Formal Specification and Verification

Propositional Dynamic Logic 1

21.01.2019, 22.01.2019 and 28.01.2019

Viorica Sofronie-Stokkermans

e-mail: sofronie@uni-koblenz.de

Overview

- **Model checking:**

  Finite transition systems / CTL properties

  States are "entities" (no precise description, except for labelling functions)

  No precise description of actions (only → important)

Overview

- **Model checking:**

  Finite transition systems / CTL properties

  States are "entities" (no precise description, except for labelling functions)

  No precise description of actions (only → important)

Extensions in two possible directions:

- More precise description of the actions/events
  - Propositional Dynamic Logic
  - Hoare logic
- More precise description of states (and possibly also of actions)
  - succinct representation: formulae represent a set of states
  - deductive verification

Motivation

Example Program: Square

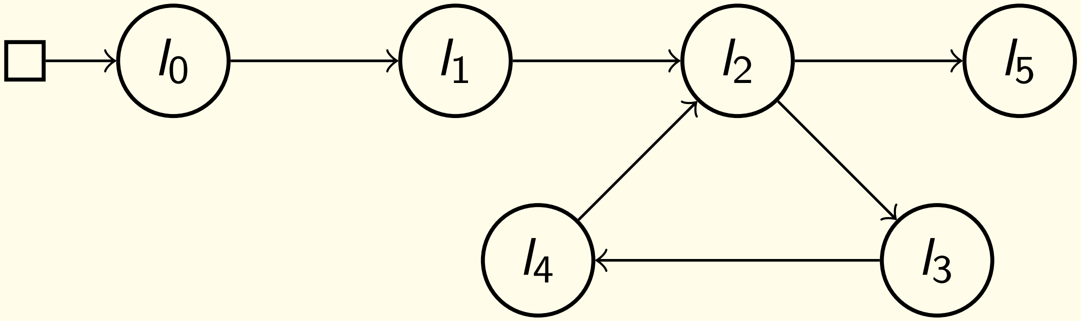
```
I := 0;
Y := 0;
while I < X do
  Y := Y+2*I+1;
  I := I+1
od
```

We would like to prove something like "$A\Diamond(\text{terminated} \wedge \texttt{Y=X*X})$".

Motivation

Example Program: Square

```
I := 0;
Y := 0;
while I < X do
  Y := Y+2*I+1;
  I := I+1
od
```

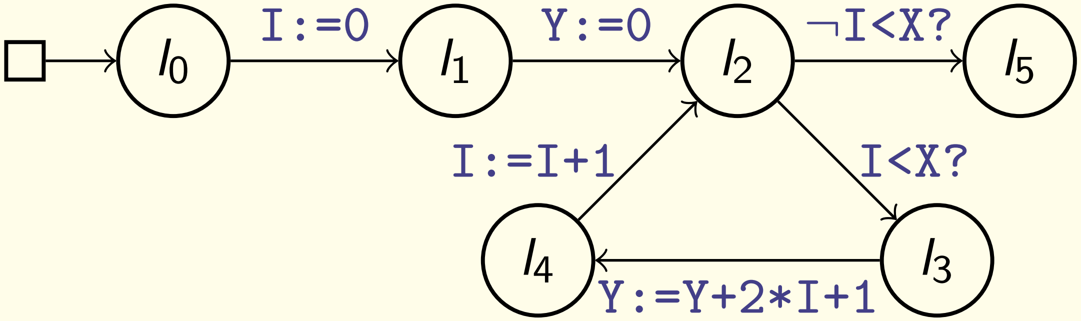


We would like to prove something like “$A\Diamond(\text{terminated} \wedge \texttt{Y=X*X})$”.

Motivation

Example Program: Square

```
I := 0;
Y := 0;
while I < X do
  Y := Y+2*I+1;
  I := I+1
od
```



We would like to prove something like “$A\Diamond$(terminated $\wedge$ `Y=X*X`)”.

CTL* too weak: Transitions carry meaning.

Dynamic Logic: [prog]`Y=X*X`

Motivation

A Simple Programming Language

Logical basis

Typed first-order predicate logic
(Types, variables, terms, formulas, . . .)

Assumption for examples

The signature contains a type Nat and appropriate symbols:

- function symbols $0, s, +, *$
  (terms s(0), s(s(0)), . . . written as 1,2, . . .)
- predicate symbols $\doteq, \leq, <, \geq, >$

NOTE: This is a "convenient assumption" not a definition

Motivation

Programs

- Assignments: $X := t$ X: variable, t:term
- Test: if B then a else b fi

  B: quant.-free formula, a, b: programs
- Loop: while B do a od

  B: quantifier-free formula, a: program
- Composition: a; b a, b programs

WHILE is computationally complete

Motivation

WHILE: Examples

Assignment: Compute the square of X and store it in Y

$$Y := X * X$$

Test: If X is positive then add one else subtract one

$$\text{if } X > 0 \text{ then } X := X + 1 \text{ else } X := X - 1 \text{ fi}$$

Motivation

WHILE: Example - Square of a Number

Program with a while loop and composition:

Compute the square of X (the complicated way)

Making use of: $n^2 = 1 + 3 + 5 + \cdots + (2 * n - 1)$

```
I := 0;
Y := 0;
while I < X do
  Y := Y+2*I+1;
  I := I+1
od
```

