Karlsruhe Institute of Technology

Formale Systeme II: Theorie

Dynamic Logic: Propositional Dynamic Logic

SS 2016

Prof. Dr. Bernhard Beckert · Dr. Mattias Ulbrich
Slides partially by Prof. Dr. Peter H. Schmitt

Goals

Overview – a family of logics

$$\begin{array}{c} \text{Modal Logics} \\ \downarrow \\ \text{Propositional Dynamic Logic} \\ \downarrow \\ \text{Dynamic Logic} \\ \swarrow \quad \searrow \\ \text{Hybrid DL} \qquad \text{Java DL} \end{array}$$

Modal Logics: → Formal Systems I (recap here)
Java DL: Logic used in KeY
→ lecture "Formal Systems II – Applications"

Goals

Karlsruhe Institute of Technology

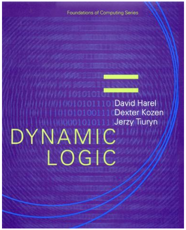
Dynamic Logic as ...

- abstract reasoning framework for descriptions of actions
- means to formalise and reason about semantics of programs
- vehicle for examining/proving theoretical results on program reasoning
  - what is decidable, what is not?
  - relative completeness
- concept of program verification on a while language
- logic for a verification engine for a realworld programming language

Literature

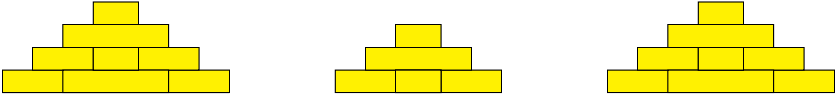
- *Formale Systeme II*
  Vorlesungsskript
  Peter H. Schmitt
  $\rightarrow$ Website

- *Dynamic Logic*
  Series: Foundations of Computing
  David Harel, Dexter Kozen and Jerzy Tiuryn
  MIT Press
  $\rightarrow$ Department Library



Motivating Example

Introductory Example



The Towers of Hanoi

The Instructions

1. Move alternatingly the smallest disk and another one.
2. If moving the smallest disk put it on the stack it did not come from in its previous move.
3. If not moving the smallest disk do the only legal move,

More formally:

sequence of actions

$$\mathit{moveS} \,;\, \mathit{moveO} \,;\, \mathit{moveS} \,;\, \mathit{moveO} \,;\, \ldots$$

more concisely:

$$(\mathit{moveS} \,;\, \mathit{moveO})^*$$

improved:

$$\mathit{moveS} \,;\, \mathit{testForStop} \,;\; (\mathit{moveO} \,;\, \mathit{moveS} \,;\, \mathit{testForStop})^*$$

Properties

Atomic statement: $S1$ true iff smallest piece on first stack

Moving away

(1) $S1 \rightarrow \langle moveS \rangle \neg S1$

... after moving the smallest, it is no longer on the first stack

Moving other

(2) $S1 \rightarrow \langle moveO \rangle S1$

... after moving something else, it is still on the first stack

Conclusions from (1) and (2)

$S1 \rightarrow \langle moveO \; ; \; moveS \rangle \neg S1$

$S1 \rightarrow \langle (moveO)^* \; ; \; moveS \rangle \neg S1$

$\Uparrow$ THAT'S DYNAMIC LOGIC $\Uparrow$

Dynamic Logic

Dynamic Logic

- Allows reasoning about properties of composite actions.
- Actions are explicitly part of the language.
- Extends modal logic
- We look at two instances:
  - Propositional Dynamic Logic
  - First Order Dynamic Logic

Recap: Modal Logic

Syntax/semantics of dynamic logic build on top of modal logic.

Syntax:

- Signature Σ: set of propositional variables
- Fml_{Σ}^{ML} smallest set with:
  - $\Sigma \subseteq Fml_{\Sigma}^{mod}$
  - $true, false \in Fml_{\sigma}^{mod}$
  - $A, B \in Fml_{\Sigma}^{mod} \implies A \wedge B, A \vee B, A \rightarrow B, \neg A \in Fml_{\Sigma}^{mod}$
  - $A \in Fml_{\Sigma}^{mod} \implies \Box A, \Diamond A \in Fml_{\Sigma}^{mod}$
- pronounced “Box” and “Diamond”

Recap: Modal Logic – Semantics

Modal Logics

Logics of *necessity* and *possibility*.

