7-1

randp : xNum → e
setfuzz → depth

randp V 0 = number
variable-reference (if vars)
bool (50% true / 50% false)
(+ 1 true)

randp : ~~(Unit Type)~~ (Type → set(vars))
x Type (T)
x Num (depth)
→ expression of type T

randp N = rande (Mt) (Random (Int or Bool)) N
rande Σ Bool 0 = true / false / vars (Σ(Bool))
Num 0 = num / vars (Σ(Num))
Bool (1+n) = ① (cmp (rande Σ Num n) (rande Σ Num n))
Ty → random

② let x := xe in b T
T := ~~random-ty~~
where xe := rande Σ ~~(Random)~~ N
b := rande Σ(T) [add x] Ty N

VS' = Σ(T) ∪ {x}
Σ' = Σ[T ⇒ VS']
Σ'' = remove x from the other types

③ (if (rande Σ Bool N)
(rande Σ Ty N) (rande Σ Ty N))

7-2/ What optimization apply to R2?

bools are "simple"

(cmp N_1 N_2) can reduce to a value

(if B_1 e_T e_F) = either e_T or e_F depending on B_1

(not (not e)) ⇒ e (not x) = (if x f t)

(if (if e f t) f t) ⇒ e

(if (not c) t f) ⇒ (if c f t)

purity

~~(~~ (= e_1 e_1) ⇒ true ; if e_1 "has no read"

opt : (V ⇒ e) × e ⇒ e ← there is no read

change: e × bool × set(vars)

(< e_1 (+ N e_1)) ⇒ true ; if N > 0 ↑ vxed vars

(if c e_1 e_1) ⇒ seq(c e_1) c' = if x e_t e_f :=

if c' = some var x, then opt Σ[x⇒true] e_t opt Σ[x⇒false] e_f

seq : e × e ⇒ e

returns this value after this effect

seq (+ N read) 12 = (let _ := read in 12)

seq true 12 = 12

opt Σ (let x := xe in b) = (xe', xe-pure?, xe-vars) = opt Σ xe

if x ∈ b-vars Σ' = Σ[x ⇒ xe' if pure o.w. x]

(seq xe' b'), (and $^{xe}_{b}$-pure?), $\cup^{xe\_vars}_{b}$ (b', b-pure?, b-vars) = are simple opt Σ' b

o.w.

(let x := xe' in b'), , xe-vars ∪ (b-vars − {x})

7-3

$C_0 \Rightarrow C_1$

C_1 := arg := ... | true | false

cmp := = | < | ≤ | >= | >

exp := ... | (not arg) | (cmp arg arg)

tail := ... | (goto label) | (goto-if (cmp a a) label label)

interpc : (label ⇒ tail) tail

L+T (goto label) = interpc L+T L+T(label)

X_0 to X_1

X_1 := arg := ... | byte-reg register (byte-reg rax) ⇒ %al

cc := e | l | le | g | ge

instr := ... | xorq a,a | cmpq a,a | | set_cc a | movzbq a,a | jmp-if cc label

(set_cc: no space in printing; jmp-if cc label: prints jmpcc label)

remember cmpq $4, $5 5 < 4

set l %al ⇒ FALSE

both cmpq and movzbq cannot have nums in 2nd position

---

rco updated for R_2

rco-e := ... | (if (cmp rco-a rco-a) rco-e rco-e)

rco-c := ... | (not rco-a) | (cmp rco-a rco-a) |

| (if (cmp rco-a rco-a) rco-e rco-e)

rco-a := ... | bools (t & f)

7-4

rco σ (if e_c e_t e_f) :=

(nv_c, a_c) := rco σ e_c

(nv_t, a_t) := rco σ e_t

(nv_f, a_f) := rco σ e_f

(nv_c ++ nv_t ++ nv_f ++ (x, if

if (cmp a_L a_R)