Small-step 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

Small-step operational semantics

How a program operates step-by-step

Example

$\langle \mathrm{x} := 1; \mathrm{x} := \mathrm{x} + 1, v\rangle \longrightarrow \langle \mathrm{x} := \mathrm{x} + 1, 1\rangle \longrightarrow 2$

A machine with an 'evaluation stack' that is processed at each step

Its uses

Describes how a program operates step-by-step

Describes evaluation techniques (*e.g.* short-circuiting)

A basis for tracing/debugging

Foundations of concurrency, complexity, . . .

Rich notions of equivalence

. . .

Its uses

Describes how a program operates step-by-step

Describes evaluation techniques (*e.g.* short-circuiting)

A basis for tracing/debugging

Foundations of concurrency, complexity, ...

Rich notions of equivalence

...

Thus an essential tool for understanding a programming language

Table of Contents

A propositional language

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

Every $\mathtt{x}$ is a proposition (*i.e.* it has either value $\mathtt{tt}$ or $\mathtt{ff}$)

A propositional language

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

Every $\mathtt{x}$ is a proposition (*i.e.* it has either value $\mathtt{tt}$ or $\mathtt{ff}$)

Can we provide a small-step semantics to this language ?

Key points

Uses a memory $\sigma : X \to \mathrm{Bool}$ that assigns to every proposition $\mathtt{x}$ its truth-value $\sigma(\mathtt{x})$

A term $\mathtt{b}$ is evaluated step-by-step until a truth-value v is reached

$$\bullet \longrightarrow \bullet \longrightarrow \cdots \longrightarrow \bullet \longrightarrow \bullet \longrightarrow v$$

Focus is on the next step (of the evaluation)

The semantics – a universe of laws

$$\frac{}{\langle \mathtt{x}, \sigma \rangle \longrightarrow \sigma(\mathtt{x})}\ (\mathsf{var}) \qquad\qquad \frac{\langle \mathtt{b}, \sigma \rangle \longrightarrow v}{\langle \neg \mathtt{b}, \sigma \rangle \longrightarrow \neg v}\ (\mathsf{neg}_1)$$

The semantics – a universe of laws

$$\frac{}{\langle \mathtt{x}, \sigma \rangle \longrightarrow \sigma(\mathtt{x})}\ (\mathsf{var}) \qquad \frac{\langle \mathtt{b}, \sigma \rangle \longrightarrow v}{\langle \neg\mathtt{b}, \sigma \rangle \longrightarrow \neg v}\ (\mathsf{neg}_1)$$

$$\frac{\langle \mathtt{b}, \sigma \rangle \longrightarrow \langle \mathtt{b}', \sigma' \rangle}{\langle \neg\mathtt{b}, \sigma \rangle \longrightarrow \langle \neg\mathtt{b}', \sigma' \rangle}\ (\mathsf{neg}_2) \qquad \frac{\langle \mathtt{b}_1, \sigma \rangle \longrightarrow \mathtt{ff}}{\langle \mathtt{b}_1 \wedge \mathtt{b}_2, \sigma \rangle \longrightarrow \mathtt{ff}}\ (\mathsf{and}_1)$$

$$\frac{\langle \mathtt{b}_1, \sigma \rangle \longrightarrow \mathtt{tt}}{\langle \mathtt{b}_1 \wedge \mathtt{b}_2, \sigma \rangle \longrightarrow \langle \mathtt{b}_2, \sigma \rangle}\ (\mathsf{and}_2) \qquad \frac{\langle \mathtt{b}_1, \sigma \rangle \longrightarrow \langle \mathtt{b}_1', \sigma' \rangle}{\langle \mathtt{b}_1 \wedge \mathtt{b}_2, \sigma \rangle \longrightarrow \langle \mathtt{b}_1' \wedge \mathtt{b}_2, \sigma' \rangle}\ (\mathsf{and}_3)$$

An example

$\neg\neg\mathtt{x} \longrightarrow \ ?$

An example

$\neg\neg\mathtt{x} \longrightarrow$?