Meaning of Modalities:

Modal

$\Box A$ It is necessary that ...

$\Diamond A$ It is possible that ...

Deontic (from Greek for duty)

$\Box A$ It is obligatory that ...

$\Diamond A$ It is permitted that ...

Epistemic (logic of knowledge)

$\Box A$ I know that ...

$\Diamond A$ I consider it possible that ...

Recap: Modal Logic – Semantics

Unified Semantics

In late 1950s Saul Kripke defined unified semantics for all "meanings" of modal operators:
"worlds" and "accessibility" between them

Kripke Frame (S, R):

- Set S of *worlds* (or *states*)
- Relation $R \subseteq S \times S$, the *accessibility relation*

Kripke Structure (S, R, I):

- Given a signature Σ
- Kripke Frame (S, R)
- Interpretation $I : S \to 2^{\Sigma}$

Recap: Modal Logic – Semantics

For a signature Σ and Kripke structure (S, R, I)

$I, s \models \varphi \iff$ Formula φ holds in state $s \in S$
$I \models \varphi \iff$ Formula φ holds in all states $s \in S$

$$I, s \models p \quad \iff \quad p \in I(s) \qquad \text{for } p \in \Sigma$$

$\models$ is homomorphic for $\wedge, \vee, \rightarrow, \neg$, i.e.:

$$\begin{array}{lcl}
I, s \models \varphi \wedge \psi & \iff & I, s \models \varphi \text{ and } I, s \models \psi \\
I, s \models \varphi \vee \psi & \iff & I, s \models \varphi \text{ or } I, s \models \psi \\
I, s \models \varphi \rightarrow \psi & \iff & I, s \models \varphi \text{ implies } I, s \models \psi \\
I, s \models \neg\varphi & \iff & \text{not } I, s \models \varphi
\end{array}$$

$$I, s \models \Box\varphi \quad \iff \quad I, s' \models \varphi \text{ for all } s' \in S \text{ with } (s, s') \in R$$

$$I, s \models \Diamond\varphi \quad \iff \quad I, s' \models \varphi \text{ for some } s' \in S \text{ with } (s, s') \in R$$

Example: Chalkboard

More than one modality

Multi-modal logic

Have different Box operators with different accessibility relations:

$$\Box_\alpha, \Box_\beta, \Box_\gamma, \ldots$$

($\rightarrow$ basic actions ins "Towers of Hanoi")

Propositional Dynamic Logic (PDL):

- Signature Σ of propositional variables
- Set $A = \{\alpha, \beta, \ldots\}$ of atomic actions/programs
- We write $[\alpha]$ instead of $\Box_\alpha$

PDL – Regular Programs

Compose Programs

Atomic programs can be into composed into larger programs

For a given signature Σ and atomic programs A,
the set of programs $\Pi_{\Sigma,A}$ is the smallest set such that

1. $A \subseteq \Pi_{\Sigma,A}$ — atomic programs
2. $p, q \in \Pi_{\Sigma,A} \implies (p\,;\,q) \in \Pi_{\Sigma,A}$ — sequential composition
3. $p, q \in \Pi_{\Sigma,A} \implies (p \cup q) \in \Pi_{\Sigma,A}$ — nondeterministic choice
4. $p \in \Pi_{\Sigma,A} \implies p^* \in \Pi_{\Sigma,A}$ — indeterminate iteration
5. $F \in Fml^{PDL}_{\Sigma,A} \implies \mathbf{?}F \in \Pi_{\Sigma,A}$ — tests

Regular Programs =
Regular Expressions over atomic programs and tests

PDL – Formulae

For a given signature Σ and atomic programs A,
the set of formulae $Fml^{PDL}_{\Sigma,A}$ is the smallest set such that

1. $true, false \in Fml^{PDL}_{\Sigma,A}$
2. $\Sigma \subseteq Fml^{PDL}_{\Sigma,A}$
3. $A, B \in Fml^{PDL}_{\Sigma,A} \implies A \wedge B, A \vee B, A \to B, \neg A \in Fml^{PDL}_{\Sigma,A}$
4. $P \in \Pi_{Sigma,A}, A \in Fml^{PDL}_{\Sigma,A} \implies [P]A, \langle P \rangle A \in Fml^{PDL}_{\Sigma,A}$

Programs and Formulae are mutually dependent definitions and must be seen simultaneously.

