Scribe: Andrew Chung and Alexa Sharp Lecturer: Andrew Myers

1 Evaluation Contexts

In call-by-name (CBN) lambda calculus we had the following evaluation rules:

$$\frac{}{(\lambda x\ e_0)\ e_1 \mapsto e_0\{e_1/x\}} \quad \frac{e_0 \mapsto e_0'}{(e_0\ e_1) \mapsto (e_0'\ e_1)}$$

We can restate the operational semantics in a different way using evaluation contexts. A *context* is a program with a hole in it. An *evaluation context* is a context where the hole is a place we can plug in a reducible expression, or *redex*. We write $\mathcal{C}\hat{}$ to mean an evaluation context and $\mathcal{C}[e]$ to denote the context applied to an expression e (which fills the hole in $\mathcal{C}$). We redefine the call-by-name and call-by-value evaluation rules in terms of these contexts:

For call-by-name:

$$\mathcal{C} \quad ::= \quad [\ \cdot\ ] \ \mid\ \mathcal{C}\ e_1$$

$$\mathcal{C}[(\lambda\ x\ e_0)\ e_1] \mapsto \mathcal{C}[e_0\{e_1/x\}]$$

For call-by-value:

$$\mathcal{C} \quad ::= \quad [\ \cdot\ ] \ \mid\ \mathcal{C}\ e_1 \ \mid\ v\ \mathcal{C}$$

$$\mathcal{C}[(\lambda x\ e)\ v] \mapsto \mathcal{C}[e\{v/x\}]$$

where $v = \lambda\ x\ e$

2 A Translation Function

Another way to understand a language (say, CBN λ-calculus) is to construct a *definitional interpreter*, an interpreter for the CBN λ-calculus written in another, better understood language. A related approach is to define a *definitional translation*, which translates CBN λ-calculus into another language.

The call-by-value calculus is a good language to translate other languages to, because it is simple and corresponds fairly well to what is easy to compile. To show this, we will demonstrate how to translate CBN λ-calculus into CBV λ-calculus by producing a desugaring function $\mathcal{D}[\![e]\!] = e'$, where e is a CBN term and e' is a CBV term that simulates e in some sense.

The key problem in translating the lazy CBN calculus to the eager CBN calculus is that we need a way to "pickle" a (possibly divergent) expression e so that we do not evaluate it right away (eagerly). This is necessary in our CBN to CBV translation in order to simulate lazy evaluation. We pickle an expression e by wrapping it into a lambda-term:

$$e \rightarrow (\lambda\ z\ e) \quad z \notin FV[\![e]\!]$$

The right-hand term is called a *thunk* (The past participle of "think"). The argument z is a dummy variable, so it doesn't matter what we apply this function to. We'll apply it to the simplest closed term, $I \equiv \lambda x\ x$.

Now let us inductively define $\mathcal{D}$, our translation function:

$$\mathcal{D}[\![x]\!] = (x\ I)$$

$$\mathcal{D}[\![e_0\ e_1]\!] = \mathcal{D}[\![e_0]\!](\lambda\ z\ \mathcal{D}[\![e_1]\!])$$
$$\mathcal{D}[\![\lambda\ x\ e]\!] = (\lambda\ x\ \mathcal{D}[\![e]\!])$$

We will check this definition by trying it on the test case *FALSE* Ω. Recall that *FALSE* $\equiv \lambda x \lambda y\ y$, $\Omega \equiv (\lambda x\ (x\ x))(\lambda x\ (x\ x))$. We would (ideally) like to have *FALSE* $\Omega \mapsto^* I$, as this is how call-by-name would evaluate the expression. Note that this same expression diverges in the call-by-value calculus. We begin with

$$\mathcal{D}[\![(\lambda\ x\ (\lambda\ y\ y))\ ((\lambda\ x\ (x\ x))\ (\lambda\ x\ (x\ x)))]\!]$$
$$= \mathcal{D}[\![(\lambda\ x\ (\lambda\ y\ y))]\!]\ (\lambda\ z\ \mathcal{D}[\![\Omega]\!])$$
$$= (\lambda\ x(\lambda\ y\ (y\ I)))\ (\lambda\ z\ \mathcal{D}[\![\Omega]\!])$$
$$= (\lambda\ y\ (y\ I))$$
$$= \mathcal{D}[\![\lambda\ y\ y]\!]$$

This evaluation satisfies the following *commutation diagram*, which seems like a reasonable way to capture the soundness of the translation:

$$\begin{array}{ccc} e & \overset{CBN*}{\longmapsto} & v \\ \mathcal{D}\downarrow & & \mathcal{D}\downarrow \\ \mathcal{D}[\![e]\!] & \overset{CBV*}{\longmapsto} & \mathcal{D}[\![v]\!] \end{array}$$

But we have a problem with this notion of soundness, which is illustrated in the following example. Consider

$$(\lambda y\ (\lambda x\ y))\ (\lambda w\ w) \mapsto \lambda x\ (\lambda w\ w)$$

We'd like to have $\mathcal{D}[\![\lambda x\ (\lambda x\ y)\ (\lambda w\ w)]\!] \mapsto^* \mathcal{D}[\![\lambda x \lambda w\ w]\!] = \lambda x \lambda w\ (w\ I)$. However, our definition above results in the following evaluation:

$$\mathcal{D}[\![(\lambda\ y\ (\lambda\ x\ y))\ (\lambda\ w\ w)]\!] = \lambda\ y\ (\lambda\ x\ (y\ I))\ (\lambda\ z\ (\lambda\ w\ (w\ I))) \mapsto^* (\lambda\ x\ (\lambda\ z\ (\lambda\ w\ (w\ I)))\ I) \neq \lambda x \lambda w\ (w\ I)$$

