# 1 Refinement types

`R |- e : r` denotes that in the type environment `R` the expression `e` has the refinement type `r`.

`R |> p : R1` denotes that in the type environment `R` the typing of the program `p` results in the refinement type environment `R1`.

## 1.1 Expressions

```
      |= P(c)                  formula(R) |= P(x)          R(b) = r        R |- e : r
-----------------------        -----------------------     ----------      ------------
R |- c : {v : t | P(v)}        R |- x : {v : t | P(v)}     R |- b : r      R |- (e) : r

R |- o : r1 -> r2   R |- e : r1        R |- e1 : r1   R |- o : r1 -> r2 -> r   R |- e2 : r2
-------------------------------        ----------------------------------------------------
R |- o e : r2                          R |- e1 o e2 : r

R, e1 |- e2 : r   R, \not e1 |- e3 : r              R |- e1 : (x : r1 -> r2)   R |- e2 : r1
--------------------------------------              ---------------------------------------
R |- if e1 then e2 else e3 : r                      R |- e1 e2 : r2 [e2/x]

R, x : r1 |- e : r2              R |> p : R1   R1 |- e : r
-------------------------        -------------------------
R |- fn x => e : r1 -> r2        R |- let p in e end : r
```

## 1.2 Declarations

```
R |- e : r  R1 = R + {x : r}
----------------------------
R |> val x = e : R1

R, f : (x : r1 -> r2 ), x : r1 |- e : r2    R1 = R + {f : (x : r1 -> r2)}
-------------------------------------------------------------------------
R |> fun f x = e : R1
```

## 1.3 Programs

```
R0 |> d1 : R1  ...  RN |> dN : R(N+1)
-------------------------------------
R0 |> d1 ... dN : R(N+1)
```