Church's Type Theory

Roger Bishop Jones

ICL Defence Systems

1. THE LAMBDA CALCULUS

1.1. Abstract Syntax

```
type name = string;
infix ! ∈ --;

datatype term =   V of name          |     (* variable        *)
                  op ! of term*term  |     (* application     *)
                  λ of name*term;    (* abstraction     *)
```

1.2. Substitution

```
fun   x ∈ []
      =     false                    |
      x ∈ (y::z)
      =     x = y orelse x ∈ z;
```

```
fun   [] -- a
      =     []                              |
      (h::t) -- a
      =      if h=a then t--a else h::(t--a);

fun   freevars (V n)
      =     [n]                             |
      freevars (a ! b)
      =     (freevars a) @ (freevars b)     |
      freevars (λ (n,t))
      =     (freevars t) -- n;

fun   primed v varl =
      if    v ∈ varl
      then  primed (v^"'") varl
      else  v;

fun   subst t1 n (V n2)
      =     if n=n2 then t1 else (V n2)             |
      subst t1 n (t2!t3)
      =     (subst t1 n t2)!(subst t1 n t3)   |
      subst t1 n (λ(n2,t2))
      =     if      n=n2
            then    (λ(n2,t2))
            else    let     val nv = primed n2 (freevars t1 @
                                    (freevars t2 -- n2))
                            val nf = subst t1 n (subst (V nv) n2 t2)
                    in      λ(nv,nf)
                    end;
```