What has happened is that the two right-hand-sides are not exact translations; instead we have an extra λ-term and application introduced by the $\mathcal{D}$ function. Thus our commutation diagram needs to look more like this:

$$\begin{array}{ccc} e & \overset{CBN}{\longmapsto} & e' \\ \downarrow & & \downarrow \\ \mathcal{D}[\![e]\!] & \overset{CBV*}{\longmapsto} & \mathcal{D}[\![e']\!] \approx\ e'' \end{array}$$

But the two terms are really equivalent in that if we were allowed to make a β reduction in the middle of the term, then the two will be exactly the same. What we would like to say is that if $e \overset{CBN}{\longmapsto} e' \Rightarrow \exists\ e''\ .\ \mathcal{D}[\![e]\!] \overset{CBV*}{\longmapsto} e''\ \wedge\ e'' \approx \mathcal{D}[\![e']\!]$

3 Equivalence Axioms and Rules

We define an equational proof system that allows us to show that two expressions are equivalent. Note that we might consider expressions to be equivalent extensionally even though the proof system cannot show it; that's okay because we only want to be able to show that certain expressions produced by $\mathcal{D}$ are equivalent.

$$\frac{}{e \approx e}\ (reflexive) \quad \frac{e' \approx e}{e \approx e'}\ (symmetric) \quad \frac{e_1 \approx e_2 \quad e_2 \approx e_3}{e_1 \approx e_3}\ (transitive)$$

$$\frac{e_0 \approx e_0' \quad e_1 \approx e_1'}{e_0\ e_1 \approx e_0'\ e_1'} \qquad \frac{e \approx e'}{\lambda\ x\ e \approx \lambda\ x\ e'} \qquad \frac{}{(\lambda\ x\ e_0)\ e_1 \approx e_0\{e_1/x\}}$$

These top three rules hold for any equational proof system: they capture the idea that $\approx$ is an equivalence relation. The next two rules tell us that substituting equivalent terms results in equivalent terms; a relation with this property is called a *congruence*. The final rule says that β-reduction preserves equivalence. For this translation, we could in place of the β rule define a more restrictive equivalence rule that captures precisely the way in which terms can fails to be syntactically equal:

$$\frac{z \notin FV(e_0)}{(\lambda z\ e_0)I \approx e_0}$$

With these axioms and rules, we can attempt to prove the *soundess property*, i.e. $e \overset{CBN}{\longmapsto} e' \Rightarrow \exists\ e''\ .\ \ \mathcal{D}[\![e]\!] \overset{CBV*}{\longmapsto} e'' \ \wedge\ e'' \approx \mathcal{D}[\![e']\!]$. Equivalently, if we can show that $e \overset{CBV}{\longmapsto} e' \Rightarrow \mathcal{D}[\![e]\!] \approx \mathcal{D}[\![e']\!]$, then we can prove the soundness property by induction on the number of evaluation steps.

Consider all the steps $e \mapsto e'$ you could take with CBN semantics:

$$\mathcal{C} \ \ ::= \ \ [\ \cdot\ ] \ \mid\ \mathcal{C}\ e_1$$

$$\mathcal{C}[(\lambda\ x\ e_0)\ e_1] \mapsto \mathcal{C}[e_0\{e1/x\}]$$

We will prove $e \overset{CBN}{\longmapsto} e' \Rightarrow \mathcal{D}[\![e]\!] \approx \mathcal{D}[\![e']\!]$ by induction on the structure of the context $\mathcal{C}_N$. Consider $\mathcal{C}_N[(\lambda\ x\ e_0)\ e_1] \mapsto \mathcal{C}_N[e_0\{e_1/x\}]$, where $\mathcal{C}_N = [\ \cdot\ ]$ or $\mathcal{C}_N = \mathcal{C}'_N\ e$, where $\mathcal{C}'_N$ is a smaller context than $\mathcal{C}_N$ (ie, induction hypothesis).

If $\mathcal{C}_N = [\ \cdot\ ]$, then we would like to show that $\mathcal{D}[\![(\lambda\ x\ e_0)\ e_1]\!] \approx \mathcal{D}[\![e_0\{e_1/x\}]\!]$. Continuing the evaluation, we have

$$\Rightarrow \mathcal{D}[\![(\lambda\ x\ e_0)]\!]\ (\lambda\ z\ \mathcal{D}[\![e_1]\!]) \approx \mathcal{D}[\![e_0\{e_1/x\}]\!]$$

$$\Rightarrow (\lambda\ x\ \mathcal{D}[\![e_0]\!])\ (\lambda\ z\ \mathcal{D}\ [\![e_1]\!]) \approx \mathcal{D}[\![e_0\{e_1/x\}]\!]$$

$$\Rightarrow \mathcal{D}[\![e_0]\!]\{(\lambda\ z\ \mathcal{D}[\![e_1]\!])/x\} \approx \mathcal{D}[\![e_0\{e_1/x\}]\!]$$

This is an example of a Substitution Lemma. The proof (including the $\mathcal{C}_N = \mathcal{C}'_N\ e$) will be continued next class.