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/sem18-19.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/sem18-19.html

Denotational semantics

The method

- define syntax (*syntactic domains*)
- define *semantic domains*
- define *semantic functions*
- use *compositional* definitions

Syntactic domains

Each syntactic category of the language forms a *syntactic domain*, which has as elements all the syntactic phrases in this category.

Semantic domains

Semantic domains capture the forms of the intended meanings (*denotations)* for syntactic phrases of the language. All the denotations live in semantic domains, but typically not all elements in semantic domains are denotable.

Semantic domains are defined from *basic domains* ($\mathbf{Int}, \mathbf{Bool}$) using *domain constructors*: product, (disjoint) sum, function spaces, etc.

There is a semantic domain for each key syntactic category of the language.

Semantic functions

For each syntactic category $\mathbf{Cat}$, define a *semantic function*

$$\mathcal{C} : \mathbf{Cat} \to \mathbf{CAT}$$

which assigns to the syntactic phrases $ph \in \mathbf{Cat}$ their *denotations* in the corresponding semantic domain $\mathbf{CAT}$:

$$\mathcal{C}[\![ph]\!] \in \mathbf{CAT}$$

BTW: This defines a semantic equivalence: phrases $ph_1, ph_2 \in \mathbf{Cat}$ are *semantically equivalent* (equivalent w.r.t. the denotational semantics)

$$ph_1 \equiv_{\mathcal{DS}} ph_2$$

whenever $\mathcal{C}[\![ph_1]\!] = \mathcal{C}[\![ph_2]\!]$.

Compositionality

Semantic functions are defined *compositionally*, so that the denotation of a phrase depends only on the denotations of its immediate components:

$$\mathcal{C}[\![\varphi(ph_1, \ldots, ph_n)]\!] = \Phi(\mathcal{C}[\![ph_1]\!], \ldots, \mathcal{C}[\![ph_n]\!])$$

Such a *semantic clause* is given for each syntactic construct.

Homomorphism property lurking out

Key consequences:

STRUCTURAL INDUCTION

Congruence properties of the semantic equivalence

Denotational semantics for TINY

Syntactic domains

Num (**Var**) **Exp** **BExp** **Stmt**

Somewhat informally:

$$N \in \mathbf{Num} ::= 0 \mid 1 \mid 2 \mid \cdots$$

$$(x \in \mathbf{Var} ::= \cdots)$$

$$e \in \mathbf{Exp} ::= N \mid x \mid e_1 + e_2 \mid e_1 * e_2 \mid e_1 - e_2$$

$$b \in \mathbf{BExp} ::= \mathbf{true} \mid \mathbf{false} \mid e_1 \leq e_2 \mid \neg b' \mid b_1 \wedge b_2$$

$$S \in \mathbf{Stmt} ::= x := e \mid \mathbf{skip} \mid S_1; S_2 \mid \mathbf{if}\ b\ \mathbf{then}\ S_1\ \mathbf{else}\ S_2 \mid \mathbf{while}\ b\ \mathbf{do}\ S'$$

Denotational semantics for TINY

Semantic domains

$$\mathbf{Int} \qquad (\mathbf{Bool}) \qquad (\mathbf{State}) \qquad \mathbf{EXP} \qquad \mathbf{BEXP} \qquad \mathbf{STMT}$$

$$\begin{aligned}
\mathbf{Int} &= \{0, 1, -1, 2, -2, \ldots\} \\
\mathbf{Bool} &= \{\mathbf{tt}, \mathbf{ff}\} \\
\mathbf{State} &= \mathbf{Var} \to \mathbf{Int} \\
\mathbf{EXP} &= \mathbf{State} \to \mathbf{Int} \\
\mathbf{BEXP} &= \mathbf{State} \to \mathbf{Bool} \\
\mathbf{STMT} &= \mathbf{State} \rightharpoonup \mathbf{State}
\end{aligned}$$

Semantic functions:

$$\begin{aligned}
\mathcal{N} &: \mathbf{Num} \to \mathbf{Int} \\
\mathcal{E} &: \mathbf{Exp} \to \mathbf{EXP} \\
\mathcal{B} &: \mathbf{BExp} \to \mathbf{BEXP} \\
\mathcal{S} &: \mathbf{Stmt} \to \mathbf{STMT}
\end{aligned}$$

