Exercises on Program Semantics

Theory of Programming Languages
Master's Degree in Formal Methods in Computer Science
Year 2021–2022

1. Given the set **AExp** of arithmetic expressions:

   (a) Define a function $cst : \mathbf{AExp} \to \mathcal{P}(\mathbb{Z})$ such that $cst(e)$ is the set of all integer constants appearing in e.

   Solution

$$
\begin{array}{rcl}
cst(n) & = & \{n\} \\
cst(x) & = & \emptyset \\
cst(e_1 + e_2) & = & cst(e_1) \cup cst(e_2) \\
cst(e_1 - e_2) & = & cst(e_1) \cup cst(e_2) \\
cst(e_1 * e_2) & = & cst(e_1) \cup cst(e_2)
\end{array}
$$

   (b) Define a function $highest : \mathbf{AExp} \to \mathbb{Z} \cup \{-\infty\}$ such that $highest(e)$ is the highest integer constant appearing in e. If there are no constants in e, then $highest(e) = -\infty$. Your definition must be compositional (i.e. you cannot use cst).

   Solution

$$
\begin{array}{rcl}
highest(n) & = & n \\
highest(x) & = & -\infty \\
highest(e_1 + e_2) & = & \max\{highest(e_1), highest(e_2)\} \\
highest(e_1 - e_2) & = & \max\{highest(e_1), highest(e_2)\} \\
highest(e_1 * e_2) & = & \max\{highest(e_1), highest(e_2)\}
\end{array}
$$

   (c) Prove that, for every $e \in \mathbf{AExp}$, $highest(e) \in cst(e) \cup \{-\infty\}$.

   Solution

   By structural induction on e. We distinguish cases according to the structure of e:

   - **Case** $e = n$.
     We get $cst(e) = \{n\}$, $highest(e) = n$, and $n \in \{n\} \cup \{-\infty\}$ holds.
   - **Case** $e = x$.
     We get $cst(e) = \emptyset$, $highest(e) = -\infty$, and $-\infty \in \emptyset \cup \{-\infty\}$ holds.
   - **Case** $e = e_1 + e_2$.
     We can assume the following induction hypotheses:

$$
\begin{array}{ll}
(H1) & highest(e_1) \in cst(e_1) \cup \{-\infty\} \\
(H2) & highest(e_2) \in cst(e_2) \cup \{-\infty\}
\end{array}
$$

We know that $highest(e_1+e_2) = \max\{highest(e_1), highest(e_2)\}$, so $highest(e_1+e_2)$ is either $highest(e_1)$ or $highest(e_2)$.

- If $highest(e_1+e_2) = highest(e_1)$, then by (H1) we get $highest(e_1+e_2) \in cst(e_1) \cup \{-\infty\}$. Since $cst(e_1) \subseteq cst(e_1+e_2)$ we finally get $highest(e_1+e_2) \in cst(e_1+e_2) \cup \{-\infty\}$.
- If $highest(e_1+e_2) = highest(e_2)$, then by (H2) we get $highest(e_1+e_2) \in cst(e_2) \cup \{-\infty\}$. Since $cst(e_2) \subseteq cst(e_1+e_2)$ we finally get $highest(e_1+e_2) \in cst(e_1+e_2) \cup \{-\infty\}$.

- **Case** $e = e_1 - e_2$. Since $highest(e_1-e_2) = highest(e_1+e_2)$, and $cst(e_1-e_2) = cst(e_1+e_2)$ and we have already proved the case $e = e_1+e_2$, the property follows from the latter.
- **Case** $e = e_1 * e_2$. The same as above.

2. Assume an expression e and two states σ_1 and σ_2. Prove that, if $\sigma_1(x) = \sigma_2(x)$ for all $x \in FV(e)$, then $\mathcal{A}[\![e]\!]\,\sigma_1 = \mathcal{A}[\![e]\!]\,\sigma_2$.

