Technische Universität Wien
Fakultät für Informatik
Assist. Prof. Florian Zuleger
Assist. Prof. Georg Weissenbacher
Univ. Prof. Agata Ciabattoni
Moritz Sinn, M.Sc.

SS 2014
Repetition sheet
Tuesday, 8 April 2014

Exercises on Semantics of Programming Languages

<u>Exercise 1</u> Proof by induction

Consider the **While** language introduced in the lectures. We define the natural semantics of arithmetic expressions as follows.

It has two kinds of configurations:

$\langle a, s\rangle$	denoting that a has to be evaluated in state s, and
z	denoting the final value (an element of $\mathbf{Z}$)

The transition relation $\to_{Aexp}$ has the form: $\langle a, z\rangle \to_{Aexp} z$. The semantics is given in Table 1.

$$\langle n, s\rangle \to_{Aexp} \mathcal{N}[\![n]\!]$$

$$\langle x, s\rangle \to_{Aexp} s\ x$$

$$\frac{\langle a_1, s\rangle \to_{Aexp} z_1,\ \langle a_2, s\rangle \to_{Aexp} z_2}{\langle a_1 + a_2, s\rangle \to_{Aexp} z} \text{ where } z = z_1 + z2$$

$$\frac{\langle a_1, s\rangle \to_{Aexp} z_1,\ \langle a_2, s\rangle \to_{Aexp} z_2}{\langle a_1 * a_2, s\rangle \to_{Aexp} z} \text{ where } z = z_1 * z2$$

$$\frac{\langle a_1, s\rangle \to_{Aexp} z_1,\ \langle a_2, s\rangle \to_{Aexp} z_2}{\langle a_1 - a_2, s\rangle \to_{Aexp} z} \text{ where } z = z_1 - z2$$

Table 1: Natural semantics of arithmetic expressions

Let us define a function $\mathcal{A} : \mathbf{Aexp} \to (\mathbf{State} \to \mathbf{Z})$ as follows:

$$\forall a \in \mathbf{Aexp},\ \forall s \in \mathbf{State}.\ \langle a, s\rangle \to_{Aexp} z \Leftrightarrow \mathcal{A}[\![a]\!]s = z$$

Prove that $\mathcal{A}$ is a total function.

Hint1: You need to prove that

- $\mathcal{A}$ is well-defined (i.e. there is at most one fuction value for each $a \in \mathbf{Aexp}$)
- $\mathcal{A}$ is total (i.e. function is defined for each $a \in \mathbf{Aexp}$).

To summarise, you have to show that for each $a \in \mathbf{Aexp}$ and each $s \in \mathbf{State}$ there is exactly one value $\mathbf{v} \in \mathbf{Z}$ such that $\mathcal{A}[\![a]\!]s = \mathbf{v}$.

Hint 2: Use structural induction on the arithmetic expressions for the proof:

- First show that the property holds for every basis element of the syntactic category.
- Then prove that the property holds for every composite element using the assumption that it holds for all the immediate consituents of the element.

<u>**Exercise 2**</u> Extension of boolean expressions. Semantic equivalence.

Again consider the **While** language introduced in the lectures. The semantic function for boolean expressions is given by the function

$$\mathcal{B} : \mathbf{Bexp} \to (\mathbf{State} \to \{\mathbf{tt}, \mathbf{ff}\})$$

The definition of $\mathcal{B}$ is given in Table 2.

$$\begin{array}{lcl}
\mathcal{B}[\![true]\!]s & = & \mathbf{tt} \\
\mathcal{B}[\![false]\!]s & = & \mathbf{ff} \\
\mathcal{B}[\![a_1 = a_2]\!]s & = & \begin{cases} \mathbf{tt} & \text{if } \mathcal{A}[\![a_1]\!]s = \mathcal{A}[\![a_2]\!]s \\ \mathbf{ff} & \text{if } \mathcal{A}[\![a_1]\!]s \neq \mathcal{A}[\![a_2]\!]s \end{cases} \\
\mathcal{B}[\![a_1 \leq a_2]\!]s & = & \begin{cases} \mathbf{tt} & \text{if } \mathcal{A}[\![a_1]\!]s \leq \mathcal{A}[\![a_2]\!]s \\ \mathbf{ff} & \text{if } \mathcal{A}[\![a_1]\!]s > \mathcal{A}[\![a_2]\!]s \end{cases} \\
\mathcal{B}[\![\neg b]\!]s & = & \begin{cases} \mathbf{tt} & \text{if } \mathcal{B}[\![b]\!]s = \mathbf{ff} \\ \mathbf{ff} & \text{if } \mathcal{B}[\![b]\!]s = \mathbf{tt} \end{cases} \\
\mathcal{B}[\![b_1 \wedge b_2]\!]s & = & \begin{cases} \mathbf{tt} & \text{if } \mathcal{B}[\![b_1]\!]s = \mathbf{tt} \text{ and } \mathcal{B}[\![b_2]\!]s = \mathbf{tt} \\ \mathbf{ff} & \text{if } \mathcal{B}[\![b_1]\!]s = \mathbf{ff} \text{ or } \mathcal{B}[\![b_2]\!]s = \mathbf{ff} \end{cases}
\end{array}$$

Table 2: The semantics of boolean expressions

The syntactic category **Bexp'** is defined as the following extension of **Bexp**:

$$\begin{array}{l}
b \ ::= \ true \mid false \mid a_1 = a_2 \mid a_1 \neq a_2 \mid a_1 \leq a_2 \mid a_1 \geq a_2 \\
\quad \mid a_1 < a_2 \mid a_1 > a_2 \mid \neg b \mid b_1 \vee b_2 \mid b_1 \wedge b_2 \\
\quad \mid b_1 \Rightarrow b_2 \mid b_1 \Leftrightarrow b_2
\end{array}$$

a) Give a *compositional* extension of the semantic function $\mathcal{B}$ of Table 2, i.e. for every new expression provide a semantic clause in terms of the semantics of their immediate constituents.

b) Two boolean expressions b_1 and b_2 are *equivalent* if for all states s,

$$\mathcal{B}[\![b_1]\!]s = \mathcal{B}[\![b_2]\!]s$$

Show that for each b' of **Bexp'** there exists a boolean expression b of **Bexp** such that b' and b are equivalent.

