Assignment 3
Semantics, WS 2007/08

Prof. Dr. Gert Smolka, Dr. Jan Schwinghammer
www.ps.uni-sb.de/courses/sem-ws07/

Read Chapter 2 of the lecture notes

Exercise 3.1 (Reduction) Write a procedure $reduce : ter \to ter$ such that $reduce\ t$ yields the term t' with which the reduction relation terminates (i.e., where $t \to \cdots \to t'$ and $\neg\exists t'' :\ t' \to t''$). **Hint:** use the coincidence between big-step and small-step semantics, and rework the procedure *eval* from the previous assignment. The procedure *isVal* from Exercise 2.3 may be useful.

Exercise 3.2 (Contexts and Redices) Consider the following terms.

a) $pred(succ\ 0)$

b) $\lambda x{:}T.((\lambda y{:}T.y)\ x)$

c) $(\lambda x{:}T.((\lambda y{:}T.y)\ x))\ (iszero\ 0)$

d) $(0\ 0)\ (iszero\ 0)$

For each of these terms t,

- determine if t is reducible, and find all redices s in t;
- determine all pairs of evaluation contexts C and terms s such that $t = Cs$;
- determine the reduction context C and reduction redex s of t, if they exist.

Exercise 3.3 (RedContext and split) In Standard ML we can be represent contexts as procedures $c : ter \to ter$ such that $c\ t$ yields the context applied to the term t.

a) Write a procedure $redContext : ter \to (ter \to ter)$ that yields the reduction context of a reducible term. For example, if t is the term $(\lambda x{:}T.((\lambda y{:}T.y)\ x))\ (iszero\ 0)$ then:

```
> val c = redContext t
val c : ter -> ter = _fn
> val t' = c True
val t' : ter = A (L ("x", Int, L ("y", Int, V "y")), True)
```

b) Now extend this to a procedure $split : ter \to (ter \to ter) * ter$ that yields both the reduction context and the redex of a reducible term. For instance,

```
> val (c,r) = split t
val c : ter -> ter = _fn
val r : ter = Iszero O
```

The procedure *isVal* from the last assignment may be useful.

Exercise 3.4 (Closure semantics) To implement the closure semantics for PCF, we represent semantic values as follows:

```
datatype value = B of bool
               | N of int
               | C of var * ter * env
               | R of var * var * ter * env
withtype env = var -> value
```

Write a procedure *clEval* : *ter* → *value* that yields the semantic value of a term, if it exists. Raise an exception if a term of the form *fix t* is encountered where *t* does not satisfy the syntactic restrictions.

Exercise 3.5 (Compilation of *fix* terms) The compilation step proposed in the lecture notes (omitting type annotations),

$$\mathit{fix}\, t \rightsquigarrow (\lambda p.\mathit{fix}(\lambda f.\lambda x.p\, f\, x))t$$

does not always preserve the behaviour of programs. Find a term t where *fix t* and $(\lambda p.\mathit{fix}(\lambda f.\lambda x.p\, f\, x))t$ are not semantically equivalent.