

# Gnarly

Spencer Tipping

August 13, 2010

# Contents

<b>I</b>	<b>Mathematical Background</b>	<b>2</b>
<b>1</b>	<b><math>\beta</math> Calculus</b>	<b>3</b>
1.1	Primitives . . . . .	3
1.2	Evaluative significance . . . . .	4

**Part I**

**Mathematical Background**

# Chapter 1

## $\beta$ Calculus

Evaluation is transparent in the  $\lambda$  calculus, since there are no side effects and the  $\lambda$  constructor is not itself a first-class value. In that case, a  $\beta$ -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  $\lambda$  calculus that could in fact be removed, and that is exactly what the  $\beta$  calculus does.

### 1.1 Primitives

The  $\beta$  calculus is defined recursively as:

$$\exists(E') : E[E' \oplus x] = E[x] \quad (1.1)$$

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

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

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

$$E[\beta x. E[y] \oplus z] = E[E[\beta x. y \oplus z]] \quad (1.5)$$

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

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

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

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

---

<sup>1</sup> $\circ'$  is necessary as a combinator to force evaluation.

## 1.2 Evaluative significance

The  $\lambda$  calculus does not have first-class primitive constructors, so evaluation order is much less important than in the  $\beta$  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  $\beta$  expression is unevaluated. These two  $\beta$  expressions have different expansions because of the placement of evaluation:

$$E[\beta x. \beta y. x \oplus z] = \beta y. z \quad (1.9)$$

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

The derivation of [equation 1.10](#) for  $z = y$  is:

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