LOGICS OF IMPERATIVE PROGRAMS

Daniel Leivant

Indiana University

AIMS AND OUTLINE

- A coherent story with math beauty and extensive applications
- The root: Propositional dynamic logic (for regular programs)
- Hoare logics for partial and total correctness
- Impact on programming methodology
- Modal vs. explicit reasoning about programs
- Completeness issues

NOT COVERED

- Program behaviour logics (e.g. temporal logics)
- Logics for applicative (functional, prolog) programs
- Most non-basic programming constructs (concurrency, OO ...)

LECTURE 1:
PROPOSITIONAL DYNAMIC LOGIC

Transition systems

A transition system:

- Set S ("$states$" $s \in S$)
- Collection of binary relations ("$actions$" $a \in A$) over S.

$EXAMPLE$

Supermarket.
- "state" $=$ possible inventory
- "action" $=$ change in inventory
  (e.g. deliveries, purchases, spoilage, shop-lifting)

Note: some state/action pairs are impossible

Mathematical examples

A Turing machine.

- "state" $=$ configuration
- "action" $=$ instruction (state-change)

Note: In a deterministic machine only one action for state
But even in ND machine pair state/action do not repeat

All Turing machines (single tape, fixed alphabet Σ)

- "states" $=$ Σ^* (=possible tape contents)
- "action" $=$ particular machine (input/output)

Note: In ND machine action may repeat for same state

Environments (assignments)
Take vocab $V = (\mathbf{0}; \mathbf{s}, +, \times; <)$

- "state" $=$ environment (mapping Variables $\mapsto \mathbb{N}$)
- "action" $=$ update $\vec{x} := \vec{\mathbf{t}}$

Compound actions

Generating compound actions α

- composition
- union
- intersdection
- reflexive-transitive closure (Kleene's star) α^*
- reversal
- Complement

We will focus on composition, union, and Kleene's *

Regular programs

More formal definition:

- Countable set A of atomic-action identifiers
- Transition system (over A): $(S, [\![\cdot]\!]_A)$
  - Set S
  - $[\![\cdot]\!]_A$ maps $\mathbf{a} \in A$ to binary relation over S

Regular programs:

$$\begin{array}{rcl} A & \ni & \mathbf{a} \\ RProg(A) & \ni & \boldsymbol{\alpha}, \boldsymbol{\beta} \ ::= \ \mathbf{a} \mid \boldsymbol{\alpha}; \boldsymbol{\beta} \mid \boldsymbol{\alpha} \cup \boldsymbol{\beta} \mid \boldsymbol{\alpha}^* \end{array}$$

Semantic interpretation In transition system $(S, [\![\cdot]\!]_A)$:

$$\begin{array}{rcll} [\![\alpha; \beta]\!] & = & [\![\alpha]\!] \circ [\![\beta]\!] & \text{(composition of relations)} \\ [\![\alpha \cup \beta]\!] & = & [\![\alpha]\!] \cup [\![\beta]\!] & \text{(union of relations)} \\ [\![\alpha^*]\!] & = & [\![\alpha]\!]^* & \text{(reflexive-transitive closure)} \end{array}$$

Arrow notation

$s \xrightarrow{\alpha} t$ for $(s,t) \in [\![\alpha]\!]_A$

When transition system $\mathcal{S}$ relevant: $s \xrightarrow{\alpha}{}_{\mathcal{S}} t$

Thus:

$$
\begin{array}{lll}
s \xrightarrow{\alpha;\beta} t & \textbf{IFF} & s \xrightarrow{\alpha} u \quad \textbf{and} \quad u \xrightarrow{\beta} t \quad \textbf{for some} \quad u \in S \\
s \xrightarrow{\alpha\cup\beta} t & \textbf{IFF} & s \xrightarrow{\alpha} t \quad \textbf{or} \quad s \xrightarrow{\beta} t \\
s \xrightarrow{\alpha^*} t & \textbf{IFF} & s \xrightarrow{\alpha} s_1 \xrightarrow{\alpha} s_2 \xrightarrow{\alpha} \cdots \xrightarrow{\alpha} s_k \xrightarrow{\alpha} t \\
 & & \quad \textbf{some } k \geq 0 \textbf{ and } s_1 \ldots s_k \in S
\end{array}
$$