PDL Formulas – Examples

→ Towers of Hanoi

$A = \{moveS, moveO\}, \qquad \Sigma = \{S1\}$

$$S1 \rightarrow \langle (moveO)^* \, ; \, moveS \rangle \neg S1$$

multi-level and nested modalities

$A = \{\alpha, \beta\}, \qquad \Sigma = \{P, Q\}$

$$[\alpha \cup (\mathbf{?}P \, ; \, \beta)^*]Q$$

$$[\alpha]P \rightarrow [\alpha^*]P$$

$$[\alpha]\langle\beta\rangle\big(P \rightarrow [\alpha^*]Q\big)$$

$$[\alpha \, ; \, \mathbf{?}\langle\beta\rangle P \, ; \, \beta]Q$$

PDL – Semantics

Given a signature Σ and atomic programs A

(multi-modal propositional) Kripke frame (S, ρ)

- set of states S
- function $\rho : A \to S \times S$ accessibility relations for atomic programs

Kripke structure (S, ρ, I)

- Kripke frame (S, ρ)
- interpretation $I : S \to 2^{\Sigma}$

$\Rightarrow$ same as for modal logic

PDL – Program Semantics

Extension of ρ

from $\rho : A \to S^2$ to $\rho : \Pi_{\Sigma,A} \to S^2$

$$
\begin{array}{ll}
\rho(\alpha) & \text{base case for } \alpha \in A \\[1ex]
\rho(\pi_1 \cup \pi_2) & = \rho(\pi_1) \cup \rho(\pi_2) \\[1ex]
\rho(\pi_1 \,;\, \pi_2) & = \rho(\pi_1) \,;\, \rho(\pi_2) \\
 & = \{(s, s') \mid \text{ex. } t \text{ with } (s, t) \in \rho(\pi_1) \text{ and } (t, s') \in \rho(\pi_2)\} \\[1ex]
\rho(\pi^*) & = \mathrm{rtcl}(\rho(\pi)) = \bigcup_{n=0}^{\infty} \rho(\pi)^n \qquad \textit{refl. transitive closure} \\
 & = \{(s_0, s_n) \mid \text{ex. } n \text{ with } (s_i, s_{i+1}) \in \rho(\pi) \text{ for } 0 \leq i < n\} \\[1ex]
\rho(\mathbf{?}A) & = \{(s, s) \mid I, s \models A\}
\end{array}
$$

PDL – Semantics

For a signature Σ, basic programs A and Kripke structure (S, ρ, I)

$$I, s \models p \quad \iff \quad p \in I(s) \qquad \text{for } p \in \Sigma$$

$\models$ is homomorphic for $\wedge, \vee, \rightarrow, \neg$.

$$I, s \models [\pi]\varphi \quad \iff \quad I, s' \models \varphi \text{ for all } s' \in S \text{ with } (s, s') \in \rho(\pi)$$

$$I, s \models \langle\pi\rangle\varphi \quad \iff \quad I, s' \models \varphi \text{ for some } s' \in S \text{ with } (s, s') \in \rho(\pi)$$

Tautologies

Dual operators

$$[\pi]\varphi \;\leftrightarrow\; \neg\langle\pi\rangle\neg\varphi$$

- $[\pi_1 \,;\, \pi_2]\varphi \;\leftrightarrow\; [\pi_1][\pi_2]\varphi$
- $[\pi_1 \cup \pi_2]\varphi \;\leftrightarrow\; [\pi_1]\varphi \wedge [\pi_2]\varphi$
- $[\mathbf{?}\psi]\varphi \;\leftrightarrow\; \psi \rightarrow \varphi$
- $[\pi^*]\varphi \;\leftrightarrow\; \varphi \wedge [\pi \,;\, \pi^*]\varphi$

