Lecture Notes on Dynamic Logic

15-816: Modal Logic
André Platzer

Lecture 19
April 1, 2010

1 Introduction to This Lecture

Dynamic logic is a language for specifying programming languages and gives a proof calculus for verifying programs.

The original work on dynamic logic is by Pratt [Pra76] and by Harel [Har79]. More recent references include [HKT00].

2 Propositional Dynamic Logic

Propositional dynamic logic (PD)L is a multi-modal logic with structured modalities. For each program α, there is a box-modality $[\alpha]$ and a diamond modality $\langle\alpha\rangle$. PDL was developed from first-order dynamic logic by Fischer-Ladner [FL79] and has become popular recently [GW09]. Here we consider regular PDL.

Definition 1 (PDL) *Let Π_0 be a set of atomic programs and let P_0 be a set of propositional letters. The set of formulas $\mathcal{F}$ of (regular) propositional dynamic logic and the set of programs Π are defined by simultaneous induction as:*

1. $true, false \in \mathcal{F}$ *propositional constants*
2. $\Pi_0 \subset \mathcal{F}$ *propositional letters*
3. *If* $F, G \in \mathcal{F}$ *then* $\neg F, (F \vee G), (F \wedge G) \in \mathcal{F}$
4. *If* $F \in \mathcal{F}$, $\alpha \in \Pi$ *then* $[\alpha]F, \langle\alpha\rangle F \in \mathcal{F}$

5. *$\Pi_0 \subset \Pi$ atomic programs*

6. *If $F \in \mathcal{F}$ then $?F \in \Pi$.*

7. *If $\alpha, \beta \in \Pi$ then $(\alpha;\beta), (\alpha \cup \beta), (\alpha^*) \in \Pi$*

Note that the simultaneous induction is formally necessary because logical formulas F can occur inside a test program $?F$ and, vice versa, programs α can occur inside modalities of logical formulas $A \to \langle\alpha\rangle B$.

The effect of state check or test $?F$ is a *skip* (i.e., no change) if formula F is true in the current state and that of *abort*, otherwise. The nondeterministic choice $\alpha \cup \beta$ expresses alternatives in the behavior of the program. Sequential composition $\alpha;\beta$ expresses a behavior in which β starts after α finishes (β never starts if α continues indefinitely). Nondeterministic repetition α^*, repeats α an arbitrary number of times, possibly zero. Other control structures can be defined in terms of the regular operations, for instance:

$$\begin{aligned}\text{if } H \text{ then } \alpha \text{ else } \beta &\equiv (?H;\alpha) \cup (?\neg H;\beta)\\ \text{while } H \text{ do } \alpha &\equiv (?H;\alpha)^*; ?\neg H\\ \text{repeat } \alpha \text{ until } H &\equiv \alpha; (?\neg H;\alpha)^*; ?H\end{aligned}$$

Often times, it is convenient to add if-then-else and while-loops directly as program constructs and investigate proof rules for them.

Atomic programs do not have a specific behavior but can be interpreted by an arbitrary accessibility relation among states.

Definition 2 (PDL-structure) *A* PDL *structure $K = (W, \rho(), \tau)$ is a multimodal Kripke structure with an accessibility relation for each atomic program. That is it consists of*

- *a non-empty set W of states*
- *an interpretation $\rho() : \Pi_0 \to W \times W$ of atomic programs that assigns a transition relation $\rho(\alpha)$ to each atomic program α*
- *an interpretation of propositional letters that assigns to each propositional letter q the set of states $\tau(q) \subseteq W$ at which q is true.*

Definition 3 (Semantics) *The interpretation of PDL relative to a* PDL *structure $K = (W, \rho(), \tau)$ is defined by extending $\rho()$ to Π and extending τ to $\mathcal{F}$ by the following simultaneously inductive definition:*

1. $\tau(\neg F) = W \setminus \tau(F)$

2. $\tau(F \lor G) = \tau(F) \cup \tau(G)$

3. $\tau(F \land G) = \tau(F) \cap \tau(G)$

4. $\tau(\langle\alpha\rangle F) = \{s \in W \ : \ \text{there is a } t \in W \text{ with } s\rho(\alpha)t \text{ and } t \in \tau(F)\}$