Motivation

WHILE: Operational Semantics

Given

A (fixed) first-order structure $\mathcal{A}$ interpreting the function and predicate symbols in the signature

State

$s = (\mathcal{A}, \beta)$ where β is a variable assignment (i.e. function interpreting the variables)

Motivation

State update

$s[e/X] = (\mathcal{A}, \beta[X \mapsto e])$

with $\beta[X \mapsto e](Y) = \begin{cases} e & \text{if } Y = X \\ \beta(Y) & \text{otherwise} \end{cases}$

Motivation

Define the relation $R(\alpha)$ as follows (we write $s[\alpha]s'$ instead of $sR(\alpha)s'$):

- $s[X := t]s'$ iff $s' = s[s(t)/X]$
- $s[\text{if } B \text{ then } \alpha \text{ else } \beta \text{ fi}]s'$ iff $(s \models B$ and $s[\alpha]s')$ or $(s \models \neg B$ and $s[\beta]s')$.
- $s[\text{while } B \text{ do } \alpha \text{ od}]s'$ iff there are states $s = s_0, \ldots, s_t = s'$ s.t. $s_i \models B$ for $0 \leq i \leq t-1$ and $s_t \models \neg B$ and $s_0[\alpha]s_1, s_1[\alpha]s_2, \ldots, s_{t-1}[\alpha]s_t$
- $s[\alpha; \beta]s'$ iff there is a state s'' such that $s[\alpha]s''$ and $s''[\beta]s'$

If α is a deterministic program, $[\alpha]$ is a partial function.

Motivation

A Different Approach to WHILE

Programs

- $X := t$ (atomic program)
- $\alpha; \beta$ (sequential composition)
- $\alpha \cup \beta$ (non-deterministic choice)
- α^* (non-deterministic iteration, n times for some $n \geq 0$)
- $F?$ (test)
  remains in initial state if F is true,
  does not terminate if F is false

Motivation

Restriction to deterministic programs

Non-deterministic program constructors may only be used in

if B then α else β fi $\equiv (B?;\alpha) \cup ((\neg B)?;\beta)$

while B do α od $\equiv (B?;\alpha)^*;(\neg B)?$

Motivation

Expressing Program Properties

Logic for expressing properties

Full first-order logic (usually with arithmetic)

Partial correctness assertion (Hoare formula)

$$\{P\}\alpha\{Q\}$$

Meaning:
If α is started in a state satisfying P and terminates, then its final state satisfies Q.

Formally:
$\{P\}\alpha\{Q\}$ is valid iff for all states s, s', if $s \models P$ and $s[\alpha]s'$, then $s' \models Q$.

Examples

$\{X > 0\}X := X + 1\{X > 1\}$

$\{\text{even}(X)\}X := X + 2\{\text{even}(X)\}$
where $\text{even}(X) \equiv \exists Z(X = 2 * Z)$

$\{\mathit{true}\}\alpha_{\text{square}}\{Y = X * X\}$

Examples

$\{X > 0\}X := X + 1\{X > 1\}$

$\{\text{even}(X)\}X := X + 2\{\text{even}(X)\}$
where $\text{even}(X) \equiv \exists Z(X = 2 * Z)$

$\{\mathit{true}\}\alpha_{\text{square}}\{Y = X * X\}$

Verification: Use annotation of programs with “invariants”

Dynamic Logic

The idea of dynamic logic

- Annotated programs use formulas within programs
- Dynamic Logic uses programs within formulas
- Instead of “assert F” after program segment α, write: $[\alpha]F$

$\mapsto$ multi-modal logic

Dynamic Logic

Dynamic logic is a language for specifying programming languages.

The original work on dynamic logic is by Vaughan Pratt (1976) and by David Harel (1979).

Propositional Dynamic Logic

Propositional dynamic logic (PDL) 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 (1979) and has become popular recently.

Here we consider regular PDL.

Propositional Dynamic Logic

Syntax

Prog set of programs

$\text{Prog}_0 \subseteq \text{Prog}$: set of atomic programs

Π: set of propositional variables

The set of formulae $\mathbf{Fma}^{PDL}_{\mathbf{Prog},\Pi}$ of (regular) propositional dynamic logic and the set of programs **Prog** are defined by simultaneous induction as follows:

PDL: Syntax

Formulae:

F, G, H	$::=$	$\bot$	(falsum)
	$\mid$	$\top$	(verum)
	$\mid$	p	$p \in \Pi$ (atomic formula)
	$\mid$	$\neg F$	(negation)
	$\mid$	$(F \wedge G)$	(conjunction)
	$\mid$	$(F \vee G)$	(disjunction)
	$\mid$	$(F \rightarrow G)$	(implication)
	$\mid$	$(F \leftrightarrow G)$	(equivalence)
	$\mid$	$[\alpha]F$	if $\alpha \in$ Prog
	$\mid$	$\langle\alpha\rangle F$	if $\alpha \in$ Prog

Programs:

α, β, γ	$::=$	α_0	$\alpha_0 \in \mathrm{Prog}_0$ (atomic program)
	$\mid$	$F?$	F formula (test)
	$\mid$	$\alpha; \beta$	(sequential composition)
	$\mid$	$\alpha \cup \beta$	(non-deterministic choice)
	$\mid$	α^*	(non-deterministic repetition)