$$\dfrac{\dfrac{\dfrac{}{\langle \mathtt{x}, \sigma \rangle \longrightarrow \ \sigma(\mathtt{x})}\ (\mathsf{var})}{\langle \neg\mathtt{x}, \sigma \rangle \longrightarrow \ \neg\sigma(\mathtt{x})}\ (\mathsf{neg}_1)}{\langle \neg\neg\mathtt{x}, \sigma \rangle \longrightarrow \ \neg\neg\sigma(\mathtt{x})}\ (\mathsf{neg}_1)$$

Another example

$(\mathtt{x} \wedge \mathtt{b}_1) \wedge \mathtt{b}_2 \longrightarrow ?$

If $\sigma(\mathtt{x}) = \mathtt{ff}$:

$$\dfrac{\dfrac{\dfrac{}{\langle \mathtt{x}, \sigma \rangle \longrightarrow \ \mathtt{ff}}\ (\mathsf{var})}{\langle \mathtt{x} \wedge \mathtt{b}_1, \sigma \rangle \longrightarrow \ \mathtt{ff}}\ (\mathsf{and}_1)}{\langle (\mathtt{x} \wedge \mathtt{b}_1) \wedge \mathtt{b}_2, \sigma \rangle \longrightarrow \ \mathtt{ff}}\ (\mathsf{and}_1)$$

Yet another example

$(\mathtt{x} \wedge \mathtt{b}_1) \wedge \mathtt{b}_2 \longrightarrow ?$

If $\sigma(\mathtt{x}) = \mathtt{tt}$:

$$\dfrac{\dfrac{\overline{\langle \mathtt{x}, \sigma \rangle \longrightarrow \ \mathtt{tt}}\ (\mathsf{var})}{\langle \mathtt{x} \wedge \mathtt{b}_1, \sigma \rangle \longrightarrow \ \langle \mathtt{b}_1, \sigma \rangle}\ (\mathsf{and}_2)}{\langle (\mathtt{x} \wedge \mathtt{b}_1) \wedge \mathtt{b}_2, \sigma \rangle \longrightarrow \ \langle \mathtt{b}_1 \wedge \mathtt{b}_2, \sigma \rangle}\ (\mathsf{and}_3)$$

Now you try !

$\mathtt{x} \wedge \neg \mathtt{x} \longrightarrow ?$

$\neg(\neg \mathtt{x} \wedge \neg \mathtt{y}) \longrightarrow ?$

Provide semantics to the Boolean implication $\mathtt{b} \Rightarrow \mathtt{b}$

From one step to many ...

One often is uninterested on the next step ...

... and rather on the output (that the sequence of steps leads to)

From one step to many ...

One often is uninterested on the next step ...

... and rather on the output (that the sequence of steps leads to)

This multi-step transition $\longrightarrow^n$ is defined by the rules

$$\frac{\langle \mathtt{b}, \sigma \rangle \longrightarrow v}{\langle \mathtt{b}, \sigma \rangle \longrightarrow^1 v}\ (\mathsf{stp}) \qquad \frac{\langle \mathtt{b}, \sigma \rangle \longrightarrow \langle \mathtt{b}', \sigma' \rangle \qquad \langle \mathtt{b}', \sigma' \rangle \longrightarrow^n v}{\langle \mathtt{b}, \sigma \rangle \longrightarrow^{n+1} v}\ (\mathsf{nxt})$$

What's next ?

Fine, we have an operational semantics; so what ?

What's next ?

Fine, we have an operational semantics; so what ?

We can now prove cool properties about our language !!

Example (Termination)

It is always the case that $\langle \mathrm{b}, \sigma \rangle \longrightarrow^{n} v$ for some v and n

Exercise 1

Define a ‘complexity function’

$$\begin{aligned} \mathrm{compl}(\mathtt{x}) &= 1 \\ \mathrm{compl}(\neg \mathtt{b}) &= \mathrm{compl}(\mathtt{b}) \\ \mathrm{compl}(\mathtt{b}_1 \wedge \mathtt{b}_2) &= \mathrm{compl}(\mathtt{b}_1) + \mathrm{compl}(\mathtt{b}_2) \end{aligned}$$

Show by induction that $\mathrm{compl}(\mathtt{b}) \geq 1$ for every $\mathtt{b}$

Exercise 2

Show by induction the following implication

$$\text{If } \langle \mathtt{b}, \sigma \rangle \longrightarrow \langle \mathtt{b}', \sigma' \rangle \text{ then } \mathrm{compl}(\mathtt{b}) > \mathrm{compl}(\mathtt{b}')$$

Recognising the pattern

Our induction proofs relied on

- a 'base' (or terminating) case
- assumption that hypothesis holds for the 'simpler parts' of the case at hand

Recognising the pattern

Our induction proofs relied on

- a '<u>base</u>' (or terminating) case
- assumption that hypothesis holds for the '<u>simpler parts</u>' of the case at hand

Often hard to see on which structure should induction be founded

- natural numbers
- syntactic structure of programs
- derivation trees
- ...

