The Deducibility Problem in Propositional Dynamic Logic

Albert R. Meyer and Robert S. Streett
Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, Massachusetts USA

Grazyna Mirkowska
Institute of Mathematics
Warsaw University
Warsaw, Poland

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

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 and versions of propositional modal logic, propositional temporal logic, and Propositional Algorithmic Logic [2] 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 \( \forall \Gamma \rightarrow p \) is valid.

2. \( p \) can be inferred from \( \Gamma \) if and only if \( p \) is valid in all structures for which \( \forall \Gamma \) 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 \( \Pi_1 \)-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 Salwicki 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 restricted versions of \( PDL \), namely, test-free \( PDL \) and deterministic \( PDL \) with atomic tests. Similarly, the implication problem is \( \Pi_1^1 \)-complete for test-free \( PDL \). 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 \( \Phi \)):

\[
\Pi:
\begin{align*}
(1) & \quad \Pi_0 \subseteq \Pi \text{ and } \theta \in \Pi \\
(2) & \quad \text{If } a, b \in \Pi \text{ then } a;b, a\cup b, a^* \in \Pi \\
(3) & \quad \text{If } p \in \Phi \text{ then } p! \in \Pi
\end{align*}
\]

\[
\Phi:
\begin{align*}
(1) & \quad \Phi_0 \subseteq \Phi \\
(2) & \quad \text{If } p, q \in \Phi \text{ then } \neg p, p\land q \in \Phi \\
(3) & \quad \text{If } a \in \Pi \text{ and } p \in \Phi \text{ then } \langle a \rangle p \in \Phi
\end{align*}
\]

**Definition:** A *PDL structure* is a triple \( S = \langle U, \models, \vdash \rangle \) 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:

\[
(1) \quad u \models_S \neg p \iff \neg u \models_S p.
\]

\[
(2) \quad u \models_S p \land q \iff u \models_S p \land u \models_S q.
\]

\[
(3) \quad u \models_S \langle A \rangle_S \iff \exists v. \ u \langle A \rangle_S v \land v \models_S p.
\]

\[
(4) \quad u \langle \theta \rangle_S v \text{ for no } u, v.
\]

\[
(5) \quad u \langle a; b \rangle_S v \iff \exists w. \ u \langle a \rangle_S w \text{ and } w \langle b \rangle_S v.
\]

\[
(6) \quad u \langle a^* \rangle_S v \iff \exists w. \ u \langle a \rangle_S w \text{ and } w \langle a^* \rangle_S v.
\]

\[
(7) \quad u \langle a* \rangle_S v \iff u \langle a* \rangle_S v, \text{ where } \langle a* \rangle_S \text{ is the reflexive transitive closure of } \langle a \rangle_S.
\]

\[
(8) \quad u \langle p? \rangle_S v \iff u = v \land u \models_S p.
\]

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 \( a;b \) 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 \( \theta \) 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 \( S = \langle U, \models_S, \langle \cdot \rangle_S \rangle \) 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 \( p^Q \) 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 \( S = \langle U, \models_S, \langle \cdot \rangle_S \rangle \) is a structure and \( L \) is a set of formulae, then the \( L \)-collapse of \( S \) is the structure \( T = \langle V, \models_T, \langle \cdot \rangle_T \rangle \), 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] ∈ V, we define the satisfaction relation $\models_T$ by the condition $[u] \models_T P$ iff $\exists v \in [u]. v \models_S P$. For atomic programs A and equivalence classes [u], [v] ∈ V, we define the map $\langle A \rangle_T^v$ by the condition $[u] \langle A \rangle_T^v$ iff $\exists w \in [u]. \exists z \in [v]. w \models_S A$.

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

Proof: Straightforward, by structural induction on formulae.  

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 $S = \langle U, \models_S, \langle \cdot \rangle_S, u_0 \rangle$, u_0 ∈ U, and α is a set of atomic programs, then the α-cut of S from u_0 is the structure $T = \langle V, \models_T, \langle \cdot \rangle_T, \rangle_U$, where $V = \{ u \in U \mid u_0^{(A_1 \cup \cdots \cup A_p)^r} \} \cup u$ for some $A_1, \ldots, A_p \in \alpha$. We let $u \models_T P$ iff $u \models_S P$ and we let $u \langle A \rangle_T^v$ iff $A \in \alpha$ and $u \langle A \rangle_S^v$.