- $\langle\pi_1 \,;\, \pi_2\rangle\varphi \;\leftrightarrow\; \langle\pi_1\rangle\langle\pi_2\rangle\varphi$
- $\langle\pi_1 \cup \pi_2\rangle\varphi \;\leftrightarrow\; \langle\pi_1\rangle\varphi \vee \langle\pi_2\rangle\varphi$
- $\langle\mathbf{?}\psi\rangle\varphi \;\leftrightarrow\; \psi \wedge \varphi$
- $\langle\pi^*\rangle\varphi \;\leftrightarrow\; \varphi \vee \langle\pi \,;\, \pi^*\rangle\varphi$

- all tautologies for modal logic **K**

A Calculus for Propositional Dynamic Logic

Axioms

All propositional tautologies

$$
\begin{array}{lcll}
[\pi](\varphi \to \psi) & \to & ([\pi]\varphi \to [\pi]\psi) & (\mathsf{ML1} = \mathsf{K}) \\
[\pi](\varphi \land \psi) & \leftrightarrow & [\pi]\varphi \land [\pi]\psi & (\mathsf{ML2}) \\
[\pi_1 ; \pi_2]\varphi & \leftrightarrow & [\pi_1][\pi_2]\varphi & (\mathsf{PDL1}) \\
[\pi_1 \cup \pi_2]\varphi & \leftrightarrow & [\pi_1]\varphi \land [\pi_2]\varphi & (\mathsf{PDL2}) \\
[A?]\varphi & \leftrightarrow & A \to \varphi & (\mathsf{PDL3}) \\
[\pi^*]\varphi & \leftrightarrow & \varphi \land [\pi][\pi^*]\varphi & (\mathsf{PDL4}) \\
\varphi \land [\pi^*](\varphi \to [\pi]\varphi) & \to & [\pi^*]\varphi & (\mathsf{IND})
\end{array}
$$

Rules

$$\frac{\varphi, \; \varphi \to \psi}{\psi} \qquad (\mathsf{MP})$$

$$\frac{\varphi}{[\pi]\varphi} \qquad (\mathsf{GEN})$$

Theorem

Theorem

The presented calculus is sound and complete.

Proof

See e.g.,pp. 559-560
in David Harel's article *Dynamic Logic*
in the *Handbook of Philosophical Logic*, *Volume II*,
published by D.Reidel in 1984.

or

D. Harel, D. Kozen and J. Tiuryn
Dynamic Logic
in *Handbook of Philosophical Logic, 2nd edition , volume 4*
by Kluwer Academic Publisher, 2001.

Higher level program constructors

Syntactic Sugar

- PDL syntax has elementary program operators
- Enrich it by defining new operators ("macros")

$$
\begin{aligned}
\textbf{skip} &:= \textbf{?}\mathit{true} \\
\textbf{fail} &:= \textbf{?}\mathit{false} \\
\textbf{if } \varphi \textbf{ then } \alpha \textbf{ else } \beta &:= \big(\textbf{?}\varphi \,;\, \alpha\big) \cup \big(\textbf{?}\neg\varphi \,;\, \beta\big) \\
\textbf{while } \varphi \textbf{ do } \alpha &:= (\textbf{?}\varphi \,;\, \alpha)^* \,;\, \textbf{?}\neg\varphi
\end{aligned}
$$

More PDL Tautologies

$$
\begin{aligned}
[\textbf{skip}]\varphi &\leftrightarrow \varphi \\
\langle\textbf{skip}\rangle\varphi &\leftrightarrow \varphi \\
[\textbf{fail}]\varphi &\leftrightarrow \mathit{true} \\
\langle\textbf{fail}\rangle\varphi &\leftrightarrow \mathit{false} \\
[\textbf{if}\ \varphi\ \textbf{then}\ \alpha\ \textbf{else}\ \beta]\varphi &\leftrightarrow (\varphi \to [\alpha]\varphi) \land (\neg\varphi \to [\beta]\varphi) \\
\langle\textbf{if}\ \varphi\ \textbf{then}\ \alpha\ \textbf{else}\ \beta\rangle\varphi &\leftrightarrow (\varphi \to \langle\alpha\rangle\varphi) \land (\neg\varphi \to \langle\beta\rangle\varphi)
\end{aligned}
$$

