JOURNAL OF COMPUTER AND SYSTEM SCIENCES 20, 231–254 (1980)

A Near-Optimal Method for Reasoning about Action*

VAUGHAN R. PRATT

Massachusetts Institute of Technology, Cambridge, Massachusetts 02139

We give an algorithm for "before-after" reasoning about action. The algorithm decides satisfiability and validity of formulas of propositional dynamic logic, a recently developed logic of change of state that subsumes the zero-order component of most other action-oriented logics. The algorithm requires time at most proportional to an exponentially growing function of the length (number of occurrences of variables and connectives) of the input. Fischer and Ladner have shown that that every algorithm for this problem must take exponential time, making this algorithm optimal to within a polynomial. Application areas include program verification, program synthesis, and discourse analysis. The algorithm is based on the method of semantic tableaux, appropriately generalized to dynamic logic. A formal treatment of the generalization, called *Hintikka structures*, is developed.

1. INTRODUCTION

Dynamic Logic

Almost all existing logics of imperative programs contain implicitly or explicitly the construct "*after*(a, p)" which asserts that after action a halts, program p holds. They also almost all cater for programming constructs to do with assignment, testing, sequencing, choice and iteration, and logical constructs to do with truth functions and quantification. Dynamic logic [18] consists of (i) a language in which such constructs appear explicitly, and (ii) a formal semantics for that language.

Propositional dynamic logic (PDL) was defined by Fischer and Ladner [6] as the natural restriction of first-order dynamic logic to the term-free case (therefore no assignments, quantifiers or non-zeroary predicates). This restriction is of interest in that it gives a convenient way of studying the term-independent part of reasoning with formulas and actions (programs). Propositional variables (as with any variables) can be viewed as expressions whose internal structure is of no concern in the reasoning at hand. Thus the techniques we develop here apply to more general reasoning about action in the same way that propositional calculus techniques apply to more general static reasoning about specific domains.

* This research was supported by the National Science Foundation under NSF grant nos. MCS76-18461 and MCS78-04338. This paper is a revision of "A Practical Decision Method for Propositional Dynamic Logic," presented at the 10th Annual ACM Symposium on Theory of Computing, San Diego, May 1978. The revision incorporates a substantially more efficient method. The material on process logic and axiom systems in the symposium paper has been omitted and will appear elsewhere.

0022-0000/80/020231-24$02.00/0

The language of PDL is a set of expressions divided into formulas and actions. Letting $a, b, c,...$ range over actions and $p, q, r,...$ over formulas, we many enumerate the expressions of PDL as follows.

The set Φ of formulas:

Atomic formulas: $P, Q, R,...$

Composite formulas: $\sim p, \langle a\rangle p$

The set Σ of actions:

Atomic actions: $A, B, C,...$

Composite actions: $a \cup b, a; b, a^*, p?$

$P, Q, R,...$ are the usual propositional or formula variables, ranging over truth values, and $\sim$ is logical negation. (We obtain all other logical connectives as abbreviations, starting with $p \wedge q$ for $\langle p?\rangle q$.) $\langle a\rangle p$ is our notation for "a can ensure p." To be more precise, $\langle a\rangle p$ is true of state u just when p is true of one of the states a can terminate in when started in u. (The need to deal with more than one state in such a definition is what makes this a dynamic logic, as opposed to say the more static propositional calculus, where $\sim p$ is true of u just when p is not true of *the same state* u.)

$A, B, C,...$ are action variables, analogous to formula variables. They may range over nondeterministic actions in general. $a \cup b$ is the *choice* of a or b. $a; b$ is the *sequence* a followed by b. a^* is the *iteration* of a an indefinite but finite number of times. $p?$ is the *test* of p, an action whose execution is permitted just when p holds. These concepts are made more precise by the semantics given in Section 2.

Typical assertions possible in dynamic logic are $\langle a \cup b\rangle p$ (one of a or b can ensure p, equivalent to $\langle a\rangle p \vee \langle b\rangle p$), $\langle a^*\rangle p$ (a can eventually ensure p), $\langle a\rangle\langle b\rangle p$ (a can ensure that b can ensure that p, equivalent to $\langle a; b\rangle p$), and $\sim\langle a\rangle p$ (p is guaranteed to be false after executing a, i.e., a cannot attain any state satisfying p).

Definability. Dynamic logic subsumes a number of other logics by offering definitions for their constructs, and we shall take advantage of this throughout the paper. Using this ability we treat $p \wedge q$ as an abbreviation for $\langle p?\rangle q$, F as an abbreviation for $P \wedge \sim P$, T as $\sim F$, $p \supset q$ as $\sim(p \wedge \sim q)$, and similarly for $\vee$ and $\equiv$. Definable programming concepts include *if* p *then* a *else* b, as $(p?; a) \cup (\sim p?; b)$, and *while* p *do* a, as $(p?; a)^*; \sim p?$. We define $[a]p$ as $\sim\langle a\rangle \sim p$; $[a]$ is the dual of $\langle a\rangle$ in the same sense that $\forall x$ is the dual of $\exists x$. Definable program correctness constructs include Hoare's [11] partial correctness construct $p\{a\}q$, definable as $p \supset [a]q$, and Basu and Yeh's [1] total correctness construct $p[a]q$ for deterministic programs, as $p \supset \langle a\rangle q$. Dijkstra's total correctness construct $wp(a, p)$ for nondeterministic programs [5] is definable as $[a]p \wedge \langle a\rangle T \wedge \sim\bigcap a$ where $\bigcap a$ asserts that a has a diverging computation. For deterministic programs $\bigcap a$ is definable in DL as $[a]F$ but for the more general case of nondeterministic programs $\bigcap a$

is not definable in propositional *DL*. However Meyer and Winklmann [15, 25] have shown that it *is* definable in first-order *DL*.

Theorems. Some formulas are always true, or *valid*. They include $p \vee \sim p$; $\langle a \cup b \rangle p \equiv \langle a \rangle p \vee \langle b \rangle p$ as we saw above along with $\langle a;b \rangle p \equiv \langle a \rangle \langle b \rangle p$; $[a](p \supset q) \wedge [a]p \supset [a]q$ (a sort of "delayed Modus Ponens"); $\langle a \rangle (p \vee q) \equiv \langle a \rangle p \vee \langle a \rangle q$; $\langle a \rangle (p \wedge q) \supset \langle a \rangle p \wedge \langle a \rangle q$ (but not the converse); $\langle a^* \rangle p \equiv p \vee \langle a \rangle \langle a^* \rangle p$ (decomposition of a^* into zero and non-zero number of iterations); $\langle (a; b)^*; a \rangle p \equiv \langle a; (b; a)^* \rangle p$; and $p \wedge [a^*](p \supset [a]p) \supset [a^*]p$ (analogous to mathematical induction). One would expect such obviously valid formulas to be among the theorems of any practical axiom system for *DL*, and to be efficiently identifiable as valid by any practical decision method for validity.

Rules. We may also observe that if p and $p \supset q$ are valid then so is q (corresponding to the rule of Modus Ponens), and if p is valid then so is $[a]p$ for any a (the rule of Necessitation from modal logic). Any rule whose conclusion is valid when its premises are valid is called *sound*. Other rules include: from $p \supset p'$ and $p'\{a\}q$ infer $p\{a\}q$; from $p\{a\}\, q'$ and $q' \supset q$ infer $p\{a\}q$; from $p\{a\}q$ and $q\{b\}r$ infer $p\{a; b\}r$; from $p \wedge r\{b\}q$ and $p \wedge \sim r\{b\}q$ infer $p\{\textit{if } r \textit{ then } a \textit{ else } b\}q$; and from $p \wedge q\{a\}p$ infer $p\{\textit{while } q \textit{ do } a\}p \wedge \sim q$ (Hoare's rules[11]). One would expect such obviously sound rules to be derivable in any practical axiom system for *DL*.

Example. The following gives a simple example of the sort of problem PDL is useful for. Consider the two programs "while P do $(A; A)$" and "while P do A". (We assume that testing P has no side-effects, that is, does not cause a change of state.) It is the case that if the first program can reach a final state when started in a given state, so can the second. This is true even if A is nondeterministic. (When A is deterministic, "can reach a final state" means "is guaranteed to halt," or "terminates.")

This valid statement about the relationship between the termination of the respective programs can be easily stated in PDL, as $\langle \textit{while } P \textit{ do } (A; A) \rangle T \supset \langle \textit{while } P \textit{ do } A \rangle T$, or $\sim\langle\langle (P?; A; A)^*; \sim P? \rangle \sim\langle P? \rangle \sim P? \rangle \sim \langle (P?; A)^*; \sim P? \rangle \sim \langle P? \rangle \sim P$ if we were to expand out *all* our abbreviations (which we obviously wouldn't want to have to do in actual applications). Hence it serves as a simple representative of a class of problems for decision which a method exists.

Outline

The main contribution of this paper is a deterministic exponential time algorithm for deciding satisfiability in PDL.

Fischer and Ladner [6, 7] showed how a nondeterministic Turing machine could accept satisfiable PDL formulas of length n within a number of steps proportional to c^n for some c, a *nondeterministic exponential upper bound*, and proved that there did not exist a deterministic Turing machine that could always decide whether an arbitrary PDL formula of length n was satisfiable in fewer than d^n steps for some $d > 1$, a *deterministic exponential lower bound*. The upper bound was obtained by using the equivalent for PDL of the method of truth tables, which enumerates all possible models of a given size and

evaluates the formula in each. With our present knowledge about nondeterministic computation the best deterministic upper bound derivable from their result is 2^{c^n} for some c. The lower bound was obtained by reducing the acceptance problem for linear-space-bounded alternating Turing machines to the decision problem for PDL satisfiability.

Using the equivalent for PDL of the method of tableaux we give a *deterministic* exponential upper bound, meeting the lower bound to within a polynomial and giving a method with worst-case time within a polynomial of all of the thus-far-analyzed methods for satisfiability in ordinary propositional calculus. Our method could be viewed as showing how to reduce the satisfiability problem for PDL to the acceptance problem for linear-space-bounded alternating Turing machines. We shall not however so view it, since the algorithm is just as easy to describe without using alternating machines. The reader familiar with alternation will see the connection without any difficulty.

