Computation and Deduction

Lecture 9: Representing Type Preservation
February 11, 1997

1. Zero
2. Successor
3. Let Value
4. Recursion

Type Preservation, Zero

```
tps : eval E V -> of E T -> of V T -> type.

ev_z : eval z z.
tp_z : of z nat.

tps_z : tps (ev_z) (tp_z) (tp_z).
% Alternative proof:
tps_z' : tps (ev_z) P P.
```

Type Preservation, Successor

```
tps : eval E V -> of E T -> of V T -> type.

% Natural Numbers, Successor
tp_s : of (s E) nat <- of E nat.
ev_s : eval (s E) (s V) <- eval E V.

tps_s : tps (ev_s D1) (tp_s P1) (tp_s Q1)
          <- tps D1 P1 Q1.

%{ From Elf:
tps_s :
   {E:exp} {E1:exp} {D1:eval E E1} {P1:of E nat}
   {Q1:of E1 nat}
      tps E E1 nat D1 P1 Q1
         -> tps (s E) (s E1) nat (ev_s E E1 D1)
                (tp_s E P1) (tp_s E1 Q1).
}%
```

Type Preservation, Let Value

```
ev_letv : eval (letv E1 E2) V
            <- eval E1 V1
            <- eval (E2 V1) V.

tp_letv : of (letv E1 E2) T2
           <- of E1 T1
           <- ({x:exp} of x T1 -> of (E2 x) T2).

tps_letv : tps (ev_letv D2 D1) (tp_letv P2 P1) Q2
            <- tps D1 P1 Q1
            <- tps D2 (P2 V1 Q1) Q2.
```

Type Preservation, Recursion

```
ev_fix  : eval (fix E) V
            <- eval (E (fix E)) V.

tp_fix : of (fix E) T
          <- ({x:exp} of x T -> of (E x) T).

tps_fix : tps (ev_fix D1) (tp_fix P1) Q1
            <- tps D1 (P1 (fix E1) (tp_fix P1)) Q1.
```