Propositional dynamic structures

- Propositional structure: Prop Ids $\rightarrow \{0,1\}$
- Here: Prop Ids P, (atomic) Action Ids A
- Propositional dynamic (PD) structure (for vocab P, A):
  - $\mathcal{K} = (S, [\![\cdot]\!]_A, [\![\cdot]\!]_P)$
  - $(S, [\![\cdot]\!]_A)$ a transition system for A
  - $[\![\cdot]\!]_P$: $S \rightarrow$ Prop Id $\rightarrow \{0,1\}$
  - I.e. $[\![\cdot]\!]_P(a)$ is a propositional structure for each $a \in A$.
  - Alternative notation: $[\![\cdot]\!]_a$ for $[\![\cdot]\!]_P(a)$
- Representation: labeled graphs

PDL formulas

Features:

- Formulas true or false in given state
- Special constructs to refer to transitions
- For program α and formula φ,
  - $[\boldsymbol{\alpha}]\boldsymbol{\varphi}$ for "φ in every state reachable via α."
  - $\langle\boldsymbol{\alpha}\rangle\boldsymbol{\varphi}$ for "φ in some state reachable via α."

BNF definition of PDL formulas:

$$\begin{array}{rcll} P & \ni & \mathbf{p} & \\ RPrg(A) & \ni & \boldsymbol{\alpha} & \\ PDFml(P,A) & \ni & \boldsymbol{\varphi},\boldsymbol{\psi} & ::= \mathbf{p} \mid \neg\boldsymbol{\varphi} \mid \boldsymbol{\varphi}\wedge\boldsymbol{\psi} \mid \boldsymbol{\varphi}\vee\boldsymbol{\psi} \mid \boldsymbol{\varphi}\rightarrow\boldsymbol{\psi} \mid \\ & & & \quad [\boldsymbol{\alpha}]\boldsymbol{\varphi} \mid \langle\boldsymbol{\alpha}\rangle\boldsymbol{\varphi} \end{array}$$

Semantics of PDL formulas

- Given PD structure $\mathcal{K}$ and state s,
- $\mathcal{K}, s \models \varphi$ for "φ true in $\mathcal{K}$ at s"
- Abbreviate $\mathcal{K}, s \models \varphi$ to $s \models \varphi$

$$
\begin{array}{rcl}
s \models \mathbf{p} & \textbf{IFF} & [\![\mathbf{p}]\!](s) = 1 \\
s \models \neg\varphi & \textbf{IFF} & s \not\models \varphi \\
s \models \varphi \wedge \psi & \textbf{IFF} & s \models \varphi \textbf{ and } s \models \psi \\
s \models \varphi \vee \psi & \textbf{IFF} & s \models \varphi \textbf{ or } s \models \psi \\
s \models \varphi \rightarrow \psi & \textbf{IFF} & s \models \varphi \textbf{ implies } s \models \psi \\
s \models [\boldsymbol{\alpha}]\varphi & \textbf{IFF} & t \models \varphi \textbf{ for all } t \textbf{ such that } s \xrightarrow{\alpha} t \\
s \models \langle \boldsymbol{\alpha} \rangle \varphi & \textbf{IFF} & t \models \varphi \textbf{ for some } t \textbf{ such that } s \xrightarrow{\alpha} t
\end{array}
$$

PDL truths

- Duality: $s \models \neg\langle\boldsymbol{\alpha}\rangle\boldsymbol{\varphi}$ iff $s \models [\boldsymbol{\alpha}]\neg\boldsymbol{\varphi}$
- One defined in terms of the other
- Analogous to conjunction/disjunction, $\forall/\exists$

