Chapter 4

Dynamic logics

Floyd [1967] and Hoare [1969] have used correctness assertions
\{ \varphi \} \alpha \{ \psi \}
for stating that executing the program \( \alpha \) in a state satisfying \( \varphi \) will always end in a state satisfying \( \psi \). There are proof systems that allow the stepwise derivation of such assertions about programs \( \alpha \) from similar assertions on components of \( \alpha \).

The possible steps a program can take can be understood as a Kripke structure, and this idea leads to the definition of dynamic logics [Pratt, 1976; Fischer and Ladner, 1977; 1979], in which we have an infinite number of modal operators \( [\alpha] \) for all possible programs \( \alpha \), and the above correctness assertion corresponds to the dynamic logic formula
\( \varphi \rightarrow [\alpha] \psi \).

4.1 Propositional dynamic logic PDL

4.1.1 The language of the propositional dynamic logic

The language of the propositional dynamic logic is based on a set \( P \) of atomic propositions and a set \( A \) of atomic programs.

Then formulas and programs based on \( A \) and \( P \) are defined recursively as follows.

1. The constant true \( \top \) is a formula.
2. Every atomic proposition \( p \in P \) is a formula.
3. If \( \varphi \) and \( \psi \) are formulae, then so are \( \neg \varphi \), \( \varphi \lor \psi \), \( \varphi \land \psi \), \( \varphi \leftrightarrow \psi \) and \( \varphi \rightarrow \psi \).
4. Every atomic program \( \alpha \in A \) is a program.
5. If \( \alpha \) and \( \beta \) are programs and \( \varphi \) is a formula, then \( \alpha \cup \beta \), \( \alpha; \beta \), \( \alpha^* \) and \( \varphi? \) are programs.
6. If $\alpha$ is a program and $\varphi$ is a formula, then $[\alpha]\varphi$ and $\langle\alpha\rangle\varphi$ are formulae.

The intuitive meaning of $\alpha \cup \beta$ is nondeterministic choice: execute either $\alpha$ or $\beta$. The construct $\alpha;\beta$ is sequential composition: first execute $\alpha$ and then execute $\beta$. The $\alpha^*$ programs correspond to looping: execute $\alpha$ zero or any finite number of times. Conditional execution is expressed by $\varphi?$: the execution continues only if $\varphi$ is true.

When considering ordinary programming languages, the atomic programs would most naturally be assignment statements that take us from one possible world to another in which one or more atomic propositions have a different truth-value. Of course, the definition of PDL allows more general types of atomic programs, for example ones that are nondeterministic and therefore do not define exactly one unique successor for each possible world.

Familiar constructs from procedural programming languages can be defined in terms of the PDL program constructs as follows.

\[
\begin{align*}
\text{if } \varphi \text{ then } \alpha \text{ else } \beta & \equiv (\varphi?;\alpha) \cup (\neg \varphi?; \beta) \\
\text{while } \varphi \text{ do } \alpha & \equiv (\varphi?;\alpha)^*; \neg \varphi? \\
\text{repeat } \alpha \text{ until } \varphi & \equiv \alpha; (\neg \varphi?; \alpha)^*; \varphi?
\end{align*}
\]

### 4.1.2 The semantics of propositional dynamic logic

For given sets $P$ of atomic propositions and sets $A = \{a_1, \ldots, a_n\}$ of atomic programs, a model in the propositional dynamic logic is a tuple $\mathcal{M} = (S, R_{a_1}, \ldots, R_{a_n}, V)$ where $S$ is a set of possible worlds and $R_a$ for every $a \in A$ is the accessibility relation for atomic program $a$, and $V : S \times P \rightarrow \{0, 1\}$ is a valuation that assigns truth values to atomic propositions in every possible world.

Analogously to modal logics defined earlier we can talk about the frame $\mathcal{F} = \{S, R_{a_1}, \ldots, R_{a_n}\}$.

The truth-definition of modal formulae in possible worlds $s \in S$ is as follows (the truth-definition of non-modal connectives is just like earlier.)

\[\mathcal{M} \models_s [\alpha]\varphi \text{ iff } \mathcal{M} \models_t \varphi \text{ for every possible world } t \text{ such that } s R_{\alpha} t.\]

Here the difference to earlier modal logics is that there is an infinite number of different accessibility relations for all of the non-atomic programs $\alpha$. The accessibility relations for these programs are defined as follows.

