Propositional Dynamic Logic (PDL) was first defined by Fischer and Ladner [3, 4]. It plays the same role in Dynamic Logic that classical propositional logic plays in classical predicate logic. It describes the properties of the interaction between programs and propositions that are independent of the domain of computation.

1 Basic Definitions

1.1 Syntax

Syntactically, PDL is a blend of propositional logic, modal logic, and the algebra of regular events. It has expressions of two sorts: *programs* $p, q, r, \ldots$ and *propositions* or *formulas* $\varphi, \psi, \ldots$. There are countably many atomic symbols of each sort, as well as a variety of operators for forming compound expressions from simpler ones:

- propositional operators $\vee$, $\neg$
- program operators $\cup$, ;, *
- mixed operators ?, < >

Compound programs and propositions are defined by mutual induction, as follows. If φ, ψ are propositions and p, q are programs, then

$\varphi \vee \psi$	(propositional disjunction)
$\neg\varphi$	(propositional negation)
$<p>\varphi$	(modal possibility)

are propositions and

$p; q$	(sequential composition)
$p \cup q$	(nondeterministic choice)
p^*	(iteration)
$\varphi?$	(test)

are programs. The intuitive meanings of the less familiar of these constructs are as follows:

$$\begin{aligned}
<p>\varphi &= \text{“It is possible to execute } p \text{ and terminate in a state satisfying } \varphi.\text{”}\\
p; q &= \text{“Execute } p\text{, then execute } q.\text{”}\\
p \cup q &= \text{“Choose either } p \text{ or } q \text{ nondeterministically and execute it.”}\\
p^* &= \text{“Execute } p \text{ repeatedly a nondeterministically chosen finite number of times.”}\\
\varphi? &= \text{“Test } \varphi\text{; proceed if true, fail if false.”}
\end{aligned}$$

The set of propositions is denoted Φ and the set of programs is denoted Ψ. We avoid parentheses by assigning precedence to the operators: unary operators, including $<p>$, bind tighter than binary ones, and ; binds tighter than $\cup$. Also, under the semantics to be given in the next section, the operators ; and $\cup$ will turn out to be associative, so we may write $p; q; r$ and $p \cup q \cup r$ without ambiguity.

The primitive operators are chosen for their mathematical simplicity. A number of more conventional programming constructs can be defined from them. The propositional operators $\wedge$, $\Rightarrow$, $\Leftrightarrow$, false, and true are defined from $\vee$ and $\neg$ in the usual way. In addition:

$$\begin{aligned}
\mathsf{skip} &= \mathsf{true}? \\
\mathsf{fail} &= \mathsf{false}? \\
[p]\,\varphi &= \neg\texttt{<}p\texttt{>}\neg\varphi \\
\mathsf{if}\ \varphi_1 \Rightarrow p_1 \mid \cdots \mid \varphi_n \Rightarrow p_n\ \mathsf{fi} &= \varphi_1?; p_1 \cup \cdots \cup \varphi_n?; p_n \\
\mathsf{do}\ \varphi_1 \Rightarrow p_1 \mid \cdots \mid \varphi_n \Rightarrow p_n\ \mathsf{od} &= (\varphi_1?; p_1 \cup \cdots \cup \varphi_n?; p_n)^*; (\neg\varphi_1 \wedge \cdots \wedge \neg\varphi_n)? \\
\mathsf{if}\ \varphi\ \mathsf{then}\ p\ \mathsf{else}\ q &= \mathsf{if}\ \varphi \Rightarrow p \mid \neg\varphi \Rightarrow q\ \mathsf{fi} = \varphi?; p \cup \neg\varphi?; q \\
\mathsf{while}\ \varphi\ \mathsf{do}\ p &= \mathsf{do}\ \varphi \Rightarrow p\ \mathsf{od} = (\varphi?; p)^*; \neg\varphi? \\
\mathsf{repeat}\ p\ \mathsf{until}\ \varphi &= p; \mathsf{while}\ \neg\varphi\ \mathsf{do}\ p = p; (\neg\varphi?; p)^*; \varphi? \\
\{\varphi\}\,p\,\{\psi\} &= \varphi \Rightarrow [p]\,\psi
\end{aligned}$$

