Propositional Dynamic Logic

Mathematical Logic for Computer Science

Alessio Bandiera

1985878

Contents

Dynamic Logics

Dynamic Logics are modal logics for representing states and events of dynamic systems

The first DL system was developed in 1976 by Vaughan Pratt, early pioneer of CS. His original DL was a *first-order* modal logic, and **Propositional Dynamic Logic** (PDL) is the propositional counterpart of it

Being propositional, its only two syntactic categories are **propositions** and **programs**, and *possibility* and *necessity* are expressed through modal operators that also indicate the programs they are referring to

- $\langle\pi\rangle\phi$ is read "there is an execution of π that ends in a state in which ϕ is true"
- $[\pi]\psi$ is read "all executions of the program π end in states in which ψ is true"

Contents

Syntax

Formulas

Given Φ_0 the set of *atomic formulas*, for any $\phi, \psi \in \Phi_0$

- $\phi \in \mathrm{Form}(\Phi_0)$
- $\neg\phi \in \mathrm{Form}(\Phi_0)$
- $\phi \vee \psi \in \mathrm{Form}(\Phi_0)$
- $[\alpha]\phi \in \mathrm{Form}(\Phi_0)$

where $\alpha \in \mathrm{Prog}(\Pi_0)$

Syntax

Programs

Given Π_0 the set of *atomic programs*, for any $\alpha, \beta \in \Pi_0$

- $\alpha \in \mathrm{Prog}(\Pi_0)$
- $(\alpha; \beta) \in \mathrm{Prog}(\Pi_0)$
- $(\alpha \cup \beta) \in \mathrm{Prog}(\Pi_0)$
- $\alpha^* \in \mathrm{Prog}(\Pi_0)$
- $\phi? \in \mathrm{Prog}(\Pi_0)$

where $\phi \in \mathrm{Form}(\Phi_0)$

Syntax

Relations

$$(x, y) \in R(\pi) \iff x \xrightarrow{\pi} y$$

- $(x, y) \in R(\alpha; \beta) \iff \exists z \in W \quad (x, z) \in R(\alpha) \wedge (z, y) \in R(\beta)$
- $(x, y) \in R(\alpha \cup \beta) \iff (x, y) \in R(\alpha) \vee (x, y) \in R(\beta)$
- $(x, y) \in R(\alpha^*) \iff \exists z_0, \ldots, z_n \in W \quad \begin{cases} z_0 = x \\ z_n = y \\ (z_{k-1}, z_k) \in R(\alpha) \end{cases}$
- $(x, y) \in R(\phi?) \iff x = y \wedge y \in V(\phi)$

