```
%
% Expressions
%
exp : type.
cell : type.

lam : (exp -> exp)->exp.
app : exp -> exp -> exp.

rf : cell -> exp.
ref : exp -> exp.
! : exp -> exp.
seq : exp -> exp -> exp.
is : exp -> exp -> exp.


%
% Stores
%
store : type.
cv : type.

estore : store.
, : store -> cv -> store.  %infix left 10 ,
holds : cell -> exp -> cv.


%
% Instructions
%
instr : type.

ev : exp -> instr.
ret : exp -> instr.
app1 : exp -> exp -> instr.
```

```
ref1 : exp -> instr.
!1 : exp -> instr.
is1 : exp -> exp -> instr.
is2 : exp -> exp -> instr.


%
% Continuations
%
cont : type.
init : cont.

klam : cont -> (exp -> instr) -> cont.


%
% Answers
%
ans : type.

close : store -> exp -> ans.
new : (cell -> ans) -> ans.


%
% Evaluation
%
eval : cont -> instr -> ans -> type.


ev_init : eval init (ret V) (close S V)
        o- collect S.

ev_cont : eval (klam K I) (ret V) A
        o- eval K (I V) A.
```

```
ev_lam : eval K (ev (lam E)) A
        o- eval K (ret (lam E)) A.


ev_app : eval K (ev (app E1 E2)) A
        o- eval (klam K ([x:exp] app1 x E2)) (ev E1) A.

ev_app1 : eval K (app1 (lam E) E') A
        o- eval (klam K ([x:exp] ev (E x))) (ev E') A.


ev_cell : eval K (ev (rf C)) A
        o- eval K (ret (rf C)) A.


ev_ref : eval K (ev (ref E)) A
        o- eval (klam K ([x:exp] ref1 x)) (ev E) A.

ev_ref1 : eval K (ref1 V) (new A)
        o- {c:cell} (contains c V -o eval K (ret (rf c)) (A c)).


ev_! : eval K (ev (! E)) A
        o- eval (klam K ([x:exp] !1 x)) (ev E) A.

ev_!1 : eval K (!1 C) A
        o- (read C V  &  eval K (ret V) A).


ev_is : eval K (is E1 E2) A
        o- eval (klam K ([x:exp] is1 x E2)) (ev E1) A.

ev_is1 : eval K (is1 C E) A
        o- eval (klam K ([x:exp] is2 C x)) (ev E) A.
```

```
ev_is2 : eval K (is2 (rf C) V) A
        o- contains C V'
        o- (contains C V -o eval K (ret unit) A).

%
% Contents of a cell
%
contains : cell -> exp -> type.  % no constructors, only parameters


%
% Collection of store
%
collect : store -> type.

col_empty : collect estore.

col_cv : collect (S , (holds C V))
        o- contains C V
        o- collect S.


%
% Reading cell value
%
read : cell -> exp -> type.

read_val : read C V
        o- contains C V
        o- <T>.
```