Lemma 2.2: Suppose that T is the α-cut from the state u of some structure S and that α 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.  

Corollary 2.3: If α 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 α-cuts of S.

Proof: Follows immediately from Lemma 2.2.  

3 Characterizing the Integer Grid by an Axiom Scheme

Notation: We define the following familiar and convenient abbreviations:

\[ [a]q =_{df} \neg a \neg q \]
\[ \lambda =_{df} \emptyset \]
\[ p \lor q =_{df} (\neg p) \land (\neg q) \]
\[ p \rightarrow q =_{df} (\neg p) \lor q \]
\[ p \leftrightarrow q =_{df} (p \rightarrow q) \land (q \rightarrow p) \]
\[ \text{true} =_{df} p \rightarrow p \]
\[ \text{false} =_{df} \neg \text{true} \]
\[ a^0 =_{df} \lambda \]
\[ a^n =_{df} a; \cdots ; a \quad (n \text{ 's}, \text{ for } n > 0) \]
\[ \text{if } p \text{ then } a \text{ else } b =_{df} (p; a) \cup (\neg p; b) \]
\[ \text{while } p \text{ do } a =_{df} (p; a)^* \neg p \]

For the remainder of this paper let $\alpha = \{ A_1, A_2, A_3, A_4, A_5, B_1, B_2, B_3, B_4, B_5 \}$ be a fixed set of atomic programs and let $Q$ and $R$ be fixed atomic propositions. For $1 \leq i \leq 5$, let $\text{zero}_i$ be an abbreviation for $[B_i]\text{false}$ and let $\text{zero}$ be an abbreviation for $\bigwedge_{1 \leq i \leq 5} \text{zero}_i$. 

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

A canonical grid is a structure $S = <N^5, \llhd_S, \llhd>$ such that $\alpha_i$ acts like $\sigma_i$, $B_i$ acts like the inverse of $\sigma_i$ (so that $\text{zero}_i = df \{B_i\text{false} \text{ is true only at vectors whose } i\text{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 $S = <U, \llhd_S, \llhd>$ with a bijection $\phi: U \rightarrow N^5$ such that:

1. For all $u, v \in U$, $u \llhd_S v$ if and only if $\phi(v) = \sigma(\phi(u))$.
2. For all $u, v \in U$, $u \llhd_S v$ if and only if $\phi(u) = \sigma(\phi(v))$.
3. For all $u \in U$, if $u \llhd_S R$ then $v \llhd_S R$ for all $v$ such that $\phi(v)_1 = \phi(u)_1$.

**Definition:** Let grid-scheme be an abbreviation for the conjunction of the following formulae:

- **zero-axiom:** $<B_1^*; B_2^*; B_3^*; B_4^*; B_5^*> \text{zero}$
- **identity-axiom:** $\forall 1 \leq i \leq 5 \ (\! \! \forall \! \! B_i \! \! \exists \! \! A_i \! \! A_i \! \! B_i \! \! B_i \! \! true)$
- **AB-axiom:** $\forall 1 \leq i < j \leq 5 \ (\! \! \forall \! \! A_i \! \! B_j \! \! true \leftrightarrow \! \! A_j \! \! B_i \! \! true)$
- **BB-axiom:** $\forall 1 \leq i < j \leq 5 \ (\! \! \forall \! \! B_i \! \! B_j \! \! true \leftrightarrow \! \! B_j \! \! B_i \! \! true)$
- **R-axiom:** $R \rightarrow \bigwedge 1 \leq i \leq 5 (A_i R \land \bigwedge 1 \leq i \leq 5 (A_i R))$
- **determinism-scheme:** $\bigwedge 1 \leq i \leq 5 (A_i Q \rightarrow [A_i Q]_R)$
- **identity-scheme:** $\bigwedge 1 \leq i \leq 5 (Q \rightarrow [A_i B_i] Q)$
- **AB-scheme:** $\bigwedge 1 \leq i < j \leq 5 (A_i B_j Q \rightarrow [B_i A_j] Q)$
- **BB-scheme:** $\bigwedge 1 \leq i < j \leq 5 (B_i B_j Q \rightarrow [B_i B_j] Q)$

The proof of the following proposition is straightforward, but quite lengthy; the reader may wish to proceed directly to section 4.