Basic functions on semantic domains

To define semantic functions, we are free to use various well-known functions on semantic domains:

$$
\begin{aligned}
+, -, *&: \mathbf{Int} \times \mathbf{Int} \to \mathbf{Int} \\
=, \leq&: \mathbf{Int} \times \mathbf{Int} \to \mathbf{Bool} \\
\neg&: \mathbf{Bool} \to \mathbf{Bool} \\
\wedge, \vee&: \mathbf{Bool} \times \mathbf{Bool} \to \mathbf{Bool} \\
&\ldots
\end{aligned}
$$

Further constructions

Some auxiliary notation:

- *λ-notation*: $\lambda x{:}D.E$ stands for the function that maps any $d \in D$ to $E[d/x]$
- *identity*: $id_D = \lambda x{:}D.x$
- *function composition*: the composition of $f\colon D_1 \to D_2$ and $g\colon D_2 \to D_3$ is written as $f;g\colon D_1 \to D_3$
- *conditional*: $ifte_D\colon \mathbf{Bool} \times D \times D \to D$ is defined by

$$ifte_D(c, d_1, d_2) = \begin{cases} d_1 & \text{if } c = \mathbf{tt} \\ d_2 & \text{if } c = \mathbf{ff} \end{cases}$$

(the index D will often be omitted)

- *indexing*: given any function $f\colon D_1 \times \cdots \times D_n \to D$, for any domain I,

$$lift^I(f)\colon (I \to D_1) \times \cdots \times (I \to D_n) \to (I \to D)$$

is defined as follows:

$$lift^I(f)(fd_1, \ldots, fd_n) = \lambda i{:}I.f(fd_1(i), \ldots, fd_n(i))$$

For instance, the conditional on state-dependent functions, like

$$cond\colon \mathbf{BEXP} \times \mathbf{EXP} \times \mathbf{EXP} \to \mathbf{EXP}$$

given explicitly by

$$cond(B, E_1, E_2)(s) = ifte_{\mathbf{Int}}(B(s), E_1(s), E_2(s)) = \begin{cases} E_1(s) & \text{if } B(s) = \mathbf{tt} \\ E_2(s) & \text{if } B(s) = \mathbf{ff} \end{cases}$$

may be defined as $cond = lift^{\mathbf{State}}(ifte_{\mathbf{Int}})$.

All these carry over
to partial functions as well

Denotational semantics for TINY

Semantic clauses

$\mathcal{N}: \mathbf{Num} \to \mathbf{Int}$

$$\mathcal{N}[\![0]\!] = 0$$
$$\mathcal{N}[\![1]\!] = 1$$
$$\mathcal{N}[\![2]\!] = 2$$
$$\ldots$$

$\mathcal{E}: \mathbf{Exp} \to \mathbf{EXP}$

$$\mathcal{E}[\![N]\!]s = \mathcal{N}[\![N]\!] \qquad \mathcal{E}[\![x]\!]s = s\,x$$
$$\mathcal{E}[\![e_1 + e_2]\!]s = \mathcal{E}[\![e_1]\!]s + \mathcal{E}[\![e_2]\!]s$$
$$\mathcal{E}[\![e_1 * e_2]\!]s = \mathcal{E}[\![e_1]\!]s * \mathcal{E}[\![e_2]\!]s$$
$$\mathcal{E}[\![e_1 - e_2]\!]s = \mathcal{E}[\![e_1]\!]s - \mathcal{E}[\![e_2]\!]s$$

$\mathcal{B}: \mathbf{BExp} \to \mathbf{BEXP}$

$$\mathcal{B}[\![\mathbf{true}]\!]s = \mathbf{tt} \qquad \mathcal{B}[\![\mathbf{false}]\!]s = \mathbf{ff} \qquad \mathcal{B}[\![\neg b]\!]s = \neg(\mathcal{B}[\![b]\!]s)$$
$$\mathcal{B}[\![e_1 \leq e_2]\!]s = (\mathcal{E}[\![e_1]\!]s \leq \mathcal{E}[\![e_2]\!]s) \qquad \mathcal{B}[\![b_1 \wedge b_2]\!]s = (\mathcal{B}[\![b_1]\!]s \wedge \mathcal{B}[\![b_2]\!]s)$$

