Semantyka i weryfikacja programów

Bartosz Klin

(slajdy Andrzeja Tarleckiego)

Instytut Informatyki
Wydział Matematyki, Informatyki i Mechaniki
Uniwersytet Warszawski

http://www.mimuw.edu.pl/~klin pok. 5680
klin@mimuw.edu.pl

Strona tego wykładu:

http://www.mimuw.edu.pl/~klin/sem19-20.html

Program Semantics & Verification

Bartosz Klin

(slides courtesy of Andrzej Tarlecki)

Institute of Informatics
Faculty of Mathematics, Informatics and Mechanics
University of Warsaw

`http://www.mimuw.edu.pl/~klin` office: 5680

`klin@mimuw.edu.pl`

This course: `http://www.mimuw.edu.pl/~klin/sem19-20.html`

Natural semantics

big-step operational semantics

Overall idea:

- define *configurations*: $\gamma \in \Gamma$
- indicate which of them are *terminal*: $\mathrm{T} \subseteq \Gamma$
- instead of computations, consider (define) *transitions* directly *to final configurations* that are reached by computations: $\leadsto \subseteq \Gamma \times \mathrm{T}$

Informally:

- if $\gamma_0 \Rightarrow \gamma_1 \Rightarrow \cdots \Rightarrow \gamma_n$, $\gamma_n \in \mathrm{T}$, then $\gamma_0 \leadsto \gamma_n$
- if $\gamma_0 \Rightarrow \gamma_1 \Rightarrow \cdots \Rightarrow \gamma_n$, $\gamma_n \notin \mathrm{T}$ and $\gamma_n \not\Rightarrow$, then $\gamma_0 \not\leadsto$
- if $\gamma_0 \Rightarrow \gamma_1 \Rightarrow \cdots$ then $\gamma_0 \not\leadsto$

Tiny: natural semantics

$$\langle x := e, s\rangle \rightsquigarrow s[x \mapsto (\mathcal{E}[\![e]\!]\ s)]$$

$$\langle \mathbf{skip}, s\rangle \rightsquigarrow s$$

$$\frac{\langle S_1, s\rangle \rightsquigarrow s' \qquad \langle S_2, s'\rangle \rightsquigarrow s''}{\langle S_1; S_2, s\rangle \rightsquigarrow s''}$$

$$\frac{\langle S_1, s\rangle \rightsquigarrow s'}{\langle \mathbf{if}\ b\ \mathbf{then}\ S_1\ \mathbf{else}\ S_2, s\rangle \rightsquigarrow s'} \text{ if } \mathcal{B}[\![b]\!]\ s = \mathbf{tt}$$

$$\frac{\langle S_2, s\rangle \rightsquigarrow s'}{\langle \mathbf{if}\ b\ \mathbf{then}\ S_1\ \mathbf{else}\ S_2, s\rangle \rightsquigarrow s'} \text{ if } \mathcal{B}[\![b]\!]\ s = \mathbf{ff}$$

$$\frac{\langle S, s\rangle \rightsquigarrow s' \qquad \langle \mathbf{while}\ b\ \mathbf{do}\ S, s'\rangle \rightsquigarrow s''}{\langle \mathbf{while}\ b\ \mathbf{do}\ S, s\rangle \rightsquigarrow s''} \text{ if } \mathcal{B}[\![b]\!]\ s = \mathbf{tt}$$

$$\langle \mathbf{while}\ b\ \mathbf{do}\ S, s\rangle \rightsquigarrow s \quad \text{if } \mathcal{B}[\![b]\!]\ s = \mathbf{ff}$$

Configurations:

$$\Gamma = (\mathbf{Stmt} \times \mathbf{State}) \cup \mathbf{State}$$

Terminal configurations: as before

$$\mathrm{T} = \mathbf{State}$$

Transitions: as given here

How to read this?

