COMP3610/6361
Principles of Programming Languages

Peter Höfner

Jul 26, 2023

Section 2

IMP
and its Operational Semantics

‘Toy’ languages

- real programming languages are large
  many features, redundant constructs
- focus on particular aspects and abstract from others (scale up later)
- even small languages can involve delicate design choices.

Design choices, from Micro to Macro

- basic values
- evaluation order
- what is guaranteed at compile-time and run-time
- how effects are controlled
- how concurrency is supported
- how information hiding is enforceable
- how large-scale development and re-use are supported
- . . .

IMP[1]– Introductory Example

IMP is an imperative language with store locations, conditionals and while loop.
For example

$$
\begin{array}{l}
l_2 := 0\ ;\\
\textbf{while}\ \ !l_1 \geq 1\ \textbf{do}\ (\\
\quad l_2 :=\ !l_2 +\ !l_1\ ;\\
\quad l_1 :=\ !l_1 + -1\\
)
\end{array}
$$

with initial store $\{l_1 \mapsto 3, l_2 \mapsto 0\}$.

[1]Basically the same as in Winskel 1993 (IMP) and in Hennessy 1990 (WhileL)

IMP – Syntax

Booleans	$b \in \mathbb{B} = \{\texttt{true}, \texttt{false}\}$
Integers (Values)	$n \in \mathbb{Z} = \{\ldots, -1, 0, 1, \ldots\}$
Locations	$l \in \mathbb{L} = \{l, l_0, l_1, l_2, \ldots\}$

Operations $\quad op ::= \; + \mid \geq$

Expressions

$$
\begin{aligned}
E ::= {} & n \mid b \mid E \; op \; E \mid \\
& l := E \mid \; !l \mid \\
& \textbf{skip} \mid E \; ; E \mid \\
& \textbf{if } E \textbf{ then } E \textbf{ else } E \\
& \textbf{while } E \textbf{ do } E
\end{aligned}
$$

Transition systems

A *transition system* consists of

- a set Config of configurations (or states), and
- a binary relation $\longrightarrow \subseteq \text{Config} \times \text{Config}$.

The relation $\longrightarrow$ is called the transition or reduction relation:
$c \longrightarrow c'$ reads as ‘state c can make a transition to state c'’.
(see DFA/NFA)

IMP Semantics (1 of 4) – Configurations

Stores are (finite) partial functions $\mathbb{L} \rightharpoonup \mathbb{Z}$.
For example, $\{l_1 \mapsto 3, l_3 \mapsto 42\}$

Configurations are pairs $\langle E \, , \, s \rangle$ of an expression E and a store s.
For example, $\langle l := 2 + \; !l \, , \, \{l \mapsto 3\} \rangle$.

Transitions have the form $\langle E \, , \, s \rangle \longrightarrow \langle E' \, , \, s' \rangle$.
For example, $\langle l := 2 + \; !l \, , \, \{l \mapsto 3\} \rangle \longrightarrow \langle l := 2 + 3 \, , \, \{l \mapsto 3\} \rangle$

Transitions – Examples

Transitions are single computation steps.
For example

$$
\begin{aligned}
& \langle l := 2 + \ !l \, , \{l \mapsto 3\}\rangle \\
\longrightarrow\ & \langle l := 2 + 3 \, , \{l \mapsto 3\}\rangle \\
\longrightarrow\ & \langle l := 5 \, , \{l \mapsto 3\}\rangle \\
\longrightarrow\ & \langle \textbf{skip} \, , \{l \mapsto 5\}\rangle \\
\not\longrightarrow\ &
\end{aligned}
$$

Keep going until reaching a value v, an expression in $\mathbb{V} = \mathbb{B} \cup \mathbb{Z} \cup \{\textbf{skip}\}$.
A configuration $\langle E \, , s\rangle$ is stuck if E is not a value and $\langle E \, , s\rangle \not\longrightarrow$.

