Elements of Programming Languages
Tutorial 7: Small-step semantics
Solution notes

1. **Imperative programming**

   (a) $y := x + x$

$$\frac{\overline{3 + 3 \Downarrow 6}}{\sigma, y := x + x \Downarrow \sigma[y := 6]}$$

   (b) $\texttt{if}\ x == y\ \texttt{then}\ x := x + 1\ \texttt{else}\ y := y + 2$

$$\frac{\dfrac{3 \Downarrow 3 \quad 4 \Downarrow 4}{3 == 4 \Downarrow \texttt{true}} \quad \dfrac{4 + 2 \Downarrow 6}{\sigma, y := y + 2 \Downarrow \sigma[y := 6]}}{\sigma, \texttt{if}\ x == y\ \texttt{then}\ x := x + 1\ \texttt{else}\ y := y + 2 \Downarrow \sigma[y := 6]}$$

   (c) ($\star$) $\texttt{while}\ x < y\ \texttt{do}\ x := x + 1$

$$\frac{3 < 4 \Downarrow \texttt{true} \quad \dfrac{3 + 1 \Downarrow 4}{\sigma, x := x + 1 \Downarrow \sigma[x := 4]} \quad \dfrac{4 < 4 \Downarrow \texttt{false}}{\sigma[x := 4], \texttt{while}\ x < y\ \texttt{do}\ x := x + 1 \Downarrow \sigma[x := 4]}}{\sigma, \texttt{while}\ x < y\ \texttt{do}\ x := x + 1 \Downarrow \sigma[x := 4]}$$

2. **Comparing large-step and small-step derivations**

   (a) For large-step:

$$\frac{\lambda x.x + 1 \Downarrow \lambda x.x + 1 \quad 42 \Downarrow 42 \quad \dfrac{42 \Downarrow 42 \quad 1 \Downarrow 1}{42 + 1 \Downarrow 43}}{(\lambda x.x + 1)\ 42 \Downarrow 43}$$

   For small-step, the step derivations are:

$$\frac{}{(\lambda x.x + 1)\ 42 \mapsto 42 + 1}$$

$$\frac{}{42 + 1 \mapsto 43}$$

   (b) For large-step:

$$\frac{f \Downarrow f \quad 42 \Downarrow 42 \quad \dfrac{\dfrac{42 \Downarrow 42 \quad 1 \Downarrow 1}{42 == 1 \Downarrow \texttt{false}} \quad \dfrac{42 \Downarrow 42 \quad 1 \Downarrow 1}{42 + 1 \Downarrow 43}}{\texttt{if}\ 42 == 1\ \texttt{then}\ 2\ \texttt{else}\ 42 + 1 \Downarrow 43}}{(\lambda x.\texttt{if}\ x == 1\ \texttt{then}\ 2\ \texttt{else}\ x + 1)\ 42 \Downarrow 43}$$

   For small-step, the step derivations are:

$$\frac{}{(\lambda x.\texttt{if}\ x == 1\ \texttt{then}\ 2\ \texttt{else}\ x + 1)\ 42 \mapsto \texttt{if}\ 42 == 1\ \texttt{then}\ 2\ \texttt{else}\ 42 + 1}$$

$$\frac{\dfrac{}{42 == 1 \mapsto \texttt{false}}}{\texttt{if}\ 42 == 1\ \texttt{then}\ 2\ \texttt{else}\ 42 + 1 \mapsto \texttt{if}\ \texttt{false}\ \texttt{then}\ 2\ \texttt{else}\ 42 + 1}$$

$$\frac{}{\texttt{if}\ \texttt{false}\ \texttt{then}\ 2\ \texttt{else}\ 42 + 1 \mapsto 42 + 1}$$

$$\frac{}{42 + 1 \mapsto 43}$$

3. **Small-step derivations that go wrong**

   (a) The small-step derivation is:

$$\begin{array}{rcl}((\lambda x.\lambda y.\texttt{let } z = x + y \texttt{ in } z + 1)\ 42)\ \texttt{true} & \mapsto & (\lambda y.\texttt{let } z = 42 + y \texttt{ in } z + 1)\ \texttt{true} \\ & \mapsto & \texttt{let } z = 42 + \texttt{true} \texttt{ in } z + 1\end{array}$$

   which is stuck because the next subexpression to evaluate, $42 + \texttt{true}$, cannot take a step.

