Denotational Semantics

Renato Neves

Universidade do Minho

HASLab
HIGH-ASSURANCE
SOFTWARE LABORATORY

Table of Contents

Outline

Semantics for every season

Operational semantics	How a program operates
<u>Denotational semantics</u>	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 <u>equivalence</u>

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

Compiler correctness and contextual equivalence

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

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

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

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

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

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

Why is that ?

Contexts

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

Exercise

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

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

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

Contextual equivalence

Contexts

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

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

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

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* $[\![b_1]\!] = [\![b_2]\!]$

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

$$\llbracket b_1 \wedge b_2 \rrbracket = (\wedge) \cdot \langle \llbracket b_1 \rrbracket, \llbracket b_2 \rrbracket \rangle$$
$$= (\wedge) \cdot \mathrm{sw} \cdot \langle \llbracket b_1 \rrbracket, \llbracket b_2 \rrbracket \rangle$$
$$= (\wedge) \cdot \langle \pi_2, \pi_1 \rangle \cdot \langle \llbracket b_1 \rrbracket, \llbracket b_2 \rrbracket \rangle$$
$$= (\wedge) \cdot \langle \pi_2 \cdot \langle \llbracket b_1 \rrbracket, \llbracket b_2 \rrbracket \rangle, \pi_1 \cdot \langle \llbracket b_1 \rrbracket, \llbracket b_2 \rrbracket \rangle \rangle$$
$$= (\wedge) \cdot \langle \llbracket b_2 \rrbracket, \llbracket b_1 \rrbracket \rangle$$
$$= \llbracket b_2 \wedge b_1 \rrbracket$$

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

Key takeaways

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

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

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

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

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

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

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

$$\llbracket \text{while } b \text{ do } \{\, p \,\} \rrbracket = \ldots\ldots$$

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 ☐

Corollary

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

Profits!

Recall when we had to prove the two equivalences

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

with the big-step semantics

Show the same via the denotational semantics

Programs and a (tentative) denotational semantics

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

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

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

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

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

Programs and a (tentative) denotational semantics

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

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

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

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

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

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 <u>more informative</u> 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

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

Data aggregation

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

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

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.*

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

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

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

Maps between ω-**CPOs**

We wish them to represent some form of computability ...

... and thus we cannot allow all maps

Maps between ω-**CPOs**

We wish them to represent some form of computability ...

... and thus we cannot allow all maps

We enforce instead the following laws

$$f(\vee_n x_n) = \vee_n f(x_n) \qquad \text{(continuity)}$$
$$x_1 \leq x_2 \Rightarrow f(x_1) \leq f(x_2) \qquad \text{(monotonicity)}$$

Continuity $\simeq$ Computability ?

> What does it mean for $p : X \to \{\bot \leq \top\}$ to be continuous ?

Suppose $x \in X$ is given by a chain of <u>finite</u> approximations

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

... then deduce that

$$p(\vee_{n \in \mathbb{N}} x_n) = \top \iff \vee_{n \in \mathbb{N}} p(x_n) = \top$$
$$\iff \exists n \in \mathbb{N}. \, p(x_n) = \top$$

What does it mean for $p : X \to \{\bot \leq \top\}$ to be continuous ?

Suppose $x \in X$ is given by a chain of <u>finite</u> approximations

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

. . . then deduce that

$$p(\vee_{n \in \mathbb{N}} x_n) = \top \iff \vee_{n \in \mathbb{N}} p(x_n) = \top$$
$$\iff \exists n \in \mathbb{N}. \, p(x_n) = \top$$

i.e. that p <u>terminates</u> with tt for x iff p can evaluate a <u>finite</u> approximation of x to tt

Continuity $\simeq$ Computability ?

Exercise 1

Show that $(\mathcal{P}(\mathbb{N}), \subseteq)$ is an ω-CPO

Exercise 2

Can isInfinite $: \mathcal{P}(\mathbb{N}) \to \{\bot \leq \top\}$ be continuous ?

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

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

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

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

Are all programs $[\![p]\!]$ continuous ?

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

Are all programs $[\![p]\!]$ continuous ?

Yes !! Just use <u>program calculus</u> 'with <u>continuous</u> functions'

Fixpoints for while-loops

Definition

An element $x \in X$ is a fixpoint of $f : X \to X$ if $f(x) = x$

A notion with applications in different fields

- economics (game theory)
- dynamical systems (equilibrium points)
- automata theory
- essentially everywhere . . .

Fixpoints for while-loops

Definition

An element $x \in X$ is a <u>fixpoint</u> of $f : X \to X$ if $f(x) = x$

A notion with applications in different fields

- economics (game theory)
- dynamical systems (equilibrium points)
- automata theory
- essentially everywhere . . .

Here while-loops will be fixpoints

. . . but a fixpoint of which function ?

Fixpoints for while-loops

> ... but a fixpoint of which function ?

Recall our previous idea

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

It translates to saying that $[\![\texttt{while b do} \{ \texttt{p} \}]\!]$ is a fixpoint of

$$k \longmapsto [k \cdot [\![\texttt{p}]\!], \mathrm{id}] \cdot \mathrm{dist} \cdot \langle [\![\texttt{b}]\!], \mathrm{id} \rangle$$

The least fixpoint theorem

Theorem

Every continuous, monotone map $f : X \to X$ has a least fixpoint

$$\text{lfp } f = \vee_{n \in \mathbb{N}} f^n(\bot)$$

Exercise

Prove the theorem

The least fixpoint theorem

Theorem

Every continuous, monotone map $f : X \to X$ has a least fixpoint

$$\mathrm{lfp}\ f = \vee_{n \in \mathbb{N}}\ f^n(\bot)$$

Exercise

Prove the theorem

And finally . . .

Programs and a denotational semantics

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

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

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

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

$$[\![\mathtt{while}\ \mathtt{b}\ \mathtt{do}\ \{\,\mathtt{p}\,\}]\!] = \mathrm{lfp}\ \Big(k \mapsto [k \cdot [\![\mathtt{p}]\!], \mathrm{id}] \cdot \mathrm{dist} \cdot \langle [\![\mathtt{b}]\!], \mathrm{id}\rangle\Big)$$

Prove the following equivalences

- while b $\{p\} \equiv$ if b then p else skip

- while b $\{p\}$; q $\equiv$ if b then p ; while b $\{p\}$ else q

- while ff $\{p\}$; q $\equiv$ q

- while tt $\{p\} \equiv$ while tt $\{q\}$

The relation between big-step and denotational semantics

Theorem

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

Corollary

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