CS520, Handout 14, 19 October 2006 (updated 31 October 2006)

Evaluation Contexts (not in [Pierce])

"Evaluation contexts" are an important concept in defining the (operational) semantics of programming languages. Evaluation contexts are not considered in Pierce's book. One goal of this handout is to introduce evaluation contexts and their use. To illustrate the main ideas, we define a simple foundational language, called $\mathcal{L}$.

We choose the language $\mathcal{L}$ to be the **pure lambda calculus** augmented with **natural numbers** and **summation**:

<u>SYNTAX</u>

t ::=		*terms*
	x	*variable*
	n	*numeral*
	$\lambda x.\, t$	*abstraction*
	$t\ t$	*application*
	$t + t$	*summation*

For the **operational semantics** of $\mathcal{L}$, we consider two different evaluation strategies for $\mathcal{L}$: **call-by-value** and **call-by-name**. We then discuss 3 different approaches for each of the two strategies: **small-step rules**, **big-step rules**, and **evaluation contexts**.

<u>VALUES</u> (for both call-by-value and call-by-name)

v ::=		*values*
	n	*numeral*
	$\lambda x.\, t$	*abstraction*

Call-by-Value

Small-Step Rules

$$(\lambda x.\, t\,)\ v \Rightarrow [x \mapsto v]t \qquad \text{(CbV-small-AppAbs)}$$

$$n + n' \Rightarrow n'' \qquad \text{(CbV-small-SumNum)} \quad n'' = \text{sum of } n \text{ and } n'$$

$$\frac{t_1 \Rightarrow t_2}{t_1\ t \Rightarrow t_2\ t} \qquad \text{(CbV-small-App1)}$$

$$\frac{t_1 \Rightarrow t_2}{v\ t_1 \Rightarrow v\ t_2} \qquad \text{(CbV-small-App2)}$$

$$\frac{t_1 \Rightarrow t_2}{t_1 + t \Rightarrow t_2 + t} \qquad \text{(CbV-small-Sum1)}$$

$$\frac{t_1 \Rightarrow t_2}{v + t_1 \Rightarrow v + t_2} \qquad \text{(CbV-small-Sum2)}$$

Big-Step Rules

$$v \Downarrow v \qquad \text{(CbV-big-Val)}$$

$$\frac{t_1 \Downarrow \lambda x.\, t' \qquad t_2 \Downarrow v \qquad [x \mapsto v]t' \Downarrow v'}{t_1\ t_2 \Downarrow v'} \qquad \text{(CbV-big-App)}$$

$$\frac{t_1 \Downarrow n_1 \qquad t_2 \Downarrow n_2 \qquad n_1 + n_2 = n_3}{t_1 + t_2 \Downarrow n_3} \qquad \text{(CbV-big-Sum)}$$

Lemma 0.1 (Equivalence of Small-Step and Big-Step). *For every term t of $\mathcal{L}$ and every value v, we have $t \Rightarrow^* v$ iff $t \Downarrow v$.* □

Evaluation Contexts

$$E_{\text{cbv}} ::= [\,] \quad | \quad E_{\text{cbv}}\ t \quad | \quad v\ E_{\text{cbv}} \quad | \quad E_{\text{cbv}} + t \quad | \quad v + E_{\text{cbv}}$$

An evaluation context is an expression with exactly one "hole", denoted [ ], at the point where the next evaluation step must take place. Placing a term t in the hole of an evaluation context E_{cbv} is formally defined by:

$$\mathsf{plug}(t, E_{\text{cbv}}) = \begin{cases} t & \text{if } E_{\text{cbv}} = [\,], \\ \mathsf{plug}(t, E'_{\text{cbv}})\ t' & \text{if } E_{\text{cbv}} = E'_{\text{cbv}}\ t', \\ v\ \mathsf{plug}(t, E'_{\text{cbv}}) & \text{if } E_{\text{cbv}} = v\ E'_{\text{cbv}}, \\ \mathsf{plug}(t, E'_{\text{cbv}}) + t' & \text{if } E_{\text{cbv}} = E'_{\text{cbv}} + t', \\ v + \mathsf{plug}(t, E'_{\text{cbv}}) & \text{if } E_{\text{cbv}} = v + E'_{\text{cbv}}, \end{cases}$$

It is standard practice to write $E_{\text{cbv}}[t]$ instead of $\mathsf{plug}(t, E_{\text{cbv}})$.

