists. The following can be proved easily from these considerations.

LEMMA 4.2. *Let* M *be an alternating machine which never repeats a configuration. Then*

$$L(M) = \{x \in \Delta^* : \textit{there exists a simplified trace of } M \textit{ accepting } x\}.$$

To continue the proof of Lemma 4.1, let x be given with $l(x) = n$ and let $m = s(n) + 1$. Assume M uses tape cells numbered 1, 2, 3,... . The atomic formulas are:

$$\begin{aligned} P_{i,\sigma}, &\quad \text{where } 0 \leqslant i \leqslant m \text{ and } \sigma \in \Gamma, \\ H_i, &\quad \text{where } 0 \leqslant i \leqslant m, \\ Q_q, &\quad \text{where } q \in Q. \end{aligned}$$

Informally "$P_{i,\sigma}$" means "cell i contains σ," "H_i" means "the head is visiting cell i," and "Q_q" means "the state is q."

There is one basic program which we denote by $\vdash$. A truth assignment to the basic formulas will correspond to a configuration of M. We define global formulas $g_1, \ldots, g_6$ which state that only "configuration-like" truth assignments are possible and the relation $\vdash$ behaves correctly.

g_1: There is exactly one state.

$$\bigvee_{q \in Q} \left(Q_q \wedge \bigwedge_{q' \in Q - \{q\}} \sim Q_{q'} \right).$$

g_2: There is exactly one symbol per cell.

$$\bigwedge_{i=0}^{m} \bigvee_{\sigma \in \Gamma} \left(P_{i,\sigma} \wedge \bigwedge_{\sigma' \in \Gamma - \{\sigma\}} \sim P_{i,\sigma'} \right).$$

g_3: The unread cells are maintained.

$$\bigwedge_{i=0}^{m} \bigwedge_{\sigma \in \Gamma} (\sim H_i \wedge P_{i,\sigma} \supset [\vdash] P_{i,\sigma}).$$

g_4: Assuming there is exactly one head position, the next head position is one away from the old one and 0, m are not head positions.

$$\begin{aligned} \bigwedge_{i=1}^{m-1} & ((H_i \supset [\vdash]((H_{i-1} \wedge \sim H_{i+1}) \vee (\sim H_{i-1} \wedge H_{i+1}))) \\ & \wedge (\sim H_{i-1} \wedge \sim H_{i+1} \supset [\vdash] \sim H_i)) \\ & \wedge \sim H_0 \wedge \sim H_m \end{aligned}$$

g_5: The universal states behave correctly.

$$\begin{aligned} \bigwedge_{i=1}^{m-1} \bigwedge_{\sigma \in \Gamma} \bigwedge_{q \in U} & (H_i \wedge P_{i,\sigma} \wedge Q_q \\ & \supset \bigwedge_{\substack{q',\sigma': \\ (q,\sigma,q',\sigma',R) \in \delta}} \langle \vdash \rangle (H_{i+1} \wedge P_{i,\sigma'} \wedge Q_{q'}) \\ & \wedge \bigwedge_{\substack{q',\sigma': \\ (q,\sigma,q',\sigma',L) \in \delta}} \langle \vdash \rangle (H_{i-1} \wedge P_{i,\sigma'} \wedge Q_{q'})) \end{aligned}$$

(Note: Empty conjunctions are defined to be *true*.)

g_6: The existential states behave correctly.

$$\bigwedge_{i=1}^{m-1} \bigwedge_{\sigma\in\Gamma} \bigwedge_{q\in Q-U} (H_i \wedge P_{i,\sigma} \wedge Q_q$$

$$\supset \bigvee_{\substack{q',\sigma':\\ (q,\sigma,q',\sigma',R)\in\delta}} \langle\vdash\!\!\rangle(H_{i+1} \wedge P_{i,\sigma'} \wedge Q_{q'})$$

$$\vee \bigvee_{\substack{q',\sigma':\\ (q,\sigma,q',\sigma',L)\in\delta}} \langle\vdash\!\!\rangle(H_{i-1} \wedge P_{i,\sigma'} \wedge Q_{q'}))$$

(Note: Empty disjunctions are defined to be *false*.)
Define

$$g = \bigwedge_{i=1}^{6} g_i .$$

Let $x = \sigma_1 \cdots \sigma_n$. We define now an initial formula h which describes the initial configuration.

$$h\colon Q_{q_0} \wedge H_1 \wedge \sim H_0 \wedge \bigwedge_{i=2}^{m} \sim H_i \wedge \bigwedge_{i=1}^{n} P_{i,\sigma_i} \wedge P_{0,b} \wedge \bigwedge_{i=n+1}^{m} P_{i,b} .$$

Finally we define

$$f(x) = h \wedge [\vdash\!\!-\!\!*]g.$$

We see by inspection that $f(x)$ satisfies conditions (ii) and (iii) of the lemma. To show that (i) holds, we describe how to construct a satisfying model for $f(x)$ given a simplified trace containing the initial configuration q_0x, and conversely, we show how to construct a simplified trace given a model for $f(x)$. We leave to the reader the details of showing that the constructed trace and model have the desired properties.