"set-theoretically"

As before:

$\rightsquigarrow \subseteq \Gamma \times \mathrm{T}$ is the least relation such that

– $\langle x := e, s\rangle \rightsquigarrow s[x \mapsto (\mathcal{E}[\![e]\!]\ s)]$, for all $x \in \mathbf{Var}$, $e \in \mathbf{Exp}$, $s \in \mathbf{State}$

– ...

– $\langle S_1; S_2, s\rangle \rightsquigarrow s''$ if $\langle S_1, s\rangle \rightsquigarrow s'$ and $\langle S_2, s'\rangle \rightsquigarrow s''$, for all $S_1, S_2 \in \mathbf{Stmt}$, $s, s', s'' \in \mathbf{State}$

– $\langle \mathbf{if}\ b\ \mathbf{then}\ S_1\ \mathbf{else}\ S_2, s\rangle \rightsquigarrow s'$ if $\langle S_1, s\rangle \rightsquigarrow s'$ and $\mathcal{B}[\![b]\!]\ s = \mathbf{tt}$, for all $b \in \mathbf{BExp}$, $S_1, S_2 \in \mathbf{Stmt}$, $s, s' \in \mathbf{State}$

– ...

How to read this?

"proof-theoretically"

We give

- axioms, like $\langle x := e, s\rangle \rightsquigarrow s[x \mapsto (\mathcal{E}[\![e]\!]\ s)]$, and
- rules, like $\dfrac{\langle S_1, s\rangle \rightsquigarrow s' \qquad \langle S_2, s'\rangle \rightsquigarrow s''}{\langle S_1; S_2, s\rangle \rightsquigarrow s''}$

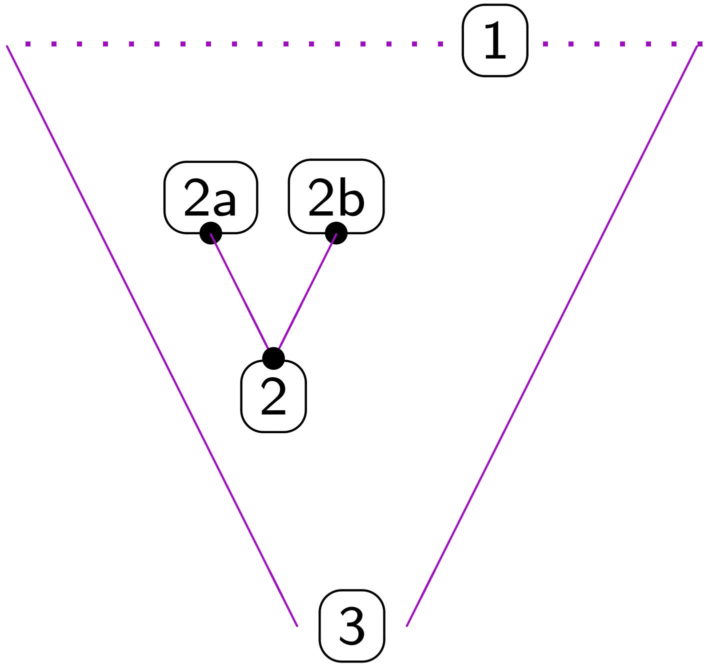
to *derive* (or better: *prove*) *judgements* of the form

$$\boxed{\langle S, s\rangle \rightsquigarrow s'}$$

Actually: we give axiom and rule *schemata*, which are *generic* in the choice of elements to be substituted for meta-variables used ($x \in \mathbf{Var}$, $e \in \mathbf{Exp}$, $s, s', s'' \in \mathbf{State}$, $S_1, S_2 \in \mathbf{Stmt}$, etc).