Semantics

A PDL structure $\mathcal{K} = (S, R(), I)$ is a multimodal Kripke structure with an accessibility relation for each atomic program. That is it consists of:

- a non-empty set S of states
- an interpretation $R() : \mathsf{Prog}_0 \to S \times S$ of atomic programs that assigns a transition relation $R(\alpha)$ to each atomic program α
- an interpretation $I : \Pi \times S \to \{0, 1\}$

PDL: Semantics

The interpretation of PDL relative to a PDL structure $\mathcal{K} = (S, R(), I)$ is defined by extending $R()$ to Prog and extending I to $\mathrm{Fma}^{PDL}_{\mathrm{Prog}_0,\Pi}$ by the following simultaneously inductive definition:

Interpretation of formulae/programs

$$
\begin{aligned}
val_{\mathcal{K}}(p, s) &= I(p, s) \qquad \text{if } p \in \Pi \\
val_{\mathcal{K}}(\neg F, s) &= \neg_{\mathsf{Bool}} val_{\mathcal{K}}(F, s) \\
val_{\mathcal{K}}(F \wedge G, s) &= val_{\mathcal{K}}(F, s) \wedge_{\mathsf{Bool}} val_{\mathcal{K}}(G, s) \\
val_{\mathcal{K}}(F \vee G, s) &= val_{\mathcal{K}}(F, s) \vee_{\mathsf{Bool}} val_{\mathcal{K}}(G, s) \\
val_{\mathcal{K}}(F \rightarrow G, s) &= val_{\mathcal{K}}(F, s) \rightarrow_{\mathsf{Bool}} val_{\mathcal{K}}(G, s) \\
val_{\mathcal{K}}(F \leftrightarrow G, s) &= val_{\mathcal{K}}(F, s) \leftrightarrow_{\mathsf{Bool}} val_{\mathcal{K}}(G, s) \\
val_{\mathcal{K}}([\alpha]F, s) = 1 \ \mathit{iff} &\quad \text{for all } t \in S \text{ with } (s, t) \in R(\alpha), val_{\mathcal{K}}(F, t) = 1 \\
val_{\mathcal{K}}(\langle\alpha\rangle\, F, s) = 1 \ \mathit{iff} &\quad \text{for some } t \in S \text{ with } (s, t) \in R(\alpha), val_{\mathcal{K}}(F, t) = 1 \\
R([F?]) &= \{(s, s) \mid val_{\mathcal{K}}(F, s) = 1\} \\
& (F? \text{ means: if } F \text{ then skip else do not terminate}) \\
R(\alpha \cup \beta) &= R(\alpha) \cup R(\beta) \\
R(\alpha; \beta) &= \{(s, t) \mid \text{ there exists } u \in S \ s.t.(s, u) \in R(\alpha) \text{ and } (u, t) \in R(\beta)\} \\
R(\alpha^*) &= R(\alpha)^* \\
&= \{(s, t) \mid \text{ there exist } n \geq 0 \text{ and } u_0, \ldots, u_n \in S \text{ with} \\
&\qquad\qquad s = u_0, t = u_n, (u_0, u_1), \ldots, (u_{n-1}, u_n) \in R(\alpha)\}
\end{aligned}
$$

Interpretation of formulae/programs

- $(\mathcal{K}, s)$ satisfies F (notation $(\mathcal{K}, s) \models F$) iff $val_{\mathcal{K}}(F, s) = 1$.
- F is valid in $\mathcal{K}$ (notation $\mathcal{K} \models F$) iff $(\mathcal{K}, s) \models F$ for all $s \in S$.
- F is valid (notation $\models F$) iff $\mathcal{K} \models F$ for all PDL-structures $\mathcal{K}$.

Hilbert-style axiom system for PDL

Axioms

$(D1)$ All propositional logic tautologies

$(D2)$ $[\alpha](A \to B) \to ([\alpha]A \to [\alpha]B)$

$(D3)$ $[\alpha](A \wedge B) \leftrightarrow [\alpha]A \wedge [\alpha]B$

$(D4)$ $[\alpha; \beta]A \leftrightarrow [\alpha][\beta]A$

$(D5)$ $[\alpha \cup \beta]A \leftrightarrow [\alpha]A \wedge [\beta]A$

$(D6)$ $[A?]B \leftrightarrow (A \to B)$

$(D7)$ $[\alpha^*]A \leftrightarrow A \wedge [\alpha][\alpha^*]A,$

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

Inference rules

$$MP \qquad \frac{F \qquad F \to G}{G}$$

$$Gen \qquad \frac{F}{[\alpha]F}$$

We will show that PDL is determined by PDL structures, and has the finite model property.

Soundness of PDL

Theorem. If the formula F is provable in the inference system for PDL then F is valid in all PDL structures.

Proof: Induction of the length of the proof, using the following facts:
1. The axioms are valid in every PDL structure. Easy computation.

2. If the premises of an inference rule are valid in a structure $\mathcal{K}$, the conclusion is also valid in $\mathcal{K}$.

(MP) If $\mathcal{K} \models F, \mathcal{K} \models F \rightarrow G$ then $\mathcal{K} \models G$ (follows from the fact that for every state s of $\mathcal{K}$ if $(\mathcal{K}, s) \models F, (\mathcal{K}, s) \models F \rightarrow G$ then $(\mathcal{K}, s) \models G$)

(Gen) Assume that $\mathcal{K} \models F$. Then $(\mathcal{K}, s) \models F$ for every state s of $\mathcal{K}$.