5. $\tau([\alpha]F) = \{s \in W \ : \ \text{for all } t \in W \text{ with } s\rho(\alpha)t \text{ we have } t \in \tau(F)\}$

6. $\rho(?F) = \{(s,s) \ : \ s \in \tau(F)\}$

7. $\rho(\alpha \cup \beta) = \rho(\alpha) \cup \rho(\beta)$

8. $\rho(\alpha;\beta) = \{(s,t) \ : \ s\rho(\alpha)z \text{ and } z\rho(\beta)t \text{ for a state } z\}$

9. $\rho(\alpha^*) = \{(s,t) \ : \ \text{there is an } n \in \mathbb{N} \text{ with } n \geq 0 \text{ and there are states } s = s_0, \dots, s_n = t \text{ such that } s_i\rho(\alpha)s_{i+1} \text{ for all } 0 \leq i < n\}$

We say that K, s satisfies *F and write $K, s \models F$ iff $s \in \tau(F)$. We say that F is* valid in *K and write $K \models F$ iff $K, s \models F$ for all $s \in W$. We say that F is* valid *and write $\vDash F$ iff $K \models F$ for all PDL-structures K.*

3 Proving Propositional Dynamic Logic

A Hilbert-style proof calculus for PDL is shown in Fig. 1.

Theorem 4 *The PDL calculus is sound and complete.*

See the literature for a proof, e.g., [HKT00]. Further useful derived rules include:

(D11) $[\alpha](F \land G) \leftrightarrow [\alpha]F \land [\alpha]G$

(D12) $[\alpha \cup \beta]F \leftrightarrow [\alpha]F \land [\beta]F$

(D13) $[?H]F \leftrightarrow (H \rightarrow F)$

(D14) $[\alpha^*]F \leftrightarrow F \land [\alpha][\alpha^*]F$

(D1) all propositional tautologies

(D2) $\langle\alpha\rangle(F \vee G) \leftrightarrow \langle\alpha\rangle F \vee \langle\alpha\rangle G$

(D3) $\langle\alpha \cup \beta\rangle F \leftrightarrow \langle\alpha\rangle F \vee \langle\beta\rangle F$

(D4) $\langle\alpha;\beta\rangle F \leftrightarrow \langle\alpha\rangle\langle\beta\rangle F$

(D5) $\langle ?H\rangle F \leftrightarrow H \wedge F$

(D6) $\langle\alpha^*\rangle F \leftrightarrow F \vee \langle\alpha\rangle\langle\alpha^*\rangle F$

(D7) $[\alpha](F \to G) \to ([\alpha]F \to [\alpha]G)$

(D8) $[\alpha^*](F \to [\alpha]F) \to (F \to [\alpha^*]F)$

(D9) $\dfrac{\phi \quad \phi \to \psi}{\psi}$

(D10) $\dfrac{\phi}{[\alpha]\phi}$

Figure 1: Hilbert calculus for PDL

4 First-Order Dynamic Logic

First-order dynamic logic (DL) extends PDL (although DL had been developed first) to a first-order logic and gives concrete atomic programs with specific effects, as opposed to abstract atomic programs with unknown effects as in PDL.

Definition 5 (DL) *Let V be a set of variables.. The set of formulas $\mathcal{F}$ of dynamic logic and the set of programs Π are defined by simultaneous induction as:*

1. $true, false \in \mathcal{F}$ *propositional constants*
2. *All instances of formulas of first-order logic are in $\mathcal{F}$*
3. *If $F, G \in \mathcal{F}$ then $\neg F, (F \vee G), (F \wedge G) \in \mathcal{F}$*
4. *If $F \in \mathcal{F}$ and $x \in V$ is a variable then $\forall x\, F, \exists x\, F \in \mathcal{F}$*
5. *If $F \in \mathcal{F}$, $\alpha \in \Pi$ then $[\alpha]F, \langle\alpha\rangle F \in \mathcal{F}$*
6. $(x := \theta) \in \Pi$ *are atomic programs for variables $x \in V$ and terms θ*

7. *If* $F \in \mathcal{F}$ *then* $?F \in \Pi$.