Proofs/derivations

Finite *proof tree* (or *derivation tree*):



- leaves: labelled by axioms, e.g.
  $\boxed{1} : \langle x := e, s\rangle \rightsquigarrow s[x \mapsto (\mathcal{E}[\![e]\!]\ s)]$
- other nodes: labelled according to the rules, e.g.
  $$\frac{\boxed{2a} : \langle S_1, s\rangle \rightsquigarrow s' \qquad \boxed{2b} : \langle S_2, s'\rangle \rightsquigarrow s''}{\boxed{2} : \langle S_1; S_2, s\rangle \rightsquigarrow s''}$$
- root: judgement proved, e.g. $\boxed{3} : \langle S, s\rangle \rightsquigarrow s'$

We often write $\boxed{\vdash \langle S, s\rangle \rightsquigarrow s'}$ to indicate that there exists a proof of $\langle S, s\rangle \rightsquigarrow s'$.

Another proof technique

Induction on the structure of derivation trees

To prove $\boxed{\text{if } \vdash \langle S, s\rangle \rightsquigarrow s' \text{ then } P(S, s, s')}$ show:

- $P(x := e, s, s[x \mapsto (\mathcal{E}[\![e]\!]\ s)])$
- $P(\mathbf{skip}, s, s)$
- $P(S_1; S_2, s, s'')$ follows from $P(S_1, s, s')$ and $P(S_2, s', s'')$
- $P(\mathbf{if}\ b\ \mathbf{then}\ S_1\ \mathbf{else}\ S_2, s, s')$ follows from $P(S_1, s, s')$ whenever $\mathcal{B}[\![b]\!]\ s = \mathbf{tt}$
- $P(\mathbf{if}\ b\ \mathbf{then}\ S_1\ \mathbf{else}\ S_2, s, s')$ follows from $P(S_2, s, s')$ whenever $\mathcal{B}[\![b]\!]\ s = \mathbf{ff}$
- $P(\mathbf{while}\ b\ \mathbf{do}\ S, s, s'')$ follows from $P(S, s, s')$ and $P(\mathbf{while}\ b\ \mathbf{do}\ S, s', s'')$ whenever $\mathcal{B}[\![b]\!]\ s = \mathbf{tt}$
- $P(\mathbf{while}\ b\ \mathbf{do}\ S, s, s)$ whenever $\mathcal{B}[\![b]\!]\ s = \mathbf{ff}$

Some properties

Fact: TINY *is deterministic, i.e.:*

for each $\vdash \langle S, s\rangle \rightsquigarrow s'$, *if* $\vdash \langle S, s\rangle \rightsquigarrow s''$ *then* $s' = s''$.

Proof: By (easy) induction on the proof of $\vdash \langle S, s\rangle \rightsquigarrow s'$.

BTW: Try also to prove this by induction on the structure of S — *is there a trouble?*

YES: the semantics of **while** is *not compositional*.

More on compositionality later

Semantic equivalence

Statements $S_1, S_2 \in \mathbf{Stmt}$ are *naturally equivalent* (equivalent w.r.t. the natural semantics)

$$\boxed{S_1 \equiv_{\mathcal{NS}} S_2}$$

if for all states $s, s' \in \mathbf{State}$,

$$\vdash \langle S_1, s \rangle \leadsto s' \quad \text{iff} \quad \vdash \langle S_2, s \rangle \leadsto s'$$

Fact: *For instance, the following can be proved:*

- $S; \mathbf{skip} \equiv_{\mathcal{NS}} \mathbf{skip}; S \equiv_{\mathcal{NS}} S$
- $(S_1; S_2); S_3 \equiv_{\mathcal{NS}} S_1; (S_2; S_3)$
- $\mathbf{while}\ b\ \mathbf{do}\ S \equiv_{\mathcal{NS}} \mathbf{if}\ b\ \mathbf{then}\ (S; \mathbf{while}\ b\ \mathbf{do}\ S)\ \mathbf{else\ skip}$
- $\mathbf{if}\ b\ \mathbf{then}\ (\mathbf{if}\ b'\ \mathbf{then}\ S_1\ \mathbf{else}\ S_1')\ \mathbf{else}\ S_2$
  $\equiv_{\mathcal{NS}} \mathbf{if}\ b \wedge b'\ \mathbf{then}\ S_1\ \mathbf{else}\ (\mathbf{if}\ b \wedge \neg b'\ \mathbf{then}\ S_1'\ \mathbf{else}\ S_2)$

