Denotational semantics

The method

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

History, late 60s, 70s:

- Strachey, Scott (PRG/Oxford)
- VDM (Bekič, Lucas, Jones, Bjørner)
- Plotkin, Smyth, Stoy, Mosses, de Bakker, Tennent, Reynolds ...

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 φ. For instance:

$$\mathcal{S}[\![\textbf{if } b \textbf{ then } S_1 \textbf{ else } S_2]\!] = \ldots \mathcal{B}[\![b]\!] \ldots \mathcal{S}[\![S_1]\!] \ldots \mathcal{S}[\![S_2]\!] \ldots$$

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

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

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

STMT: *partial functions*

Semantic functions:

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

Semantic function definitions

- Use any mathematical constructions/functions/notations that make sense for our semantic domains.
- In particular, feel free to use basic operations on data in our semantic domains:

$$+, *, -: \mathbf{Int} \times \mathbf{Int} \to \mathbf{Int}$$
$$\leq: \mathbf{Int} \times \mathbf{Int} \to \mathbf{Bool}$$
$$\neg: \mathbf{Bool} \to \mathbf{Bool}$$
$$\vee, \wedge: \mathbf{Bool} \times \mathbf{Bool} \to \mathbf{Bool}$$
$$\ldots$$

Remember though: **COMPOSITIONALITY!**

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, when clear from the context)

- *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 $cond = lift^{\mathbf{State}}(ifte_{\mathbf{Int}})$.

$\mathbf{EXP} = \mathbf{State} \nrightarrow \mathbf{Int}$
$\mathbf{BEXP} = \mathbf{State} \nrightarrow \mathbf{Bool}$

All these carry over
to partial functions as well

Currying

named after Haskell Curry
by Christopher Strachey

$$D_1 \times D_2 \times \cdots \times D_n \to D \cong D_1 \to (D_2 \to \cdots \to (D_n \to D) \cdots)$$

Notational conventions:

- $D_1 \to D_2 \to \cdots \to D_n \to D$ means $D_1 \to (D_2 \to \cdots \to (D_n \to D) \cdots)$
- for $f : D_1 \to D_2 \to \cdots \to D_n \to D$, $d_1 \in D_1$, $d_2 \in D_2$, $\ldots$, $d_n \in D_n$, $f\, d_1\, d_2 \cdots d_n$ means $(\cdots((f\, d_1)\, d_2) \cdots)\, d_n$
- where as before $f\, d_1$ means $f(d_1)$, etc

Denotational semantics for TINY

Semantic clauses

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

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

$\mathcal{E}$: **Exp** $\to$ **EXP**, where **EXP** = **State** $\to$ **Int**

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

$\mathcal{B}$: **BExp** $\to$ **BEXP**, where **BEXP** = **State** $\to$ **Bool**

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

Denotational semantics for TINY

Semantic clauses

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

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

$\mathcal{E}: \mathbf{Exp} \to \mathbf{EXP}$, where $\mathbf{EXP} = \mathbf{State} \to \mathbf{Int}$

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

$\mathcal{B}: \mathbf{BExp} \to \mathbf{BEXP}$, where $\mathbf{BEXP} = \mathbf{State} \to \mathbf{Bool}$

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

Denotational semantics for TINY

Semantic clauses

$\mathcal{S}$: **Stmt** $\to$ **STMT**, where **STMT** = **State** $\rightharpoonup$ **State**

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

Denotational semantics for TINY

Semantic clauses

$\mathcal{S}$: **Stmt** $\to$ **STMT**, where **STMT** = **State** $\rightharpoonup$ **State**

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

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

We need *fixed-point definitions*

For $\Phi : (\textbf{State} \rightharpoonup \textbf{State}) \to (\textbf{State} \rightharpoonup \textbf{State})$, we want to use the clause

$$F = \Phi(F)$$

to define an "appropriate" $F : \textbf{State} \rightharpoonup \textbf{State}$???

Potential problems with fixed-point definitions

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

$$F = \Phi(F)$$

- Does a fixed point always exist?

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

Only some functionals Φ *may be allowed*

- If a fixed point exists, is it unique?

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

(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 Φ: **STMT** $\to$ **STMT** is defined as follows:

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

Whichever fixed point we choose, we want it to be adequate for our operational intuitions; we want a denotation $fix(\Phi) \in$ **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

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

Right choice!

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

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

where $\emptyset_{\textbf{State}\rightharpoonup\textbf{State}} : \textbf{State} \rightharpoonup \textbf{State}$ is the function undefined everywhere, $\Phi^0(\emptyset_{\textbf{State}\rightharpoonup\textbf{State}}) = \emptyset_{\textbf{State}\rightharpoonup\textbf{State}}$, and $\Phi^{n+1}(\emptyset_{\textbf{State}\rightharpoonup\textbf{State}}) = \Phi(\Phi^n(\emptyset_{\textbf{State}\rightharpoonup\textbf{State}}))$, and $\Phi(F) = cond(\mathcal{B}[\![b]\!], \mathcal{S}[\![S]\!]; F, id_{\textbf{State}})$. Proof: in a (long :-) moment.

Conclusion

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

This is well-defined, and yields the *least* fixed point of Φ.

Believe this for now; more on this later.

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

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

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

Adequacy of denotational semantics

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

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

Proof:

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

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