MIT/LCS/TM-192

THE DEDUCIBILITY PROBLEM
IN
PROPOSITIONAL DYNAMIC LOGIC

Albert R. Meyer
Robert S. Streett
Grazina Mirkowska

February 1981
The Deducibility Problem in Propositional Dynamic Logic

by

Albert R. Meyer
and
Robert S. Streett

Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, Massachusetts USA

and

Grazina Mirkowska

Institute of Mathematics
Warsaw University
Warsaw, Poland

January 13, 1981

Abstract: The problem of whether an arbitrary formula of Propositional Dynamic Logic (PDL) is deducible from a fixed axiom scheme of PDL is \( \Pi_1^1 \)-complete. This contrasts with the decidability of the problem when the axiom scheme is replaced by any single PDL formula.

This research was supported in part by the National Science Foundation, Grant Nos. MCS 7719754, MCS 8010707, and MCS 7910261, and by a grant to the MIT Laboratory for Computer Science by the IBM Corporation.
The Deducibility Problem in Propositional Dynamic Logic

1 Introduction

Propositional Dynamic Logic (PDL) [1] is an extension of propositional logic in which "before-after" assertions about the behavior of regular program schemes can be made directly. Propositional calculus, temporal logic and the most familiar versions of propositional modal logic are all embeddable in PDL, but PDL nevertheless has a validity problem decidable in (deterministic) exponential time [4].

In this paper we consider the deducibility problem for PDL, namely the problem of when a formula \( p \) follows from a set \( \Gamma \) of formulae. The problem comes in two versions:

1. \( p \) is implied by \( \Gamma \) if and only if \( \land \Gamma \rightarrow p \) is valid.
2. \( p \) can be inferred from \( \Gamma \) if and only if \( p \) is valid in all structures for which \( \land \Gamma \) is valid.

Note that if \( p \) is implied by \( \Gamma \) then it can be inferred from \( \Gamma \), but the converse does not hold in general.

For a finite set \( \Gamma \), the question whether \( p \) is implied or inferred from \( \Gamma \) reduces to whether a formula of PDL is valid and so is decidable. However, axiomatizations of logical languages such as the propositional calculus or PDL are often given in terms of axiom schemes, namely, formulae whose variables may be replaced by arbitrary formulae. Thus, a single axiom scheme actually represents the infinite set of all formulae which are substitution instances of the scheme. Our main result is that

the problem of whether an arbitrary PDL formula \( p \) is deducible from a single fixed axiom scheme is of extremely high degree of undecidability, namely \( \Pi_1^{1}\)-complete.

This result appears unexpected for at least two reasons. First, the easily recognizable infinite set of substitution instances of a single scheme seems initially to provide little more power than a single formula. For example, the problem of whether a single PDL scheme is a sound axiom, i.e., whether all its substitution instances are valid, is equivalent to the question of whether the scheme itself regarded as a formula is valid. Hence it is decidable whether a scheme is sound.

Second, many familiar logical languages satisfy the compactness property, namely, that if \( p \) is deducible from \( \Gamma \), then in fact \( p \) is deducible from a finite subset of \( \Gamma \). It follows directly from compactness that the deducibility problem from \( \Gamma \) is recursively enumerable relative to \( \Gamma \) and the set of valid formulae of the language. Since the set \( \Gamma \) obtained from a single axiom scheme and the set of valid formulae of PDL are each decidable, compactness of PDL would imply that the deducibility problem was recursively enumerable, whereas \( \Pi_1^{1}\)-completeness in fact implies that the deducibility problem for
PDL is not even in the arithmetic hierarchy. This provides a dramatic illustration of the familiar fact that PDL is not compact.

The idea of our proof is based on an observation of Mirkowska and Pratt [2] that with a finite set of axiom schemes one can essentially define the integers up to isomorphism. This idea is extended below to define structures isomorphic to the five dimensional nonnegative integer grid with coordinatewise successor and predecessor functions and an arbitrary monadic predicate. Program schemes interpreted over these grids can compute arbitrary recursive functions of integer and monadic predicate variables. The validity of formulae asserting termination of program schemes corresponds to the validity of arithmetic formulae asserting the existence of roots of such recursive functions. Validity of such arithmetic formulae with predicate variables is well known to be a $\Pi_1^1$-complete problem.

In the next section we review the syntax and semantics of PDL and give formal definitions of the implication and inference problems from axiom schemes. In Section 3 we define the structures called grids and show that they are precisely characterized by a single axiom scheme. This easily yields the main result in Section 4 that the deducibility problems are $\Pi_1^1$-complete for PDL schemes. The argument is then sharpened to show that $\Pi_1^1$-completeness of the inference problem holds even for a restricted version of PDL, namely, deterministic PDL with atomic tests. Section 5 lists some open problems and related results.
2 Propositional Dynamic Logic

We are given a set of atomic programs $\Pi_0$ and a set of atomic propositions $\Phi_0$. Capital letters $A, B, C, \ldots$ from the beginning of the alphabet will be used to denote elements of $\Pi_0$, and capital letters $P, Q, R, \ldots$ from the middle of the alphabet will be used to denote elements of $\Phi_0$.

