Advanced Logic 2014–15

Dimitri Hendriks

VU University Amsterdam
Theoretical Computer Science

week 5

partial correctness statements

- Euclid's gcd program (for positive integers):

$$\begin{aligned} \text{gcd} = \; & \text{while } x \neq y \text{ do } \{ \\ & \quad \text{if } x > y \\ & \quad \text{then } x := x - y \\ & \quad \text{else } y := y - x \; \}; \\ & \text{return } x; \end{aligned}$$

- think of a state as a set of pairs $x = t$, one for each variable x
- an example gcd-run on start state $\{x = 420, y = 96, \ldots\}$:

$$(x, y) = (420, 96) \rightarrow_{\text{gcd}} (12, 12)$$

- correctness of the gcd-program can be expressed by the PDL formula

$$\underbrace{(x = u) \wedge (y = v)}_{\text{precondition}} \rightarrow [\text{gcd}](\underbrace{x = gcd(u, v)}_{\text{postcondition}})$$

PDL: Propositional Dynamic Logic

- PDL is a formal system for reasoning about programs
- shares the goals of computer-assisted verification via model checking and theorem proving:
  - formalizing correctness specifications
  - proving that a program meets its specification
  - determining equivalence of programs
  - comparing expressive power of program constructs
- PDL is (modal, and so) dynamic, to model computation:
  - programs change values assigned to variables: $x := x + 1$
  - and so change the truth of formulas: x *is even*
- PDL abstracts from details of program execution, programs are interpreted as input–output relations
- PDL has explicit syntax for building regular programs out of atomic ones: composition, choice, iteration, and test
- these program constructors are interpreted as operations on input-output relations

PDL: Propositional Dynamic Logic

- to every program α we associate a modality $\langle\alpha\rangle$
- $\langle\alpha\rangle\varphi$: *it is possible to execute program* α *starting from the current state and halt in a state satisfying* φ
- $[\alpha]\varphi$: *if program* α *halts, it does so in a state satisfying* φ

PDL-programs and formulas

- let $\mathrm{A} = \{a, b, c, \dots\}$ be a set of atomic programs
- let $\textsc{Var} = \{p, q, r, \dots\}$ be a set of atomic propositions
- the sets $\textsc{Prog}$ and $\textsc{Form}$ of PDL-programs and PDL-formulas over $(\mathrm{A}, \textsc{Var})$ are mutually defined by:

$$\alpha ::= a \mid \alpha\,;\alpha \mid \alpha \cup \alpha \mid \alpha^* \mid \varphi? \qquad (a \in \mathrm{A})$$
$$\varphi ::= p \mid \top \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle\alpha\rangle\varphi \qquad (p \in \textsc{Var})$$

- let $*$ and $?$ bind stronger than $;$ and $;$ stronger than $\cup$
- sometimes we write $\alpha\beta$ instead of $\alpha\,;\beta$

informal meaning of program constructions

a atomic programs
basic, indecomposable; execute in one single step

$\alpha\,;\beta$ sequential composition
do α, then do β

$\alpha \cup \beta$ non-deterministic choice
choose α or β, and execute it

α^* iteration
choose an integer $n \geq 0$, and execute α n times

$\varphi?$ test
if φ then skip else abort
if φ holds, it continues without changing state, if not it blocks without halting

why allow non-determinism?

- non-determinism in PDL is due to the choice operator $\cup$ and the iterator construct (α^* is iterating α a non-deterministically chosen number of times)
- so in the programming language of PDL traces of a program need not be uniquely determined by their start states
- ... not so realistic, why?
- non-determinism is a useful tool when modelling situations where we cannot predict the outcome of a particular choice:
  computations may depend on external info outside of the programmer's control (e.g. user input)
- often we use $\cup$ and $*$ to build programs that force deterministic choice, like the standard constructs:

$$\begin{aligned}\text{if } \varphi \text{ then } \alpha \text{ else } \beta &= (\varphi? \,;\, \alpha) \cup (\neg\varphi? \,;\, \beta)\\ \text{while } \varphi \text{ do } \alpha &= (\varphi? \,;\, \alpha)^* \,;\, \neg\varphi?\end{aligned}$$

some example PDL-formulas

- $[\alpha \cup \beta]\varphi$

  always if we execute α or β, we arrive at a state where φ holds
- $\langle(\alpha\beta)^*\rangle\psi$

  there is a sequence of alternating executions of α and β such that we arrive at a state where ψ holds
- $\langle\alpha^*\rangle\varphi \leftrightarrow \varphi \vee \langle\alpha\,;\alpha^*\rangle\varphi$

