

# Concrete Semantics for Pushdown Analysis

**The Essence of Summarization**

**J. Ian Johnson** and David Van Horn

`{ianj,dvanhorn}@ccs.neu.edu`

Northeastern University

Boston, MA, USA



Two things:

Two things:

Pushdown analysis is easy

Two things:

Pushdown analysis is easy

You should model your analyses concretely

Two things:

**Pushdown analysis is easy**

You should model your analyses concretely

# Analyses are derivable [Might & Van Horn 2010]

- Start: Concrete machine semantics

# Analyses are derivable [Might & Van Horn 2010]

- Start: Concrete machine semantics
- Simple transforms: Put semantics in right form

# Analyses are derivable [Might & Van Horn 2010]

- Start: Concrete machine semantics
- Simple transforms: Put semantics in right form
- Analysis: finitize addresses

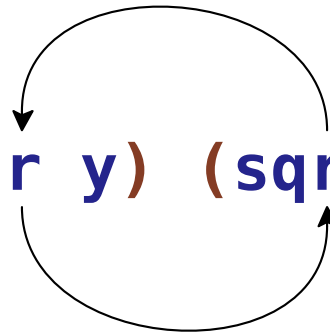


```
(define (sqr x) (* x x))
```

```
(sqrt (+ (sqr y) (sqr z)))
```

```
(define (sqr x) (* x x))
```

```
(sqrt (+ (sqr y) (sqr z)))
```



# Pushdown Higher-Order Flow Analysis

```

01  Summary, Callers, TCallers, Final  $\leftarrow \emptyset$ 
02  Seen, W  $\leftarrow \{(\tilde{I}(pr), \tilde{I}(pr))\}$ 
03  while W  $\neq \emptyset$ 
04    remove  $(\zeta_1, \zeta_2)$  from W
05    switch  $\zeta_2$ 
06      case  $\zeta_2$  of Entry, CApply, Inner-CEval
07        for each  $\zeta_3$  in succ( $\zeta_2$ ) Propagate( $\zeta_1, \zeta_3$ )
08      case  $\zeta_2$  of Call
09        for each  $\zeta_3$  in succ( $\zeta_2$ )
10          Propagate( $\zeta_3, \zeta_3$ )
11          insert  $(\zeta_1, \zeta_2, \zeta_3)$  in Callers
12          for each  $(\zeta_3, \zeta_4)$  in Summary Update( $\zeta_1, \zeta_2, \zeta_3, \zeta_4$ )
13      case  $\zeta_2$  of Exit-CEval
14        if  $\zeta_1 = \tilde{I}(pr)$  then
15          Final( $\zeta_2$ )
16        else
17          insert  $(\zeta_1, \zeta_2)$  in Summary
18          for each  $(\zeta_3, \zeta_4, \zeta_1)$  in Callers Update( $\zeta_3, \zeta_4, \zeta_1, \zeta_2$ )
19          for each  $(\zeta_3, \zeta_4, \zeta_1)$  in TCallers Propagate( $\zeta_3, \zeta_2$ )
20      case  $\zeta_2$  of Exit-TC
21        for each  $\zeta_3$  in succ( $\zeta_2$ )
22          Propagate( $\zeta_3, \zeta_3$ )
23          insert  $(\zeta_1, \zeta_2, \zeta_3)$  in TCallers
24          for each  $(\zeta_3, \zeta_4)$  in Summary Propagate( $\zeta_1, \zeta_4$ )

Propagate( $\zeta_1, \zeta_2$ )  $\triangleq$ 
25  if  $(\zeta_1, \zeta_2)$  not in Seen then insert  $(\zeta_1, \zeta_2)$  in Seen and W

Update( $\zeta_1, \zeta_2, \zeta_3, \zeta_4$ )  $\triangleq$ 
26   $\zeta_1$  of the form  $([\lambda_{l_1}(u_1 \ k_1) \ call_1]) , \hat{d}_1, h_1)$ 
27   $\zeta_2$  of the form  $([\lambda_{\gamma_2}(u_2) \ call_2])^{t_2}, tf_2, h_2)$ 
28   $\zeta_3$  of the form  $([\lambda_{l_3}(u_3 \ k_3) \ call_3]) , \hat{d}_3, h_2)$ 
29   $\zeta_4$  of the form  $([\lambda_{\gamma_4}(u_4) \ call_4]) , tf_4, h_4)$ 
30   $\hat{d} \leftarrow \hat{\mathcal{A}}_u(e_4, \gamma_4, tf_4, h_4)$ 
31   $tf \leftarrow \begin{cases} tf_2[f \mapsto \{[\lambda_{l_3}(u_3 \ k_3) \ call_3]\}] & S_{\gamma}(l_2, f) \\ tf_2 & H_{\gamma}(l_2, f) \vee Lam_{\gamma}(f) \end{cases}$ 
32   $\zeta \leftarrow ([\lambda_{\gamma_2}(u_2) \ call_2]) , \hat{d}, tf, h_4)$ 
33  Propagate( $\zeta_1, \zeta$ )

Final( $\zeta$ )  $\triangleq$ 
34   $\zeta$  of the form  $([\lambda_{\gamma}(u) \ call], tf, h)$ 
35  insert  $(halt, \hat{\mathcal{A}}_u(e, \gamma, tf, h), \emptyset, h)$  in Final

```

Figure 8: CFA2 workset algorithm

# Pushdown Higher-Order Flow Analysis

```

01  Summary, Callers, TCallers, Final ← ∅
02  Seen, W ← {(I(pr), I(pr))}
03  while W ≠ ∅
04    remove (ξ1, ξ2) from W
05    switch ξ2
06      case ξ2 of Entry, CApply, Inner-CEval
07        for each ξ3 in succ(ξ2) Propagate(ξ1, ξ3)
08      case ξ2 of Call
09        for each ξ3 in succ(ξ2)
10          Propagate(ξ3, ξ3)
11          insert (ξ1, ξ2, ξ3) in Callers
12          for each (ξ3, ξ4) in Summary Update(ξ1, ξ2, ξ3, ξ4)
13      case ξ2 of Exit-CEval
14        if ξ1 = I(pr) then
15          Final(ξ2)
16        else
17          insert (ξ1, ξ2) in Summary
18          for each (ξ3, ξ4, ξ1) in Callers Update(ξ3, ξ4, ξ1, ξ2)
19          for each (ξ3, ξ4, ξ1) in TCallers Propagate(ξ3, ξ2)
20      case ξ2 of Exit-TC
21        for each ξ3 in succ(ξ2)
22          Propagate(ξ3, ξ3)
23          insert (ξ1, ξ2, ξ3) in TCallers
24          for each (ξ3, ξ4) in Summary Propagate(ξ1, ξ4)

Propagate(ξ1, ξ2) ≜
25  if (ξ1, ξ2) not in Seen then insert (ξ1, ξ2) in Seen and W

Update(ξ1, ξ2, ξ3, ξ4) ≜
26  ξ1 of the form ([λl1(u1 k1) call1]), d1, h1)
27  ξ2 of the form ([λl2(u2 k2) call2])t2, tf2, h2)
28  ξ3 of the form ([λl3(u3 k3) call3]), d3, h2)
29  ξ4 of the form ([λl4(u4 k4) call4]), tf4, h4)
30  d ← Au(e4, γ4, tf4, h4)
31  tf ← { tf2[f ↦ { [λl3(u3 k3) call3]] } Sγ(l2, f)
           { tf2 } Hγ(l2, f) ∨ Lamγ(f)
32  ξ ← ([λl2(u2 k2) call2]), d, tf, h4)
33  Propagate(ξ1, ξ)

Final(ξ) ≜
34  ξ of the form ([λl(u) call]), tf, h)
35  insert (halt, Au(e, γ, tf, h), ∅, h) in Final

```

Figure 8: CFA2 workset algorithm

$$\begin{aligned}
\mathcal{F}'(M) &= f, \text{ where} \\
M &= (Q, \Gamma, \delta, q_0) \\
f(G, G_e, \Delta G, \Delta H) &= (G', G'_e, \Delta G', \Delta H' - H), \text{ where} \\
(S, \Gamma, E, s_0) &= G \\
(S, H) &= G_e \\
(\Delta S, \Delta E) &= \Delta G \\
(\Delta E_0, \Delta H_0) &= \bigcup_{s \in \Delta S} \text{sprout}_M(s) \\
(\Delta E_1, \Delta H_1) &= \bigcup_{(s, \gamma_+, s') \in \Delta E} \text{addPush}_M(G, G_e)(s, \gamma_+, s') \\
(\Delta E_2, \Delta H_2) &= \bigcup_{(s, \gamma_-, s') \in \Delta E} \text{addPop}_M(G, G_e)(s, \gamma_-, s') \\
(\Delta E_3, \Delta H_3) &= \bigcup_{(s, e, s') \in \Delta E} \text{addEmpty}_M(G, G_e)(s, s') \\
(\Delta E_4, \Delta H_4) &= \bigcup_{(s, s') \in \Delta H} \text{addEmpty}_M(G, G_e)(s, s') \\
S' &= S \cup \Delta S \\
E' &= E \cup \Delta E \\
H' &= H \cup \Delta H \\
\Delta E' &= \Delta E_0 \cup \Delta E_1 \cup \Delta E_2 \cup \Delta E_3 \cup \Delta E_4 \\
\Delta S' &= \{s' : (s, g, s') \in \Delta E'\} \\
\Delta H' &= \Delta H_0 \cup \Delta H_1 \cup \Delta H_2 \cup \Delta H_3 \cup \Delta H_4 \\
G' &= (S \cup \Delta S, \Gamma, E', q_0) \\
G'_e &= (S', H') \\
\Delta G' &= (\Delta S' - S', \Delta E' - E').
\end{aligned}$$

Figure 3. The fixed point of the function  $\mathcal{F}'(M)$  contains the Dyck state graph of the rooted pushdown system  $M$ .