- As for first order logic, define
  - $\mathcal{K}, \eta \models \boldsymbol{\varphi}$
  - $\mathcal{K} \models \boldsymbol{\varphi}$ (φ is true in $\mathcal{K}$)
  - $\models \boldsymbol{\varphi}$ (φ is valid)

Programs with tests

- Box and diamond inject programs into formulas.
- Inject formulas in programs?
- Yes! Tests (guards)

Revised definition:
formulas and programs generated in tandem

$$
\begin{array}{rcll}
A & \ni & \mathbf{a} & \\
RProg(A) & \ni & \boldsymbol{\alpha}, \boldsymbol{\beta} & ::= \mathbf{a} \mid \boldsymbol{\alpha}; \boldsymbol{\beta} \mid \boldsymbol{\alpha} \cup \boldsymbol{\beta} \mid \boldsymbol{\alpha}^* \mid ?\varphi \\
P & \ni & \mathbf{p} & \\
PDFml & \ni & \boldsymbol{\varphi}, \boldsymbol{\psi} & ::= \mathbf{p} \mid \neg \boldsymbol{\varphi} \mid \boldsymbol{\varphi} \wedge \boldsymbol{\psi} \mid \boldsymbol{\varphi} \vee \boldsymbol{\psi} \mid \boldsymbol{\varphi} \rightarrow \boldsymbol{\psi} \mid \\
 & & & \quad [\boldsymbol{\alpha}] \boldsymbol{\varphi} \mid \langle \boldsymbol{\alpha} \rangle \boldsymbol{\varphi}
\end{array}
$$

Need additional clause for semantics:

$$[\![?\varphi]\!] = \{(s, s) \mid s \models \varphi\}$$

I.e., $s \xrightarrow{?\varphi} s'$ iff $s \models \varphi$ and $s' = s$

Expressive power of formulas

- $s \models [\boldsymbol{\alpha}]\bot$: no (terminating) execution of $\boldsymbol{\alpha}$ starting at s
- "Exsists (terminating) execution of $\boldsymbol{\alpha}$ starting at current state."
- "Deterministic program $\boldsymbol{\alpha}$ can be iterated any number of times."
- "Exists $n \geq 0$ s.t. iterating deterministic program $\boldsymbol{\alpha}$ n times leads to a state where $\boldsymbol{\beta}$ has no terminating execution."

However, not all properties of PD structures are expressed.

- $\mathbf{p}$ is true after every iterated execution of $\mathbf{a}$
  a certain number of times,
  followed by an iteration of $\mathbf{b}$
  the same number of times.
- $\mathbf{p}$ is true after iterated execution of α
  a prime number of times.

Guarded iterative programs

- Tests are used as guards
- Guarded branching: `if` χ `then` α `else` β
- Guarded iteration: `while` χ `do` α

$GIP(A, P)$ = guarded iterative programs over A and P is generated by:

$$
\begin{array}{rcl}
A & \ni & \mathbf{a} \\
P & \ni & \mathbf{p} \\
GIP(A,P) & \ni & \boldsymbol{\alpha}, \boldsymbol{\beta} \ ::= \ \mathbf{a} \mid \mathtt{skip} \mid \mathtt{abort} \mid \boldsymbol{\alpha}; \boldsymbol{\beta} \mid \\
 & & \quad \mathtt{if}\ \mathbf{p}\ \mathtt{then}\ \boldsymbol{\alpha}\ \mathtt{else}\ \boldsymbol{\beta} \mid \\
 & & \quad \mathtt{while}\ \mathbf{p}\ \mathtt{do}\ \boldsymbol{\alpha}
\end{array}
$$

Note: The definition above is for propositional identifiers as guards.
An lternative is to allow all formulas, as for regular programs.