Keep in mind . . .

$\boxed{\text{Induction is a basic tool of every programming theorist}}$

Exercise 3

Show by induction the following implication

$$\text{If } \langle \mathtt{b}, \sigma \rangle \longrightarrow^{n} v \text{ then } \mathrm{compl}(\mathtt{b}) \geq n$$

Table of Contents

When the number of steps does not matter . . .

One often is uninterested on the number of steps . . .

. . . and rather just on the output

When the number of steps does not matter ...

One often is uninterested on the number of steps ...

... and rather just on the output

This multi-step transition $\longrightarrow^\star$ is defined by the rules

$$\frac{\langle \mathtt{b}, \sigma \rangle \longrightarrow v}{\langle \mathtt{b}, \sigma \rangle \longrightarrow^\star v} \text{ (stp)} \qquad \frac{\langle \mathtt{b}, \sigma \rangle \longrightarrow \langle \mathtt{b}', \sigma' \rangle \qquad \langle \mathtt{b}', \sigma' \rangle \longrightarrow^\star v}{\langle \mathtt{b}, \sigma \rangle \longrightarrow^\star v} \text{ (nxt)}$$

Exercise 4

Show by induction the following equivalence

$$\langle \mathtt{b}, \sigma \rangle \longrightarrow^{n} v \text{ (for some } n\text{) iff } \langle \mathtt{b}, \sigma \rangle \longrightarrow^{\star} v$$

Table of Contents

A simple while-language

Arithmetic expressions

$\mathtt{e} ::= \mathtt{n} \mid \mathtt{e} \cdot \mathtt{e} \mid \mathtt{x} \mid \mathtt{e} + \mathtt{e}$

Programs

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

A simple while-language

Arithmetic expressions

$\mathtt{e} ::= \mathtt{n} \mid \mathtt{e} \cdot \mathtt{e} \mid \mathtt{x} \mid \mathtt{e} + \mathtt{e}$

Programs

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

Homework: provide semantics to the arithmentic expressions

Key points

Similar to before but now with assignments, conditionals . . .

Unlike before memory can be altered throughout the computation

The output values will now be memories

Key points

Similar to before but now with assignments, conditionals ...

Unlike before memory can be altered throughout the computation

The output values will now be memories

We will use $\sigma[v/\mathtt{x}]$ to denote the memory that is like σ except for the fact that $\mathtt{x}$ has now value v

A while-language and its semantics

$$\frac{\langle \mathtt{e}, \sigma \rangle \longrightarrow^{\star} v}{\langle \mathtt{x} := \mathtt{e}, \sigma \rangle \longrightarrow \sigma[v/\mathtt{x}]} \text{ (asg)} \qquad \frac{\langle \mathtt{p}, \sigma \rangle \longrightarrow \sigma'}{\langle \mathtt{p} \, ; \, \mathtt{q}, \sigma \rangle \longrightarrow \langle \mathtt{q}, \sigma' \rangle} \text{ (seq}_1\text{)}$$

$$\frac{\langle \mathtt{p}, \sigma \rangle \longrightarrow \langle \mathtt{p}', \sigma' \rangle}{\langle \mathtt{p} \, ; \, \mathtt{q}, \sigma \rangle \longrightarrow \langle \mathtt{p}' \, ; \, \mathtt{q}, \sigma' \rangle} \text{ (seq}_2\text{)} \qquad \frac{\langle \mathtt{b}, \sigma \rangle \longrightarrow^{\star} \mathtt{tt}}{\langle \mathtt{if} \ \mathtt{b} \ \mathtt{then} \ \mathtt{p} \ \mathtt{else} \ \mathtt{q}, \sigma \rangle \longrightarrow \langle \mathtt{p}, \sigma \rangle} \text{ (if}_1\text{)}$$