# Pushdown Higher-Order Flow Analysis

```

01  Summary, Callers, TCallers, Final ← ∅
02  Seen, W ← {(I(pr), I(pr))}
03  while W ≠ ∅
04    remove (ξ1, ξ2) from W
05    switch ξ2
06      case ξ2 of Entry, CApply, Inner-CEval
07        for each ξ3 in succ(ξ2) Propagate(ξ1, ξ3)
08      case ξ2 of Call
09        for each ξ3 in succ(ξ2)
10          Propagate(ξ3, ξ3)
11          insert (ξ1, ξ2, ξ3) in Callers
12          for each (ξ3, ξ4) in Summary Update(ξ1, ξ2, ξ3, ξ4)
13      case ξ2 of Exit-CEval
14        if ξ1 = I(pr) then
15          Final(ξ2)
16        else
17          insert (ξ1, ξ2) in Summary
18          for each (ξ3, ξ4, ξ1) in Callers Update(ξ3, ξ4, ξ1, ξ2)
19          for each (ξ3, ξ4, ξ1) in TCallers Propagate(ξ3, ξ2)
20      case ξ2 of Exit-TC
21        for each ξ3 in succ(ξ2)
22          Propagate(ξ3, ξ3)
23          insert (ξ1, ξ2, ξ3) in TCallers
24          for each (ξ3, ξ4) in Summary Propagate(ξ1, ξ4)

Propagate(ξ1, ξ2) ≜
25  if (ξ1, ξ2) not in Seen then insert (ξ1, ξ2) in Seen and W

Update(ξ1, ξ2, ξ3, ξ4) ≜
26  ξ1 of the form ([λi1(u1 k1) call1]), d1, h1)
27  ξ2 of the form ([λi2(u2 k2) call2])i2, tf2, h2)
28  ξ3 of the form ([λi3(u3 k3) call3]), d3, h2)
29  ξ4 of the form ([λi4(u4 k4) call4]), tf4, h4)
30  d ← Au(e4, γ4, tf4, h4)
31  tf ← { tf2[f ↦ { [λi3(u3 k3) call3]] } Sγ(l2, f)
        { tf2 Hγ(l2, f) ∨ Lamγ(f)
32  ξ ← ([λi2(u2 k2) call2]), d, tf, h4)
33  Propagate(ξ1, ξ)

Final(ξ) ≜
34  ξ of the form ([λi(ui ki) calli]), tf, h)
35  insert (halt, Au(e, γ, tf, h), ∅, h) in Final

```

Figure 8: CFA2 workset algorithm

$$\begin{aligned}
& \text{sprout}_{(Q, \Gamma, \delta)}(s) = (\Delta E, \Delta H), \text{ where} \\
& \Delta E = \left\{ s \xrightarrow{e} q : s \xrightarrow{e} q \in \delta \right\} \cup \left\{ s \xrightarrow{\gamma_+} q : s \xrightarrow{\gamma_+} q \in \delta \right\} \\
& \Delta H = \left\{ s \xrightarrow{e} q : s \xrightarrow{e} q \in \delta \right\}. \\
& (\Delta S, \Delta E) = \Delta G \\
& (\Delta E_0, \Delta H_0) = \bigcup_{s \in \Delta S} \text{sprout}_M(s) \\
& (\Delta E_1, \Delta H_1) = \bigcup_{(s, \gamma_+, s') \in \Delta E} \text{addPush}_M(G, G_e)(s, \gamma_+, s') \\
& (\Delta E_2, \Delta H_2) = \bigcup_{(s, \gamma_-, s') \in \Delta E} \text{addPop}_M(G, G_e)(s, \gamma_-, s') \\
& (\Delta E_3, \Delta H_3) = \bigcup_{(s, e, s') \in \Delta E} \text{addEmpty}_M(G, G_e)(s, s') \\
& (\Delta E_4, \Delta H_4) = \bigcup_{(s, s') \in \Delta H} \text{addEmpty}_M(G, G_e)(s, s') \\
& S' = S \cup \Delta S \\
& E' = E \cup \Delta E \\
& H' = H \cup \Delta H \\
& \Delta E' = \Delta E_0 \cup \Delta E_1 \cup \Delta E_2 \cup \Delta E_3 \cup \Delta E_4 \\
& \Delta S' = \{ s' : (s, g, s') \in \Delta E' \} \\
& \Delta H' = \Delta H_0 \cup \Delta H_1 \cup \Delta H_2 \cup \Delta H_3 \cup \Delta H_4 \\
& G' = (S \cup \Delta S, \Gamma, E', q_0) \\
& G'_e = (S', H') \\
& \Delta G' = (\Delta S' - S', \Delta E' - E').
\end{aligned}$$

Figure 3. The fixed point of the function  $\mathcal{F}'(M)$  contains the Dyck state graph of the rooted pushdown system  $M$ .

# Pushdown Higher-Order Flow Analysis

```

01  Summary, Callers, TCallers, Final ← ∅
02  Seen, W ← {(I(pr), I(pr))}
03  while W ≠ ∅
04    remove (ξ1, ξ2) from W
05    switch ξ2
06      case ξ2 of Entry, CApply, Inner-CEval
07        for each ξ3 in succ(ξ2) Propagate(ξ1, ξ3)
08      case ξ2 of Call
09        for each ξ3 in succ(ξ2)
10          Propagate(ξ3, ξ3)
11          insert (ξ1, ξ2, ξ3) in Callers
12          for each (ξ3, ξ4) in Summary Update(ξ1, ξ2, ξ3, ξ4)
13      case ξ2 of Exit-CEval
14        if ξ1 = I(pr) then
15          Final(ξ2)
16        else
17          insert (ξ1, ξ2) in Summary
18          for each (ξ3, ξ4, ξ1) in Callers Update(ξ3, ξ4, ξ1, ξ2)
19          for each (ξ3, ξ4, ξ1) in TCallers Propagate(ξ3, ξ2)
20      case ξ2 of Exit-TC
21        for each ξ3 in succ(ξ2)
22          Propagate(ξ3, ξ3)
23          insert (ξ1, ξ2, ξ3) in TCallers
24          for each (ξ3, ξ4) in Summary Propagate(ξ1, ξ4)
25
26  Propagate(ξ1, ξ2) ≜
27    if (ξ1, ξ2) not in Seen then insert (ξ1, ξ2) in Seen and W
28
29  Update(ξ1, ξ2, ξ3, ξ4) ≜
30    ξ1 of the form ([λi1(u1 k1) call1]), d1, h1)
31    ξ2 of the form ([λi2(u2 k2) call2])t2, tf2, h2)
32    ξ3 of the form ([λi3(u3 k3) call3]), d3, h2)
33    ξ4 of the form ([λi4(u4 k4) call4]), tf4, h4)
34    d ← Au(e4, γ4, tf4, h4)
35    tf ← { tf2[f ↦ { [λi3(u3 k3) call3]] } Sγ(l2, f)
36           { tf2 } Hγ(l2, f) ∨ Lamγ(f)
37    ξ ← ([λi2(u2 k2) call2]), d, tf, h4)
38    Propagate(ξ1, ξ)
39
40  Final(ξ) ≜
41    ξ of the form ([λi(ui ki) calli]), tf, h)
42    insert (halt, Au(e, γ, tf, h), ∅, h) in Final

```

Figure 8: CFA2 workset algorithm

$\text{sprout}_{(Q, \Gamma, \delta)}(s) = (\Delta E, \Delta H), \text{ where}$ $\Delta E = \left\{ s \xrightarrow{\epsilon} q : s \xrightarrow{\epsilon} q \in \delta \right\} \cup \left\{ s \xrightarrow{\gamma_+} q : s \xrightarrow{\gamma_+} q \in \delta \right\}$ $\Delta H = \left\{ s \xrightarrow{\epsilon} q : s \xrightarrow{\epsilon} q \in \delta \right\}.$
$\text{addPush}_{(Q, \Gamma, \delta)}(G, G_e)(s \xrightarrow{\gamma_+} q) = (\Delta E, \Delta H), \text{ where}$ $\Delta E = \left\{ q' \xrightarrow{\gamma_-} q'' : q' \in \overline{G_e}[q] \text{ and } q' \xrightarrow{\gamma_-} q'' \in \delta \right\}$ $\Delta H = \left\{ s \xrightarrow{\epsilon} q'' : q' \in \overline{G_e}[q] \text{ and } q' \xrightarrow{\gamma_-} q'' \in \delta \right\}.$

$$(\Delta E_3, \Delta H_3) = \bigcup_{(s, \gamma_-, s') \in \Delta E} \text{addEmpty}_M(G, G_e)(s, s')$$

$$(\Delta E_4, \Delta H_4) = \bigcup_{(s, s') \in \Delta H} \text{addEmpty}_M(G, G_e)(s, s')$$

$$S' = S \cup \Delta S$$

$$E' = E \cup \Delta E$$

$$H' = H \cup \Delta H$$

$$\Delta E' = \Delta E_0 \cup \Delta E_1 \cup \Delta E_2 \cup \Delta E_3 \cup \Delta E_4$$

$$\Delta S' = \{ s' : (s, g, s') \in \Delta E' \}$$

$$\Delta H' = \Delta H_0 \cup \Delta H_1 \cup \Delta H_2 \cup \Delta H_3 \cup \Delta H_4$$

$$G' = (S \cup \Delta S, \Gamma, E', q_0)$$

$$G'_e = (S', H')$$

$$\Delta G' = (\Delta S' - S', \Delta E' - E').$$

Figure 3. The fixed point of the function  $\mathcal{F}'(M)$  contains the Dyck state graph of the rooted pushdown system  $M$ .