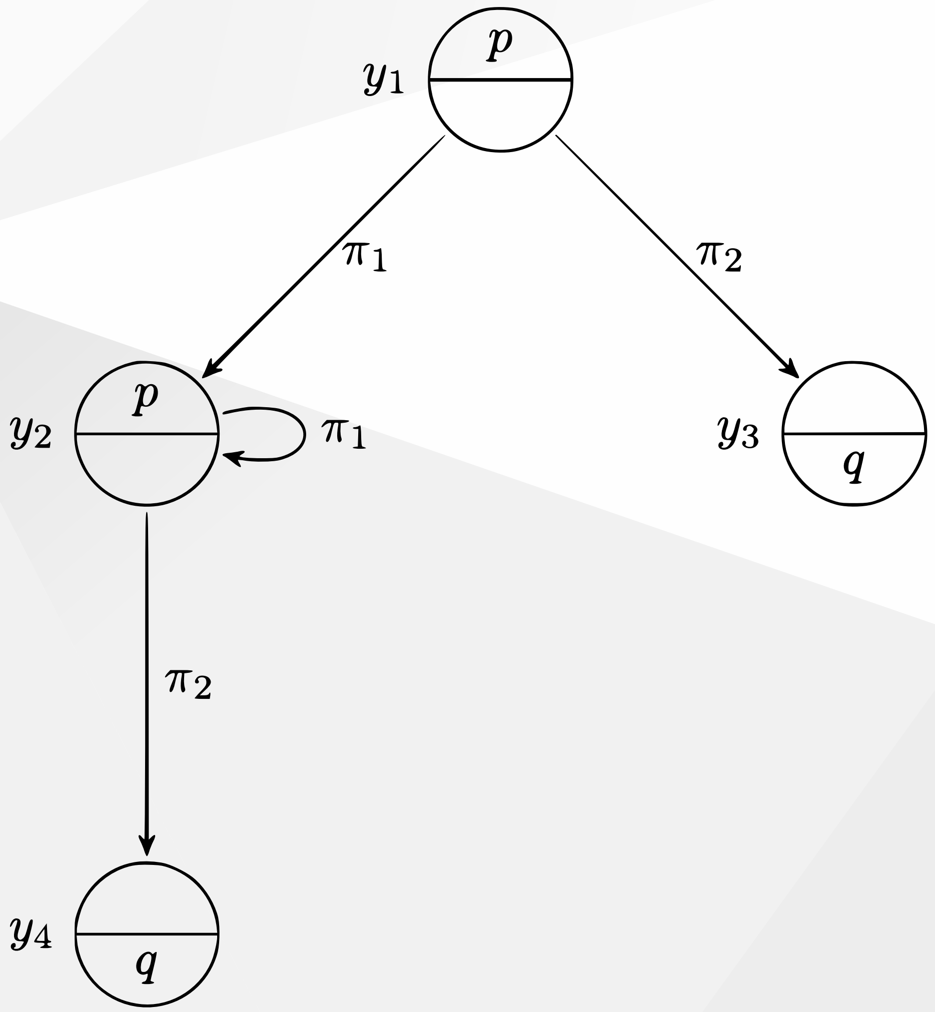
Syntax

Valuations

$$x \in V(p) \iff p \text{ is true at } x$$

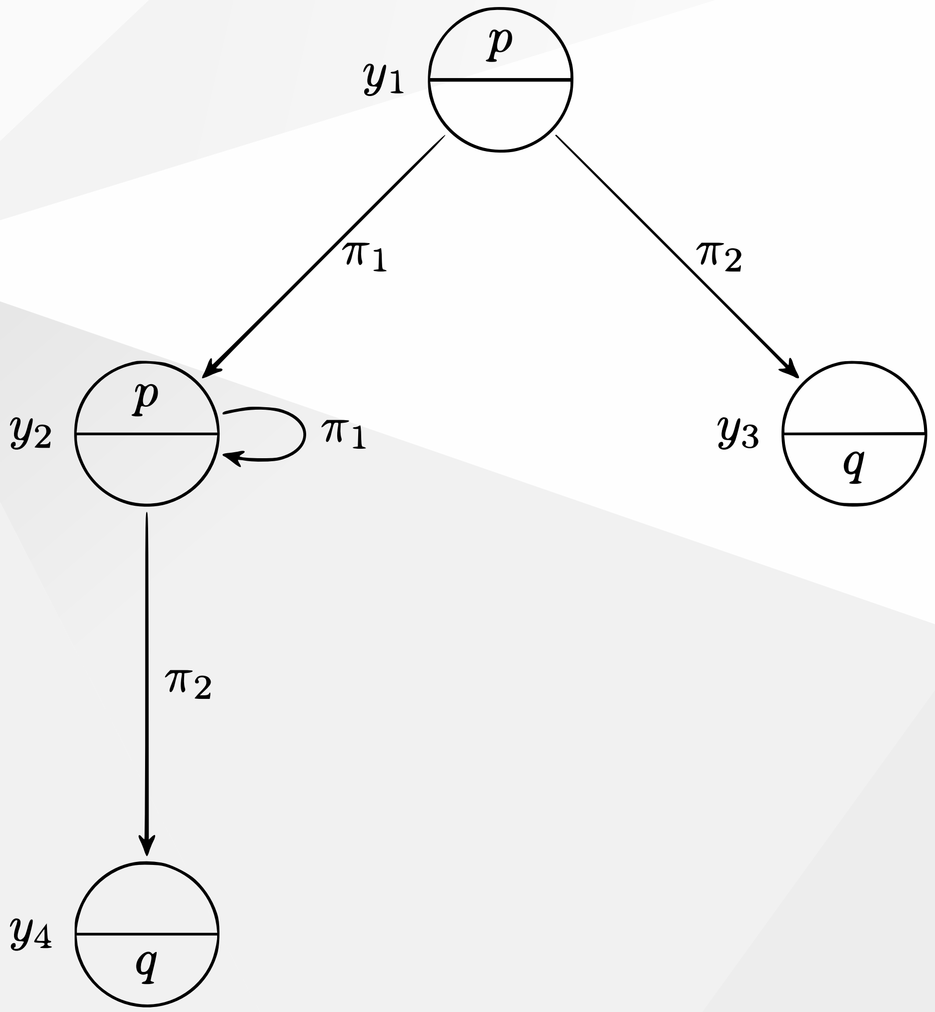
- $V(\bot) = \varnothing$
- $V(\top) = W$
- $V(\neg\phi) = W - V(\phi)$
- $V(\phi \vee \psi) = V(\phi) \cup V(\psi)$
- $V([\alpha]\phi) = \{x \mid \forall y \in W \quad (x, y) \in R(\alpha) \implies y \in V(\phi)\}$

