Semantics, WS 2005 – Assignment 1

Prof. Dr. Gert Smolka, Dipl.-Inform. Andreas Rossberg
http://www.ps.uni-sb.de/courses/sem-ws05/

Recommended reading: Types and Programming Languages, chapters 1–5, emphasis on 5

We consider variables, numbers, terms and values as follows:

$$
\begin{aligned}
x &\in \mathit{Var} \\
n &\in \mathbb{N} \\
t &\in \mathit{Ter} \;=\; x \mid t\,t \mid \lambda x.\,t \mid n \mid S \\
v &\in \mathit{Val} \;=\; \lambda x.\,t \mid n \mid S
\end{aligned}
$$

A term is *pure* if it doesn't contain numbers or the successor operator S. The reduction relation $\rightarrow \subseteq \mathit{Ter}^2$ is defined as follows:

$$
\textbf{Beta}\ \frac{}{(\lambda x.\,t)v \rightarrow t[x := v]} \qquad \textbf{S}\ \frac{n' = n+1}{Sn \rightarrow n'}
$$

$$
\textbf{DAL}\ \frac{t_1 \rightarrow t_1'}{t_1 t_2 \rightarrow t_1' t_2} \qquad \textbf{DAR}\ \frac{t \rightarrow t'}{vt \rightarrow vt'}
$$

A *procedure* is a closed term of the form $\lambda x.\,t$. Boolean values, pairs and the natural numbers can be represented as pure values as follows:

$$
\begin{aligned}
\mathit{true} &\stackrel{\text{def}}{=} \lambda xy.\,x \\
\mathit{false} &\stackrel{\text{def}}{=} \lambda xy.\,y \\
(t_1, t_2) &\stackrel{\text{def}}{=} (\lambda xyf.\,fxy)t_1\; t_2 \\
c_0 &\stackrel{\text{def}}{=} \lambda fs.\,s \\
c_n &\stackrel{\text{def}}{=} \lambda fs.\,c_{n-1}f(fs) \quad (n \geq 1)
\end{aligned}
$$

Exercise 1.1: Numbers We say that a *term t represents a number n* if t is pure and the term $tS0$ evalutes to n. Find a pure procedure

(a) *add* that given values representing m and n yields a value representing $m + n$.

(b) *mul* that given values representing m and n yields a value representing $m \cdot n$.

(c) *exp* that given values representing m and n yields a value representing m^n.

(d) *fac* that given a value representing n yields a value representing $n!$.

(e) *pre* that given a value representing n yields a value representing $\max\{0, n-1\}$.

(f) *sub* that given values representing m and n yields a value representing $\max\{0, m-n\}$.

(g) *leq* that given values representing m and n tests wether $m \leq n$.

(h) *foo* that given a value representing n diverges if and only if $n > 0$.

(i) *chu2int* that given a value representing n yields the value n (this procedure has to be impure).

Exercise 1.2: Lists Let lists be represented as pure terms as follows (think of *foldl* in Standard ML):

$$\begin{aligned} nil &\stackrel{\text{def}}{=} \lambda f s . s \\ x :: y &\stackrel{\text{def}}{=} \lambda f s . y f (f x s) \end{aligned}$$

Find a pure procedure

(a) *null* that tests wether a list is empty.

(b) *hd* that yields the head of a list.

(c) *rev* that reverses a list.

(d) *tl* that yields the tail of a list.

Exercise 1.3: Implementation in Standard ML We implement the lambda terms of the lambda calculus introduced in the lecture as follows:

```
type var = string
datatype ter = V of var | A of ter*ter | L of var*ter | I of int | S
```

(a) Declare a procedure $isvalue : ter \to bool$ that tests whether a term is a value.

(b) Declare a procedure $closed : var\ list \to ter \to bool$ that tests for a list *xs* and a term t whether all free variables of t occur in *xs*.

(c) Declare a procedure $subst : ter \to var \to ter \to ter$ that yields for a term t, a variable x and a closed (!) term u the term $t[x := u]$.

(d) Declare a procedure $chu : int \to ter$ that for $n \in \mathbb{N}$ yields a value representing n.

(e) Declare a procedure $reduce : ter \to ter$ that attempts to reduce a term (by one step). If the term is not reducible, the exception *Error* should be thrown.

(f) Declare a procedure *eval* : *ter* → *ter* that evaluates a term. If repeated reduction of a term yiels a non-reducible term that is not a value, the exception *Error* should be thrown.

(g) Check that your procedures for Church numerals from the first exercise do what they are supposed to do. For instance, try $A(chu2int,\ A(A(mul,\ chu\ 5),\ chu\ 9))$.