Let D be a simplified trace of M accepting x and using space at most $s(n)$. We define a structure $\mathscr{A} = (W, \pi, \rho)$:

$W = D$;

$\pi(P_{i,\sigma}) = \{\alpha\colon \text{tape}(\alpha)_i = \sigma\}$, $0 \leqslant i \leqslant m$, $\sigma \in \Gamma$ ($\text{tape}(\alpha)_i$ is the ith symbol of tape(α) if $1 \leqslant i \leqslant l(\text{tape}(\alpha))$ and is "b" otherwise);

$\pi(H_i) = \{\alpha\colon \text{pos}(\alpha) = i\}$, $0 \leqslant i \leqslant m$;

$\pi(Q_q) = \{\alpha\colon \text{state}(\alpha) = q\}$, $q \in Q$;

$\rho(\vdash\!\!-) = \{(\alpha, \beta)\colon \beta \text{ is a next configuration of } \alpha\}$.

We claim without proof that $\mathscr{A}, q_0x \models f(x)$, and hence $f(x)$ is satisfiable.

Conversely, let $\mathscr{A} = (W, \pi, \rho)$ be a structure and $w_0 \in W$ such that $\mathscr{A}, w_0 \models f(x)$. By Theorem 3.2, we can assume that $\mathscr{A}$ is finite. We extract from $\mathscr{A}$, w_0 a simplified trace.

First note that g holds in every state u accessible from w_0. Since also h holds at w_0, g_1, g_2 and g_4 imply that the propositional variables true at u describe in a natural way a unique configuration $\alpha(u)$. It is *not* true that $(u, v) \in \rho(\vdash)$ implies that $\alpha(v)$ is a next configuration of $\alpha(u)$, although g_3 does tell us that $\alpha(v)$ is a possible next configuration of *some* machine (*not necessarily of* M).

We may now construct our simplified trace. We first define inductively a set $S \subseteq W$ of states. The desired simplified trace is then $\alpha(S)$.

(1) $w_0 \in S$.

(2) Suppose $w \in S$ and $\alpha(w)$ is a universal configuration. g_5 holds at w, so for each next configuration β of $\alpha(w)$, there is a state u_β so that $(w, u_\beta) \in \rho(\vdash)$ and $\alpha(u_\beta) = \beta$. Put u_β in S for each such β.

(3) Suppose $w \in S$ and $\alpha(w)$ is an existential configuration. g_6 holds at w, so there is a next configuration β of $\alpha(w)$ and a state u so that $(w, u) \in \rho(\vdash)$ and $\alpha(u) = \beta$. Put u in S.

We claim without proof that $\alpha(S)$ is a simplified trace. That $q_0 x \in \alpha(S)$ follows from the facts that $w_0 \in S$ and h holds at w_0. ∎

The proof of Lemma 4.1 allows us to conclude that the exponential upper bound of Theorem 3.2 cannot be improved except possibly for the choice of constant.

COROLLARY 4.3. *There is a constant $d > 1$ for which there exist arbitrarily long satisfiable formulas f of* PDL, *but every model for f has* size $\geqslant d^{\text{size}(f)}$.

Proof. Let M be a deterministic linear-space Turing machine which, for every input x of length n, counts up to 2^n and then halts. Regarded as an alternating machine, M operates in space n and $L(M) = \Sigma^*$, but every simplified trace of M accepting x has cardinality $\geqslant 2^n$ since it contains each of the reachable configurations in the computation of M. Let $f(x)$ be the formula of size at most cn constructed in the proof of Lemma 4.1, and let $\mathscr{A}$ be any model for $f(x)$. The proof shows how to construct a simplified trace D for M which accepts x, and $|D| \leqslant \text{size}(\mathscr{A})$. Since $|D| \geqslant 2^n$, we have that

$$\text{size}(\mathscr{A}) \geqslant 2^n \geqslant 2^{\text{size}(f)/c} = d^{\text{size}(f)}$$

for $d = 2^{1/c}$. ∎

THEOREM 4.4. *There is a constant $c > 1$ such that the satisfiability (validity) problem for* PDL *is not a member of* DTIME($c^n/\log n$), *where n is the length of the formula.*