LTS



- $W = \{y_1, y_2, y_3, y_4\}$
- $R(\pi_1) = \{(y_1, y_2), (y_2, y_2)\}$
- $R(\pi_2) = \{(y_1, y_3), (y_2, y_4)\}$
- $V(p) = \{y_1, y_2\}$
- $V(q) = \{y_3, y_4\}$

LTS



- $\mathfrak{M}, y_1 \models \langle \pi_1^*; \pi_2 \rangle q$
- $\mathfrak{M}, y_2 \models [\pi_1^*] p$
- $\mathfrak{M}, y_1 \models [\pi_1 \cup \pi_2](p \vee q)$
- $\mathfrak{M}, y_3 \models [\pi_1 \cup \pi_2]\bot$

Contents

Axiomatization

Goal

The goal is to define a **decudibility predicate** $\vdash$ such that $\vdash$-deductions are both *sound* and *complete* in terms of *validity*, i.e. for any ϕ it holds that

$$\vdash \phi \iff \models \phi$$

where $\models \phi$ means that ϕ is **valid**

Axiomatization

Validity

We write $\mathfrak{M}, w \models \phi$ if and only if $w \in V(\phi)$

ϕ is *valid* in $\mathfrak{M}$, written as $\mathfrak{M} \models \phi$, if and only if

$$\mathfrak{M} \models \phi \iff \forall w \in W \quad \mathfrak{M}, w \models \phi$$

ϕ is **valid**, written as $\models \phi$, if and only if

$$\models \phi \iff \forall \mathfrak{M} \quad \mathfrak{M} \models \phi$$

Axiomatization

K and N axioms

$$
\begin{array}{ll}
(\mathrm{K}) & [\alpha](\phi \to \psi) \to ([\alpha]\phi \to [\alpha]\psi) \\
(\mathrm{N}) & \dfrac{\phi}{[\pi]\phi}
\end{array}
$$

A modal logic is **normal** if it obeys (K) and (N)

Axiomatization

PDL axioms

PDL is the *least normal* modal logic containing every instance of

$$\begin{array}{ll} \text{(A1)} & [\alpha;\beta]\phi \leftrightarrow [\alpha][\beta]\phi \\ \text{(A2)} & [\alpha\cup\beta]\phi \leftrightarrow [\alpha]\phi \wedge [\beta]\phi \\ \text{(A3)} & [\alpha^*]\phi \leftrightarrow \phi \wedge [\alpha][\alpha^*]\phi \\ \text{(A4)} & [\phi?]\psi \leftrightarrow (\phi \rightarrow \psi) \end{array}$$

and closed under the *loop invariance* rule of inference

$$\text{(I)} \qquad \frac{\phi \rightarrow [\alpha]\phi}{\phi \rightarrow [\alpha^*]\phi}$$

Axiomatization

$\vdash$-deducibility

A formula ϕ is $\vdash$-*deducible* from $\Sigma \subseteq \mathrm{Form}(\Phi_0)$ if there exists a sequence $\phi_0, \ldots, \phi_n$ such that $\phi_n = \phi$, and for all $i \in [n]$

- ϕ_i is an instance of an axiom schema
- ϕ_i is an instance of a formula of Σ
- ϕ_i comes from earlier formulas of the sequence by inference

Are $\vdash$-deductions sound and complete?

