CS520, Handout 15, 24 October 2006 (updated 31 October 2006)

Evaluation Contexts (Continued)

We reproduce the definition of the language $\mathcal{L}$ in Handout 14, which we can do here in a more compact form because we restrict attention to the call-by-value semantics (ignoring call-by-name) and because we choose the approach based on evaluation contexts (ignoring the approaches based on small-step and big-step rules). We can drop the subscript "cbv" as we do not consider the call-by-name semantics of $\mathcal{L}$ in this handout.

<u>SYNTAX OF $\mathcal{L}$</u>

t	$::=$		*terms*
		nf	*normal form*
		$t\ t$	*application*
		$t + t$	*summation*
nf	$::=$		*normal forms*
		x	*variable*
		v	*value*
v	$::=$		*values*
		n	*numeral*
		$\lambda x.\, t$	*abstraction*

Warning: What we call here "normal form" is more restrictive than usual. A term is a normal form if it cannot be evaluated further, such as $x + 3$ or $\lambda x.\, x + 3$.

<u>SEMANTICS OF $\mathcal{L}$</u>

Contexts:

$$C ::= [\,] \quad | \quad C\ t \quad | \quad t\ C \quad | \quad C + t \quad | \quad t + C$$

Evaluation Contexts:

$$E ::= [\,] \quad | \quad E\ t \quad | \quad v\ E \quad | \quad E + t \quad | \quad v + E$$

An "evaluation context" is a special case of a "context": Every evaluation context is a context, but not the other way around. The operation of "plugging a term t in the hole of a context C" is defined as the operation of "plugging a term t in the hole of an evaluation context E" in Handout 14. For convenience, we write $C[t]$ rather $\mathsf{plug}(t, C)$.

Lemma 0.1 (Ordering of Contexts). *Let t be a term of $\mathcal{L}$. Let C_1 and C_2 be distinct contexts, and t_1 and t_2 non-overlapping subterm occurrences in t, such that $t = C_1[t_1]$ and $t = C_2[t_2]$. Then there are unique contexts C, C_1' and C_2', and terms t_1' and t_2', such that:*

$$C_1 = C[C_1' \# t_1'] \quad \textit{and} \quad C_2 = C[t_2' \# C_2']$$

where $\# = @$ (the unwritten symbol for "application") or $\# = +$ ("summation").

Informally, if we view t, C_1 and C_2, as abstract syntax trees, then C_1 and C_2 are initial fragments of t (starting from the root at the top) with the hole in C_1 on the left of the hole in C_2. We can thus say that context C_1 is to the left of context C_2 in t. $\square$

For an arbitrary term t, consider all the contexts $C_1, \ldots, C_n$ such that $t = C_i[\mathit{nf}_i]$ for some nf_i. Because all these normal-form occurrences $\mathit{nf}_1, \ldots, \mathit{nf}_n$ in t are non-overlapping, we can use the preceding lemma to linearly order $C_1, \ldots, C_n$ – and among these, the *leftmost context for term t* is well-defined.

Exercise 0.2. For a term t of your choice, list all contexts C such that $t = C[\mathit{nf}]$ for some normal form nf. Among these contexts, determine which are evaluation contexts. If all such contexts are evaluation contexts, choose another t for which this is not the case. $\square$

Exercise 0.3. Let t be an arbitrary term of $\mathcal{L}$, and consider a context C for t such that $t = C[\mathit{nf}]$ for some nf.

1. If t is closed, show that nf is a value.
2. If t is not closed, show that nf is not necessarily a value. $\square$

Lemma 0.4 (Unique Decomposition Lemma). *Let t be a term of $\mathcal{L}$ which is not a value. There is a unique decomposition of t as $t = E[t']$ where E is an evaluation context and t' is a term in one of three forms:*

1. $t' = v\ v'$,
2. $t' = v + v'$,
3. $t' = x$. $\square$

In Lemma 0.4, if t is closed then $t = E[t']$ where t' is not a variable, i.e., the third case does not occur.

Exercise 0.5. Let t be a term of $\mathcal{L}$. Let $C_1, \ldots, C_n$ be all contexts such that $t = C_i[\mathit{nf}_i]$ for some normal form nf_i, with $1 \leqslant i \leqslant n$. As pointed out after Lemma 0.1, $C_1, \ldots, C_n$ can be linearly ordered – we assume they are listed according to this linear order from left to right. Suppose t has a decomposition $t = E[t']$, as in the hypothesis of Lemma 0.4, where t' is in one of two forms:

1. $t' = (\lambda x.\, t'')\ v$,
2. $t' = n + n'$.

Show there is a k with $1 \leqslant k < n$ such that each of $\mathit{nf}_1, \ldots, \mathit{nf}_{k-1}$ is a value and

1. If $t' = (\lambda x.\, t'')\ v$
   then $C_k = E[[\,]\ v]$ with $\mathit{nf}_k = (\lambda x.\, t'')$ and $C_{k+1} = E[(\lambda x.\, t'')\ [\,]]$ with $\mathit{nf}_{k+1} = v$,

2. If $t' = n + n'$
   then $C_k = E[[\,] + n']$ with $nf_k = n$ and $C_{k+1} = E[n + [\,]]$ with $nf_{k+1} = n'$.

Moreover, show there is a context C and a term $\tilde{t}$ such that $t = C[\tilde{t}]$ where

$$\tilde{t} = \ (nf_1 \#_1 (nf_2 \#_2 (\cdots (nf_{k-1} \#_{k-1} t') \cdots)))$$

with $\#_1, \#_2, \ldots, \#_{k-1} \in \{@, +\}$. $\square$

Evaluation Rules: We give two rules, neither requires premises – which, in certain situations, is an advantage over an approach based on small-step or big-step rules. We have two rules, corresponding to the first two cases in Lemma 0.4:

1. $E[(\lambda x.\, t''\,)\; v] \rightsquigarrow E[\,[x \mapsto v]t''\,]$,
2. $E[n + n'] \rightsquigarrow E[n'']$ where n'' is the sum of n and n'.