# Pushdown Higher-Order Flow Analysis

```

01  Summary, Callers, TCallers, Final ← ∅
02  Seen, W ← {(I(pr), I(pr))}
03  while W ≠ ∅
04    remove (ξ1, ξ2) from W
05    switch ξ2
06      case ξ2 of Entry, CApply, Inner-CEval
07        for each ξ3 in succ(ξ2) Propagate(ξ1, ξ3)
08      case ξ2 of Call
09        for each ξ3 in succ(ξ2)
10          Propagate(ξ3, ξ3)
11          insert (ξ1, ξ2, ξ3) in Callers
12          for each (ξ3, ξ4) in Summary Update(ξ1, ξ2, ξ3, ξ4)
13      case ξ2 of Exit-CEval
14        if ξ1 = I(pr) then
15          Final(ξ2)
16        else
17          insert (ξ1, ξ2) in Summary
18          for each (ξ3, ξ4, ξ1) in Callers Update(ξ3, ξ4, ξ1, ξ2)
19          for each (ξ3, ξ4, ξ1) in TCallers Propagate(ξ3, ξ2)
20      case ξ2 of Exit-TC
21        for each ξ3 in succ(ξ2)
22          Propagate(ξ3, ξ3)
23          insert (ξ1, ξ2, ξ3) in TCallers
24          for each (ξ3, ξ4) in Summary Propagate(ξ1, ξ4)
25  Propagate(ξ1, ξ2) ≜
    if (ξ1, ξ2) not in Seen then insert (ξ1, ξ2) in Seen and W
26  Update(ξ1, ξ2, ξ3, ξ4) ≜
    ξ1 of the form ([λi1(u1 k1) call1]), d1, h1)
    ξ2 of the form ([λi2(u2 k2) call2])i2, tf2, h2)
    ξ3 of the form ([λi3(u3 k3) call3]), d3, h2)
    ξ4 of the form ([λi4(u4 k4) call4]), tf4, h4)
    d ← Au(e4, γ4, tf4, h4)
    tf ← { tf2[f ↦ { [λi3(u3 k3) call3]] } Sγ(l2, f)
          { tf2 } Hγ(l2, f) ∨ Lamγ(f)
    ξ ← ([λi2(u2) call2]), d, tf, h4)
    Propagate(ξ1, ξ)
27  Final(ξ) ≜
    ξ of the form ([λi(u) call]), tf, h)
    insert (halt, Au(e, γ, tf, h), ∅, h) in Final

```

Figure 8: CFA2 workset algorithm

$\text{sprout}_{(Q, \Gamma, \delta)}(s) = (\Delta E, \Delta H), \text{ where}$ $\Delta E = \left\{ s \xrightarrow{\epsilon} q : s \xrightarrow{\epsilon} q \in \delta \right\} \cup \left\{ s \xrightarrow{\gamma^+} q : s \xrightarrow{\gamma^+} q \in \delta \right\}$ $\Delta H = \left\{ s \xrightarrow{\epsilon} q : s \xrightarrow{\epsilon} q \in \delta \right\}.$
$\text{addPush}_{(Q, \Gamma, \delta)}(G, G_e)(s \xrightarrow{\gamma^+} q) = (\Delta E, \Delta H), \text{ where}$ $\Delta E = \left\{ q' \xrightarrow{\gamma^-} q'' : q' \in \overrightarrow{G_e}[q] \text{ and } q' \xrightarrow{\gamma^-} q'' \in \delta \right\}$ $\Delta H = \left\{ s \xrightarrow{\epsilon} q'' : q' \in \overrightarrow{G_e}[q] \text{ and } q' \xrightarrow{\gamma^-} q'' \in \delta \right\}.$
$\text{addEmpty}_{(Q, \Gamma, \delta)}(G, G_e)(s'' \xrightarrow{\gamma^+} s''') = (\Delta E, \Delta H), \text{ where}$ $\Delta E = \left\{ s'''' \xrightarrow{\gamma^-} q : s' \in \overleftarrow{G_e}[s''] \text{ and } s'''' \in \overrightarrow{G_e}[s'''] \text{ and } s \xrightarrow{\gamma^+} s' \in G \right\}$ $\Delta H = \left\{ s \xrightarrow{\epsilon} q : s' \in \overleftarrow{G_e}[s''] \text{ and } s'''' \in \overrightarrow{G_e}[s'''] \text{ and } s \xrightarrow{\gamma^+} s' \in G \right\}$ $\cup \left\{ s' \xrightarrow{\epsilon} s'' : s' \in \overleftarrow{G_e}[s''] \right\}$ $\cup \left\{ s'' \xrightarrow{\epsilon} s'''' : s'''' \in \overrightarrow{G_e}[s'''] \right\}$ $\cup \left\{ s' \xrightarrow{\epsilon} s'''' : s' \in \overleftarrow{G_e}[s''] \text{ and } s'''' \in \overrightarrow{G_e}[s'''] \right\}.$

$$G' = (S \cup \Delta S, \Gamma, E', q_0)$$

$$G'_e = (S', H')$$

$$\Delta G' = (\Delta S' - S', \Delta E' - E').$$

Figure 3: The fixed point of the function  $\mathcal{F}'(M)$  contains the Dyck state graph of the rooted pushdown system  $M$ .

# Pushdown Higher-Order Flow Analysis

```

01  Summary, Callers, TCallers, Final ← ∅
02  Seen, W ← {(I(pr), I(pr))}
03  while W ≠ ∅
04    remove (ξ1, ξ2) from W
05    switch ξ2
06      case ξ2 of Entry, CApply, Inner-CEval
07        for each ξ3 in succ(ξ2) Propagate(ξ1, ξ3)
08      case ξ2 of Call
09        for each ξ3 in succ(ξ2)
10          Propagate(ξ3, ξ3)
11          insert (ξ1, ξ2, ξ3) in Callers
12          for each (ξ3, ξ4) in Summary Update(ξ1, ξ2, ξ3, ξ4)
13      case ξ2 of Exit-CEval
14        if ξ1 = I(pr) then
15          Final(ξ2)
16        else
17          insert (ξ1, ξ2) in Summary
18          for each (ξ3, ξ4, ξ1) in Callers Update(ξ3, ξ4, ξ1, ξ2)
19          for each (ξ3, ξ4, ξ1) in TCallers Propagate(ξ3, ξ2)
20      case ξ2 of Exit-TC
21        for each ξ3 in succ(ξ2)
22          Propagate(ξ3, ξ3)
23          insert (ξ1, ξ2, ξ3) in TCallers
24          for each (ξ3, ξ4) in Summary Propagate(ξ1, ξ4)
25
26  Propagate(ξ1, ξ2) ≜
27    if (ξ1, ξ2) not in Seen then insert (ξ1, ξ2) in Seen and W
28
29  Update(ξ1, ξ2, ξ3, ξ4) ≜
30    ξ1 of the form ([λl1(u1 k1) call1]), d1, h1)
31    ξ2 of the form ([λl2(u2 k2) call2])t2, tf2, h2)
32    ξ3 of the form ([λl3(u3 k3) call3]), d3, h2)
33    ξ4 of the form ([λl4(u4 k4) call4])t4, tf4, h4)
34    d ← Au(e4, γ4, tf4, h4)
35    tf ← { tf2[f ↦ { [λl3(u3 k3) call3]] } Sγ(l2, f)
36           { tf2 } Hγ(l2, f) ∨ Lamγ(f)
37    ξ ← ([λl2(u2) call2]), d, tf, h4)
38    Propagate(ξ1, ξ)
39
40  Final(ξ) ≜
41    ξ of the form ([λl(u) call]), tf, h)
42    insert (halt, Au(e, γ, tf, h), ∅, h) in Final