Solution

By structural induction on e. We distinguish cases:

- **Case** $e = n$. For any σ_1 and σ_2 we get:

$$\mathcal{A}[\![n]\!]\,\sigma_1 = n = \mathcal{A}[\![n]\!]\,\sigma_2$$

- **Case** $e = x$. In this case $FV(e) = \{x\}$, so by assumption we get that $\sigma_1(x) = \sigma_2(x)$. Therefore:

$$\mathcal{A}[\![x]\!]\,\sigma_1 = \sigma_1(x) = \sigma_2(x) = \mathcal{A}[\![x]\!]\,\sigma_2$$

- **Case** $e = e_1 + e_2$. The induction hypotheses are:

$$\begin{array}{ll} (H1) & \text{If } \sigma_1(x) = \sigma_2(x) \text{ for all } x \in FV(e_1), \text{ then } \mathcal{A}[\![e_1]\!]\,\sigma_1 = \mathcal{A}[\![e_1]\!]\,\sigma_2 \\ (H2) & \text{If } \sigma_1(x) = \sigma_2(x) \text{ for all } x \in FV(e_2), \text{ then } \mathcal{A}[\![e_2]\!]\,\sigma_1 = \mathcal{A}[\![e_2]\!]\,\sigma_2 \end{array}$$

We know that $\sigma_1(x) = \sigma_2(x)$ for all $x \in FV(e)$. Since $FV(e_1) \subseteq FV(e)$ and $FV(e_2) \subseteq FV(e)$ it follows that $\sigma_1(x) = \sigma_2(x)$ also holds for all $x \in FV(e_1)$ and for all $x \in FV(e_2)$. Therefore we get $\mathcal{A}[\![e_1]\!]\,\sigma_1 = \mathcal{A}[\![e_1]\!]\,\sigma_2$ and $\mathcal{A}[\![e_2]\!]\,\sigma_1 = \mathcal{A}[\![e_2]\!]\,\sigma_2$. By unfolding the semantic definitions of $e_1 + e_2$ we get:

$$\mathcal{A}[\![e_1+e_2]\!]\,\sigma_1 = \mathcal{A}[\![e_1]\!]\,\sigma_1 + \mathcal{A}[\![e_2]\!]\,\sigma_1 = \mathcal{A}[\![e_1]\!]\,\sigma_2 + \mathcal{A}[\![e_2]\!]\,\sigma_2 = \mathcal{A}[\![e_1+e_2]\!]\,\sigma_2$$

- **Cases** $e = e_1 - e_2$ and $e = e_1 * e_2$. Similarly as above.

3. Assume we extend our syntax for arithmetic expressions with a new construct `or`:

$$e ::= n \mid x \mid e_1 + e_2 \mid e_1 - e_2 \mid e_1 * e_2 \mid e_1 \texttt{ or } e_2$$

An expression of the form $e_1 \texttt{ or } e_2$ may be evaluated to either e_1 or e_2. Whether to execute e_1 or e_2 is chosen nondeterministically. Add the necessary rules to the big-step and small-step semantics of arithmetic expressions to deal with this new construct. How would be our denotational semantics affected?

Solution

Big-step rules:

$$\frac{\langle e_1, \sigma\rangle \Downarrow v}{\langle e_1 \texttt{ or } e_2, \sigma\rangle \Downarrow v} \qquad \frac{\langle e_2, \sigma\rangle \Downarrow v}{\langle e_1 \texttt{ or } e_2, \sigma\rangle \Downarrow v}$$

Small-step rules:

$$\frac{}{\sigma \vdash e_1 \texttt{ or } e_2 \longrightarrow e_1} \qquad \frac{}{\sigma \vdash e_1 \texttt{ or } e_2 \longrightarrow e_2}$$

These semantic definitions are no longer deterministic. An expression could be evaluated to different values. For example, we get $\langle 1 \texttt{ or } 4, \sigma\rangle \Downarrow 1$ and $\langle 1 \texttt{ or } 4, \sigma\rangle \Downarrow 4$ for any σ. This means that the value of $\mathcal{A}[\![e_1 \texttt{ or } e_2]\!]\,\sigma$ is no longer a single value, but a **set** of values.

$$\mathcal{A}[\![\_]\!] : \textbf{AExp} \to \textbf{State} \to \mathcal{P}(\mathbb{Z})$$

$$\begin{array}{rcl}
\mathcal{A}[\![n]\!]\,\sigma & = & \{n\} \\
\mathcal{A}[\![x]\!]\,\sigma & = & \{\sigma(x)\} \\
\mathcal{A}[\![e_1 + e_2]\!]\,\sigma & = & \{x + y \mid x \in \mathcal{A}[\![e_1]\!]\,\sigma, y \in \mathcal{A}[\![e_2]\!]\,\sigma\} \\
\mathcal{A}[\![e_1 - e_2]\!]\,\sigma & = & \{x - y \mid x \in \mathcal{A}[\![e_1]\!]\,\sigma, y \in \mathcal{A}[\![e_2]\!]\,\sigma\} \\
\mathcal{A}[\![e_1 * e_2]\!]\,\sigma & = & \{x * y \mid x \in \mathcal{A}[\![e_1]\!]\,\sigma, y \in \mathcal{A}[\![e_2]\!]\,\sigma\}
\end{array}$$