  φ holds after a finite number ($n \geq 0$) of α-steps
  if and only if
  either φ holds here ($n = 0$), or (when $n = 1 + n'$) we can do one α-step and then n' more α-steps to reach a state with φ

semantics of PDL

- PDL-formulas are multi-modal formulas over PROG, and so have to be interpreted in models over the index set PROG ...
- ... but arbitrary PROG-models don't do justice to the intended meaning of the program constructions
- we formalize the intuitive meaning by posing conditions on the transition relations!

preliminaries (1): identity, composition, union

- the identity relation Id is defined by

$$\mathrm{Id} = \{(x, y) \mid x = y\}$$

- the composition $R \circ S$ of relations R and S is defined by:

$$R \circ S = \{\, (x, z) \mid \exists\, y.\, Rxy \wedge Syz \,\}$$

- the union $R \cup S$ of relations R and S is defined by:

$$R \cup S = \{(x, y) \mid Rxy \vee Sxy\}$$

preliminaries (2): reflexive-transitive closure

- the *n*-fold composition R^n of a relation R is defined by:

$$R^0 = \mathrm{Id} \qquad\qquad R^{n+1} = R^n \circ R$$

- the reflexive-transitive closure R^* of R is defined by:

$$R^* = \bigcup_{n \geq 0} R^n$$

- if $x \; R^* \; y$, then, for some $n \geq 0$ and $x_0, \ldots, x_n$ we have

$$x = x_0 \; R \; x_1 \; R \cdots R \; x_n = y$$

- R^* is the smallest reflexive, transitive relation that contains R:

$$R^* = \bigcap \{ \, R' \mid R' \text{ is reflexive and transitive, and } R \subseteq R' \, \}$$

PDL-frames

- a $\textsc{Prog}$-frame $\mathcal{F} = (W, \{R_\alpha \mid \alpha \in \textsc{Prog}\})$ is a PDL-frame if:

$$
\begin{aligned}
R_{\alpha\beta} &= R_\alpha \circ R_\beta \\
R_{\alpha\cup\beta} &= R_\alpha \cup R_\beta \\
R_{\alpha^*} &= (R_\alpha)^*
\end{aligned}
$$

  for all $\alpha, \beta \in \textsc{Prog}$
- hence, as soon as the interpretation of the atomic programs is fixed, we know what the relations corresponding to all composed programs are

PDL-models

a model $\mathcal{M} = (W, \{R_\alpha \mid \alpha \in \textsc{Prog}\}, V)$ is a PDL-model if $(W, \{R_\alpha \mid \alpha \in \textsc{Prog}\})$ is a PDL-frame, and

$$R_{\varphi?} = \{\,(w, w) \mid \mathcal{M}, w \vDash \varphi\,\}$$

PDL-extension

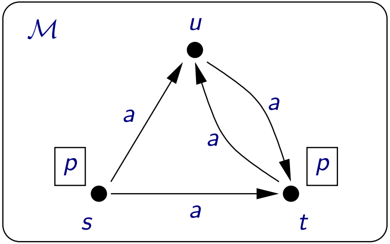
let $\mathcal{M} = (W, \{R_a \mid a \in \mathrm{A}\}, V)$ be an A-model.

the PDL-extension $\widehat{\mathcal{M}}$ of $\mathcal{M}$ is the $\textsc{Prog}$-model $\widehat{\mathcal{M}} = (W, \{\widehat{R}_\alpha \mid \alpha \in \textsc{Prog}\}, V)$, where $\widehat{R}_\alpha$ is inductively defined on the structure of α:

$$
\begin{aligned}
\widehat{R}_a &= R_a \\
\widehat{R}_{\alpha\beta} &= \widehat{R}_\alpha \circ \widehat{R}_\beta \\
\widehat{R}_{\alpha\cup\beta} &= \widehat{R}_\alpha \cup \widehat{R}_\beta \\
\widehat{R}_{\alpha^*} &= (\widehat{R}_\alpha)^* \\
\widehat{R}_{\varphi?} &= \{(x, x) \mid \mathcal{M}, x \vDash \varphi\}
\end{aligned}
$$