8. *If* $\alpha, \beta \in \Pi$ *then* $(\alpha;\beta), (\alpha \cup \beta), (\alpha^*) \in \Pi$

The semantics of DL is extended from that of PDL in the obvious way where the set of states is chosen to be $W := D^V$, i.e., the set of assignments $s : V \to D$ of elements of the domain D (of the first-order structure) to the variables V. Predicate symbols, function symbols, and terms are interpreted as usual in first-order logic. Because there are no other atomic programs, we only need to specify the accessibility relation belonging to an assignment $x := \theta$:

$$\rho(x := \theta) = \{(s,t) \ : \ t(x) = [\![\theta]\!]_s, \text{ and } t(z) = s(z) \text{ for all } z \neq x\}$$

DL can inherit the axioms of first-order logic and of PDL. One typical axiom that DL needs in addition is an axiom that relates assignment to substitution:

$$\text{(D15)} \quad \langle x := \theta \rangle F \leftrightarrow F_x^\theta$$

It says that formula F is true after assigning θ to x if and only if F is true after substituting the new value θ for x.

DL cannot be decidable because it includes first-order logic, where validity is only semidecidable. But DL does not have a sound and complete effective calculus. Note here that DL formulas can state the halting problem for Turing machines. Nevertheless, there are proofs showing that DL has a relatively complete or arithmetically complete proof calculus [HKT00].

5 Example

Consider the following program that computes the square of x:

```
s:=0;
i:=0;
while (i<x) {
  s := s+2*i+1;
  i := i+1;
}
```

Let β be the program above. Then we consider the DL formula $[\beta]s = x * x$ saying that β always computes the square of x in s.

For proving this DL formula we use a sequent calculus for DL. The sequent calculus in Fig. 2 consists of the axioms that we have seen already

$$\text{(R1)}\quad \frac{\Gamma \vdash [\alpha][\beta]\phi, \Delta}{\Gamma \vdash [\alpha; \beta]\phi, \Delta} \qquad \text{(R5)}\quad \frac{\Gamma \vdash [\alpha]\phi \wedge [\beta]\phi, \Delta}{\Gamma \vdash [\alpha \cup \beta]\phi, \Delta}$$

$$\text{(R2)}\quad \frac{\Gamma \vdash \langle\alpha\rangle\langle\beta\rangle\phi, \Delta}{\Gamma \vdash \langle\alpha; \beta\rangle\phi, \Delta} \qquad \text{(R6)}\quad \frac{\Gamma \vdash \langle\alpha\rangle\phi \vee \langle\beta\rangle\phi, \Delta}{\Gamma \vdash \langle\alpha \cup \beta\rangle\phi, \Delta}$$

$$\text{(R3)}\quad \frac{\Gamma \vdash H \to \psi, \Delta}{\Gamma \vdash [?H]\psi, \Delta} \qquad \text{(R7)}\quad \frac{\Gamma, \hat{x} = \theta \vdash \phi_x^{\hat{x}}, \Delta}{\Gamma \vdash [x := \theta]\phi, \Delta}$$

$$\text{(R4)}\quad \frac{\Gamma \vdash H \wedge \psi, \Delta}{\Gamma \vdash \langle ?H\rangle\psi, \Delta} \qquad \text{(R8)}\quad \frac{\Gamma, \hat{x} = \theta \vdash \phi_x^{\hat{x}}, \Delta}{\Gamma \vdash \langle x := \theta\rangle\phi, \Delta}$$

$$\text{(R9)}\quad \frac{\Gamma \vdash I, \Delta \quad I, H \vdash [\alpha]I \quad I, \neg H \vdash \phi}{\Gamma \vdash [\mathtt{while}(H)\,\alpha]\phi, \Delta}$$

Figure 2: A part of a sequent calculus for DL

oriented into the direction that turns properties of complex programs into properties of simpler programs. It also includes an assignment axiom that takes care of renaming variables appropriately. Most importantly, the calculus includes an induction / invariant rule for while-loops as a replacement for the D8 rule of the Hilbert calculus. In the left most branch, rule R9 proves that the induction invariant I is true in the beginning. On the middle branch, rule R9 proves that the invariant is true again after executing the loop body α once, if only I was true before executing the loop body and the loop test H was true (otherwise the loop doesn't execute). On the right branch, rule R9 proves that the invariant I together with the knowledge that the loop test H must have failed for the loop to terminate at all imply the original postcondition ϕ.