4. Given the following *While* program S:

$$x := 0;\ \texttt{while } n > 0 \texttt{ do } (x := x + n;\ n := n - 1)$$

Build the big-step derivation tree that results from executing S under the state $\sigma = [n \mapsto 1]$.

Solution

$$\dfrac{\langle x := 0, \sigma\rangle \Downarrow \sigma[x \mapsto 0] \qquad \dfrac{(*) \quad \langle \texttt{while}\ n > 0\ \texttt{do}\ (x := x + n;\ n := n - 1), [x \mapsto 1, n \mapsto 0]\rangle \Downarrow [x \mapsto 1, n \mapsto 0]}{\langle \texttt{while}\ n > 0\ \texttt{do}\ (x := x + n;\ n := n - 1), \sigma[x \mapsto 0]\rangle \Downarrow [x \mapsto 1, n \mapsto 0]}}{\langle x := 0;\ \texttt{while}\ n > 0\ \texttt{do}\ (x := x + n;\ n := n - 1), \sigma\rangle \Downarrow [x \mapsto 1, n \mapsto 0]}$$

Where (*) is the following derivation tree:

$$\dfrac{\dfrac{}{\langle x := x + n, \sigma[x \mapsto 0]\rangle \Downarrow \sigma[x \mapsto 1]} \quad \dfrac{}{\langle n := n - 1, \sigma[x \mapsto 1]\rangle \Downarrow [x \mapsto 1, n \mapsto 0]}}{\langle x := x + n; n := n - 1, \sigma[x \mapsto 0]\rangle \Downarrow [x \mapsto 1, n \mapsto 0]}$$

5. Prove that $S_1; (S_2; S_3)$ is semantically equivalent to $(S_1; S_2); S_3$.

Solution

We have to prove, for any σ and σ':

$$\langle S_1; (S_2; S_3), \sigma\rangle \Downarrow \sigma' \iff \langle (S_1; S_2); S_3, \sigma\rangle \Downarrow \sigma'$$

We first prove the $(\Longrightarrow)$ direction, then we prove the $(\Longleftarrow)$ direction.

- $(\Longrightarrow)$
  Assume that $\langle S_1; (S_2; S_3), \sigma\rangle \Downarrow \sigma'$. We must have applied the [Seq] rule in order to obtain this derivation:

  $$\dfrac{\langle S_1, \sigma\rangle \Downarrow \sigma_1 \quad \langle S_2; S_3, \sigma_1\rangle \Downarrow \sigma'}{\langle S_1; (S_2; S_3), \sigma\rangle \Downarrow \sigma'}$$

  So we know that there exists some σ_1 such that:

  $$\langle S_1, \sigma\rangle \Downarrow \sigma_1 \tag{1}$$

  Moreover, the judgement $\langle S_2; S_3, \sigma_1\rangle \Downarrow \sigma'$ must have been obtained by using the [Seq] rule:

  $$\dfrac{\langle S_2, \sigma_1\rangle \Downarrow \sigma_2 \quad \langle S_3, \sigma_2\rangle \Downarrow \sigma'}{\langle S_2; S_3, \sigma_1\rangle \Downarrow \sigma'}$$

  Therefore, there exists some σ_2 such that:

  $$\langle S_2, \sigma_1\rangle \Downarrow \sigma_2 \tag{2}$$
  $$\langle S_3, \sigma_2\rangle \Downarrow \sigma' \tag{3}$$

  We can now apply the [Seq] rule by using (1) and (2) as assumptions:

  $$\dfrac{\langle S_1, \sigma\rangle \Downarrow \sigma_1 \quad \langle S_2, \sigma_1\rangle \Downarrow \sigma_2}{\langle S_1; S_2, \sigma\rangle \Downarrow \sigma_2}$$