Lemma 0.2 (Unique Decomposition). *Every closed term t in $\mathcal{L}$ which is not a value can be uniquely decomposed into a (call-by-value) evaluation context E_{cbv} and a term t' such that $t = E_{\text{cbv}}[t']$ where $t' = v\ v'$ or $t' = v + v'$.* □

Definition 0.3 (One-Step Via Evaluation Contexts). Let t be a closed term which is not a value. Corresponding to the two cases in Lemma 0.2, we define two rules, using " $\rightsquigarrow$ " to denote a single-step evaluation:

1. $E_{\text{cbv}}[(\lambda x.\, t\ )\ v] \rightsquigarrow E_{\text{cbv}}[[x \mapsto v]t]$,
2. $E_{\text{cbv}}[n + n'] \rightsquigarrow E_{\text{cbv}}[n'']$ where n'' is the sum of n and n'. □

Lemma 0.4 (Small-Step Evaluation Is Equivalent to One-Step Via Evaluation Contexts). *For all closed terms t_1 and t_2 of $\mathcal{L}$, we have $\ t_1 \Rightarrow t_2\ $ iff $\ t_1 \rightsquigarrow t_2$.* □

Call-by-Name

<u>Small-Step Rules</u>

$$(\lambda x.\, t\,)\; t' \Rightarrow [x \mapsto t']t \qquad \text{(CbN-small-AppAbs)}$$

$$n + n' \Rightarrow n'' \qquad \text{(CbN-small-SumNum)} \quad n'' = \text{sum of } n \text{ and } n'$$

$$\frac{t_1 \Rightarrow t_2}{t_1\; t \Rightarrow t_2\; t} \qquad \text{(CbN-small-App1)}$$

$$\frac{t_1 \Rightarrow t_2}{t_1 + t \Rightarrow t_2 + t} \qquad \text{(CbN-small-Sum1)}$$

$$\frac{t_1 \Rightarrow t_2}{v + t_1 \Rightarrow v + t_2} \qquad \text{(CbN-small-Sum2)}$$

Comparing **call-by-value** small-step rules and **call-by-name** small-step rules, the notion of a β-redex is more restrictive in **call-by-value** than in **call-by-name**, so that:

CbV-small-AppAbs $\neq$ CbN-small-AppAbs

This forces the inclusion of rule CbV-small-App2 in **call-by-value**, which is omitted in **call-by-name**. The remaining rules are the same:

CbV-small-SumNum	=	CbN-small-SumNum
CbV-small-App1	=	CbN-small-App1
CbV-small-Sum1	=	CbN-small-Sum1
CbV-small-Sum2	=	CbN-small-Sum2

<u>Big-Step Rules</u>

$$v \Downarrow v \qquad \text{(CbN-big-Val)}$$

$$\frac{t_1 \Downarrow \lambda x.\, t' \qquad [x \mapsto t_2]t' \Downarrow v}{t_1\; t_2 \Downarrow v} \qquad \text{(CbN-big-App)}$$

$$\frac{t_1 \Downarrow n_1 \qquad t_2 \Downarrow n_2 \qquad n_1 + n_2 = n_3}{t_1 + t_2 \Downarrow n_3} \qquad \text{(CbN-big-Sum)}$$

Comparing **call-by-value** big-step rules and **call-by-name** big-step rules:

CbV-big-App $\neq$ CbN-big-App

but the other rules are the same:

CbV-big-Val	=	CbN-big-Val
CbV-big-Sum	=	CbN-big-Sum

Lemma 0.1 holds for **call-by-name**, where now we use the rules for **call-by-name** instead of those for **call-by-value**.

Evaluation Contexts

$$E_{\text{cbn}} ::= [\,] \quad | \quad E_{\text{cbn}}\ t \quad | \quad E_{\text{cbn}} + t \quad | \quad v + E_{\text{cbn}}$$

The definition of a call-by-name evaluation context E_{cbn} is the same as that of a call-by-value evaluation context E_{cbv}, except that there is no case here corresponding to $(v\ E_{\text{cbv}})$. The definition of $\mathsf{plug}(t, E_{\text{cbn}})$ here follows the definition of $\mathsf{plug}(t, E_{\text{cbv}})$ for call-by-value.

Lemma 0.5 (Unique Decomposition). *Every closed term t in $\mathcal{L}$ which is not a value can be uniquely decomposed into a (call-by-name) evaluation context E_{cbn} and a term t' such that $t = E_{\text{cbn}}[t']$ where $t' = v\ t$ or $t' = v_1 + v_2$.* □

Definition 0.6 (One-Step Via Evaluation Contexts). Let t be a closed term which is not a value. Corresponding to the two cases in Lemma 0.5, we define two rules, using " $\rightsquigarrow$ " to denote a single-step evaluation:

1. $E_{\text{cbn}}[(\lambda x.\, t\,)\ t'] \rightsquigarrow E_{\text{cbn}}[[x \mapsto t']t]$,

2. $E_{\text{cbn}}[n + n'] \rightsquigarrow E_{\text{cbn}}[n'']$ where n'' is the sum of n and n'. □

Lemma 0.7 (Small-Step Evaluation Is Equivalent to One-Step Via Evaluation Contexts). *For all closed terms t_1 and t_2 of $\mathcal{L}$, we have $t_1 \Rightarrow t_2$ iff $t_1 \rightsquigarrow t_2$.* □