<u>Exercise 3</u> Substitution

We define semantic function $\mathcal{A}$ for arithmetic expression in Table 3.

$$\begin{array}{lcl}
\mathcal{A}[\![n]\!]s & = & \mathcal{N}[\![n]\!] \\
\mathcal{A}[\![x, s]\!] & = & s\ x \\
\mathcal{A}[\![a_1 + a_2]\!]s & = & \mathcal{A}[\![a_1]\!]s + \mathcal{A}[\![a_2]\!]s \\
\mathcal{A}[\![a_1 - a_2]\!]s & = & \mathcal{A}[\![a_1]\!]s - \mathcal{A}[\![a_2]\!]s \\
\mathcal{A}[\![a_1 * a_2]\!]s & = & \mathcal{A}[\![a_1]\!]s * \mathcal{A}[\![a_2]\!]s
\end{array}$$

Table 3: Semantic function for arithmetic expressions

We define substitution for arithmetic expressions $a[y \mapsto a_0]$ in Table 4.

$$\begin{array}{lcl}
n[y \mapsto a_0] & = & n \\
x[y \mapsto a_0] & = & \begin{cases} a_0 & \text{if } x = y \\ x & \text{if } x \neq y \end{cases} \\
(a_1 + a_2)[y \mapsto a_0] & = & a_1[y \mapsto a_0] + a_2[y \mapsto a_0] \\
(a_1 * a_2)[y \mapsto a_0] & = & a_1[y \mapsto a_0] * a_2[y \mapsto a_0] \\
(a_1 - a_2)[y \mapsto a_0] & = & a_1[y \mapsto a_0] - a_2[y \mapsto a_0]
\end{array}$$

Table 4: Substitution for arithmetic expressions

We also define a notion of substitution for states:

$$(s[y \mapsto v])x = \begin{cases} v & \text{if } x = y \\ s\ x & \text{if } x \neq y \end{cases}$$

Prove that the following property holds:

$$\mathcal{A}[\![a[y \mapsto a_0]]\!]s = \mathcal{A}[\![a]\!](s[y \mapsto \mathcal{A}[\![a_0]\!]s])$$

Exercise 4 Determinism of Structural Operational Semantics

Consider the structural operational semantics of the **While** language from Table 5.

$[\text{ass}_{sos}]$	$\langle x := a, s\rangle \Rightarrow s[x \mapsto \mathcal{A}[\![x]\!]s]$
$[\text{skip}_{sos}]$	$\langle skip, s\rangle \Rightarrow s$
$[\text{comp}^1_{sos}]$	$\dfrac{\langle S_1, s\rangle \Rightarrow \langle S_1', s'\rangle}{\langle S_1; S_2\rangle \Rightarrow \langle S_1'; S_2, s'\rangle}$
$[\text{comp}^2_{sos}]$	$\dfrac{\langle S_1, s\rangle \Rightarrow s'}{\langle S_1; S_2\rangle \Rightarrow \langle S_2, s'\rangle}$
$[\text{if}^{tt}_{sos}]$	$\langle \text{if b then } S_1 \text{ else } S_2, s\rangle \Rightarrow \langle S_1, s\rangle$ if $\mathcal{B}[\![b]\!]s = \mathbf{tt}$
$[\text{if}^{ff}_{sos}]$	$\langle \text{if b then } S_1 \text{ else } S_2, s\rangle \Rightarrow \langle S_2, s\rangle$ if $\mathcal{B}[\![b]\!]s = \mathbf{ff}$
$[\text{while}_{sos}]$	$\langle \text{while b do } S\rangle \Rightarrow \langle \text{if b then } (S;\ \text{while b do } S) \text{ else skip}, s\rangle$

Table 5: Structural operational semantics of **While**

Prove that the semantics is deterministic, i.e. for all choices of S, s, γ and γ' we have that
$\langle S, s\rangle \Rightarrow \gamma$ and $\langle S, s\rangle \Rightarrow \gamma'$ imply $\gamma = \gamma'$

Hint: Use rule induction:

- Prove that the property holds for all *axioms* of the transition system.
- Prove that the property holds for all *composite rules*: for each rule assume that the property holds for its premises (this is called the induction hypothesis) and prove that it also holds for the conclusion of the rule provided that the conditions of the rule are satisfied.

Exercise 5 Operational semantics

We extend the language **While** with the statement

`for x :=` a_1 `to` a_2 `do` S

a) Define the semantics of the `for` statement with

1. natural semantics
2. structural operational semantics

Note: The semantics of the `for`-construct is not allowed to rely on the existence of the `while`-construct in the language.

b) Evaluate the statement

```
y := 1; for z := 1 to x do (y := y*x; x := x-1)
```

from a state where `x` has the value 5 using both semantics.

Provide a derivation tree for the evaluation which uses natural semantics.
For the evaluation which uses structural operational semantics provide a derivation sequence.