The propositions $\texttt{<}p\texttt{>}\varphi$ and $[p]\,\varphi$ are read "diamond p φ" and "box p φ", respectively. The latter has the intuitive meaning, "Whenever p terminates, it must do so in a state satisfying φ." Unlike $\texttt{<}p\texttt{>}\varphi$, $[p]\,\varphi$ does not imply that p terminates. Indeed, the formula $[p]\,\mathsf{false}$ asserts that no computation of p terminates. For fixed program p, the operator $\texttt{<}p\texttt{>}$ behaves like a possibility operator of modal logic (see [2,8]). The operator $[p]$ is the modal dual of $\texttt{<}p\texttt{>}$ and behaves like a modal necessity operator.

The ternary if-then-else and binary while-do operators are the usual *conditional test* and *while loop* constructs found in conventional imperative programming languages. The constructs if-|-fi and do-|-od are the *alternative guarded command* and *iterative guarded command* constructs, respectively (see [5]). The construct $\{\varphi\}\,p\,\{\psi\}$ is the classical Hoare partial correctness assertion [6].

1.2 Semantics

The formal semantics of PDL comes from modal logic. A *Kripke model* is a pair $M = (S^M, I^M)$ where $S^M = \{u, v, \ldots\}$ is an abstract set of *states* and I^M is an *interpretation function*. Each proposition φ is interpreted as a subset $\varphi^M \subseteq S^M$, and each program p is interpreted as binary relation p^M on S^M. We may think of φ^M as the set of states satisfying the proposition φ, and p^M as the input/output relation of the program p.

The function I^M assigns an arbitrary subset of S^M to each atomic proposition symbol and an arbitrary binary relation on S^M to each atomic program symbol. Compound programs and propositions receive their meanings inductively:

$$\begin{aligned}
(\varphi \vee \psi)^M &= \varphi^M \cup \psi^M \\
(\neg\varphi)^M &= S^M - \varphi^M \\
(\texttt{<}p\texttt{>}\varphi)^M &= p^M \circ \varphi^M = \{u \mid \exists v\ (u, v) \in p^M \text{ and } v \in \varphi^M\} \\
(p; q)^M &= p^M \circ q^M = \{(u, v) \mid \exists w\ (u, w) \in p^M \text{ and } (w, v) \in q^M\} \\
(p \cup q)^M &= p^M \cup q^M \\
(p^*)^M &= \text{reflexive transitive closure of } p^M \\
(\varphi?)^M &= \{(u, u) \mid u \in \varphi^M\}
\end{aligned}$$

where the reflexive-transitive closure of a binary relation ρ is

$$\bigcup_{n<\omega} \rho^n\ ,$$

where

$$\rho^0 = \{(u, u) \mid u \in S^M\} \qquad\qquad \rho^{n+1} = \rho \circ \rho^n.$$

The symbol $\circ$ denotes relational composition.

The defined operators inherit their meanings from these definitions:

$$\begin{aligned}
(\varphi \wedge \psi)^M &= \varphi^M \cap \psi^M \\
(\texttt{[}p\texttt{]}\,\varphi)^M &= \{u \mid \forall v\ (u,v) \in p^M \Rightarrow v \in \varphi^M\} \\
\mathsf{true}^M &= S^M \\
\mathsf{false}^M &= \varnothing \\
\mathsf{skip}^M &= \{(u,u) \mid u \in S^M\} \\
\mathsf{fail}^M &= \varnothing
\end{aligned}$$

The if-then-else, while-do, and guarded commands also receive meanings from the above definitions.

It can be argued that the input/output relations given by these definitions capture the intuitive operational meanings of these constructs. For example, the relation associated with the program while φ do p is the set of pairs (u, v) for which there exist states $u_0, u_1, \ldots, u_n$, $n \geq 0$, such that $u = u_0$, $v = u_n$, $u_i \in \varphi^M$ and $(u_i, u_{i+1}) \in p^M$ for $0 \leq i < n$, and $u_n \in \neg\varphi^M$.