Using these proof rules, we know prove the property $[\beta]s = x * x$ using the following abbreviations and the following choice for the invariant I:

$$I \equiv i \leq x \wedge s = i * i$$
$$\alpha \equiv s := s + 2 * i + 1; i := i + 1;$$

For notational convenience, the following proof uses one optimization where the R7 step for $s := 0$ keeps using s instead of a new name $\hat{s}$, because the context Γ, Δ is empty and θ does not mention s.

$$\begin{array}{c}
s=0,i=0\vdash I \qquad I,i<x\vdash[\alpha]I \qquad I,\neg(i<x)\vdash s=x*x \\ \hline
s=0,i=0 \;\vdash\; [\mathit{while}(i<x)\alpha]s=x*x \\ \hline
s=0 \;\vdash\; [i:=0][\mathit{while}(i<x)\alpha]s=x*x \\ \hline
s=0 \;\vdash\; [i:=0;\mathit{while}(i<x)\alpha]s=x*x \\ \hline
\vdash\; [s:=0][i:=0;\mathit{while}(i<x)\alpha]s=x*x \\ \hline
\vdash\; [s:=0;i:=0;\mathit{while}(i<x)\alpha]s=x*x
\end{array}$$

The middle branch can be proved:

$$\begin{array}{r@{\;\vdash\;}l}
 & \mathit{true} \\ \hline
i\le x,s=i*i,i<x,r=s+2*i+1,j=i+1 & i+1\le x\wedge s+2*i+1=s+2*i+1 \\ \hline
i\le x,s=i*i,i<x,r=s+2*i+1,j=i+1 & i+1\le x\wedge s+2*i+1=i*i+2*i+1 \\ \hline
i\le x,s=i*i,i<x,r=s+2*i+1,j=i+1 & i+1\le x\wedge s+2*i+1=(i+1)*(i+1) \\ \hline
i\le x,s=i*i,i<x,r=s+2*i+1,j=i+1 & j\le x\wedge r=j*j \\ \hline
i\le x,s=i*i,i<x,r=s+2*i+1 & [i:=i+1](i\le x\wedge r=i*i) \\ \hline
i\le x,s=i*i,i<x & [s:=s+2*i+1][i:=i+1](i\le x\wedge s=i*i) \\ \hline
i\le x\wedge s=i*i,i<x & [s:=s+2*i+1;i:=i+1](i\le x\wedge s=i*i)
\end{array}$$

The right branch can be proved:

$$\begin{array}{r@{\;\vdash\;}l}
 & \mathit{true} \\ \hline
s=i*i,i=x & s=i*i \\ \hline
s=i*i,i=x & s=x*x \\ \hline
i\le x,s=i*i,\neg(i<x) & s=x*x \\ \hline
i\le x\wedge s=i*i,\neg(i<x) & s=x*x \\ \hline
I\wedge\neg(i<x) & s=x*x
\end{array}$$

Now we try to prove the left branch:

$$\begin{array}{r@{\;\vdash\;}l}
s=0,i=0 & 0\le x \\ \hline
s=0,i=0 & 0\le x\wedge 0=0*0 \\ \hline
s=0,i=0 & i\le x\wedge s=i*i
\end{array}$$

Here we find that we cannot prove the property because we do not know if $0\le x$ is true in the beginning, which is good news because the property is not even valid for negative x.

References

[FL79] Michael J. Fischer and Richard E. Ladner. Propositional dynamic logic of regular programs. *J. Comput. Syst. Sci.*, 18(2):194–211, 1979.

[GW09] Rajeev Goré and Florian Widmann. An optimal on-the-fly tableau-based decision procedure for pdl-satisfiability. In Renate A. Schmidt, editor, *CADE*, volume 5663 of *Lecture Notes in Computer Science*, pages 437–452. Springer, 2009.

[Har79] David Harel. *First-Order Dynamic Logic*. Springer, New York, 1979.

[HKT00] David Harel, Dexter Kozen, and Jerzy Tiuryn. *Dynamic logic*. MIT Press, Cambridge, 2000.

[Pra76] Vaughan R. Pratt. Semantical considerations on Floyd-Hoare logic. In *FOCS*, pages 109–121, 1976.