Decidability

Decidability

Is PDL decidable?

$$\Longleftrightarrow$$

Is there an algorithm that terminates on every input and computes whether a PDL-formula $\phi \in Fml^{PDL}_{\Sigma,A}$ is satisfiable.

$$\Longleftrightarrow$$

Is there an algorithm that terminates on every input and computes whether a PDL-formula $\phi \in Fml^{PDL}_{\Sigma,A}$ is valid.

Answer:

YES, PDL is decidable!

Fischer and Ladner (1979)

General Idea:

$\varphi \in Fml^{PDL}$ has a model $\iff$ φ has a model of bounded size.

For every Kripke structure, a bounded Kripke structure can be defined which is indistinguishable for φ.

Preliminary lemma: Decidability for modal logic

The proof idea is the same, yet simpler.

Fischer-Ladner Closure

Reduced syntax

Only connectors $\rightarrow$, *false*, $\Box$ are allowed $\Rightarrow$ simplifies proofs.

Operator

$$FL^{mod} : Fml^{mod} \rightarrow 2^{Fml^{mod}}$$

assigns to φ the set of subformulas of φ.

$$\begin{aligned}
FL^{mod}(\varphi \rightarrow \psi) &= \{\varphi \rightarrow \psi\} \cup FL^{mod}(\varphi) \cup FL^{mod}(\psi) \\
FL^{mod}(\mathit{false}) &= \{\mathit{false}\} \\
FL^{mod}(p) &= \{p\} \qquad p \in \Sigma \\
FL^{mod}(\Box\varphi) &= \{\Box\varphi\} \cup FL^{mod}(\varphi)
\end{aligned}$$

Observation

$$|FL^{mod}(\varphi)| \leq |\varphi|$$

Fischer-Ladner Filtration

Filtration

For a Kripke structure S, R, I define a bounded structure $\tilde{S}, \tilde{R}, \tilde{I}$ with $$S, R, I, s \models \varphi \iff \tilde{S}, \tilde{R}, \tilde{I}, \tilde{s} \models \varphi$$

Central Idea

States are **undistinguishable** for φ if they are equal on $FL^{mod}(\varphi)$.

$$s \equiv t \iff (I, s \models \psi \Leftrightarrow I, t \models \psi \text{ for all } \psi \in FL^{mod}(\varphi))$$

$$\begin{aligned}
\widetilde{s} &:= \{s' \mid s' \equiv s\} && \ldots \text{ equivalence classes} \\
\widetilde{S} &:= \{\widetilde{s} \mid s \in S\} \\
\widetilde{R} &:= \{(\widetilde{s}, \widetilde{s'}) \mid (s, s') \in R\} \\
\widetilde{I}(p) &:= \{\widetilde{s} \mid s \in I(p)\}
\end{aligned}$$

Fischer-Ladner Filtration

$$\widetilde{s} := \{s' \mid s' \equiv s\}$$
$$\widetilde{S} := \{\widetilde{s} \mid s \in S\}$$
$$\widetilde{R} := \{(\widetilde{s}, \widetilde{t}) \mid (s, t) \in R\}$$
$$\widetilde{I}(p) := \{\widetilde{s} \mid s \in I(p)\}$$

Lemma

$$|\widetilde{S}| \leq 2^{|FL^{mod}(\varphi)|} \leq 2^{|\varphi|}$$

Lemma (proved by structural induction)

$$S, R, I, s \models \varphi \iff \tilde{S}, \tilde{R}, \tilde{I}, \widetilde{s} \models \varphi$$

Theorem

Modal Logic (K) can be decided by inspecting a bounded number of models.