Definition: The set of programs, $\Pi$, and the set of formulae, $\Phi$, of propositional dynamic logic (PDL) are defined inductively as follows (note the use of letters $a, b, c, \ldots$ to denote elements of $\Pi$ and $p, q, r, \ldots$ to denote elements of $\Phi$):

\[ \Pi: \]
1. $\Pi_0 \subseteq \Pi$ and $\theta \in \Pi$
2. If $a, b \in \Pi$ then $a; b, a \cup b, a^* \in \Pi$
3. If $p \in \Phi$ then $\neg p \in \Pi$

\[ \Phi: \]
1. $\Phi_0 \subseteq \Phi$
2. If $p, q \in \Phi$ then $\neg p, p \& q \in \Phi$
3. If $a \in \Pi$ and $p \in \Phi$ then $\langle a \rangle p \in \Phi$

Definition: A PDL structure is a triple $S = \langle U, \models_S, \langle \cdot \rangle_S \rangle$ where

1. $U$ is a non-empty set, the universe of states.
2. $\models_S$ is a satisfiability relation on the atomic propositions, i.e. a predicate on $U \times \Pi_0$.
3. $\langle \cdot \rangle_S$ maps each atomic program $A$ to a binary relation $\langle A \rangle_S$ on states, i.e. $\langle A \rangle_S \subseteq U \times U$.

Definition: For any structure $S$, the relation $\models_S$ and map $\langle \cdot \rangle_S$ can be extended to arbitrary formulae and programs as follows:

1. $u \models_S \neg p$ iff $u \not\models_S p$.
2. $u \models_S p \& q$ iff $u \models_S p$ and $u \models_S q$.
3. $u \models_S \langle a \rangle p$ iff $\exists v. u \langle a \rangle_S v \& v \models_S p$.
4. $u \langle \theta \rangle_S v$ for no $u, v$.
5. $u \langle a; b \rangle_S v$ iff $\exists w. u \langle a \rangle_S w$ and $w \langle b \rangle_S v$.
6. $u \langle a \cup b \rangle_S v$ iff $u \langle a \rangle_S v$ or $u \langle b \rangle_S v$.
7. $u \langle a^* \rangle_S v$ iff $u \langle a \rangle_S^* v$, where $\langle a \rangle_S^*$ is the reflexive transitive closure of $\langle a \rangle_S$.
8. $u \langle p \rangle_S v$ iff $u = v$ and $u \models_S p$.

The standard semantics for PDL given above fix the meaning of the program $\theta$ as the empty program. If $a$ and $b$ are two programs, then $a; b$ is the program in which $a$ is followed by $b$. The program $a \cup b$ permits the nondeterministic choice of either $a$ or $b$. The program $a^*$ permits a nondeterministic choice
of some number (possibly zero) of repetitions of \( a \). If \( p \) is a formula, then \( p? \) is a test or guard program which acts as the identity program if \( p \) is true and acts as the empty program \( \emptyset \) otherwise.

**Notation:** If \( \Gamma \) is a set of formulae, then we write \( u \models_S \Gamma \) if and only if \( u \models_S p \) for every \( p \in \Gamma \).

**Definition:** If \( p \) is a formula and \( S = \langle U, \models_S \rangle \) is a structure, then \( p \) is *valid in \( S \)* if and only if \( u \models_S p \) for all \( u \in U \). If \( \Gamma \) is a set of formulae, then \( \Gamma \) is *valid in \( S \)* if and only if every formula in \( \Gamma \) is valid in \( S \). We say that \( \Gamma \) *implies \( p \)* if and only if for all structures \( S \) and states \( u \), if \( u \models_S \Gamma \) then \( u \models_S p \). We say that \( \Gamma \) *infers \( q \)* if and only if \( q \) is valid in every structure in which \( \Gamma \) is valid.

**Remark:** If \( \Gamma \) implies \( p \) then \( \Gamma \) infers \( p \), but the converse does not hold in general.

**Definition:** If \( p \) and \( q \) are formulae and \( Q \) is a primitive proposition, then \( p_Q^q \) is the formula obtained by substituting \( q \) simultaneously for every occurrence of \( Q \) in \( p \). If \( L \) is a set of formulae, then \( p_Q^L \) is the set of formulae obtainable by substituting an arbitrary formula of \( L \) for \( Q \) in \( p \), i.e. \( p_Q^L = \{ p_Q^q \mid q \in L \} \).

**Definition:** The *scheme implication problem* for a set of formulae \( L \) is to determine, for given formulae \( p \) and \( q \) and primitive proposition \( Q \), whether \( p_Q^L \) implies \( q \). The *scheme inference problem* for \( L \) is to determine whether \( p_Q^L \) infers \( q \).

It is technically convenient, given a structure, to identify or *collapse* states which are indistinguishable by formulae.

**Definition:** If \( S = \langle U, \models_S \rangle \) is a structure and \( L \) is a set of formulae, then the *\( L \)-collapse of \( S \)* is the structure \( T = \langle V, \models_T \rangle \), where the elements of \( V \) are equivalence classes of \( U \) modulo \( L \), where \( u \) is equivalent to \( v \) modulo \( L \) if and only if \( u \) and \( v \) satisfy exactly the same formulae of \( L \). For atomic propositions \( P \) and equivalence classes \([u] \in V\), we define the satisfaction relation \( \models_T \) by the condition \([u] \models_T P \) if and only if \( \exists v \in [u] \). \( v \models_S P \). For atomic programs \( A \) and equivalence classes \([u], [v] \in V\), we define the map \( \langle \cdot \rangle_T \) by the condition \([u] \langle A \rangle_T [v] \) if and only if \( \exists w \in [u]. \exists z \in [v]. w \langle A \rangle T z \).

