advanced logic
2019 03 11
lecture 11

overview

- about program correctness
- propositional dynamic logic

overview

- about program correctness
- propositional dynamic logic

program correctness

correctness specification:

formal description about how a program is supposed to behave

program is correct:

its executions satisfy the specification

partial and total correctness

partial correctness:

if the program starts satisfying ϕ,

and if it halts,

then when it halts ψ is satisfied

total correctness:

it is partially correct,

and it terminates whenever started satisfying ϕ

approach to program correctness

we restrict attention to input-output behaviour

specification consists of

input condition ϕ and output condition ψ

recall: program for gcd

while $y \neq 0$ **do**
  begin
    $z := x \mod y$;
    $x := y$;
    $y := z$;
  end
return x

correctness for gcd

if the input variables x and y are c and d

the output value of x is the gcd of c and d

and the program halts

while programs

atomic instruction: assignment

$x := t$ with x a variable and t a term

sequential composition

$\alpha; \beta$

conditional

$\text{if}\, \phi\, \text{then}\, \alpha\, \text{else}\, \beta$

iteration: while

$\text{while}\, \phi\, \text{do}\, \alpha$

rules for Hoare logic

one rule for every program construct, for example for while:

$$\frac{\{\phi \wedge \sigma\}\alpha\{\phi\}}{\{\phi\}\text{while}\, \sigma\, \text{do}\, \alpha\{\phi \wedge \neg\sigma\}}$$

and a weakening rule:

$$\frac{\phi \to \phi' \qquad \{\phi'\}\alpha\{\psi'\} \qquad \psi' \to \psi}{\{\phi\}\alpha\{\psi\}}$$

illustration

the gcd program is $\text{while}\, \sigma\, \text{do}\, \alpha$ with σ is $y \neq 0$

precondition ϕ is $\neg(x = 0 \wedge y = 0) \wedge x = c \wedge y = d$

postcondition ψ is $x = gcd(c, d)$

invariant I is $\neg(x = 0 \wedge y = 0) \wedge gcd(x, y) = gcd(c, d)$

we have $\phi \to I$

we have $I \wedge \neg\sigma \to \psi$

overview

- about program correctness
- propositional dynamic logic

about propositional dynamic logic (PDL)

PDL is a formal system for reasoning about programs:

proving that a program meets its specification, comparing expressive power, ...

PDL is modal so dynamic, so suitable to model computation

PDL interprets programs as input-output relation
(abstracts away from program execution details)

more or less: programs are supposed to halt

program constructors such as composition are interpreted as operations on input-output relations

propositional dynamic logic (PDL): starting point

for every program α we have a modality $\langle\alpha\rangle$

$\langle\alpha\rangle\phi$ intuitively means

it is possible to execute α starting in the current state,

and halt (successfully) in a state satisfying ϕ

$[\alpha]\phi$ intuitively means

for all executions of α:
if α halts (successfully), then it halts in a state satisfying ϕ

ingredients of PDL

multi-modal logic

regular programs

mixed ingredients $[\alpha]\phi$, $\langle\alpha\rangle\phi$, $?\phi$

set Prog of PDL or regular programs: definition

atomic program

a from a set *A* of atomic programs

sequential composition

$\alpha;\beta$

non-deterministic choice

$\alpha \cup \beta$

iteration

α^*

test

ϕ? with ϕ a formula, so depends on the grammar for formulas

PDL programs: intuitive meaning

a

atomic, indecomposable, step

$\phi?$

if ϕ then skip else abort, that is,

if ϕ holds then continue without changing state,

if ϕ does not hold then block without halting

$\alpha; \beta$

do α, then do β

$\alpha \cup \beta$

nondeterministivally choose α or β and execute it

α^*

nondeterministically choose $n \geq 0$ and execute α n times

non-determinism

we have non-determinism due to choice $\cup$

we have non-determinism due to iteration α^*

a trace may not be uniquely determined by its start state

nondeterminism is useful to model situations where we may know the range of possibilities

often a deterministic choice is forced for example for if-then-else

PDL formulas: definition

atomic formula

p from a set *Var* of atomic propositions

true and false

$\top$ and $\bot$

negation

$\neg\phi$

conjunction

$\phi \wedge \psi$

diamond

$\langle\alpha\rangle\phi$, with α a program, so depends on the grammar for programs

PDL formulas: examples

$[\alpha \cup \beta]\phi$

always if we execute α or β we arrive at a state where ϕ holds

$\langle(\alpha\beta)^*\rangle\phi$

there is a sequence of alternating executions of α and β bringing us to a state where ϕ holds