Denotational semantics for TINY

The same clauses, using some notational sugar

$\mathcal{N} : \mathbf{Num} \to \mathbf{Int}$

$$\mathcal{N}[\![0]\!] = 0$$
$$\mathcal{N}[\![1]\!] = 1$$
$$\mathcal{N}[\![2]\!] = 2$$
$$\ldots$$

$\mathcal{E} : \mathbf{Exp} \to \mathbf{EXP}$

$$\mathcal{E}[\![N]\!] = \lambda s{:}\mathbf{State}.\mathcal{N}[\![N]\!] \quad \mathcal{E}[\![x]\!] = \lambda s{:}\mathbf{State}.s\ x$$
$$\mathcal{E}[\![e_1 + e_2]\!] = lift^{\mathbf{State}}(+)(\mathcal{E}[\![e_1]\!], \mathcal{E}[\![e_2]\!])$$
$$\mathcal{E}[\![e_1 * e_2]\!] = lift^{\mathbf{State}}(*)(\mathcal{E}[\![e_1]\!], \mathcal{E}[\![e_2]\!])$$
$$\mathcal{E}[\![e_1 - e_2]\!] = lift^{\mathbf{State}}(-)(\mathcal{E}[\![e_1]\!], \mathcal{E}[\![e_2]\!])$$

$\mathcal{B} : \mathbf{BExp} \to \mathbf{BEXP}$

$$\mathcal{B}[\![\mathbf{true}]\!] = \lambda s{:}\mathbf{State}.\mathbf{tt} \quad \mathcal{B}[\![\mathbf{false}]\!] = \lambda s{:}\mathbf{State}.\mathbf{ff} \quad \mathcal{B}[\![\neg b]\!] = lift^{\mathbf{State}}(\neg)(\mathcal{B}[\![b]\!])$$
$$\mathcal{B}[\![e_1 \leq e_2]\!] = lift^{\mathbf{State}}(\leq)(\mathcal{E}[\![e_1]\!], \mathcal{E}[\![e_2]\!]) \qquad \mathcal{B}[\![b_1 \wedge b_2]\!] = lift^{\mathbf{State}}(\wedge)(\mathcal{B}[\![b_1]\!], \mathcal{B}[\![b_2]\!])$$

Denotational semantics for TINY

Semantic clauses

$\mathcal{S}$: **Stmt** $\to$ **STMT**

$$
\begin{aligned}
\mathcal{S}[\![x := e]\!]s &= s[x \mapsto \mathcal{E}[\![e]\!]\ s] \\
\mathcal{S}[\![\textbf{skip}]\!]s &= s \\
\mathcal{S}[\![S_1 ; S_2]\!]s &= \mathcal{S}[\![S_2]\!](\mathcal{S}[\![S_1]\!]s) \\
\mathcal{S}[\![\textbf{if } b \textbf{ then } S_1 \textbf{ else } S_2]\!]s &= ifte_{\textbf{State}}(\mathcal{B}[\![b]\!]s, \mathcal{S}[\![S_1]\!]s, \mathcal{S}[\![S_2]\!]s) \\
\mathcal{S}[\![\textbf{while } b \textbf{ do } S]\!]s &= ifte_{\textbf{State}}(\mathcal{B}[\![b]\!]s, \mathcal{S}[\![\textbf{while } b \textbf{ do } S]\!]s', s) \\
&\qquad \text{where } s' = \mathcal{S}[\![S]\!]s
\end{aligned}
$$