**Lemma 2.1:** If \( T \) is the \( PDL \)-collapse of a structure \( S \), then for all \( PDL \) formulae \( p \) and states \( u \) of \( S \), \( u \models_S p \) if and only if \([u] \models_T p \).

**Proof:** Straightforward, by structural induction on formulae. \( \Box \)

It will be convenient to consider structures in which there is a designated initial state \( u \), and the entire universe is accessible from \( u \) by programs using a given set of primitives.
Definition: If \( S = \langle U, \models_S, \models'_S \rangle \), \( u \in U \), and \( \alpha \) is a set of atomic programs, then the \( \alpha \)-cut of \( S \) from \( u \) is the structure \( T = \langle V, \models_T, \models'_T \rangle \), where \( V = \{ v \in U \mid u \models (A_1 \cup \cdots \cup A_n)^* \models_S v \} \) for some \( A_1, \ldots, A_n \in \alpha \). We let \( u \models_T P \) iff \( u \models_S P \) and we let \( u \models_T \phi \) iff \( \alpha \models_T \phi \) if and only if \( \models_S \phi \).

Lemma 2.2: Suppose that \( T \) is the \( \alpha \)-cut from the state \( u \) of some structure \( S \) and that \( \alpha \) contains all the atomic programs appearing in some PDL formula \( p \). Then for all states \( v \) of \( T \), \( v \models_T p \) if and only if \( v \models_S p \).

Proof: Straightforward, by structural induction on formulae. \( \Box \)

Corollary 2.3: If \( \alpha \) contains all the atomic programs appearing in a PDL formula \( p \), then for all structures \( S \), \( p \) is valid in \( S \) if and only if \( p \) is valid in all the \( \alpha \)-cuts of \( S \).

Proof: Follows immediately from Lemma 2.2. \( \Box \)
3 Characterizing the Integer Grid by an Axiom Scheme

Notation: We define the following familiar and convenient abbreviations:

\[ [a]q \overset{df}{=} \neg \langle a \rangle \neg q \]
\[ \lambda \overset{df}{=} \theta^* \]
\[ p \lor q \overset{df}{=} \neg \neg p \land \neg \neg q \]
\[ p \rightarrow q \overset{df}{=} \neg p \lor q \]
\[ p \leftrightarrow q \overset{df}{=} (p \rightarrow q) \land (q \rightarrow p) \]
\[ \text{true} \overset{df}{=} p \rightarrow p \]
\[ \text{false} \overset{df}{=} \neg \text{true} \]
\[ a^n \overset{df}{=} \alpha \ast \cdots \ast a \text{ (for } n > 0 \text{)} \]
\[ \text{if } p \text{ then } a \text{ else } b \overset{df}{=} (p \Rightarrow a) \cup (\neg p \Rightarrow b) \]
\[ \text{while } p \text{ do } a \overset{df}{=} (p \Rightarrow a) \ast \neg p \]

For the remainder of this paper let \( \mathcal{A} = \{ A_1, A_2, A_3, A_4, A_5, B_1, B_2, B_3, B_4, B_5 \} \) be a fixed set of atomic propositions and let \( Q \) and \( R \) be fixed atomic propositions. For \( 1 \leq i \leq 5 \), let \( \text{zero}_i \) be an abbreviation for \( \bigwedge_{1 \leq j \leq 5} \text{zero}_j \).

Notation: \( N^5 \) is the set of quintuples of natural numbers. We will use variables \( x, y, \ldots \) to denote vectors \( \langle x_1, x_2, x_3, x_4, x_5 \rangle, \langle y_1, y_2, y_3, y_4, y_5 \rangle, \ldots \). The five successor functions \( \sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5 \) are defined by \( y = \sigma_i(x) \) if and only if \( y_j = x_j + 1 \) and \( y_j = x_j \) for \( j \neq i \).

A canonical grid is a structure \( S = \langle N^5, \equiv_S, \langle S \rangle \rangle \) such that \( A_i \) acts like \( \sigma_i \), \( B_i \) acts like the inverse of \( \sigma_i \) (so that \( \text{zero}_i = (B_i) \text{false} \) is true only at vectors whose \( i \)-th coordinate is zero), and \( R \) depends only on the first coordinate of vectors. A grid is any structure isomorphic to a canonical grid; we give a formal definition below.

Definition: A grid is a structure \( S = \langle U, \equiv_S, \langle S \rangle \rangle \) with a bijection \( \varphi: U \rightarrow N^5 \) such that:

1. For all \( u, v \in U \), \( u \equiv_S v \) if and only if \( \varphi(v) = \sigma_i(\varphi(u)) \).
2. For all \( u, v \in U \), \( u \equiv_S v \) if and only if \( \varphi(u) = \sigma_i(\varphi(v)) \).
3. For all \( u \in U \), if \( u \equiv_S R \) then \( v \equiv_S R \) for all \( v \) such that \( \varphi(v) = \varphi(u) \).
Definition: Let grid-scheme be an abbreviation for the conjunction of the following formulae:

zero-axiom: \( \langle B_1^*; B_2^*; B_3^*; B_4^*; B_5^* \rangle \text{ zero} \)
identity-axiom: \( \wedge \ 1 \leq i \leq 5 \langle A \times B \rangle \text{ true} \)
AB-axiom: \( \wedge \ 1 \leq i \neq j \leq 5 \langle A \times B \rangle \text{ true} \iff \langle B \times A \rangle \text{ true} \)
BB-axiom: \( \wedge \ 1 \leq i \leq 5 \langle B \times B \rangle \text{ true} \iff \langle B \times B \rangle \text{ true} \)
R-axiom: \( R \iff \wedge \ 1 \leq i \leq 5 (\langle A \rangle R \vee \langle B \rangle R) \)
determinism-scheme: \( \wedge \ 1 \leq i \leq 5 (\langle A \rangle Q \rightarrow [A]_Q) \)
identity-scheme: \( \wedge \ 1 \leq i \leq 5 (Q \rightarrow [A]_Q) \)
AA-scheme: \( \wedge \ 1 \leq i, j \leq 5 (\langle A \rangle; A \rangle Q \rightarrow [A]_Q) \)
AB-scheme: \( \wedge \ 1 \leq i \neq j \leq 5 (\langle A \rangle; B \rangle Q \rightarrow [B]_Q \)
BB-scheme: \( \wedge \ 1 \leq i, j \leq 5 (\langle B \rangle; B \rangle Q \rightarrow [B]_Q) \)

Proposition 3.1: The grids are precisely (up to isomorphism) the \( \alpha \)-cuts of PDL-collapses of structures \( S \) in which grid-scheme \( \text{PDL} \) is valid.

Proof: It is straightforward to verify that grid-scheme \( \text{PDL} \) is valid in every grid and that every grid is (isomorphic to) the \( \alpha \)-cut of the PDL-collapse of a grid.

For the converse, suppose that \( T = V \), \( \models \langle \gamma \rangle \) is the \( \alpha \)-cut from an equivalence class \( [u_{\text{start}}] \) of the PDL-collapse of a structure \( S = U \), \( \models \langle \gamma \rangle \) in which grid-scheme \( \text{PDL} \) is valid. We shall show that \( T \) is a grid. Lemmas 3.2 through 3.13 will establish the existence of a bijection \( \varphi: V \rightarrow N^5 \) which makes \( T \) a grid.

Lemma 3.2: There is an equivalence class \( [u_{\text{zero}}] \in V \) such that \( [u_{\text{zero}}] \models (\gamma) \text{ zero} \).

Proof: Since grid-scheme \( \text{PDL} \) is valid in \( S \), zero-axiom is valid in \( S \), hence \( u_{\text{start}} \models \langle B_1^*; B_2^*; B_3^*; B_4^*; B_5^* \rangle \text{ zero}. \) Hence there is a state \( u_{\text{zero}} \in U \) such that \( u_{\text{start}} \models \langle B_1^*; B_2^*; B_3^*; B_4^*; B_5^* \rangle u_{\text{zero}} \) and \( u_{\text{zero}} \models (\gamma) \text{ zero}. \) Then \( [u_{\text{zero}}] \models (\gamma) \text{ zero}, \) since \( T \) is the \( \alpha \)-cut from \( u_{\text{start}} \) of the PDL-collapse of \( S \).

Definition: An AB-program is any program of the form \( \langle a_1; \ldots; a_\ell \rangle \) where each \( a_j \) is \( \lambda \) or an \( A_j \) or a \( B_j \). An A-program is simply an AB-program without any \( B_j \)'s. A canonical A-program is an A-program of the form \( A_1^{x_1}; A_2^{x_2}; A_3^{x_3}; A_4^{x_4}; A_5^{x_5} \) for some \( x_1, x_2, x_3, x_4, x_5 \geq 0 \). We abbreviate \( A_1^{x_1}; A_2^{x_2}; A_3^{x_3}; A_4^{x_4}; A_5^{x_5} \) by \( \text{prog}(x) \).

Lemma 3.3: If \( [u] \in V \) and \( a \) is an A-program, then there is at least one \( [v] \) such that \( [u] \langle a \rangle [v] \).

Proof: We first prove this lemma for the case where \( a \) is \( A_i \) for some \( i \). By identity-axiom, \( u \models \langle A \rangle [v] \) \( \langle A \times B \rangle \text{ true}, \) so that there is at least one \( v \in U \) such that \( u \langle A \rangle [v] \). Then \( [u] \langle A \rangle [v] \), since \( T \) is
an $\alpha$-cut of the $PDL$-collapse of $S$. The lemma can now be proved for arbitrary $A$-programs by
an easy induction on the length of programs.

Lemma 3.4: If $[v] \in V$ and $a$ is an $A$-program, then there is at most one $[v]$ such that $[u]a[v]$. 

Proof: We first prove this lemma for the case where $a$ is $A_i$ for some $i$. Suppose that $[u]A_i[v]$ and $[u]A_i[w]$. Then $u \vdash A_i v$ and $u \vdash A_i w$. Let $q$ be any formula such that $v \vdash q$, so that $u \vdash A_q$. By determinism-scheme, $u \vdash A_q \rightarrow [A_i]q$. Since $u \vdash A_q$, $u \vdash [A_i]q$, so $w \vdash q$. Hence $v$ and $w$ agree, in $S$, on all formulae, so $[v] = [w]$. Therefore there is at most one $[v]$ such that $[u]A_i[v]$. The lemma can now be proved for arbitrary $A$-programs by an easy induction on the length of programs.

Lemma 3.5: If $a$ is an $A$-program and $b$ is any program and $[u]a[v]$ and $[u]a;b[v]$, then $[v]b[v]$.

Proof: If $[u]a;b[v]$ then there is a $z$ such that $[u]a;z$ and $[z]b[v]$. By Lemma 3.4, it follows from $[u]a;z$ and $[u]a;z$ that $[v] = [z]$. So $[v]b[v]$.

