**LAMBDA CALCULUS IN BOUNDARY NOTATION**

William Bricken
April 1992

In conventional notation, ABSTRACT is specified by Lx.E, a preceding lambda. The boundary notation makes the lambda a labeled container, x< E >.

```
  Lx.E  = x< E >    [ subst after-lambda-boundary for x in contained-by-lambda ]
```

In conventional notation, APPLY is left-associative, notated by precedence parentheses. When precedence is not ambiguous, parentheses are omitted, notating APPLY by sequential juxtaposition. Boundary notation is the same as left-associative parentheses notation, making application ordering explicit.

```
  E1 E2 = (E1)E2    APPLY E1 to E2, by substituting for the containing lambda.
```

**Substitution Rules**

```
B0.  (x< E1 >)E2        ==>  [subst E2 x E1]

B1.  (x< x >)E          ==>  E

B2.  (x< y >)E          ==>  y                        x not in y

B3.  (x< x< E1 >>)E2    ==>  x< E1 >

B4.  (x1< x2< E1 >>)E2  ==>  x2< (x1< E1 >) E2 >

                        ==>  x2< [subst E2 x1 E1] >

B5.  (x< (E1)E2 >)E3    ==>  ((x< E1 >)E3) (x< E2 >)E3

                        ==>  [subst E3 x (E1)E2]

                        ==>  ([subst E3 x E1]) [subst E3 x E2]
```

**Combinators**

```
(I)E           ==>  E

(((C)E1)E2)E   ==>  ((E1)(E2)) E

((T)E1)E2      ==>  E1

((F)E1)E2      ==>  E2

(((S)E1)E2)E3  ==>  (((E1) E3) (E2)) E3
```