We have derived thus the judgement $\langle S_1; S_2, \sigma\rangle \Downarrow \sigma_2$, which we can use with (3) to get:

$$\frac{\langle S_1; S_2, \sigma\rangle \Downarrow \sigma_2 \quad \langle S_3, \sigma_2\rangle \Downarrow \sigma'}{\langle (S_1; S_2); S_3, \sigma\rangle \Downarrow \sigma'}$$

- $(\Longleftarrow)$
  Assume that $\langle (S_1; S_2); S_3, \sigma\rangle \Downarrow \sigma'$. We must have applied the [Seq] rule:

$$\frac{\langle S_1; S_2, \sigma\rangle \Downarrow \sigma_1 \quad \langle S_3, \sigma_1\rangle \Downarrow \sigma'}{\langle (S_1; S_2); S_3, \sigma\rangle \Downarrow \sigma'}$$

  So there exists some σ_1 such that

$$\langle S_3, \sigma_1\rangle \Downarrow \sigma' \tag{4}$$

  If we focus on the judgement $\langle S_1; S_2, \sigma\rangle \Downarrow \sigma_1$, it must have been obtained by using the [Seq] rule:

$$\frac{\langle S_1, \sigma\rangle \Downarrow \sigma_2 \quad \langle S_2, \sigma_2\rangle \Downarrow \sigma_1}{\langle S_1; S_2, \sigma\rangle \Downarrow \sigma_1}$$

  Hence there exists some σ_2 such that:

$$\langle S_1, \sigma\rangle \Downarrow \sigma_2 \tag{5}$$
$$\langle S_2, \sigma_2\rangle \Downarrow \sigma_1 \tag{6}$$

  Now we use (4), (5), (6) and apply the [Seq] rule twice in order to obtain the required judgement:

$$\frac{\langle S_1, \sigma\rangle \Downarrow \sigma_2 \qquad \dfrac{\langle S_2, \sigma_2\rangle \Downarrow \sigma_1 \quad \langle S_3, \sigma_1\rangle \Downarrow \sigma'}{\langle S_2; S_3, \sigma_2\rangle \Downarrow \sigma'}}{\langle S_1; (S_2; S_3), \sigma\rangle \Downarrow \sigma'}$$

6. Prove that $\texttt{while}\ b\ \texttt{do}\ S$ is semantically equivalent to $(\texttt{if}\ b\ \texttt{then}\ (S; \texttt{while}\ b\ \texttt{do}\ S)\ \texttt{else}\ \texttt{skip})$.

Solution

We have to prove for any $\sigma, \sigma' \in$ **State**:

$$\langle \texttt{while}\ b\ \texttt{do}\ S, \sigma\rangle \Downarrow \sigma' \Longleftrightarrow \langle \texttt{if}\ b\ \texttt{then}\ (S; \texttt{while}\ b\ \texttt{do}\ S)\ \texttt{else}\ \texttt{skip}, \sigma\rangle \Downarrow \sigma'$$

- $(\Longrightarrow)$

  Assume that $\langle \texttt{while}\ b\ \texttt{do}\ S, \sigma\rangle \Downarrow \sigma'$. If $\mathcal{B}[\![b]\!]\ \sigma = \mathit{false}$ we get that $\sigma = \sigma'$. In this case we can build the following derivation:

  $$\frac{\langle \texttt{skip}, \sigma\rangle \Downarrow \sigma \quad \mathcal{B}[\![b]\!]\ \sigma = \mathit{true}}{\langle \texttt{if}\ b\ \texttt{then}\ (S; \texttt{while}\ b\ \texttt{do}\ S)\ \texttt{else}\ \texttt{skip}, \sigma\rangle \Downarrow \sigma}$$

  and we are done. In the case in which $\mathcal{B}[\![b]\!]\ \sigma = \mathit{true}$ we know that we have the following derivation for $\langle \texttt{while}\ b\ \texttt{do}\ S, \sigma\rangle \Downarrow \sigma'$:

  $$\frac{\langle S, \sigma\rangle \Downarrow \sigma'' \quad \langle \texttt{while}\ b\ \texttt{do}\ S, \sigma''\rangle \Downarrow \sigma'}{\langle \texttt{while}\ b\ \texttt{do}\ S, \sigma\rangle \Downarrow \sigma'}$$