The significance of our result will be felt first in the area of automatic program verification, where the primary objective is to minimize the amount of detail the programmer must supply in a proof of correctness of a program that is to be certified mechanically. In fact our motivation for studying this problem was to find and apply such an algorithm to the program verification system we have been constructing at MIT during the past two years. Our experience was that the amount of detail required from the user in the area characterizable as the propositional dynamic logic component of the system was the greatest bottleneck in user productivity. Observing the work being done with other verification systems, particularly those of Wegbreit and of Oppen, convinced us that our problems were common to most and probably all program verification systems. The role of dynamic logic in this was to simplify the problem domain, making it easier for us to formulate and solve this particular problem.

We hope that at some time the need for logics of action will be felt by the computational linguistics community, at which time our algorithm will find application there also.

The result is of theoretical interest because of the scarcity of naturally occurring problems with such tight non-linear bounds. In fact the only previously known such problem is that of testing for circularity in Knuth-type attribute grammars, which was shown by Jazayeri, Ogden, and Rounds [12] to have a deterministic time lower bound of $c^{n/\log n}$ for some $c > 1$ and a deterministic time upper bound of d^n for some d for grammars of length n bits. Even this gap, small as it is, is larger than any polynomial. Shortly after we discovered this result, H. Lewis established similarly tight bounds for the decision problems for two fragments of predicate calculus [14].

Our result is of further theoretical interest in that, like the Jazayeri et al result, the tight bounds can be proved easily via a correspondence with a family of automata that characterize deterministic exponential time. There are at present two such families: Cook's family of linear-space auxiliary push-down automata [4], and the Chandra–Kozen–Stockmeyer family of linear-space alternating Turing machines (ATM's) [3, 13]. The Jazayeri et al result uses a correspondence with the former, our PDL result with the latter. The PDL lower bound is proved explicitly in [6] via a reduction *from* ATM's. Our upper bound could be proved via a reduction *to* ATM's, but a more direct proof is just as convenient.

Our decision method can be viewed as symbolic execution, an approach to program verification that has attracted interest in some circles in the past few years [16, 24]. The connection will become apparent when the tableau method is described.

Our application to PDL of the Hintikka set approach [10, 22, 23] may be of interest to theoretical logicians. The applicability of the Hintikka set approach to binary relations is not immediately obvious until it is seen. We broaden the scope of applicability of the Hintikka approach with the help of the notion of a *Hintikka structure*, which has two functions and four predicates over a domain of *theories*.

More recently [20] the author has developed an alternative decision method for PDL that is to the method of truth tables for propositional calculus as the method described here is to the method of tableaux for propositional calculus.

The theory underlying our algorithm also contains the bulk of the material needed for a proof of completeness of the Segerberg axioms for PDL [21]. Such a proof, using a Gentzen type axiomatization as an obvious intermediate step, was sketched in [19]. (The observant reader of [19] may have noticed two errors in our axiomatization: the two P's in the first rule should be lower case, and, as pointed out to us by M. Valiev, the first premise of our induction rule should read $\Gamma \rightarrow p, \Delta$, not $\Gamma \rightarrow p$.) This application of the theory has been removed from this paper and will appear later as a separate paper. The existence of other proofs of the completeness result, particularly [17], which appears to be the first satisfactory such proof, has lessened for us the urgency of publication of yet another proof. (There also exists a sketch for a proof by D. Gabbay [8].) Furthermore, complexity results and completeness proofs tend to appeal to different audiences; this paper accordingly has focused on complexity to the exclusion of axiomatizations.

Much of our theory is also applicable to our newly developed logic of processes, which also appeared in [19]. In the interests of factoring out our substantive novel contributions the process logic material, like the axiom system, has been transferred to a separate paper.

2. Hintikka Structures

In this section we develop a novel method for specifying the semantics of a logic. This semantic approach is mathematically attractive in its own right. However our primary purpose in developing it is to facilitate a more formal treatment of the tableau method. The tableau method for propositional calculus is sufficiently trivial that its formalization serves little or no purpose as an aid to understanding. For PDL however the tableau method is rather more involved, warranting a more formal treatment.

We shall adopt the trick of letting the language itself supply the basis for defining a semantic model. This approach is not very effective for supplying the semantics of number theory or complex variable theory, but it works well for propositional calculus, predicate calculus, and as we shall see in the next section, PDL.

Suppose for example that we wished to define the semantics of the propositional calculus, whose language we shall consider to be a set of formulas containing propositional variables and being closed under disjunction and negation. Intuitively we would want

a *model* to be an assignment of truth values to the variables of the language. Of course this assignment then forces the values of the remaining formulas of the language, via the standard interpretation of $\sim$ and $\vee$. So we could take a model to be any assignment of truth values to *all* the statements of the language, subject in the case of propositional calculus to the constraints that p and $\sim p$ be assigned different truth values, and $p \vee q$ be assigned *true* just when at least one of p or q is assigned *true*.

The formalization of this that we shall use here takes a model to be a set. The elements of the set are called *statements*, and are drawn from a larger set of statements called the *semantic language*. The role of statements is to define the model by asserting all of its properties. We let the variables s, t,... range over statements.

For propositional calculus it suffices to use statements of the form "p is true" and "p is false," which we can abbreviate to Tp and Fp. This is exactly the approach adopted by Smullyan [22] for propositional calculus.

A model should have two properties. First, every statement should occur either positively or negatively (completeness), but not both (consistency). Second, the meanings of the formulas should be respected. Thus for propositional calculus $T(p \vee q)$ should be in the model if and only if one of Tp or Tq is in the model. Moreover if $F(p \vee q)$ is in the model then both Fp and Fq should be in it as well. And $T \sim p$ should be in the model if and only if Tp is not in.

There is a certain amount of redundancy with the signs and the negation symbol that makes this approach seem a little clumsy. The rules about consistency and completeness seem to say the same thing as the semantic condition on $\sim$. To eliminate this redundancy we shall modify Smullyan's approach slightly by assuming that the double negation rule is incorporated into the *syntax* of any language we treat with this approach. That is, if the negation of $\sim p$ is to be formed the outcome is p, not $\sim\sim p$. In this way we may assume that the formulas of the treated language come in pairs, each the negation of the other.

In the case of propositional calculus this then means that the semantic statements can be taken to be the formulas themselves, letting the language's negation operator serve double duty in determining truth and falsity. The rules that a model of propositional calculus has to meet then are that p is present if and only if $\sim p$ is absent, and $p \vee q$ is present if and only if p or q is present.

We shall capture this sort of semantic information with the aid of a relational structure whose domain consists of theories x, y,... (sets of statements) and which has two unary operations: $\bar{x}$ and $x\vdash$, and four unary relations: consistent(x), complete(x), Hintikka(x), and closed(x). The semantic information itself is coded in $\vdash$. We call such a structure a *Hintikka structure*.

The theory $\bar{x}$ is always interpreted as the set of negations of elements of x. Since double negations cancel, we have $\bar{\bar{x}} = x$. We also have that if $x = \bar{x}$ then x partitions into disjoint blocks y, $\bar{y}$. Writing x' for the set of statements not in x, we may read $\bar{x}'$ as either the complement of the negations or the negations of the complement, without ambiguity.

The interpretation of $\vdash$ determines the semantics of the language. In the case of propositional calculus the theory $x\vdash$ consists of the literals of x (so $\vdash$ is *literal-preserving*),

together with any formulas $p \vee q$ such that either p or q is in x, and any formulas $\sim(p \vee q)$ such that both $\sim p$ and $\sim q$ are in x. Notice that $\vdash$ acts *monotonically* on theories: $x\vdash \subseteq (x \cup y)\vdash$. Intuitively $\vdash$ describes how truth values "propagate upwards" through formulas in the course of their evaluation. This intuition also applies to our use of $\vdash$ to give the semantics of PDL, below. (An earlier version of this treatment had $\vdash$ preserve *all* statements, but this was found to lead to considerable clumsiness.)

We shall sometimes write "$s \in x\vdash$" as "$x \vdash s$".

The four predicates are as follows, where x' denotes the complement of x.

x *consistent*: $\bar{x} \subseteq x'$

x *complete*: $x' \subseteq \bar{x}$

x *Hintikka*: $x \subseteq x\vdash$

x *closed*: $x\vdash \subseteq x$

The first three of these predicates on theories are standard. Hintikka theories, or Hintikka sets, are sometimes called "downward saturated" sets. Thus one might call a theory that is closed in our sense "upward saturated."

Now suppose x is both consistent and complete, so $\bar{x} = x'$. We may interpret x as defining a model by taking x to be the set of those statements assigned *true*, so that x' is the set of those assigned *false*. Such an assignment satisfies the condition that s and $\sim s$ are assigned different truth values. Suppose further that x is both Hintikka and closed, so $x\vdash = x$. It follows in the case of propositional calculus that $p \vee q$ is in x if and only if one of p or q is in x. For if p or q is in x, $p \vee q$ is in $x\vdash = x$. Conversely if $p \vee q$ is in $x = x\vdash$ then $\sim(p \vee q)$ is not in $x\vdash$ by consistency, so not both $\sim p$ and $\sim q$ are in x, so one of p or q is in x by completeness.

With this reasoning in mind we formally define a *model* to be a complete consistent closed Hintikka theory. A *model of* the formula s is a model containing s. If s has a model then s is *satisfiable*, otherwise we say that $\sim s$ is *valid*.

The sort of problem we want to solve is whether a given statement s is satisfiable. We shall show how to deal with this problem algorithmically by showing that the definition of "model" can be relaxed to just "consistent Hintikka theory" without affecting the definition of "satisfiable." Weakening the four conditions to two in this way makes the search for a model feasible.

Given that we shall be varying $\vdash$, there are certain properties of $\vdash$ that we will want to be able to rely on, in order to develop a theory of theories that is independent of the particular choice of $\vdash$. We want $\vdash$ to be:

(i) Literal-preserving: if s is a literal in x then $x \vdash s$.

(ii) Continuous: if $x_0 \subseteq x_1 \subseteq \cdots$ is an increasing chain of theories with union x then $x\vdash = x_0\vdash \cup x_1\vdash \cup \cdots$

(iii) Dual: $y\vdash \sim s$ if and only if $\forall x(x \vdash s \rightarrow x \cap \bar{y} \neq \varnothing)$.

(iv) Inductive: there exists a function h, the *height* function, mapping statements to natural numbers, such that literals are mapped to 0, and for any nonliteral s, if $x \vdash s$

for some x then there is some subset z of x such that $z \vdash\!\!\!- s$ and every element of z has height less than that of s.

Continuity implies the monotonicity we mentioned earlier, as can be seen by considering chains of length two. We use continuity in the main theorem of this section, that every consistent Hintikka set extends to a model.