Proof. Let K be a set which is a member of DTIME(3^n) − DTIME(2^n). Hence $K \in$ ASPACE(n). Let f be a function satisfying the conditions of Lemma 4.1. Since $\lambda n.\ n$ is suitably honest, then f is computable in time $t(n)$ for some polynomial t. Also there is a constant d with the property that if $n = l(x)$, then size($f(x)$) $\leqslant dn$. The atomic formulas and programs can be coded as strings of length $\leqslant \log n$ so that $l(f(x)) \leqslant dn \cdot \log n$. There is $c > 1$ such that $c^{dn \cdot \log n / \log(dn \cdot \log n)} + t(n) < 2^n$.

If the satisfiability problem for L were a member of DTIME($c^{n/\log n}$), then the inequality above would guarantee that $K \in$ DTIME(2^n), which is an impossibility. ▮

We remark that the proof of our lower bound makes no use of the test operator "?," so the bound applies equally well to the language $L \subset$ PDL of test-free formulas. Berman and Paterson have shown L to be strictly weaker than PDL in the sense that there is a formula p of PDL such that $p \equiv q$ is not a valid formula (of PDL) for any $q \in L$ [2].

We can restrict still further. Inspection of $f(x)$ in the proof of Lemma 4.1 shows that no use is made of union and concatenation either, and that only a single basic program symbol is needed for the lower bound to apply.

We note also that there is still a "gap" between the upper and lower bounds because the upper bound is non-deterministic while the lower bound is deterministic.

5. Applications

Equivalence of Regular Expressions

As we noted earlier, programs of PDL can be thought of as regular expressions. It can be shown that a and b are equivalent regular expressions if and only if $\langle a\rangle P \equiv \langle b\rangle P$ is valid. Hence, the validity problem for PDL (and also L) contains the equivalence problem for regular expressions as a subproblem. Meyer and Stockmeyer have shown that the equivalence problem for regular expressions is polynomial space complete [17].

Nondeterministic Ianov Schemes

A *Ianov scheme* is an uninterpreted program scheme with only one variable (cf. Greibach [6]). We say two schemes are *strongly equivalent* if they compute the same result or they both fail to halt for each initial value in each interpretation.

Given a Ianov scheme, we can use automaton-theoretic techniques to construct a PDL program a, whose tests have the form $P?$ or $\sim P?$. The program a describes the set of paths $W(a)$ from the start box to a halt box. If a and b are strongly equivalent, then it does not necessarily follow that $W(a) = W(b)$. However, a and b are strongly equivalent if and only if $\langle a\rangle Q \equiv \langle b\rangle Q$ is valid, where Q does not appear in a or b. Thus, we get another decision procedure for strong equivalence of Ianov schemes. Moreover, the above goes through unchanged even when the schemes are permitted to be nondeterministic.

Classical Modal Systems

The modal systems K, T, $S4$, $S5$ (cf. Ladner [16]) are recognizable subsystems of propositional dynamic logic.

K allows only the modality A,

T allows only the modality $A \cup \lambda$,

$S4$ allows only the modality A^*,

$S5$ allows only the modality $(A \cup A^-)^*$.

Kripke [14] has already given decision procedures for validity in each of these modal systems. Our decision procedure subsumes these four problems as special cases.

6. Conclusion

A language for describing properties of programs requires the blending of program statements with logical assertions. There have been several attempts to integrate assertions and programs into one language. Partial correctness assertions [10] were one of the first. They were unified and generalized by Pratt [19] using ideas of modal logic. Ideas reminiscent of modal logic also appear in the work of the Polish group investigating "Algorithmic Logic" (cf. Kreczmar [13] and Salwicki [21]), and in the recent work of Kröger [15] and Constable [4].

Such work has been motivated by a desire to define a language rich enough to describe interesting properties of realistic programs. We have attempted to abstract from that work, and notably from [19], the "pure" logical structure underlying these formal systems. We feel a thorough understanding of this structure is a prerequisite to obtaining a good grasp on the more complicated, albeit more applicable, systems, just as classical propositional logic is fundamental to the understanding of first-order predicate calculus.

We have shown that every satisfiable formula of propositional dynamic logic has a model of size at most exponential in the length of the formula. This leads to a nondeterministic decision procedure for satisfiability which runs in time c^n on formulas of size n, namely, "guess" a model and check that it satisfies the formula. This algorithm is impractical, not only because it is nondeterministic, but also because it uses the worst-case time on all formulas. Pratt has developed a decision procedure which is fast on many natural formulas [20]. That such a procedure cannot be fast on all formulas follows from our lower bound of deterministic time $d^{n/\log n}$.