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

**Proof:** It is straightforward to verify that grid-scheme$_{PDL}$ 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 $T = <V, \llhd_T, \llhd>$ is the $\alpha$-cut from an equivalence class $[u_{\text{start}}]$ of the PDL-collapse of a structure $S = <U, \llhd_S, \llhd>$ in which grid-scheme$_{PDL}$ is valid. We shall show that $T$ is a grid. Lemmas 3.2 through 3.13 will establish the existence of a bijection $\phi: V \rightarrow N^5$ which makes $T$ a grid.

**Lemma 3.2:** There is an equivalence class $[u_{\text{zero}}] \in V$ such that $[u_{\text{zero}}] \models T_{\text{zero}}$.

**Proof:** Since grid-scheme$_{PDL}$ is valid in $S$, zero-axiom is valid in $S$, hence $u_{\text{start}} \models_S <B_1^*; B_2^*; B_3^*; B_4^*; B_5^*> \text{zero}$. Hence there is a state $u_{\text{zero}} \in U$ such that $u_{\text{start}} <B_1^*; B_2^*; B_3^*; B_4^*; B_5^*> u_{\text{zero}}$ and $u_{\text{zero}} \models_S \text{zero}$. Then $[u_{\text{zero}}] \models_T$
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 $a_1; \ldots; a_n$ where each $a_j$ is $\lambda$ or an $A_i$ or a $B_i$. An $A$-program is simply an AB-program without any $B_i$'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 prog($x$).

**Lemma 3.3:** If $[u] \in V$ and $a$ is an A-program, then there is at least one $[v]$ such that $[u] < a \prec [v]$.

**Proof:** We first prove this lemma for the case where $a$ is $A_i$ for some $i$. By identity-axiom, $u \models_S <A_i \prec B_j > \text{true}$, so that there is at least one $v \in U$ such that $u < A_i \prec v$. Then $[u] < A_i \prec [v]$, 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 $[u] \in V$ and $a$ is an A-program, then there is at most one $[v]$ such that $[u] < a \prec [v]$.

**Proof:** We first prove this lemma for the case where $a$ is $A_i$ for some $i$. Suppose that $[u] < A_i \prec [v]$ and $[u] < A_i \prec [w]$. Then $u < A_i \prec v$ and $u < A_i \prec w$. Let $q$ be any formula such that $v \models_S q$, so that $u \models_S <A_i > q$. By determinism-scheme, $u \models_S <A_i > q \rightarrow [A_i]q$. Since $u \models_S <A_i > q$, $u \models_S [A_i]q$, so $w \models_S q$. Hence $v$ and $w$ agree, in $S$, on all formulae, so $[v] = [w]$. Therefore there is at most one $[v]$ such that $[u] < A_i \prec [v]$. 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 $[u] < a \prec [v]$ and $[u] < a ; b \prec [w]$, then $[v] < b \prec [w]$.

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

**Definition:** Given two programs $a$ and $b$, we say that $a$ and $b$ are $T$-equivalent if and only if $< a > T = < b > T$, i.e., for all states $u$ and $v$, $u < a \prec v$ iff $u < b \prec v$.

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

**Proof:** By identity-axiom, $u \models_S <A_i > B_j \text{true}$. Hence there is a state $w \in U$ such that $u < A_i \prec w$ and $w \models_S <B_j > \text{true}$. Hence there is a $v$ such that $w < B_j \prec v$ and $v \models_S A_i \text{true}$. Now let $v$ be any state in $U$ such that $u < A_i ; B_j \prec v$. Let $q$ be any formula such that $u \models_S q$. By identity-scheme, $u \models_S q \rightarrow [A_i]q$. Since $u \models_S q$, $u \models_S [A_i]q$, so $v \models_S q$. 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 $A_f$, then $a ; B_i$ and $B_i ; a$ 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 $A_1$ or $B_1$, and if $[u] < a \prec [v]$, then $[u] \models_T R$ if and only if $[v] \models_T R$.

**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 \( b;A_i;B_i;c;d \), 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 \([u] \leq_{T} \langle \alpha \rangle \).