To understand duality consider first the only-if direction. This may be expressed as, if $x\vdash\!\!\!- \cup\, y\vdash\!\!\!-$ is inconsistent (contains s and $\sim s$) then so is $x \cup y$; equivalently, if $x \cup y$ is consistent so is $x\vdash\!\!\!- \cup\, y\vdash\!\!\!-$. So taking $x = y$ shows that if x is consistent so is $x\vdash\!\!\!-$.

Now consider the other direction. If every theory x satisfying $x \vdash\!\!\!- s$ is inconsistent with y then we deduce that $y\vdash\!\!\!- \sim s$; equivalently, if $y \not\vdash \sim s$ then there is some theory x consistent with y (though not necessarily consistent with all of $y\vdash\!\!\!-$, $y\vdash\!\!\!-\vdash\!\!\!-$, etc.) such that $x \vdash\!\!\!- s$. Viewed in terms of evaluating formulas, this says that if there are not enough values available to deduce in one step that s has value *false*, then it is possible to supply additional values not contradicting any existing values such that in one step s turns out to have value *true*. Nothing is said here about whether those supplied values are consistent with values calculated by $y\vdash\!\!\!-$, $y\vdash\!\!\!-\vdash\!\!\!-$, etc., the supplied values are there just to make a point about one step of computation.

For example in propositional calculus s may be either a literal P, a disjunction $p \vee q$, or a conjunction $\sim(p \vee q)$. If every theory x for which $x\vdash\!\!\!-$ contains P is inconsistent with y then $\{P\}$ is inconsistent with y, so y contains $\sim P$. If every theory x for which $x\vdash\!\!\!-$ contains $p \vee q$ is inconsistent with y then $\{p\}$ and $\{q\}$ are inconsistent with y, so y contains $\sim p$ and $\sim q$, so $y\vdash\!\!\!-$ contains $\sim(p \vee q)$. If every theory x for which $x\vdash\!\!\!-$ contains $\sim(p \vee q)$ is inconsistent with y then $\{\sim p, \sim q\}$ is inconsistent with y, so y contains either p or q, so $y\vdash\!\!\!-$ contains $p \vee q = \sim\sim(p \vee q)$.

The inductive property of $\vdash\!\!\!-$ formalizes the idea that statements of the language are all built from smaller statements, with the literals being the smallest statements, and that information about values of statements flows from smaller to larger statements. To see that propositional calculus is inductive, take $h(P) = h(\sim P) = 0$, and $h(p \vee q) = h(\sim(p \vee q)) = 1 + \max(h(p), h(q))$. If our intuition about the language of propositional calculus is correct this defines a function from formulas to natural numbers. If $p \vee q$ is in $x\vdash\!\!\!-$ then either p or q is in x, so one of $\{p\}$ or $\{q\}$ supplies the appropriate subset z of x every element of which has height less than that of $p \vee q$; and similarly for $\sim(p \vee q)$.

We now use these properties of $\vdash\!\!\!-$, and the fixed definitions of $\bar{x}$ and the four predicates, to establish some computationally useful facts.

LEMMA 2.1. *If x is consistent, complete and Hintikka then x is closed.*

Proof. x not closed means that $x\vdash\!\!\!-$ contains some s not in x, so since x is complete $\sim s$ is in x. But then $\sim s$ is in $x\vdash\!\!\!-$ since x is Hintikka. So $x\vdash\!\!\!-$ is not consistent, whence x is not consistent by duality, a contradiction. ∎

So our definition of "model" is unnecessarily strong; "closed" may be omitted from the definition.

LEMMA 2.2. *If x is Hintikka so is $x\vdash\!\!\!-$.*

Proof. If $x \subseteq x\vdash$ then by monotonicity of $\vdash$, $x\vdash \subseteq x\vdash\vdash$. ∎

So if x is Hintikka and consistent, so are $x\vdash$, $x\vdash\vdash$, etc. Write $x\vdash^*$ for $\cup x\vdash^i$.
We now define x to be ***n-complete*** when $x \cup \bar{x}$ contains all statements of height at most n.

LEMMA 2.3. *If y is n-complete then $y\vdash$ is $n+1$-complete.*

Proof. Suppose $\sim s$ is of height $n+1$ but is not in $y\vdash$. We shall show that s is in $y\vdash$. By duality there is some x consistent with y for which s is in $x\vdash$. But then by inductiveness there is some $z \subseteq x$ every element of which is of height at most n, yet for which s is in $z\vdash$. Since x is consistent with y, so is z, but since y is n-complete z must therefore be contained in y, so s is in $y\vdash$ by monotonicity. ∎

THEOREM 2.4. *If x is consistent, Hintikka, and 0-complete then $x\vdash^*$ is a model.*

Proof. Immediate from Lemmas 2.1 to 2.3. ∎

A corollary of this that we shall not use directly in the sequel is that if x is consistent and Hintikka then x can be extended to a model (i.e. is a subset of a model). For x can be extended to a consistent 0-complete Hintikka theory by adjoining to x one literal from each matched pair of literals neither of which appears in x. So to test whether s is satisfiable it suffices to search for some consistent Hintikka x containing s. For example the propositional calculus formula $\sim((P \vee \sim(\sim P \vee Q)) \vee (R \vee \sim(\sim P \vee (\sim Q \vee R))))$ can be seen to be satisfiable because it is contained in the consistent Hintikka theory containing the given formula together with $\sim(P \vee \sim(\sim P \vee Q))$, $\sim P$, $\sim P \vee Q$, Q, $\sim(R \vee \sim(\sim P \vee (\sim Q \vee R)))$, $\sim R$, and $\sim P \vee (\sim Q \vee R)$. If there is no consistent Hintikka theory containing s then there is no model, since a model is itself a consistent Hintikka theory.

Theorem 3.3 below states the version of this corollary that we shall actually need for our algorithm.

3. PDL SEMANTICS

We now specify the semantic language and $\vdash$ for Propositional Dynamic Logic. We begin with the set Φ of PDL formulas and the set Σ of PDL programs, as defined in section 1. We augment the language so that Σ also includes all programs of the form $\sim a$, intended to mean that program that can do exactly what a cannot do.

In order to define a statement, so that theories and models can then be defined, we assume we have a set W of states. W is not part of the syntax of PDL but rather part of the semantics, and may change from one model to another. However once W is chosen it then determines a semantic language for PDL whose statements are taken to be the elements of $(W \times \Phi) \cup (W \times \Sigma \times W)$. We call elements (u, p) of $W \times \Phi$ *facts*, writing them as $u \models p$. We call elements (u, a, v) of $W \times \Sigma \times W$ *transitions*, writing them as $u\langle a\rangle v$. The *literals* of this semantic language are those facts and transitions whose second component is a possibly negated PDL atom, namely $u \models P$, $u \models \sim P$, $u\langle A\rangle v$, and $u\langle \sim A\rangle v$, for arbitrary u, v.

A PDL theory x may be envisaged as a graph each of whose vertices $u \in W$ is labelled with formulas p, namely those such that $(u \models p) \in x$, and each of whose edges from u to v is labelled with programs a, namely those such that $u\langle a\rangle v \in x$. Alternatively a separate labelled edge may be drawn for each transition. If the theory x is complete and consistent then for every fact $u \models p$ either p or $\sim p$ will label u, and for every transition $u\langle a\rangle v$ an edge will go from u to v labelled either a or $\sim a$.

To present $\vdash$ for PDL we adopt a tabular form of presentation, exemplified by the following table for propositional calculus.

$$p \vdash p \vee q$$
$$q \vdash p \vee q$$
$$\sim p, \sim q \vdash \sim(p \vee q)$$

In this notation we write $s_1, \ldots, s_k \vdash t$ for $\{s_1, \ldots, s_k\} \vdash t$. The function $\vdash$ defined in this way is taken to be minimal such that $x\vdash$ contains s for each literal s in x, and contains t for each condition of the above form for which the left-hand side of the condition is a subset of x. (Minimality is defined by the obvious pointwise-inclusion ordering of functions.) It is apparent that $\vdash$ defined thus is continuous. We shall arrange the tables we use so that duality and inductiveness follow from the particular tables, as in the above example. (We could save on about half of the table for PDL by making duality a requirement, but the reader would find it harder to deduce the nature of $\vdash$.)

We shall make a practice of using such abbreviations as $p \vee \sim q$ for $\sim(p \wedge q)$, so that the last line of the above table would read

$$p, q \vdash p \wedge q.$$

In the following table we abbreviate $u \models \sim\langle a\rangle p$ to $u \models [a] \sim p$.

(A cut-set of a directed graph separating vertices u, v is a minimal set of edges whose removal from the graph would eliminate all paths from u to v. K_W is the complete graph $W \times W$ on vertex set W.)

TABLE I

Standard Semantics

$\langle a\rangle$	$u\langle a\rangle v, v \models p$	$\vdash$	$u \models \langle a\rangle p$
$\langle ?\rangle$	$u \models p$	$\vdash$	$u\langle p?\rangle u$
$\langle \cup\rangle$	$u\langle a\rangle v$	$\vdash$	$u\langle a \cup b\rangle v$
$\langle \cup\rangle$	$u\langle b\rangle v$	$\vdash$	$u\langle a \cup b\rangle v$
$\langle ;\rangle$	$u\langle a\rangle v, v\langle b\rangle w$	$\vdash$	$u\langle a; b\rangle w$
$\langle *\rangle$	$u_0\langle a\rangle u_1, \ldots, u_{k-1}\langle a\rangle u_k$	$\vdash$	$u_0\langle a^*\rangle u_k \; k \geqslant 0$
$[a]$	$\{u\langle \sim a\rangle v$ or $v \models p \mid v \in W\}$	$\vdash$	$u \models [a]p$
$[?]$	$u \models \sim p$	$\vdash$	$u\langle \sim(p?)\rangle u$
$[?]$		$\vdash$	$u\langle \sim(p?)\rangle v \; (v \neq u)$
$[\cup]$	$u\langle \sim a\rangle v, u\langle \sim b\rangle v$	$\vdash$	$u\langle \sim(a \cup b)\rangle v$
$[;]$	$\{u\langle \sim a\rangle v$ or $v\langle \sim b\rangle w \mid v \in W\}$	$\vdash$	$u\langle \sim(a; b)\rangle v$
$[*]$	$\{w\langle \sim a\rangle w' \mid (w, w') \in C\}$	$\vdash$	$u\langle \sim(a^*)\rangle v$ for any cut-set C of K_W separating u, v.