IMP Semantics (2 of 4) – Rules (basic operations)

(op+) $\langle n_1 + n_2 \, , \, s\rangle \longrightarrow \langle n \, , \, s\rangle$ if $n = n_1 + n_2$

(op≥) $\langle n_1 \geq n_2 \, , \, s\rangle \longrightarrow \langle b \, , \, s\rangle$ if $b = (n_1 \geq n_2)$

(op1) $$\frac{\langle E_1 \, , \, s\rangle \longrightarrow \langle E_1' \, , \, s'\rangle}{\langle E_1 \; op \; E_2 \, , \, s\rangle \longrightarrow \langle E_1' \; op \; E_2 \, , \, s'\rangle}$$

(op2) $$\frac{\langle E_2 \, , \, s\rangle \longrightarrow \langle E_2' \, , \, s'\rangle}{\langle v \; op \; E_2 \, , \, s\rangle \longrightarrow \langle v \; op \; E_2' \, , \, s'\rangle}$$

Rules (basic operations) – Examples

Find the possible sequences of transitions for

$$\langle (2 + 3) + (4 + 5) \, , \, \emptyset \rangle$$

The answer is 14 – but how do we show this formally?

IMP Semantics (3 of 4) – Store and Sequencing

(deref) $\langle\, !l\,,\, s\rangle \longrightarrow \langle n\,,\, s\rangle$ if $l \in \mathrm{dom}(s)$ and $s(l) = n$

(assign1) $\langle l := n\,,\, s\rangle \longrightarrow \langle \textbf{skip}\,,\, s + \{l \mapsto n\}\rangle$

(assign2) $$\frac{\langle E\,,\, s\rangle \longrightarrow \langle E'\,,\, s'\rangle}{\langle l := E\,,\, s\rangle \longrightarrow \langle l := E'\,,\, s'\rangle}$$

(seq1) $\langle \textbf{skip}; E_2\,,\, s\rangle \longrightarrow \langle E_2\,,\, s\rangle$

(seq2) $$\frac{\langle E_1\,,\, s\rangle \longrightarrow \langle E_1'\,,\, s'\rangle}{\langle E_1; E_2\,,\, s\rangle \longrightarrow \langle E_1'; E_2\,,\, s'\rangle}$$

Australian National University

Store and Sequencing – Examples

$$
\begin{aligned}
\langle l := 3 \,;\; !l \,,\, \{l \mapsto 0\}\}\rangle &\longrightarrow \langle \textbf{skip} \,;\; !l \,,\, \{l \mapsto 3\}\rangle \\
&\longrightarrow \langle\, !l \,,\, \{l \mapsto 3\}\rangle \\
&\longrightarrow \langle 3 \,,\, \{l \mapsto 3\}\rangle
\end{aligned}
$$

Australian National University

Store and Sequencing – Examples

$$\langle l := 3\, ;\, l := \,!l\, ,\, \{l \mapsto 0\}\rangle \longrightarrow \quad ?$$

$$\langle 42 + \,!l\, ,\, \emptyset\rangle \longrightarrow \quad ?$$

IMP Semantics (4 of 4) – Conditionals and While

(if1) $$\langle \textbf{if } \texttt{true} \textbf{ then } E_2 \textbf{ else } E_3 \, , \, s \rangle \longrightarrow \langle E_2 \, , \, s \rangle$$

(if2) $$\langle \textbf{if } \texttt{false} \textbf{ then } E_2 \textbf{ else } E_3 \, , \, s \rangle \longrightarrow \langle E_3 \, , \, s \rangle$$

(if3) $$\frac{\langle E_1 \, , \, s \rangle \longrightarrow \langle E_1' \, , \, s' \rangle}{\langle \textbf{if } E_1 \textbf{ then } E_2 \textbf{ else } E_3 \, , \, s \rangle \longrightarrow \langle \textbf{if } E_1' \textbf{ then } E_2 \textbf{ else } E_3 \, , \, s' \rangle}$$

(while)

$$\langle \textbf{while } E_1 \textbf{ do } E_2 \, , \, s \rangle \longrightarrow \langle \textbf{if } E_1 \textbf{ then } (E_2 \, ; \textbf{while } E_1 \textbf{ do } E_2) \textbf{ else skip} \, , \, s \rangle$$

IMP – Examples

If

$$E = \big(l_2 := 0 \; ; \textbf{while} \; !l_1 \geq 1 \; \textbf{do} \; (l_2 := \; !l_2 + \; !l_1 \; ; l_1 := \; !l_1 + -1)\big)$$
$$s = \{l_1 \mapsto 3, l_2 \mapsto 0\}$$

then

$$\langle E \, , \, s \rangle \longrightarrow^* \quad ?$$

Determinacy

Theorem (Determinacy)