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 \( B_i;B_i;c \). Since \( u_{zero} \leq_{T} \langle \alpha \rangle \), there can be no \( u \) such that \( u_{zero} \leq_{T} \langle B_i \rangle \), hence no \( u \) such that \( u_{zero} \leq_{T} \langle \alpha \rangle \), since \( a \) is T-equivalent to \( B_i;B_i;c \). Hence there is no \( [u] \) such that \([u] \leq_{T} \langle \alpha \rangle \). 

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 \( u_{zero} \) be that element of \( V \) such that \( u_{zero} \leq_{T} \langle \alpha \rangle \).

Lemma 3.12: For all \( u \in V \), there is at most one \( x \) such that \( u_{zero} \leq_{T} \langle \text{prog}(x) \rangle \).

Proof: Suppose \( x \neq y \), but \( u_{zero} \leq_{T} \langle \text{prog}(x) \rangle \) and \( u_{zero} \leq_{T} \langle \text{prog}(y) \rangle \). Without loss of generality we can suppose that \( x_1 > y_1 \). \( \text{prog}(y) \); \( B_1 \) is not nonnegative, so by Lemma 3.11, there is no \( v \) such that \( u_{zero} \leq_{T} \langle \text{prog}(y) \rangle \); \( B_1 \). Hence \( x \neq y \). Therefore \( u \not=_{T} \langle B_1 \rangle \langle \text{false} \rangle \). \( \text{prog}(x) \); \( B_1 \) is, by Lemmas 3.8 and 3.6, T-equivalent to \( \text{prog}(z) \) for some \( z \). By Lemma 3.3, there is a \( w \) such that \( u_{zero} \leq_{T} \langle z \rangle \). Hence \( u \not=_{T} \langle B_1 \rangle \langle \text{true} \rangle \), a contradiction. So \( x = y \) is not possible.

We now prove that the relation between a state \( u \in V \) and a vector \( x \) defined by \( u_{zero} \leq_{T} \langle \text{prog}(x) \rangle \) is the desired bijection.

Lemma 3.13: There is a bijection \( \varphi: V \rightarrow N^5 \) such that \( \varphi(u) = x \) if and only if \( u_{zero} \leq_{T} \langle \text{prog}(x) \rangle \).

Proof: Let \( u \in V \). Since \( T \) is an \( \alpha \)-cut, there is an AB-program \( a \) such that \( u_{zero} \leq_{T} \langle A \rangle \). 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 \( \text{prog}(x) \) 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 \( \varphi \), \( u_{zero} \leq_{T} \langle \text{prog}(x) \rangle \) and \( u_{zero} \leq_{T} \langle \text{prog}(x) \rangle \). By Lemma 3.4, \( u = v \). To show that \( \varphi \) is a surjection, let \( x \in N^5 \). By Lemma 3.3, there is a \( u \) such that \( u_{zero} \leq_{T} \langle \text{prog}(x) \rangle \), 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 \( u \langle A \rangle \). Then \( u_{zero} \leq_{T} \langle \text{prog}(\varphi(u)) \rangle \) and \( u_{zero} \leq_{T} \langle \text{prog}(\varphi(u)) \rangle \). By Lemma 3.7, \( u_{zero} \leq_{T} \langle \text{prog}(\alpha(\varphi(u))) \rangle \). By Lemma 3.13, \( \varphi(v) = \alpha(\varphi(u)) \).
Conversely, suppose \( \varphi(v) = \sigma_1(\varphi(u)) \). Then \( u_{zero} \langle \text{prog}(\varphi(u)) \rangle_{TV} \) and \( u_{zero} \langle \text{prog}(\sigma_1(\varphi(u))) \rangle_{TV} \). By Lemma 3.7, \( u_{zero} \langle \text{prog}(\varphi(u)) \rangle_{TV} \). By Lemma 3.5, \( u \in A^* \).

(2) Without loss of generality let \( i = 1 \). Suppose \( u \in B_1 > y \). Then \( u_{zero} \langle \text{prog}(x); B_1 > y \). By Lemma 3.6, \( u_{zero} \langle A_1 x_1; A_2 x_2; A_3 x_3; A_4 x_4; A_5 > y \). Therefore \( x = \varphi(u) = \sigma_1(\varphi(v)) = \sigma_2(y) \).

Conversely, suppose \( \varphi(u) = \sigma_1(\varphi(v)) \) and \( \langle \text{prog}(x) \rangle_{TV} \). By Lemma 3.8, \( u_{zero} \langle \text{prog}(\sigma_1(x)); B_1 > y \). By Lemma 3.5, \( u \in B_1 > y \).

