Gnarly

Spencer Tipping

August 14, 2010

Contents

Part I

Mathematical Background

Chapter 1

β Calculus

Evaluation is transparent in the λ calculus, since there are no side effects and the λ constructor is not itself a first-class value. In that case, a β-rewrite is simply a rewrite, without any evaluation or unification (since that could all be done later on with the same results). This implies a degree of freedom in the λ calculus that could in fact be removed, and that is exactly what the β calculus does.

1.1 Primitives

The β calculus is defined recursively as:

$$\exists(E') : E[E' \oplus x] = E[x] \tag{1.1}$$

$$\exists(\circ') : E[\circ' \oplus x] = \beta e_1.\beta e_2.E[E[x] \oplus E[e_1] \oplus E[e_2]] \tag{1.2}$$

$$\exists(\beta') : E[\beta' \oplus x] = \beta e.\beta x.e \tag{1.3}$$

$$\exists(\oplus') : E[\oplus' \oplus x] = \beta e.(x \oplus e) \tag{1.4}$$

$$E[\beta x.E[y] \oplus z] = E[E[\beta x.y \oplus z]] \tag{1.5}$$

$$E[\beta x.y \oplus z] = \begin{cases} z & x = y \\ y & x \neq y \end{cases} \tag{1.6}$$

$$E[\beta x.(y_1 \oplus y_2) \oplus z] = E[E[\beta x.y_1] \oplus E[\beta x.y_2]] \tag{1.7}$$

$$E[\beta x.(\beta y_1.y_2) \oplus z] = \beta y_1.E[\beta x.y_2 \oplus z] \tag{1.8}$$

Statements 1.1, 1.2, 1.3, and 1.4 stipulate that there must exist first-class values E', $\circ'$, β', and $\oplus'$ that construct the primitives in the language.[1] Equation 1.6 states that single-variable substitution happens as it would within the λ calculus, and equations 1.7 and 1.8 define the properties required to make lexical scoping possible.

[1] $\circ'$ is necessary as a combinator to force evaluation.

1.2 Evaluative significance

The λ calculus does not have first-class primitive constructors, so evaluation order is much less important than in the β calculus. Consider, for instance, the expression $(\lambda x.\lambda y.x)z$ for some z. The substitution $[z/x]$ can be performed immediately to yield $\lambda y.z$, which is the only reasonable interpretation.

This is true by equation 1.8 as well, but the nuance arises when the right-hand side of a β expression is unevaluated. These two β expressions have different expansions because of the placement of evaluation:

$$E[\beta x.\beta y.x \oplus z] = \beta y.z \tag{1.9}$$

$$E[\beta x.E[\beta' \oplus y \oplus x] \oplus z] = \begin{cases} \beta y.z & z \neq y \\ \beta y.y & z = y \end{cases} \tag{1.10}$$

The derivation of equation 1.10 for $z = y$ is:

$$\begin{aligned}
E[\beta x.E[\beta' \oplus y \oplus x] \oplus y] &= E[E[\beta x.(\beta' \oplus y \oplus x) \oplus y]] && \text{by 1.5} \\
&= E[E[E[\beta x.\beta' \oplus y] \oplus E[\beta x.y \oplus y] \oplus E[\beta x.x \oplus y]]] && \text{by 1.7} \\
&= E[E[\beta' \oplus y \oplus y]] && \text{by 1.6} \\
&= E[\beta e.\beta y.e \oplus y] && \text{by 1.3} \\
&= \beta y.E[\beta e.e \oplus y] && \text{by 1.8} \\
&= \beta y.y && \text{by 1.6}
\end{aligned}$$

1.3 Encoding of λ calculus

A given λ expression can be encoded in the β calculus by a function T as follows:

$$\begin{aligned}
T[\lambda x.y] &= \beta x.E[T[y]] \\
T[xy] &= E[T[x] \oplus T[y]] \\
T[x] &= x
\end{aligned}$$

(Proof later)

Note that while T^{-1} does exist, it is nontrivial to implement given the presence of β', $\oplus'$, E', and $\circ'$.

Chapter 2

Evaluation Ordering

The significance of evaluation ordering enables different forms of expression to take place. In particular, either lexical or dynamic scoping semantics are possible depending on whether forms are evaluated inside or outside β closures (as demonstrated in section 1.2). Also, because evaluation is selective, some forms can be left unevaluated. (Note, however, that using delayed evaluation to achieve dynamic scoping is probably detrimental to performance in most implementations.)