```

Figure 8: CFA2 workset algorithm

$$\begin{aligned}
 \hat{f}((\hat{P}, \hat{E}), \hat{H}, \hat{\sigma}) &= ((\hat{P}', \hat{E}'), \hat{H}', \hat{\sigma}'), \text{ where} \\
 \hat{T}_+ &= \left\{ (\hat{\psi} \xrightarrow{\hat{\phi}_+} \hat{\psi}', \hat{\sigma}') : \hat{\psi} \xrightarrow{\hat{\phi}_+} (\hat{\psi}', \hat{\sigma}') \right\} \\
 \hat{T}_e &= \left\{ (\hat{\psi} \xrightarrow{e} \hat{\psi}', \hat{\sigma}') : \hat{\psi} \xrightarrow{e} (\hat{\psi}', \hat{\sigma}') \right\} \\
 \hat{T}_- &= \left\{ (\hat{\psi}'' \xrightarrow{\hat{\phi}_-} \hat{\psi}', \hat{\sigma}') : \hat{\psi}'' \xrightarrow{\hat{\phi}_-} (\hat{\psi}', \hat{\sigma}') \text{ and} \right. \\
 &\quad \left. \hat{\psi} \xrightarrow{\hat{\phi}_+} \hat{\psi}' \in \hat{E} \text{ and} \right. \\
 &\quad \left. \hat{\psi}' \xrightarrow{\hat{\phi}_-} \hat{\psi}'' \in \hat{H} \right\} \\
 \hat{T}' &= \hat{T}_+ \cup \hat{T}_e \cup \hat{T}_- \\
 \hat{E}' &= \left\{ \hat{e} : (\hat{e}, \_) \in \hat{T}' \right\} \\
 \hat{\sigma}'' &= \bigsqcup \left\{ \hat{\sigma}' : (\_, \hat{\sigma}') \in \hat{T}' \right\} \\
 \hat{H}_e &= \left\{ \hat{\psi} \mapsto \hat{\psi}' : \hat{\psi} \mapsto \hat{\psi}' \in \hat{H} \text{ and } \hat{\psi}' \mapsto \hat{\psi}'' \in \hat{H} \right\} \\
 \hat{H}_{+-} &= \left\{ \hat{\psi} \mapsto \hat{\psi}'' : \hat{\psi} \xrightarrow{\hat{\phi}_+} \hat{\psi}' \in \hat{E} \text{ and } \hat{\psi}' \mapsto \hat{\psi}'' \in \hat{H} \right. \\
 &\quad \left. \text{and } \hat{\psi}'' \xrightarrow{\hat{\phi}_-} \hat{\psi}''' \in \hat{E} \right\} \\
 \hat{H}' &= \hat{H}_e \cup \hat{H}_{+-} \\
 \hat{P}' &= \hat{P} \cup \left\{ \hat{\psi}' : \hat{\psi} \xrightarrow{g} \hat{\psi}' \right\}. \\
 G' &= (S \cup \Delta S, \Gamma, E', q_0) \\
 G'_e &= (S', H') \\
 \Delta G' &= (\Delta S' - S', \Delta E' - E').
 \end{aligned}$$

Figure 3. The fixed point of the function  $\mathcal{F}'(M)$  contains the Dyck state graph of the rooted pushdown system  $M$ .



# Pushdown Higher-Order Flow Analysis

```

01  Summary, Callers, TCallers, Final ← ∅
02  Seen, W ← {(I(pr), I(pr))}
03  while W ≠ ∅
04    remove (ξ1, ξ2) from W
05    switch ξ2
06      case ξ2 of Entry, CApply, Inner-CEval
07        for each ξ3 in succ(ξ2) Propagate(ξ1, ξ3)
08      case ξ2 of Call
09        for each ξ3 in succ(ξ2)
10          Propagate(ξ3, ξ3)
11          insert (ξ1, ξ2, ξ3) in Callers
12          for each (ξ3, ξ4) in Summary Update(ξ1, ξ2, ξ3, ξ4)
13      case ξ2 of Exit-CEval
14        if ξ1 = I(pr) then
15          Final(ξ2)
16        else
17          insert (ξ1, ξ2) in Summary
18          for each (ξ3, ξ4, ξ1) in Callers Update(ξ3, ξ4, ξ1, ξ2)
19          for each (ξ3, ξ4, ξ1) in TCallers Propagate(ξ3, ξ2)
20      case ξ2 of Exit-TC
21        for each ξ3 in succ(ξ2)
22          Propagate(ξ3, ξ3)
23          insert (ξ1, ξ2, ξ3) in TCallers
24          for each (ξ3, ξ4) in Summary Propagate(ξ1, ξ4)
25
26  Propagate(ξ1, ξ2) ≜
27    if (ξ1, ξ2) not in Seen then insert (ξ1, ξ2) in Seen and W
28
29  Update(ξ1, ξ2, ξ3, ξ4) ≜
30    ξ1 of the form ([λl1(u1 k1) call1]), d1, h1)
31    ξ2 of the form ([λl2(u2 k2) call2])t2, tf2, h2)
32    ξ3 of the form ([λl3(u3 k3) call3]), d3, h2)
33    ξ4 of the form ([λl4(u4 k4) call4])t4, tf4, h4)
34    d ← Au(e4, γ4, tf4, h4)
35    tf ← { tf2[f ↦ ([λl3(u3 k3) call3])] Sγ(l2, f)
           tf2 Hγ(l2, f) ∨ Lamγ(f)
36    ξ ← ([λl2(u2 k2) call2]), d, tf, h4)
37    Propagate(ξ1, ξ)
38
39  Final(ξ) ≜
40    ξ of the form ([λl(u) k] e), tf, h)
41    insert (halt, Au(e, γ, tf, h), ∅, h) in Final

```

Figure 8: CFA2 workset algorithm

$$\begin{aligned}
 \hat{f}((\hat{P}, \hat{E}), \hat{H}, \hat{\sigma}) &= ((\hat{P}', \hat{E}'), \hat{H}', \hat{\sigma}'), \text{ where} \\
 \hat{T}_+ &= \left\{ (\hat{\psi} \xrightarrow{\hat{\phi}_+} \hat{\psi}', \hat{\sigma}') : \hat{\psi} \xrightarrow{\hat{\phi}_+} (\hat{\psi}', \hat{\sigma}') \right\} \\
 \hat{T}_e &= \left\{ (\hat{\psi} \xrightarrow{e} \hat{\psi}', \hat{\sigma}') : \hat{\psi} \xrightarrow{e} (\hat{\psi}', \hat{\sigma}') \right\} \\
 \hat{T}_- &= \left\{ (\hat{\psi}'' \xrightarrow{\hat{\phi}_-} \hat{\psi}', \hat{\sigma}') : \hat{\psi}'' \xrightarrow{\hat{\phi}_-} (\hat{\psi}', \hat{\sigma}') \text{ and} \right. \\
 &\quad \left. \hat{\psi} \xrightarrow{\hat{\phi}_+} \hat{\psi}' \in \hat{E} \text{ and} \right. \\
 &\quad \left. \hat{\psi}' \xrightarrow{\hat{\phi}_-} \hat{\psi}'' \in \hat{H} \right\} \\
 \hat{T}' &= \hat{T}_+ \cup \hat{T}_e \cup \hat{T}_- \\
 \hat{E}' &= \left\{ \hat{e} : (\hat{e}, \_) \in \hat{T}' \right\} \\
 \hat{\sigma}'' &= \bigsqcup \left\{ \hat{\sigma}' : (\_, \hat{\sigma}') \in \hat{T}' \right\} \\
 \hat{H}_e &= \left\{ \hat{\psi} \mapsto \hat{\psi}' : \hat{\psi} \mapsto \hat{\psi}' \in \hat{H} \text{ and } \hat{\psi}' \mapsto \hat{\psi}'' \in \hat{H} \right\} \\
 \hat{H}_{+-} &= \left\{ \hat{\psi} \mapsto \hat{\psi}'' : \hat{\psi} \xrightarrow{\hat{\phi}_+} \hat{\psi}' \in \hat{E} \text{ and } \hat{\psi}' \mapsto \hat{\psi}'' \in \hat{H} \right. \\
 &\quad \left. \text{and } \hat{\psi}'' \xrightarrow{\hat{\phi}_-} \hat{\psi}''' \in \hat{E} \right\} \\
 \hat{H}' &= \hat{H}_e \cup \hat{H}_{+-} \\
 \hat{P}' &= \hat{P} \cup \left\{ \hat{\psi}' : \hat{\psi} \xrightarrow{g} \hat{\psi}' \right\}.
 \end{aligned}$$

$$\begin{aligned}
 \text{addPop}_{(Q, \Gamma, \delta)}(G, G_e)(s'' \xrightarrow{\gamma_-} q) &= (\Delta E, \Delta H), \text{ where} \\
 \Delta E = \emptyset \text{ and } \Delta H &= \left\{ s \mapsto q : s' \in \overline{G}_e[s''] \text{ and } s \xrightarrow{\gamma_+} s' \in G \right\}
 \end{aligned}$$