Let t be a state of $\mathcal{K}$. $(\mathcal{K}, t) \models [\alpha]F$ if for all t' with $(t, t') \in R(\alpha)$ we have $(\mathcal{K}, t') \models F$. But under the assumption that $\mathcal{K} \models F$ the latter is always the case. This shows that $(\mathcal{K}, t) \models [\alpha]F$ for all t.

Summary

Until now:

- Motivation: WHILE programs and Hoare triples
- Syntax and semantics of PDL
- Soundness of the axiom system

Next:

- Completeness and decidability
- A sequent calculus for PDL

Completeness of PDL

Theorem. If the formula F is is valid in all PDL structures then F is provable in the inference system for PDL.

Proof

Idea:

Assume that F is not provable in the inference system for PDL.

We show that:

(1) $\neg F$ is consistent with the set L of all theorems of PDL

(2) We can construct a “canonical” PDL structure $\mathcal{K}_L$ and a state w in this PDL structure such that $(\mathcal{K}_L, w) \models \neg F$.

Contradiction!

Consistent sets of formulae

Let L be a set of PDL formulae which:

(1) contains all propositional tautologies

(2) contains axiom PDL

(3) is closed under modus ponens and generalization

(4) is closed under instantiation

Definition. A subset $\mathcal{F} \subseteq L$ is called L-inconsistent iff there exist formulae $A_1, \ldots, A_n \in \mathcal{F}$ such that

$$(\neg A_1 \vee \cdots \vee \neg A_n) \in L$$

$\mathcal{F}$ is called L-consistent iff it is not L-inconsistent.

Definition. A consistent set $\mathcal{F}$ of PDL formulae is called maximal L-consistent if for every formula A either $A \in \mathcal{F}$ or $\neg A \in \mathcal{F}$.

Consistent sets of formulae

Let L be as before. In what follows we assume that L is consistent.

Theorem. Let $\mathcal{F}$ be a maximal L-consistent set of formulae. Then:

(1) For every formula A, either $A \in \mathcal{F}$ or $\neg A \in \mathcal{F}$, but not both.

(2) $A \vee B \in \mathcal{F}$ iff $A \in \mathcal{F}$ or $B \in \mathcal{F}$

(3) $A \wedge B \in \mathcal{F}$ iff $A \in \mathcal{F}$ and $B \in \mathcal{F}$

(4) $L \subseteq \mathcal{F}$

(5) $\mathcal{F}$ is closed under Modus Ponens

Proof. (1) $A \in F$ or $\neg A \in F$ by definition.

Assume $A \in F$ and $\neg A \in F$.
We know that $\neg A \vee \neg\neg A \in L$ (propositional tautology), so F is inconsistent.
Contradiction.

Consistent sets of formulae

Let L be as before. In what follows we assume that L is consistent.

Theorem. Let F be a maximal L-consistent set of formulae. Then:

(1) For every formula A, either $A \in F$ or $\neg A \in F$, but not both.

(2) $A \vee B \in F$ iff $A \in F$ or $B \in F$

(3) $A \wedge B \in F$ iff $A \in F$ and $B \in F$

(4) $L \subseteq F$

(5) F is closed under Modus Ponens

Proof. (2) "$\Rightarrow$" Assume $A \vee B \in F$, but $A, B \notin F$. Then $\neg A, \neg B \in F$. As $\neg\neg A \vee \neg\neg B \vee \neg(A \vee B) \in L$ (classical tautology) it follows that F is inconsistent.

(2) "$\Leftarrow$" Assume $A \in F$ and $A \vee B \notin F$. Then $\neg(A \vee B) \in F$. Then $\neg A \vee (A \vee B) \in L$, so F is inconsistent.

Consistent sets of formulae

Let L be as before. In what follows we assume that L is consistent.

Theorem. Let F be a maximal L-consistent set of formulae. Then:

(1) For every formula A, either $A \in F$ or $\neg A \in F$, but not both.

(2) $A \vee B \in F$ iff $A \in F$ or $B \in F$

(3) $A \wedge B \in F$ iff $A \in F$ and $B \in F$

(4) $L \subseteq F$

(5) F is closed under Modus Ponens

Proof. (3) Analogous to (2)

Consistent sets of formulae

Let L be as before. In what follows we assume that L is consistent.

Theorem. Let F be a maximal L-consistent set of formulae. Then:

(1) For every formula A, either $A \in F$ or $\neg A \in F$, but not both.

(2) $A \vee B \in F$ iff $A \in F$ or $B \in F$

(3) $A \wedge B \in F$ iff $A \in F$ and $B \in F$

(4) $L \subseteq F$

(5) F is closed under Modus Ponens

Proof. (4) If $A \in L$ then $\neg A$ is inconsistent. Hence, $\neg A \not\in F$, so $A \in F$.

(5) Assume $A \in F, A \to B \in F$ and $B \not\in F$. Then $\neg A \vee \neg(A \to B) \vee B$ is a tautology, hence in L. Thus, F inconsistent.

Consistent sets of formulae

Theorem. Every consistent set F of formulae is contained in a maximally consistent set of formulae.