Definition: Given two programs $a$ and $b$, we say that $a$ and $b$ are $T$-equivalent if and only if $\langle a \rangle_T = \langle b \rangle_T$, i.e. for all states $u$ and $v$, $u \vdash a v$ if and only if $u \vdash b v$.

Lemma 3.6: The program $A_i;B_j$ is $T$-equivalent to the identity program $\lambda$.

Proof: By identity-axiom, $u \vdash A_i \times B_j \rightarrow q$. Hence there is a state $w \in U$ such that $u \vdash A_i w$ and $w \vdash B_j q$. Hence there is a state $v$ such that $w \vdash B_j v$ and $u \vdash A_i B_j v$. Now let $q$ be any formula such that $u \vdash A_i q$. By identity-scheme, $u \vdash A_i q \rightarrow [A_i;B_j]q$. Since $u \vdash q$, $u \vdash [A_i;B_j]q$, so $v \vdash q$. Hence $u$ and $v$ agree, in $S$, on all formulae, so $[u] = [v]$. Therefore, $A_i;B_j$ is the identity program in the $PDL$-collapse of $S$, hence also in $T$.

Lemma 3.7: If $a$ and $b$ are $A$-programs and $a$ is a permutation of $b$, then $a$ and $b$ are $T$-equivalent.

Proof: By an induction on the length of $a$ and $b$, using $AA$-scheme.

Lemma 3.8: If $a$ is an $AB$-program not containing $A_i$, then $a;B_j$ and $B_j;a$ are $T$-equivalent.

Proof: By an induction on the length of $a$, using $AB$-axiom, $BB$-axiom, $AB$-scheme, and $BB$-scheme.

Lemma 3.9: If $a$ is an $AB$ program not containing $A_i$ or $B_j$, and if $[u]a[v]$, then $[u] \vdash T R$ if and only if $[v] \vdash T R$.

Proof: By an induction on the length of $a$, using $R$-axiom.
Definition: An $AB$ program $a$ is nonnegative if and only if every prefix of $a$ contains at least as many $A_i$’s as $B_i$’s, for $1 \leq i \leq 5$.

Lemma 3.10: Every nonnegative $AB$-program is $T$-equivalent to an $A$-program. 

Proof: If $a$ is a nonnegative $AB$-program, then $a$ is $T$-equivalent to $b; A_i; c; B_i; d$ where $b$ and $c$ are (possibly trivial) $A$-programs, $c$ contains no $A_i$’s, and $d$ is an $AB$-program. By Lemma 3.8, $a$ is $T$-equivalent to $b; A_i; B_i; e; d$, and by Lemma 3.6, $a$ is $T$-equivalent to $b; c; d$, which is nonnegative and contains one less $B_i$ than $a$. The lemma follows by an easy induction on the number of $B_i$’s in $a$.

Lemma 3.11: If the $AB$-program $a$ is not nonnegative, then there is no $[u]$ such that $[u] \not\rightarrow [u]$.

Proof: If $a$ is not nonnegative, then $a$ is equivalent to $b; B_i; c$ where $b$ and $c$ are $AB$-programs such that $b$ contains no $A_i$’s. By Lemma 3.8, $a$ is $T$-equivalent to $b; B_i; c$. Since $u_{\text{zero}} \not\rightarrow \text{false}$, there can be no $u$ such that $u_{\text{zero}} \not\rightarrow [u]$, hence no $u$ such that $u_{\text{zero}} \not\rightarrow [u]$, since $a$ is $T$-equivalent to $b; B_i; c$. Hence there is no $[u]$ such that $[u_{\text{zero}}] \not\rightarrow [u]$. 

For the rest of the proof of Proposition 3.1, we will use $u, v, w, \ldots$ to denote elements of $V$, since we no longer need to make use of the fact that elements of $V$ are equivalence classes of elements of $U$. Let $u_{\text{zero}}$ be that element of $V$ such that $u_{\text{zero}} \not\rightarrow \text{false}$.

Lemma 3.12: For all $u \in V$, there is at most one $x$ such that $u_{\text{zero}} \not\rightarrow [u]$.

Proof: Suppose $x \neq y$, but $u_{\text{zero}} \not\rightarrow [x]$, $u_{\text{zero}} \not\rightarrow [y]$. Without loss of generality we can suppose that $\not\rightarrow [x]$, $\not\rightarrow [y]$, $\not\rightarrow [y]$. By Lemma 3.11, there is no $v$ such that $u_{\text{zero}} \not\rightarrow [v]$, hence no $v$ such that $u_{\text{zero}} \not\rightarrow [v]$. Therefore $u \not\rightarrow [u]$. By Lemma 3.3, there is a $w$ such that $u_{\text{zero}} \not\rightarrow [w]$ and hence such that $u_{\text{zero}} \not\rightarrow [w]$. By Lemma 3.5, $u_{\text{zero}} \not\rightarrow [w]$. Hence $u \not\rightarrow [u]$, a contradiction. So $x \neq y$ is not possible.

We now prove that the relation between a state $u \in V$ and a vector $x$ defined by $u_{\text{zero}} \not\rightarrow [u]$ is the desired bijection.

Lemma 3.13: There is a bijection $\varphi: V \rightarrow N^S$ such that $\varphi(u) = x$ if and only if $u_{\text{zero}} \not\rightarrow [u]$.