Figure 3: The fixed point of the function  $\mathcal{F}'(M)$  contains the Dyck state graph of the rooted pushdown system  $M$ .

```

01 Summary, Callers,
02 Seen, W ←
03 while W ≠ ∅
04   remove (ξ1, ξ2) from W
05   switch
06     case ξ1 of Exit-CEval
07       for each ξ3 in succ(ξ2) Propagate(ξ1, ξ3)
08     case ξ1 of Exit-TC
09       for each ξ3 in succ(ξ2)
10         Propagate(ξ1, ξ3)
11         insert (ξ1, ξ2, ξ3) in Callers
12       for each (ξ3, ξ4) in TCallers
13         case ξ3 of Exit-CEval
14           if ξ4 = I(pr) then
15             Final(ξ2)
16           else
17             insert (ξ1, ξ2) in Summary
18             for each (ξ3, ξ4, ξ1) in Callers
19               for each (ξ3, ξ4, ξ1) in TCallers
20                 case ξ2 of Exit-TC
21                   for each ξ4 in succ(ξ2)
22                     Propagate(ξ1, ξ4)
23                   insert (ξ1, ξ2, ξ4) in TCallers
24                   for each (ξ3, ξ4) in Summary
25                     Propagate(ξ1, ξ4)
26   if (ξ1, ξ2) in Seen then
27     Update(ξ1, ξ2, ξ1)
28   ξ1 of the form (λ1 (u1 k1) ... h1)
29   ξ2 of the form (λ2 (u2 k2) ... t2)
30   ξ3 of the form (λ3 (u3 k3) ... h2)
31   ξ4 of the form (λ4 (u4 k4) ... t2)
32   d ← Au(e4, γ4, h4)
33   tf ← (t2 | f) ↦ {[(u1 k1) ... (u2 k2) ... (u3 k3) ... (u4 k4) ... (t2 | f)]}
34   ξ ← (λ1 (u1 call2) ... h1)
35   Propagate(ξ, ξ)
36   Final(ξ)
37   ξ of the form (λ1 (u1 k1) ... t2)
38   insert (half (e1), t2, h1) in W

```

Figure 8: CFA2 workset algorithm

$$\text{and } \text{Pop}_{(Q, \Gamma, \delta)}(G, G_e)(s'' \xrightarrow{\gamma^-} q) = (\Delta E, \Delta H), \text{ where}$$

$$\Delta E = \emptyset \text{ and } \Delta H = \left\{ s \mapsto q : s' \in \overleftarrow{G_e}[s''] \text{ and } s \xrightarrow{\gamma^+} s' \in G \right\}$$

# Pushdown Higher-Order Flow Analysis

```

01  Summary, Callers, TCallers, Final ← ∅
02  Seen, W ← {(I(pr), I(pr))}
03  while W ≠ ∅
04    remove (ξ1, ξ2) from W
05    switch ξ2
06      case ξ2 of Entry, CApply, Inner-CEval
07        for each ξ3 in succ(ξ2) Propagate(ξ1, ξ3)
08      case ξ2 of Call
09        for each ξ3 in succ(ξ2)
10          Propagate(ξ3, ξ3)
11          insert (ξ1, ξ2, ξ3) in Callers
12          for each (ξ3, ξ4) in Summary Update(ξ1, ξ2, ξ3, ξ4)
13      case ξ2 of Exit-CEval
14        if ξ1 = I(pr) then
15          Final(ξ2)
16        else
17          insert (ξ1, ξ2) in Summary
18          for each (ξ3, ξ4, ξ1) in Callers Update(ξ3, ξ4, ξ1, ξ2)
19          for each (ξ3, ξ4, ξ1) in TCallers Propagate(ξ3, ξ2)
20      case ξ2 of Exit-TC
21        for each ξ3 in succ(ξ2)
22          Propagate(ξ3, ξ3)
23          insert (ξ1, ξ2, ξ3) in TCallers
24          for each (ξ3, ξ4) in Summary Propagate(ξ1, ξ4)

Propagate(ξ1, ξ2) ≜
25  if (ξ1, ξ2) not in Seen then insert (ξ1, ξ2) in Seen and W
Update(ξ1, ξ2, ξ3, ξ4) ≜
26  ξ1 of the form ([λi1(u1 k1) call1]), d1, h1)
27  ξ2 of the form ([λi2(u2 k2) call2])t2, tf2, h2)
28  ξ3 of the form ([λi3(u3 k3) call3]), d3, h2)
29  ξ4 of the form ([λi4(u4 k4) call4]), tf4, h4)
30  d ← Au(e4, γ4, tf4, h4)
31  tf ← { tf2[f ↦ { [λi3(u3 k3) call3]] } Sγ(l2, f)
           { tf2 Hγ(l2, f) ∨ Lamγ(f)
32  ξ ← ([λi2(u2 k2) call2]), d, tf, h4)
33  Propagate(ξ1, ξ)

Final(ξ) ≜
34  ξ of the form ([λi(ui ki) calli]), tf, h)
35  insert (halt, Au(e, γ, tf, h), ∅, h) in Final

```

Figure 8: CFA2 workset algorithm

$$\begin{aligned}
 \hat{f}((\hat{P}, \hat{E}), \hat{H}, \hat{\sigma}) &= ((\hat{P}', \hat{E}'), \hat{H}', \hat{\sigma}'), \text{ where} \\
 \hat{T}_+ &= \left\{ (\hat{\psi} \xrightarrow{\hat{\phi}_+} \hat{\psi}', \hat{\sigma}') : \hat{\psi} \xrightarrow{\hat{\phi}_+} (\hat{\psi}', \hat{\sigma}') \right\} \\
 \hat{T}_e &= \left\{ (\hat{\psi} \xrightarrow{e} \hat{\psi}', \hat{\sigma}') : \hat{\psi} \xrightarrow{e} (\hat{\psi}', \hat{\sigma}') \right\} \\
 \hat{T}_- &= \left\{ (\hat{\psi}'' \xrightarrow{\hat{\phi}_-} \hat{\psi}''', \hat{\sigma}') : \hat{\psi}'' \xrightarrow{\hat{\phi}_-} (\hat{\psi}''', \hat{\sigma}') \text{ and} \right. \\
 &\quad \left. \hat{\psi} \xrightarrow{\hat{\phi}_+} \hat{\psi}' \in \hat{E} \text{ and} \right. \\
 &\quad \left. \hat{\psi}' \xrightarrow{\hat{\phi}_-} \hat{\psi}'' \in \hat{H} \right\} \\
 \hat{T}' &= \hat{T}_+ \cup \hat{T}_e \cup \hat{T}_- \\
 \hat{E}' &= \{ \hat{e} : (\hat{e}, \cdot) \in \hat{T}' \} \\
 \hat{\sigma}'' &= \bigsqcup \{ \hat{\sigma}' : (\cdot, \hat{\sigma}') \in \hat{T}' \} \\
 \hat{H}_e &= \{ \hat{\psi} \mapsto \hat{\psi}' : \hat{\psi} \mapsto \hat{\psi}' \in \hat{H} \text{ and } \hat{\psi}' \mapsto \hat{\psi}'' \in \hat{H} \} \\
 \hat{H}_{+-} &= \{ \hat{\psi} \mapsto \hat{\psi}'' : \hat{\psi} \xrightarrow{\hat{\phi}_+} \hat{\psi}' \in \hat{E} \text{ and } \hat{\psi}' \mapsto \hat{\psi}'' \in \hat{H} \\
 &\quad \text{and } \hat{\psi}'' \xrightarrow{\hat{\phi}_-} \hat{\psi}''' \in \hat{E} \} \\
 \hat{H}' &= \hat{H}_e \cup \hat{H}_{+-} \\
 \hat{P}' &= \hat{P} \cup \{ \hat{\psi}' : \hat{\psi} \xrightarrow{g} \hat{\psi}' \}.
 \end{aligned}$$

$$\begin{aligned}
 \text{addPop}_{(Q, \Gamma, \delta)}(G, G_e)(s'' \xrightarrow{\gamma_-} q) &= (\Delta E, \Delta H), \text{ where} \\
 \Delta E &= \emptyset \text{ and } \Delta H = \left\{ s \mapsto q : s' \in \overline{G_e}[s''] \text{ and } s \xrightarrow{\gamma_+} s' \in G \right\}
 \end{aligned}$$

Figure 3: The fixed point of the function  $\mathcal{F}^*(M)$  contains the Dyck state graph of the rooted pushdown system  $M$ .

# Deriving Pushdown Analyses

- Start: Concrete machine semantics

# Deriving Pushdown Analyses

- Start: Concrete machine semantics
- Simple transform: memoize functions

# Deriving Pushdown Analyses

- Start: Concrete machine semantics
- Simple transform: memoize functions
- Simple transform: store functions' calling contexts

# Deriving Pushdown Analyses

- Start: Concrete machine semantics
- Simple transform: memoize functions
- Simple transform: store functions' calling contexts
- Analysis: finitize addresses

# CESK + M

- $M : \text{Context} \rightarrow \wp(\text{Value})$



# CESK + M

- $M : \text{Context} \rightarrow \wp(\text{Value})$
- $\text{Context} = \text{CES}$

# CESK + M

- $M : \text{Context} \rightarrow \wp(\text{Value})$
- $\text{Context} = \text{CES}$
- New continuation frame:  $\text{rt}(\text{ctx} : \text{Context})$

# CESK + M

- $M : \text{Context} \rightarrow \wp(\text{Value})$
- $\text{Context} = \text{CES}$
- New continuation frame:  $\text{rt}(\text{ctx} : \text{Context})$
- Eliminate frame = add value to  $M$

# CESK + M

- $M : \text{Context} \rightarrow \wp(\text{Value})$
- $\text{Context} = \text{CES}$
- New continuation frame:  $\text{rt}(\text{ctx} : \text{Context})$
- Eliminate frame = add value to M
- Function calls bypass if next CES in M

# CESKM + $\Xi$

- $\Xi : \text{Context} \rightarrow \wp(\text{Kont})$

# CESKM + $\Xi$

- $\Xi : \text{Context} \rightarrow \wp(\text{Kont})$
- Store up to rt/mt in  $\Xi$  on call