   (b) The small-step derivation is:

$$\begin{array}{rcl}(\lambda x.\texttt{if } x \texttt{ then } x + 1 \texttt{ else } x + 2)\ \texttt{true} & \mapsto & \texttt{if true then true} + 1 \texttt{ else true} + 2 \\ & \mapsto & \texttt{true} + 1\end{array}$$

4. **Small-step rules for** $\mathsf{L}_{\mathsf{Data}}$

   (a) Here are some possible rules:

   $\boxed{e \mapsto e'}$

$$\frac{e_1 \mapsto e_1'}{(e_1, e_2) \mapsto (e_1', e_2)} \qquad \frac{e_2 \mapsto e_2'}{(v_1, e_2) \mapsto (v_1, e_2')}$$

$$\frac{}{\texttt{fst}\ (v_1, v_2) \mapsto v_1} \qquad \frac{}{\texttt{snd}\ (v_1, v_2) \mapsto v_2}$$

$$\frac{e \mapsto e'}{\texttt{fst}\ e \mapsto \texttt{fst}\ e'} \qquad \frac{e \mapsto e'}{\texttt{snd}\ e \mapsto \texttt{snd}\ e'}$$

$$\frac{e \mapsto e'}{\texttt{left}(e) \mapsto \texttt{left}(e')} \qquad \frac{e \mapsto e'}{\texttt{right}(e) \mapsto \texttt{right}(e')}$$

$$\frac{e \mapsto e'}{\texttt{case}\ e\ \texttt{of}\ \{\texttt{left}(x) \Rightarrow e_1\ ;\ \texttt{right}(y) \Rightarrow e_2\} \mapsto \texttt{case}\ e'\ \texttt{of}\ \{\texttt{left}(x) \Rightarrow e_1\ ;\ \texttt{right}(y) \Rightarrow e_2\}}$$

$$\frac{}{\texttt{case left}(v)\ \texttt{of}\ \{\texttt{left}(x) \Rightarrow e_1\ ;\ \texttt{right}(y) \Rightarrow e_2\} \mapsto e_1[v/x]}$$

$$\frac{}{\texttt{case right}(v)\ \texttt{of}\ \{\texttt{left}(x) \Rightarrow e_1\ ;\ \texttt{right}(y) \Rightarrow e_2\} \mapsto e_2[v/y]}$$

   One example of a design choice is whether to evaluate the subexpressions of a pair left-to-right, or right-to-left (or not specify the order, or not evaluate them eagerly at all...)

   (b) i. The step derivations are:

$$\frac{}{(\lambda p.(\texttt{snd}\ p, \texttt{fst}\ p + 2))\ (17, 42) \mapsto (\texttt{snd}\ (17, 42), \texttt{fst}\ (17, 42) + 2)}$$

$$\frac{\dfrac{}{\texttt{snd}\ (17, 42) \mapsto 42}}{(\texttt{snd}\ (17, 42), \texttt{fst}\ (17, 42) + 2) \mapsto (42, \texttt{fst}\ (17, 42) + 2)}$$

$$\frac{\dfrac{\dfrac{}{\texttt{fst}\ (17, 42) \mapsto 17}}{\texttt{fst}\ (17, 42) + 2 \mapsto 17 + 2}}{(42, \texttt{fst}\ (17, 42) + 2) \mapsto (42, 17 + 2)}$$

$$\frac{}{(42, 17 + 2) \mapsto (42, 19)}$$

   ii. The step derivations are:

$$\frac{}{(\lambda x.\texttt{case}\ x\ \texttt{of}\ \{\texttt{left}(y).y + 1\ ;\ \texttt{right}(z).\ z\})\ (\texttt{left}(42)) \mapsto \texttt{case left}(42)\ \texttt{of}\ \{\texttt{left}(y).y + 1\ ;\ \texttt{right}(z).\ z\}}$$

$$\frac{}{\texttt{case left}(42)\ \texttt{of}\ \{\texttt{left}(y).y + 1\ ;\ \texttt{right}(z).\ z\} \mapsto 42 + 1}$$

$$\frac{}{42 + 1 \mapsto 43}$$

5. ($\star$) **Type soundness for nondeterminism**

   (a) Preservation means that if $\vdash e : \tau$ holds and $e \mapsto e'$ then $\vdash e' : \tau$ holds.
   To prove preservation, suppose we have a well-formed nondeterminstic choice expression:

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

   such that $e_1 \Box e_2 \mapsto e'$.
   There are two cases. If $e_1 \Box e_2 \mapsto e_1$ then we know $\vdash e_1 : \tau$ by assumption. Similarly, if $e_1 \Box e_2 \mapsto e_2$ then we know $\vdash e_2 : \tau$.

   (b) Progress means that if $\vdash e : \tau$ holds then either e is a value, or e can take a step ($e \mapsto e'$).
   To prove progress for a nondeterministic expression $\vdash e_1 \Box e_2 : \tau$, this is immediate since such an expression can definitely step to e_1 (or e_2). Note that progress does not require that the step taken is unique; indeed, type soundness can hold for a nondeterminstic language as long as both alternatives in a choice expression have the same type.