Denotational Semantics

Renato Neves

Universidade do Minho

HASLab
HIGH-ASSURANCE
SOFTWARE LABORATORY

Table of Contents

Semantics for every season

Operational semantics	How a program operates
Denotational semantics	What a program is
Axiomatic semantics	Which logical properties a program satisfies

Table of Contents

Compiler correctness and contextual equivalence

We adopted the following notion of equivalence

$$\mathrm{p} \equiv_o \mathrm{q} \text{ iff } \Big(\text{for every } \sigma.\ \langle \mathrm{p}, \sigma \rangle \Downarrow \sigma' \text{ iff } \langle \mathrm{q}, \sigma \rangle \Downarrow \sigma' \Big)$$

Compiler correctness and contextual equivalence

We adopted the following notion of equivalence

$$\mathrm{p} \equiv_o \mathrm{q} \text{ iff } \Big(\text{for every } \sigma.\ \langle \mathrm{p}, \sigma \rangle \Downarrow \sigma' \text{ iff } \langle \mathrm{q}, \sigma \rangle \Downarrow \sigma' \Big)$$

Compilers adopt the stronger version

$$\mathrm{p} \equiv \mathrm{q} \text{ iff } \Big(\text{for every context } C.\ C[\mathrm{p}] \equiv_o C[\mathrm{q}] \Big)$$

Compiler correctness and contextual equivalence

We adopted the following notion of equivalence

$$\mathrm{p} \equiv_o \mathrm{q} \text{ iff } \Big(\text{for every } \sigma.\ \langle \mathrm{p}, \sigma \rangle \Downarrow \sigma' \text{ iff } \langle \mathrm{q}, \sigma \rangle \Downarrow \sigma'\Big)$$

Compilers adopt the stronger version

$$\mathrm{p} \equiv \mathrm{q} \text{ iff } \Big(\text{for every context } C.\ C[\mathrm{p}] \equiv_o C[\mathrm{q}]\Big)$$

$\boxed{\text{Why is that ?}}$

Contextual equivalence

Contexts

$C ::= [-] \mid C \wedge \mathtt{b} \mid \mathtt{b} \wedge C \mid \neg C$

Exercise

Prove the equivalence $\mathtt{b}_1 \equiv_o \mathtt{b}_2 \iff \mathtt{b}_1 \equiv \mathtt{b}_2$

Contextual equivalence

Contexts

$C ::= [-] \mid C \wedge \mathtt{b} \mid \mathtt{b} \wedge C \mid \neg C$

Exercise

Prove the equivalence $\mathtt{b}_1 \equiv_o \mathtt{b}_2 \iff \mathtt{b}_1 \equiv \mathtt{b}_2$

Homework: repeat the exercise now for arithmetic expressions

Contextual equivalence

Contexts

$C ::= [-] \mid C \,\texttt{;}\, \mathtt{p} \mid \texttt{if}\ \mathtt{b}\ \texttt{then}\ C\ \texttt{else}\ \mathtt{p} \mid \texttt{while}\ \mathtt{b}\ \texttt{do}\ \{\ C\ \} \mid \ldots$

Contextual equivalence

Contexts

$C ::= [-] \mid C \,\texttt{;}\, \mathrm{p} \mid \texttt{if}\ \mathrm{b}\ \texttt{then}\ C\ \texttt{else}\ \mathrm{p} \mid \texttt{while}\ \mathrm{b}\ \texttt{do}\ \{\ C\ \} \mid \ldots$

$\boxed{\text{Can we still prove } \mathrm{p} \equiv_o \mathrm{q} \iff \mathrm{p} \equiv \mathrm{q}\ ?}$

Next challenge: programs as part of a mathematical theory

$$\boxed{\text{Programming language}} \quad \hookrightarrow \quad \boxed{\text{Mathematical theory}}$$

The latter include *e.g.*

- functions (recall program calculus)
- linear algebra
- relations
- domain theory (theory of computability and beyond)
- ...

