Computation and Deduction

Lecture 14: Continuation-Passing Style
February 27, 1997

1. Expressions in CPS
2. Translating to CPS
3. Evaluating CPS

Expressions in CPS

```
cexp : type.
cval : type.

z+ : cval.
s+ : cval -> cval.
case+ : cval -> cexp -> (cval -> cexp) -> cexp.

pair+ : cval -> cval -> cval.
fst+ : cval -> (cval -> cexp) -> cexp.
snd+ : cval -> (cval -> cexp) -> cexp.

lam+ : (cval -> (cval -> cexp) -> cexp) -> cval.
app+ : cval -> cval -> (cval -> cexp) -> cexp.

% no letv+, omitting letn+, fix+ for now
vl+ : cval -> cexp.
```

Compilation to CPS I

```
cmp : exp -> (cval -> cexp) -> cexp -> type.

cmp_z : cmp z K (K z+).
cmp_s : cmp (s E) K E'
         <- cmp E ([x:cval] K (s+ x)) E'.

% note duplication of K
cmp_case : cmp (case E1 E2 E3) K E'
           <- cmp E2 K E2'
           <- ({x:exp} {x':cval}
                 ({K'} cmp x K' (K' x'))
                 -> cmp (E3 x) K (E3' x'))
           <- cmp E1 ([x1:cval] case+ x1 E2' E3') E'.
```

Compilation to CPS II

```
cmp_pair :
   cmp (pair E1 E2) K E'
     <- ({x1':cval}
         cmp E2 ([x2':cval] K (pair+ x1' x2')) (E2' x1'))
     <- cmp E1 E2' E'.

cmp_fst : cmp (fst E) K E'
           <- cmp E ([x:cval] fst+ x K) E'.

cmp_snd : cmp (snd E) K E'
           <- cmp E ([x:cval] snd+ x K) E'.
```

Compilation to CPS III

```
cmp_lam : cmp (lam E) K (K (lam+ E'))
           <- ({x:exp} {x':cval}
                 ({K'} cmp x K' (K' x'))
                 -> {k:cval -> cexp}
                    cmp (E x) k (E' x' k)).

cmp_app : cmp (app E1 E2) K E'
           <- ({x1:cval}
                cmp E2 ([x2:cval] app+ x1 x2 K) (E2' x1))
           <- cmp E1 ([x1:cval] E2' x1) E'.
```

Evaluation of CPS I

```
ceval : cexp -> cval -> type.

ceval_vl : ceval (vl+ V) V.

% no ceval_z or ceval_ss
ceval_case_z : ceval (case+ z+ E2 E3) V
               <- ceval E2 V.
ceval_case_s : ceval (case+ (s+ V1') E2 E3) V
                <- ceval (E3 V1') V.
```

Evaluation of CPS II

```
% no ceval_pair
ceval_fst : ceval (fst+ (pair+ V1 V2) K) V
             <- ceval (K V1) V.
ceval_snd : ceval (snd+ (pair+ V1 V2) K) V
             <- ceval (K V2) V.

% no ceval_lam
ceval_app : ceval (app+ (lam+ E1') V2 K) V
             <- ceval (E1' V2 K) V.

% no ceval_letv
% letn and fix omitted for now
```

Examples I

```
?- cmp (lam [x] pair (snd x) (fst x)) ([x] vl+ x) E.

E =
   vl+ (lam+ [x':cval] [k:cval -> cexp]
            snd+ x' ([x1':cval] fst+ x' ([x2':cval] k (pair+ x1' x2')))).
no more solutions

?- cmp (app (lam [x] pair (snd x) (fst x)) (pair z (s z)))
   ([x] vl+ x) E.

E =
   app+
      (lam+ [x':cval] [k:cval -> cexp]
          snd+ x' ([x1':cval] fst+ x' ([x2':cval] k (pair+ x1' x2'))))
      (pair+ z+ (s+ z+)) ([x:cval] vl+ x).
```

Examples II

```
?- D: ceval
    (app+
       (lam+ [x':cval] [k:cval -> cexp]
          snd+ x' ([x1':cval] fst+ x' ([x2':cval] k (pair+ x1' x2'))))
       (pair+ z+ (s+ z+)) ([x:cval] vl+ x))
    V.

V = pair+ (s+ z+) z+,
D = ceval_app (ceval_snd (ceval_fst ceval_vl)).
no more solutions
```