The $\langle a \rangle$ rule says that if a can go from u to v, and if p holds in v, then in u it is true that a can bring about p. The $\langle ? \rangle$ rule says that if p holds in u then $p?$ can go from u to u. The next two rules say that if either a or b can go from u to v, so can $a \cup b$. Rule $\langle ; \rangle$ says that if a can go from u to v and b can go from v to w then a; b can go from u to w. Finally rule $\langle * \rangle$ says that if a can go from u_0 to u_1, and from u_1 to u_2, and so on up to u_k, then a^* can go from u_0 to u_k.

The remaining rules are present simply to make $\vdash$— dual. We may interpret them in a natural way however. The $[a]$ rule says that if for every v in W, either a cannot get from u to v or p holds in v, then in u it is the case that a guarantees p (in the sense that if a halts p must then hold). The $[?]$ rules say that if p does not hold in u then $p?$ cannot get from u to u, and that $p?$ cannot travel from one state to another (i.e. is side-effect free). The $[\cup]$ rule says that if neither a nor b can get from u to v, neither can $a \cup b$. Rule $[;]$ says that if for every state v in W, either a cannot get from u to v or b cannot get from v to w, then a; b cannot get from u to w. And the rule $[*]$ says that if every sequence of states starting with u and ending with v includes a consecutive pair w, w' such that a cannot get from w to w', then a^* cannot get from u to v.

The reader should not be in doubt that this semantics captures the appropriate meaning of the PDL constructs.

Nonstandard PDL semantics

We now modify the above semantics in several ways to facilitate the development of our tableau method. None of these modifications have to do with propositional issues—in fact the standard semantics for propositional calculus can be used directly in the derivation of the usual tableau method.

We first give a nonstandard semantics that is not quite right, though very plausible. It is the semantics one would infer from the methods of [6].

TABLE II

Tentative Nonstandard Semantics

$\langle A \rangle\_$	$u\langle A \rangle v,\ v \models p$	$\vdash\_$	$u \models \langle A \rangle p$
$\langle ? \rangle\_$	$u \models p,\ u \models q$	$\vdash\_$	$u \models \langle p? \rangle q$
$\langle \cup \rangle\_$	$u \models \langle a \rangle p$	$\vdash\_$	$u \models \langle a \cup b \rangle p$
$\langle \cup \rangle\_$	$u \models \langle b \rangle p$	$\vdash\_$	$u \models \langle a \cup b \rangle p$
$\langle ; \rangle\_$	$u \models \langle a \rangle \langle b \rangle p$	$\vdash\_$	$u \models \langle a; b \rangle p$
$\langle * \rangle\_$	$u \models p$	$\vdash\_$	$u \models \langle a^* \rangle p$
$\langle * \rangle\_$	$u \models \langle a \rangle \langle a^* \rangle p$	$\vdash\_$	$u \models \langle a^* \rangle p$
$[A]\_$	$\{u\langle \sim A \rangle v$ or $v \models p \mid v \vdash W\}$	$\vdash\_$	$u \models [A]p$
$[?]\_$	$u \models q$	$\vdash\_$	$u \models [p?]q$
$[?]\_$	$u \models \sim p$	$\vdash\_$	$u \models [p?]q$
$[\cup]\_$	$u \models [a]p,\ u \models [b]p$	$\vdash\_$	$u \models [a \cup b]p$
$[;]\_$	$u \models [a][b]p$	$\vdash\_$	$u \models [a; b]p$
$[*]\_$	$u \models p,\ u \models [a][a^*]p$	$\vdash\_$	$u \models [a^*]p.$

The rule $\langle A\rangle$ restates rule $\langle a\rangle$ of the standard semantics, restricted to atomic programs, and similarly $[A]_-$ restates $[a]$. These are the only rules of this semantics that mention more than one state. Rule $\langle ?\rangle_-$ says that if p and q hold then $\langle p?\rangle q$ holds, always in the same state u. The two $\langle\cup\rangle_-$ rules say that if $\langle a\rangle p$ or $\langle b\rangle p$ hold then so does $\langle a \cup b\rangle p$. The $\langle;\rangle_-$ rule says that if $\langle a\rangle\langle b\rangle p$ holds then so does $\langle a; b\rangle p$. The $\langle *\rangle$ rules say that if either p or $\langle a\rangle\langle a^*\rangle p$ hold then so does $\langle a^*\rangle p$. The $[]_-$ rules are the obvious duals of these rules.

The problem with this semantics lies entirely with the $\langle *\rangle_-$ rules, which are too permissive. $\langle a\rangle\langle a^*\rangle p$ may hold in every state, whence $\langle a^*\rangle p$ would also hold in every state, yet p may still hold in no state! This is clearly inconsistent with the standard semantics for $\langle a^*\rangle p$, which requires the existence of a path of a's leading to a state in which p holds. What is missing is some form of an *induction rule.*

One way to deal with this problem is to retain the standard semantics of the nonliteral programs ($\langle;\rangle$ etc.), and replace the two $\langle *\rangle_-$ rules with the one rule:

$$\langle *\rangle_- \qquad u\langle a^*\rangle v,\ v \models p \qquad \vdash_- \qquad u \models \langle a^*\rangle p$$

We shall adopt another approach, which avoids reintroducing the standard semantics of the nonliteral programs, at the price of introducing a third type of statement, the *link* $s \Rightarrow t$ where s and t are facts. The role of links is to keep track of paths between states that would otherwise have to be looked after by the semantics of the nonliteral programs. We present the nonstandard semantics based on links, then discuss the role of the links.

The $\langle\Rightarrow\rangle_+$ rule says that if there is a chain of links from the fact $u \models \langle a\rangle p$ to the fact $v \models p$ then the fact $u \models \langle a\rangle p$ is true. Thus we should expect that links are taking over

TABLE III

Nonstandard Semantics

$\langle\Rightarrow\rangle_+$	$u \models \langle a\rangle p \Rightarrow^* v \models p$	$\vdash_+$	$u \models \langle a\rangle p$
$\langle A\rangle_+$	$u\langle A\rangle v,\ v \models p$	$\vdash_+$	$u \models \langle A\rangle p \Rightarrow v \models p$
$\langle ?\rangle_+$	$u \models p,\ u \models q$	$\vdash_+$	$u \models \langle p?\rangle q \Rightarrow u \models q$
$\langle\cup\rangle_+$	$u \models \langle a\rangle p$	$\vdash_+$	$u \models \langle a \cup b\rangle p \Rightarrow u \models \langle a\rangle p$
$\langle\cup\rangle_+$	$u \models \langle b\rangle p$	$\vdash_+$	$u \models \langle a \cup b\rangle p \Rightarrow u \models \langle b\rangle p$
$\langle;\rangle_+$	$u \models \langle a\rangle\langle b\rangle p$	$\vdash_+$	$u \models \langle a; b\rangle p \models u \models \langle a\rangle\langle b\rangle p$
$\langle *\rangle_+$	$u \models p$	$\vdash_+$	$u \models \langle a^*\rangle p \models u \models p$
$\langle *\rangle_+$	$u \models \langle a\rangle\langle a^*\rangle p$	$\vdash_+$	$u \models \langle a^*\rangle p \Rightarrow u \models \langle a\rangle\langle a^*\rangle p$
$[A]_+$	$\{u\langle\sim A\rangle v$ or $v \models p \mid v \vdash W\}$	$\vdash_+$	$u \models [A]p$
$[?]_+$	$u \models q$	$\vdash_+$	$u \models [p?]q$
$[?]_+$	$u \models \sim p$	$\vdash_+$	$u \models [p?]q$
$[\cup]_+$	$u \models [a]p,\ u \models [b]p$	$\vdash_+$	$u \models [a \cup b]p$
$[;]_+$	$u \models [a][b]p$	$\vdash_+$	$u \models [a; b]p$
$[*]_+$	$u \models p,\ u \models [a][a^*]p$	$\vdash_+$	$u \models [a^*]p$

the role of transitions in keeping track of relevant paths. How this is accomplished can be seen in part by considering the remaining rules. All the $\langle\ \rangle$ rules have only a link on their right hand side. In general the rule will have the form $s \vdash\!\!\!-_+ t \Rightarrow s$, corresponding to the previous semantics, which had $s \vdash\!\!\!-_+ t$. The idea is that we shall be a little more cautious about admitting t given s. We first admit a link $t \Rightarrow s$, and then if a chain of links from t to an appropriate fact can be found we will admit t itself. This caution makes itself felt mainly on the $\langle * \rangle$ rules, which now cannot result in making $\langle a^* \rangle p$ true without more concrete evidence of the existence of a path of a's.

For example, suppose we have the literals $u\langle A\rangle v$, $v\langle B\rangle w$, and $w \models P$ in a theory x. (So x is Hintikka since $\vdash\!\!\!-$ is literal-preserving, so $x \subseteq x\vdash\!\!\!- \subseteq x\vdash\!\!\!-^2 \subseteq \cdots$) We would expect that $u \models \langle A; B\rangle P$ should be in $x\vdash\!\!\!-^n$ for some n. Rule $\langle A\rangle_+$ puts $v \models \langle B\rangle P \Rightarrow w \models P$ into $x\vdash\!\!\!-$, along with other facts and links we do not care about. Then $\langle \Rightarrow\rangle_+$ puts $v \models \langle B\rangle P$ into $x\vdash\!\!\!-^2$. Now $\langle A\rangle_+$ puts $u \models \langle A\rangle\langle B\rangle P \Rightarrow v \models \langle B\rangle P$ into $x\vdash\!\!\!-^3$. Rule $\langle \Rightarrow\rangle_+$ then puts $u \models \langle A\rangle\langle B\rangle P$ into $x\vdash\!\!\!-^4$. By rule $\langle ;\rangle_+$ $x\vdash\!\!\!-^5$ gets $u \models \langle A; B\rangle P \Rightarrow u \models \langle A\rangle\langle B\rangle P$. But then $x\vdash\!\!\!-^5$ contains the chain $u \models \langle A; B\rangle P \Rightarrow u \models \langle A\rangle\langle B\rangle P \Rightarrow v \models \langle B\rangle P \Rightarrow w \models P$, so rule $\langle \Rightarrow\rangle_+$ puts $u \models \langle A; B\rangle P$ in $x\vdash\!\!\!-^6$. So our objective has been reached, in 6 steps of computation.