Contents

Soundness and Completeness

Segerberg's axioms

In 1977 Segerberg proposed to replace

$$(\mathrm{I}) \qquad \frac{\phi \to [\alpha]\phi}{\phi \to [\alpha^*]\phi}$$

with the following fifth axiom

$$(\mathrm{A5}) \qquad \phi \wedge [\alpha^*](\phi \to [\alpha]\phi) \to [\alpha^*]\phi$$

in order to prove that such axiomatization was sound and complete

Soundness and Completeness

Segerberg's axioms

Indeed, it is easy to prove that (I) can be replaced with $(\mathrm{A5})$

1. $\vdash \phi \to [\alpha]\phi$	(premise)
2. $\vdash [\alpha^*](\phi \to [\alpha]\phi)$	(from 1 using (N) with $\pi = \alpha^*$)
3. $\vdash \phi \wedge [\alpha^*](\phi \to [\alpha]\phi) \to [\alpha^*]\phi$	(A5)
4. $\vdash [\alpha^*](\phi \to [\alpha]\phi) \to (\phi \to [\alpha^*]\phi)$	(from 3 through prop. reasoning)
5. $\vdash \phi \to [\alpha^*]\phi$	(from 2 and 4 using *Modus Ponens*)

Soundness and Completeness

Soundness

To prove that $\vdash$ is sound w.r.t. $\models$, i.e. that

$$\vdash \phi \implies \models \phi$$

a proof by induction on the length of ϕ's deduction in $\vdash$ suffices

So, what about completeness? It requires to prove that

$$\models \phi \implies \vdash \phi$$

Soundness and Completeness

Completeness

Segerberg's work was the first attempt to prove the completeness of $\vdash$, however in 1978 he found a flaw in his argument

Then in the same year Parikh published what is now considered the first proof of the completeness of $\vdash$

Contents

Complexity

PDL satisfiability

ϕ is *satisfiable* in $\mathfrak{M}$ if there is a world $w \in W$ such that $\mathfrak{M}, w \models \phi$

ϕ is **satisfiable** if there is a model $\mathfrak{M}$ such that ϕ is satisfiable in $\mathfrak{M}$

$$\text{PDL-SAT} := \{\langle\phi\rangle \mid \phi \text{ is a satisfiable PDL formula}\}$$

Unsatisfiable formulas

ϕ is *unsatisfiable* if and only if $\neg\phi$ is *valid*

Therefore, we can use the recursive definition of valid PDL formulas and build a procedure P that enumerates all the $\vdash$-deducible formulas

Hence, given enough time if $\neg\phi$ is $\vdash$-deducible P will eventually find it and determine that ϕ is *unsatisfiable*

This proves that $\mathrm{PDL\text{-}SAT} \in$ coREC

Complexity

Satisfiable formulas

However, if ϕ is satisfiable P never terminates.

Nonetheless, we can leverage the **finite model property** of PDL

$$\forall \phi \in \mathrm{Form}(\Phi_0) \quad \langle \phi \rangle \in \mathrm{PDL\text{-}SAT} \implies \exists \mathfrak{M}_{fin} \text{ finite} \quad \phi \text{ satisfiable in } \mathfrak{M}_{fin}$$

Therefore, there is a procedure P' that enumerates all the finite models $\mathfrak{M}_{fin}$ and checks for each model if ϕ is satisfiable in $\mathfrak{M}_{fin}$

Thus, P and P' can be run in parallel to decide $\mathrm{PDL\text{-}SAT}$. However, this is *very* inefficient, can we do any better?