Proof: Let $u \in V$. Since $T$ is an $\alpha$-cut, there is an $AB$-program $a$ such that $u_{\text{zero}} \not\rightarrow [u]$. By Lemma 3.11, $a$ must be nonnegative. By Lemma 3.10, $a$ is $T$-equivalent to some $A$-program $b$, which, by Lemma 3.7, is $T$-equivalent to $\varphi(x)$ for some $x$. By Lemma 3.12, $x$ is unique, so we may define $\varphi(u) = x$. To show that $\varphi$ is an injection, suppose that $\varphi(u) = \varphi(v) = x$. By the
definition of \( \varphi \). \( u_{\Var{zero}} \langle \varphi, \text{prog}(x) \rangle \gamma^u \) and \( u_{\Var{zero}} \langle \varphi, \text{prog}(x) \rangle \gamma^v \). By Lemma 3.4, \( u = v \). To show that \( \varphi \) is a surjection, let \( x \in \mathbb{N}^5 \). By Lemma 3.3, there is a \( u \) such that \( u_{\Var{zero}} \langle \varphi, \text{prog}(x) \rangle \gamma^u \).

so \( \varphi(u) = x \). 

Finally, we will show that \( \varphi \) makes \( T \) a grid, by proving that the three defining properties of grids hold of \( T \) and \( \varphi \).

1. Suppose \( u \langle B \rangle \gamma^u \). Then \( u_{\Var{zero}} \langle \text{prog}(\varphi(u)) \rangle \gamma^u \) and \( u_{\Var{zero}} \langle \text{prog}(\varphi(u)) \rangle \gamma^v \). By Lemma 3.7, \( u_{\Var{zero}} \langle \text{prog}(\varphi(u)) \rangle \gamma^v \). By Lemma 3.13, \( \varphi(v) = \sigma(\varphi(u)) \).

Conversely, suppose \( \varphi(v) = \sigma(\varphi(u)) \). Then \( u_{\Var{zero}} \langle \text{prog}(\varphi(u)) \rangle \gamma^u \) and \( u_{\Var{zero}} \langle \text{prog}(\varphi(u)) \rangle \gamma^v \). By Lemma 3.7, \( u_{\Var{zero}} \langle \text{prog}(\varphi(u)) \rangle \gamma^v \). By Lemma 3.5, \( u \langle B \rangle \gamma^v \).

2. Without loss of generality let \( i = 1 \). Suppose \( u \langle B_i \rangle \gamma^v \) where \( \varphi(u) = x \) and \( \varphi(v) = y \). Then

\[ u_{\Var{zero}} \langle \text{prog}(x) \rangle \gamma^u \] by Lemma 3.8, \( u_{\Var{zero}} \langle A_1 \rangle \gamma^1 \).

\( \varphi(v) = \sigma(\varphi(u)) \). By Lemma 3.6, \( u_{\Var{zero}} \langle A_1 \rangle \gamma^1 \).

Therefore \( x = \varphi(u) = \sigma(\varphi(v)) \).

Conversely, suppose \( \varphi(u) = \sigma(\varphi(v)) \). Then \( u_{\Var{zero}} \langle \text{prog}(\sigma(\varphi(u))) \rangle \gamma^u \) and \( u_{\Var{zero}} \langle \text{prog}(\sigma(\varphi(v))) \rangle \gamma^v \).

3. Suppose \( u \iff R \) and \( \varphi(u) = \varphi(v) \). Then \( u_{\Var{zero}} \langle \text{prog}(x) \rangle \gamma^u \) and

\[ u_{\Var{zero}} \langle A_1 \rangle \gamma^1 \]

\[ u_{\Var{zero}} \langle \text{prog}(x) \rangle \gamma^v \] by Lemma 3.9, \( v \iff R \). This completes the proof of Proposition 3.1.

Corollary 3.14: If \( \alpha \) contains all primitive programs appearing in a formula \( p \), then \( p \) is valid in all grids if and only if grid-scheme\( PDL \) infers \( p \).

Proof: By definition, grid-scheme\( PDL \) infers \( p \) if and only if \( p \) is valid in all structures in which grid-scheme\( PDL \) is valid. By Lemma 2.1, the latter is true if and only if \( p \) is valid in all PDLCollapses of structures in which grid-scheme\( PDL \) is valid. By Corollary 2.3, this is so if and only if \( p \) is valid in all \( \alpha \)-cuts of PDLCollapses of structures in which grid-scheme\( PDL \) is valid. By Proposition 3.1, this is so if and only if \( p \) is valid in all grids.

Notation: Let \( \alpha^* \) abbreviate \( (A_1 \cup A_2 \cup A_3 \cup A_4 \cup A_5 \cup B_1 \cup B_2 \cup B_3 \cup B_4 \cup B_5)^\alpha \).

Corollary 3.15: If \( p \) is a formula all of whose atomic programs are in \( \alpha \), then \( p \) is valid in all grids if and only if \( ([\alpha^*]) \text{grid-scheme}_{PDL} \) implies \( p \).