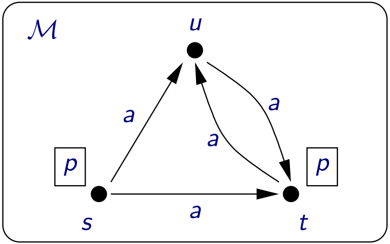
so $\widehat{\mathcal{M}}$ is a PDL-model , for all A-models $\mathcal{M}$

example 1

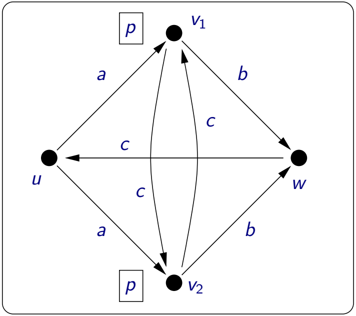


1. Show $\widehat{\mathcal{M}} \models \langle a^* \rangle [(aa)^*] p \wedge \langle a^* \rangle [(aa)^*] \neg p$

example 1



2. let $\alpha =$ if p then aa else a
   (a) does the formula $\langle\alpha\rangle p$ hold throughout $\widehat{\mathcal{M}}$?
   (b) and $[\alpha]p$?



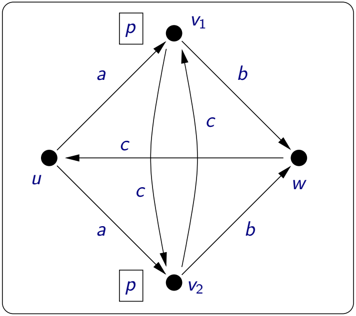
$\mathcal{M}$

(d) let $\widehat{\mathcal{M}}$ be the PDL-extension of $\mathcal{M}$.

compute the transition relations $\widehat{R}_\alpha$, $\widehat{R}_\beta$, $\widehat{R}_\gamma$ of $\widehat{\mathcal{M}}$ corresponding to the PDL-programs α, β, γ:

$$\alpha = bca \qquad \beta = \alpha \cup c \qquad \gamma = \beta^*$$

example 2



$\mathcal{M}$

(e) do we have $\widehat{\mathcal{M}} \models [\gamma]p \leftrightarrow p$?

deriving the truth definition of $\langle \text{while } \varphi \text{ do } \alpha \rangle \psi$

$$\text{while } \varphi \text{ do } \alpha = (\varphi? \,;\, \alpha)^* \,;\, \neg\varphi?$$

let $\mathcal{M} = (W, \{R_\alpha \mid \alpha \in \textsc{Prog}\}, V)$ be a PDL-model. then:

$$\begin{aligned}
&\mathcal{M}, x \vDash \langle \text{while } \varphi \text{ do } \alpha \rangle \psi \\
&\qquad\qquad \text{if and only if} \\
&\exists n \geq 0.\ \exists x_0, \ldots, x_n.\ (\ x_0 = x \\
&\qquad\qquad\qquad \&\ R_\alpha x_i x_{i+1}\ (0 \leq i < n) \\
&\qquad\qquad\qquad \&\ \mathcal{M}, x_i \vDash \varphi\ (0 \leq i < n) \\
&\qquad\qquad\qquad \&\ x_n \vDash \neg\varphi \wedge \psi\ )
\end{aligned}$$

PDL-formulas defining the class of PDL-frames

the following formulas are valid in all PDL-frames, and if they are valid in a frame, the frame is a PDL-frame.

$$\begin{aligned}
\langle\alpha\beta\rangle p &\leftrightarrow \langle\alpha\rangle\langle\beta\rangle p \\
\langle\alpha\cup\beta\rangle p &\leftrightarrow \langle\alpha\rangle p \vee \langle\beta\rangle p \\
\langle\alpha^*\rangle p &\leftrightarrow p \vee \langle\alpha\rangle\langle\alpha^*\rangle p \\
\langle p?\rangle q &\leftrightarrow p \wedge q \\
[\alpha^*]p &\leftrightarrow p \wedge [\alpha^*](p \rightarrow [\alpha]p)
\end{aligned}$$

the last formula is called the induction axiom, direction $\leftarrow$ reads

> *if p is true initially, and if, after any number of iterations of α, the truth of p is preserved by one more iteration of α, then p will be true after any number of iterations of α*

some relational algebra

some laws that can help you to determine relations corresponding to PDL-programs

$$
\begin{aligned}
\mathrm{Id} \circ R = R &= R \circ \mathrm{Id} \\
R \circ (S \cup T) &= (R \circ S) \cup (R \circ T) \\
(S \cup T) \circ R &= (S \circ R) \cup (T \circ R) \\
(R^*)^* &= R^* \\
(R \cup S)^* &= (R^* \circ S^*)^*
\end{aligned}
$$