# CESKM + $\Xi$

- $\Xi : \text{Context} \rightarrow \wp(\text{Kont})$
- Store up to rt/mt in  $\Xi$  on call
- Return to all calling contexts after memoizing

# CESKM + $\Xi$

- $\Xi : \text{Context} \rightarrow \wp(\text{Kont})$
- Store up to rt/mt in  $\Xi$  on call
- Return to all calling contexts after memoizing
- Bypassing via memo still stores continuation



```
(let* ([id (λ (x) x)]  
      [app (λ (f y) (f y))]  
      [n1 (app id 1)]  
      [n2 (app id 2)])  
  (+ n1 n2))
```

Store:  $\sigma_0$

**Memo**

Store in rt: N/A

**Contexts**

```

(let* ([id (λ (x) x)]
       [app (λ (f y) (f y))]
       [n1 (app id 1)]
       [n2 (app id 2)])
  (+ n1 n2))

```

Store:  $\sigma_1$

$f_0$	id
$y_0$	1

Memo

Store in rt:  $\sigma_1$

Contexts

$\langle (f\ y)\ \rho_1\ \sigma_1 \rangle\ (\text{let}^* (\dots [n1\ \bullet] \dots) \dots)$

```

(let* ([id (λ (x) x)]
       [app (λ (f y) (f y))]
       [n1 (app id 1)]
       [n2 (app id 2)])
  (+ n1 n2))

```

Store:  $\sigma_2$

$f_0$	id
$y_0$	1
$x_0$	1

Memo

Store in rt:  $\sigma_2$

Contexts

```

<(f y) ρ1 σ1> (let* (... [n1 •] ...) ...)
<x ρ1 σ2>      (let* (... [app (λ (f y) •)] ...) ...)

```

```

(let* ([id (λ (x) x)]
       [app (λ (f y) (f y))]
       [n1 (app id 1)]
       [n2 (app id 2)])
  (+ n1 n2))

```

Store:  $\sigma_2$

$f_0$  id

$y_0$  1

$x_0$  1

**Memo**

$\langle (f\ y)\ \rho_1\ \sigma_1 \rangle\ 1$

Store in rt:  $\sigma_2$

**Contexts**

$\langle (f\ y)\ \rho_1\ \sigma_1 \rangle\ (\text{let}^* (\dots [n1\ \bullet] \dots) \dots)$

$\langle x\ \rho_1\ \sigma_2 \rangle\ (\text{let}^* (\dots [app\ (\lambda\ (f\ y)\ \bullet)] \dots) \dots)$

```

(let* ([id (λ (x) x)]
       [app (λ (f y) (f y))]
       [n1 (app id 1)]
       [n2 (app id 2)])
  (+ n1 n2))

```

Store:  $\sigma_2$

$f_0$  id

$y_0$  1

$x_0$  1

## Memo

$\langle (f\ y)\ \rho_1\ \sigma_1 \rangle$  1

$\langle x\ \rho_1\ \sigma_2 \rangle$  1

Store in  $rt: \sigma_1$

## Contexts

$\langle (f\ y)\ \rho_1\ \sigma_1 \rangle$  (let\* (... [n1 •] ...) ...)

$\langle x\ \rho_1\ \sigma_2 \rangle$  (let\* (... [app (λ (f y) •)] ...) ...)

```

(let* ([id (λ (x) x)]
      [app (λ (f y) (f y))]
      [n1 (app id 1)]
      [n2 (app id 2)])
  (+ n1 n2))

```

Store:  $\sigma_3$

$f_0$	id
$y_0$	1
$x_0$	1
$n1_0$	1

## Memo

$\langle (f\ y)\ \rho_1\ \sigma_1 \rangle\ 1$   
 $\langle x\ \rho_1\ \sigma_2 \rangle\ 1$

Store in rt: N/A

## Contexts

$\langle (f\ y)\ \rho_1\ \sigma_1 \rangle\ (\text{let}^* (\dots [n1\ \bullet] \dots) \dots)$   
 $\langle x\ \rho_1\ \sigma_2 \rangle\ (\text{let}^* (\dots [app\ (\lambda\ (f\ y)\ \bullet)] \dots) \dots)$

```

(let* ([id (λ (x) x)]
       [app (λ (f y) (f y))]
       [n1 (app id 1)]
       [n2 (app id 2)])
  (+ n1 n2))

```

## Memo

$\langle (f\ y)\ \rho_1\ \sigma_1 \rangle\ 1$   
 $\langle x\ \rho_1\ \sigma_2 \rangle\ 1$

## Store: $\sigma_4$

$f_0, f_1$	id
$y_0$	1
$x_0$	1
$n1_0$	1
$y_1$	2

## Store in rt: $\sigma_4$

## Contexts

$\langle (f\ y)\ \rho_1\ \sigma_1 \rangle\ (\text{let}^* (\dots [n1\ \bullet] \dots) \dots)$   
 $\langle x\ \rho_1\ \sigma_2 \rangle\ (\text{let}^* (\dots [\text{app}\ (\lambda\ (f\ y)\ \bullet)] \dots) \dots)$   
 $\langle (f\ y)\ \rho_4\ \sigma_4 \rangle\ (\text{let}^* (\dots [n2\ \bullet]) \dots)$

```

(let* ([id (λ (x) x)]
      [app (λ (f y) (f y))]
      [n1 (app id 1)]
      [n2 (app id 2)])
  (+ n1 n2))

```

## Memo

$\langle (f\ y)\ \rho_1\ \sigma_1 \rangle\ 1$   
 $\langle x\ \rho_1\ \sigma_2 \rangle\ 1$

## Store: $\sigma_5$

$f_0, f_1$	id
$y_0$	1
$x_0$	1
$n1_0$	1
$y_1$	2
$x_1$	2

## Store in $rt:\sigma_5$

## Contexts

$\langle (f\ y)\ \rho_1\ \sigma_1 \rangle\ (\text{let}^* (\dots [n1\ \bullet] \dots) \dots)$   
 $\langle x\ \rho_1\ \sigma_2 \rangle\ (\text{let}^* (\dots [app\ (\lambda\ (f\ y)\ \bullet)] \dots) \dots)$   
 $\langle (f\ y)\ \rho_4\ \sigma_4 \rangle\ (\text{let}^* (\dots [n2\ \bullet]) \dots)$   
 $\langle x\ \rho_5\ \sigma_5 \rangle\ (\text{let}^* (\dots [app\ (\lambda\ (f\ y)\ \bullet)] \dots) \dots)$



```

(let* ([id (λ (x) x)]
       [app (λ (f y) (f y))]
       [n1 (app id 1)]
       [n2 (app id 2)])
  (+ n1 n2))

```

Store:  $\sigma_5$

$f_0, f_1$  id

$y_0$  1

$x_0$  1

$n1_0$  1

$y_1$  2

$x_1$  2

## Memo

$\langle (f\ y)\ \rho_1\ \sigma_1 \rangle$  1

$\langle x\ \rho_1\ \sigma_2 \rangle$  1

$\langle (f\ y)\ \rho_4\ \sigma_4 \rangle$  2

Store in rt:  $\sigma_5$

## Contexts

$\langle (f\ y)\ \rho_1\ \sigma_1 \rangle$  (let\* (... [n1 •] ...) ...)

$\langle x\ \rho_1\ \sigma_2 \rangle$  (let\* (... [app (λ (f y) •)] ...) ...)

$\langle (f\ y)\ \rho_4\ \sigma_4 \rangle$  (let\* (... [n2 •] ...) ...)

$\langle x\ \rho_5\ \sigma_5 \rangle$  (let\* (... [app (λ (f y) •)] ...) ...)

```

(let* ([id (λ (x) x)]
      [app (λ (f y) (f y))]
      [n1 (app id 1)]
      [n2 (app id 2)])
  (+ n1 n2))

```

Store:  $\sigma_5$

$f_0, f_1$  id

$y_0$  1

$x_0$  1

$n1_0$  1

$y_1$  2

$x_1$  2

## Memo

$\langle (f\ y)\ \rho_1\ \sigma_1 \rangle$  1

$\langle x\ \rho_1\ \sigma_2 \rangle$  1

$\langle (f\ y)\ \rho_4\ \sigma_4 \rangle$  2

$\langle x\ \rho_5\ \sigma_5 \rangle$  2

Store in rt:  $\sigma_4$

## Contexts

$\langle (f\ y)\ \rho_1\ \sigma_1 \rangle$  (let\* (... [n1 •] ...) ...)

$\langle x\ \rho_1\ \sigma_2 \rangle$  (let\* (... [app (λ (f y) •)] ...) ...)

$\langle (f\ y)\ \rho_4\ \sigma_4 \rangle$  (let\* (... [n2 •]) ...)

