Program Analysis

Operational Semantics (Part 2)

Prof. Dr. Michael Pradel

Software Lab, University of Stuttgart

Winter 2023/2024

Warm-up Quiz

What does this Python code print?

```
foo = 42
bar = "hello"

if foo is not 23:
    if bar is (not None):
        print("a")
    else:
        print("b")
else:
    print("c")
```

Warm-up Quiz

What does this Python code print?

```
foo = 42
bar = "hello"

if foo is not 23:
    if bar is (not None):
        print("a")
    else:
        print("b")
else:
    print("c")
```

Answer: b

Warm-up Quiz

What does this Python code print?

```
foo = 42
bar = "hello"

if foo is not 23:
    if bar is (not None):
        print("a")
    else:
        print("b")
else:
    print("c")
```

- `is not` is a single operator
- Evaluates to `True` if both operands are the same object

Answer: b

Warm-up Quiz

What does this Python code print?

```
foo = 42
bar = "hello"

if foo is not 23:
    if bar is (not None):
        print("a")
    else:
        print("b")
else:
    print("c")
```

`not None` evaluates to `True` because `None` is coerced to `False`

Answer: b

Semantics of SIMP Expressions

Value of E in state with memory m is v if there is a sequence of transitions

$$\langle E \circ c, r, m \rangle \longrightarrow^{*} \langle c, v \circ r, m' \rangle$$

m = m' in SIMP

(similar for Boolean expr.)

Example: What's the value of $E = !a + !b$ in state $m = \{a \mapsto 3, b \mapsto 1\}$?

$\langle\, !a + !b \circ nil,\ nil,\ m \,\rangle$

$\rightarrow \langle\, !a \circ !b \circ + \circ nil,\ nil,\ m \,\rangle$

$\rightarrow \langle\, !b \circ + \circ nil,\ 3 \circ nil,\ m \,\rangle$

$\rightarrow \langle\, + \circ nil,\ 1 \circ 3 \circ nil,\ m \,\rangle$

$\rightarrow \langle\, nil,\ \underline{\underline{4}} \circ nil,\ m \,\rangle$

Answer: 4

b) Executing commands

$$\langle \text{skip} \circ c, r, m \rangle \longrightarrow \langle c, r, m \rangle$$

$$\langle (\underbrace{\ell := E) \circ c, r, m \rangle \longrightarrow \langle E \circ := \circ c, \ell}_{\text{push } \ell \text{ onto result stack}} \circ r, m \rangle$$

$$\langle := \circ c, n \circ \ell \circ r, m \rangle \longrightarrow \langle c, r, m[\ell \mapsto n] \rangle$$

$$\langle (C_1 ; C_2) \circ c, r, m \rangle \longrightarrow \langle C_1 \circ C_2 \circ c, r, m \rangle$$

$$\langle (\text{if } B \text{ then } C_1 \text{ else } C_2) \circ c,\ r,\ m \rangle$$

$$\to \langle B \circ \text{if} \circ c,\ \underbrace{C_1 \circ C_2}_{\text{store commands for later}} \circ r,\ m \rangle$$

$$\langle \text{if} \circ c,\ \text{True} \circ C_1 \circ C_2 \circ r,\ m \rangle$$

$$\to \langle C_1 \circ c,\ r,\ m \rangle$$

$$\langle \text{if} \circ c,\ \text{False} \circ C_1 \circ C_2 \circ r,\ m \rangle$$

$$\to \langle C_2 \circ c,\ r,\ m \rangle.$$

execute either then-branch or else-branch

$$\langle (\text{while } B \text{ do } C) \circ c,\ r,\ m \rangle$$

$$\longrightarrow \langle B \circ \text{while} \circ c,\ B \circ C \circ r,\ m \rangle$$

$$\langle \text{while} \circ c,\ \text{True} \circ B \circ C \circ r,\ m \rangle$$

$$\longrightarrow \langle C \circ (\text{while } B \text{ do } C) \circ c,\ r,\ m \rangle$$

$$\langle \text{while} \circ c,\ \text{False} \circ B \circ C \circ r,\ m \rangle$$

$$\longrightarrow \langle c,\ r,\ m \rangle .$$

Semantics of SIMP Commands

Program C executed in state with memory m terminates successfully & produces memory m' if there is a sequ. of transitions

$$\langle C \circ nil, nil, m \rangle \longrightarrow^{*} \langle nil, nil, m' \rangle$$

