Computation and Deduction

Lecture 10: Evaluation with Environments
February 13, 1997

Expressions in de Bruijn Form

```
% Expressions
exp'    : type.  %name exp' F

1       : exp'.
^       : exp' -> exp'.  %postfix 20 ^
lam'    : exp' -> exp'.
app'    : exp' -> exp' -> exp'.

% Environments and values
env     : type.  %name env K
val     : type.  %name val W

empty   : env.
;       : env -> val -> env.   %infix left 10 ;

clo     : env -> exp' -> val.
```

Evaluation with Environments

```
feval : env -> exp' -> val -> type.  %name feval D

% Variables
fev_1 : feval (K ; W) 1 W.
fev_^ : feval (K ; W') (F ^) W
          <- feval K F W.

% Functions
fev_lam : feval K (lam' F) (clo K (lam' F)).
fev_app : feval K (app' F1 F2) W
          <- feval K F1 (clo K' (lam' F1'))
          <- feval K F2 W2
          <- feval (K' ; W2) F1' W.
```

Translation to de Bruijn Form

```
trans  : env -> exp' -> exp -> type.  %name trans C
vtrans : val -> exp -> type.          %name vtrans U

% Functions
tr_lam : trans K (lam' F) (lam E)
           <- {w:val} {x:exp}
                 vtrans w x -> trans (K ; w) F (E x).
tr_app : trans K (app' F1 F2) (app E1 E2)
           <- trans K F1 E1
           <- trans K F2 E2.

% Variables
tr_1  : trans (K ; W) 1 E <- vtrans W E.
tr_^  : trans (K ; W) (F ^) E <- trans K F E.

% Values
vtr_lam : vtrans (clo K (lam' F)) (lam E)
           <- trans K (lam' F) (lam E).
```

Mapping between Evaluations

```
map_eval : eval E V -> trans K F E
             -> feval K F W -> vtrans W V -> type.

mp_1 : map_eval (ev_lam) (tr_1 (vtr_lam (tr_lam C2)))
                (fev_1) (vtr_lam (tr_lam C2)).

mp_^ : map_eval D (tr_^ C1) (fev_^ D1') U1
          <- map_eval D C1 D1' U1.

mp_lam : map_eval (ev_lam) (tr_lam C1)
                  (fev_lam) (vtr_lam (tr_lam C1)).

mp_app : map_eval (ev_app D3 D2 D1) (tr_app C2 C1)
          (fev_app D3' D2' D1') U3
          <- map_eval D1 C1 D1' (vtr_lam (tr_lam C3))
          <- map_eval D2 C2 D2' U2
          <- map_eval D3 (C3 W2 V2 U2) D3' U3.
```