$\langle x\ \rho_5\ \sigma_5 \rangle$  (let\* (... [app (λ (f y) •)] ...) ...)

```

(let* ([id (λ (x) x)]
       [app (λ (f y) (f y))]
       [n1 (app id 1)]
       [n2 (app id 2)])
  (+ n1 n2))

```

## Memo

$\langle (f\ y)\ \rho_1\ \sigma_1 \rangle\ 1$   
 $\langle x\ \rho_1\ \sigma_2 \rangle\ 1$   
 $\langle (f\ y)\ \rho_4\ \sigma_4 \rangle\ 2$   
 $\langle x\ \rho_5\ \sigma_5 \rangle\ 2$

## Contexts

$\langle (f\ y)\ \rho_1\ \sigma_1 \rangle\ (\text{let}^* (\dots [n1\ \bullet] \dots) \dots)$   
 $\langle x\ \rho_1\ \sigma_2 \rangle\ (\text{let}^* (\dots [\text{app}\ (\lambda\ (f\ y)\ \bullet)] \dots) \dots)$   
 $\langle (f\ y)\ \rho_4\ \sigma_4 \rangle\ (\text{let}^* (\dots [n2\ \bullet]) \dots)$   
 $\langle x\ \rho_5\ \sigma_5 \rangle\ (\text{let}^* (\dots [\text{app}\ (\lambda\ (f\ y)\ \bullet)] \dots) \dots)$

## Store: $\sigma_6$

$f_0, f_1$	id
$y_0$	1
$x_0$	1
$n1_0$	1
$y_1$	2
$x_1$	2
$n2_0$	2

Store in rt: N/A

We can extend the analogy

# We can extend the analogy

Everyone's favorite: delimited composable control

$E[(\text{reset } F[(\text{shift } k \ e)])]$

$\mapsto$

$E[e\{k := (\lambda (x) (\text{reset } F[x]))\}]$

$E[(\text{reset } F[(\text{shift } k \ e)])]$

$\mapsto$

$E[e\{k := (\lambda (x) (\text{reset } F[x]))\}]$

(F doesn't contain a reset)

# Metacontinuations


$$\langle (\text{reset } e), \rho, \sigma, \kappa, C \rangle \mapsto \langle e, \rho, \sigma, \text{end}, \kappa \circ C \rangle$$



# Metacontinuations

$$\begin{aligned} & \langle (\text{reset } e), \rho, \sigma, \kappa, C \rangle \\ & \quad \mapsto \\ & \quad \langle e, \rho, \sigma, \text{end}, \kappa \circ C \rangle \\ & \langle (\text{shift } k \ e), \rho, \sigma, \kappa, C \rangle \\ & \quad \mapsto \\ & \langle e, \rho[k \mapsto a], \sigma \sqcup [a \mapsto \{\kappa\}], \text{end}, C \rangle \end{aligned}$$

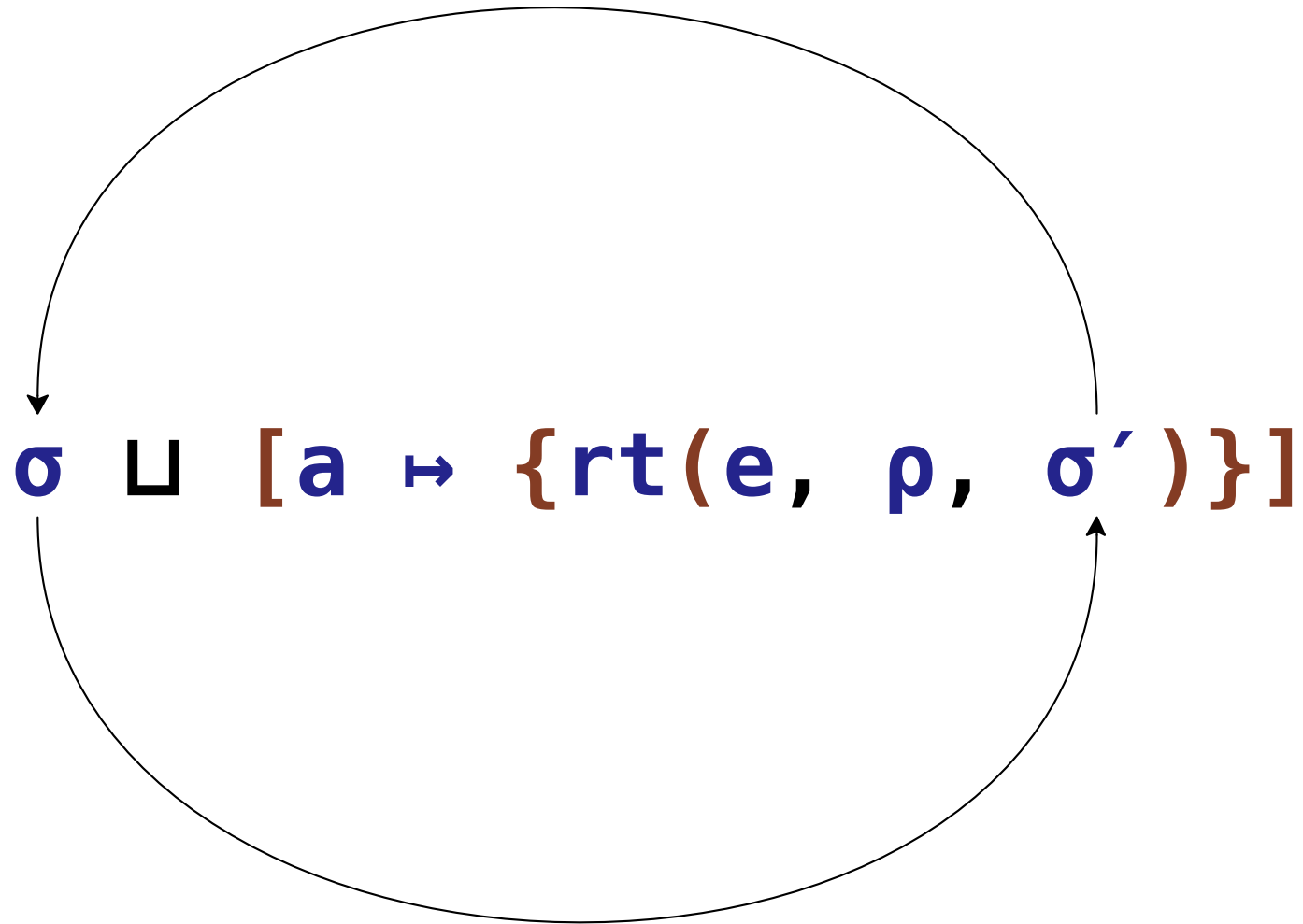
# Metacontinuations

$$\begin{aligned} & \langle (\text{reset } e), \rho, \sigma, \kappa, C \rangle \\ & \quad \mapsto \\ & \quad \langle e, \rho, \sigma, \text{end}, \kappa \circ C \rangle \\ & \langle (\text{shift } k \ e), \rho, \sigma, \kappa, C \rangle \\ & \quad \mapsto \\ & \langle e, \rho[k \mapsto a], \sigma \sqcup [a \mapsto \{\kappa\}], \text{end}, C \rangle \end{aligned}$$