Denotational semantics for TINY

The same clauses with notational sugar

$\mathcal{S}: \mathbf{Stmt} \to \mathbf{STMT}$

$$
\begin{aligned}
\mathcal{S}[\![x := e]\!] &= \lambda s{:}\mathbf{State}.s[x \mapsto \mathcal{E}[\![e]\!]\ s] \\
\mathcal{S}[\![\mathbf{skip}]\!] &= id_{\mathbf{State}} \\
\mathcal{S}[\![S_1; S_2]\!] &= \mathcal{S}[\![S_1]\!]; \mathcal{S}[\![S_2]\!] \\
\mathcal{S}[\![\mathbf{if}\ b\ \mathbf{then}\ S_1\ \mathbf{else}\ S_2]\!] &= cond(\mathcal{B}[\![b]\!], \mathcal{S}[\![S_1]\!], \mathcal{S}[\![S_2]\!]) \\
\mathcal{S}[\![\mathbf{while}\ b\ \mathbf{do}\ S]\!] &= cond(\mathcal{B}[\![b]\!], \mathcal{S}[\![S]\!]; \mathcal{S}[\![\mathbf{while}\ b\ \mathbf{do}\ S]\!], id_{\mathbf{State}})
\end{aligned}
$$

Something wrong?

The clause for **while**:

$$\mathcal{S}[\![\textbf{while } b \textbf{ do } S]\!] = cond(\mathcal{B}[\![b]\!], \mathcal{S}[\![S]\!]; \mathcal{S}[\![\textbf{while } b \textbf{ do } S]\!], id_{\textbf{State}})$$

is *not* compositional!

We "define":

$$??? \quad \mathcal{S}[\![\textbf{while } b \textbf{ do } S]\!] = \Phi(\ldots, \mathcal{S}[\![\textbf{while } b \textbf{ do } S]\!], \ldots) \quad ???$$

We need *fixed point definitions*

Potential problems with fixed point definitions

Consider fixed point definitions in $\mathbf{STMT} = \mathbf{State} \rightharpoonup \mathbf{State}$, as

$$\mathcal{S}[\![\textbf{while } b \textbf{ do } S]\!] = \Phi(\ldots, \mathcal{S}[\![\textbf{while } b \textbf{ do } S]\!], \ldots)$$

- Does a fixed point always exist?

$$f = \lambda s{:}\mathbf{State}.\mathit{ifte}_{\mathbf{State}}(f(s) \text{ is not defined}, s, f(s)[var \mapsto (f(s)\ var) + 1])$$

Only some functionals Φ *may be allowed*

- If a fixed point exists, is it unique?

$$f = \lambda s{:}\mathbf{State}.f(s)[var \mapsto 2 * (f(s)\ var)]$$

(or even: $f = \lambda s{:}\mathbf{State}.f(s)$)

Some "best" fixed point must be chosen

The guiding fixed point definition

Looking closer at the clause for **while**:

$$\mathcal{S}[\![\textbf{while } b \textbf{ do } S]\!] = \Phi(\mathcal{S}[\![\textbf{while } b \textbf{ do } S]\!])$$

where $\Phi\colon \mathbf{STMT} \to \mathbf{STMT}$ is defined as follows:

$$\Phi(F) = cond(\mathcal{B}[\![b]\!], \mathcal{S}[\![S]\!]; F, id_{\mathbf{State}})$$

Whatever fixed point we choose, we want it to be adequate for our operational intuitions; we want a denotation $fix(\Phi) \in \mathbf{STMT}$ that is a fixed point of Φ (so that $\Phi(fix(\Phi)) = fix(\Phi)$) and is adequate for the operational semantics of **while**, i.e., such that

$$\langle \textbf{while } b \textbf{ do } S, s\rangle \Rightarrow^* s' \quad \text{iff} \quad fix(\Phi)\, s = s'$$

Right guess!

Suppose that we have such adequacy for S, i.e., $\langle S, s\rangle \Rightarrow^* s'$ iff $\mathcal{S}[\![S]\!]\, s = s'$.

Right guess:

$$\langle \textbf{while } b \textbf{ do } S, s\rangle \Rightarrow^* s' \quad \text{iff} \quad \text{for some } n \geq 0, \Phi^n(\emptyset_{\mathbf{State} \rightharpoonup \mathbf{State}})\, s = s'$$

where $\emptyset_{\mathbf{State} \rightharpoonup \mathbf{State}} : \mathbf{State} \rightharpoonup \mathbf{State}$ is the function undefined everywhere, $\Phi^0(\emptyset_{\mathbf{State} \rightharpoonup \mathbf{State}}) = \emptyset_{\mathbf{State} \rightharpoonup \mathbf{State}}$, and $\Phi^{n+1}(\emptyset_{\mathbf{State} \rightharpoonup \mathbf{State}}) = \Phi(\Phi^n(\emptyset_{\mathbf{State} \rightharpoonup \mathbf{State}}))$.