We often write $M, u \models \varphi$ for $u \in \varphi^M$ and say that u *satisfies* φ *in* M. We may write $u \models \varphi$ when M is understood. We write $M \models \varphi$ if $M, u \models \varphi$ for all $u \in M$, and we write $\models \varphi$ and say that φ is *valid* if $M \models \varphi$ for all M. We say that φ is *satisfiable* if $M, u \models \varphi$ for some M, u. If Σ is a set of propositions, we write $M \models \Sigma$ if $M \models \varphi$ for all $\varphi \in \Sigma$. A proposition ψ is said to be a *logical consequence* of Σ if for all M, $M \models \psi$ whenever $M \models \Sigma$, in which case we write $\Sigma \models \psi$. (Note that this is *not* the same as saying that $M, u \models \psi$ whenever $M, u \models \Sigma$.) We say that an inference rule

$$\frac{\Sigma}{\psi}$$

is *sound* if ψ is a logical consequence of Σ.

This version of PDL is usually called *regular* PDL because of the primitive operators $\cup, ;, *$, which are familiar from Kleene algebra. Programs can be viewed as regular expressions over the atomic programs and tests. In fact, it can be shown that if φ is an atomic proposition symbol, then any two test-free programs p, q are equivalent as regular expressions if and only if the formula $\texttt{<}p\texttt{>}\varphi \Leftrightarrow \texttt{<}q\texttt{>}\varphi$ is valid.

Example 1. *Let φ be an atomic proposition, let p be an atomic program, and let $M = (S^M, I^M)$ be a Kripke model with*

$$S^M = \{u, v, w\} \qquad \varphi^M = \{u, v\} \qquad p^M = \{(u, v), (u, w), (v, w), (w, v)\}$$

Then $u \models \texttt{<}p\texttt{>}\neg\varphi \wedge \texttt{<}p\texttt{>}\varphi$, but $v \models \texttt{[}p\texttt{]}\,\neg\varphi$ and $w \models \texttt{[}p\texttt{]}\,\varphi$. Moreover,

$$M \models \texttt{<}p^*\texttt{>}\,\texttt{[}(p;p)^*\texttt{]}\,\varphi \wedge \texttt{<}p^*\texttt{>}\,\texttt{[}(p;p)^*\texttt{]}\,\neg\varphi.$$

Other semantics besides Kripke semantics have been studied [1, 9–12, 14].

1.3 Computation Sequences

Let p be a program. A *finite computation sequence* of p is a finite-length string of atomic programs and tests representing a possible sequence of atomic steps that may occur in a halting execution of p. The set of all such sequences is denoted $\mathsf{CS}(p)$. We use the word "possible" loosely—$\mathsf{CS}(p)$ is determined by the syntax of p alone, and may contain strings that are never executed in any interpretation.

Formally, the set $\mathsf{CS}(p)$ is defined by induction on syntax:

$$\begin{aligned}
\mathsf{CS}(p) &= \{p\},\ \ p \text{ an atomic program or test}\\
\mathsf{CS}(p;q) &= \{\alpha;\beta \mid \alpha \in \mathsf{CS}(p),\ \ \beta \in \mathsf{CS}(q)\}\\
\mathsf{CS}(p \cup q) &= \mathsf{CS}(p) \cup \mathsf{CS}(q)\\
\mathsf{CS}(p^*) &= \bigcup_{n\geq 0} \mathsf{CS}(p^n)
\end{aligned}$$

where $p^0 = \mathsf{skip}$ and $p^{n+1} = p;p^n$. For example, if p is an atomic program and φ is an atomic formula, then the program

$$\mathsf{while}\ \varphi\ \mathsf{do}\ p = (\varphi?;p)^*;\neg\varphi?$$

has as computation sequences all strings of the form

$$\varphi?;p;\varphi?;p;\cdots;\varphi?;p;\mathsf{skip};\neg\varphi?.$$

Note that each finite computation sequence q of a program p is itself a program, and

$$\mathsf{CS}(q) = \{q\}.$$

Moreover, the following proposition is not difficult to prove by induction on syntax:

Proposition 2. $p^M = \bigcup_{q\in\mathsf{CS}(p)} q^M$.

2 A Deductive System for PDL

The following Hilbert-style axiom system for PDL was formulated by Segerberg [13].