(3) Suppose \( u \models_T R \) and \( \varphi(u)_1 = \varphi(v)_1 \). Let \( \varphi(u) = x \), \( \varphi(v) = y \). Then \( u_{zero} \langle \text{prog}(x) \rangle_{TV} \) and \( u_{zero} \langle \text{prog}(x); B_2 > y \). By Lemmas 3.6 and 3.8, \( u_{zero} \langle A_1 x_1; A_2 x_2; A_3 x_3; A_4 x_4; A_5 > y \). By Lemma 3.5, \( u \in B_2 > y \).

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 \( \text{grid-scheme}^PDL \) infers \( p \).

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

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

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 \( \Pi_1 \)-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 \( S \), \( u \langle a_f S \rangle_S \) if and only if \( [\varphi(v)]_1 = f(X_S, \varphi(u)_1, \varphi(u)_2, \varphi(u)_3) \), where \( X_S = \{ \varphi(w)_1 \mid w \models_S R \} \).

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 \( 1 \leq i \)
246

\[ \leq 5): A_i \) to increment register \( i, B_i \) to decrement register \( i, \text{zero}_i \) to test register \( i \) for zero, and \( \neg R ? \) and \( \neg \neg R ? \) to test whether the contents of register 1 is in the oracle set \( X_S \). In a grid \( S \) the standard \( \text{PDL} \) semantics interprets \( a_f \) as a program which computes \( f \) i.e. that \( u \prec a_f \prec S, \varphi(u)_1, \varphi(u)_2, \varphi(u)_3 \).

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 \( f(X, x, y, z) \) of one set variable and three integer variables such that \( Y = \{ x | \forall x \subseteq N. \exists y. \forall z. f(X, x, y, z) = 0 \} \).

Corollary 4.2: There is a \( \text{PDL} \) formula \( p_Y \) such that for all natural numbers \( m \), the formula \( \text{zero}_1 \rightarrow <A_1^m>p_Y \) is valid in all grids if and only if \( m \in Y \).

Proof: By the preceding lemma, for all grids \( S \) and states \( u, u \sim_s (a_f) \text{zero}_1 \) if and only if \( \forall z \in N. f(X_S, \varphi(u)_1, \varphi(u)_2, \varphi(u)_3) = 0 \). The program \( B_i^*; A_i^* \) is capable of arbitrarily altering the contents of the \( i \)th register. Hence \( u \sim_s (a_f) \text{zero}_1 \) if and only if \( \forall z \in N. f(X_S, \varphi(u)_1, \varphi(u)_2, \varphi(u)_3) = 0 \). Similarly, \( u \sim_s (a_f) \text{zero}_1 \) if and only if \( \exists y \in N. \forall z \in N. f(X_S, \varphi(u)_1, \varphi(u)_2, \varphi(u)_3) = 0 \). Let \( p_Y \) be \( B_i^*; A_i^* \) if \( f(X_S, \varphi(u)_1, \varphi(u)_2, \varphi(u)_3) = 0 \). If \( u \sim_s \text{zero}_1 \), then \( u \sim_s (A_1^m)p_Y \) if and only if \( \exists y \in N. \forall z \in N. f(X_S, \varphi(u)_1, \varphi(u)_2, \varphi(u)_3) = 0 \). As \( S \) ranges over all grids, \( X_S \) ranges over all sets of nonnegative integers. Therefore, \( \text{zero}_1 \rightarrow <A_1^m>p_Y \) is valid in all grids if and only if \( \forall X \subseteq N. \exists y \in N. \forall z \in N. f(X, m, y, z) = 0 \), i.e. if and only if \( m \in Y \).

Proposition 4.3: The scheme implication (respectively, implication) problem for \( \text{PDL} \) is \( \Pi^1_1 \)-complete.

Proof: By Corollaries 3.14 (3.15) and 4.2, there is a \( \text{PDL} \) formula \( p_Y \) such that \( m \in Y \) if and only if grid-scheme \( \text{PDL} \) (respectively, \( \{a^*\}\text{scheme} \text{PDL} \) infers (implies) \( \text{zero}_1 \rightarrow <A_1^m>p_Y \). This proves that \( \Pi^1_1 \) is many-one reducible to the scheme implication (implication) problem for \( \text{PDL} \). It is not hard to show that either problem is in \( \Pi^1_1 \); we omit the proof.

