CS520, Handout 16, 31 October 2006 (updated 1 Nov 2006)

Evaluation Contexts (Continued, Once More)

We define the syntax and operational semantics of a call-by-value language which extends the λ-calculus with "exceptions" and "side-effects". We call this language $\mathcal{L} + (\mathsf{exception}, \mathsf{store})$, which is inspired by the presentation in the paper *Contrasting Exceptions and Continuations*, by Hayo Thielecke.

SYNTAX OF $\mathcal{L} + (\mathsf{exception}, \mathsf{store})$

μ ::=		*stores*
	$\varnothing$	*empty store*
	$\mu, \ell \mapsto v$	*location binding*
t ::=		*terms*
	v	*value*
	$t\ t$	*application*
	$\texttt{succ}\ t$	*successor*
	$\texttt{pred}\ t$	*predecessor*
	$\texttt{if0}\ t\ \texttt{then}\ t\ \texttt{else}\ t$	*conditional*
	$\texttt{ref}\ t$	*reference creation*
	$!\,t$	*dereference*
	$t := t$	*assignment*
	$\texttt{raise}\ t\ t$	*raise exception*
	$\texttt{handle}\ t\ t\ t$	*handle exception*
v ::=		*values*
	x	*variable*
	n	*decimal numeral*
	ℓ	*location*
	ex	*exception name*
	$\lambda x.\, t$	*abstraction*

In contrast to [Pierce] and all preceding handouts, we choose to include variables among "values". This convention is perhaps a little counter-intuitive, but simplifies some of the formulations and is found in many studies of programming languages.

SEMANTICS OF $\mathcal{L} + (\mathsf{exception}, \mathsf{store})$

Evaluation Contexts:

$$\begin{aligned} E ::=\ & [\,] \mid E\ t \mid v\ E \mid \\ & \texttt{succ}\ E \mid \texttt{pred}\ E \mid \texttt{if0}\ E\ \texttt{then}\ t_1\ \texttt{else}\ t_2 \mid \\ & \texttt{ref}\ E \mid !\,E \mid E := t \mid v := E \mid \\ & \texttt{raise}\ E\ t \mid \texttt{raise}\ v\ E \mid \\ & \texttt{handle}\ E\ t\ t \mid \texttt{handle}\ v\ E\ t \mid \texttt{handle}\ v_1\ v_2\ E \end{aligned}$$

Lemma 0.1 (Unique Decomposition Lemma). *Let t be a term of $\mathcal{L} + (\mathsf{exception}, \mathsf{store})$ which is not a value. Then t can be uniquely decomposed as $t = E[t']$ where E is an evaluation context and t' is a term in one of the 9 following forms:*

1. $t' =\ v_1\ v_2$,
2. $t' =\ \texttt{succ}\ v$,
3. $t' =\ \texttt{pred}\ v$,
4. $t' =\ \texttt{if0}\ v\ \texttt{then}\ t_1\ \texttt{else}\ t_2$,
5. $t' =\ \texttt{ref}\ v$,
6. $t' =\ !\,v$,
7. $t' =\ v_1 := v_2$,
8. $t' =\ \texttt{raise}\ v_1\ v_2$,
9. $t' =\ \texttt{handle}\ v_1\ v_2\ v_3$. $\square$

Evaluation Rules: The rules now include a store μ to account for side-effects. We have 8 rules, corresponding to the 9 cases in Lemma 0.1, with the last two cases working in tandem in a single rule:

1. $E[(\lambda x.\, t\,)\; v] \mid \mu \;\rightsquigarrow\; E[\,[x \mapsto v]t\,] \mid \mu$
2. $E[\texttt{succ}\; n] \mid \mu \;\rightsquigarrow\; E[n+1] \mid \mu$
3. $E[\texttt{pred}\; n] \mid \mu \;\rightsquigarrow\; \begin{cases} E[0] \mid \mu & \text{if } n = 0 \\ E[n-1] \mid \mu & \text{if } n > 0 \end{cases}$
4. $E[\texttt{if0}\; n\; \texttt{then}\; t_1\; \texttt{else}\; t_2] \mid \mu \;\rightsquigarrow\; \begin{cases} E[t_1] \mid \mu & \text{if } n = 0 \\ E[t_2] \mid \mu & \text{if } n > 0 \end{cases}$
5. $E[\texttt{ref}\; v] \mid \mu \;\rightsquigarrow\; E[\,\ell\,] \mid (\mu, \ell \mapsto v)$ where ℓ is fresh
6. $E[!\,\ell] \mid \mu \;\rightsquigarrow\; E[\mu(\ell)] \mid \mu$
7. $E[\ell := v] \mid \mu \;\rightsquigarrow\; E[\,v\,] \mid [\ell \mapsto v]\mu$
8. $E[\texttt{handle}\; ex\; v\; t] \mid \mu \;\rightsquigarrow\; \begin{cases} E[\,v'\,] \mid \mu & \text{if } t = v' \\ E[v\; v'] \mid \mu & \text{if } t = E'[\texttt{raise}\; ex\; v'] \\ & \text{and } E' \neq E_1[\texttt{handle}\; ex\; v''\; E_2] \end{cases}$

The uniqueness of E specified by Lemma 0.1 guarantees that at most one of these 8 rules can be used on the same term.

Note how rule 8 works: The evaluation of a `raise` is only possible if it is inside a dynamically enclosing `handle` and, moreover, if `raise` and the enclosing `handle` are for the same exception name ex. The condition that E' is not of the form $E_1[\texttt{handle}\; ex\; v''\; E_2]$ guarantees that E' does not handle an exception with name ex, i.e., t is not of the form:

$$E_1[\texttt{handle}\; ex\; v''\; E_2[\texttt{raise}\; ex\; v']]$$

Put differently, the evaluation of a `raise` is in tandem with the *closest* enclosing `handle` for the same exception name.

Typing Rules for $\mathcal{L} + (\mathsf{exception}, \mathsf{store})$

constants

$$\frac{}{\Gamma \vdash n : \texttt{Int}} \qquad \frac{}{\Gamma \vdash ex : \texttt{Exn}\; T}$$

lambda calculus

$$\frac{(x : T) \in \Gamma}{\Gamma \vdash x : T}$$

$$\frac{\Gamma, x : T \vdash t : T'}{\Gamma \vdash (\lambda x : T.\, t\,) : T \to T'} \qquad \frac{\Gamma \vdash t_1 : T \to T' \qquad \Gamma \vdash t_2 : T}{\Gamma \vdash t_1\; t_2 : T'}$$

integers and conditionals

$$\frac{\Gamma \vdash t : \texttt{Int}}{\Gamma \vdash \texttt{succ}\; t : \texttt{Int}}$$

$$\frac{\Gamma \vdash t : \texttt{Int}}{\Gamma \vdash \texttt{pred}\; t : \texttt{Int}} \qquad \frac{\Gamma \vdash t_0 : \texttt{Int} \qquad \Gamma \vdash t_1 : T \qquad \Gamma \vdash t_2 : T}{\Gamma \vdash \texttt{if0}\; t_0\; \texttt{then}\; t_1\; \texttt{else}\; t_2 : T}$$

exceptions

$$\frac{\Gamma \vdash t_1 : \texttt{Exn}\; T \quad \Gamma \vdash t_2 : T}{\Gamma \vdash \texttt{raise}\; t_1\; t_2 : T} \qquad \frac{\Gamma \vdash t_0 : \texttt{Exn}\; T \quad \Gamma \vdash t_1 : T \to T' \quad \Gamma \vdash t_2 : T'}{\Gamma \vdash \texttt{handle}\; t_0\; t_1\; t_2 : T'}$$

references

$$\frac{\Gamma \vdash t : T}{\Gamma \vdash \texttt{ref}\; t : \texttt{Ref}\; T}$$

$$\frac{\Gamma \vdash t : \texttt{Ref}\; T}{\Gamma \vdash !\,t : T} \qquad \frac{\Gamma \vdash t_1 : \texttt{Ref}\; T \qquad \Gamma \vdash t_2 : T}{\Gamma \vdash (t_1 := t_2) : T}$$