Lecturer: Andrew Myers

1 Evaluation Contexts

The rules for structural operational semantics can be classified into two types:

- *reduction rules*, which describe the actual computation steps; and
- *structural congruence rules*, which constrain the choice of reductions that can be performed next, thus defining both the order of evaluation and whether subexpressions are evaluated lazily.

For example, the CBV reduction strategy for the λ-calculus was captured in the following rules:

$$\frac{}{(\lambda x.\, e)\ v \longrightarrow e\{v/x\}} \tag{1}$$

$$\frac{e_1 \longrightarrow e_1'}{e_1\ e_2 \longrightarrow e_1'\ e_2} \qquad \frac{e_2 \longrightarrow e_2'}{v\ e_2 \longrightarrow v\ e_2'} \tag{2}$$

Rule (1), β-reduction, is a reduction rule, whereas rules (2) are structural congruence rules. The rules (2) say essentially that a reduction may be applied to a redex on the left-hand side of an application anytime, and may be applied to a redex on the right-hand side of an application provided the left-hand side is already fully reduced.

Although there are only two structural congruence rules in the CBV λ-calculus, there are typically many more in real-world programming languages. It would be nice to have a more compact way to express them.

Evaluation contexts provide a mechanism to do just that. An evaluation context E, sometimes written $E[\,\bullet\,]$, is a λ-term or a metaexpression representing a family of λ-terms with a special variable $[\,\cdot\,]$ called the *hole*. If $E[\,\bullet\,]$ is an evaluation context, then $E[\,e\,]$ represents E with the term e substituted for the hole.

Every evaluation context $E[\,\bullet\,]$ represents a *context rule*

$$\frac{e \longrightarrow e'}{E[\,e\,] \longrightarrow E[\,e'\,]},$$

which says that we may apply the reduction $e \longrightarrow e'$ in the context $E[\,e\,]$.

For the case of the CBV λ-calculus, the two structural congruence rules (2) are specified by the two evaluation context schemes ($[\,\cdot\,]\ e$) and ($v\ [\,\cdot\,]$). These are just a compact way of representing the rules (2). Thus we could specify the CBV λ-calculus simply:

$$(\lambda x.\, e)\ v \longrightarrow e\{v/x\} \qquad [\,\cdot\,]\ e \qquad v\ [\,\cdot\,]$$

The CBN λ-calculus has an equally compact specification:

$$(\lambda x.\, e)\ e' \longrightarrow e\{e'/x\} \qquad [\,\cdot\,]\ e$$

2 Nested Contexts

Note that in CBV, the evaluation contexts $[\,\cdot\,]\ e$ and $v\ [\,\cdot\,]$ do not specify *all* contexts in which the reduction rule (1) may be applied. There are also compound contexts obtained from nested applications of the rules (2). For example, the context

$$(v\ [\,\cdot\,])\ e \tag{3}$$

is also a valid evaluation context for CBV, since it can be derived from two applications of the rules (2):

$$\cfrac{\cfrac{e_1 \longrightarrow e_2}{v\ e_1 \longrightarrow v\ e_2}}{(v\ e_1)\ e \longrightarrow (v\ e_2)\ e}. \tag{4}$$

Here we have applied the right-hand rule of (2) in the first step and the left-hand rule of (2) in the second. The evaluation context (3) represents the abbreviated rule

$$\frac{e_1 \longrightarrow e_2}{(v\ e_1)\ e \longrightarrow (v\ e_2)\ e}$$

obtained by collapsing the two steps of (4).

The set of *all* valid evaluation contexts for the CBV λ-calculus is represented by the grammar

$$E \quad ::= \quad [\,\cdot\,] \ \mid \ E\ e \ \mid \ v\ E.$$

3 Annotated Proof Trees

We can also use evaluation contexts to indicate exactly where a reduction is applied in each step of a proof tree. For example, consider the annotated proof tree

$$\cfrac{\cfrac{(\lambda x.\,x)\ 0 \longrightarrow 0}{(\lambda x.\,x)\ ((\lambda x.\,x)\ 0) \longrightarrow (\lambda x.\,x)\ 0}\ ((\lambda x.\,x)\ [\,\cdot\,])}{(\lambda x.\,x)\ ((\lambda x.\,x)\ 0)\ \lambda z.\,z\ z \longrightarrow (\lambda x.\,x)\ 0\ \lambda z.\,z\ z}\ ([\,\cdot\,]\ \lambda z.\,z\ z)$$

We have labeled each step to indicate the context in which the β-reduction was applied.

As above, we can simplify the tree by collapsing the two steps and annotating the resulting abbreviated tree with the corresponding nested context:

$$\frac{(\lambda x.\,x)\ 0 \longrightarrow 0}{(\lambda x.\,x)\ ((\lambda x.\,x)\ 0)\ \lambda z.\,z\ z \longrightarrow (\lambda x.\,x)\ 0\ \lambda z.\,z\ z}\ ((\lambda x.\,x)\ [\,\cdot\,]\ \lambda z.\,z\ z)$$

4 Error Propagation

Evaluation contexts can be used to define the semantics of error exceptions. If we have a special error value error, we can very easily propagate it using the rule

$$E\,[\,\mathsf{error}\,] \rightarrow \mathsf{error}.$$

This obviates the need to show in painstaking detail how error propagates up through a series of applications of rewrite rules. We will revisit this idea later on when we talk about exception handling mechanisms.

The benefits of evaluation contexts will become more apparent as we add more features to the language.