Assignment 10
Semantics, WS 2007/08

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

Read Section 5.4 in the lecture notes and Chapter 14 of Pierce's TAPL

Exercise 10.1 (Explicitly typed exceptions) Consider the simply typed lambda calculus extended with *error*. One could try to restore the unique typing property by annotating *error* with its intended type, and refining the typing rule (T-ERROR) to $\Gamma \vdash error_T : T$. What goes wrong?

Exercise 10.2 (Semantics of exceptions) Consider the simply typed lambda calculus extended with exceptions that carry values.

a) Make sure that you can state the (proper) reduction rules, evaluation contexts, typing rules, and the revised progress property.

b) State the inference rules that define an equivalent big-step semantics $t \Downarrow r$ for this language. Here r is a *result*, which can either be a value or a term of the form *raise* v. Take care to treat exceptions correctly in the rule for applications.

Exercise 10.3 (Recursion) Consider the simply typed lambda calculus extended with exceptions that carry values.

a) Show that reduction is not normalizing on well-typed terms of this calculus.

b) Let T_1, T_2 be any types. Find a well-typed term that evaluates to a procedure that behaves like a recursion operator $fix : ((T_1 \to T_2) \to T_1 \to T_2) \to T_1 \to T_2$.

Use SML to test your terms. **Note:** the exercise is quite tricky; don't spend too much time on this one!

Exercise 10.4 (Coincidence of small-step and abstract machine semantics) Prove that

$$t \Downarrow v \quad \Longleftrightarrow \quad (t, nil) \mapsto \ldots \mapsto (v, nil)$$

Hints: For '⇒', you need to prove the more general property that if $t \Downarrow v$ then $\forall k\colon\ (t,k) \mapsto \ldots \mapsto (v,k)$. For '⇐', define the 'dismantling' $k \bullet t$ of a stack k onto a term t inductively by $nil \bullet t := t$ and $(F::k) \bullet t := k \bullet F[t]$. Then prove (1) that $t \to t'$ implies $\forall k\colon\ k \bullet t \to k \bullet t'$, and (2) that $\forall k\colon\ (t,k) \mapsto (t',k')$ implies $k \bullet t \to \ldots \to k' \bullet t'$. You can then use the coincidence result between small-step and big-step semantics to infer the proposition.

Exercise 10.5 (Abstract machine for PCF and control)

a) Simulate (using pencil and paper) the abstract machine on the terms
   - $(\lambda xy.y)(\lambda xy.x)$ and
   - $(\lambda xy.y)(\mathit{callcc}(\lambda k.\mathit{throw}(\lambda xy.x, k)))$.

b) We implement terms and frames of the pure λ calculus in SML as follows:

```
type var = int
datatype ter = V of var | A of ter * ter | L of var * ter
and      frm = AppL of ter | AppR of ter   (* [] t and v [], resp *)
```

   Write a procedure $\mathit{run} : \mathit{ter} * \mathit{frm\ list} \to \mathit{ter} * \mathit{frm\ list}$ that implements reduction in the abstract machine for λ calculus. For simplicity you may assume that the machine is only run on closed terms.

c) Extend your implementation from (b) to PCF.

d) Extend your implementation with *callcc*.

Exercise 10.6 (Control operators and Curry-Howard correspondence) Use the lambda calculus with *callcc* to obtain proof terms for the following propositions, where for emphasis we write $\neg T$ for $\mathit{Cont}\ T$:

a) $(\neg T \to T) \to T$

b) $(\neg T \to \neg T') \to T' \to T$

c) $(T \to T') \to \neg T' \to \neg T$

d) $0 \to T$

e) $\neg(\neg T) \to T$