Denotational Semantics

Renato Neves

Universidade do Minho

Table of Contents

Outline

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

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

Compiler correctness and contextual equivalence

We adopted the following notion of equivalence

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

Compilers adopt the stronger version

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

Compiler correctness and contextual equivalence

We adopted the following notion of <u>equivalence</u>

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

Compilers adopt the <u>stronger</u> version

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

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

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

Exercise

Prove the equivalence $b_1 \equiv_o b_2 \iff b_1 \equiv b_2$

Homework: repeat the exercise now for arithmetic expressions

Contextual equivalence

Contexts

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

Contextual equivalence

Contexts

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

Can we still prove $\mathrm{p} \equiv_o \mathrm{q} \Longleftrightarrow \mathrm{p} \equiv \mathrm{q}$?

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

$$b ::= x \mid b \wedge b \mid \neg b$$

Terms interpreted as <u>functions</u> $[\![b]\!] : State \to 2$

Term operations interpreted via the <u>boolean algebra</u> 2

$$[\![x]\!](\sigma) = \sigma(x)$$
$$[\![b_1 \wedge b_2]\!] = (\wedge) \cdot \langle [\![b_1]\!], [\![b_2]\!] \rangle$$
$$[\![\neg b]\!] = (\neg) \cdot [\![b]\!]$$

The relation between big-step and denotational semantics

Theorem

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

Proof.

Straightforward <u>induction</u> □

Corollary

$b_1 \equiv b_2$ *iff* $b_1 \equiv_o b_2$ *iff* $[\![b_1]\!] = [\![b_2]\!]$

Profits!

We can now reduce checking for equivalence to . . .

> Program calculus and Boolean algebra

Profits!

We can now reduce checking for equivalence to . . .

Program calculus and Boolean algebra

Example

$$
\begin{aligned}
[\![b_1 \wedge b_2]\!] &= (\wedge) \cdot \langle [\![b_1]\!], [\![b_2]\!] \rangle \\
&= (\wedge) \cdot \mathrm{sw} \cdot \langle [\![b_1]\!], [\![b_2]\!] \rangle \\
&= (\wedge) \cdot \langle \pi_2, \pi_1 \rangle \cdot \langle [\![b_1]\!], [\![b_2]\!] \rangle \\
&= (\wedge) \cdot \langle \pi_2 \cdot \langle [\![b_1]\!], [\![b_2]\!] \rangle, \pi_1 \cdot \langle [\![b_1]\!], [\![b_2]\!] \rangle \rangle \\
&= (\wedge) \cdot \langle [\![b_2]\!], [\![b_1]\!] \rangle \\
&= [\![b_2 \wedge b_1]\!]
\end{aligned}
$$

1. Show that $b \wedge b \equiv b$ via the denotational semantics
2. Define a denotational semantics for arithmetic expressions e
3. Prove that $[\![e_1 + e_2]\!] = [\![e_2 + e_1]\!]$
4. Prove the equivalence $\langle e, \sigma \rangle \Downarrow v$ iff $[\![e]\!](\sigma) = v$

Table of Contents

Programs interpreted as <u>functions</u> $[\![p]\!] : State_\bot \to State_\bot$

$State_\bot = State \cup \{\bot\}$ where $\bot$ represents <u>non-termination</u>

Sequential composition is <u>function composition</u>

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

$$[\![x := e]\!] = \sigma \mapsto \sigma[[\![e]\!]/x]$$

$$[\![p\,;q]\!] = [\![q]\!] \cdot [\![p]\!]$$

$$[\![\texttt{if } b \texttt{ then } p \texttt{ else } p]\!] = [[\![p]\!], [\![q]\!]] \cdot \mathrm{dist} \cdot \langle [\![b]\!], \mathrm{id} \rangle$$

$$[\![\texttt{while } b \texttt{ do } \{\,p\,\}]\!] = \ \ldots\ldots$$

The relation between big-step and denotational semantics

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

Theorem

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

Proof.

Straightforward induction $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad$ $\square$

Corollary

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

Profits!

Recall when we had to prove the two equivalences

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

with the big-step semantics

Show the same via the denotational semantics

Programs and a (tentative) denotational semantics

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

$$[\![\texttt{x} := \texttt{e}]\!] = \sigma \mapsto \sigma[[\![\texttt{e}]\!]/\texttt{x}]$$

$$[\![\texttt{p} \,;\, \texttt{q}]\!] = [\![\texttt{q}]\!] \cdot [\![\texttt{p}]\!]$$

$$[\![\texttt{if } \texttt{b} \texttt{ then } \texttt{p} \texttt{ else } \texttt{p}]\!] = [[\![\texttt{p}]\!], [\![\texttt{q}]\!]] \cdot \mathrm{dist} \cdot \langle [\![\texttt{b}]\!], \mathrm{id} \rangle$$

$$[\![\texttt{while } \texttt{b} \texttt{ do } \{\, \texttt{p} \,\}]\!] = [[\![\texttt{while } \texttt{b} \texttt{ do } \{\, \texttt{p} \,\}]\!] \cdot [\![\texttt{p}]\!], \mathrm{id}] \cdot \mathrm{dist} \cdot \langle [\![\texttt{b}]\!], \mathrm{id} \rangle$$

Programs and a (tentative) denotational semantics

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

$$\llbracket x := e \rrbracket = \sigma \mapsto \sigma[\llbracket e \rrbracket / x]$$

$$\llbracket p\,;q \rrbracket = \llbracket q \rrbracket \cdot \llbracket p \rrbracket$$

$$\llbracket \text{if } b \text{ then } p \text{ else } p \rrbracket = [\llbracket p \rrbracket, \llbracket q \rrbracket] \cdot \text{dist} \cdot \langle \llbracket b \rrbracket, \text{id} \rangle$$

$$\llbracket \text{while } b \text{ do } \{\, p\, \} \rrbracket = [\llbracket \text{while } b \text{ do } \{\, p\, \} \rrbracket \cdot \llbracket p \rrbracket, \text{id}] \cdot \text{dist} \cdot \langle \llbracket b \rrbracket, \text{id} \rangle$$

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

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

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

$$x_1 \leq x_2 \leq x_3 \leq \ldots$$

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

Data aggregation

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

$$x_1 \leq x_2 \leq x_3 \leq \ldots$$

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

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

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

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

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

Posets + Data aggregation

Definition (ω-CPO)

A poset with data aggregation as previously described

Examples

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

Posets + Data aggregation

Definition (ω-CPO)

A poset with data aggregation as previously described

Examples

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

Exercise

Show that $State_\perp$ is an ω-CPO