Proof: in a moment.

Conclusion

$$\mathcal{S}[\![\textbf{while } b \textbf{ do } S]\!] = \mathit{fix}(\Phi) = \bigcup_{n \geq 0} \Phi^n(\emptyset_{\mathbf{State} \rightharpoonup \mathbf{State}})$$

This is well-defined, and yields the *least* fix-point of Φ, see below.

$$\textbf{while } sqr \leq n \textbf{ do } rt := rt + 1; sqr := sqr + 2 * rt + 1$$

$$\Phi(F) = cond(\mathcal{B}[\![sqr \leq n]\!], \mathcal{S}[\![rt := rt + 1; sqr := sqr + 2 * rt + 1]\!]; F, id_{\textbf{State}})$$

$s(n, rt, sqr)$	$\Phi^0(\emptyset)(s)$	$\Phi^1(\emptyset)(s)$	$\Phi^2(\emptyset)(s)$	$\Phi^3(\emptyset)(s)$	$\Phi^4(\emptyset)(s)$	$\cdots$	$\bigcup \Phi^n(\emptyset)(s)$
$0, 0, 1$	?	$0, 0, 1$	$0, 0, 1$	$0, 0, 1$	$0, 0, 1$	$\cdots$	$0, 0, 1$
$1, 0, 1$	?	?	$1, 1, 4$	$1, 1, 4$	$1, 1, 4$	$\cdots$	$1, 1, 4$
$2, 0, 1$	?	?	$2, 1, 4$	$2, 1, 4$	$2, 1, 4$	$\cdots$	$2, 1, 4$
$3, 0, 1$	?	?	$3, 1, 4$	$3, 1, 4$	$3, 1, 4$	$\cdots$	$3, 1, 4$
$4, 0, 1$	?	?	?	$4, 2, 9$	$4, 2, 9$	$\cdots$	$4, 2, 9$
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
$8, 0, 1$	?	?	?	$8, 2, 9$	$8, 2, 9$	$\cdots$	$8, 2, 9$
$9, 0, 1$	?	?	?	?	$9, 3, 16$	$\cdots$	$9, 3, 16$
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$

$$\Phi(F) = cond(\mathcal{B}[\![sqr \leq n]\!], \mathcal{S}[\![rt := rt + 1; sqr := sqr + 2 * rt + 1]\!]; F, id_{\mathbf{State}})$$

$s(n, rt, sqr)$	$\Phi^0(\emptyset)(s)$	$\Phi^1(\emptyset)(s)$	$\Phi^2(\emptyset)(s)$	$\Phi^3(\emptyset)(s)$	$\Phi^4(\emptyset)(s)$	$\cdots$	$\bigcup \Phi^n(\emptyset)(s)$
$0, 0, 1$	?	$0, 0, 1$	$0, 0, 1$	$0, 0, 1$	$0, 0, 1$	$\cdots$	$0, 0, 1$
$1, 0, 1$	?	?	$1, 1, 4$	$1, 1, 4$	$1, 1, 4$	$\cdots$	$1, 1, 4$
$1, 1, 4$	?	$1, 1, 4$	$1, 1, 4$	$1, 1, 4$	$1, 1, 4$	$\cdots$	$1, 1, 4$
$2, 0, 1$	?	?	$2, 1, 4$	$2, 1, 4$	$2, 1, 4$	$\cdots$	$2, 1, 4$
$2, 1, 4$	?	$2, 1, 4$	$2, 1, 4$	$2, 1, 4$	$2, 1, 4$	$\cdots$	$2, 1, 4$
$3, 0, 1$	?	?	$3, 1, 4$	$3, 1, 4$	$3, 1, 4$	$\cdots$	$3, 1, 4$
$3, 1, 4$	?	$3, 1, 4$	$3, 1, 4$	$3, 1, 4$	$3, 1, 4$	$\cdots$	$3, 1, 4$
$4, 0, 1$	?	?	?	$4, 2, 9$	$4, 2, 9$	$\cdots$	$4, 2, 9$
$4, 1, 4$	?	?	$4, 2, 9$	$4, 2, 9$	$4, 2, 9$	$\cdots$	$4, 2, 9$
$4, 2, 9$	?	$4, 2, 9$	$4, 2, 9$	$4, 2, 9$	$4, 2, 9$	$\cdots$	$4, 2, 9$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
$9, 0, 1$	?	?	?	?	$9, 3, 16$	$\cdots$	$9, 3, 16$
$9, 1, 4$	?	?	?	$9, 3, 16$	$9, 3, 16$	$\cdots$	$9, 3, 16$
$9, 2, 9$	?	?	$9, 3, 16$	$9, 3, 16$	$9, 3, 16$	$\cdots$	$9, 3, 16$
$9, 3, 16$	?	$9, 3, 16$	$9, 3, 16$	$9, 3, 16$	$9, 3, 16$	$\cdots$	$9, 3, 16$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$

