Assignment 9
Semantics, WS 2007/08

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

Read Section 4.9 in the lecture notes and Chapters 13 and 14 of Pierce's TAPL

Exercise 9.1 (Contextual equivalence) In the lambda calculus with printing, give separating contexts for the following programs:

a) $\mathit{print}\ \sigma$ and $\mathit{print}\ \sigma'$, where $\sigma \neq \sigma'$

b) $x()$ and $x();x()$

Show that $t \Downarrow v|\sigma \ \Rightarrow\ t \sim_{\mathit{print}} v$ does not hold.

Exercise 9.2 (Sequence generators in SML) By a *generator for a sequence of natural numbers* $a = (a_i)_{i\in\mathbb{N}}$ we mean a procedure of type $\mathit{unit} \to \mathit{int}$ which returns a_i upon the i-th call. In Standard ML, implement the following procedures:

a) a generator *squares* for the sequence of square numbers $0, 1, 4, 9, 16, \ldots$

b) a (non-recursive) generator *fac* for the factorial numbers $1, 1, 2, 6, 24, \ldots$

c) a generator *fibs* for the sequence of Fibonacci numbers $0, 1, 1, 2, 3, \ldots$

d) a procedure $\mathit{newGen} : (\mathit{int} \to \mathit{int}) \to \mathit{unit} \to \mathit{int}$ that returns a generator for the sequence $f\ 0, f\ 1, f\ 2, \ldots$ when given f.

Exercise 9.3 (Resettable counters in SML)

a) Complete the following SML declaration

```
val (count, inc, reset) =
```

such that it yields an encapsulated counter with initial value 0 and three procedures to access and modify the counter as follows:

- $\mathit{count} : \mathit{unit} \to \mathit{int}$ returns the value of the counter.
- $\mathit{inc} : \mathit{unit} \to \mathit{unit}$ increases the value of the counter by one.
- $\mathit{reset} : \mathit{unit} \to \mathit{unit}$ resets the counter to its initial state.

b) Write a procedure

$$\mathit{newCounter} : \mathit{int} \to (\mathit{unit} \to \mathit{int}) \times (\mathit{unit} \to \mathit{unit}) \times (\mathit{unit} \to \mathit{unit})$$

such that each call with argument n yields a new counter with initial value n.

Exercise 9.4 (Semantics of lambda calculus with references)

a) Make sure that you can state the proper reduction rules, reduction contexts, contextual equivalence, typing relation and the progress and preservation properties for the lambda calculus with references.

b) Give an equivalent bigstep semantics.

c) Give an example of a well-typed, closed term such that its evaluation creates a heap μ with a cycle $\mu(l) = \lambda x{:}Unit.(!l)x$, for some l.

d) Find Γ, μ, and $\Sigma_1 \neq \Sigma_2$ such that $\Gamma|\Sigma_i \vdash \mu$ holds for $i = 1, 2$.

e) At some types it is not necessary to add the comparison operator $r = s$ on references as a new primitive: give a closed term t (depending on r, s) such that t evaluates to *true* if r and s evaluate to the same location, and to *false* otherwise, where

- r and s have type $Ref\ Int$;
- r and s have type $Ref\ Bool$.

Use SML to test your terms.

Exercise 9.5 (Recursion)

a) Show that the reduction relation is not normalizing on well-typed terms of the lambda calculus with references. **Hint:** Use a reference to a location of type $1 \to 1$. Test your term using SML.

b) Let T be any type. Find a well-typed term $Fix : ((T \to T) \to T \to T) \to T \to T$ that evaluates to a procedure that behaves like a recursion operator. **Hint:** for any type T it is possible to allocate a reference of type $Ref(T \to T)$. Use SML to test your terms.

Exercise 9.6 (Contextual equivalence) Consider the simply typed lambda calculus with pairs and references, over base types *Int* and *Bool*. Find separating contexts for the following expressions.

a) $(ref\ 0, ref\ 0)$ and $(\lambda r.(r, r))\ (ref\ 0)$

b) $r := 1; s := 2$ and $s := 2; r := 1$

c) $r := 1; r := !s$ and $r := !s$

d) $let\ r = ref\ 0\ in\ let\ s = ref\ 0\ in\ \lambda x.if\ x = r\ then\ r\ else\ s$ and
$let\ r = ref\ 0\ in\ let\ s = ref\ 0\ in\ \lambda x.if\ x = s\ then\ r\ else\ s$