We now define some sublanguages of \( \text{PDL} \) 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.

Theorem 4.4: If \( L \) is a subset of \( \text{PDL} \) which contains test-free-\( \text{PDL} \), then the scheme inference (respectively, implication) problem for \( L \) is \( \Pi^1_1 \)-complete.

Proof: The tests of \( p_Y \) are of the form \( \text{zero}_1, \neg \text{zero}_1, R, \) and \( \neg \neg R \). Choose new atomic programs \( C_1, \ldots, C_{12} \). Let \( q_Y \) be the result of substituting \( C_1 \) for \( \text{zero}_1, \ldots, C_{12} \) for \( \neg \neg R \) in \( p_Y \). Let free-scheme be grid-scheme \( \land (\land \leq 12 Q \rightarrow [C]Q) \land \text{zero}_1 \leftrightarrow <C>\text{true} \land \ldots \land \neg R \leftrightarrow <C_{12}>\text{true} \). We leave it to the reader to show that the set of deciding, for a given \( m \), whether or not free-scheme \( \text{PDL} \) (respectively, \( \{a^*\}\text{free-scheme} \text{PDL} \) infers (implies) \( \text{zero}_1 \rightarrow <A_1^m>q_Y \) is \( \Pi^1_1 \)-complete.

Definition: The set of programs, \( \Pi_d \), and the set of formulae, \( \Phi_d \) of deterministic propositional dynamic logic (\( \text{DPDL} \)) 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 \( b \)).

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

Theorem 4.5: If \( L \) is a subset of PDPL which contains atomic-test-DPDL, then the scheme inference problem for \( L \) is \( \Pi^1_1 \)-complete.

Proof. First, note that \( a \) of Lemma 4.1 can easily be written as a program in \( \Pi_d \). Second, note that for all programs \( a \) and formulae \( p \), \( <a>\ p \) is equivalent to while \( \neg p \) do \( \top \). Hence, there is a formula \( \Phi \) in \( \Pi_d \) which is equivalent to \( p \). Finally, note that every conjunct of grid-scheme is in \( \Pi_d \) except for zero-axiom =df\( <B_1^*;B_2^*;B_3^*;B_4^*;B_5^*>zero \). 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. Let the non-atomic tests of det-scheme and \( \Phi \) be \( r_1^?, \ldots, r_n^? \). A maximal occurrence of a test \( r_i^? \) in a formula \( q \) is an occurrence which is not itself contained in an occurrence of another test \( r_j^? \) occurring in \( q \). Choose new atomic propositions \( T_1, \ldots, T_k \). Let new-scheme and \( s_\Phi \) be the result of substituting, for \( 1 \leq i \leq k \), \( T_i \) for each maximal occurrence of \( r_i^? \) in det-scheme and \( \Phi \), and for \( 1 \leq i \leq k \), let \( i \) be the result of substituting, for \( j \neq i \), \( T_j \) for each maximal occurrence of \( r_i^? \) in \( \Phi \). Let equiv-scheme be new-scheme \& (\( \forall 1 \leq i \leq k \) \( T_i \leftrightarrow i \)). We leave it to the reader to show that the problem of deciding, for a given \( m \), whether or not equiv-scheme \( Q \) is \( \Pi^1_1 \)-complete. \( \square \)

5 Conclusions and Open Problems

Because of its many decidable properties, PDPL appears to be a reasonably tractable extension of propositional logic. However, we have revealed a dramatic contrast between PDPL and ordinary propositional logic in the case of the scheme deducibility problem, which is \( \Pi^1_1 \)-complete for PDPL, but decidable for propositional logic.

An important hint at the power of PDPL axiom schemes was provided by the observation of Salwicki and Pratt [2], who showed that the nonnegative integers could be characterized (as cuts of PDPL-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 PDPL ultimately rest on the finite model property of PDPL formulae, the Salwicki-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 PDPL extended with a looping predicate and the converse operation on programs [3]. Nevertheless, Streett has shown that this extension of PDPL is still decidable (in fact, elementary recursive) [7].

The degrees of undecidability (or decidability) of several restricted deducibility problems remain open questions.
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. David Harel is partially responsible for the proof of Theorem 4.4.

References


*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.