If $\langle E\,,\,s\rangle \longrightarrow \langle E_1\,,\,s_1\rangle$ *and* $\langle E\,,\,s\rangle \longrightarrow \langle E_2\,,\,s_2\rangle$
then $\langle E_1\,,\,s_1\rangle = \langle E_2\,,\,s_2\rangle$*.*

Proof.

later $\square$

Reminder

- basic and simple imperative while-language
- with *formal* semantics
- given in the format structural operational semantics
- rules usually have the form $\dfrac{A \qquad B}{C}$

  (special rule is $\overline{C}$, which we often write as C)
- derivation tree

$$\text{(R1)}\ \dfrac{\text{(R3)}\ \dfrac{}{A} \qquad \dfrac{\text{(R4)}\ \dfrac{}{B_1} \quad \text{(R5)}\ \dfrac{}{B_2}}{B}\ \text{(R2)}}{C}$$

Language design I

Order of Evaluation

IMP uses left-to-right evaluation. For example

$$\langle (l := 1 ; 0) + (l := 2 ; 0) , \{l \mapsto 0\} \rangle \longrightarrow^5 \langle 0 , \{l \mapsto \mathbf{2}\} \rangle$$

For right-to-left we could use

(op1') $$\frac{\langle E_2 , s \rangle \longrightarrow \langle E_2' , s' \rangle}{\langle E_1 \ op \ E_2 , s \rangle \longrightarrow \langle E_1 \ op \ E_2' , s' \rangle}$$

(op2') $$\frac{\langle E_1 , s \rangle \longrightarrow \langle E_1' , s' \rangle}{\langle E_1 \ op \ v , s \rangle \longrightarrow \langle E_1' \ op \ v , s' \rangle}$$

In this language

$$\langle (l := 1 ; 0) + (l := 2 ; 0) , \{l \mapsto 0\} \rangle \longrightarrow^5 \langle 0 , \{l \mapsto \mathbf{1}\} \rangle$$

Language design II

Assignment results

Recall

(assign1) $\langle l := n\,,\, s\rangle \longrightarrow \langle \textbf{skip}\,,\, s + \{l \mapsto n\}\rangle$ if $l \in \mathrm{dom}(s)$

(seq1) $\langle \textbf{skip}\ ;\, E_2\,,\, s\rangle \longrightarrow \langle E_2\,,\, s\rangle$

We have chosen to map an assignment to **skip**, and $e_1\ ;\, e_2$ to progress iff $e_1 = \textbf{skip}$.

Instead we could have chosen the following.

(assign1') $\langle l := n\,,\, s\rangle \longrightarrow \langle n\,,\, s + \{l \mapsto n\}\rangle$ if $l \in \mathrm{dom}(s)$

(seq1') $\langle v\ ;\, E_2\,,\, s\rangle \longrightarrow \langle E_2\,,\, s\rangle$

Language design III

Store initialisation

Recall

(deref) $\langle\, !l\,,\, s\rangle \longrightarrow \langle n\,,\, s\rangle$ if $l \in \mathrm{dom}(s)$ and $s(l) = n$

Assumes $l \in \mathrm{dom}(s)$.

Instead we could have

- initialise *all* locations to 0, or
- allow assignments to an $l \notin \mathrm{dom}(s)$.

Language design IV

Storable values

- our language only allows integer values (store: $\mathbb{L} \rightharpoonup \mathbb{Z}$)
- could we store any value? Could we store locations, or even programs?
- store is global and cannot create new locations

Language design V

Operators and Basic values

- Booleans are different from integers (unlike in C)
- Implementation is (probably) different to semantics
  Exercise: fix the semantics to match 32-bit integers

Expressiveness

Is our language expressive enough to write ‘interesting’ programs?

- **yes**: it is Turing-powerful
  Exercise: try to encode an arbitrary Turing machine in IMP
- **no**: no support for standard feature, such as functions, lists, trees, objects, modules, …

Is the language too expressive?

- **yes**: we would like to exclude programs such as $3 + \texttt{true}$
  clearly 3 and `true` are of different type