$$\frac{\langle \mathtt{b}, \sigma \rangle \longrightarrow^{\star} \mathtt{ff}}{\langle \mathtt{if} \ \mathtt{b} \ \mathtt{then} \ \mathtt{p} \ \mathtt{else} \ \mathtt{q}, \sigma \rangle \longrightarrow \langle \mathtt{q}, \sigma \rangle} \text{ (if}_2\text{)} \qquad \frac{\langle \mathtt{b}, \sigma \rangle \longrightarrow^{\star} \mathtt{ff}}{\langle \mathtt{while} \ \mathtt{b} \ \mathtt{do} \ \{ \ \mathtt{p} \ \}, \sigma \rangle \longrightarrow^{\star} \sigma} \text{ (wh}_2\text{)}$$

$$\frac{\langle \mathtt{b}, \sigma \rangle \longrightarrow^{\star} \mathtt{tt}}{\langle \mathtt{while} \ \mathtt{b} \ \mathtt{do} \ \{ \ \mathtt{p} \ \}, \sigma \rangle \longrightarrow \langle \mathtt{p} \, ; \, \mathtt{while} \ \mathtt{b} \ \mathtt{do} \ \{ \ \mathtt{p} \ \}, \sigma \rangle} \text{ (wh}_1\text{)}$$

Exercises

1. Write down the sequence of steps that originates from

$$\langle \texttt{while}\,\texttt{tt}\,\texttt{do}\,\{\, \texttt{x} := \texttt{x} + 1 \,\}, \sigma \rangle$$

2. Conclude that our previous termination property was lost

Table of Contents

A simple concurrent language

Arithmetic expressions

$\mathrm{e} ::= \mathrm{n} \mid \mathrm{e} \cdot \mathrm{e} \mid \mathrm{x} \mid \mathrm{e} + \mathrm{e}$

Programs

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

A concurrent language and its semantics

$$\frac{\langle \mathtt{p}, \sigma \rangle \longrightarrow \sigma'}{\langle \mathtt{p} \parallel \mathtt{q}, \sigma \rangle \longrightarrow \langle \mathtt{q}, \sigma' \rangle}\ (\mathsf{par}_1) \qquad \frac{\langle \mathtt{q}, \sigma \rangle \longrightarrow \sigma'}{\langle \mathtt{p} \parallel \mathtt{q}, \sigma \rangle \longrightarrow \langle \mathtt{p}, \sigma' \rangle}\ (\mathsf{par}_2)$$

$$\frac{\langle \mathtt{p}, \sigma \rangle \longrightarrow \langle \mathtt{p}', \sigma' \rangle}{\langle \mathtt{p} \parallel \mathtt{q}, \sigma \rangle \longrightarrow \langle \mathtt{p}' \parallel \mathtt{q}, \sigma' \rangle}\ (\mathsf{par}_3) \qquad \frac{\langle \mathtt{q}, \sigma \rangle \longrightarrow \langle \mathtt{q}', \sigma' \rangle}{\langle \mathtt{p} \parallel \mathtt{q}, \sigma \rangle \longrightarrow \langle \mathtt{p} \parallel \mathtt{q}', \sigma' \rangle}\ (\mathsf{par}_4)$$

Exercises

1. Write down the possible outputs of

$$\langle\, (\mathtt{y} := \mathtt{y} + 1\ ;\mathtt{x} := \mathtt{x} + 1) \parallel \mathtt{x} := 0, \sigma \rangle$$

2. Conclude that our previous determinacy property was lost

Conclusions

We (briefly) studied our first style of semantics

The gist: it describes how a program operates step-by-step

We also saw how valuable induction is in our context

Conclusions

We (briefly) studied our first style of semantics

The gist: it describes how a program operates step-by-step

We also saw how valuable induction is in our context

Further details about small-step semantics and induction can be consulted *e.g.* in [Rey98, Chapter 6] and [Win93, Chapter 3] respectively

John C Reynolds, *Theories of programming languages*, Cambridge University Press, 1998.

Glynn Winskel, *The formal semantics of programming languages - an introduction*, Foundation of computing series, MIT Press, 1993.