Complexity

Small model property

Kozen and Parikh proved that PDL has also the **small model property**

$$\forall\phi \in \mathrm{Form}(\Phi_0) \quad \langle\phi\rangle \in \mathrm{PDL\text{-}SAT} \implies \exists\mathfrak{M}_{fin} \text{ finite } \quad \begin{cases} |\mathfrak{M}_{fin}| < \exp(|\phi|) \\ \phi \text{ satisfiable in } \mathfrak{M}_{fin} \end{cases}$$

This property implies that we can stop P' as soon as all the "small" models have been exhausted, to conclude that ϕ is not satisfiable

This concludes that $\mathrm{PDL\text{-}SAT} \in$ NEXP. In 1980 Pratt was able to prove that $\mathrm{PDL\text{-}SAT} \in$ EXP-complete

Contents

Variants

- Variants
  - **Test-free PDL**
  - CPDL

Test-free PDL

Expressive power

The "$?$" operator seems *different* with respect to the other programs, can we remove this operator from PDL?

Let PDL_0 be the test-free version of PDL. In 1981 Berman and Paterson proved that this PDL formula

$$\langle (P?; A)^*; \neg P?; A; P?\rangle \top$$

has no PDL_0 equivalent formula

Test-free PDL

Ultimate periodicity

The idea of their counterexample is based on this result in the theory of context-free languages:

A *unary language* $L = \{1^n \mid n \in \mathbb{N}\}$ is *regular* if and only if the set $\{n \in \mathbb{N} \mid 1^n \in L\}$ is *ultimately periodic*

A set $S \subseteq \mathbb{N}$ is **ultimately periodic** if there are integers $X \in \mathbb{N}$ and $Y > 0$ such that

$$\forall k \geq X \quad k \in S \iff k + Y \in S$$

For instance, this set S is ultimately periodic

$$S = \{0, 1, 2, 4, 6, 7, 9, 11, 13, 15, \ldots\}$$

since it holds that $\forall k \geq 7 \quad k \in S \iff k + 2 \in S$

Test-free PDL

Ultimate periodicity

By removing tests from PDL formulas, programs are restricted to regular expressions

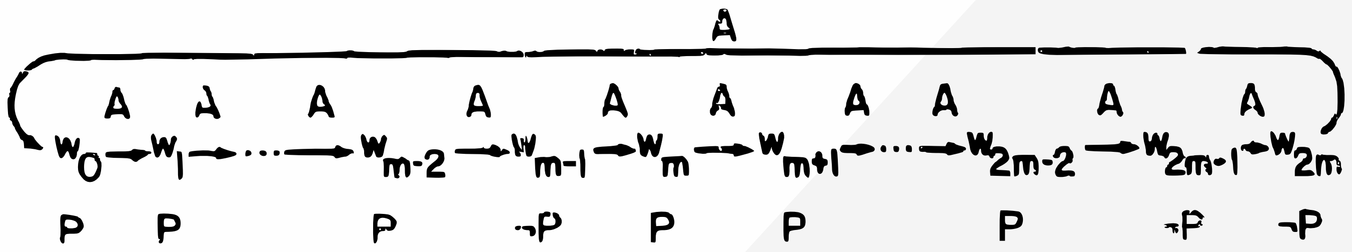
Hence, Berman and Paterson built a family of models $\mathfrak{A}_m$ for $m \geq 2$ in which the only program present is A

Therefore, by ultimate periodicity each program over $\mathfrak{A}_m$ can be rewritten as a regex