Proof. We enumerate all modal formulae: $A_0, A_1, \dots$ and inductively define an ascending chain of sets of formulae:

$$F_0 := F$$

$$F_{n+1} := \begin{cases} F_n \cup \{A_n\} & \text{if this set is consistent} \\ F_n \cup \{\neg A_n\} & \text{otherwise} \end{cases}$$

It can be proved by induction that F_n is consistent for all n.

Let $F_{\max} = \bigcup_{n \in \mathbb{N}} F_n$.
Then $F_{\max}$ is maximal consistent and contains F.

Consistent sets of formulae

Lemma. If F is not provable in PDL then $\neg F$ is consistent with the set L of all theorems of PDL, so it is contained in a maximally conststent set of formulae $W_{\neg F}$.

Canonical models

Goal: Assume F is not a theorem. Construct a PDL structure $\mathcal{K}$ and a state w of $\mathcal{K}$ such that $(\mathcal{K}, w) \models \neg F$.

States:

State of $\mathcal{K}$: maximal consistent set of formulae.

Intuition: $(\mathcal{K}, W) \models \phi$ iff $\phi \in W$.

Then: $(\mathcal{K}, W_{\neg F}) \models \neg F$

Accessibility relation:

Intuition:
$(\mathcal{K}, W) \models [\alpha]F$ iff for all $W', ((W, W') \in R(\alpha) \rightarrow (\mathcal{K}, W') \models F)$

Canonical models

Goal: Assume F is not a PDL theorem. Construct a PDL structure $\mathcal{K}$ and a state w of $\mathcal{K}$ such that $(\mathcal{K}, w) \models \neg F$.

States:

State of $\mathcal{K}$: maximal consistent set of formulae.

Intuition: $(\mathcal{K}, W) \models \phi$ iff $\phi \in W$.

Then: $(\mathcal{K}, W_{\neg F}) \models \neg F$

Accessibility relation:

Intuition:
$(\mathcal{K}, W) \models [\alpha]F$ iff for all $W', ((W, W') \in R(\alpha) \rightarrow (\mathcal{K}, W') \models F$
$[\alpha]F \in W$ iff for all $W', ((W, W') \in R(\alpha) \rightarrow F \in W')$

$(W, W') \in R(\alpha)$ iff $W' \supseteq \{F \mid [\alpha]F \in W\}$

Canonical models

Theorem. $\mathcal{K}$ satisfies all PDL structure conditions except $R(\alpha^*) \subseteq (R(\alpha))^*$.

Proof: By direct checking.

Example: $R(\alpha;\beta) \subseteq R(\alpha) \circ R(\beta)$
Assume $(W, W') \in R(\alpha;\beta)$. Then $\{F \mid [\alpha;\beta]F \in W\} \subseteq W'$.

We want to show that there exists W_0 with $(W, W_0) \in R(\alpha)$ and $(W_0, W') \in R(\beta)$.

- $(W, W_0) \in R(\alpha)$ iff $\{A \mid [\alpha]A \in W\} \subseteq W_0$
- $(W_0, W') \in R(\beta)$ iff $\{B \mid [\beta]B \in W_0\} \subseteq W'$ iff $\{\neg[\beta]D \mid D \not\in W'\} \subseteq W_0$.

It is sufficient to show that $W_0 = \{B \mid [\alpha]B \in W\} \cup \{\neg[\beta]D \mid D \not\in W'\}$ is PDL-consistent.

For this, the PDL-theorem $[\alpha;\beta]A \leftrightarrow [\alpha][\beta]A$ is used.

Canonical models

Proof: (ctd.) We show that $\{A \mid [\alpha]A \in W\} \cup \{\neg[\beta]B \mid B \not\in W'\}$ is PDL-consistent.

Assume that the set is not PDL consistent. Then there is a theorem

$$\vdash A_1 \wedge \cdots \wedge A_m \wedge \neg[\beta]B_1 \wedge \ldots \neg[\beta]B_n \rightarrow \perp$$

where $[\alpha]A_i \in W$ and $B_j \not\in W'$. Let $B = B_1 \vee \cdots \vee B_n$.

Since $\vdash [\beta]B_1 \vee \cdots \vee [\beta]B_n \rightarrow [\beta]B$ it follows that $\vdash A_1 \wedge \cdots \wedge A_m \rightarrow [\beta]B$ hence:

$$\vdash [\alpha](A_1 \wedge \cdots \wedge A_m) \rightarrow [\alpha][\beta]B$$

and since $\vdash [\alpha]A_1 \wedge \cdots \wedge [\alpha]A_m \rightarrow [\alpha](A_1 \wedge \cdots \wedge A_m)$ we showed that

$$\vdash [\alpha]A_1 \wedge \cdots \wedge [\alpha]A_m \rightarrow [\alpha][\beta]B$$

Using the PDL-theorem $[\alpha][\beta]B \rightarrow [\alpha;\beta]B$ it then follows that

$$\vdash [\alpha]A_1 \wedge \cdots \wedge [\alpha]A_m \rightarrow [\alpha;\beta]B$$

Since $[\alpha]A_i \in W$ and W is maximally consistent it follows that $[\alpha;\beta]B \in W$, hence $B = B_1 \vee \cdots \vee B_n \in W'$. But then (as W' maximally consistent) $B_j \in W'$ for some j which is a contradiction.

Canonical models

Theorem. Assume F is not a PDL theorem. We can construct a PDL structure $\mathcal{K}'$ and a state w of $\mathcal{K}'$ such that $(\mathcal{K}', w) \models \neg F$.