Axioms of PDL

1. *Axioms for propositional logic*
2. $\texttt{<}p\texttt{>}\varphi \wedge [p]\psi \Rightarrow \texttt{<}p\texttt{>}(\varphi \wedge \psi)$
3. $\texttt{<}p\texttt{>}(\varphi \vee \psi) \Leftrightarrow \texttt{<}p\texttt{>}\varphi \vee \texttt{<}p\texttt{>}\psi$
4. $\texttt{<}p \cup q\texttt{>}\varphi \Leftrightarrow \texttt{<}p\texttt{>}\varphi \vee \texttt{<}q\texttt{>}\varphi$
5. $\texttt{<}p;q\texttt{>}\varphi \Leftrightarrow \texttt{<}p\texttt{>}\texttt{<}q\texttt{>}\varphi$
6. $\texttt{<}\psi?\texttt{>}\varphi \Leftrightarrow \psi \wedge \varphi$
7. $(\varphi \vee \texttt{<}p\texttt{>}\texttt{<}p^*\texttt{>}\varphi) \Rightarrow \texttt{<}p^*\texttt{>}\varphi$
8. $\texttt{<}p^*\texttt{>}\varphi \Rightarrow (\varphi \vee \texttt{<}p^*\texttt{>}(\neg\varphi \wedge \texttt{<}p\texttt{>}\varphi))$

Axioms 2 and 3 are not particular to PDL, but hold in all normal modal systems (see [2, 8]). Axiom 8 is called the PDL *induction axiom*, and is better known in its dual form (Theorem 3(viii) below).

Rules of Inference

1. *Modus ponens:*
$$\frac{\varphi,\ \varphi \Rightarrow \psi}{\psi}$$

2. *Modal generalization:*
$$\frac{\varphi}{[p]\,\varphi}$$

We write $\vdash \varphi$ if the proposition φ is a theorem of this system, and say that φ is *consistent* if not $\vdash \neg\varphi$. A set Σ of propositions is *consistent* if all finite conjunctions of elements of Σ are consistent.

The soundness of these axioms and rules over the Kripke semantics can be established by elementary arguments in relational algebra.

3 Basic Properties

We list some basic theorems and derived rules of PDL that are provable in the deductive system of the previous section.

Theorem 3. *The following are theorems of* PDL*:*

(i) *all propositional tautologies*

(ii) $[p]\,(\varphi \Rightarrow \psi) \Rightarrow ([p]\,\varphi \Rightarrow [p]\,\psi)$

(iii) $[p]\,(\varphi \wedge \psi) \Leftrightarrow [p]\,\varphi \wedge [p]\,\psi$

(iv) $[p \cup q]\,\varphi \Leftrightarrow [p]\,\varphi \wedge [q]\,\varphi$

(v) $[p;q]\,\varphi \Leftrightarrow [p]\,[q]\,\varphi$

(vi) $[\varphi?]\,\psi \Leftrightarrow (\varphi \Rightarrow \psi)$

(vii) $[p^*]\,\varphi \Rightarrow \varphi \wedge [p]\,[p^*]\,\varphi$

(viii) $(\varphi \wedge [p^*]\,(\varphi \Rightarrow [p]\,\varphi)) \Rightarrow [p^*]\,\varphi$

(ix) $<p>(\varphi \wedge \psi) \Rightarrow <p>\varphi \wedge <p>\psi$

(x) $[p]\,\varphi \vee [p]\,\psi \Rightarrow [p]\,(\varphi \vee \psi)$

(xi) $<p^*;p^*>\varphi \Leftrightarrow <p^*>\varphi$

(xii) $<p^{**}>\varphi \Leftrightarrow <p^*>\varphi$

(xiii) $(\varphi \vee <p><p^*>\varphi) \Leftrightarrow <p^*>\varphi$

(xiv) $<p^*>\varphi \Leftrightarrow (\varphi \vee <p^*>(\neg\varphi \wedge <p>\varphi))$

(xv) $(\varphi \wedge [p^*]\,(\varphi \Rightarrow [p]\,\varphi)) \Leftrightarrow [p^*]\,\varphi$