Fischer-Ladner Closure for PDL

Operator

$$FL : Fml^{PDL} \to 2^{Fml^{PDL}}$$

$FL(\varphi)$ smallest set satisfying

1	$\varphi \in FL(\varphi)$		
2	$(\psi_1 \to \psi_2) \in FL(\varphi)$	$\Rightarrow$	$\psi_1 \in FL(\varphi)$ and $\psi_2 \in FL(\varphi)$
3	$[\pi]\psi \in FL(\varphi)$	$\Rightarrow$	$\psi \in FL(\varphi)$
4	$[\pi_1;\pi_2]\psi \in FL(\varphi)$	$\Rightarrow$	$[\pi_1][\pi_2]\psi \in FL(\varphi)$
5	$[\pi_1 \cup \pi_2]\psi \in FL(\varphi)$	$\Rightarrow$	$[\pi_1]\psi \in FL(\varphi)$ and $[\pi_2]\psi \in FL(\varphi)$
6	$[\pi]^*\psi \in FL(\varphi)$	$\Rightarrow$	$[\pi][\pi^*]\psi \in FL(\varphi)$
7	$[\psi_1?]\psi_2 \in FL(\varphi)$	$\Rightarrow$	$\psi_1 \in FL(\varphi)$

Lemma (not obvious)

$$|FL(\varphi)| \;\leq\; |\varphi|$$

Fischer-Ladner Filtration

Same construction as for modal logic

extended: $\tilde{\rho}(a) := \{(\widetilde{s}, \widetilde{t}) \mid (s,t) \in \rho(a)\}$ for all $a \in A$

Lemma

$$S, R, I, s \models \varphi \iff \tilde{S}, \tilde{R}, \tilde{I}, \widetilde{s} \models \varphi$$

Prove by structural induction:

- If $\psi \in FL(\varphi)$ then $s \models \psi$ iff $\widetilde{s} \models \psi$
- $(s,t) \in \rho(\pi)$ implies $(\widetilde{s}, \widetilde{t}) \in \tilde{\rho}(\pi)$ for all $\pi \in \Pi_{\Sigma,A}$
- If $(\widetilde{s}, \widetilde{t}) \in \rho(\pi)$ and $s \models [\pi]\psi$, then $t \models \psi$ for $[\pi]\psi \in FL(\varphi)$

full proof: $\rightsquigarrow$ lecture notes or [Harel et al., Lemma 6.4]

Complexity

KIT Karlsruhe Institute of Technology

Naive approach used for proof

- $FL(\varphi) \in O(|\varphi|)$
- $|\tilde{S}| \leq 2^{FL(\varphi)} \in O(2^{|\varphi|})$ many states in filtration
- $|\text{models}| \leq (2^{\Sigma})^{|S|} \in O(2^{2^{|\varphi|}})$

$\Rightarrow$ double exponential complexity

One can do better:

Complexity of Deciding PDL

The decision problem for PDL is in EXPTIME:
can be decided by a deterministic algorithm in $O(2^{p(n)})$ for some polynomial p.

$\rightsquigarrow$[Harel et al. Ch. 8]

Deduction Theorem and Compactness

Logical Consequence

$M \subseteq Fml^{PDL}, \quad \varphi \in Fml^{PDL}$

Global Consequence

$M \models^G \varphi :\iff$

for all Kripke structures (S, ρ, I):

$I, s \models M$ for all $s \in S$ implies $I, s \models \varphi$ for all $s \in S$

Local Consequence

$M \models^L \varphi :\iff$

for all Kripke structures (S, ρ, I):

for all $s \in S$: $I, s \models M$ implies $I, s \models \varphi$

Local consequence is stronger: $M \models^L \varphi \quad \underset{\not\Longleftarrow}{\Longrightarrow} \quad M \models^G \varphi$

Deduction Theorem

Recall: In propositional logic:

$$M \cup \{\varphi\} \models \psi \quad \iff \quad M \models \varphi \to \psi$$

Not valid for PDL:

$$p \models^G [\alpha]p \;\text{ but }\; \not\models^G p \to [\alpha]p$$

Problem:
Decidability has been shown only for $\models \varphi$.