Proof. To obtain a PDL structure that falsifies F we will collapse $\mathcal{K}$ by a suitable Γ that contains F. The closure rules for Γ that will be needed are:

- Γ is closed under subformulae;
- $[B?]D \in \Gamma$ implies $B \in \Gamma$;
- $[\alpha; \beta]B \in \Gamma$ implies $[\alpha][\beta]B \in \Gamma$;
- $[\alpha \cup \beta]B \in \Gamma$ implies $[\alpha]B, [\beta]B \in \Gamma$;
- $[\alpha^*]B \in \Gamma$ implies $[\alpha][\alpha^*]B \in \Gamma$

A set Γ satisfying these conditions will be called closed.

Completeness/Decidability of PDL

Theorem. If Γ is the smallest closed set containing a given formula F, then Γ is finite.

Proof. The point is to show that closing Subformulae(F) under the above rules produces only finitely many new formulae.

Define a formula to be boxed if it is prefixed by a modal connective, i.e. is of the form $[\alpha]B$ for some α and B. Each time we apply a closure rule, new boxed formulae appear on the right side of the rule, and further rules may apply to these new formulae.

But observe that the programs α indexing prefixes $[\alpha]$ on the right side are in all cases shorter in length than those indexing the prefix on the left of the rule in question. Hence we will eventually produce only atomic prefixes on the right, and run out of rules to apply.

Completeness/Decidability of PDL

Having determined that Γ, the smallest closed set containing F, is finite, we identify the states which satisfy the same formulae in Γ:

Completeness/Decidability of PDL

Fix a model $\mathcal{K} = (S, R, I)$ and a set $\Gamma \subseteq Fma_\Sigma$ that is closed under subformulae, i.e. $B \in \Gamma$ implies Subformulae$(B) \subseteq \Gamma$.

For each $s \in S$, define

$$\Gamma_s = \{B \in \Gamma \mid (\mathcal{K}, s) \models B\}$$

and put $s \sim_\Gamma t$ iff $\Gamma_s = \Gamma_t$,

Then $s \sim_\Gamma t$ iff for all $B \in \Gamma$, $(\mathcal{K}, s) \models B$ iff $(\mathcal{K}, t) \models B$.

Fact: $\sim_\Gamma$ is an equivalence relation on S.

Let $[s] = \{t \mid s \sim_\Gamma t\}$ be the $\sim_\Gamma$-equivalence class of s.

Let $S_\Gamma := \{[s] \mid s \in S\}$ be the set of all such equivalence classes.

Decidability/Completeness

Goal: $(\mathcal{K}, s) \models A \quad \mapsto \quad (\mathcal{K}', s') \models A$, $\mathcal{K}' = (S', R', I')$.

Step 1: $S' := S_\Gamma$, where $\Gamma = \text{Subformulae}(S)$

Step 2: $I' : (\Pi \cap \Gamma) \times S' \to \{0, 1\}$ def. by $I'(P, [s]) = I(P, s)$

Step 3: $R'(\alpha)$ def. e.g. by: $([s], [t]) \in R'(\alpha)$ iff $\exists s' \in [s], \exists t' \in [t]$: $(s', t') \in R(\alpha)$

Theorem: $\mathcal{K}'$ is a PDL structure (a filtration of $\mathcal{K}$).

Since $(\mathcal{K}, W_{\neg F}) \models \neg F$ it can easily be seen that $(\mathcal{K}', [W_{\neg F}]) \models \neg F$.
↦ completeness.

Lemma. If Γ is finite, then S_Γ is finite and has at most 2^n elements, where n is the number of elements of Γ.
↦ decidability

Conclusions

PDL is decidable (it has the finite model propety).

Proof calculi for PDL exist (e.g. sequent calculi, tableau calculi)

For really reasoning about programs, often *first order dynamic logic* is needed (undecidable)

Nevertheless, many systems used for verification use sequent or tableau calculi also for first order dynamic logic.

Sequent calculi

In what follows we illustrate a way of designing sequent calculi for propositional dynamic logic.

We do not give here any completeness results; for a sound and complete sequent calculus we refer e.g. to:

- Vaughan R. Pratt: A Practical Decision Method for Propositional Dynamic Logic: Preliminary Report STOC 1978: 326-337

  `http://dl.acm.org/citation.cfm?doid=800133.804362`

For a sound and complete tableau calculus we refer e.g. to:

- Rajeev Goré, Florian Widmann: An Optimal On-the-Fly Tableau-Based Decision Procedure for PDL-Satisfiability. CADE 2009: 437-452

A sequent calculus for PDL

Reminder (Classical propositional logic)

Sequent Calculus based on notion of sequent

$$\underbrace{\psi_1, \ldots, \psi_m}_{\text{Antecedent}} \Rightarrow \underbrace{\phi_1, \ldots, \phi_n}_{\text{Succedent}}$$

Has same semantics as

$$\models \psi_1 \wedge \cdots \wedge \psi_m \to (\phi_1 \vee \cdots \vee \phi_n)$$

$$\{\psi_1, \ldots, \psi_m\} \models \phi_1 \vee \cdots \vee \phi_n$$

Notation for Sequents

$$\underbrace{\psi_1, \ldots, \psi_m}_{\text{Antecedent}} \Rightarrow \underbrace{\phi_1, \ldots, \phi_n}_{\text{Succedent}}$$

Consider antecedent/succedent as sets of formulas, may be empty

Schema Variables:

$\phi, \psi, \ldots$ match formulas, $\Gamma, \Delta, \ldots$ match sets of formulas

Characterize infinitely many sequents with a single schematic sequent:

Example: $\Gamma \Rightarrow \Delta, \phi \wedge \psi$

Matches any sequent with occurrence of conjunction in succedent

We call $\phi \wedge \psi$ main formula and Γ, Δ side formulae of sequent.

