Propositional Dynamic Logic (PDL) is a formal language for reasoning about flowchart programs. Although simple in structure, PDL can be used to represent both sequential and iterative programs and to assert program correctness, termination and failure.

This paper is based on a lecture given at the 1981 Symposium on Algorithmic Logic and outlines results from [Be]. We describe three classes of models which differ in their interpretation of the iterative program operator \( ^* \) (the Kleene star) and show that these interpretations cannot be distinguished by any PDL assertion. We also discuss why the continuity of the Kleene star cannot be precisely characterized within a propositional system. We maintain however that continuity is primarily an aid to our intuition and that nonstandard models of PDL are perfectly adequate for interpreting properties of flowchart programs.

We briefly introduce PDL. For a more detailed discussion, see [F&L] or [Be].

Syntax

PDL assertions and programs are built from primitive sets \( \Phi_0 \) (basic assertions) and \( \Sigma_0 \) (basic programs) respectively as follows:

1) Basic assertions are assertions.
2) If \( P \) and \( Q \) are assertions then so are \( \neg P \) and \( P \lor Q \).
3) Basic programs are programs. If \( a \) and \( b \) are programs and \( P \) is an assertion then \( a;b, a\lor b, a^* \) and \( P\? \) are programs.
4) If \( a \) is a program and \( P \) is an assertion then \( <a>P \) and [a]P are assertions.

Intuitively, we interpret
\( a;b \) as "execute \( a \) then execute \( b \),"
\( a\lor b \) as "execute \( a \) or execute \( b \),"
\( a^* \) as "execute \( a \) a nondeterministically chosen number of times,"
\( P\? \) as "program \( P \) may be executed but is not forced to be executed,"
\( \neg P \) as "not program \( P \)."
\(<\text{a}\>\text{P}\) as "there is an execution of program a which terminates with P true,"
\([\text{a}\>]\text{P}\) as "whenever program a terminates, assertion P is true."

(Note that \(<\text{a}\>\text{P}\) and \([\text{a}\>]\text{P}\) behave as dual operators, i.e. \(<\text{a}\>\text{P} \equiv -[\text{a}]\neg \text{P}\).)

With the intended interpretation, we can describe a surprising number of properties of programs by PDL assertions. A few examples:

**Partial Correctness** \((\text{P[a]}\text{Q})\) can be represented in PDL by the assertion \(\text{P} \rightarrow [\text{a}]\text{Q}\) interpreted as the statement "Whenever P is true then whenever a halts, Q is true."

**Nontermination**, can be represented in PDL by the assertion \([\text{a}]\text{false}\) interpreted as the statement "Program a never halts."

A **Loop Invariant** for a program a can be represented by the PDL assertion \([\text{a}^*]\text{P}\) interpreted as the statement "P is true after any number of iterations of program a."

**Semantics**

There are several different ways of constructing models faithful to the usual interpretation of the propositional constructs.

A **model** of PDL is a triple \(M = (W, \Pi, \rho)\) where

- \(W\) is a set of states,
- \(\Pi:\emptyset \rightarrow 2^W\) evaluates all basic assertions,
- \(\rho:\{\text{all programs}\} \rightarrow 2^W \times W\) evaluates all programs.

We extend \(\Pi\) to evaluate all PDL assertions as follows:

\[
\Pi(\text{P} \lor \text{Q}) = \Pi(\text{P}) \cup \Pi(\text{Q})
\]

\[
\Pi(\neg \text{P}) = W - \Pi(\text{P})
\]

\[
\Pi(<\text{a}>\text{P}) = \{w|\exists v((w,v) \in \text{ep}(a) \text{ and } v \in \Pi(\text{P}))\}
\]

\[
\Pi([\text{a}]\text{P}) = \{w|\exists x((w,v) \in \text{ep}(a) \text{ implies } v \in \Pi(\text{P}))\}.
\]

This definition of model ensures that the boolean and modal operators \((-, \lor, <>, [])\) are interpreted in the usual way but does not constrain the interpretation of the program operators \(;\), \(\lor\), \(*\), \(?\). For example, the generality of this definition permits us to have a model \(M\) and a state \(w\) in \(M\) in which the program \(a \lor b\) ("select a or b") is executable but neither program a nor program b is executable. Therefore, we want to consider models of PDL in which the program operator

\;
behaves like sequential execution,
\(\lor\) behaves like nondeterministic branching,
\(*\) behaves like iteration and
\(?\) behaves like test.