Questions

1. Is $\psi \models^G \varphi$ decidable for PDL?
2. Is $M \models^G \varphi$ decidable for PDL?

Deduction Theorem Revised

Lemma

$$\psi \models^G \varphi \quad \Longleftrightarrow \quad \models [(\beta_1 \cup \ldots \cup \beta_k)^*]\psi \to \varphi$$

with $B := \{\beta_1, ..., \beta_k\}$ the atomic programs occurring in ψ, φ.

$\Longleftarrow$ simple $\rightsquigarrow$ Exercise

$\Longrightarrow$

1. Kripke structure (S, ρ, I), $s \in S$.
2. $S^-(s) := \{s' \mid s'$ reachable from s via $B.\} \subseteq S$
3. $S^-(s) \models \psi \iff S^-(s), s \models [B^*]\psi$
4. $S^-(s) \models \psi$ entails $S^-(s) \models \varphi$ by assumption
5. $S^-(s), s \models \varphi \iff S, s \models \varphi$

Decidable:

The consequence problem $\psi \models^G \varphi$ **is** decidable for PDL.

Compactness of PDL

Recall: Compactness Theorem

$$M \models^G \varphi \quad \Longleftrightarrow \quad \text{exists finite } E \subseteq M \text{ with } E \models^G \varphi$$

Holds for:

Propositional Logic, First Order Logic, **not** for higher order logic

Counterexample for PDL

$$M := \{p \to [\underbrace{\alpha\,;\ldots;\alpha}_{n \text{ times}}]q \mid n \in \mathbb{N}\}, \qquad \varphi := p \to [\alpha^*]q$$

- $M \models^G \varphi$? *yes*
- $E \subset M,\ E \models^G \varphi$? *no*

PDL is not compact

because it has transitive closure "built in".

Deducibility Problem in PDL

Quote:

> *[T]he problem of whether an arbitrary PDL formula p is deducible from a single fixed axiom scheme is of extremely high degree of undecidability, namely* Π^1_1*-complete.*

Meyer, Streett, Mirkowska:
The Deducibility Problem in Propositional Dynamic Logic, 1981

Variants and Conclusion

Variant: Converse Programs

Idea: Add actions reverting action effects

Add further program constructor $\cdot^{-1}$:
$\pi \in \Pi \implies \pi^{-1} \in \Pi$
with $\rho(\pi^{-1}) = \rho(\pi)^{-1}$

Axiom schemes: for all $\varphi \in Fml^{PDL}$, $\pi \in \Pi$

- $\varphi \to [\pi]\langle \pi^{-1} \rangle \varphi$
- $\varphi \to [\pi^{-1}]\langle \pi \rangle \varphi$

Complete

Adding the axioms to the known PDL calculus gives a correct and complete calculus for PDL with Converse.

Variant: Context-free Programs

Idea: Go beyond regular programs

Instead of regular programs, allow context-free grammar

For example:

Produced context-free grammar $X ::= \alpha X \gamma \mid \beta$
with $L(X) = \{\alpha^n \beta \gamma^n \mid n \in \mathbb{N}\}$

Undecidability result

Validity is undecidable if instead of regular programs, context-free programs are allowed.

Expressiveness

Without fixed semantics of $\mathbb{N}$, recursion is strictly more expressive than looping.

State Vector Semantics

A propositional Kripke structure $\mathcal{K} = (S, \rho, I)$ is determined by:

S	the set of states
$\rho : A \to S \times S$	the accessibility relations for atomic programs e
$I : S \to 2^{\Sigma}$	evaluation of propositional atoms in states

Choose now: $S \subseteq 2^{\Sigma}$ the set of states

We call this the state vector semantics.

- Strictly larger set of tautologies.
- Obviously decidable.
- Evaluation of propositional variables fixes the state (and the accessibility of successor states)

Lemma

Let

- $A = \{a_1, \ldots, a_k\}$
- π_{all} stands for the program $(a_1 \cup \ldots \cup a_k)^*$.
- $U \subseteq \Sigma$ be a subset of the set of propositional atoms.
- $state_U$ abbreviate $\bigwedge_{p \in U} p \wedge \bigwedge_{p \notin U} \neg p$.
- F an arbitrary PDL formula.