Proof

"$\Longrightarrow$": By induction on the length of the computation $\langle \textbf{while } b \textbf{ do } S, s\rangle \Rightarrow^k s'$.

$k > 0$: Then $\langle \textbf{while } b \textbf{ do } S, s\rangle \Rightarrow \gamma \Rightarrow^{k-1} s'$. By cases on this first step:

- $\mathcal{B}[\![b]\!]\, s = \textbf{ff}$ and $\gamma = s$. Then $s' = s$, and $\Phi(\emptyset_{\textbf{State}\rightharpoonup\textbf{State}})\, s = s$. OK
- $\mathcal{B}[\![b]\!]\, s = \textbf{tt}$ and $\gamma = \langle S; \textbf{while } b \textbf{ do } S, s\rangle \Rightarrow^{k-1} s'$. Then $\langle S, s\rangle \Rightarrow^{k_1} \hat{s}$ and $\langle \textbf{while } b \textbf{ do } S, \hat{s}\rangle \Rightarrow^{k_2} s'$, for some $\hat{s} \in \textbf{State}$ and $k_1, k_2 > 0$ with $k_1 + k_2 = k - 1$. Hence, $\mathcal{S}[\![S]\!]\, s = \hat{s}$ and $\Phi^n(\emptyset_{\textbf{State}\rightharpoonup\textbf{State}})\, \hat{s} = s'$ for some $n \geq 0$. Thus, $\Phi^{n+1}(\emptyset_{\textbf{State}\rightharpoonup\textbf{State}})\, s = s'$. OK

BTW: This relies only on $\boxed{\langle S, s\rangle \Rightarrow^* s' \Longrightarrow \mathcal{S}[\![S]\!]\, s = s'}$

Proof

"$\Longleftarrow$": By induction on $n \geq 0$, assuming $\Phi^n(\emptyset_{\mathbf{State} \rightharpoonup \mathbf{State}})\ s = s'$.

$n > 0$: Then

$\Phi^n(\emptyset_{\mathbf{State} \rightharpoonup \mathbf{State}})\ s = cond(\mathcal{B}[\![b]\!], \mathcal{S}[\![S]\!]; \Phi^{n-1}(\emptyset_{\mathbf{State} \rightharpoonup \mathbf{State}}), id_{\mathbf{State}})\ s$.

- $\mathcal{B}[\![b]\!]\ s = \mathbf{ff}$: then $\Phi^n(\emptyset_{\mathbf{State} \rightharpoonup \mathbf{State}})\ s = s$, so $s' = s$, and also $\langle \mathbf{while}\ b\ \mathbf{do}\ S, s\rangle \Rightarrow s$. OK
- $\mathcal{B}[\![b]\!]\ s = \mathbf{tt}$: then $\Phi^n(\emptyset_{\mathbf{State} \rightharpoonup \mathbf{State}})\ s = \Phi^{n-1}(\emptyset_{\mathbf{State} \rightharpoonup \mathbf{State}})\ (\mathcal{S}[\![S]\!]\ s)$. Hence, $\langle \mathbf{while}\ b\ \mathbf{do}\ S, \mathcal{S}[\![S]\!]\ s\rangle \Rightarrow^*\ s'$, and since $\langle S, s\rangle \Rightarrow^*\ (\mathcal{S}[\![S]\!]\ s)$, we get $\langle \mathbf{while}\ b\ \mathbf{do}\ S, s\rangle \Rightarrow \langle S; \mathbf{while}\ b\ \mathbf{do}\ S, s\rangle \Rightarrow^*$ $\langle \mathbf{while}\ b\ \mathbf{do}\ S, \mathcal{S}[\![S]\!]\ s\rangle \Rightarrow^*\ s'$. OK