Table of Contents

Boolean terms and their denotational semantics

$$\mathtt{b} ::= \mathtt{x} \mid \mathtt{b} \wedge \mathtt{b} \mid \neg \mathtt{b}$$

Terms interpreted as functions $[\![\mathtt{b}]\!] : \mathit{State} \to 2$

Term operations interpreted via the boolean algebra 2

$$[\![\mathtt{x}]\!](\sigma) = \sigma(\mathtt{x})$$
$$[\![\mathtt{b}_1 \wedge \mathtt{b}_2]\!] = (\wedge) \cdot \langle [\![\mathtt{b}_1]\!], [\![\mathtt{b}_2]\!] \rangle$$
$$[\![\neg \mathtt{b}]\!] = (\neg) \cdot [\![\mathtt{b}]\!]$$

The relation between big-step and denotational semantics

Theorem

For every term $\mathtt{b}$ *and memory* σ *we have* $\langle \mathtt{b}, \sigma \rangle \Downarrow v$ *iff* $[\![\mathtt{b}]\!](\sigma) = v$

Proof.

Straightforward induction $\square$

Corollary

$\mathtt{b}_1 \equiv \mathtt{b}_2$ *iff* $\mathtt{b}_1 \equiv_o \mathtt{b}_2$ *iff* $[\![\mathtt{b}_1]\!] = [\![\mathtt{b}_2]\!]$

Profits!

We can now reduce checking for equivalence to . . .

$\boxed{\text{Program calculus and Boolean algebra}}$

Profits!

We can now reduce checking for equivalence to ...

Program calculus and Boolean algebra

Example

$$
\begin{aligned}
[\![\mathtt{b}_1 \wedge \mathtt{b}_2]\!] &= (\wedge) \cdot \langle [\![\mathtt{b}_1]\!], [\![\mathtt{b}_2]\!] \rangle \\
&= (\wedge) \cdot \mathrm{sw} \cdot \langle [\![\mathtt{b}_1]\!], [\![\mathtt{b}_2]\!] \rangle \\
&= (\wedge) \cdot \langle \pi_2, \pi_1 \rangle \cdot \langle [\![\mathtt{b}_1]\!], [\![\mathtt{b}_2]\!] \rangle \\
&= (\wedge) \cdot \langle \pi_2 \cdot \langle [\![\mathtt{b}_1]\!], [\![\mathtt{b}_2]\!] \rangle, \pi_1 \cdot \langle [\![\mathtt{b}_1]\!], [\![\mathtt{b}_2]\!] \rangle \rangle \\
&= (\wedge) \cdot \langle [\![\mathtt{b}_2]\!], [\![\mathtt{b}_1]\!] \rangle \\
&= [\![\mathtt{b}_2 \wedge \mathtt{b}_1]\!]
\end{aligned}
$$

Exercises

1. Show that $\mathtt{b} \wedge \mathtt{b} \equiv \mathtt{b}$ via the denotational semantics
2. Define a denotational semantics for arithmetic expressions e
3. Prove that $[\![\mathsf{e}_1 + \mathsf{e}_2]\!] = [\![\mathsf{e}_2 + \mathsf{e}_1]\!]$
4. Prove the equivalence $\langle \mathsf{e}, \sigma \rangle \Downarrow v$ iff $[\![\mathsf{e}]\!](\sigma) = v$

Table of Contents

Key takeaways

Programs interpreted as functions $[\![\mathrm{p}]\!] : State_{\perp} \to State_{\perp}$

$State_{\perp} = State \cup \{\perp\}$ where $\perp$ represents non-termination

Sequential composition is function composition

Programs and their denotational semantics

$$\mathtt{p} ::= \mathtt{x} := \mathtt{e} \mid \mathtt{p} \,;\, \mathtt{p} \mid \mathtt{if}\ \mathtt{b}\ \mathtt{then}\ \mathtt{p}\ \mathtt{else}\ \mathtt{p} \mid \mathtt{while}\ \mathtt{b}\ \mathtt{do}\ \{\ \mathtt{p}\ \}$$

$$\begin{aligned}
[\![\mathtt{x} := \mathtt{e}]\!] &= \sigma \mapsto \sigma[[\![\mathtt{e}]\!]/\mathtt{x}] \\
[\![\mathtt{p} \,;\, \mathtt{q}]\!] &= [\![\mathtt{q}]\!] \cdot [\![\mathtt{p}]\!] \\
[\![\mathtt{if}\ \mathtt{b}\ \mathtt{then}\ \mathtt{p}\ \mathtt{else}\ \mathtt{q}]\!] &= [[\![\mathtt{p}]\!], [\![\mathtt{q}]\!]] \cdot \mathrm{dist} \cdot \langle [\![\mathtt{b}]\!], \mathrm{id} \rangle \\
[\![\mathtt{while}\ \mathtt{b}\ \mathtt{do}\ \{\ \mathtt{p}\ \}]\!] &= \ \ldots\ldots
\end{aligned}$$

The relation between big-step and denotational semantics

Danger, Will Robinson: no while-loops yet ...