Theorem 4. *The following are sound rules of inference of* PDL*:*

(i) *monotonicity of* $<p>$*:*
$$\frac{\varphi \Rightarrow \psi}{<p>\varphi \Rightarrow <p>\psi}$$

(ii) *monotonicity of* [p]*:*

$$\frac{\varphi \Rightarrow \psi}{[p]\,\varphi \Rightarrow [p]\,\psi}$$

(iii) *reflexive-transitive closure:*

$$\frac{(\varphi \vee \texttt{<}p\texttt{>}\psi) \Rightarrow \psi}{\texttt{<}p^*\texttt{>}\varphi \Rightarrow \psi}$$

(iv) *loop invariance rule:*

$$\frac{\psi \Rightarrow [p]\,\psi}{\psi \Rightarrow [p^*]\,\psi}$$

(v) *Hoare composition rule:*

$$\frac{\{\varphi\}\,p\,\{\sigma\} \quad \{\sigma\}\,q\,\{\psi\}}{\{\varphi\}\,p;q\,\{\psi\}}$$

(vi) *Hoare conditional rule:*

$$\frac{\{\varphi \wedge \sigma\}\,p\,\{\psi\} \quad \{\neg\varphi \wedge \sigma\}\,q\,\{\psi\}}{\{\sigma\}\,\mathsf{if}\ \varphi\ \mathsf{then}\ p\ \mathsf{else}\ q\,\{\psi\}}$$

(vii) *Hoare* while *rule:*

$$\frac{\{\varphi \wedge \psi\}\,p\,\{\psi\}}{\{\psi\}\,\mathsf{while}\ \varphi\ \mathsf{do}\ p\,\{\neg\varphi \wedge \psi\}}$$

The properties of Theorem 3(ii-viii) are the modal duals of Axioms 2-8. The converses of Theorem 3(ix,x) are *not* valid. They are violated in state u of the model of Example 1. The rules of Theorem 4(i,ii) say that the constructs $\texttt{<}p\texttt{>}\varphi$ and $[p]\,\varphi$ are *monotone* in φ with respect to the ordering of logical implication. These constructs are also monotone and antitone in p, respectively, as asserted by the following metatheorem:

Proposition 5. *If $p^M \subseteq q^M$, then for all φ,*

1. $M \models \texttt{<}p\texttt{>}\varphi \Rightarrow \texttt{<}q\texttt{>}\varphi$
2. $M \models [q]\,\varphi \Rightarrow [p]\,\varphi$.

These follow from Axiom 4 and Theorem 3(iv).

3.1 The * Operator, Induction, and Reflexive-Transitive Closure

The iteration operator * is interpreted as the reflexive-transitive closure operator on binary relations. It is the means by which looping is coded in PDL. Looping introduces a level of complexity to PDL beyond the other operators. Because of it, PDL is not compact: the set

$$\{\texttt{<}p^*\texttt{>}\varphi\} \cup \{\neg\varphi,\ \neg\texttt{<}p\texttt{>}\varphi,\ \neg\texttt{<}p^2\texttt{>}\varphi,\ \ldots\} \tag{1}$$

is finitely satisfiable but not satisfiable. This seems to say that looping is inherently infinitary; it is thus rather surprising that there should be a finitary complete axiomatization.

The dual propositions Axiom 8 and Theorem 3(viii) are jointly called the PDL *induction axiom*. Together with Axiom 7, they completely axiomatize the behavior of *. Intuitively, the induction axiom in the form of Theorem 3(viii) says: if φ is true initially, and if the truth of φ is preserved by the program p, then φ will be true after any number of iterations of p. It is very similar to the induction axiom of Peano arithmetic:

$$\varphi(0) \wedge \forall n\ (\varphi(n) \Rightarrow \varphi(n+1)) \ \Rightarrow\ \forall n\ \varphi(n).$$

Here $\varphi(0)$ is the basis of the induction and $\forall n\ (\varphi(n) \Rightarrow \varphi(n+1))$ is the induction step, from which the conclusion $\forall n\ \varphi(n)$ may be drawn. In the PDL induction axiom, the basis is φ and the induction step is $[p^*](\varphi \Rightarrow [p]\varphi)$, from which the conclusion $[p^*]\varphi$ may be drawn.