The other $\langle\ \rangle_+$ rules may be exercised in an analogous manner, for example to show that $v \models \langle A \cup B\rangle P$, $w \models \langle B^*\rangle P$, and $v \models \langle B^*\rangle P$ will all eventually be added to x by $\vdash\!\!\!-_+$.

By now the reader should suspect that if x starts out with various literal facts and transitions then $x\vdash\!\!\!-^*$ and $x\vdash\!\!\!-_+^*$ will end up with the same facts. The following lemmas state and prove this intuition formally.

We define $+-$Hintikka to mean Hintikka with respect to the nonstandard semantics.

The observant reader will have noticed that $\vdash\!\!\!-_+$ is not inductive, thanks to the rules $[*]_+$ and $\langle * \rangle_+$. Also it is not dual since negative links have not been treated. However this is a minor detail; we could easily make it dual by adding further rules, or just saying that the $\vdash\!\!\!-$ defined was the least dual one satisfying the conditions; in fact it will turn out that we do not need duality for $\vdash\!\!\!-_+$ but only for the standard semantics $\vdash\!\!\!-$.

We now wish to show that, for the purposes of defining satisfiability of a PDL formula, the nonstandard semantics is equivalent to the standard semantics. Our strategy will be as follows. Let h be the height function for the standard semantics, let x_n be defined as $\{s \in x \mid h(s) \leqslant n\}$, and let x_ω be the part of x containing only facts and literal transitions. Taking $\hat{x}$ as $x_0 \cup (V - (x_0 \cup \bar{x}_0))$ where V is the set of all propositional and program variables, we shall prove that if x is consistent and $+-$Hintikka, then $x_\omega \subseteq \hat{x}\vdash\!\!\!-^*$. By observing that $\hat{x}$ is consistent, 0-complete and Hintikka we infer that if $u \models p \in x$ then $\hat{x}\vdash\!\!\!-^*$ is a model of $u \models p$, whence p is satisfiable. We also prove the converse: if $u \models p$ has a model it occurs in some consistent $+-$Hintikka theory. This gives us a useful test for satisfiability: see whether $u \models p$ belongs to some consistent $+-$Hintikka theory.

The double induction of the proof of the main theorem has a sufficiently delicate "inner induction" that we isolate it as two lemmas, which in essence treat the two directions of the inner induction. For each of these lemmas we assume (as part of the "outer" induction hypothesis) that x is $+-$Hintikka and that n is a positive integer such that for $0 \leqslant m \leqslant n$, $x_m \subseteq \hat{x}\vdash\!\!\!-^m$, and $\hat{x}\vdash\!\!\!-^m$ is consistent, m-complete and Hintikka.

The proofs are not unduly long by line count, but they are rather tedious to check, and the reader may prefer to skip the details on a first reading of the paper.

LEMMA 3.1. *If* $u \models [c]p \in x$ *and* $h(c) \leqslant n$, *then for all* $v \in W$ *either* $u\langle\sim c\rangle v \in \mathfrak{A}\!\!\!-^n$ *or* $v \models p \in x$.

Proof. We use induction on $h(c)$.

$u \models [A]p \in x$. By $[A]_+$, $u\langle\sim A\rangle v$ or $v \models p \in x$ for all $v \in W$. Since $h(\sim A) = 0$, $u\langle\sim A\rangle v \in x_0 \subseteq \mathfrak{x} \subseteq \mathfrak{A}\!\!\!-^n$ or $v \models p \in x$ for all $v \in W$.

$u \models [p\,?]q \in x$. By $[\,?]$, for all $v \neq u$ $u\langle\sim(p\,?)\rangle v \in \mathfrak{A}\!\!\!- \subseteq \mathfrak{A}\!\!\!-^n$. For $v = u$, we have by $[\,?]_+$ $u \models \sim p$ or $u \models q \in x$. If the latter we are done. Otherwise, since $h(\sim p) \leqslant n - 1$, $u \models \sim p \in x_{n-1} \subseteq \mathfrak{A}\!\!\!-^{n-1}$, so by $\langle\,?\rangle$ $u\langle\sim(p\,?)\rangle u \in \mathfrak{A}\!\!\!-^n$.

$u \models [a \cup b]p \in x$. By $[\cup]_+$ $u \models [a]p$, $u \models [b]p \in x$. So by induction, for all $v \in W$ $u\langle\sim a\rangle v \in \mathfrak{A}\!\!\!-^{n-1}$ or $v \models p \in x$, and for all $v \in W$ $u\langle\sim b\rangle v \in \mathfrak{A}\!\!\!-^{n-1}$ or $v \models p \in x$. Thus for all $v \in W$ $u\langle\sim a\rangle v$, $u\langle\sim b\rangle v \in \mathfrak{A}\!\!\!-^{n-1}$ or $v \models p \in x$. So by $[\cup]$, for all $v \in W$ $u\langle\sim(a \cup b)\rangle v \in \mathfrak{A}\!\!\!-^n$ or $v \models p$.

$u \models [a; b]p \in x$. By $[;]_+$ $u \models [a][b]p \in x$, so by induction $u\langle\sim a\rangle v \in \mathfrak{A}\!\!\!-^{n-1}$ or $v \models [b]p \in x$ for all v, and so again by induction $u\langle\sim a\rangle v \in \mathfrak{A}\!\!\!-^{n-1}$ or $v\langle\sim b\rangle w \in \mathfrak{A}\!\!\!-^{n-1}$ or $w \models p \in x$ for all v, w. But then $u\langle\sim(a; b)\rangle w \in \mathfrak{A}\!\!\!-^n$ or $w \models p \in x$ for all w.

$u \models [a^*]p \in x$: Suppose that for some $v \in W$, $u\langle\sim(a^*)\rangle v \notin \mathfrak{A}\!\!\!-^n$ and $v \models p \notin x$. Since $h(u\langle\sim a^*\rangle v) \leqslant n$ and $\mathfrak{A}\!\!\!-^n$ is n-complete, $u\langle a^*\rangle v \in \mathfrak{A}\!\!\!-^n$, i.e. $(\mathfrak{A}\!\!\!-^{n-1}) \vdash u\langle a^*\rangle v$. Since $\mathfrak{x}\!\!\!-^{n-1}$ is Hintikka, $u_0\langle a\rangle u_1, \ldots, u_{k-1}\langle a\rangle u_k \in \mathfrak{A}\!\!\!-^{n-1}$ for some $u_0, \ldots, u_k$, $k \geqslant 0$, where $u_0 = u$ and $u_k = v$, by $\langle^*\rangle$. Now by $[^*]p$, $u_0 \models p$, $u_0 \models [a][a^*]p \in x$. So by induction either $u_0\langle\sim a\rangle u_1 \in \mathfrak{A}\!\!\!-^{n-1}$ or $u_1 \models p \in x$. The former is ruled out by the presence in $\mathfrak{A}\!\!\!-^{n-1}$ of $u_0\langle a\rangle u_1$ and the consistency of $\mathfrak{A}\!\!\!-^{n-1}$. We may continue in this way to show $u_2 \models p, \ldots, u_k \models p \in x$ by induction on k, contradicting $v \models p \notin x$ since $v = u_k$. ∎

LEMMA 3.2. *For all formulas of the form* $Lp = \langle c_1\rangle\langle c_2\rangle \cdots \langle c_g\rangle p$, $g \geqslant 0$, *if* $u \models \langle c\rangle Lp \Rightarrow^* w \models p \in x$ *and* $h(c) \leqslant n$, *there exists* $v \in W$ *such that* $u \models \langle c\rangle Lp \Rightarrow^* v \models Lp \Rightarrow^* w \models p \in x$ *and* $u\langle c\rangle v \in \mathfrak{A}\!\!\!-^n$.

Proof. We use induction on $h(c)$, assuming $u \models \langle c\rangle Lp \Rightarrow^* w \models p \in x$.

$c = A$. Then $u \models \langle A\rangle Lp \Rightarrow v \models Lp \Rightarrow^* w \models p \in x$ and $u\langle A\rangle v \in x$, by $\langle A\rangle$. Since $h(A) = 0$, $u\langle A\rangle v \in x_0 \subseteq \mathfrak{A}\!\!\!-^n$.

$c = p\,?$. Then $u \models \langle p\,?\rangle Lp \Rightarrow u \models Lp \Rightarrow^* w \models p \in x$, and $u \models p \in x$, by $\langle\,?\rangle$. So $u \models p \in x_{n-1} \subseteq \mathfrak{A}\!\!\!-^{n-1}$ by induction, whence $u\langle p\,?\rangle u \in \mathfrak{A}\!\!\!-^n$ by $\langle\,?\rangle$.

$c = a \cup b$. Then without loss of generality $u \models \langle a \cup b\rangle Lp \Rightarrow u \models \langle a\rangle Lp \Rightarrow^* v \models Lp \Rightarrow^* w \models p \in x$ and $u\langle a\rangle v \in \mathfrak{A}\!\!\!-^{n-1}$, by $\langle\cup\rangle$ and induction. So $u\langle a \cup b\rangle v \in \mathfrak{A}\!\!\!-^n$ by $\langle\cup\rangle$.

$c = a; b$. Then $u \models \langle a; b\rangle Lp \Rightarrow u \models \langle a\rangle\langle b\rangle Lp \Rightarrow^* v \models \langle b\rangle Lp \Rightarrow^* v' \models Lp \Rightarrow^* w \models p \in x$ and $u\langle a\rangle v$, $v\langle b\rangle v' \in \mathfrak{A}\!\!\!-^{n-1}$, by $\langle;\rangle$ and induction (twice). So $u\langle a; b\rangle v' \in \mathfrak{A}\!\!\!-^n$ by $\langle;\rangle$.

$c = a^*$. Then, taking $u_0 = u$, $u_0 \models \langle a^* \rangle Lp \Rightarrow u_0 \models \langle a \rangle \langle a^* \rangle Lp \Rightarrow^* u_1 \models \langle a^* \rangle Lp \Rightarrow u_1 \models \langle a \rangle \langle a^* \rangle Lp \Rightarrow^* u_2 \models \cdots \Rightarrow u_m \models \langle a^* \rangle Lp \Rightarrow u_m \models Lp \Rightarrow^* w \models p \in x$ and $u_0 \langle a \rangle u_1$, $u_1 \langle a \rangle u_2, \ldots, u_{m-1} \langle a \rangle u_m \in \mathfrak{X}\!\!\vdash\!\!\!-^{n-1}$, by $\langle * \rangle$ and induction ($m \geqslant 0$ times). Since $u \models \langle c \rangle Lp \Rightarrow^* w \models p$ is finite, m must be too. Hence $u_0 \langle a^* \rangle u_m \in \mathfrak{X}\!\!\vdash\!\!\!-^n$ by $\langle * \rangle$. ▮

THEOREM 3.3. *Let x be consistent and $+-$Hintikka. Then x_ω can be extended to a PDL model.*

Proof. We first show by induction that for all $n \geqslant 0$, $x_n \subseteq \mathfrak{X}\!\!\vdash\!\!\!-^n$.