Sequent Calculus Rules of Propositional Logic

Write syntactic transformation schema for sequents that reflects semantics of connectives as closely as possible

$$\text{Rule Name}\quad \frac{\overbrace{\Gamma_1 \Rightarrow \Delta_1 \ldots\ \Gamma_n \Rightarrow \Delta_n}^{\text{premises}}}{\underbrace{\Gamma \Rightarrow \Delta}_{\text{conclusion}}}\ .$$

Example:

$$\text{andRight}\quad \frac{\Gamma \Rightarrow \phi, \Delta \quad \Gamma \Rightarrow \psi, \Delta}{\Gamma \Rightarrow \phi \land \psi, \Delta}\ .$$

Informal meaning:

In order to prove that Γ entails $(\phi \land \psi) \lor \Delta$ we need to prove that:

Γ entails $\phi \lor \Delta$ and

Γ entails $\psi \lor \Delta$

Sequent Calculus Rules of Propositional Logic

Write syntactic transformation schema for sequents that reflects semantics of connectives as closely as possible

$$\text{Rule Name}\quad \frac{\overbrace{\Gamma_1 \Rightarrow \Delta_1 \ldots \ \Gamma_n \Rightarrow \Delta_n}^{\text{premises}}}{\underbrace{\Gamma \Rightarrow \Delta}_{\text{conclusion}}}\ .$$

Example:

$$\text{andRight}\quad \frac{\Gamma \Rightarrow \phi, \Delta \quad \Gamma \Rightarrow \psi, \Delta}{\Gamma \Rightarrow \phi \wedge \psi, \Delta}\ .$$

Sound rule (essential): If $\models (\Gamma_1 \to \Delta_1)$ and ... and $\models (\Gamma_n \to \Delta_n)$ then $\models (\Gamma \to \Delta)$

Complete rule (desirable): If $\models (\Gamma \to \Delta)$ then $\models (\Gamma_1 \to \Delta_1), \ldots \models (\Gamma_n \to \Delta_n)$

Rules of Propositional Sequent Calculus

main	left side (antecedent)	right side (succedent)
not	$\dfrac{\Gamma \Rightarrow \phi, \Delta}{\Gamma, \neg\phi \Rightarrow \Delta}$	$\dfrac{\Gamma, \phi \Rightarrow \Delta}{\Gamma \Rightarrow \neg\phi, \Delta}$
and	$\dfrac{\Gamma, \phi, \psi \Rightarrow \Delta}{\Gamma, \phi \wedge \psi \Rightarrow \Delta}$	$\dfrac{\Gamma \Rightarrow \phi, \Delta \quad \Gamma \Rightarrow \psi, \Delta}{\Gamma \Rightarrow \phi \wedge \psi, \Delta}$
or	$\dfrac{\Gamma, \phi \Rightarrow \Delta \quad \Gamma, \psi \Rightarrow \Delta}{\Gamma, \phi \vee \psi \Rightarrow \Delta}$	$\dfrac{\Gamma \Rightarrow \phi, \psi, \Delta}{\Gamma \Rightarrow \phi \vee \psi, \Delta}$
imp	$\dfrac{\Gamma \Rightarrow \phi, \Delta \quad \Gamma, \psi \Rightarrow \Delta}{\Gamma, \phi \rightarrow \psi \Rightarrow \Delta}$	$\dfrac{\Gamma, \phi \Rightarrow \psi, \Delta}{\Gamma \Rightarrow \phi \rightarrow \psi, \Delta}$

close $\dfrac{}{\Gamma, \phi \Rightarrow \phi, \Delta}$ true $\dfrac{}{\Gamma \Rightarrow \text{true}, \Delta}$ false $\dfrac{}{\Gamma, \text{false} \Rightarrow \Delta}$

Example: Part of a sequent calculus for PDL

In addition to the classical propositional rules we can consider:

