Lund University
Computer Science
Lennart Andersson

Seminar 1
EDA145
2011-03-25

Problems

These problems will be discussed on the seminar in week 2. Programs in Haskell are not required to compile.

---

1 Define multiplication and exponentiation for the datatype `N` given that

```
data N = Zero | Suc N

add m Zero = m
add m (Suc n) = Suc (add m n)
```

(This problem has been slightly changed 2011-03-23. Either solution is acceptable, but the optional part becomes more interesting this way.)

Optional problem. Generalize this construction by defining functions beyond exponentiation. Define a recursive function

```
f :: Integer -> Integer -> Integer -> Integer
```

such that

```
f 0 m n = add m n
f 1 m n = mul m n
f 2 m n = exp m n
...
```

2 Prove

Lemma. *Let `m` be any element in `N`. Then `add n (Suc m) = add (Suc n) m` for all `n` in `N`.*

3 State an induction principle for

```
data Tree a = Leaf a | Node (Tree a) (Tree a)
```

4 Define the $\mathcal{N}$ for the abstract grammar (Exercise 1.4)

```
n ::= 0 | 1 | 0 n | 1 n
```

5 Prove that $\mathcal{A}[\![a[y \mapsto a_0]]\!]\sigma = \mathcal{A}[\![a]\!](\sigma[y \mapsto \mathcal{A}[\![a_0]\!]\sigma])$. (Exercise 1.14.)

6 Define substitution for boolean expressions $b[y \mapsto a_0]$ and prove that $\mathcal{B}[\![b[y \mapsto a_0]]\!]\sigma = \mathcal{B}[\![b]\!](\sigma[y \mapsto \mathcal{A}[\![a_0]\!]\sigma])$. (Exercise 1.15.)

7 Define a natural semantics for binary numerals, **Num**, defined by the grammar $n ::= 0 \mid 1 \mid n\ 0 \mid n\ 1$. The formulae of the theory should be on the form $\langle n \rangle \to z$, where $n \in \mathbf{Num}$ and $z \in \mathbb{N}$ is a natural number.

8 With the semantics of the previous problem and $\mathcal{N}$ from the text book use structural induction and induction over the shape of derivation trees to prove that $\langle n \rangle \to z$ if and only if $\mathcal{N}[\![n]\!] = z$.

9 Extend the language **While** with an if-statement without an else part and define natural semantics inference rules for it. The rules must not rely on the existence of the standard if-statement.

10 Extend the language **While** with a Java-like `do` S `while` b statement and define structural operational semantics inference rules for the statement. The rules must not rely on the existence of the standard while-statement.

11 Will the semantics of **While** change if we replace [$\text{if}_{\text{ns}}^{\text{tt}}$] and [$\text{if}_{\text{ns}}^{\text{ff}}$] by

$$\frac{\langle S_0, s\rangle \to s'}{\langle \texttt{if true then } S_0 \texttt{ else } S_1, s\rangle \to s'} \qquad \frac{\langle S_1, s\rangle \to s'}{\langle \texttt{if false then } S_0 \texttt{ else } S_1, s\rangle \to s'}$$

How about

$$\frac{\langle S_0, s\rangle \to s'_0 \qquad \langle S_1, s\rangle \to s'_1}{\langle \texttt{if b then } S_0 \texttt{ else } S_1, s\rangle \to s'}$$

where $s' = s'_0$ if $\mathcal{B}[\![\texttt{b}]\!]s = \text{tt}$ and $s' = s'_1$ otherwise.

12 (Exercise 2.21) Prove that

$$\langle S_1, s\rangle \Rightarrow^k s' \text{ implies } \langle S_1; S_2, s\rangle \Rightarrow^k \langle S_2, s'\rangle$$

13 (Exercise 2.20) Find statements S_1 and S_2 and states s and s' such that

$$\langle S_1; S_2, s\rangle \Rightarrow^* \langle S_2, s'\rangle$$

holds while $\langle S_1, s\rangle \Rightarrow^* s'$ does not.