1. The accessibility relations of atomic programs $a \in A$ are given explicitly by the model $\mathcal{M}$.
2. If $\alpha$ and $\beta$ have respectively accessibility relations $R_\alpha$ and $R_\beta$, then
   \[
   \begin{align*}
   (a) & \quad R_{\alpha \cup \beta} = R_\alpha \cup R_\beta \text{ (union of relations)}, \\
   (b) & \quad R_{\alpha;\beta} = \{ (s, t) | u \in S, s R_\alpha u, u R_\beta t \} \text{ (composition of relations),} \\
   (c) & \quad R_{\alpha^*} = (R_\alpha)^* \text{ (the reflexive transitive closure of a relation), and} \\
   (d) & \quad R_{\varphi?} = \{ (s, s) | s \in S, \mathcal{M} \models_s \varphi \}
   \end{align*}
   \]
4.1.3 Propositional dynamic logic axiomatically

The propositional dynamic logic is a logic as defined in Definition 2.8, and hence its set of theorems includes all the theorems of the classical propositional logic and its (modal) substitution instances, and it is closed under substitution and modus ponens. Additionally, axioms for describing the properties of the modal operators are needed.

**Definition 4.1 (Propositional dynamic logic)** PDL is the smallest logic (according to Definition 2.8) that is closed under the necessitation/generalization rule

\[ \text{If } \vdash_{PDL} \varphi, \text{ then } \vdash_{PDL} [\alpha] \varphi \text{ for any program } \alpha. \]

and contains the following axiom schemata for programs \( \alpha \) and \( \beta \) and formulae/propositions \( p \) and \( q \).

1. \( [\alpha](p \rightarrow q) \rightarrow ([\alpha]p \rightarrow [\alpha]q) \)
2. \( [\alpha; \beta]p \leftrightarrow [\alpha][\beta]p \)
3. \( [\alpha \cup \beta]p \leftrightarrow [\alpha]p \wedge [\beta]p \)
4. \( [\alpha^*]p \leftrightarrow p \wedge [\alpha][\alpha^*]p \)
5. \( [\alpha^*](p \rightarrow [\alpha]p) \rightarrow (p \rightarrow [\alpha^*]p) \)
6. \( [q?]p \leftrightarrow (q \rightarrow p) \)

4.1.4 Relation to linear time temporal logics

Variants of the temporal operators \( F, X, G \) and \( U \) can be translated to the propositional dynamic logic. Essentially, temporal logics can be understood as dynamic logics that do not make the different programs explicit, or alternatively, have only one atomic program \( a \).

Computer scientists define the operators \( F, G \) and \( U \) to include the current time point in the scope of quantification over future; that is, the current time is part of the future. With this definition of the operators we have the following embedding of the linear time temporal operators in the propositional dynamic logic.

<table>
<thead>
<tr>
<th>operator</th>
<th>translation to PDL</th>
</tr>
</thead>
<tbody>
<tr>
<td>( F\varphi )</td>
<td>( (a^*)\varphi )</td>
</tr>
<tr>
<td>( X\varphi )</td>
<td>( (a)\varphi )</td>
</tr>
<tr>
<td>( G\varphi )</td>
<td>( [a^*]\varphi )</td>
</tr>
<tr>
<td>( \varphi U \psi )</td>
<td>( (\varphi^<em>; a^</em>)\psi )</td>
</tr>
</tbody>
</table>

4.1.5 Relation to unimodal logics

There are simple embeddings of some the unimodal logics with operators \( \Box \) and \( \Diamond \) in the dynamic logic.
logic | translation of $\square \varphi$ to PDL
---|---
$K$ | $\left[a\right]\varphi$
$S4$ | $\left[a^*\right]\varphi$
$S5$ | $\left[(a \cup a^{-1})^*\right]\varphi$

Here we need the converse connective $^{-1}$ that is given a semantics as follows.

$$R_{\alpha^{-1}} = \{ \langle t, s \rangle | \langle s, t \rangle \in \alpha \}.$$  

The converse program $\alpha^{-1}$ of $\alpha$ computes the computation of $\alpha$ in reverse from the end states back to the starting states.

With the above translation the satisfiability problems in the logics in question can be directly translated to PDL. Notice that the relations in the PDL frames may be arbitrary (K frames), and the reflexivity and transitivity for S4 and reflexivity, transitivity and symmetry for S5 are properties of the PDL modal operator, and need not be properties of the frames in question.

### 4.1.6 Relation to Hoare logic

The Hoare logic is a programming logic that may be used for proving the correctness of programs in procedural programming languages [Hoare, 1969]. There are different variants of the logic, depending on the definition of programs and the language in which the correctness assertions are expressed. Here we consider a propositional variant of the logic, in which the assertions are propositional formulae.