The induction axiom is closely related to the reflexive-transitive closure rule (Theorem 4(iii)). The significance of this rule is best described in terms of its relationship to Axiom 7. This axiom is obtained by substituting $<p^*>\varphi$ for ψ in the premise of the reflexive-transitive closure rule. Axiom 7 thus says that $<p^*>\varphi$ is a solution of

$$(\varphi \vee <p>X) \Rightarrow X\ ; \tag{2}$$

the reflexive-transitive closure rule says that it is the *strongest* solution to (2) (with respect to logical implication) among all PDL propositions.

The relationship between the induction axiom (Axiom 8), the reflexive-transitive closure rule (Theorem 4(iii)), and the rule of loop invariance (Theorem 4(iv)) is summed up in the following proposition. We emphasize that this result is purely proof-theoretic and is independent of the semantics of §1.2.

Proposition 6. *In* PDL *without the induction axiom, the following axioms and rules are interderivable:*

(i) *the induction axiom (Axiom 8);*

(ii) *the loop invariance rule (Theorem 4(iv));*

(iii) *the reflexive-transitive closure rule (Theorem 4(iii)).*

Proof. First we show that the monotonicity rule (Theorem 4(ii)) is derivable in PDL without induction. Assuming the premise $\varphi \Rightarrow \psi$ and applying modal generalization, we obtain $[p](\varphi \Rightarrow \psi)$; the conclusion $[p]\varphi \Rightarrow [p]\psi$ then follows from Theorem 3(ii) and modus ponens. The dual monotonicity rule (Theorem 4(i)) can be derived from this rule by pure propositional reasoning.

(i) $\Rightarrow$ (ii): Assume the premise of (ii):

$$\varphi \ \Rightarrow\ [p]\varphi.$$

By modal generalization,

$$[p^*](\varphi \Rightarrow [p]\varphi)\ ,$$

thus

$$\begin{aligned}\varphi \ &\Rightarrow\ \varphi \wedge [p^*](\varphi \Rightarrow [p]\varphi)\\ &\Rightarrow\ [p^*]\varphi.\end{aligned}$$

The first implication is by propositional reasoning, and the second is the box form of the induction axiom (Theorem 3(viii)). By transitivity of implication, we obtain

$$\varphi \ \Rightarrow\ [p^*]\varphi\ ,$$

which is the conclusion of (ii).

(ii) $\Rightarrow$ (iii): Dualizing the rule (iii) by purely propositional reasoning, we obtain a rule

$$\frac{\psi \ \Rightarrow\ \varphi \wedge [p]\psi}{\psi \ \Rightarrow\ [p^*]\varphi} \tag{3}$$

equipollent with (iii). It thus suffices to derive (3) from (ii). From the premise of (3), we obtain by propositional reasoning the two formulas

$$\psi \;\Rightarrow\; \varphi \tag{4}$$
$$\psi \;\Rightarrow\; [p]\,\psi. \tag{5}$$

Applying (ii) to (5), we obtain

$$\psi \;\Rightarrow\; [p^*]\,\psi\;,$$

which by (4) and monotonicity gives

$$\psi \;\Rightarrow\; [p^*]\,\varphi.$$

This is the conclusion of (3).

(iii) $\Rightarrow$ (i): By Axiom 3, propositional reasoning, and Axiom 7, we have

$$\begin{aligned}
&\varphi \vee \langle p\rangle(\varphi \vee \langle p^*\rangle(\neg\varphi \wedge \langle p\rangle\varphi))\\
&\Rightarrow\;\; \varphi \vee \langle p\rangle\varphi \vee \langle p\rangle\langle p^*\rangle(\neg\varphi \wedge \langle p\rangle\varphi)\\
&\Rightarrow\;\; \varphi \vee (\neg\varphi \wedge \langle p\rangle\varphi) \vee \langle p\rangle\langle p^*\rangle(\neg\varphi \wedge \langle p\rangle\varphi)\\
&\Rightarrow\;\; \varphi \vee \langle p^*\rangle(\neg\varphi \wedge \langle p\rangle\varphi).
\end{aligned}$$

