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 $\phi$, and if it halts, then when it halts $\psi$ is satisfied

total correctness:
it is partially correct, and it terminates whenever started satisfying $\phi$
approach to program correctness

we restrict attention to input-output behaviour

specification consists of

input condition \( \phi \) and output condition \( \psi \)
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
if \( \phi \) then \( \alpha \) else \( \beta \)

iteration: while
while \( \phi \) do \( \alpha \)
rules for Hoare logic

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

\[
\frac{\{\phi \land \sigma\}\alpha\{\phi\}}{\{\phi\}\text{while }\sigma\text{ do }\alpha\{\phi \land \neg \sigma\}}
\]

and a weakening rule:

\[
\frac{\phi \rightarrow \phi'}{\{\phi\}\alpha\{\psi\}} \quad \frac{\{\phi'\}\alpha\{\psi'\}}{\psi' \rightarrow \psi} \quad \frac{}{\{\phi\}\alpha\{\psi\}}
\]
the gcd program is while \( \sigma \) do \( \alpha \) with \( \sigma \) is \( y \neq 0 \)

precondition \( \phi \) is \( \neg(x = 0 \land y = 0) \land x = c \land y = d \)

postcondition \( \psi \) is \( x = gcd(c, d) \)

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

we have \( \phi \rightarrow I \)

we have \( I \land \neg\sigma \rightarrow \psi \)
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
for every program $\alpha$ we have a modality $\langle \alpha \rangle$

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

it is possible to execute $\alpha$ starting in the current state,

and halt (successfully) in a state satisfying $\phi$

$[\alpha] \phi$ intuitively means

for all executions of $\alpha$:
if $\alpha$ halts (successfully), then it halts in a state satisfying $\phi$
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

\(\alpha^*\)

test

\(\phi?\) with \(\phi\) a formula, so depends on the grammar for formulas
PDL programs: intuitive meaning

atomic, indecomposable, step

\[ \phi \]

if \( \phi \) then skip else abort, that is,

if \( \phi \) holds then continue without changing state,

if \( \phi \) does not hold then block without halting

\[ \alpha; \beta \]

do \( \alpha \), then do \( \beta \)

\[ \alpha \cup \beta \]

nondeterministically choose \( \alpha \) or \( \beta \) and execute it

\[ \alpha^* \]

nondeterministically choose \( n \geq 0 \) and execute \( \alpha \) \( n \) times
non-determinism

we have non-determinism due to choice \( \cup \)

we have non-determinism due to iteration \( \alpha^* \)

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 $\text{Var}$ of atomic propositions

true and false

$\top$ and $\bot$

negation

$\neg \phi$

conjunction

$\phi \land \psi$

diamond

$\langle \alpha \rangle \phi$, with $\alpha$ a program, so depends on the grammar for programs
PDL formulas: examples

\[ [\alpha \cup \beta] \phi \]
always if we execute \( \alpha \) or \( \beta \) we arrive at a state where \( \phi \) holds

\[ \langle (\alpha \beta)^* \rangle \phi \]
there is a sequence of alternating executions of \( \alpha \) and \( \beta \) bringing us to a state where \( \phi \) holds

\[ \langle \alpha^* \rangle \phi \leftrightarrow \phi \lor \langle \alpha ; \alpha^* \rangle \phi \]
\( \phi \) holds after a finite number \( (n \geq 0) \) of \( \alpha \) steps
if and only if

either \( \phi \) holds here \( (n = 0) \), or \( (n > 0) \) we can do an \( \alpha \) step and then more \( \alpha \) steps to reach a state where \( \phi \) holds
PDL formulas: more examples

\[ [\alpha](\phi \land \psi) \leftrightarrow [\alpha]\phi \land [\alpha]\psi \]  
(seems a tautology)

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

\[ [\alpha]\rho \leftrightarrow [\beta]\rho \]  
(gives an equivalence between \( \alpha \) and \( \beta \))
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

\([\alpha]\bot\)
\(\alpha\) never terminates

\([\alpha]\top\)
is always true

\(\top?\)
is skip

\(\bot?\)
is fail (\(\bot\) unsuccessful halt)
towards a semantics for PDL formulas

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

in particular:

\[ M, s \models \langle \alpha \rangle \phi \text{ iff there is } s' \text{ such that } (s, s') \in R_\alpha \text{ and } M, s' \models \phi \]

however:

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

due to imposing conditions on the relations \( R_\alpha \).
example

\[ W = \{ u, v, w \} \]

\[ R_a = \{(u, v), (u, w), (v, w), (w, v)\} \]

\[ V(p) = \{ u, v \} \]

we have \( u \models \langle a \rangle \neg p \land \langle a \rangle p \)

we have \( v \models [a] \neg p \)

we have \( w \models [a] p \)

in every world (state) we have \( \langle a^* \rangle [(aa)^*] p \land \langle a^* \rangle [(aa)^*] \neg p \)
\[ W = \{s, t, u, v\} \]

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

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

\[ V(p) = \{u, v\} \]

\[ V(q) = \{t, v\} \]

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 $R_\phi$?
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 $S$: $R; S = \{(x, z) \mid \exists y : Rxy \land Syz\}$

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

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

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

the reflexive-transitive closure of $R$: $R^* = \bigcup_{n \geq 0} R^n$

note: if $x R^* y$, 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 $R$
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

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

$\hat{R}_a = R_a$

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

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

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

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