$\boxed{\text{CIS 500 Software Foundations}}$
Homework Assignment 8

Exceptions

Due: Monday, November 15, 2004, by noon

Submission instructions: Same as last time.

1 Exercise Write out the statements of progress and preservation theorems for the system described in Section 14.2– i.e. the simply typed lambda calculus with base type Unit and the rules for exceptions listed in Figures 14.1 and 14.2.

2 Exercise For each new extension to the simply typed lambda calculus with exceptions, we must add new rules to propagate error. For example, in the language STLC + exceptions + booleans we need the rule:

$$\texttt{if error then } \texttt{t}_1 \texttt{ else } \texttt{t}_2 \longrightarrow \texttt{error}$$

in order to show the progress theorem.

What new rules must be added to STLC + exceptions with the following extensions?

1. Sequencing, described by the rules (E-Seq) (E-SeqNext) and (T-Seq).
2. Pairs, as in Figure 11-5
3. Sums, as in Fig 11-10
4. Recursion, as in Fig 11-12

3 Exercise Consider the following extension to the language STLC + booleans + exceptions:

$$\texttt{if error then } \texttt{t}_1 \texttt{ else } \texttt{t}_2 \longrightarrow \texttt{error}$$

$$\texttt{if } (\lambda \texttt{x} : \texttt{T}_1\texttt{.t}) \texttt{ then } \texttt{t}_1 \texttt{ else } \texttt{t}_2 \longrightarrow \texttt{error}$$

$$\texttt{true v} \longrightarrow \texttt{error}$$

$$\texttt{false v} \longrightarrow \texttt{error}$$

Are the progress and preservation theorems still true? Are they meaningful?

4 Exercise Evaluation contexts are an alternative way to present the operational semantics of a programming language. The idea is to separate congruence rules from computation rules. The computation rules, which have the form $t \rightarrow_\beta t'$ are the same as always; for the STLC with booleans and ERROR they are:

$$(\lambda x : T.t)\ v \rightarrow_\beta [x \mapsto v]t \qquad \text{(C-AppAbs)}$$

$$\textsf{if true then } t_2 \textsf{ else } t_3 \rightarrow_\beta t_2 \qquad \text{(C-IfTrue)}$$

$$\textsf{if false then } t_2 \textsf{ else } t_3 \rightarrow_\beta t_3 \qquad \text{(C-IfFalse)}$$

The congruence rules and the rules for propagating errors are folded into just two rules that take all contexts into account:

$$\frac{t \to_\beta t'}{E[t] \to E[t']} \qquad \text{(E-CTX)}$$

$$E[\mathsf{error}] \to \mathsf{error} \qquad \text{(E-ERROR)}$$

where the evaluation contexts E are defined as follows:

$$E ::= [\cdot] \;|\; E\ t \;|\; v\ E \;|\; \mathsf{if}\ E\ \mathsf{then}\ t_2\ \mathsf{else}\ t_3$$

and $E[t]$ is defined inductively on the structure of E:

$$\begin{array}{rcl} [\cdot][t] & = & t \\ (E\ t_1)[t] & = & E[t]\ t_1 \\ (v\ E)[t] & = & v\ E[t] \\ (\mathsf{if}\ E\ \mathsf{then}\ t_2\ \mathsf{else}\ t_3)[t] & = & \mathsf{if}\ E[t]\ \mathsf{then}\ t_2\ \mathsf{else}\ t_3 \end{array}$$

Intuitively, the hole, $[\cdot]$ in an evaluation context marks the point at which the next computation step can happen. To see how this all works, consider the following evaluation step (using the old rules):

$$\dfrac{\dfrac{}{(\lambda y : \mathsf{Unit}.y)\ \mathsf{unit}) \to \mathsf{unit}}\ \text{[E-APPABS]}}{(\lambda x : \mathsf{Unit}.x)\ ((\lambda y : \mathsf{Unit}.y)\ \mathsf{unit}) \to (\lambda x : \mathsf{Unit}.x)\ \mathsf{unit}}\ \text{[E-APP2]}$$

We can take the same step in the system where evaluation is defined using evaluation contexts by picking a context of the form $(v\ E)$. Specifically, we choose $E = (\lambda x : \mathsf{Unit}.x)\ [\cdot]$ and noting that our term is $E[(\lambda y : \mathsf{Unit}.y)\ \mathsf{unit}]$. The evaluation behavior above follows from E-CTX (using computation rule C-APPABS):

$$\dfrac{(\lambda y : \mathsf{Unit}.y)\ \ \to_\beta \mathsf{unit}}{E[(\lambda y : \mathsf{Unit}.y)\ \mathsf{unit}] \to E[\mathsf{unit}]}\ \text{[E-CTX]}$$

(Note that the result, $E[\mathsf{unit}]$, is just $(\lambda x : \mathsf{Unit}.x)\ \mathsf{unit}$).

(a) State and prove the progress theorem for this system. You may use any standard auxiliary lemmas you need (e.g., inversion of typing, canonical forms, substitution, etc.) as long as you give their statements clearly. Any new lemmas must be proved. (Warning: The structure of this proof is different from the proofs we have done so far—still it is a proof by induction).

(b) State and prove the preservation theorem for the same system.

5 Debriefing

1. How many hours (per person) did you spend on this assignment?
2. Would you rate it as easy, moderate, or difficult?
3. Did everyone in your study group participate?
4. How deeply do you feel you understand the material it covers (0%–100%)?

If you have any other comments, we would like to hear them; please send them to `cis500@cis.upenn.edu`.