ETH zürich

P. Müller and D. Basin

Formal Methods and Functional Programming

Solutions of Exercise Sheet 10: States and Expressions

Assignment 1 (Simplifying State Updates)

Task 1.1: Let the state σ, the variable x, and the values v_1 and v_2 be arbitrary. We need to show $\forall y \cdot \sigma[x \mapsto v_1][x \mapsto v_2](y) = \sigma[x \mapsto v_2](y)$. Using the definition of state update, for an arbitrary variable y, we have

$$\begin{aligned}
\sigma[x \mapsto v_1][x \mapsto v_2](y) &= \begin{cases} v_2 & \text{if } y \equiv x \\ \sigma[x \mapsto v_1](y) & \text{if } y \not\equiv x \end{cases} \\
&= \begin{cases} v_2 & \text{if } y \equiv x \\ \sigma(y) & \text{if } y \not\equiv x \end{cases} \\
&= \sigma[x \mapsto v_2](y).
\end{aligned}$$

Task 1.2: Let the state σ, the variables x and y with $x \not\equiv y$, and the values v and w be arbitrary. We need to show that $\forall z \cdot \sigma[x \mapsto v][y \mapsto w] = \sigma[y \mapsto w][x \mapsto v]$. Using the definition of state update, for an arbitrary z, we have

$$\begin{aligned}
\sigma[x \mapsto v][y \mapsto w](z) &= \begin{cases} w & \text{if } z \equiv y \\ \sigma[x \mapsto v](z) & \text{if } z \not\equiv y \end{cases} \\
&= \begin{cases} w & \text{if } z \equiv y \\ v & \text{if } z \not\equiv y \text{ and } z \equiv x \\ \sigma(z) & \text{if } z \not\equiv y \text{ and } z \not\equiv x \end{cases} \\
&\stackrel{(*)}{=} \begin{cases} v & \text{if } z \equiv x \\ w & \text{if } z \not\equiv x \text{ and } z \equiv y \\ \sigma(z) & \text{if } z \not\equiv x \text{ and } z \not\equiv y \end{cases} \\
&= \begin{cases} v & \text{if } z \equiv x \\ \sigma[y \mapsto w](z) & \text{if } z \not\equiv x \end{cases}
\end{aligned}$$

$$= \sigma[y \mapsto w][x \mapsto v](z)$$

Note that the rewriting of the cases in the step marked with $(*)$ only works because we assumed that $x \not\equiv y$. And, indeed, the overall result is not true with this condition, as we see from the counter-example with $v = 1$, $w = 2$, and $x \equiv y \equiv z$:

$$\sigma[x \mapsto v][y \mapsto w](z) = w = 2 \neq 1 = v = \sigma[y \mapsto w][x \mapsto v](z)$$

Task 1.3: Let the variable x, the values v_1 and v_2 be arbitrary. We define

$$P(n) \equiv \forall \sigma, \vec{y}, \vec{w} \cdot \big(|\vec{y}| = |\vec{w}| = n \implies \sigma[x \mapsto v_1][\vec{y} \mapsto \vec{w}][x \mapsto v_2] = \sigma[\vec{y} \mapsto \vec{w}][x \mapsto v_2]\big)$$

and prove $\forall n \cdot P(n)$ by weak induction over n:

- **Base Case:** We show $P(0)$. We consider some arbitrary state σ and sequences $\vec{y}$ and $\vec{w}$ of variables and values, respectively. We assume $|\vec{y}| = |\vec{w}| = 0$. In this case, the sequences $\vec{y}$ and $\vec{w}$ can only be empty. Thus, the claim to be proved is $\sigma[x \mapsto v_1][x \mapsto v_2] = \sigma[x \mapsto v_2]$, which immediately follows from Task 1.1.

- **Step Case:** For some arbitrary n, we assume that $P(n)$ holds and aim to prove that $P(n+1)$ also holds. Let the state σ and the sequences $\vec{y}$ and $\vec{w}$ of variables and values, respectively, be arbitrary. We assume that $|\vec{y}| = |\vec{w}| = n + 1$, i.e., that $\vec{y} \equiv \langle y_1, \ldots, y_{n+1} \rangle$ and $\vec{w} = \langle w_1, \ldots, w_{n+1} \rangle$ for some appropriate variables y_i and values w_i.

  We proceed with a case analysis on the variable y_1:

  - **Case** $y_1 \equiv x$: By Task 1.1, we have $\sigma[x \mapsto v_1][y_1 \mapsto w_1] = \sigma[y_1 \mapsto w_1]$ and therefore

    $$\begin{aligned} &\sigma[x \mapsto v_1][y_1 \mapsto w_1][y_2 \mapsto w_2] \ldots [y_{n+1} \mapsto w_{n+1}][x \mapsto v_2] \\ = \; &\sigma[y_1 \mapsto w_1][y_2 \mapsto w_2] \ldots [y_{n+1} \mapsto w_{n+1}][x \mapsto v_2], \end{aligned}$$

    as required.