We can consider atomic programs $a$ that are associated with assertions $\{ \varphi \} a \{ \psi \}$ explaining their behavior. Assertions about more complex programs can be proved by using these simplest assertions and inference rules concerning the program constructs.

The Hoare logic has the following inference rules.

$$\frac{\{ \varphi \} \alpha \{ \sigma \}, \{ \sigma \} \beta \{ \psi \}}{\{ \varphi \} \alpha ; \beta \{ \psi \}} (4.1)$$

$$\frac{\{ \varphi \land \sigma \} \alpha \{ \psi \}, \{ \neg \varphi \land \sigma \} \beta \{ \psi \}}{\{ \sigma \} \text{ if } \varphi \text{ then } \alpha \text{ else } \beta \{ \psi \}} (4.2)$$

$$\frac{\{ \varphi \land \psi \} \alpha \{ \psi \}}{\{ \psi \} \text{ while } \varphi \text{ do } \alpha \{ \neg \varphi \land \psi \}} (4.3)$$

$$\frac{\varphi' \rightarrow \varphi, \{ \varphi \} \alpha \{ \psi \}, \psi \rightarrow \psi'}{\{ \varphi' \} \alpha \{ \psi' \}} (4.4)$$

These inference rules are derivable in PDL. We need some auxiliary results for the proofs.

**Lemma 4.2** The following inference rules are valid in PDL.

$$\frac{\varphi \rightarrow \psi}{\left[a\right]\varphi \rightarrow \left[a\right]\psi} (4.5)$$

---

$^1$Additionally, a proof system for reasoning about the formulae in the assertions is needed.
Proof: Inference rule 4.5: See the proof of Lemma 2.15.

Inference rule 4.6: So assume $\vdash_{PDL} \psi \rightarrow [\alpha] \psi$.

By the necessitation rule $\vdash_{PDL} [\alpha*](\psi \rightarrow [\alpha] \psi)$.

By the tautology $\chi \rightarrow (\psi \rightarrow (\psi \land \chi))$ and MP we have $\vdash_{PDL} \psi \rightarrow (\psi \land [\alpha*](\psi \rightarrow [\alpha] \psi))$.

Axiom 4.1.5 is logically equivalent to $(\psi \land [\alpha*](\psi \rightarrow [\alpha] \psi)) \rightarrow [\alpha*] \psi$.

By the tautology $((\chi_1 \rightarrow \chi_2) \land (\chi_2 \rightarrow \chi_3)) \rightarrow (\chi_1 \rightarrow \chi_3)$ and the previous two formulae we finally have $\vdash_{PDL} \psi \rightarrow [\alpha*] \psi$. □

Theorem 4.3 The inference rule 4.3 is derivable in PDL.

Proof: So we assume that $\{\varphi \land \psi\} \alpha\{\psi\}$ and have to derive $\{\psi\}$ while $\varphi$ do $\alpha\{\neg \varphi \land \psi\}$. In PDL these assertions are

$$(\varphi \land \psi) \rightarrow [\alpha] \psi$$

and

$$\psi \rightarrow [(\varphi?; \alpha)*; \neg \varphi?] (\neg \varphi \land \psi).$$

So assume that $\vdash_{PDL} (\varphi \land \psi) \rightarrow [\alpha] \psi$.

By tautological equivalence $\vdash_{PDL} \psi \rightarrow (\varphi \rightarrow [\alpha] \psi)$.

With Axiom 4.1.6 we have $\vdash_{PDL} \psi \rightarrow [\varphi?] [\alpha] \psi$.

With Axiom 4.1.2 we have $\vdash_{PDL} \psi \rightarrow [\varphi?; \alpha] \psi$.

With inference rule 4.6 we have $\vdash_{PDL} \psi \rightarrow [(\varphi?; \alpha)*] \psi$.

With inference rule 4.5, MP, and the propositional tautology $\psi \rightarrow (\neg \varphi \rightarrow (\neg \varphi \land \psi))$ we have $\vdash_{PDL} \psi \rightarrow [(\varphi?; \alpha)*] (\neg \varphi \rightarrow (\neg \varphi \land \psi))$.

With Axiom 4.1.6 we have $\vdash_{PDL} \psi \rightarrow [(\varphi?; \alpha)*][\neg \varphi?] (\neg \varphi \land \psi)$.

With Axiom 4.1.2 we have $\vdash_{PDL} \psi \rightarrow [(\varphi?; \alpha)*; \neg \varphi?] (\neg \varphi \land \psi)$. □

Derivation of the remaining three rules is left as an exercise.