Proof: Left to the reader.
4 \Pi_1^1\text{-completeness of the Deducibility Problem for PDL}

**Lemma 4.1:** Let \( f: 2^N \times N^3 \rightarrow N \) be a partial recursive function of one set variable and three integer variables. There is a PDL program \( a_f \) such that, in every grid \( S \), \( u \vdash a_f S \vdash v \) if and only if 
\[
\phi(v)_1 = \mathcal{A}(X_S, \phi(u)_1, \phi(u)_2, \phi(u)_3), \text{ where } X_S = \{ \phi(w)_1 \mid w \vdash_S R \}.
\]

**Proof:** An oracle counter machine is a computing device possessing registers capable of holding arbitrary nonnegative integers and a processor capable of incrementing and decrementing (when the result is nonnegative) the contents of a specified register, testing whether the contents of a specified register is zero or not, and testing the contents of the first register for membership in a fixed but arbitrary set called the "oracle". (The formal definition is analogous to that of oracle Turing machines [5, 6] and is omitted.) A 5-counter machine is capable of computing any partial recursive function of one set variable and three integer variables, where we assume that the three inputs are initially stored in the first three registers (the extra two registers are for temporary results and may initially contain arbitrary values) and that the single integer output is stored, at the end, in the first register. A program \( a_f \) to compute such a function \( f \) can be written as a regular program using the primitives (where \( 1 \leq i \leq 5 \)): \( A_i \) to increment register \( i \), \( B_i \) to decrement register \( i \), \( \text{zero}_i \) and \( \neg \text{zero}_i \) to test register \( i \) for zero, and \( R \) and \( \neg R \) to test whether the contents of register \( i \) is in the oracle set \( X_S \). In a grid \( S \) the standard PDL semantics interprets \( a_f \) as a program which computes \( f \), i.e. that \( u \vdash a_f S \vdash v \) if and only if \( \phi(v)_1 = \mathcal{A}(X_S, \phi(u)_1, \phi(u)_2, \phi(u)_3) \).

For the remainder of this paper let \( Y \) be a fixed \( \Pi_1^1 \)-complete set of natural numbers, so that there is a fixed recursive function \( \mathcal{A}(X, x, y, z) \) of one set variable and three integer variables such that \( Y = \{ x \mid \forall X \subseteq N. \exists y. \forall z. \mathcal{A}(X, x, y, z) = 0 \} \).

**Corollary 4.2:** There is a PDL formula \( \rho_Y \) such that for all natural numbers \( m \), the formula 
\[
\text{zero}_1 \rightarrow \mathcal{A}(\rho_Y) p_Y \text{ is valid in all grids if and only if } m \in Y.
\]

**Proof:** By the preceding lemma, for all grids \( S \) and states \( u, u \vdash_S \langle a \rangle \text{zero}_1 \) if and only if 
\[
\mathcal{A}(X_S, \phi(u)_1, \phi(u)_2, \phi(u)_3) = 0.
\]
The program \( B_1^*; A_1^* \) is capable of arbitrarily altering the contents of the \( i \text{-th} \) register. Hence \( u \vdash_S \langle B_1^*; A_1^* \rangle \mathcal{A}(\text{zero}_1 \) if and only if 
\[
\forall x \in \mathcal{N} . \mathcal{A}(X_S, \phi(u)_1, \phi(u)_2, x) = 0.
\]
Similarly, \( u \vdash_S \langle B_2^*; A_2^* \rangle \mathcal{A}(\text{zero}_1 \) if and only if 
\[
\forall y \in \mathcal{N} . \mathcal{A}(X_S, \phi(u)_1, y, \phi(u)_3) = 0.
\]
Let \( p_Y \) be \( \langle B_2^*; A_2^* \rangle \mathcal{A}(\text{zero}_1 \) if and only if 
\[
\forall x \in \mathcal{N} . \forall y \in \mathcal{N} . \forall z \in \mathcal{N} . \mathcal{A}(X, x, y, z) = 0.
\]
If \( u \vdash_S \text{zero}_1 \), then \( u \vdash_S \langle A_1^* \rangle \mathcal{A}(\text{zero}_1 \) if and only if \( \forall X \subseteq \mathcal{N} . \exists y \in \mathcal{N} . \exists z \in \mathcal{N} . \mathcal{A}(X, m, y, z) = 0 \). As \( S \) ranges over all grids, \( X_S \) ranges over all sets of nonnegative integers. Therefore, 
\[
\text{zero}_1 \rightarrow \mathcal{A}(\rho_Y) p_Y \text{ is valid in all grids if and only if } \forall X \subseteq \mathcal{N} . \exists y \in \mathcal{N} . \forall z \in \mathcal{N} . \mathcal{A}(X, m, y, z) = 0, \text{ i.e. if and only if } m \in Y.
\]

**Proposition 4.3:** The scheme inference (respectively, implication) problem for PDL is \( \Pi_1^1 \)-complete.
Proof: By Corollaries 3.14 (3.15) and 4.2, there is a PD1 formula \( p_Y \) such that \( m \in Y \) if and only if grid-scheme \( Q^{PD1} \) infers (implies) \( \text{zero}_1 \rightarrow \langle A^m, m > p_Y \). This proves that \( \Pi_1^1 \) is many-one reducible to the scheme inference (implication) problem for PD1. It is not hard to show that either problem is in \( \Pi_1^1 \); we omit the proof.

We now define some sublanguages of PD1 and show that the scheme implication and inference problems are \( \Pi_1^1 \)-complete for some of these sublanguages.

**Definition:** The formulae of *test-free propositional dynamic logic* are those in which no tests appear; the formulae of *atomic test propositional dynamic logic* are those in which the construction \( p? \) appears only when \( p \) is an atomic proposition.

**Theorem 4.4:** If \( L \) is a subset of PD1 which contains atomic-test-PD1, then the scheme implication problem for \( L \) is \( \Pi_1^1 \)-complete.