main	left side (antecedent)	right side (succedent)
$[\alpha]$	$\frac{\Gamma,[\alpha]\phi,[\alpha]\psi\Rightarrow\Delta}{\Gamma,[\alpha](\phi\wedge\psi)\Rightarrow\Delta}$	$\frac{\Gamma\Rightarrow\Delta,[\alpha]\phi \quad \Gamma\Rightarrow\Delta,[\alpha]\psi}{\Gamma\Rightarrow\Delta,[\alpha](\phi\wedge\psi)}$
$<\alpha>$	$\frac{\Gamma,<\alpha>\phi\Rightarrow\Delta \quad \Gamma,<\alpha>\psi\Rightarrow\Delta}{\Gamma,<\alpha>(\phi\vee\psi)\Rightarrow\Delta}$	$\frac{\Gamma\Rightarrow\Delta,<\alpha>\phi,<\alpha>\psi}{\Gamma\Rightarrow\Delta,<\alpha>(\phi\vee\psi)}$
$[\alpha^*]$	$\frac{\Gamma,[\alpha][\alpha^*]\phi,\phi\Rightarrow\Delta}{\Gamma,[\alpha^*]\phi\Rightarrow\Delta}$	$\frac{\Gamma\Rightarrow\phi,\Delta \quad \Gamma\Rightarrow[\alpha][\alpha^*]\phi,\Delta}{\Gamma\Rightarrow\Delta,[\alpha^*]\phi}$
$[\phi?]$	$\frac{\Gamma\Rightarrow\phi,\Delta \quad \Gamma,\psi\Rightarrow\Delta}{\Gamma,[\phi?]\psi\Rightarrow\Delta}$	$\frac{\Gamma,\phi\Rightarrow\psi,\Delta}{\Gamma\Rightarrow[\phi?]\psi,\Delta}$
$< \phi? >$	$\frac{\Gamma,\phi,\psi\Rightarrow\Delta}{\Gamma,<\phi?>\psi\Rightarrow\Delta}$	$\frac{\Gamma,\Rightarrow\phi,\Delta \quad \Gamma,\Rightarrow\psi,\Delta}{\Gamma\Rightarrow<\phi?>\psi,\Delta}$
$\alpha\cup\beta$	$\frac{\Gamma,[\alpha]\phi,[\beta]\phi\Rightarrow\Delta}{\Gamma,[\alpha\cup\beta]\phi\Rightarrow\Delta}$	$\frac{\Gamma\Rightarrow[\alpha]\phi,\Delta \quad \Gamma\Rightarrow[\beta]\phi,\Delta}{\Gamma\Rightarrow[\alpha\cup\beta]\phi,\Delta}$
$\alpha;\beta$	$\frac{\Gamma,[\alpha][\beta]\phi\Rightarrow\Delta}{\Gamma,[\alpha;\beta]\phi\Rightarrow\Delta}$	$\frac{\Gamma\Rightarrow[\alpha][\beta]\phi,\Delta}{\Gamma\Rightarrow[\alpha;\beta]\phi,\Delta}$

Example: Part of a sequent calculus for PDL

We also use:

$$\frac{\Gamma \Rightarrow [\alpha](\phi \to \psi), \Delta}{\Gamma \Rightarrow ([\alpha]\phi \to [\alpha]\psi), \Delta}$$

$$\frac{\Gamma, [\alpha]\neg\phi \Rightarrow \Delta}{\Gamma \Rightarrow <\alpha>\phi, \Delta} \qquad \frac{\Gamma \Rightarrow [\alpha]\neg\phi, \Delta}{\Gamma, <\alpha>\phi \Rightarrow \Delta}$$

$$\frac{\Gamma, [\alpha][\beta]\neg\phi \Rightarrow \Delta}{\Gamma, [\alpha]\neg<\beta>\phi \Rightarrow \Delta} \qquad \frac{\Gamma \Rightarrow [\alpha][\beta]\neg\phi, \Delta}{\Gamma \Rightarrow [\alpha]\neg<\beta>\phi, \Delta}$$

Example

Prove $<\alpha^*>\phi \rightarrow \phi \vee <\alpha><\alpha^*>\phi$ using the sequent calculus.

Example

Prove $<\alpha^*>\phi \rightarrow \phi \vee <\alpha><\alpha^*>\phi$ using the sequent calculus.

$$
\begin{array}{ccl}
\text{close} & & \\
\cline{1-2}
[\alpha][\alpha^*]\neg\phi, \phi \quad \Rightarrow \quad \phi & \text{close} & (\neg, \text{right}) \\
\cline{1-2}
[\alpha][\alpha^*]\neg\phi \quad \Rightarrow \quad \neg\phi, \phi & [\alpha][\alpha^*]\neg\phi \quad \Rightarrow \quad [\alpha][\alpha^*]\neg\phi, \phi & ([\alpha^*], \text{right}) \\
\cline{1-2}
\multicolumn{2}{c}{[\alpha][\alpha^*]\neg\phi \quad \Rightarrow \quad [\alpha^*]\neg\phi, \phi} & (\text{not}<\alpha>) \\
\cline{1-2}
\multicolumn{2}{c}{[\alpha]\neg<\alpha^*>\phi \quad \Rightarrow \quad [\alpha^*]\neg\phi, \phi} & (<\alpha>) \\
\cline{1-2}
\multicolumn{2}{c}{<\alpha^*>\phi, [\alpha]\neg<\alpha^*>\phi \quad \Rightarrow \quad \phi} & (<\alpha>) \\
\cline{1-2}
\multicolumn{2}{c}{<\alpha^*>\phi \Rightarrow \phi, <\alpha><\alpha^*>\phi} & (\text{or}, \text{right}) \\
\cline{1-2}
\multicolumn{2}{c}{<\alpha^*>\phi \Rightarrow \phi \vee <\alpha><\alpha^*>\phi} & (\text{imp}, \text{right}) \\
\cline{1-2}
\multicolumn{2}{c}{\Rightarrow \quad <\alpha^*>\phi \rightarrow \phi \vee <\alpha><\alpha^*>\phi} &
\end{array}
$$

Summary

Dynamic logic

- Syntax and semantics
- Axiom system
- Soundness and completeness
- Sequent calculus