Certainly $x_0 \subseteq \mathfrak{X}\!\!\vdash\!\!\!-^0$, by construction of $\mathfrak{X}$. Now suppose that the inductive claim holds for n, and that $s \in x_{n+1}$. If s is $u \models [c]p$ with $h(\sim c)$, $h(p) \leqslant n$ then by lemma 3.1 $u \langle \sim c \rangle v \in \mathfrak{X}\!\!\vdash\!\!\!-^n$ or $v \models p \in x$ for all $v \in W$, and $v \models p \in x_n \subseteq \mathfrak{X}\!\!\vdash\!\!\!-^n$ by induction. So by $[a]$, $u \models [c]p \in \mathfrak{X}\!\!\vdash\!\!\!-^{n+1}$. If s is $u \models \langle c \rangle p$ with $h(c)$, $h(p) \leqslant n$ then by $\langle \Rightarrow \rangle$, for some $w \in W$ $u \models \langle a \rangle p \Rightarrow^* w \models p$. So by lemma 3.2 (taking $g = 0$) there exists v such that $u \models \langle c \rangle p \Rightarrow^* v \models p \in x$ and $u \langle c \rangle v \in \mathfrak{X}\!\!\vdash\!\!\!-^n$. So by $\langle \Rightarrow \rangle$, $s \Rightarrow v \models p$ for some $s \in \Phi$, so by the appropriate $\langle\ \rangle_+$ condition $v \models p \in x_n \subseteq \mathfrak{X}\!\!\vdash\!\!\!-^n$. Hence $u \models \langle c \rangle p \in \mathfrak{X}\!\!\vdash\!\!\!-^{n+1}$. This establishes that $x_{n+1} \subseteq \mathfrak{X}\!\!\vdash\!\!\!-^{n+1}$.

It follows that $x_\omega \subseteq \mathfrak{X}\!\!\vdash\!\!\!-^*$. Since $\mathfrak{X}$ is consistent (since x is), 0-complete (by construction), and Hintikka, $\mathfrak{X}\!\!\vdash\!\!\!-^*$ is a model by Theorem 3.4. ▮

THEOREM 3.4. *If x is a PDL model then x can be extended to a consistent $+-$Hintikka theory merely by adding links.*

Proof. The reader may verify that the $[\ ]_+$ conditions are all met by any PDL model x. The links to be added to x to meet the $\langle\ \rangle_+$ conditions are $u \models \langle A \rangle p \Rightarrow v \models p$ whenever $u \langle A \rangle v$, $v \models p$, $u \models \langle A \rangle p \in x$, $u \models \langle p? \rangle \Rightarrow u \models q$ whenever $u \models p$, $u \models q$, $u \models \langle p? \rangle q \in x$, and similarly for the links named in the other $\langle\ \rangle_+$ conditions. These additions satisfy $\langle A \rangle_+$ through $\langle * \rangle_+$ by construction. The reader may verify that the additions also satisfy $\langle \Rightarrow \rangle_+$ automatically. ▮

It follows that $u \models p$ is satisfiable if and only if $u \models p \in x$ for some consistent $+-$Hintikka theory x, by Theorems 3.3 and 3.4 respectively. Henceforth we shall work only with $\vdash\!\!\!-_+$ semantics, permitting us to drop the $+$ without ambiguity.

4. TABLEAUX

In section 2 we disposed of the properties of being closed and complete, leaving the properties of being consistent and Hintikka. We now show how to use the classical method of tableaux [2, 9, 22, 23], via the nonstandard semantics, to test for satisfiability of PDL formulas. The tableau method has two parts: tableau construction, which takes care of the Hintikka property, and tableau testing, which takes care of consistency. We begin with easy-to-grasp but not very effective constructions and tests, and gradually make them more effective to yield the final algorithm.

A *tableau for* the PDL formula r is a rooted unordered tree whose vertices are labelled with finite theories, the root being labelled with $w \models r$ for an arbitrarily chosen state w. We use the terms "parent," "child," "sibling," for the obvious relationships among vertices, using "descendant" for the reflexive transitive closure of the "child" relation.

A *representative* of a Hintikka theory x is a theory y such that there exists a renaming of the states in y making $y \subseteq x$. (The renaming need not keep distinct states distinct, but it should rename all occurrences of a given state to the same new state.) The properties we want a tableau to have are that the union of the theories along any path of a tableau is a Hintikka theory, and every Hintikka theory has a representative corresponding to some path. We now give a construction for a tableau for r that has this property.

There will be no fixed W during this construction; instead each step of the construction will take W to be whatever states have been needed so far for the construction.

We will grow the tableau starting from the root and adding to the leaves. Begin by selecting an arbitrary state w (drawn from some suitable supply such as the natural numbers) and making $\{w \models r\}$ the root of the tableau.

We now describe the process of *extending* a leaf V of the tableau, by adding V's children to the tableau. Suppose V is labelled with theory x. We shall assume that x is finite and has some order $<$ on its statements. The purpose of the order is to prescribe a rule for selecting elements of x, which we can imagine form a *queue*.

If x is Hintikka V has no children. Otherwise the children of V are determined by the first s in x (under the order $<$ on x) not in x—. For each minimal z (minimal under set inclusion) such that $z \vdash\!\!- s$ construct a child V' of V whose theory is $x \cup z$, ordered so that $x - s < z < s$. (Thus z goes on the end of the queue, then s is moved to the end of the queue.) The order on z can be arbitrary.

The minimal z's will be precisely the sets matching the left hand side of the non-standard semantics rule having s on its right hand side (see Table III). In most cases z will have one or two statements, and there will be one or two z's. Thus if s were $u \models [p?]q$ then there would be two singleton z's, $\{u \models q\}$ and $\{u \models \sim p\}$. If s were $u \models [a^*]p$ then there would be just one z, namely $\{u \models p,\ u \models [a][a^*]p\}$. If s were the link $u \models \langle a^* \rangle p \Rightarrow u \models \langle a \rangle \langle a^* \rangle p$ then there would be just $\{u \models \langle a \rangle \langle a^* \rangle p\}$.

A more complicated case is when s is of the form $u \models [A]p$. In this case we take W to consist of all states mentioned in facts, statements, or links of theories labelling vertices from here back to the root of the tableau. This will be a finite set, since the theories are all finite by construction and the path itself is finite (each step of the process happens after a finite time). If this W has n states then there will be 2^n z's, each determined by whether it contains $u \langle \sim A \rangle v$ or $v \models p$ for each v in W.

The least obvious case is when s has the form $u \models \langle a \rangle p$. Then there will be infinitely many z's, each consisting of a finite chain $s \Rightarrow s_1 \Rightarrow s_2 \Rightarrow \cdots \Rightarrow v \models p$ of links. In each such chain all states except u are new, that is, do not appear along the path back to the root; we may assume that all these new states are drawn from the natural numbers, or some equally large set. The intermediate facts in the chain may be arbitrary.

LEMMA 4.1. *The union of theories along any path of a tableau constructed in this way is a Hintikka theory.*

Proof. Suppose not. Let x be such a union with $x \not\vdash s$ for some $s \in x$. If s is not of the form $u \models [A]p$ then $y \not\vdash s$ for every y along the path in question. At some point along the path s appears, say with m statements ahead of it on the queue. The construction ensures that within m steps s will be attended to, guaranteeing that $x \vdash s$ in the union. If s is of the form $u \models [A]p$, the construction may not deal fully with s at any one step because not all of W is present. Now suppose $x \not\vdash s$ on account of the absence of both $u\langle\sim A\rangle v$ and $v \models p$ for some v appearing in x. When v first appears on the path, s immediately becomes eligible for processing, and will be attended to within m steps, where m is the number of statements ahead of s on the queue as before. When s appears one of $u\langle\sim A\rangle v$ or $v \models p$ will be added to the theory, a contradiction. ▮

LEMMA 4.2. *Every Hintikka theory containing $w \models r$ has a representative that is the union of theories along some path of the tableau constructed above.*

Proof. Let y be a Hintikka theory containing $w \models r$. We show by induction on i that there exists a path such that every theory $x_0, x_1, \ldots$ on that path is a representative of y. This is immediate for the root theory $x_0 = \{w \models r\}$. Now suppose $x_i' \subseteq y$ where x_i' is x_i labelling V with states renamed appropriately. If x_i is Hintikka we are at a leaf and x_i is the desired representative union. Otherwise let s be the element of x_i chosen by the tableau for processing at V. Let z be such that $x_i \cup z$ labels a child of V and $z' \subseteq y$ for some renaming of states consistent with the renaming of x_i to x_i'. Such a z must exist since $x_i' \subseteq y$ and y is Hintikka. The choice of z ensures that x_{i+1} is a representative of y.

It follows that $x_0 \cup x_1 \cup \cdots$ as constructed is a representative of y. ▮

Thus our tableau construction amounts to a thorough search for Hintikka theories. Every path in the tableau yields a Hintikka theory, and every Hintikka theory containing $w \models r$ is represented by some path. In this way we have taken care of the Hintikka property, leaving only the issue of consistency.

A Test for Satisfiability

We call a test of satisfiability of statements *sound* when it never claims that a satisfiable statement is unsatisfiable, and *complete* when it never claims that an unsatisfiable statement is satisfiable. These terms are motivated by the application of a satisfiability tester to validity checking, where their meaning coincides with normal usage.

Our basic test is that r is satisfiable if and only if the union of the theories along some path of a tableau for r is consistent. Completeness follows from the fact that if such a consistent theory is found, it must also be Hintikka by lemma 4.1 and contain $w \models r$, whence r is satisfiable. To verify soundness, suppose that r is satisfiable. Then there exists a consistent Hintikka theory y containing $w \models r$, whence some path yields a representative of y by lemma 4.2. If the representative is inconsistent then so is the result of renaming states in that representative, whence so is y, a contradiction. Thus the representative is consistent.