Another interesting problem with obvious application to automatic program verification is to find a complete and natural proof system for propositional dynamic logic. Such a proof system was proposed by Segerberg [22] and proven to be complete by Parikh [18].

Several interesting extensions to propositional dynamic logic have begun to appear (cf. [7, 9, 1]).

Acknowledgments

We thank S. K. Thomason and R. Woodrow for some helpful suggestions which simplified the proof of Theorem 3.3.

References

1. K. Abrahamson and M. J. Fischer, "Applications of Boolean Variables to Automata Theory and Dynamic Logic," Tech. Report 78–08–02, University of Washington, 1978.
2. F. Berman and M. S. Paterson, "Test-Free Propositional Dynamic Logic is Weaker than PDL," Tech. Report TR 77–10–02, 1977.

3. A. K. CHANDRA AND L. J. STOCKMEYER, Alternation, *in* "17th IEEE Symposium on Foundations of Computer Science, 1976," pp. 98–108.
4. R. L. CONSTABLE, On the theory of programming logics, *in* "Ninth ACM Symposium on Theory of Computing, 1977," pp. 269–285.
5. E. W. DIJKSTRA, Guarded commands, nondeterminacy and formal derivation of programs, *Comm. ACM 18* 8 (1975), 453–457.
6. S. A. GREIBACH, "Theory of Program Structures: Schemes, Semantics, Verification," Lecture Notes in Computer Science No. 36, Springer–Verlag, New York/Berlin, 1975.
7. D. HAREL, "Logics of Programs: Axiomatics and Descriptive Power," Ph. D. dissertation, M.I.T., 1978. Available as Tech. Report MIT/LCS/TR200, M. I. T., 1978.
8. D. HAREL, A. R. MEYER, AND V. R. PRATT, Computatibility and completeness in logics of programs: preliminary report, *in* "Ninth ACM Symposium on Theory of Computing, 1977," pp. 261–268.
9. D. HAREL AND V. R. PRATT, Nondeterminism in logics of programs, *in* "Fifth ACM Symposium on Principles of Programming Languages, 1978."
10. C. A. R. HOARE, An axiomatic basis for computer programming, *Comm. ACM 12* 10 (1969), 576–580, 583.
11. C. A. R. HOARE AND P. LAUER, Consistent and complementary formal theories of the semantics of programming languages, *Acta Inform. 3* 2 (1974), 135–153.
12. D. KOZEN, On parallelism in Turing machines, *in* "17th IEEE Symposium on Foundations of Computer Science, 1976," pp. 89–97.
13. A. KRECZMAR, Effectivity problems of algorithmic logic, *in* "Automata, Languages and Programming, 2nd Colloquium, University of Saarbrücken, July 29-Aug. 2, 1974" (J. Loeckx, Ed.), Lecture Notes in Computer Science No. 14, Springer–Verlag, Berlin/New York.
14. S. A. KRIPKE, Sementical analysis of modal logic I: Normal modal propositional calculi, *Z. Math. Logik Grundlagen Math.* 9 (1963), 67–96.
15. F. KRÖGER, Logical rules of natural reasoning about programs, *in* "Automata, Languages and Programming" (S. Michaelson and R. Milner, Eds.), pp. 87–98, Edinburgh Univ. Press, Edinburgh, 1976.
16. R. E. LADNER, The computational complexity of provability in systems of modal propositional logic, *SIAM J. Comput. 6* 3 (1977), 467–480.
17. A. R. MEYER AND L. J. STOCKMEYER, Word problems requiring exponential time, *in* "Fifth ACM Symposium on Theory of Computing, 1973," pp. 1–9.
18. R. PARIKH, A completeness result for PDL, *in* "Symposium on Mathematical Foundations of Computer Science, Zakopane, Warsaw, Sept. 1978."
19. V. R. PRATT, Semantical considerations on Floyd-Hoare logic, *in* "17th IEEE Symposium on Foundations of Computer Science, 1976," pp. 109–121.
20. V. R. PRATT, A practical decision method for propositional dynamic logic, *in* "10th ACM Sympoisum on Theory of Computing, 1978," pp. 326–337.
21. A. SALWICKI, Formalized algorithmic languages, *Bull. Acad. Polon. Sci. Sér. Sci. Math. Astronom. Phys.* 18 (1970), 227–232.
22. K. SEGERBERG, A completeness theorem in the modal logic of programs, *Notices Amer. Math. Soc. 24* 6 (1977), A-552.

Printed in Belgium