  - **Case** $y_1 \not\equiv x$: We have

    $$\begin{aligned} &\sigma[x \mapsto v_1][y_1 \mapsto w_1][y_2 \mapsto w_2] \ldots [y_{n+1} \mapsto w_{n+1}][x \mapsto v_2] \\ = \; &\sigma[y_1 \mapsto w_1][x \mapsto v_1][y_2 \mapsto w_2] \ldots [y_{n+1} \mapsto w_{n+1}][x \mapsto v_2] && \text{(Task 1.2)} \\ = \; &\sigma[y_1 \mapsto w_1][y_2 \mapsto w_2] \ldots [y_{n+1} \mapsto w_{n+1}][x \mapsto v_2] && \text{(IH)} \end{aligned}$$

    where the first equality follows from Task 1.2, and the second equality follows from the induction hypothesis $P(n)$ (instantiating the quantifiers as follows: $\sigma \rightsquigarrow \sigma[y_1 \mapsto w_1]$, $\vec{y} \rightsquigarrow \langle y_2, \ldots, y_{n+1} \rangle$, and $\vec{w} \rightsquigarrow \langle w_2, \ldots, w_{n+1} \rangle$; note that we are only able to conclude the desired claim since $\vec{y}$ and $\vec{w}$ are both instantiated with sequences of length n).

Assignment 2 (Substitution Properties)

Recall the definition of substitution on boolean expressions

$$b[x\mapsto e] \equiv \begin{cases} e_1[x\mapsto e]\ op\ e_2[x\mapsto e] & \text{if } b \equiv e_1\ op\ e_2 \\ \texttt{not}\ \ b'[x\mapsto e] & \text{if } b \equiv \texttt{not}\ \ b' \\ b_1[x\mapsto e] \circ b_2[x\mapsto e] & \text{if } b \equiv b_1 \circ b_2 \text{ for some } \circ \in \{\texttt{and}, \texttt{or}\} \end{cases} \tag{*}$$

and the lemma proved in the exercise session

$$\forall \sigma, e, e', x \cdot \big(\mathcal{A}[\![e[x\mapsto e']]\!]\sigma = \mathcal{A}[\![e]\!](\sigma[x\mapsto \mathcal{A}[\![e']\!]\sigma])\big). \tag{**}$$

Proof: Let the state σ, the variable x, and the expression e be arbitrary (note that we deal with the inner quantifiers first here; several consecutive for-all quantifiers can always be reordered). We define

$$P(b) \equiv \big(\mathcal{B}[\![b[x\mapsto e]]\!]\sigma = \mathcal{B}[\![b]\!](\sigma[x\mapsto \mathcal{A}[\![e]\!]\sigma])\big)$$

and prove $\forall b \cdot P(b)$ by structural induction on the boolean expression b:

- **Or Case:** We need to prove $P(b_1 \texttt{ or } b_2)$, for some boolean expressions b_1, b_2, and may assume $P(b_1)$ and $P(b_2)$ as our induction hypothesis. We have

$$\begin{aligned} \mathcal{B}[\![(b_1 \texttt{ or } b_2)[x\mapsto e]]\!]\sigma &= \mathcal{B}[\![b_1[x\mapsto e] \texttt{ or } b_2[x\mapsto e]]\!]\sigma && (*) \\ &= \mathcal{B}[\![b_1[x\mapsto e]]\!]\sigma \vee \mathcal{B}[\![b_2[x\mapsto e]]\!]\sigma && (\mathcal{B}) \\ &= \mathcal{B}[\![b_1]\!](\sigma[x\mapsto \mathcal{A}[\![e]\!]\sigma]) \vee \mathcal{B}[\![b_2]\!](\sigma[x\mapsto \mathcal{A}[\![e]\!]\sigma]) && (\text{IH}) \\ &= \mathcal{B}[\![b_1 \texttt{ or } b_2]\!](\sigma[x\mapsto \mathcal{A}[\![e]\!]\sigma]), && (\mathcal{B}) \end{aligned}$$

  where $\vee$ denotes the function that maps to $\mathbf{tt}$ if at least one of its arguments is $\mathbf{tt}$.

- **And Case:** Analogous to the previous case.