$\langle\alpha^*\rangle\phi \leftrightarrow \phi \vee \langle\alpha; \alpha^*\rangle\phi$

ϕ holds after a finite number ($n \geq 0$) of α steps

if and only if

either ϕ holds here ($n = 0$), or ($n > 0$) we can do an α step and then more α steps to reach a state where ϕ holds

PDL formulas: more examples

$[\alpha](\phi \wedge \psi) \quad \leftrightarrow \quad [\alpha]\phi \wedge [\alpha]\psi$ (seems a tautology)

$[\alpha;\beta]\phi \quad \leftrightarrow \quad [\alpha][\beta]\phi$ (seems a tautology)

$[\alpha]p \quad \leftrightarrow \quad [\beta]p$ (gives an equivalence between α and β)

mutual dependency: examples

$[p?]p$

if $p?$ halts then in a state satisfying p with p an atomic proposition

$\langle p? \rangle p$

it is possible to execute $p?$ and halt in a state where p holds

$[\alpha]\bot$

α never terminates

$[\alpha]\top$

is always true

$\top?$

is skip

$\bot?$

is fail (? unsuccessful halt)

towards a semantics for PDL formulas

we obtain the semantics as an instance of multi-modal logic

in particular:
$\mathcal{M}, s \models \langle \alpha \rangle \phi$ iff there is s' such that $(s, s') \in R_\alpha$ and $\mathcal{M}, s' \models \phi$

however:

an arbitrary model does not respect the intended meaning of the programs

therefore we will impose conditions on the relations R_α

example

$W = \{u, v, w\}$

$R_a = \{(u, v), (u, w), (v, w), (w, v)\}$

$V(p) = \{u, v\}$

we have $u \models \langle a \rangle \neg p \wedge \langle a \rangle p$

we have $v \models [a]\neg p$

we have $w \models [a]p$

in every world (state) we have $\langle a^* \rangle [(aa)^*]p \wedge \langle a^* \rangle [(aa)^*]\neg p$

example

$W = \{s, t, u, v\}$

$R_a = \{(t, v), (v, t), (s, u), (u, s)\}$

$R_b = \{(u, v), (v, u), (s, t), (t, s)\}$

$V(p) = \{u, v\}$

$V(q) = \{t, v\}$

we have $p \leftrightarrow [(ab^*a)^*]p$

we have $q \leftrightarrow [(ba^*b)^*]q$

intuitive requirements for a PDL model

consider $a; b$ and $R_{a;b}$

consider $a \cup b$ and $R_{a \cup b}$

consider a^* and R_{a^*}

this suggests to start from all the R_a with $a \in A$ an atomic program

but what to do with $R_{\phi?}$?

PDL frame: definition

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

$R_{\alpha\beta} = R_\alpha; R_\beta$, and

$R_{\alpha \cup \beta} = R_\alpha \cup R_\beta$, and

$R_{\alpha^*} = (R_\alpha)^*$

so if we know all R_a then we know enough!

what are the definitions on the relations?

definitions on relations

the composition of R and S: $R; S = \{(x, z) \mid \exists y : Rxy \wedge Syz\}$

the union of R and S: $R \cup S = \{(x, y) \mid Rxy \vee Sxy\}$

the identity relation: $\text{Id} = \{(x, x)\}$

the n-fold composition of R: $R^0 = \text{Id}$ and $R^{n+1} = R^n; R$

the reflexive-transitive closure of R: $R^* = \bigcup_{n \geq o} R^n$

note: if xR^*y, then there exists $n \geq 0$ and there exist $x_1, \ldots, x_{n-1}$ such that $x = x_0 R x_1 R \ldots R x_n = y$

note: R^* is the smallest reflexive and transitive relation containing R

PDL model: definition

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

$(W, \{R_\alpha \mid \alpha \in \text{Prog}\}$ is a PDL-frame, and

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

PDL extension: definition

we can get a PDL model as the extension of a model over labels A

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

Its PDL-extension is defined as $\hat{\mathcal{M}} = (W, \{\hat{R}_\alpha \mid \alpha \in \text{Prog}\}, V)$ with

$\hat{R}_a = R_a$

$\hat{R}_{\alpha;\beta} = \hat{R}_\alpha; \hat{R}_\beta$

$\hat{R}_{\alpha\cup\beta} = \hat{R}_\alpha \cup \hat{R}_\beta$

$\hat{R}_{\alpha^*} = (R_\alpha)^*$

$\hat{R}_{\phi?} = \{(x, x) \mid \mathcal{M}, x \vDash \phi\}$