Congruence properties

Fact: *The semantic equivalence is preserved by the linguistic constructs:*

- *if* $S_1 \equiv_{\mathcal{NS}} S_1'$ *and* $S_2 \equiv_{\mathcal{NS}} S_2'$ *then*
  $$\boxed{S_1; S_2 \equiv_{\mathcal{NS}} S_1'; S_2'}$$
- *if* $S_1 \equiv_{\mathcal{NS}} S_1'$ *and* $S_2 \equiv_{\mathcal{NS}} S_2'$ *then*
  $$\boxed{\textbf{if } b \textbf{ then } S_1 \textbf{ else } S_2 \equiv_{\mathcal{NS}} \textbf{if } b \textbf{ then } S_1' \textbf{ else } S_2'}$$
- *if* $S \equiv_{\mathcal{NS}} S'$ *then*
  $$\boxed{\textbf{while } b \textbf{ do } S \equiv_{\mathcal{NS}} \textbf{while } b \textbf{ do } S'}$$

BTW: This can be extended to incorporate a similarly defined equivalence for expressions and boolean expressions.

Operational vs. natural semantics for TINY

"They are essentially the same"

Fact: *The two semantics are equivalent w.r.t. the final results described:*

$$\vdash \langle S, s\rangle \leadsto s' \quad \text{iff} \quad \langle S, s\rangle \Rightarrow^* s'$$

for all statements $S \in \mathbf{Stmt}$ *and states* $s, s' \in \mathbf{State}$.

Proof:

" $\Longrightarrow$": By induction on the structure of the derivation for $\langle S, s\rangle \leadsto s'$.

" $\Longleftarrow$": By induction on the length of the computation $\langle S, s\rangle \Rightarrow^* s'$.

"$\Longrightarrow$": By induction on the structure of the derivation for $\langle S, s\rangle \rightsquigarrow s'$.

- $\langle x := e, s\rangle \Rightarrow s[x \mapsto (\mathcal{E}[\![e]\!]\ s)]$. OK
- $\langle \mathbf{skip}, s\rangle \Rightarrow s$. OK
- Suppose $\langle S_1, s\rangle \rightsquigarrow s'$ and $\langle S_2, s'\rangle \rightsquigarrow s''$, so that $\langle S_1, s\rangle \Rightarrow^* s'$ and $\langle S_2, s'\rangle \Rightarrow^* s''$. Then $\langle S_1; S_2, s\rangle \Rightarrow^* \langle S_2, s'\rangle \Rightarrow^* s''$. OK
- Suppose $\mathcal{B}[\![b]\!]\ s = \mathbf{tt}$ and $\langle S_1, s\rangle \rightsquigarrow s'$, so that $\langle S_1, s\rangle \Rightarrow^* s'$. Then $\langle \mathbf{if}\ b\ \mathbf{then}\ S_1\ \mathbf{else}\ S_2, s\rangle \Rightarrow \langle S_1, s\rangle \Rightarrow^* s'$. OK
- Suppose $\mathcal{B}[\![b]\!]\ s = \mathbf{ff}$ and $\langle S_2, s\rangle \rightsquigarrow s'$, so that $\langle S_2, s\rangle \Rightarrow^* s'$. Then $\langle \mathbf{if}\ b\ \mathbf{then}\ S_1\ \mathbf{else}\ S_2, s\rangle \Rightarrow \langle S_2, s\rangle \Rightarrow^* s'$. OK
- Suppose $\mathcal{B}[\![b]\!]\ s = \mathbf{tt}$ and $\langle S, s\rangle \rightsquigarrow s'$ and $\langle \mathbf{while}\ b\ \mathbf{do}\ S, s'\rangle \rightsquigarrow s''$, so that $\langle S, s\rangle \Rightarrow^* s'$ and $\langle \mathbf{while}\ b\ \mathbf{do}\ S, s'\rangle \Rightarrow^* s''$. Then $\langle \mathbf{while}\ b\ \mathbf{do}\ S, s\rangle \Rightarrow \langle S; \mathbf{while}\ b\ \mathbf{do}\ S, s\rangle \Rightarrow^* \langle \mathbf{while}\ b\ \mathbf{do}\ S, s'\rangle \Rightarrow^* s''$. OK
- If $\mathcal{B}[\![b]\!]\ s = \mathbf{ff}$ then $\langle \mathbf{while}\ b\ \mathbf{do}\ S, s\rangle \Rightarrow s$. OK

