

Elements of Programming Languages

Tutorial 7: Small-step semantics and type soundness

Week 9 (November 11–15, 2024)

Exercises marked \star are more advanced. Please try all unstarred exercises before the tutorial meeting.

1. Imperative programming

Write evaluation derivations for the following imperative programs, starting with the environment $\sigma = [x = 3, y = 4]$.

- (a) $y := x + x$
- (b) **if** $x == y$ **then** $x := x + 1$ **else** $y := y + 2$
- (c) \star **while** $x < y$ **do** $x := x + 1$

2. Comparing large-step and small-step derivations

Write both large-step and small-step derivations for the following expressions. For the small-step derivations, construct the derivations of each $e \mapsto e'$ step explicitly.

- (a) $(\lambda x. x + 1) 42$
- (b) $(\lambda x. \text{if } x == 1 \text{ then } 2 \text{ else } x + 1) 42$

3. Small-step derivations that go wrong

For each of the following expressions, show the small-step evaluation leading to the point where evaluation becomes stuck due to a dynamic type error. (There is no need to show the derivations of each step.)

- (a) $((\lambda x. \lambda y. \text{let } z = x + y \text{ in } z + 1) 42) \text{ true}$
- (b) $(\lambda x. \text{if } x \text{ then } x + 1 \text{ else } x + 2) \text{ true}$

4. Small-step rules for L_{Data}

Recall that we defined the semantics for L_{Data} using big-step rules, as follows:

$$\boxed{e \Downarrow v}$$

$$\frac{e_1 \Downarrow v_1 \quad e_2 \Downarrow v_2}{(e_1, e_2) \Downarrow (v_1, v_2)} \quad \frac{e \Downarrow (v_1, v_2)}{\text{fst } e \Downarrow v_1} \quad \frac{e \Downarrow (v_1, v_2)}{\text{snd } e \Downarrow v_2}$$

$$\frac{e \Downarrow v}{\text{left}(e) \Downarrow \text{left}(v)} \quad \frac{e \Downarrow \text{left}(v_1) \quad e_1[v_1/x] \Downarrow v}{\text{case } e \text{ of } \{\text{left}(x) \Rightarrow e_1; \text{right}(y) \Rightarrow e_2\} \Downarrow v}$$

$$\frac{e \Downarrow v}{\text{right}(e) \Downarrow \text{right}(v)} \quad \frac{e \Downarrow \text{right}(v_2) \quad e_2[v_2/y] \Downarrow v}{\text{case } e \text{ of } \{\text{left}(x) \Rightarrow e_1; \text{right}(y) \Rightarrow e_2\} \Downarrow v}$$

- (a) For each construct, write out equivalent small-step rules. Are there any design choices in translating the big-step rules to small-step rules?
- (b) (★) Construct small-step derivations reducing the following expressions to values:
 - i. $(\lambda p.(\text{snd } p, \text{fst } p + 2)) (17, 42)$
 - ii. $(\lambda x.\text{case } x \text{ of } \{\text{left}(y).y + 1 ; \text{right}(z).z\}) (\text{left}(42))$

5. (★) **Type soundness for nondeterminism**

This question builds on the *nondeterministic choice* construct mentioned in an earlier tutorial, with the following typing rules:

$$\boxed{\Gamma \vdash e : \tau}$$

$$\frac{\Gamma \vdash e_1 : \tau \quad \Gamma \vdash e_2 : \tau}{\Gamma \vdash e_1 \square e_2 : \tau}$$

and small-step evaluation rules:

$$\boxed{e \mapsto e'}$$

$$\overline{e_1 \square e_2 \mapsto e_1} \quad \overline{e_1 \square e_2 \mapsto e_2}$$

- (a) State the *preservation* property. Outline how we could prove the cases of preservation for nondeterministic expressions.
- (b) State the *progress* property. Outline how we could prove the cases of progress for nondeterministic expressions.