Example: $C = \text{while } !l > 0 \text{ do } (\underbrace{f := !f * !l \; ; \; l := !l - 1}_{C'})$

$m = \{ l \mapsto 4, \; f \mapsto 1 \}$

$\langle C \circ \text{nil}, \; \text{nil}, \; m \rangle$

$\longrightarrow \langle !l > 0 \circ \text{while} \circ \text{nil}, \; !l > 0 \circ C' \circ \text{nil}, \; m \rangle$

$\longrightarrow \langle !l \circ 0 \circ > \circ \text{while} \circ \text{nil}, \; !l > 0 \circ C' \circ \text{nil}, \; m \rangle$

$\longrightarrow \; \ldots$

$\longrightarrow \langle \text{nil}, \; \text{nil}, \; m[l \mapsto 0, \; f \mapsto 24] \rangle$

Plan for Today

- **Motivation & preliminaries**
- **Abstract syntax of SIMP**
- **An abstract machine for SIMP**
- **Structural operation semantics for SIMP**
  - Small-step semantics
  - Big-step semantics

Small-step semantics for SIMP

- Transitions = only computational steps
- Transition system with
  - configuration $\langle P, s \rangle$

    where P .. program (or command)

    s .. store (fct. from locations to integers)
  - transition relation between configurations

    ("reduction relations")

    ↳ defined via axioms & rules

Axioms & rules for expressions

$$\frac{}{\langle !\ell, s\rangle \to \langle n, s\rangle} \quad \text{if } s(\ell) = n \qquad (\text{var})$$

$$\frac{}{\langle n_1 \text{ op } n_2, s\rangle \to \langle n, s\rangle} \quad \text{if } n = (n_1 \text{ op } n_2) \qquad (\text{op})$$

$$\frac{\langle E_1, s\rangle \to \langle E_1', s'\rangle}{\langle E_1 \text{ op } E_2, s\rangle \to \langle E_1' \text{ op } E_2, s'\rangle} \ (\text{op}_L) \qquad \frac{\langle E_2, s\rangle \to \langle E_2', s'\rangle}{\langle n_1 \text{ op } E_2, s\rangle \to \langle n_1 \text{ op } E_2', s'\rangle} \ (\text{op}_R)$$

. . .

Axioms & Rules for commands

$$\frac{}{\langle \ell := n, s\rangle \longrightarrow \langle \mathrm{skip}, s[\ell \mapsto n]\rangle} \quad (:=)$$

$$\frac{\langle E, s\rangle \to \langle E', s'\rangle}{\langle \ell := E, s\rangle \longrightarrow \langle \ell := E', s'\rangle} \quad (:=_R)$$

$$\frac{}{\langle \mathrm{skip}; C, s\rangle \to \langle C, s\rangle} \quad (\mathrm{skip})$$

$$\frac{\langle C_1, s\rangle \longrightarrow \langle C_1', s'\rangle}{\langle C_1; C_2, s\rangle \longrightarrow \langle C_1'; C_2, s'\rangle} \quad (\mathrm{seq})$$

$$\frac{}{\langle \text{if True then } C_1 \text{ else } C_2, s\rangle \to \langle C_1, s\rangle} \quad (\text{if}_T)$$

$$\frac{}{\langle \text{if False then } C_1 \text{ else } C_2, s\rangle \to \langle C_2, s\rangle} \quad (\text{if}_F)$$

$$\frac{\langle B, s\rangle \to \langle B', s'\rangle}{\langle \text{if } B \text{ then } C_1 \text{ else } C_2, s\rangle \to \langle \text{if } B' \text{ then } C_1 \text{ else } C_2, s'\rangle} \quad (\text{if})$$

$$\frac{}{\langle \text{while } B \text{ do } C, s\rangle \to \langle \text{if } B \text{ then } (C; \text{while } B \text{ do } C) \text{ else skip}, s\rangle} \quad (\text{while})$$