Then

$$\langle \pi_{all} \rangle (state_U \wedge F) \rightarrow [\pi_{all}](state_U \rightarrow F)$$

is true in all state vector Kripke structures.

Theorem

Let H be the set of all formulas

$$\langle \pi_{all} \rangle (state_U \wedge F) \rightarrow [\pi_{all}](state_U \rightarrow F)$$

with the notation from the previous slide.

Then:

1. $\{F\} \cup H$ is satisfiable iff F is state vector satisfiable.
2. $H \models F$ iff $\models_{sv} F$.

Propositional Dynamic Logic – Summary

- extension of modal logic
- abstract notion of actions / atomic logic statements
- regular programs, with non-deterministic choice and Kleene-interation
- correct and complete calculus for tautologies
- satisfiability is decidable (in EXPTIME)
- logic is not compact
- deducibility is utterly undecidable
- deduction theorem can be rescued

Detection of dynamic execution errors in

IBM system automation's rule-based expert system

An Application of PDL

Reference

[SinzEtAl02]

Carsten Sinz, Thomas Lumpp, Jürgen Schneider, and Wolfgang Küchlin:
Detection of dynamic execution errors in IBM System Automation's rule-based expert system.
Information and Software Technology, 44(14):857–873, November 2002.

Context

Context

IBM zSeries

- **z** $=$ zero downtime
- high availability: 99.999%
- $< 5.3 min/yr$ downtime

System Automation

- full automation of a data center
- starting, stopping, migration of applications
- recovery from system failures
- ...
- **complex, rule-based configuration**

Example

Flight booking center: 100s of users, many parallel apps

Example Rule

```
correlation set/status/compound/satisfactory :
when        status/compound NOT E {Satisfactory}
        AND status/startable E {Yes}
        AND ( ( status/observed E {Available, WasAvailable}
                AND status/desired E {Available}
                AND status/automation E {Idle, Internal}
                AND correlation/external/stop/failed E {false}
              )
              OR
              ( status/observed E {SoftDown, StandBy}
                AND status/desired E {Unavailable}
                AND status/automation E {Idle, Internal}
              )
            )
then    SetVariable status/compound = Satisfactory
        RecordVariableHistory status/compound
```

Fig. 4. Example of a correlation rule.

(taken from [SinzEtAl02])

Rules

when *cond* **then** $var = d$

- **AND**, **OR**, **NOT** allowed in conditions
- *var* **E** $\{\ d_1, \ldots, d_2\ \}$ – “element of”
- the **then** part can be executed if **cond** is true

Logical Encoding

- One boolean atom per var/value-pair
- $P_{var,d} = true \iff var = d$

- Encode that *var* has exactly one value (of d_1, ..., d_k)
- $\left(\bigvee_{i=1..k} P_{var,d_i} \right) \wedge \left(\bigwedge_{\substack{i,j=1..k \\ i<j}} \neg(P_{var,d_i} \wedge P_{var,d_j}) \right)$

- Atomic Actions: $var = d \rightsquigarrow \alpha_{var,d}$
- Axiom $[\alpha_{var,d}]P_{var,d}$

Logical Encoding

Semantics of a rule as program:

$$\mathbf{?}\mathit{when}\,;\,\mathit{then}$$

Semantics of all rules as program:

$$R := \big((\mathbf{?}\mathit{when}_1\,;\,\mathit{then}_1) \cup \ldots \cup (\mathbf{?}\mathit{when}_r\,;\,\mathit{then}_r)\big)^*$$

Proof Obligations

Uniqueness of final state:
under assumption of a precondition *PRE*

$$PRE \rightarrow \big(\langle R \rangle p \leftrightarrow [R]p\big)$$

Confluence:

$$PRE \rightarrow \big(\langle R \rangle [R]p \rightarrow [R]\langle R \rangle p\big)$$

Absence of Oscillation:
modelled using an extension of PDL with non-termination operator

Verification Experiment

Verification Technique

- state vector semantics
- translation of PDL to boolean SAT
- solving using SAT solver (Davies-Putnam)

Experiment:

- $\sim$40 rules
- resulted in $\sim$1500 boolean variables
- SAT solving < 1 sec

!! **violations found – before deployment**