By transitivity of implication,

$$\varphi \vee \langle p\rangle(\varphi \vee \langle p^*\rangle(\neg\varphi \wedge \langle p\rangle\varphi)) \;\Rightarrow\; \varphi \vee \langle p^*\rangle(\neg\varphi \wedge \langle p\rangle\varphi).$$

Applying (iii), we obtain the induction axiom:

$$\langle p^*\rangle\varphi \;\Rightarrow\; \varphi \vee \langle p^*\rangle(\neg\varphi \wedge \langle p\rangle\varphi). \qquad \square$$

4 Encoding Hoare Logic

Dynamic Logic subsumes Hoare Logic. As an illustration, we show how to derive the Hoare while rule (Theorem 4(vii)) in PDL. The other Hoare rules are also derivable.

Assume that the premise

$$\{\varphi \wedge \psi\}p\{\psi\} = (\varphi \wedge \psi) \Rightarrow [p]\,\psi \tag{6}$$

holds. We wish to derive the conclusion

$$\{\psi\}\mathsf{while}\ \varphi\ \mathsf{do}\ p\{\neg\varphi \wedge \psi\} = \psi \Rightarrow [(\varphi?;p)^*;\neg\varphi?]\,(\neg\varphi \wedge \psi). \tag{7}$$

Using Theorem 3(v,vi) and propositional reasoning, the right-hand-side of (6) is equivalent to

$$\psi \;\Rightarrow\; [\varphi?;p]\,\psi.$$

Applying the loop invariance rule (Theorem 4(iv)), we obtain

$$\psi \;\Rightarrow\; [(\varphi?;p)^*]\,\psi.$$

By the monotonicity of $[(\varphi?;p)^*]$ (Theorem 4(ii)) and propositional reasoning,

$$\psi \;\Rightarrow\; [(\varphi?;p)^*]\,(\neg\varphi \Rightarrow \neg\varphi \wedge \psi).$$

Again by Theorem 3(v,vi) and propositional reasoning, we obtain the right-hand-side of (7).

References

[1] F. Berman. A completeness technique for D-axiomatizable semantics. In *Proc. 11th Symp. Theory of Comput.*, pages 160–166. ACM, 1979.

[2] B. F. Chellas. *Modal Logic: An Introduction.* Cambridge University Press, 1980.

[3] Michael J. Fischer and Richard E. Ladner. Propositional modal logic of programs. In *Proc. 9th Symp. Theory of Comput.*, pages 286–294. ACM, 1977.

[4] Michael J. Fischer and Richard E. Ladner. Propositional dynamic logic of regular programs. *J. Comput. Syst. Sci.*, 18(2):194–211, 1979.

[5] D. Gries. *The Science of Programming.* Springer-Verlag, 1981.

[6] C. A. R. Hoare. An axiomatic basis for computer programming. *Comm. Assoc. Comput. Mach.*, 12:576–80, 1969.

[7] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley, 1979.

[8] G. E. Hughes and M. J. Cresswell. *An Introduction to Modal Logic.* Methuen, 1968.

[9] Dexter Kozen. On the duality of dynamic algebras and Kripke models. In E. Engeler, editor, *Proc. Workshop on Logic of Programs*, volume 125 of *Lecture Notes in Computer Science*, pages 1–11. Springer-Verlag, 1979.

[10] Dexter Kozen. A representation theorem for models of *-free *PDL*. In *Proc. 7th Colloq. Automata, Languages, and Programming*, pages 351–362. EATCS, July 1980.

[11] H. Nishimura. Sequential method in propositional dynamic logic. *Acta Informatica*, 12:377–400, 1979.

[12] V. R. Pratt. Models of program logics. In *Proc. 20th Symp. Found. Comput. Sci.*, pages 115–122. IEEE, 1979.

[13] K. Segerberg. A completeness theorem in the modal logic of programs (preliminary report). *Not. Amer. Math. Soc.*, 24(6):A–552, 1977.

[14] V. Trnkova and J. Reiterman. Dynamic algebras which are not Kripke structures. In *Proc. 9th Symp. on Math. Found. Comput. Sci.*, pages 528–538, 1980.