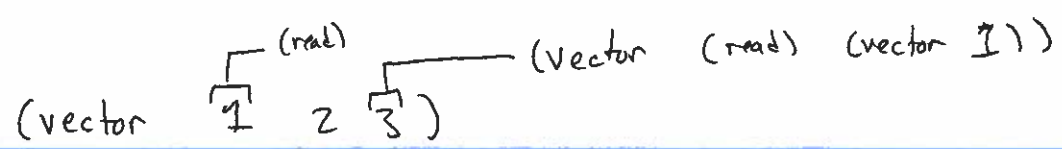


expose (allocations) $R_3 \rightarrow R_3$



⇒

(let $e_0 := 1$ in

$e_1 := 2$ in

$e_2 := 3$ in

— := $\left\{ \begin{array}{l} \text{if } (+ \text{ free-ptr } 4) < \text{ fromspace-end then} \\ \text{unit} \\ \text{else} \\ \text{(collect } \text{~~4~~ 4) in} \end{array} \right.$

$v := \text{allocate } 4 \text{ (Vector } 564 \text{ } 564 \text{ } 564) \text{ in}$

— := vector-set! v 0 e_0 in

— := vector-set! v 1 e_1 in

— := vector-set! v 2 e_2 in

v)

- expose :
- fix the order of operations for vector
 - detect when GC is needed
 - allocate space
 - initialize object

R_3 $e := \dots$ | (collect number) (removing vector)
 | (allocate number ty)
 | (global string)

$\Gamma \vdash (\text{collect num}) : \text{Unit}$

$\Gamma \vdash (\text{global str}) : \text{S64}$

$\Gamma \vdash (\text{allocate num ty}) : \text{ty}$

fake \rightarrow growable heap \rightarrow fixed heap w/ gc

fake interp

global free ptr $\Rightarrow 0$

collect \rightarrow nothing

allocate $n \rightarrow$ vector of size $n-1$

global fromspace-end $\Rightarrow +\infty$

11-2 / update unify

update rco → collect →
 allocate → expr/complex (add)
 global → arg (num)
 unit → arg (num)
 vector-ref → expr/complex (add)
 vector-set! → just like add, but remove unitvars
 for unit constants

$C_1 \Rightarrow C_2$

C_2 :
 arg := ... | (global str) | (unit) | ~~var~~ | var: ty
 exp := ... | (allocate num ty) | (vector-ref arg num)
 stmt := ... | (collect num) | (vector-set! arg num arg)

econ $R_2(\text{rco form}) \Rightarrow C_2$

econ (let x := (allocate num ty) in body) =
 seq (set! x (allocate num ty)) (econ body)
 econ (let _ := collect num in body) =
 seq (collect num) (econ body)

uncover-locals : $C_2 \Rightarrow C_2$ (only different is info)

old: return a set of variables

uncover (program mt [type \mapsto bool] [BODY \mapsto seq (set! x 5)
 (set! y (read)
 (set! z (< x y)
 (return z))])
 ↓
 mt [ty \mapsto bool] [vars \mapsto (x y z)]

now: return a mapping of vars to types

↓
 [x \mapsto int, y \mapsto int
 z \mapsto bool]

$X_1 \rightarrow X_2$
 arg := ... | (global str) | (type ty)
 instr := ... | leag arg, arg (Q in C)
 ↗ ↖ ↗
 src dst
 load effective address quad