P. Müller and D. Basin

Formal Methods and Functional Programming

Session Sheet 10: IMP States and Expressions

Assignment 1 (Simplifying State Updates)

Task 1.1: Prove that for all states σ and variables x, it holds that $\sigma[x \mapsto \sigma(x)] = \sigma$.

Task 1.2: Assume that for all states σ, for all variables x, y, and for all values v, w:

$$x \not\equiv y \implies \sigma[x \mapsto v][y \mapsto w] = \sigma[y \mapsto w][x \mapsto v] \quad (1)$$

The proof of this statement is left for the exercise sheet.

Prove that for all variables x, for all values v, for all natural numbers n, for all sequences of length n of variables $\vec{y} \equiv \langle y_1, \ldots, y_n \rangle$ and corresponding values $\vec{w} \equiv \langle w_1, \ldots, w_n \rangle$, and for all states σ:

$$x \notin \vec{y} \implies \sigma[x \mapsto v][\vec{y} \mapsto \vec{w}] = \sigma[\vec{y} \mapsto \vec{w}][x \mapsto v].$$

Note: By $x \notin \vec{y}$, we mean that $x \not\equiv y_i$, for all $i \in \{1, \ldots, n\}$.

Note: We use $\sigma[\vec{y} \mapsto \vec{w}]$ to denote the sequence of updates $\sigma[y_1 \mapsto w_1] \ldots [y_n \mapsto w_n]$.

Assignment 2 (Substitution on Arithmetic Expressions)

Intuitively, $e[x \mapsto e']$ denotes the arithmetic expression e with all occurrences of x replaced with the arithmetic expression e'. Recall the formal definition:

$$e[x \mapsto e'] \equiv \begin{cases} n & \text{if } e \equiv n \text{ for some numerical value } n \\ e' & \text{if } e \equiv y \text{ for some variable } y \text{ with } y \equiv x \\ y & \text{if } e \equiv y \text{ for some variable } y \text{ with } y \not\equiv x \\ e_1[x \mapsto e'] \mathbin{op} e_2[x \mapsto e'] & \text{if } e \equiv e_1 \mathbin{op} e_2 \text{, for some } e_1, e_2 \text{, and } op \end{cases}$$

Task: Prove the following statement:

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

Hint: Define a suitable predicate $P(e)$ and prove $\forall e \cdot P(e)$ by either *weak structural induction* or *strong structural induction* on the arithmetic expression e. If you choose to do a strong structural induction, you have to prove $P(e)$ for some arbitrary e and may assume $\forall e'' \sqsubset e \cdot P(e'')$ as your induction hypothesis. Note that, here, $e'' \sqsubset e$ denotes that e'' is a proper sub-expression of e. Since arithmetic expressions are finite, the relation $\sqsubset$ is a well-founded ordering. Thus, strong structural induction on arithmetic expressions can be seen as a special case of well-founded induction.

Assignment 3 (Big-Step Semantics)

Let s be the following statement:

```
y := 1;
while x > 0 do
   y := y * 2;
   x := x - 1
end
```

Task 3.1. What function does the **IMP** statement s compute when the variable x initially stores a non-negative integer?

Task 3.2. Let σ be a state with $\sigma(\mathtt{x}) = 2$. Prove that there is a state σ' with $\sigma'(\mathtt{y}) = 4$ such that $\langle s, \sigma \rangle \to \sigma'$ using the rules of the big-step semantics for **IMP**.