Semantics of GIPs

- $\xrightarrow{\textbf{skip}} =_S$
- $\xrightarrow{\textbf{abort}} = \emptyset$
- $s \xrightarrow{\alpha;\beta} t =_{\mathrm{df}} s \xrightarrow{\alpha} u$ and $u \xrightarrow{\beta} t$ for some state u
- for $\boldsymbol{\gamma} =$ `if` $\boldsymbol{\chi}$ `then` $\boldsymbol{\alpha}$ `else` $\boldsymbol{\beta}$

  $s \xrightarrow{\gamma} t$ iff $s \models \boldsymbol{\chi}$ and $s \xrightarrow{\alpha} t$, or $s \not\models \boldsymbol{\chi}$ and $s \xrightarrow{\beta} t$
- for $\boldsymbol{\gamma} =$ `while` $\boldsymbol{\chi}$ `do` $\boldsymbol{\alpha}$

  $s \xrightarrow{\gamma} t$ IFF

  exist $u_1 \ldots u_k \in S$ s.t.

  $s = u_1$, $u_i \xrightarrow{\alpha} u_{i+1}$ and $u_i \models \boldsymbol{\chi}$ $(i < k)$, and $u_k = t$.

Defining constructs in terms of others

- skip defined by $?\top$
- abort
- `if` χ `then` α `else` β
  $(?\chi; \alpha) \cup (?\neg\chi; \beta)$
- `while` $\mathbf{p}$ `do` α
  $(?\mathbf{p}\,;\, \alpha) \cup (?\neg\mathbf{p}\,;\, \beta)$
- Can $\mathbf{a}^*$ be defined by a guarded iterative program?

A deductive calculus for PDL

I Axiom-templates and rules for programs in general:

Box Distribution: $\Rightarrow\ [\alpha](\varphi \to \psi) \to ([\alpha]\varphi \to [\alpha]\psi)$

Generalization: from $\Rightarrow\ \varphi$ (with no assumption in the proof line)
derive $\Rightarrow\ [\alpha]\varphi$

II Axiom-templates and rules for program constructs:

Composition: $\Rightarrow\ [\alpha;\beta]\varphi \leftrightarrow [\alpha][\beta]\varphi$

Branching: $\Rightarrow\ [\alpha \cup \beta]\varphi \leftrightarrow [\alpha]\varphi \wedge [\beta]\varphi$

Schema: $\Rightarrow\ [\alpha^*]\varphi \leftrightarrow \varphi \wedge [\alpha][\alpha^*]\varphi$

Induction Rule: from $\Rightarrow\ \varphi \to [\alpha]\varphi$
derive $\Rightarrow\ \varphi \to [\alpha^*]\varphi$

Test: $\Rightarrow\ [?\chi]\varphi \leftrightarrow (\chi \to \varphi)$

Soundness of the deductive calculus

Soundness Theorem for PDL. Every provable formula provable in PDL is valid: if $\vdash_{PDL} \varphi$ then $\models \varphi$

Proof by structural induction on derivations.

Some useful templates

- *Box Distribution Rule:*
  From $\Rightarrow\ \varphi \to \psi$ *derive* $\Rightarrow\ [\alpha]\varphi \to [\alpha]\psi$*:*

  From $\Rightarrow\ \varphi \to \psi$*:*
  $\Rightarrow\ [\alpha](\varphi \to \psi)$ *(Generalization)*
  But $\Rightarrow\ [\alpha](\varphi \to \psi) \to ([\alpha]\varphi \to [\alpha]\psi)$ *(Box Distribution)*
  so $\Rightarrow\ [\alpha]\varphi \to [\alpha]\psi$

- *Conjunction Distribution Template:*
  $[\alpha](\varphi \wedge \psi) \leftrightarrow ([\alpha]\varphi \wedge [\alpha]\psi)$

