18-1

$P \Longrightarrow A$ $\qquad$ $\underbrace{P \equiv P'}_{}$ semantic equiv

$|||$ $\qquad$ $||$ $\qquad$ P and P' "do the same thing"

$P' \Longrightarrow A$ $\qquad$ $\underbrace{P \leq P'}_{}$ optimization metric

$\underbrace{\text{discovering } P'}_{}$ optimizer

"low-level languages" have fewer optimizations

Meaning : program $\rightarrow$ ans $\qquad$ ans = state $\rightarrow$ state

$$\text{meaning}\begin{pmatrix} \text{movq } \%rax, \%rcx \\ \text{addq } \%rcx, \$8 \end{pmatrix} = \lambda s.\ s\big[rcx \mapsto s(rax)+8$$
$$pc \mapsto pc+2$$
$$\text{flags} \mapsto \text{FLAGS}(s(rax)+8)\big]$$

meaning $(\lambda x.\ x+8) = $ [?]

$(\lambda x.\ x+4+4) =$

$(\lambda x.$ if $x == 8$ the $x << 2$ o.w $x+8) =$

$(\lambda x.\ x + fib(6))$

Inlining

$(\lambda x.\ A)\ B \equiv A[x \mapsto B]$ $\qquad$ inlining (β-rule)

most languages use β_v, add B must be a value

$+(n, m) \Longrightarrow n+m$ $\qquad$ $n, m \in \mathbb{N}$ $\qquad$ (δ-rule for +)

(if true A B) $\Longrightarrow$ A $\qquad$ (dead-code-elim) $\qquad$ (if-rule for true)

$(\lambda x.\ (+\ x\ 8))\ 17 \xRightarrow{\beta} (+\ 17\ 8) \xRightarrow{\delta} 25$

$(\lambda x.\ A)\ B \equiv B_e\ ;\ A[x \leftarrow B_v]$

(while C B) ⇒ (if C (B; while C B) void)

$(\lambda x. A)\ B$ let x = e in b

(let f = λx. A = (λx. b) e
(f B))

let clo = (vector _fun22 3 4) in
((vector-ref clo 0) clo B)

(first-class fun)

control-flow = data-flow in C, Racket but not
(what code runs) (what values are produced) Pascal

Waddell inlining algorithm from 1997

e = (const c) | (ref x) | (primref p)
| (if e_1 e_2 e_3) | (seq e_1 e_2) | (assign x e)
| (lambda (x) e) | (letrec ([x_1 e_1] ... [x_n e_n]) e_b)
| (call e_0 e_1)

NO TYPES
C-if (1 false, all else true)

$I : e \to (\text{Context} \times \text{Env} \times \text{Kont} \times \text{Store}) \to e$

Context = Test | Effect | Value | App (Operand, Context, Loc_γ)
Operand = Opnd (e, Env, Loc_e)
Env = Var → Var
Var = (Identifier, Operand ∪ {null}, VarFlags, Loc_x)
VarFlags ∈ {ref, assign}
Kont = e → Store → e
Store = (Loc_x → VarFlags) ContextFlags ⊆ {inlined}
× (Loc_γ → ContextFlags)
× (Loc_e → e ∪ {unvisited})

18-3

$I\ (\text{const } c)\ (r, p, k, \sigma)$

if r = Effect, k (const void) σ

if r = Test and $c \neq$ false,

k (const true) σ

o.w. k (const c) σ

result(e) =

e_2 if $e = \text{seq } e_1 e_2$

e o.w.

$I\ (\text{seq } e_1\ e_2)\ (r, p, k, \sigma)$

$I\ e_1\ (\text{Effect}, p,$

$(\lambda e_1', \sigma_1'.\ I\ e_2\ (r, p,$

$(\lambda e_2', \sigma_2'.\ k\ (\text{seq } e_1'\ e_2')\ \sigma_2'), \sigma_1')), \sigma)$

Suppose e_1' == (const void), return $k\ e_2'\ \sigma_2'$

$I\ (\text{if } e_1\ e_2\ e_3)\ (r, p, k, \sigma) =$

$I\ e_1$ Test $p\ k_1$

$k_1\ e_1'\ \sigma_1'$ = if result(e_1') = const true =

$I\ e_2\ r\ p\ (\lambda e_2', \sigma_2'.\ k\ (\text{seq } e_1', e_2')\ \sigma_2')\ \sigma_1'$

if result(e_1') = const false =

$I\ e_3\ r\ p\ (\lambda e_3', \sigma_3'.\ k\ (\text{seq } e_1'\ e_3')\ \sigma_3'))\ \sigma_1'$

o.w.

$I\ e_2\ r_1\ p\ (\lambda e_2', \sigma_2'.$

$I\ e_3\ r_1\ p\ k_2\ \sigma_2')\ \sigma_1'$

$k_2\ e_3'\ \sigma_3'$ = if $e_2' = e_3'$ = const c,

$k\ (\text{seq } e_1'\ e_2')\ \sigma_3'$

o.w. $k\ (\text{if } e_1'\ e_2'\ e_3')\ \sigma_3'$

r_1 = Value if $r = \text{App}(op, r_x, l_r)$

r o.w.