Inspecting all paths is less convenient than inspecting all vertices, or even all vertices

standing in the descendant relation to one another. Define the predicate *bad* on vertices to be the least predicate such that bad(V) holds when:

(i) V's label is inconsistent, or

(ii) all V's children are bad, unless V has no children.

The least predicate satisfying these conditions may be formed as the intersection of all predicates satisfying them; the intersection can readily be seen to satisfy the conditions.

LEMMA 4.3. *In the tableau constructed above for r, r is satisfiable if and only if the root is not bad.*

Proof. (If.) If the root is not bad there must be a path of good vertices from the root, the union of whose theories must then be consistent. The union is a Hintikka theory by Lemma 4.1, and contains $w \models r$ by construction, so by Theorem 3.3 can be extended to a model. Hence r is satisfiable.

(Only if.) If r is satisfiable then by Theorem 3.4 there is a consistent Hintikka theory y such that $w \models r \in y$. Since y is Hintikka there is a path in the tableau the union of whose theories represents y. Since y is consistent so is the union. Now suppose some vertex on this path were bad. Then we could find a lesser badness predicate just by making all vertices on this path good. It should be clear that the resulting badness predicate is lesser, and satisfies the above conditions (i) and (ii). ▮

The "if" direction corresponds to completeness, the "only if" direction to soundness.

The Marking Procedure

We would like to identify the bad vertices by a process that takes ω stages to mark the bad vertices. In stage 0 it marks the vertices whose theories are inconsistent. In stage $i+1$ it marks those vertices all of whose children if any have been marked at earlier stages (not marking if there are no children) and all vertices labelled with a theory containing $u \models \langle a \rangle p$ with no descendant labelled with a theory in which $u \models \langle a \rangle p$ closes. Then the root is bad if and only if it is marked by this procedure. (We defer proof of correctness of the marking procedure until we have the tableau in its final form.) This is our basic marking procedure, almost in its final form despite the fact that a quite different kind of tableau will be used ultimately.

It now remains to make the process finite. We shall achieve this in two steps. First we modify the way in which we deal with chains in connection with rule $\langle \Rightarrow \rangle$. Then take advantage of a theorem due to Fischer and Ladner [6] that says that only finitely many distinct formulas will appear in the tableau constructed for r by this modified method. It then becomes possible to represent the entire tableau as a finite object by introducing loops. With the tableau represented in this way the above marking procedure then requires only finitely many steps.

Dealing with Chains

We modify the rule $\langle \Rightarrow \rangle$ used in constructing tableaux to:

$\langle \Rightarrow' \rangle$ $\qquad u \models \langle a \rangle p \Rightarrow t \vdash u \models \langle a \rangle p.$

That is, $u \models \langle a \rangle p$ now calls only for the first link, not the whole chain connecting $u \models \langle a \rangle p$ to $v \models p$. With this modification we lose the completeness property of tableaux that every path yields a theory that is Hintikka with respect to the unmodified conditions. On the other hand we do retain the soundness property that for every Hintikka y such that $x_0 \subseteq y$ some path yields a subset of y. This is because of the "self-propagating" nature of intermediate links of a chain $u \models \langle a \rangle p \Rightarrow s_1$, $s_1 \Rightarrow s_2, \ldots, s_{k-1} \Rightarrow v \models p$. Every s_i except possibly the final $v \models p$ is of the form $u' \models \langle a' \rangle p'$, and hence there must be some path which eventually accumulates all the links of such a chain, by inspection of the semantics.

We restore completeness as follows. Define $u \models \langle a \rangle p$ to *complete* in x when there is a chain of links $u \models \langle a \rangle p \Rightarrow^* v \models p$ in x for some v. Define x to be *chain-complete* when for every $u \models \langle a \rangle p \subseteq x$, $u \models \langle a \rangle p$ completes in x. Modify the path-oriented test to consider only paths the union of whose labels is chain-complete. The modified test can now be seen to be sound and complete, as the tableau overlooks no possible way of achieving chain-completeness.

It can be shown that if suffices to require every $u \models \langle a^* \rangle p$ in x to complete, which leads to a more efficient test. We do not explore this issue here.

The corresponding modification to our vertex-oriented test is to add a third alternative to the definition of *bad*.

(iii) $u \models \langle a \rangle p$ is in the theory labelling V but every descendant of V in whose theory $u \models \langle a \rangle p$ completes is bad.

That this new criterion for badness is well-defined follows from the fact that the set of all badness predicates satisfying the disjunction of (i), (ii), and (iii) is closed under arbitrary intersection and so has a minimal element, the intended badness predicate.

The new criterion also restores Lemma 4.3 when $\langle \Rightarrow' \rangle$ replaces $\langle \Rightarrow \rangle$. The "only if" (soundness) direction of the proof goes through by the above remarks about intermediate links of a chain propagating themselves. The "if" direction (completeness) follows because condition (iii) amounts to requiring that there exist a path from any good vertex containing $u \models \langle a \rangle p$ to a good vertex where $u \models \langle a \rangle p$ completes. Hence we can construct a path from the root in such a way that every $u \models \langle a \rangle p$ completes. To do this, follow good vertices down from the root until a vertex V is reached where $u \models \langle a \rangle p$ appears in the theory at V. Now choose some good vertex below V where $u \models \langle a \rangle p$ completes; such a vertex must exist by condition (iii) on badness. The path goes down to this vertex. Every vertex along this path must be good, otherwise the lowest bad vertex on this path would be labelled with an inconsistent theory, whence so would the vertex at the end of the path, contradicting its goodness. Now if there are any other facts $u' \models \langle a' \rangle p'$ in the theory *at* V (not at the current vertex) then continue this path further to another vertex where $u' \models \langle a' \rangle p'$ completes, and so on until all facts of the form $u \models \langle a \rangle p$ at V have been taken care of. Now resume following good vertices until some new fact of this form appears, and repeat the above procedure for it and all its colleagues of that form in the theory of that vertex. This process defines a path in the tableau which yields a theory in which every fact of the form $u \models \langle a \rangle p$ completes, which therefore is Hintikka with respect to the original $\vdash_+$ semantics containing $\langle \Rightarrow \rangle_+$.

Finiteness

We have now accomplished as much as is possible without the following theorem, which we need to make our tableau construction into an algorithm that always terminates, and to make our marking procedure both effective and terminating.

THEOREM 4.4. *If r is of length n, $|\{p \mid \exists u(u \models p$ appears in a tableau for $r)\}| \leqslant n$.*

Note that a tableau for r is one constructed according to the above rules, and as such will be quite limited as to what $u \models p$'s it will contain. This is essentially lemma 3.2 of [6], to which we refer the reader for a proof.

This leads us to the observation that, for any u, at most n distinct statements $u \models p$ can appear in a tableau for a formula r of length n. A second observation is that any statement $v \models p \neq w \models r$ in the tableau can be attributed to one of only three sources: $u \models \langle A\rangle p \Rightarrow v \models p$, $u \models [A]p$, or some statement naming no state other than v. We shall arrange things so that these three sources are processed in three batches in the order given. In this way all statements of the form $u \models p$ and $u \models p \Rightarrow t$ will be formed while processing these three batches. Furthermore, processing of $u \models \langle A\rangle p \Rightarrow v \models p$ and $u \models [A]p$ will be left until the end of the third batch, as they belong to the first and second batches respectively of other states. We postpone a more specific account to after the next step.

Lean Tableaux

Our next objective is to reduce the theories in the tableau to the point where the only statements in a theory are of the form $u \models p$ and $u \models p \Rightarrow t$, for fixed u; we call such a theory *a theory of u*.

To achieve this requires not merely reducing the theories but also splitting a simple vertex into as many vertices as are required for each to carry a theory of a single state and still have the new tableau retain the information in the old tableau.

We make these vague notions more precise by introducing the concept of a full theory, which is one that is "as Hintikka as a theory of a state can be." More formally, a theory x of state u is *full* when $x \vdash s$ for all $s \in x$ except those of the form $u \models \langle A\rangle p \Rightarrow v \models p$ and $u \models [A]p$, and otherwise is *partial*. A full (partial) vertex is one so labelled.

We introduce a new kind of tableau, the *lean tableau*, in which $x_i \subseteq x_{i+1}$ is no longer guaranteed for consecutive theories along a path. Lean tableaux enjoy the following properties.

(i) The root as before is labelled with $\{w \models r\}$, w arbitrary.

(ii) For each vertex V labelled with a partial theory x there exists some $s \in x$ not of the form $u \models \langle A\rangle p \Rightarrow v \models q$ or $u \models [A]p$ such that the descendants of V are each labelled with some minimal theory z such that $z \vdash s$ and $x \subseteq z$, and every such minimal theory labels some descendant.

(iii) For each vertex V labelled with a full theory x each statement $u \models \langle A\rangle p \Rightarrow v \models p \in x$ corresponds to a descendant of V whose theory is $\{v \models p\} \cup \{v \models q \mid u \models [A]q \in x\}$.

Note that lean tableaux contain no transitions. Moreover, a partial vertex has either one or two children while a full vertex may have up to n children where n is the length of r.

We now describe a test for satisfiability in terms of lean tableaux. The test is just a straightforward modification of the one we used on ordinary tableaux. A *fat* path in a lean tableau is a maximal set of paths such that any two paths have a common initial segment terminating in a full vertex. Thus for any full vertex appearing in a fat path, all the children of that vertex also appear in the path, while for any partial vertex appearing, exactly one child also appears. The point of fat paths is that they correspond to ordinary paths in ordinary tableaux.

