# 1   The Lambda Calculus (Continued)

## 1.1   Normal Forms

A **redux** is a $\alpha$-term of the form

```
(\x -> E1) E2
```

An $\alpha$-term is in **normal form** if it contains no reduxes. In normal form, you cannot apply more $\beta$-steps.

## 1.2   Semantics: Evaluation

A $\lambda$-term `E` evalutes to `E'` if

1.  There is a sequence of steps

    ```
    E =?> E1 =?> ... =?> EN =?> E'
    ```

    where each `=?>` is either `=a>` or `=b>` and $N \geq 0$.

2.  `E'` is in normal form.

As an example, consider the following evaluation:

```
eval test :
    (\x -> x x) (\x -> x)
    =a> (\x -> x x) (\z -> z)
    =b> (\z -> z) (\z -> z)
    =b> (\z -> z)
```

## 1.3   Non-Terminating Evaluation

Consider the following program:

```
    (\x -> x x) (\x -> x x)
    =a> (\x -> x x) (\y -> y y)
    =b> (\y -> y y) (\y -> y y)
    =a> (\x -> x x) (\x -> x x)
```

This program can actually loop back to itself and thus reduce to a normal form. This combinator is called $\Omega$.