" $\Longleftarrow$": By induction on the length of the computation $\langle S, s\rangle \Rightarrow^* s'$.

$\langle S, s\rangle \Rightarrow^k s'$**:** Take $k > 0$ and $\boxed{\langle S, s\rangle \Rightarrow \gamma \Rightarrow^{k-1} s'}$. By cases on the first step (few sample cases only):

- $\langle x := e, s\rangle \Rightarrow s[x \mapsto (\mathcal{E}[\![e]\!]\ s)]$. Then $s' = s[x \mapsto (\mathcal{E}[\![e]\!]\ s)]$; $\langle x := e, s\rangle \rightsquigarrow s[x \mapsto (\mathcal{E}[\![e]\!]\ s)]$. OK
- $\langle S_1; S_2, s\rangle \Rightarrow \langle S_1'; S_2, s''\rangle$, with $\langle S_1, s\rangle \Rightarrow \langle S_1', s''\rangle$. Then $\langle S_1'; S_2, s''\rangle \Rightarrow^{k-1} s'$, and so $\langle S_1', s''\rangle \Rightarrow^{k_1} \widehat{s''}$ and $\langle S_2, \widehat{s''}\rangle \Rightarrow^{k_2} s'$, for $k_1, k_2 > 0$ with $k_1 + k_2 = k - 1$. Hence also $\langle S_1, s\rangle \Rightarrow^{k_1+1} \widehat{s''}$. Then $\langle S_1, s\rangle \rightsquigarrow \widehat{s''}$ and $\langle S_2, \widehat{s''}\rangle \rightsquigarrow s'$, and so $\langle S_1; S_2, s\rangle \rightsquigarrow s'$. OK
- $\langle \textbf{if}\ b\ \textbf{then}\ S_1\ \textbf{else}\ S_2, s\rangle \Rightarrow \langle S_1, s\rangle$, with $\mathcal{B}[\![b]\!]\ s = \textbf{tt}$. Then $\langle S_1, s\rangle \Rightarrow^{k-1} s'$, so $\langle S_1, s\rangle \rightsquigarrow s'$ and $\langle \textbf{if}\ b\ \textbf{then}\ S_1\ \textbf{else}\ S_2, s\rangle \rightsquigarrow s'$. OK
- $\langle \textbf{while}\ b\ \textbf{do}\ S, s\rangle \Rightarrow \langle S; \textbf{while}\ b\ \textbf{do}\ S, s\rangle$, with $\mathcal{B}[\![b]\!]\ s = \textbf{tt}$. Then $\langle S; \textbf{while}\ b\ \textbf{do}\ S, s\rangle \Rightarrow^{k-1} s'$, hence $\langle S, s\rangle \Rightarrow^{k_1} \hat{s}$ and $\langle \textbf{while}\ b\ \textbf{do}\ S, \hat{s}\rangle \Rightarrow^{k_2} s'$, for $k_1, k_2 > 0$ with $k_1 + k_2 = k - 1$. Thus $\langle S, s\rangle \rightsquigarrow \hat{s}$, $\langle \textbf{while}\ b\ \textbf{do}\ S, \hat{s}\rangle \rightsquigarrow s'$, and so $\langle \textbf{while}\ b\ \textbf{do}\ S, s\rangle \rightsquigarrow s'$. OK

