Program Analysis

Operational Semantics (Part 1)

Prof. Dr. Michael Pradel

Software Lab, University of Stuttgart

Winter 2020/2021

Big Picture

Last lecture:

- Syntax of languages
- Representations of programs

This lecture:

- Assign meaning (= semantics) to programs
- Focus: Operational semantics of imperative languages
- Formal foundation for specifying languages and for describing analyses

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

Note: Mistake in video is fixed

Why do we need operational semantics?

Example (C code):

```
int i = 5;
f(i++, --i);
```

← What are the arguments passed to f?

Option 1: 5, 5 (left-to-right)

Option 2: 4, 4 (right-to-left)

Both options are possible in C!

→ Unspecified semantics

→ Compiler decides

Want: (Almost) all behavior should be clearly specified

Specifying the semantics of programs

- Static semantics, e.g., type
- Dynamic semantics
  - Denotational
  - Axiomatic
  - Operational ← Focus here

↳ Useful for:
- Lang. design
- Lang. implementation
- Programming
- Program analysis

Preliminaries

a) Transition systems

* set Config of configurations or states
* binary relation $\rightarrow \subseteq \text{Config} \times \text{Config}$
  ("transition relation")

$c \rightarrow c'$ .. transition (or change of state)
↳ step of computation

deterministic : $c \rightarrow c_1 \wedge c \rightarrow c_2 \Rightarrow c_1 = c_2$

$\rightarrow^*$ .. reflexive, transitive closure of $\rightarrow$

$\forall c.\ c \rightarrow^* c$ $\qquad$ $\forall c, c', c''.\ c \rightarrow^* c' \wedge c' \rightarrow^* c'' \Rightarrow c \rightarrow^* c''$

b) <u>Rule induction</u>

↳ define a set ("inductive set") with :

* a finite set of basic elements ("axioms") : $\dfrac{}{t}$ (empty)
* a finite set of rules that specify how to generate more elements :

$$\frac{h_1 \quad \dots \quad h_n}{c}$$

hypotheses

conclusion

↳ <u>Ex. 1</u> set of natural nbs.

axiom : $\dfrac{}{0}$

rule : $\dfrac{n}{n+1}$

<u>Ex. 2</u> Evaluation of expressions, e.g., $+(3,4)$

↳ Set = pairs of ASTs & values

Notation: $E \Downarrow n$ .. expression E evaluates to nb. n

Axioms: $\underbrace{1 \Downarrow 1 \quad , \quad 2 \Downarrow 2, \text{ etc.}}_{\text{axiom scheme: } n \Downarrow n}$

Rules: $\underbrace{\dfrac{E_1 \Downarrow n_1 \qquad E_2 \Downarrow n_2}{+(E_1, E_2) \Downarrow n} \text{ if } n = n_1 \dot{+} n_2 \quad \text{etc.}}_{\text{rule scheme: } \dfrac{E_1 \Downarrow n_1 \qquad E_2 \Downarrow n_2}{Op\,(E_1, E_2) \Downarrow n} \text{ if } n = n_1 \; Op \; n_2}$

c) <u>Proof tree</u>

↳ show that an element is in an inductive set

<u>Ex. 1</u>

$$\frac{\dfrac{\dfrac{}{0}}{1}}{2}$$

<u>Ex. 2</u> show that

$$-(+(3,4),1) \Downarrow 6$$

$$\dfrac{\dfrac{\dfrac{}{3 \Downarrow 3} \qquad \dfrac{}{4 \Downarrow 4}}{+(3,4) \Downarrow 7} \qquad \dfrac{}{1 \Downarrow 1}}{-(+(3,4),1) \Downarrow 6}$$