Theorem

For every p *and* σ *we have* $\langle \mathrm{p}, \sigma \rangle \Downarrow \sigma'$ *iff* $[\![\mathrm{p}]\!](\sigma) = \sigma'$

Proof.

Straightforward induction $\square$

Corollary

$\mathrm{p} \equiv \mathrm{q}$ *iff* $\mathrm{p} \equiv_o \mathrm{q}$ *iff* $[\![\mathrm{p}]\!] = [\![\mathrm{q}]\!]$

Profits!

Recall when we had to prove the two equivalences

- $(\mathrm{p} \, ; \mathrm{q}) \, ; \mathrm{r} \equiv \mathrm{p} \, ; (\mathrm{q} \, ; \mathrm{r})$
- $(\texttt{if}\ \mathrm{b}\ \texttt{then}\ \mathrm{p}\ \texttt{else}\ \mathrm{q}) \, ; \mathrm{r} \equiv \texttt{if}\ \mathrm{b}\ \texttt{then}\ \mathrm{p} \, ; \mathrm{r}\ \texttt{else}\ \mathrm{q} \, ; \mathrm{r}$

with the big-step semantics

Show the same via the denotational semantics

Programs and a (tentative) denotational semantics

$$\mathtt{p} ::= \mathtt{x} := \mathtt{e} \mid \mathtt{p}\,;\,\mathtt{p} \mid \mathtt{if}\ \mathtt{b}\ \mathtt{then}\ \mathtt{p}\ \mathtt{else}\ \mathtt{p} \mid \mathtt{while}\ \mathtt{b}\ \mathtt{do}\ \{\ \mathtt{p}\ \}$$

$$\begin{aligned}
[\![\mathtt{x} := \mathtt{e}]\!] &= \sigma \mapsto \sigma[[\![\mathtt{e}]\!]/\mathtt{x}] \\
[\![\mathtt{p}\,;\,\mathtt{q}]\!] &= [\![\mathtt{q}]\!] \cdot [\![\mathtt{p}]\!] \\
[\![\mathtt{if}\ \mathtt{b}\ \mathtt{then}\ \mathtt{p}\ \mathtt{else}\ \mathtt{q}]\!] &= [[\![\mathtt{p}]\!], [\![\mathtt{q}]\!]] \cdot \mathrm{dist} \cdot \langle [\![\mathtt{b}]\!], \mathrm{id} \rangle \\
[\![\mathtt{while}\ \mathtt{b}\ \mathtt{do}\ \{\ \mathtt{p}\ \}]\!] &= [[\![\mathtt{while}\ \mathtt{b}\ \mathtt{do}\ \{\ \mathtt{p}\ \}]\!] \cdot [\![\mathtt{p}]\!], \mathrm{id}] \cdot \mathrm{dist} \cdot \langle [\![\mathtt{b}]\!], \mathrm{id} \rangle
\end{aligned}$$

Programs and a (tentative) denotational semantics

$$\mathrm{p} ::= \mathrm{x} := \mathrm{e} \mid \mathrm{p} \mathbin{\texttt{;}} \mathrm{p} \mid \texttt{if}\ \mathrm{b}\ \texttt{then}\ \mathrm{p}\ \texttt{else}\ \mathrm{p} \mid \texttt{while}\ \mathrm{b}\ \texttt{do}\ \{\ \mathrm{p}\ \}$$