Example: $P = z := !x \; ; \; (x := !y \; ; \; y := !z)$

$s = \{ z \mapsto 0, \; x \mapsto 1, \; y \mapsto 2 \}$

$$\langle P, s \rangle \xrightarrow{seq} \dots \longrightarrow \langle skip, \; s[z \mapsto 1, \; x \mapsto 2, \; y \mapsto 1] \rangle$$

each step: axiom or rule

↳ proof tree

Exerpt of proof tree:

$$\dfrac{\dfrac{\dfrac{}{\langle !x, s \rangle \to \langle 1, s \rangle}\ (var)}{\langle z := !x, s \rangle \to \langle z := 1, s \rangle}\ (:=_R)}{\langle P, s \rangle \to \langle z := 1 \; ; \; (x := !y \; ; \; y := !z), \; s \rangle}\ (seq)$$

Evaluation sequence

For $\langle P, s\rangle$, the evaluation sequence is a uniquely defined sequ. of transitions that starts with $\langle P, s\rangle$ and maximal length.

3 possible outcomes:

1. infinite sequence → program is **divergent** (i.e., non-terminating)
2. finite sequence that reaches $\langle n, s\rangle$, $\langle b, s\rangle$, or $\langle skip, s\rangle$
   → program **terminates**, final config. = result of program
3. finite sequence that stops in some other config.
   → program **blocked**

Examples

① $\langle$ while True do skip , s $\rangle$

$\to$ $\langle$ if True then (skip ; while True do skip) else skip , s $\rangle$

$\to$ $\langle$ skip ; while True do skip , s $\rangle$ $\to$ ...

$\Rightarrow$ divergent

② $\langle$ if 4 = 0 then skip else skip , s $\rangle$

$\to$ $\langle$ if False then skip else skip , s $\rangle$

$\to$ $\langle$ skip , s $\rangle$

$\Rightarrow$ terminating

③ $\langle$ if !x > 0 then C else skip , s $\rangle$

where $x \notin dom(s)$

$\Rightarrow$ blocked

Big-step semantics

- small-step semantics: transition relation
  = indiv. steps of computation

- now: transition = full evaluation

- config. remains the same

- evaluation relation:

$$\langle P, s \rangle \Downarrow \underbrace{\langle P', s' \rangle}_{\text{last config. of } \langle P, s\rangle\text{'s evaluation seq. (if } P \text{ terminates)}}$$

"evaluates to"

(Some) axioms & rules

$$\frac{}{\langle !\ell, s\rangle \Downarrow \langle n, s\rangle} \text{ if } s(\ell) = n \qquad \text{(var)}$$

$$\frac{\langle E, s\rangle \Downarrow \langle n, s'\rangle}{\langle \ell := E, s\rangle \Downarrow \langle \text{skip}, s'[\ell \mapsto n]\rangle} \qquad (:=)$$

Much simpler:
1 rule instead of 2

$$\frac{\langle C_1, s\rangle \Downarrow \langle \text{skip}, s'\rangle \qquad \langle C_2, s'\rangle \Downarrow \langle \text{skip}, s''\rangle}{\langle C_1 ; C_2, s\rangle \Downarrow \langle \text{skip}, s''\rangle} \qquad \text{(seq)}$$

Example: $P = \underbrace{(z := !x \;;\; x := !y)}_{P'} \;;\; \underbrace{y := !z}_{P''}$

$$s = \{z \mapsto 0,\ x \mapsto 1,\ y \mapsto 2\}$$

$\langle P, s\rangle \Downarrow \langle \text{skip}, s'\rangle$ where $s' = \{z \mapsto 1,\ x \mapsto 2,\ y \mapsto 1\}$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\ldots}\,(\text{var})}{\ldots}\,(:=) \qquad \dfrac{\dfrac{}{\ldots}\,(\text{var})}{\ldots}\,(:=)}{\langle P', s\rangle \Downarrow \langle \text{skip}, s''\rangle}\,(\text{seq}) \qquad \dfrac{\dfrac{\dfrac{}{\ldots}\,(\text{var})}{\ldots}\,(:=)}{\langle P'', s''\rangle \Downarrow \langle \text{skip}, s'\rangle}}{\langle P, s\rangle \Downarrow \langle \text{skip}, s'\rangle}\,(\text{seq})$$