Technische Universität Wien
Fakultät für Informatik
Assist. Prof. Florian Zuleger
Assist. Prof. Georg Weissenbacher
Yulia Demyanova, Mag.
Moritz Sinn, M.Sc.

SS 2015
Exercise 1
Thursday, 26 March 2015

Exercises on Semantics of Programming Languages

Solutions are to be handed in at the lecture on April 13th. Later submissions will not be accepted.

Exercise 1 Determinism of Big-step Semantics **(5 Points)**

Consider the Big-step semantics of the **While** language as defined in the lecture. Prove the following theorem from the lecture:

If $\langle C, s\rangle \Downarrow s_1$ and $\langle C, s\rangle \Downarrow s_2$ then $s_1 = s_2$.

Hint: Use rule-based induction.

Exercise 2 Properties of Small-step Semantics **(5 Points)**

Proof the following theorems from the lecture:

a) If $\langle C_1; C_2, s\rangle \to^k s'$ then there exists a state s'' and natural numbers k_1 and k_2 s.t. $\langle C_1; s\rangle \to^{k_1} s''$ and $\langle C_2; s''\rangle \to^{k_2} s'$ where $k_1 + k_2 = k$.

b) If $\langle C_1, s\rangle \to^k s'$ then $\langle C_1; C_2, s\rangle \to^k \langle C_2, s'\rangle$.

c) Does b) also hold the other way around? I.e., does $\langle C_1, s\rangle \to^k s'$ follow from $\langle C_1; C_2, s\rangle \to^k \langle C_2, s'\rangle$? Prove or disprove.

Exercise 3 Equivalence of Small-step and Big-step Semantics **(5 Points)**

Consider the definition of Small-step semantics given in the lecture and the following alternative rule for the while-construct:

$$\text{S-WHILE}\quad \frac{}{\langle \text{while b do } C, s\rangle \to \langle \text{if b then } (C;\ \text{while b do } C) \text{ else skip}, s\rangle}$$

Let $[\![C]\!]_{S'}$ be defined as:

$$[\![C]\!]_{S'} = \begin{cases} s' \text{ if } \langle C, s\rangle \to^* s' \text{ using S-WHILE instead of S-WHILE.T and S-WHILE.F} \\ \bot \text{ otherwise} \end{cases}$$

Proof that $[\![C]\!]_{S'} = [\![C]\!]_B$. You may cite appropriate parts of the proof for $[\![C]\!]_S = [\![C]\!]_B$ given in the lecture.

Exercise 4 Small-step Semantics of Arithmetic Expressions **(5 Points)**

In Exercise 1 from the repetition sheet (dated 26th of March)(*) Big-step semantics of arithmetic expressions are defined, i.e., by the specified rules a given arithmetic expression is evaluated to an integer in one step. We give an example:

Consider the state $s = \{x \mapsto 5, y \mapsto 3, z \mapsto 4\}$. With the definition of $\rightarrow_{Aexp}$ in (*) it holds that $\langle (x + y) + z, s\rangle \rightarrow_{Aexp} 12$.

We now want to define Small-step semantics of arithmetic expressions by a derivation relation $\langle e, s\rangle \rightarrow_{AS} \gamma$ where γ is either of the form $\langle e', s\rangle$ or n where n is a numeral. If γ is of form $\langle e', s\rangle$ then the evaluation of e in state s is *not* completed and the partial evaluation is expressd by the intermediate configuration $\langle e', s\rangle$. If γ is of form n then e was evaluated to n in s.

E.g., instead of evaluating $\langle (x + y) + z, s\rangle$ directly to its final value 12 in s, this expression is evaluated as follows:
$\langle (x + y) + z, s\rangle \rightarrow_{AS} \langle (5 + y) + z, s\rangle \rightarrow_{AS} \langle (5 + 3) + z, s\rangle \rightarrow_{AS} \langle 8 + z, s\rangle \rightarrow_{AS} \langle 8 + 4, s\rangle \rightarrow_{AS} 12$

a) Define Small-step semantics of arithmetic expressions by defining the derivation relation $\langle e, s\rangle \rightarrow_{AS} \gamma$ appropriately. It is sufficient to consider the addition operator $+$ since the operators $*$ and $-$ can be handled accordingly.

b) Extend the definition of Small-step semantics of the WHILE language given in the lecture s.t. arithmetic expressions are evaluated stepwise (by $\rightarrow_{AS}$) in assignments. I.e., given an assignment $x := e$ where e is an arithmetic expression, e shall first be evaluated stepwise to a numeral before the state is finaly updated in order to handle the assignment.