**Proof:** The test-free tests of \( p_Y \) are of the form \( \text{zero}_0? \), \( \neg \text{zero}_0? \), and \( \neg R? \). Choose new atomic propositions \( Z, N, \) and \( M \). Let \( q_Y \) be the result of substituting \( Z \) for \( \text{zero}_0? \), \( N \) for \( \neg \text{zero}_0? \), and \( M \) for \( \neg R? \) in \( p_Y \). Let grid-scheme be grid-scheme \& \( [\alpha]^x)(Z \leftrightarrow \text{zero}_0 \& \ldots \& M \leftrightarrow \neg R) \). We leave it to the reader to show that the problem of deciding, for a given \( m \), whether or not grid-scheme \( Q^L \) implies \( \text{zero}_1 \rightarrow \langle A^m, m > q_Y \) is \( \Pi_1^1 \)-complete.

**Definition:** The set of programs, \( \Pi_d \), and the set of formulae, \( \Phi_d \), of *deterministic propositional dynamic logic* (DPDL) are defined inductively as follows:

\[ \Pi_d : \]
1. \( \Pi_0 \subseteq \Pi_d \) and \( \theta, \lambda \in \Pi_d \)
2. If \( a, b \in \Pi_d \) and \( p \in \Phi_d \), then \((a, b), (if p then a else b), (while p do a) \in \Pi_d \)

\[ \Phi_d : \]
1. \( \Phi_0 \subseteq \Phi_d \)
2. If \( p, q \in \Phi_d \), then \( \neg p, p \& q \in \Phi_d \)
3. If \( a \in \Pi_d \) and \( p \in \Phi_d \), then \( \langle a \& p \rangle \in \Phi_d \)

**Proposition 4.5:** If \( L \) is a subset of PD1 which contains DPDL, then the scheme implication problem for \( L \) is \( \Pi_1^1 \)-complete.

**Proof:** First, note that \( g \) of Lemma 4.1 can easily be written as a program in \( \Pi_d \). Second, note that for all programs \( a \) and formulae \( p \), \( \langle a > p \) is equivalent to \( \langle while \neg p \ do \ a > \text{true} \). Hence, there is a formula \( r_Y \) in \( \Pi_d \) which is equivalent to \( p_Y = d_f \langle B_2 \ast; A_2 \ast; B_3 \ast; A_3 \ast \rangle \text{zero}_1 \). Finally, note that every conjunct of grid-scheme is in \( \Pi_d \) except for zero-axiom \( = d_f \langle B_1 \ast; B_2 \ast; B_3 \ast; B_4 \ast; B_5 \ast > \text{zero} \). There is a formula in \( \Pi_d \) which is equivalent to zero-axiom in all structures; let det-scheme be grid-scheme with zero-axiom replaced by this formula. We leave it to the reader to show that the problem of deciding, for a given \( m \), whether or not det-scheme \( Q^L \) infers \( \text{zero}_1 \rightarrow \langle A^m, m > r_Y \) is
\[ \Pi_1^1 \text{-complete.} \]

**Definition.** The formulae of atomic-test-DPDL are those in which the constructions if \( p \) then \( a \) else \( b \) and while \( p \) do \( a \) appear only when \( p \) is an atomic proposition.

**Theorem 4.6:** If \( I \) is a subset of \( PDL \), which contains atomic-test-DPDL, then the scheme inference problem for \( I \) is \( \Pi_1^1 \)-complete.

**Proof:** Let \( det\text{-scheme} \) and \( q_I \), be as in the proof of Proposition 4.5. Replace their non-atomic tests by new atomic tests as in the proof of Theorem 4.4. (This replacement must be performed recursively on nested tests.)

5 Conclusions and Open Problems

Because of its many decidable properties, \( PDL \) appears to be a reasonably tractable extension of propositional logic. However, we have revealed a dramatic contrast between \( PDL \) and ordinary propositional logic in the case of the scheme deducibility problem, which is \( \Pi_1^1 \)-complete for \( PDL \), but decidable for propositional logic.

An important hint at the power of \( PDL \) axiom schemes was provided by the observation of Mirkowska and Pratt [2], who showed that the nonnegative integers could be characterized (as cuts of \( PDL \)-collapsed structures) by a finite set of axiom schemes. Hence this set of axiom schemes does not satisfy the finite model property, namely these schemes have a model but no finite model. Since all the previously known decidability results for \( PDL \), ultimately rest on the finite model property of \( PDL \) formulae, the Mirkowska-Pratt observation helps clarify the contrast between schemes and finite sets of axioms.

However, violation of the finite model property should not be taken as *prima facie* evidence of undecidability. For example, Mirkowska has observed that the nonnegative integers can also be uniquely characterized by a single formula of \( PDL \), extended with a looping predicate and the converse operation on programs [3]. Nevertheless, by extending the results of [7], Streett can show that this extension of \( PDL \) is still decidable (in fact, elementary recursive). This result will appear in a later paper.

The degrees of undecidability (or decidability) of several restricted deducibility problems remain open questions.
Open Problem: Are the scheme implication and inference problems for test-free-PDL $\Pi_1^1$-complete?

Open Problem: Is the scheme implication problem for DPDL or atomic-test-DPDL $\Pi_1^1$-complete?

Open Problem: How hard are the scheme deducibility problems for propositional temporal and modal logics?

Acknowledgement: We are grateful to A. Salwicki for pointing out the possibility of characterizing the integers by PDL axiom schemes, and for several useful discussions about these results.
References


