8-1) R_2 :

```
e ::=  .... | true | false
     | (and e e) | (or e e) | (not e)
     | (cmp e e) | (if e e e) | (- e e)
cmp ::= == | < | ≤ | ≥ | >
ty ::= S64 | Bool
e' ::= .... | true | false | (cmp e e) | (if e e e)
(and x y) = (if x y false)
(or x y) = (if x true y)
(not x) = (if x false true)
(- x y) = (+ x (- y))
```

8-2) interp : ~~A~~ → ans → ~~num~~ num or bool

~~A1~~ → A2

```
interp env true = true
           false = false
      (If ec et ef) = interp env k
           where k = if (interp env ec) then et else ef
      ((Cmp op eL eR) = (interp env eL) op (interp env eR)
```

(+ 1 true) (not 5) (< 5 true)

(and true 5) (if 5 false true)

↓

(if true 5 false)

don't match

not a bool

8-3) typec : e → ty (or errors)
x (v→ty)

$\Gamma \vdash e : ty$ means typec Γ e = ty

$\Gamma \vdash \text{true} : \text{Bool}$

$\Gamma \vdash \text{false} : \text{Bool}$

$\Gamma \vdash n : S64$

$$\frac{\Gamma \vdash e_L : S64 \qquad \Gamma \vdash e_R : S64}{\Gamma \vdash (+\ e_L\ e_R) : S64}$$

```
typec Γ (Add eL eR) =
  let tyL = typec Γ eL
      tyR = typec Γ eR
  case tyL of
    S64 → case tyR of
      S64 → S64
      _ → err
    _ → err
```

8-4/

$$\frac{\Gamma \vdash e_1 : S64}{\Gamma \vdash (- \ e_1) : S64}$$

$$\frac{\Gamma \vdash e_c : Bool \quad \begin{matrix} \Gamma \vdash e_f : T \\ \Gamma \vdash e_t : T \end{matrix}}{\Gamma \vdash (\text{if } e_c \ e_t \ e_f) : T}$$

$$\Gamma \vdash x : \Gamma(x)$$

$$\frac{\Gamma \vdash xe : T_x \qquad \Gamma[x \mapsto T_x] \vdash be : T_b}{\Gamma \vdash (\text{let } x = xe \text{ in } be) : T_b}$$

8-9/

randp: set(vars) × num → e

$$\text{randp} \quad \text{vs} \quad \text{o} = \left.\begin{array}{l} \text{number} \\ \text{variable-refencer} \\ \text{(read)} \end{array}\right\} \text{randomly}$$

bool

$\frac{1}{4} + \frac{1}{4} + \frac{1}{8} = \frac{5}{8}^2 \approx 40\%$

8-6)

```
randp : num $\Rightarrow$ e
randp n = rande $\emptyset$ (randomly SGY Bool) n
rande : (Ty $\Rightarrow$ set(vars)) $\times$ Ty $\times$ num $\Rightarrow$ e
rande $\Sigma$ Bool 0 = true or false or $\Sigma$(Bool)
               Num 0 = num or exat or $\Sigma$(SGY)
Bool (n+1) = ① cmp$_{\hat{}\text{random}}$ (rande $\Sigma$ SGY n) (rande $\Sigma$ SGY n)
             ② let x := (rand e $\Sigma$ (rand ty$_{\hat{}xt}$) n) in
                  rande $\Sigma'$ Bool n
                where $\Sigma' = \Sigma[xt \mapsto \Sigma(xt) \cup \{x\}]$
             ③ if (rande $\Sigma$ Bool n) (rande $\Sigma$ Bool n)
                  (rande $\Sigma$ Bool n)
```

8-7/

```
rande Σ S64 (n+1) =
   ① (+ (rande Σ S64 n) (rande Σ S64 n))
   ② let x := (rande Σ xt n) in
         (rande Σ[xt ↦ Σ(xt) ∪ {x}] S64 n)
      where xt = rand ty
   ③ (if (rande Σ Bool n) (rande Σ S64 n)
          (rande Σ S64 n)
```

8-8) opt : $r_2 \to r_2$

bools are simple, so don't opt

(cmp N_1 N_2) reduce to value

(if B_1 e_t e_f) reduce to either e_t or e_f

(not (not e)) $\xrightarrow{opt}$ e

(if (if e F T) F T) $\to$ e

(if (not e) e_t e_f) $\to$ (if e e_f e_t)

(if e T F) $\to$ e

(= e_1 e_1) $\to$ true if e_1 had no "reads"

(< e_1 (+ N e_1)) $\to$ true if $N > 0$

(if c e_1 e_1) = let _ = c in e_1

(if x e_t e_f) = (if x $e_t[x \mapsto true]$
$e_f[x \mapsto false]$)

(if (not x) e_t e_f) = (if x e_f e_t) =
= (if x $e_f[x \mapsto true]$
$e_t[x \mapsto false]$)

8-9

```
opt Σ (let x := xe in be) =
  let (xe', xe-pure?, xe-vars) = opt Σ xe
      Σ' = Σ [x ↦ xe' if xe' pure? o.w. x]
      (be', be-pure?, be-vars) = opt Σ' be
  in
    if x ∉ be-vars then
      if xe-pure? then (be', be-pure?, be-vars)
      o.w. (let _ = xe' in be'), false, be-vars ∪ xe-vars
    o.w.
      (let x := xe' in be'), (and be-pure? xe-pure?), be-vars \ {x}
                                                          ∪ xe-vars
```