Semantics, WS 2005 – Assignment 2

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

Recommended reading: Types and Programming Languages, chapters 6–7

Exercise 2.1: An ADT for λ-terms Under the URL

http://www.ps.uni-sb.de/courses/sem-ws05/assignments/lambda.sml

you can find ML implementations of pure λ-terms that ensure absence of capturing with the following signature:

```
eqtype var
type term
datatype view = Var of var | App of term * term | Lam of var * term

val var  : unit -> var
val term : view -> term
val view : term -> view
```

Use this interface to define the following procedures.

(a) Write a procedure *free* : *term* → *var list* that returns a list of the free variables contained in a term.

(b) Write a procedure *subst* : (*var* × *term*) → *term* → *term* such that *subst* $(x, t)\ t'$ substitutes t for x in t'.

(c) A *redex* ("reducible expression") is a λ-term of the form $(\lambda x.\, t_1)t_2$. In the following, we say that a λ-term is in *normal form* if it contains no subterms that are redexes.

Write a procedure *simplify* : *term* → *term* that given a term t either finds a redex in t and simplifies it by applying general β-reduction once:

$$(\lambda x.\, t_1)t_2 \rightarrow t_1[x := t_2]$$

or raises the exception *NF* if t is in normal form already.

(d) (Challenge) Consider the procedure

```
fun nf t = nf (simplify t) handle NF => t
```

that tries to transforms a term into normal form. Give an example of a term t for which *nf* t will either terminate or diverge, depending on which redex the *simplify* procedure chooses to reduce first.

Exercise 2.2: De Bruijn Notation

(a) Given a naming context $\{x \mapsto 0, y \mapsto 1, z \mapsto 2\}$, express the following terms in de Bruijn notation:

(i) $\lambda xy.\, zyx$

(ii) $(\lambda x.\, xy(\lambda y.\, yx))yx$

(iii) $\lambda x.\, (\lambda z.\, xz)(\lambda y.\, yz)$

(iv) $\lambda v.\, v(\lambda xw.\, wx(\lambda v.\, xyzvw))$

(b) Give the result of the following substitutions:

(i) $((\lambda 10)0)[0 := \lambda 0]$

(ii) $(\lambda\lambda 021)[0 := \lambda 0]$

(iii) $(\lambda 2(\lambda 032)01)[1 := \lambda\lambda 10]$

(iv) $(\lambda(\lambda 2301)321)[2 := \lambda 01]$

Exercise 2.3: Environment-based Interpreter Recall the syntax and big-step evaluation rules for the impure λ-terms in de Bruijn representation:

$$\begin{aligned} x \in Var &= \mathbb{N} \\ t \in Ter &= x \mid tt \mid \lambda t \mid S \mid \text{`}n\text{'} \\ v \in Val &= \langle E, t\rangle \mid S \mid n \\ E \in Env &= \epsilon \mid E, v \end{aligned}$$

$$\frac{}{E, v \vdash 0 \Rightarrow v} \qquad \frac{E \vdash x \Rightarrow v'}{E, v \vdash x+1 \Rightarrow v'} \qquad \frac{}{E \vdash \text{`}n\text{'} \Rightarrow n} \qquad \frac{}{E \vdash S \Rightarrow S} \qquad \frac{}{E \vdash \lambda t \Rightarrow \langle E, t\rangle}$$

$$\frac{E \vdash t_1 \Rightarrow S \qquad E \vdash t_2 \Rightarrow n}{E \vdash t_1 t_2 \Rightarrow n+1} \qquad \frac{E \vdash t_1 \Rightarrow \langle E', t\rangle \qquad E \vdash t_2 \Rightarrow v_2 \qquad E', v_2 \vdash t \Rightarrow v}{E \vdash t_1 t_2 \Rightarrow v}$$

Using the following type definitions,

```
datatype term  = Var of int | App of term * term | Lam of term | Lit of int | Suc
datatype value = Clos of env * term | Nat of int | Succ
withtype env   = value list
```

write a procedure $eval : term-> value$ that evaluates a term according to the above rules. The procedure should raise an exception *Error* in case evaluation gets stuck.