BTW: This relies only on $\boxed{\langle S, s\rangle \Rightarrow^*\ s' \Longleftarrow \mathcal{S}[\![S]\!]\ s = s'}$

Adequacy of denotational semantics

Fact: *For each statement* $S \in \mathbf{Stmt}$ *and states* $s, s' \in \mathbf{State}$,

$$\mathcal{S}[\![S]\!]\, s = s' \quad \text{iff} \quad \langle S, s\rangle \Rightarrow^* s'$$

Proof:

"$\Longrightarrow$": By structural induction on S.

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

How general is this fixpoint construction?

For any sets X and Y, define the *information ordering* on partial functions from X to Y:

$$f \sqsubseteq g \text{ iff } [f(x) \text{ is defined implies } g(x) \text{ is defined and } f(x) = g(x)]$$

This is a *complete partial order* (c.p.o.) on $X \rightharpoonup Y$, i.e., it has the least element $\emptyset_{X \rightharpoonup Y}$ and limits (least upper bounds) of increasing chains:

$$f_0 \sqsubseteq f_1 \sqsubseteq f_2 \sqsubseteq f_3 \sqsubseteq \cdots \sqsubseteq \bigcup_{n \in \mathbb{N}} f_n$$

Putting $X = Y = \mathbf{State}$, we get a c.p.o. on the set $\mathbf{STMT}$.

Continuous functions

Fact: *For any* $S \in \mathbf{Stmt}$ *and* $b \in \mathbf{BExp}$*, the function*

$$\Phi(F) = cond(\mathcal{B}[\![b]\!], \mathcal{S}[\![S]\!]; F, id_{\mathbf{State}})$$

is monotone and even continuous:

- $\Phi(f) \sqsubseteq \Phi(g)$ *if* $f \sqsubseteq g$,
- $\Phi\left(\bigcup_{n\in\mathbb{N}} f_n\right) = \bigcup_{n\in\mathbb{N}} \Phi(f_n)$ *if* $f_1 \sqsubseteq f_2 \sqsubseteq f_3 \sqsubseteq \cdots$.

Fact: (Kleene fixpoint theorem) *Every continuous function* Φ *on a c.p.o.* X *has a least fixpoint, defined by:*

$$\bigcup_{n\in\mathbb{N}} \Phi^n(\emptyset).$$

Continuous functions

Some functions are not continuous, or even not monotone. Recall our flawed "fixpoint definition":

$$f = \lambda s{:}\mathbf{State}.\mathit{ifte}_{\mathbf{State}}(f(s) \text{ is not defined}, s, f(s)[\mathit{var} \mapsto (f(s)\ \mathit{var}) + 1])$$

This is supposed to be a fixpoint of the function $\Phi : \mathbf{STMT} \to \mathbf{STMT}$:

$$\Phi(F) = \lambda s{:}\mathbf{State}.\mathit{ifte}_{\mathbf{State}}(F(s) \text{ is not defined}, s, F(s)[\mathit{var} \mapsto (F(s)\ \mathit{var}) + 1])$$

This Φ is well-defined, but it is not monotone: $\emptyset_{\mathbf{State} \rightharpoonup \mathbf{State}} \sqsubseteq \mathit{id}_{\mathbf{State}}$, but

$$\Phi(\emptyset_{\mathbf{State} \rightharpoonup \mathbf{State}}) = \mathit{id}_{\mathbf{State}} \not\sqsubseteq \Phi(\mathit{id}_{\mathbf{State}}).$$

In practice, all "reasonable" functions that we are likely to write are continuous.