<u>TYPING RULES FOR $\mathcal{L}$</u>

Left to the reader. A system of simple types for $\mathcal{L}$ is straightforward, similar to the system in [Pierce, Chapts. 9 and 11].

We define another language, called $\mathcal{L}+\mathsf{store}$, which is obtained from $\mathcal{L}$ by augmenting it with features to manipulate a “store”, “assignable variables” and “side-effects”. We can study $\mathcal{L}+\mathsf{store}$ as an alternative to the study of side-effects in [Pierce, Chapt 13]. Our study of $\mathcal{L}+\mathsf{store}$ is simpler than the material in [Pierce, Chapt 13], in part because we do not set up a type system for $\mathcal{L}+\mathsf{store}$. Setting up a type system for $\mathcal{L}+\mathsf{store}$ is not a simple matter, which requires some thinking and design decisions, and which may produce a presentation almost as complicated as that of [Pierce, Chapt 13].

Syntax of $\mathcal{L}+\mathsf{store}$

$$
\begin{array}{lllll}
\mu & ::= & & \textit{stores} & \\
 & & \varnothing & & \textit{empty store} \\
 & & \mu, x \mapsto v & & \textit{variable binding} \\
t & ::= & & \textit{terms} & \\
 & & nf & & \textit{normal form} \\
 & & t\ t & & \textit{application} \\
 & & t + t & & \textit{summation} \\
 & & \texttt{let}\ x = t\ \texttt{in}\ \ t & & \texttt{let}\text{-}\textit{binding} \\
 & & x\ :=\ t & & \textit{assignment} \\
nf & ::= & & \textit{normal forms} & \\
 & & x & & \textit{variable} \\
 & & v & & \textit{value} \\
v & ::= & & \textit{values} & \\
 & & n & & \textit{numeral} \\
 & & \lambda x.\, t & & \textit{abstraction}
\end{array}
$$

Several remarks:

- A “variable binding” here is not the same as a “location binding” in [Pierce, Fig. 13-1]. We do not introduce the notion of “location” in the formulation of $\mathcal{L}+\mathsf{store}$.
- A “`let`-binding” here is similar to a `let`-binding in [Pierce, Fig. 11-4, p. 124], but its evaluation will be different (see below).
- An “assignment” here is different from an “assignment” in [Pierce, Fig. 13-1, p. 166].

Semantics of $\mathcal{L} + \mathsf{store}$

Contexts:

$$C ::= [\,] \quad|\quad C\ t \quad|\quad t\ C \quad|\quad C + t \quad|\quad t + C \quad|\quad x := C \quad|\quad \texttt{let}\ x = C\ \texttt{in}\ \ t$$

Evaluation Contexts:

$$E ::= [\,] \quad|\quad E\ t \quad|\quad v\ E \quad|\quad E + t \quad|\quad v + E \quad|\quad x := E \quad|\quad \texttt{let}\ x = E\ \texttt{in}\ \ t$$

Exercise 0.6. State the generalization of Lemma 0.1 for terms in $\mathcal{L} + \mathsf{store}$. □

Exercise 0.7. State the generalization of Exercise 0.2 for terms in $\mathcal{L}+\mathsf{store}$, and then solve the exercise. □

Exercise 0.8. State the generalization of Exercise 0.3 for terms in $\mathcal{L}+\mathsf{store}$, and then solve the exercise. □

Lemma 0.9 (Unique Decomposition Lemma). *Let t be a term of $\mathcal{L}+\mathsf{store}$ which is not a value. There is a unique decomposition of t as $t = E[t']$ where E is an evaluation context and t' is a term in one of five forms:*

1. $t' = v\ v'$,
2. $t' = v + v'$,
3. $t' = (\texttt{let}\ x = v\ \texttt{in}\ \ t'')$,
4. $t' = (x\ :=\ v)$,
5. $t' = x$. □

Exercise 0.10. State the generalization of Exercise 0.5 for terms in $\mathcal{L} + \mathsf{store}$, and then solve the exercise. □

Evaluation Rules: The rules now include a store μ to account for side-effects. We have 5 rules, corresponding to the 5 cases in Lemma 0.9:

1. $E[(\lambda x.\, t''\,)\ v] \mid \mu \quad \rightsquigarrow \quad E[\,[x \mapsto v]t''\,] \mid \mu$,
2. $E[n + n'] \mid \mu \quad \rightsquigarrow \quad E[n''] \mid \mu \qquad$ where n'' is the sum of n and n',
3. $E[\texttt{let}\ x = v\ \texttt{in}\ \ t''] \mid \mu \quad \rightsquigarrow \quad E[\,[x \mapsto y]t''\,] \mid (\mu, y \mapsto v) \qquad$ where y is fresh,
4. $E[x\ :=\ v] \mid \mu \quad \rightsquigarrow \quad E[\,v\,] \mid [x \mapsto v]\mu$,

The third rule adds a binding to the store. The fourth rules adds a binding to, or updates a binding already in, the store. We need the fifth rule to retrieve values from the store:

5. $E[\,x\,] \mid \mu \quad \rightsquigarrow \quad E[\,\mu(x)\,] \mid \mu$, $\qquad$ provided $\mu(x)$ is defined.

The uniqueness of E specified by Lemma 0.9 guarantees that at most one of these 5 rules can be used on the same term.