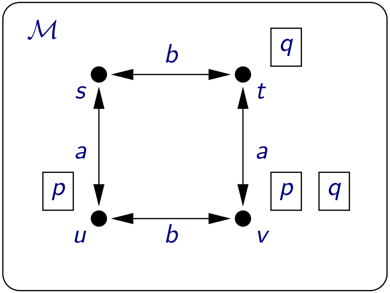
PDL's program constructors are safe for bisimulation

- in order to verify whether two PDL-models are bisimilar ... do we have to check bisimilarity of infinitely many relations ?
- no, it suffices to check bisimilarity of the relations interpreting the atomic programs!
- an n-ary relational operator O is called safe for bisimulation if E is a bisimulation for the relation $O(R_1, \ldots, R_n)$ whenever E is a bisimulation for the relations $R_1, \ldots, R_n$
- PDL's constructors are safe for bisimulation! hence we get:

$$Z : \mathcal{M} \leftrightarroweq \mathcal{M}' \implies Z : \widehat{\mathcal{M}} \leftrightarroweq \widehat{\mathcal{M}'}$$

- examples of operations not safe for bisimulation are converse $\_^{-1}$ and intersection $\cap$
- again: truth of PDL-formulas is preserved under bisimulations

example 3



$$\widehat{\mathcal{M}} \vDash p \leftrightarrow [(ab^* a)^*]p$$

example 4

find a PDL formula which expresses the following property of a state in a PDL-model:

p is alternately true and false along all execution paths of a from the current state (starting with p true)

example 4

find a PDL formula which expresses the following property of a state in a PDL-model:

p is alternately true and false along all execution paths of a from the current state (starting with p true)

the following two formulas both express this property (and are equivalent in all PDL-frames):

$$p \wedge [a^*]((p \to [a]\neg p) \wedge (\neg p \to [a]p))$$
$$[(aa)^*]p \wedge [a(aa)^*]\neg p$$

example 4

find a PDL formula which expresses the following property of a state in a PDL-model:

p is alternately true and false along all execution paths of a from the current state (starting with p true)

the following two formulas both express this property (and are equivalent in all PDL-frames):

$$p \wedge [a^*]((p \rightarrow [a]\neg p) \wedge (\neg p \rightarrow [a]p))$$

$$[(aa)^*]p \wedge [a(aa)^*]\neg p$$

a formula that characterizes the property *p is alternately true and false along all execution paths of a* is

$$(p \rightarrow [a]\neg p) \wedge (\neg p \rightarrow [a]p)$$

example 5

which of the two directions

$$[(\alpha \cup \beta)^*]p \leftrightarrow [\alpha^*]p \wedge [\beta^*]p$$

is valid in PDL?

example 6

which property is expressed by the formula (valid in all PDL-frames!):

$$\langle \text{while } p \text{ do } \alpha \rangle \top \;\leftrightarrow\; \langle \alpha^* \rangle \neg p \quad ?$$

example 6

which property is expressed by the formula (valid in all PDL-frames!):

$$\langle \text{while } p \text{ do } \alpha \rangle \top \;\leftrightarrow\; \langle \alpha^* \rangle \neg p \quad ?$$

this says that while p do α terminates if and only if it is possible, by repeated executions of α, to reach a state where $\neg p$ holds.

example 7

yet another PDL-validity:

$$[\beta]q \;\leftrightarrow\; (\neg p \wedge q) \vee (p \wedge [\alpha\beta]q)$$

where $\beta = \textsf{while } p \textsf{ do } \alpha$.

example 8

$$[\text{if } p \text{ then } \alpha \text{ else } \beta]q \;\leftrightarrow\; (p \wedge [\alpha]q) \vee (\neg p \wedge [\beta]q)$$

valid in all PDL-frames