  That is, there exists some σ'' such that $\langle S, \sigma\rangle \Downarrow \sigma''$ and $\langle \texttt{while}\ b\ \texttt{do}\ S, \sigma''\rangle \Downarrow \sigma'$. We can apply the [Seq] rule as follows:

  $$\frac{\langle S, \sigma\rangle \Downarrow \sigma'' \quad \langle \texttt{while}\ b\ \texttt{do}\ S, \sigma''\rangle \Downarrow \sigma'}{\langle S; \texttt{while}\ b\ \texttt{do}\ S, \sigma\rangle \Downarrow \sigma'}$$

  and, finally, the [IfT] rule:

  $$\frac{\langle S; \texttt{while}\ b\ \texttt{do}\ S, \sigma\rangle \Downarrow \sigma' \quad \mathcal{B}[\![b]\!]\ \sigma = \mathit{true}}{\langle \texttt{if}\ b\ \texttt{then}\ (S; \texttt{while}\ b\ \texttt{do}\ S)\ \texttt{else}\ \texttt{skip}, \sigma\rangle \Downarrow \sigma'}$$

- $(\Longleftarrow)$

  Assume that $\langle \texttt{if}\ b\ \texttt{then}\ (S; \texttt{while}\ b\ \texttt{do}\ S)\ \texttt{else}\ \texttt{skip}, \sigma\rangle \Downarrow \sigma'$. Let us distinguish cases on whether $\mathcal{B}[\![b]\!]\ \sigma = \mathit{false}$ or $\mathcal{B}[\![b]\!]\ \sigma = \mathit{true}$. In the first case, we must have obtained the following derivation:

  $$\frac{\langle \texttt{skip}, \sigma\rangle \Downarrow \sigma' \quad \mathcal{B}[\![b]\!]\ \sigma = \mathit{false}}{\langle \texttt{if}\ b\ \texttt{then}\ (S; \texttt{while}\ b\ \texttt{do}\ S)\ \texttt{else}\ \texttt{skip}, \sigma\rangle \Downarrow \sigma'}$$

  But because of [Skip] rule, we get that $\sigma = \sigma'$. Therefore, we can use the [WhileF] rule in order to derive the following:

  $$\frac{\mathcal{B}[\![b]\!]\ \sigma = \mathit{false}}{\langle \texttt{while}\ b\ \texttt{do}\ S, \sigma\rangle \Downarrow \sigma}$$

  and hence $\langle \texttt{while}\ b\ \texttt{do}\ S, \sigma\rangle \Downarrow \sigma'$ since $\sigma = \sigma'$. Now assume that $\mathcal{B}[\![b]\!]\ \sigma = \mathit{true}$. The judgement $\langle \texttt{if}\ b\ \texttt{then}\ (S; \texttt{while}\ b\ \texttt{do}\ S)\ \texttt{else}\ \texttt{skip}, \sigma\rangle \Downarrow \sigma'$ must have been obtained from the [IfT] and [Seq] rules:

  $$\frac{\dfrac{\langle S, \sigma\rangle \Downarrow \sigma'' \quad \langle \texttt{while}\ b\ \texttt{do}\ S, \sigma''\rangle \Downarrow \sigma'}{\langle S; \texttt{while}\ b\ \texttt{do}\ S, \sigma\rangle \Downarrow \sigma'} \qquad \mathcal{B}[\![b]\!]\ \sigma = \mathit{true}}{\langle \texttt{if}\ b\ \texttt{then}\ (S; \texttt{while}\ b\ \texttt{do}\ S)\ \texttt{else}\ \texttt{skip}, \sigma\rangle \Downarrow \sigma'}$$

So there exists some σ'' such that $\langle S, \sigma \rangle \Downarrow \sigma''$ and $\langle \texttt{while}\ b\ \texttt{do}\ S, \sigma'' \rangle \Downarrow \sigma'$. We can then apply the [WhileT] rule as follows:

$$\frac{\langle S, \sigma \rangle \Downarrow \sigma'' \quad \langle \texttt{while}\ b\ \texttt{do}\ S, \sigma'' \rangle \Downarrow \sigma' \quad \mathcal{B}[\![b]\!]\,\sigma = \mathit{true}}{\langle \texttt{while}\ b\ \texttt{do}\ S, \sigma \rangle \Downarrow \sigma'}$$