```
%%% The Mini-ML Language
%%% Author: Frank Pfenning, based on [Michaylov & Pfenning 92]

exp  : type.  %name exp E.

z     : exp.
s     : exp -> exp.
case  : exp -> exp -> (exp -> exp) -> exp.
pair  : exp -> exp -> exp.
fst   : exp -> exp.
snd   : exp -> exp.
lam   : (exp -> exp) -> exp.
app   : exp -> exp -> exp.
letv  : exp -> (exp -> exp) -> exp.
letn  : exp -> (exp -> exp) -> exp.
fix   : (exp -> exp) -> exp.


%%% Mini-ML types.
%%% Author: Frank Pfenning, based on [Michaylov & Pfenning 92]

tp : type.  %name tp T.

nat   : tp.
cross : tp -> tp -> tp.
arrow : tp -> tp -> tp.


%%% Mini-ML Natural Semantics
%%% Author: Frank Pfenning, based on [Michaylov & Pfenning 92]

eval : exp -> exp -> type.  %name eval D.
%mode +{E:exp} -{V:exp} eval E V.

% Natural Numbers
ev_z      : eval z z.
```

```
ev_s       : eval (s E) (s V)
      <- eval E V.
ev_case_z  : eval (case E1 E2 E3) V
      <- eval E1 z
      <- eval E2 V.
ev_case_s  : eval (case E1 E2 E3) V
      <- eval E1 (s V1')
      <- eval (E3 V1') V.

% Pairs
ev_pair : eval (pair E1 E2) (pair V1 V2)
     <- eval E1 V1
     <- eval E2 V2.
ev_fst  : eval (fst E) V1
     <- eval E (pair V1 V2).
ev_snd  : eval (snd E) V2
     <- eval E (pair V1 V2).

% Functions
ev_lam  : eval (lam E) (lam E).
ev_app  : eval (app E1 E2) V
    <- eval E1 (lam E1')
    <- eval E2 V2
    <- eval (E1' V2) V.

% Definitions
ev_letv : eval (letv E1 E2) V
     <- eval E1 V1
     <- eval (E2 V1) V.
ev_letn : eval (letn E1 E2) V
     <- eval (E2 E1) V.

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

```
%%% The Mini-ML typing rules which can be used as a complete
%%% implementation oftp type inference.
%%% Author: Frank Pfenning, based on [Michaylov & Pfenning 92]

oftp : exp -> tp -> type.  %name oftp P.
%mode oftp +E *T.

% Natural Numbers
tp_z     : oftp z nat.
tp_s     : oftp (s E) nat
    <- oftp E nat.
tp_case  : oftp (case E1 E2 E3) T
    <- oftp E1 nat
    <- oftp E2 T
    <- ({x:exp} oftp x nat -> oftp (E3 x) T).

% Pairs
tp_pair : oftp (pair E1 E2) (cross T1 T2)
   <- oftp E1 T1
   <- oftp E2 T2.
tp_fst  : oftp (fst E) T1
   <- oftp E (cross T1 T2).
tp_snd  : oftp (snd E) T2
   <- oftp E (cross T1 T2).

% Functions
tp_lam : oftp (lam E) (arrow T1 T2)
  <- ({x:exp} oftp x T1 -> oftp (E x) T2).
tp_app : oftp (app E1 E2) T1
  <- oftp E1 (arrow T2 T1)
  <- oftp E2 T2.

% Definitions
tp_letv : oftp (letv E1 E2) T2
   <- oftp E1 T1
```

```
   <- ({x:exp} oftp x T1 -> oftp (E2 x) T2).
tp_letn : oftp (letn E1 E2) T2
   <- oftp E1 T1
   <- oftp (E2 E1) T2.

% Recursion
tp_fix : oftp (fix E) T
  <- ({x:exp} oftp x T -> oftp (E x) T).


%%% Representation of the Proof of Type Preservation
%%% Author: Frank Pfenning, based on [Michaylov & Pfenning 92]

tps : eval E V -> of E T -> of V T -> type.
%mode tps +D +P -Q.

% Natural Numbers
tps_z     : tps (ev_z) (tp_z) (tp_z).
tps_s     : tps (ev_s D1) (tp_s P1) (tp_s Q1)
     <- tps D1 P1 Q1.

tps_case_z: tps (ev_case_z D2 D1) (tp_case P3 P2 P1) Q2
     <- tps D2 P2 Q2.

tps_case_s: tps (ev_case_s D3 D1) (tp_case P3 P2 P1) Q3
     <- tps D1 P1 (tp_s Q1')
     <- tps D3 (P3 V1 Q1') Q3.

% Pairs
tps_pair : tps (ev_pair D2 D1) (tp_pair P2 P1) (tp_pair Q2 Q1)
      <- tps D1 P1 Q1
      <- tps D2 P2 Q2.
tps_fst  : tps (ev_fst D1) (tp_fst P1) Q1
      <- tps D1 P1 (tp_pair Q2 Q1).
tps_snd  : tps (ev_snd D1) (tp_snd P1) Q2
      <- tps D1 P1 (tp_pair Q2 Q1).
```

```
% Functions
tps_lam  : tps (ev_lam) (tp_lam P) (tp_lam P).

tps_app  : tps (ev_app D3 D2 D1) (tp_app P2 P1) Q3
    <- tps D1 P1 (tp_lam Q1')
    <- tps D2 P2 Q2
    <- tps D3 (Q1' V2 Q2) Q3.

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


tps_letn : tps (ev_letn D2) (tp_letn P2 P1) Q2
    <- tps D2 P2 Q2.


% Recursion
tps_fix : tps (ev_fix D1) (tp_fix P1) Q1
    <- tps D1 (P1 _ (tp_fix P1)) Q1.


%%% Decidablity proof for type preservation theorem
%terminates D (tps D _ _).
```