LEMMA 4.5. *In a lean tableau, if the union of the labels of a fat path is chain-complete it is a Hintikka theory.*

Proof. The construction of lean tableaux is such that only nonstandard conditions $[A]$, $\langle\Rightarrow\rangle$ and $\langle A\rangle$ might be violated. $\langle\Rightarrow\rangle$ is taken care of explicitly by requiring chain-completeness. Now for each $u \models \langle A\rangle p \Rightarrow v \models p$ in the union add $u\langle A\rangle v$. This suffices for condition $\langle A\rangle$ as property (iii) of lean tableaux supplies $v \models p$. Finally, for each pair of states u, v appearing in the union such that $u\langle A\rangle v$ is not in the theory add $u\langle\sim A\rangle v$. Then for every $u \models [A]p$ and for every v appearing in the theory, one of $u\langle\sim A\rangle v$ or $v \models p$ is also in the theory, by property (iii) of lean tableaux. ∎

LEMMA 4.6. *For every Hintikka theory y containing r, any lean tableau for r contains a fat path the union of whose labels is a chain-complete subset of y to within renaming of states.*

Proof. The construction of such a path parallels the construction in the ordinary tableau case, except that (i) instead of a sequence of vertices being developed, each labelled with a subset of y to within renaming, we have an advancing frontier of vertices, and (ii) the induction hypothesis must be that the union of the labels of all vertices seen thus far is a subset of y to within renaming (since we no longer have $x_i \subseteq x_{i+1}$ all along a path, even if x_i is taken to be the union of the frontier's labels at the ith step). ∎

We immediately infer:

COROLLARY 4.7. *Given a lean tableau for r, r is satisfiable if and only if the union of the labels of some fat path is consistent and chain-complete.*

The Lean Procedure

This path oriented test can be translated into a vertex oriented test as for ordinary tableaux, the *lean procedure*. The procedure is to mark all the inconsistent vertices at stage 0, then at stage $i + 1$ mark all full vertices with a child marked at stage i, all partial vertices with all children marked at stage i or before, and every vertex containing some $u \models \langle a\rangle p$ not linked by a chain of $\Rightarrow$ statements at vertices (anywhere in the tableau) unmarked at stage i leading to $v \models p$ in an unmarked vertex.

LEMMA 4.8. *The lean procedure leaves the root of a tableau for r unmarked if and only if r is satisfiable.*

Proof. Suppose r is satisfiable. Then the union of some fat path is a consistent chain-complete theory. Now consider the first stage in the marking process at which a vertex on that path was marked. If it was marked on account of inconsistency then the union cannot be inconsistent. If it was marked as a full vertex with marked child then this cannot be the first stage at which a vertex on this path was marked. If it was marked as a partial vertex with only marked children then again this cannot be the first stage. (Note that in a lean tableau every partial vertex has at least one child.) If it was marked for not being linked to $v \models p$ via unmarked vertices then we use the fact that in the union of the labels of a fat path all statements of the form $u \models p$ and $u \models p \Rightarrow t$ label vertices of a single ordinary path within that fat path and going from a child of a full vertex to another full vertex. Hence all constituents of a chain starting $u \models \langle a \rangle p \Rightarrow \cdots$ that are in the theory of u can be found in the full vertex at the end of a path containing the occurrence of $u \models \langle a \rangle p$ responsible for the marking. This chain must continue in exactly one of the children of this full vertex, and so on until completion. The whole chain (to be precise, an occurrence of every link of the chain) then appears entirely within the given fat path, whence if there is a link of the chain in the fat path no occurrence of which is at an unmarked vertex, again we cannot be at the first stage when this fat path was marked. Hence at no stage can any vertex of this fat path, including the root, be marked.

Conversely, suppose the root is not marked at stage i for any finite i. Construct a fat path as follows. Initialize the set S of vertices to contain just the root. Proceeding by stages, at each stage, add to S all the children of each full vertex in S, and an unmarked child of each partial vertex in S not already having a child in S. Whenever a vertex labelled with $u \models \langle a \rangle p$ is added to S, add to S some full vertex below that vertex that contains as much of the promised chain for $u \models \langle a \rangle p$ as appears in the theory of u (this full vertex must exist), along with all vertices between the two vertices, then follow the chain into the next state and continue adding states until the end of the chain is reached, a finite process. (As we have described it, an entire chain is added as part of a single stage.) By the marking process it should be evident that every vertex in S remains unmarked. Hence at the end the theory labelling that path must be consistent (since inconsistent formulas must belong to the theory of the same state and hence appear together in the full vertex of the part of the path going through that full state) and chain-complete (by the construction). Hence r must be satisfiable. ∎

We define two vertices to be equivalent when their theories are the same to within renaming of states, and two trees to be equivalent when their roots are equivalent and to every subtree of one root corresponds an equivalent subtree of the other. Since each theory can have at most n formulas to within renaming, there can be at most 2^n mutually inequivalent vertices. We also observe that if equivalent vertices always have the same set of labels on their children (easily arranged by having a method of choosing s that depends only on the particular partial theory being extended) then two trees with equivalent roots are equivalent.

Inspection of the marking algorithm reveals that equivalent trees will be marked identically, whence equivalent vertices will all be marked simultaneously. Hence the

marking cannot proceed for more than 2^n stages without reaching some stage at which nothing new is marked, after which nothing new is ever marked.

This suggests that we *filter* the tableau (the term used by modal logicians for the process used in the proof in [6] of the finite model theorem). That is, we identify equivalent vertices to yield a directed graph, instead of a tree, having at most 2^n vertices. Such a graph can be effectively constructed by a machine.

To construct the graph using a random-access machine it suffices to construct a lean tableau, represented say using bit vectors of length n to encode each theory, and to keep an eye out for repeated states. Since our algorithm is going to use exponential storage anyway in the worst case, one may as well set aside an array of 2^n locations indexed by the bit vector representation of theories to represent which theories (modulo state names) have been encountered and where they are in storage. All this can be done in time proportional to 2^n times some small polynomial in n.

The marking procedure applied to this filtered tableau will at stage i mark exactly those vertices that are the image of vertices in the unfiltered tableau marked at stage i. Hence the root of the filtered graph will be marked if and only if the root of the unfiltered graph was marked by the procedure.

While the marking time is not linear in the number of vertices in the graph, it is clearly proportional to a small polynomial in the number of vertices, establishing our upper bound of c^n for some c. The non-linearity is due entirely to checking for chains. A crude method of checking would be to compute the transitive closure of the chain relation on the $n2^n$ facts appearing in the graph after each stage of checking, which would lead $c = 16$ if a cubic-time transitive-closure algorithm were used. This can be reduced to $c = 8$ by taking advantage of the fact that only two links can have the same first statement, whence their are only $2n2^n$ links. Further reduction may be possible. As a practical matter, one can expect the number of vertices in the graph to be quite small typically (e.g. in a program verification context), whence the important issue is whether one can reduce the marking time to say the square or better of the number of vertices, an issue we do not go into here.

The marking procedure as described requires the marking to be done in stages, with all marks made in stage $i + 1$ depending only on marks made at or before stage i. This is a little inconvenient to program, and the simpler procedure of having each mark depend on all marks made up till now, including those made at the current stage, will also work. This can be seen to be true by carrying out the proof of correctness of the unfiltered marking procedure modified so that at each stage only one equivalence class of vertices is marked. The procedure still halts in finitely many steps since there are only finitely many equivalence classes.

References

1. S. K. Basu and R. T. Yeh, Strong verification of programs, *IEEE Trans. Software Engrg.* **SE-1**, No. 3 (1975), 339–345.
2. E. W. Beth, "The Foundations of Mathematics," North–Holland, Amsterdam, 1959.
3. A. E. Chandra and L. J. Stockmeyer, Alternation, *in* "Proceedings, 17th IEEE Symposium on Foundations of Comp. Sci., 98–108, October 1976.

4. S. A. Cook, Characterization of pushdown machines in terms of time bounded computers, *J. Assoc. Comput. Mach.* **18** (1971), 4–18.
5. E. W. Dijkstra, "A Discipline of Programming," Prentice–Hall, Englewood Cliffs, N. J., 1976.
6. M. J. Fischer and R. E. Ladner, Propositional modal logic of programs, *in* "Proceedings, 9th Ann. ACM Symp. on Theory of Computing, 286–294, Boulder, Colo., May 1977.
7. M. J. Fischer and R. E. Ladner, Propositional dynamic logic of regular programs, manuscript, Dept. of Computor Science, University of Washington, 1977.
8. D. Gabbay, Axiomatizations of logics of programs, manuscript, November 1977.
9. G. Gentzen, Untersuchungen über das Logische Schliessen, *Math. Z.* **39** (1934–1935), 176–210, 405–431 (English tr.: Investigations into logical deduction, *in* "The Collected Papers of G. Gentzen," pp. 69–131, 1969).
10. K. J. J. Hintikka, Form and content in quantification theory, *Acta Philos. Fenn.* **8** (1955), 7–55.
11. C. A. R. Hoare, An axiomatic basis for computer programming, *Comm. ACM* **12** (1969), 576–580.
12. M. Jazayeri, W. F. Ogden, and W. C. Rounds, The intrinsically exponential complexity problem for attribute grammars, *Comm. ACM* **18** (1975), 697–706.
13. D. Kozen, On parallelism in Turing machines, *in* "Proceedings, 17th IEEE Symp. on Foundations of Comp. Sci.," 89–97, Oct. 1976.
14. H. Lewis, Complexity of solvable cases of the decision problem for the predicate calculus, *in* "Proceedings, 19th Annual Symposium on Foundations of Computer Science," Ann Arbor, Michigan, Oct., 1978.
15. A. R. Meyer, "Equivalence of DL, DL+ and ADL for Regular Programs with Array Assignments," Internal report, MIT, August 1977.
16. J. Misra, Prospects and limitations of automatic assertion generation for loop programs, *SIAM J. Comput.* **6** (1977), 718–729.
17. R. Parikh, A completeness result for PDL, *in* "Proceedings, Symp. on Mathematical Foundations of Computer Science," Zakopane, Poland, Sept. 1978.
18. V. R. Pratt, Semantical considerations on Floyd–Hoare logic, *in* "Proceedings, 17th Ann. IEEE Symp. on Foundations of Comp. Sci.," 109–121, 1976.
19. V. R. Pratt, A practical decision method for propositional dynamic logic, *in* "Proceedings, 10th Ann. ACM Symp. on Theory of Computing," 326–337, San Diego, Calif., May 1977.
20. V. R. Pratt, Models of program logics, *in* "Proceedings, 20th IEEE Conference on Foundations of Computer Science," San Juan, PR, Oct. 1979.
21. K. Segerberg, A completeness theorem in the modal logic of programs, Preliminary report, *Notices Amer. Math. Soc.* **24** (1977), A-552.
22. R. M. Smullyan, "First-Order Logic," Springer–Verlag, Berlin, 1968.
23. R. M. Smullyan, Trees and nest structures, *J. Symbolic Logic* **31** (1966), 303–321.
24. B. Wegbreit, The synthesis of loop predicates, *Comm. ACM* **17** (1974), 102–112.
25. K. Winklmann, "Equivalence of DL and DL$^+$ for Regular Programs," Internal report, Lab. for Comp. Sci. MIT, 1978.