$$A^X(A^Y)^*$$

Test-free PDL

The counterexample

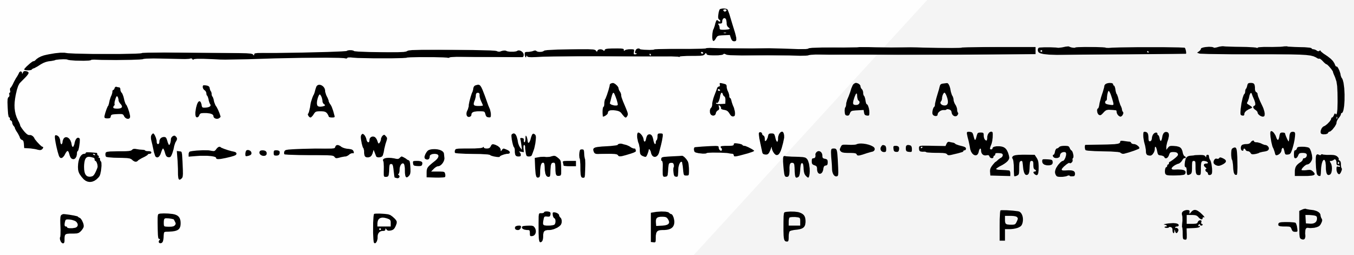

A
A A A A A A A A A A
w_0 w_1 ... w_{m-2} w_{m-1} w_m w_{m+1} ... w_{2m-2} w_{2m-1} w_{2m}
P P P ¬P P P P ¬P ¬P

Each $\mathfrak{A}_m$ consists of $2m + 1$ worlds, where $2m + 1$ is *prime*

This forces $(A^Y)^*$ to generate all the possible residues modulo $2m + 1$, i.e. each world will be able to reach any other world

Hence, test-free PDL formulas *cannot distinguish* the worlds in which we are performing the evaluation

The counterexample



However, tests *can* distinguish the worlds by building programs which **depend on the truthness of propositions**

$$\langle (P?; A)^*; \neg P?; A; P? \rangle \top$$

In fact, this formula is satisfied at w_0 but not satisfied at w_m

Variants

- Variants
    - Test-free PDL
    - **CPDL**

CPDL

The converse operator

CPDL is a variant which adds the **converse** operator to PDL programs

$$(x, y) \in R\left(\alpha^{-1}\right) \iff (y, x) \in R(\alpha)$$

To get a sound a complete system, two additional axioms are needed

$$\begin{aligned} &(\mathrm{A6}) \qquad \phi \to [\alpha]\left\langle \alpha^{-1} \right\rangle \phi \\ &(\mathrm{A7}) \qquad \phi \to \left[\alpha^{-1}\right]\langle \alpha \rangle \phi \end{aligned}$$

As for PDL, CPDL has the **small model property** too, and $\mathrm{CPDL\text{-}SAT} \in \mathsf{EXP}\text{-complete}$ as well

Expressive power

What about the expressive power? Consider these two models

$$\mathfrak{M} = (M, R, V) \qquad \mathfrak{M}' = (W', R', V')$$

$\mathfrak{M}$	$\mathfrak{M}'$
$W = \{x, y\}$	$W' = \{y'\}$
$R(\pi) = \{(x, y)\}$	$R'(\pi) = \varnothing$
$V(x) = V(y) = \varnothing$	$V'(y') = \varnothing$

CPDL

Expressive power

$\mathfrak{M}$	$\mathfrak{M}'$
$W = \{x, y\}$	$W' = \{y'\}$
$R(\pi) = \{(x, y)\}$	$R'(\pi) = \varnothing$
$V(x) = V(y) = \varnothing$	$V'(y') = \varnothing$

From the perspective of PDL y and y' are *indistinguishable*, in fact

$$\mathfrak{M}, y \models \phi \iff \mathfrak{M}', y' \models \phi$$

CPDL

Expressive power

$\mathfrak{M}$	$\mathfrak{M}'$
$W = \{x, y\}$	$W' = \{y'\}$
$R(\pi) = \{(x, y)\}$	$R'(\pi) = \varnothing$
$V(x) = V(y) = \varnothing$	$V'(y') = \varnothing$

However CPDL *can* distinguish y and y' because

$$\mathfrak{M}, y \models \langle \pi^{-1} \rangle \top \qquad \mathfrak{M}', y' \not\models \langle \pi^{-1} \rangle \top$$

meaning that CPDL has **more expressive power** than PDL

Thanks for your attention

Mathematical Logic for Computer Science

Alessio Bandiera

1985878