$$\begin{aligned}
[\![\mathrm{x} := \mathrm{e}]\!] &= \sigma \mapsto \sigma[[\![\mathrm{e}]\!]/\mathrm{x}] \\
[\![\mathrm{p} \mathbin{\texttt{;}} \mathrm{q}]\!] &= [\![\mathrm{q}]\!] \cdot [\![\mathrm{p}]\!] \\
[\![\texttt{if}\ \mathrm{b}\ \texttt{then}\ \mathrm{p}\ \texttt{else}\ \mathrm{q}]\!] &= [[\![\mathrm{p}]\!], [\![\mathrm{q}]\!]] \cdot \mathrm{dist} \cdot \langle [\![\mathrm{b}]\!], \mathrm{id} \rangle \\
[\![\texttt{while}\ \mathrm{b}\ \texttt{do}\ \{\ \mathrm{p}\ \}]\!] &= [[\![\texttt{while}\ \mathrm{b}\ \texttt{do}\ \{\ \mathrm{p}\ \}]\!] \cdot [\![\mathrm{p}]\!], \mathrm{id}] \cdot \mathrm{dist} \cdot \langle [\![\mathrm{b}]\!], \mathrm{id} \rangle
\end{aligned}$$

I'm very clear, Brexit does mean brexit

(Theresa May) https://www.youtube.com/watch?v=oRDfFJAu6Bo

Table of Contents

Partially ordered set

Definition (Poset)

A set with a reflexive, anti-symmetric, and transitive relation $\leq$

Examples

- $(\mathbb{N}, $ the usual order $\leq$ on natural numbers$)$
- $(\mathbb{R}, $ the usual order $\leq$ on the real numbers$)$
- $(X, =)$ (for any set X)

Partially ordered set

Definition (Poset)

A set with a reflexive, anti-symmetric, and transitive relation $\leq$

Examples

- $(\mathbb{N},$ the usual order $\leq$ on natural numbers$)$
- $(\mathbb{R},$ the usual order $\leq$ on the real numbers$)$
- $(X, =)$ (for any set X)

In our context $x \leq y$ reads as

$$\boxed{y \text{ more informative than } x}$$

New posets from old ones

Addition of a bottom element

If $(X, \leq_X)$ is a poset then $(X_\perp, \leq)$ is a poset when defined as

- $x_1 \leq x_2$ iff $x_1 \leq_X x_2$
- $\perp \leq x$ (for all $x \in X$)

$\perp$ is the least informative element, akin to non-termination

New posets from old ones

Addition of a bottom element

If $(X, \leq_X)$ is a poset then $(X_\perp, \leq)$ is a poset when defined as

- $x_1 \leq x_2$ iff $x_1 \leq_X x_2$
- $\perp \leq x$ (for all $x \in X$)

$\perp$ is the least informative element, akin to non-termination

Example

In what way is $State_\perp$ a poset ?

Data aggregation

We will often wish to collect an increasing seq. of information

$$x_1 \leq x_2 \leq x_3 \leq \dots$$

into a single datum, denoted by $\bigvee_{i \in \mathbb{N}} x_i$

Data aggregation

We will often wish to collect an increasing seq. of information

$$x_1 \leq x_2 \leq x_3 \leq \dots$$

into a single datum, denoted by $\bigvee_{i \in \mathbb{N}} x_i$

This element should be more informative than any x_j ($j \in \mathbb{N}$), *i.e.*

$$x_j \leq \bigvee_{i \in \mathbb{N}} x_i$$

and contain no more information than the one in the chain, *i.e.*

$$(\forall j \in \mathbb{N}.\, x_j \leq y) \implies \bigvee_{i \in \mathbb{N}} x_i \leq y$$

Definition (ω-CPO)

A poset with data aggregation as previously described

Examples

- $\mathbb{N}$ is not an ω-CPO but $\mathbb{N} \cup \{\infty\}$ is
- $\mathbb{R}$ is not an ω-CPO but $\mathbb{R} \cup \{\infty\}$ and $[0, 1]$ are

Definition (ω-CPO)

A poset with data aggregation as previously described

Examples

- $\mathbb{N}$ is not an ω-CPO but $\mathbb{N} \cup \{\infty\}$ is
- $\mathbb{R}$ is not an ω-CPO but $\mathbb{R} \cup \{\infty\}$ and $[0, 1]$ are

Exercise

Show that $State_\perp$ is an ω-CPO