  – Forward implication:
    $\vdash \varphi \wedge \psi \to \varphi$, *so*
    $\vdash [\alpha](\varphi \wedge \psi) \to [\alpha]\varphi$, *(Box Distribution)*
    Similarly,
    $\vdash [\alpha](\varphi \wedge \psi) \to [\alpha]\psi$,
    so $\vdash [\alpha](\varphi \wedge \psi) \to ([\alpha]\varphi \wedge [\alpha]\psi)$

  – Backward implication: $\vdash \varphi \to (\psi \to \varphi \wedge \psi)$
    so $\vdash [\alpha]\varphi \to [\alpha](\psi \to \varphi \wedge \psi)$ *(Box Distribution).*
    But $\vdash [\alpha](\psi \to \varphi \wedge \psi) \ \to\ ([\alpha]\psi \to [\alpha](\varphi \wedge \psi))$ *(Box Distribution)*
    so $\vdash [\alpha]\varphi \to ([\alpha]\psi \to [\alpha](\varphi \wedge \psi))$,
    hence $\vdash ([\alpha]\varphi \wedge [\alpha]\psi) \ \to\ [\alpha](\varphi \wedge \psi)$

More interesting examples of proofs

- $[\alpha^*]\psi \to [\alpha]\psi$:

  $[\alpha^*]\psi \to \psi$ (Iteration)
  so $[\alpha][\alpha^*]\psi \to [\alpha]\psi$ (Box Distribution).
  Also, $[\alpha^*]\psi \to [\alpha][\alpha^*]\psi$ (Iteration)
  Combine: $[\alpha^*]\psi \to [\alpha]\psi$

- $[\alpha^*]\psi \to [\alpha^*][\alpha^*]\psi$:

  Let φ be $[\alpha^*]\psi$
  Then $\varphi \to [\alpha]\varphi$ (Iteration)
  so $\varphi \to [\alpha^*]\varphi$ (Induction Rule)

- $[\alpha^*]\psi \to [\alpha^*][\alpha]\psi$:

  $[\alpha^*]\psi \to [\alpha]\psi$ (Above)
  so $[\alpha^*][\alpha^*]\psi \to [\alpha^*][\alpha]\psi$ (Box Distribution)
  Combining with above, $[\alpha^*]\psi \to [\alpha^*][\alpha]\psi$.

Independent fun

Show that the following formulas are provable in PDL.

- $[?[\alpha]\bot;\alpha]\varphi$
- $[\alpha]p \wedge \langle\alpha\rangle q \rightarrow \langle\alpha\rangle p$
- $[\alpha]p \wedge \langle\alpha \cup \beta\rangle\neg p \rightarrow \langle\beta\rangle\top$
- $[(\alpha \cup \beta)^*]\varphi \rightarrow [\alpha^*][\beta^*]\varphi$
- $[(\alpha \cup \beta)^*]\varphi \rightarrow [\alpha][\beta^*]\varphi$

PDL for GIPs

Skip: $\Rightarrow [\mathtt{skip}]\varphi \leftrightarrow \varphi$

Abort $\Rightarrow [\mathtt{abort}]\varphi$

Guarded Branching: $\Rightarrow [\mathtt{if}\ \chi\ \mathtt{then}\ \alpha\ \mathtt{else}\ \beta]\varphi \leftrightarrow (\chi \wedge [\alpha]\varphi) \vee (\neg\chi \wedge [\beta]\varphi)$

Guarded Iteration: $\Rightarrow [\mathit{while}]\varphi \leftrightarrow (\neg\chi \wedge \varphi) \vee (\chi \wedge [\alpha][\mathit{while}]\,\varphi)$,
where while abbrreviates `while` χ `do` α

Guarded Induction Rule: from $\Rightarrow (\chi \wedge \varphi) \rightarrow [\alpha]\varphi$
infer $\Rightarrow \varphi \rightarrow [\mathit{while}](\varphi \wedge \neg\chi)$