“Denotational” semantics of statements

$$\mathcal{S}_{\mathcal{OS}} : \mathbf{Stmt} \to (\mathbf{State} \rightharpoonup \mathbf{State})$$

extracted from the natural or operational semantics as follows:

$$\mathcal{S}_{\mathcal{OS}}[\![S]\!]\, s = s' \;\text{ iff }\; \langle S, s\rangle \leadsto s' \quad (\text{ iff }\; \langle S, s\rangle \Rightarrow^* s')$$

BTW: TINY is deterministic, so this indeed defines a function

$$\mathcal{S}_{\mathcal{OS}}[\![S]\!] : \mathbf{State} \rightharpoonup \mathbf{State}$$

However, this function in general is *partial*.

So, in fact we define:

$$\mathcal{S}_{\mathcal{OS}}[\![S]\!]\, s = \begin{cases} s' & \text{if } \langle S, s\rangle \leadsto s', \text{ i.e. } \langle S, s\rangle \Rightarrow^* s' \\ \text{undefined} & \text{if } \langle S, s\rangle \not\leadsto \end{cases}$$

Operational vs. natural semantics

“They are quite different”

Natural semantics is more abstract than operational semantics

There are naturally equivalent statements with quite different sets of computations given by the operational semantics.

- Natural semantics disregards all computations that “block” or “loop”.
- Natural semantics does not provide detailed view of computations.

Operational equivalence, naively

Statements $S_1, S_2 \in \mathbf{Stmt}$ are *operationally equivalent* (equivalent w.r.t. the operational semantics)

$$\boxed{S_1 \equiv_{\mathcal{OS}} S_2}$$

if for all states $s \in \mathbf{State}$, $\langle S_1, s\rangle \approx \langle S_2, s\rangle$, where:

configurations $\gamma_1, \gamma_2 \in \Gamma$ are equivalent, $\gamma_1 \approx \gamma_2$, if:

- $\gamma_1 = s'$ iff $\gamma_2 = s'$
- if $\gamma_1 \Rightarrow \gamma_1'$ then $\gamma_2 \Rightarrow^* \gamma_2'$ with $\gamma_1' \approx \gamma_2'$
- if $\gamma_2 \Rightarrow \gamma_2'$ then $\gamma_1 \Rightarrow^* \gamma_1'$ with $\gamma_1' \approx \gamma_2'$

THIS IS WRONG:

a circular definition!

Bisimulation

Consider any directed graph $\langle \Gamma, \Rightarrow \rangle$ with some basic observation $\mathrm{Obs}(\gamma)$ associated to every $\gamma \in \Gamma$.

A binary relation $R \subseteq \Gamma \times \Gamma$ is a strong (weak) bisimulation iff, for every $\gamma_1, \gamma_2 \in \Gamma$, if $\gamma_1 \, R \, \gamma_2$ then:

- $\mathrm{Obs}(\gamma_1) = \mathrm{Obs}(\gamma_2)$,
- for every $\gamma_1 \Rightarrow \gamma_1'$ exists $\gamma_2 \Rightarrow \gamma_2'$ ($\gamma_2 \Rightarrow^* \gamma_2'$) such that $\gamma_1' \, R \, \gamma_2'$,
- for every $\gamma_2 \Rightarrow \gamma_2'$ exists $\gamma_1 \Rightarrow \gamma_1'$ ($\gamma_1 \Rightarrow^* \gamma_1'$) such that $\gamma_1' \, R \, \gamma_2'$.

Then $\gamma_1, \gamma_2 \in \Gamma$ are strongly (weakly) bisimilar iff there exists a strong (weak) bisimulation R such that $\gamma_1 \, R, \gamma_2$.

Fact: *Strong (weak) bisimilarity is an equivalence relation and it is the largest strong (weak) bisimulation.*