- **Not Case:** We need to prove $P(\texttt{not}\ \ b')$, for some boolean expression b', and may assume $P(b')$ as our induction hypothesis. We have

$$\begin{aligned} \mathcal{B}[\![(\texttt{not}\ \ b')[x\mapsto e]]\!]\sigma &= \mathcal{B}[\![\texttt{not}\ \ b'[x\mapsto e]]\!]\sigma && (*) \\ &= \neg\mathcal{B}[\![b'[x\mapsto e]]\!]\sigma && (\mathcal{B}) \\ &= \neg\mathcal{B}[\![b']\!](\sigma[x\mapsto \mathcal{A}[\![e]\!]\sigma]) && (\text{IH}) \\ &= \mathcal{B}[\![(\texttt{not}\ \ b')]\!](\sigma[x\mapsto \mathcal{A}[\![e]\!]\sigma]), && (\mathcal{B}) \end{aligned}$$

  where $\neg$ denotes the function that maps $\mathbf{tt}$ to $\mathbf{ff}$ and $\mathbf{ff}$ to $\mathbf{tt}$.

- **Relation Case:** We need to show $P(e_1\ op\ e_2)$, for some arithmetic expressions e_1, e_2 and and arithmetic relation op. We have

$$\begin{aligned} \mathcal{B}[\![(e_1\ op\ e_2)[x\mapsto e]]\!]\sigma &= \mathcal{B}[\![e_1[x\mapsto e]\ op\ e_2[x\mapsto e]]\!]\sigma && (*) \\ &= \mathcal{A}[\![e_1[x\mapsto e]]\!]\sigma\ \overline{op}\ \mathcal{A}[\![e_2[x\mapsto e]]\!]\sigma && (\mathcal{B}) \\ &= \mathcal{A}[\![e_1]\!](\sigma[x\mapsto \mathcal{A}[\![e]\!]\sigma])\ \overline{op}\ \mathcal{A}[\![e_2]\!](\sigma[x\mapsto \mathcal{A}[\![e]\!]\sigma]) && (**) \\ &= \mathcal{B}[\![e_1\ op\ e_2]\!](\sigma[x\mapsto \mathcal{A}[\![e]\!]\sigma]), && (\mathcal{B}) \end{aligned}$$

  where $\overline{op}$ denotes the operation corresponding to op.

Assignment 3 (Applying Big-Step Semantics)

We use the following abbreviations: l is the statement `(a := a+n; b := b*n); n := n-1` and w is the statement `while n#0 do` l `end`. To save space, we also introduce the following abbreviation: The notation

$$[v_1, v_2, v_3],$$

where v_1, v_2, v_3 are integer values, stands for the state

$$\sigma[\mathtt{a} \mapsto v_1][\mathtt{b} \mapsto v_2][\mathtt{n} \mapsto v_3],$$

where σ is the initial state mentioned in the exercise.

We construct the derivation tree shown in the following page.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\langle \texttt{a := a+n}, [0,1,2]\rangle \to [2,1,2]}\,(\textsc{Ass}_{NS}) \quad \dfrac{}{\langle \texttt{b := b*n}, [2,1,2]\rangle \to [2,2,2]}\,(\textsc{Ass}_{NS})}{\langle \texttt{a := a+n; b := b*n}, [0,1,2]\rangle \to [2,2,2]}\,(\textsc{Seq}_{NS}) \quad \dfrac{}{\langle \texttt{n := n-1}, [2,2,2]\rangle \to [2,2,1]}\,(\textsc{Ass}_{NS})}{\langle \texttt{(a := a+n; b := b*n); n := n-1}, [0,1,2]\rangle \to [2,2,1]}\,(\textsc{Seq}_{NS}) \quad \dfrac{T_1}{\langle \texttt{while n\#0 do l end}, [2,2,1]\rangle \to [3,2,0]}}{\langle \texttt{while n\#0 do l end}, [0,1,2]\rangle \to [3,2,0]}\,(\textsc{WhT}_{NS})$$

where T_1 is the following derivation tree:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\langle \texttt{a := a+n}, [2,2,1]\rangle \to [3,2,1]}\,(\textsc{Ass}_{NS}) \quad \dfrac{}{\langle \texttt{b := b*n}, [3,2,1]\rangle \to [3,2,1]}\,(\textsc{Ass}_{NS})}{\langle \texttt{a := a+n; b := b*n}, [2,2,1]\rangle \to [3,2,1]}\,(\textsc{Seq}_{NS}) \quad \dfrac{}{\langle \texttt{n := n-1}, [3,2,1]\rangle \to [3,2,0]}\,(\textsc{Ass}_{NS})}{\langle \texttt{(a := a+n; b := b*n); n := n-1}, [2,2,1]\rangle \to [3,2,0]}\,(\textsc{Seq}_{NS}) \quad \dfrac{}{\langle w, [3,2,0]\rangle \to [3,2,0]}\,(\textsc{WhF}_{NS})}{\langle \texttt{while n\#0 do l end}, [2,2,1]\rangle \to [3,2,0]}\,(\textsc{WhT}_{NS})$$