Naive doesn't cut it

$$\sigma \sqcup [a \mapsto \{rt(e, \rho, \sigma')\}]$$

Naive doesn't cut it



```

01 Summary, Callers, TCallers, EntriesEsc, Escapes, Final  $\leftarrow \emptyset$ 
02 Seen, W  $\leftarrow \{(\tilde{I}(pr), \tilde{I}(pr))\}$ 
03 while W  $\neq \emptyset$ 
04   remove  $(\zeta_1, \zeta_2)$  from W
05   switch  $\zeta_2$ 
06     case  $\zeta_2$  of Entry
07       for the  $\zeta_3$  in succ( $\zeta_2$ ), Propagate( $\zeta_1, \zeta_3$ , false)
08        $\zeta_2$  of the form  $([\lambda_l(u\ k)\ call], \hat{d}, h)$ 
09         if  $H_l(k)$  then
10           insert  $\zeta_2$  in EntriesEsc
11           for each  $\zeta_3$  in Escapes that calls  $k$ , Propagate( $\zeta_2, \zeta_3$ , true)
12     case  $\zeta_2$  of CApply, Inner-CEval
13       for the  $\zeta_3$  in succ( $\zeta_2$ ), Propagate( $\zeta_1, \zeta_3$ , false)
14     case  $\zeta_2$  of Call
15       for each  $\zeta_3$  in succ( $\zeta_2$ )
16         Propagate( $\zeta_3, \zeta_1$ , false)
17         insert  $(\zeta_1, \zeta_2, \zeta_3)$  in Callers
18         for each  $(\zeta_3, \zeta_4)$  in Summary, Update( $\zeta_1, \zeta_2, \zeta_3, \zeta_4$ )
19     case  $\zeta_2$  of Exit-Ret
20       if  $\zeta_1 = \tilde{I}(pr)$  then Final( $\zeta_2$ )
21       else
22         insert  $(\zeta_1, \zeta_2)$  in Summary
23         for each  $(\zeta_3, \zeta_4, \zeta_5)$  in Callers, Update( $\zeta_3, \zeta_4, \zeta_5, \zeta_2$ )
24         for each  $(\zeta_3, \zeta_4, \zeta_5)$  in TCallers, Propagate( $\zeta_3, \zeta_2$ , false)
25     case  $\zeta_2$  of Exit-Esc
26       if  $(\zeta_1, \zeta_2)$  not in Summary then
27         insert  $\zeta_2$  in Escapes
28          $\zeta_2$  of the form  $([\lambda_l(k\ e)^{\gamma}], tf, h)$ 
29         for each  $\zeta_3$  in EntriesEsc over  $def_{\lambda}(k)$ , Propagate( $\zeta_3, \zeta_2$ , true)
30       else if  $\zeta_1 = \tilde{I}(pr)$  then Final( $\zeta_2$ )
31       else
32         for each  $(\zeta_3, \zeta_4, \zeta_5)$  in Callers, Update( $\zeta_3, \zeta_4, \zeta_5, \zeta_2$ )
33         for each  $(\zeta_3, \zeta_4, \zeta_5)$  in TCallers, Propagate( $\zeta_3, \zeta_2$ , true)
34     case  $\zeta_2$  of Exit-TC
35       for each  $\zeta_3$  in succ( $\zeta_2$ )
36         Propagate( $\zeta_3, \zeta_1$ , false)
37         insert  $(\zeta_1, \zeta_2, \zeta_3)$  in TCallers
38       S  $\leftarrow \emptyset$ 
39       for each  $(\zeta_3, \zeta_4)$  in Summary
40         insert  $(\zeta_1, \zeta_4)$  in S
41         Propagate( $\zeta_3, \zeta_4$ , false)
42       Summary  $\leftarrow \text{Summary} \cup S$ 
43   Propagate( $\zeta_1, \zeta_2$ , esc)  $\triangleq$ 
44     if esc then insert  $(\zeta_1, \zeta_2)$  in Summary
45     if  $(\zeta_1, \zeta_2)$  not in Seen then insert  $(\zeta_1, \zeta_2)$  in Seen and W
46   Update( $\zeta_1, \zeta_2, \zeta_3, \zeta_4$ )  $\triangleq$ 
47      $\zeta_1$  of the form  $([\lambda_{l_1}(u_1\ k_1)\ call_1], \hat{d}_1, h_1)$ 
48      $\zeta_2$  of the form  $([\lambda_{l_2}(u_2\ k_2)\ call_2], \hat{d}_2, h_2)$ 
49      $\zeta_3$  of the form  $([\lambda_{l_3}(u_3\ k_3)\ call_3], \hat{d}_3, h_3)$ 
50      $\zeta_4$  of the form  $([\lambda_{l_4}(u_4\ k_4)\ call_4], \hat{d}_4, h_4)$ 
51      $\hat{d} \leftarrow \hat{A}_u(e_4, \gamma_4, tf_4, h_4)$ 
52      $tf \leftarrow \begin{cases} tf_2[f \mapsto \{[\lambda_{l_3}(u_3\ k_3)\ call_3] \}] & S_7(l_2, f) \\ tf_2 & H_7(l_2, f) \vee Lam_7(f) \end{cases}$ 
53      $\zeta \leftarrow ([\lambda_{l_2}(u_2)\ call_2], \hat{d}, tf, h_4)$ 
54     Propagate( $\zeta_1, \zeta$ , false)
55   Final( $\zeta$ )  $\triangleq$ 
56      $\zeta$  of the form  $([\lambda_l(k\ e)^{\gamma}], tf, h)$ 
57     insert  $(halt, \hat{A}_u(e, \gamma, tf, h), \emptyset, h)$  in Final

```

Figure 8: CFA2 workset algorithm

```

01 Summary, Callers, TCallers, EntriesEsc, Escapes, Final  $\leftarrow \emptyset$ 
02 Seen, W  $\leftarrow \{(\tilde{I}(pr), \tilde{I}(pr))\}$ 
03 while W  $\neq \emptyset$ 
04   remove  $(\zeta_1, \zeta_2)$  from W
05   switch  $\zeta_2$ 
06     case  $\zeta_2$  of Entry
07       for the  $\zeta_3$  in succ( $\zeta_2$ ), Propagate( $\zeta_1, \zeta_3$ , false)
08        $\zeta_2$  of the form  $([\lambda r(u\ k)\ call], \hat{d}, h)$ 
09       if  $H_r(k)$  then
10         insert  $\zeta_2$  in EntriesEsc
11         for each  $\zeta_3$  in Escapes that calls  $k$ , Propagate( $\zeta_2, \zeta_3$ , true)
12     case  $\zeta_2$  of CApply, Inner-CEval
13       for the  $\zeta_3$  in succ( $\zeta_2$ ), Propagate( $\zeta_1, \zeta_3$ , false)
14     case  $\zeta_2$  of Call
15       for each  $\zeta_3$  in succ( $\zeta_2$ )
16         Propagate( $\zeta_3, \zeta_1$ , false)
17         insert  $(\zeta_1, \zeta_2, \zeta_3)$  in Callers
18         for each  $(\zeta_3, \zeta_4)$  in Summary, Update( $\zeta_1, \zeta_2, \zeta_3, \zeta_4$ )
19     case  $\zeta_2$  of Exit-Ret
20       if  $\zeta_1 = \tilde{I}(pr)$  then Final( $\zeta_2$ )
21       else
22         insert  $(\zeta_1, \zeta_2)$  in Summary
23         for each  $(\zeta_3, \zeta_4, \zeta_5)$  in Callers, Update( $\zeta_3, \zeta_4, \zeta_5, \zeta_2$ )
24         for each  $(\zeta_3, \zeta_4, \zeta_5)$  in TCallers, Propagate( $\zeta_3, \zeta_2$ , false)
25     case  $\zeta_2$  of Exit-Esc
26       if  $(\zeta_1, \zeta_2)$  not in Summary then
27         insert  $\zeta_2$  in Escapes
28          $\zeta_2$  of the form  $([k\ e]^\gamma, tf, h)$ 
29         for each  $\zeta_3$  in EntriesEsc over  $def_\lambda(k)$ , Propagate( $\zeta_3, \zeta_2$ , true)
30       else if  $\zeta_1 = \tilde{I}(pr)$  then Final( $\zeta_2$ )
31       else
32         for each  $(\zeta_3, \zeta_4, \zeta_5)$  in Callers, Update( $\zeta_3, \zeta_4, \zeta_5, \zeta_2$ )
33         for each  $(\zeta_3, \zeta_4, \zeta_5)$  in TCallers, Propagate( $\zeta_3, \zeta_2$ , true)
34     case  $\zeta_2$  of Exit-TC
35       for each  $\zeta_3$  in succ( $\zeta_2$ )
36         Propagate( $\zeta_3, \zeta_1$ , false)
37         insert  $(\zeta_1, \zeta_2, \zeta_3)$  in TCallers
38       S  $\leftarrow \emptyset$ 
39       for each  $(\zeta_3, \zeta_4)$  in Summary
40         insert  $(\zeta_1, \zeta_4)$  in S
41         Propagate( $\zeta_3, \zeta_4$ , false)
42       Summary  $\leftarrow \text{Summary} \cup S$ 
43   Propagate( $\zeta_1, \zeta_2$ , esc)  $\triangleq$ 
44     if esc then insert  $(\zeta_1, \zeta_2)$  in Summary
45     if  $(\zeta_1, \zeta_2)$  not in Seen then insert  $(\zeta_1, \zeta_2)$  in Seen and W
46   Update( $\zeta_1, \zeta_2, \zeta_3, \zeta_4$ )  $\triangleq$ 
47      $\zeta_1$  of the form  $([\lambda_{l_1}(u_1\ k_1)\ call_1], \hat{d}_1, h_1)$ 
48      $\zeta_2$  of the form  $([\lambda_{l_2}(u_2\ k_2)\ call_2], \hat{d}_2, h_2)$ 
49      $\zeta_3$  of the form  $([\lambda_{l_3}(u_3\ k_3)\ call_3], \hat{d}_3, h_3)$ 
50      $\zeta_4$  of the form  $([\lambda_{l_4}(u_4\ k_4)\ call_4], \hat{d}_4, h_4)$ 
51      $\hat{d} \leftarrow \hat{A}_u(e_4, \gamma_4, tf_4, h_4)$ 
52      $tf \leftarrow \begin{cases} tf_2[f \mapsto \{[\lambda_{l_3}(u_3\ k_3)\ call_3]\}] & S_7(l_2, f) \\ tf_2 & H_7(l_2, f) \vee Lam_7(f) \end{cases}$ 
53      $\zeta \leftarrow ([\lambda_{l_2}(u_2)\ call_2], \hat{d}, tf, h_4)$ 
54     Propagate( $\zeta_1, \zeta$ , false)
55   Final( $\zeta$ )  $\triangleq$ 
56      $\zeta$  of the form  $([k\ e]^\gamma, tf, h)$ 
57     insert  $(halt, \hat{A}_u(e, \gamma, tf, h), \emptyset, h)$  in Final

```

Figure 8: CFA2 workset algorithm

# The AAM way

Break circularity with indirection

Approximation tuning: **alloc**

$$\sigma \sqcup [a \mapsto \{rt(e, \rho, \sigma')\}]$$



## Approximation tuning: **alloc**

$$\sigma \sqcup [a \mapsto \{\text{rt}(e, \rho, \sigma')\}]$$



$$\sigma \sqcup [a \mapsto \{\text{rt}(e, \rho, a)\}]$$

$$\Xi \sqcup [a \mapsto \{\sigma'\}]$$

## Approximation tuning: **alloc**

$\sigma \sqcup [a \mapsto \{\text{rt}(e, \rho, \sigma')\}]$



$\sigma \sqcup [a \mapsto \{\text{rt}(e, \rho, a)\}]$

$\Xi \sqcup [a \mapsto \{\sigma'\}]$

**K**

# Metacontinuations

$C ::= \text{halt} \mid k \circ C$

# Metacontinuations

$C ::= \text{halt} \mid \kappa \circ C$



$C ::= \text{halt} \mid \#(\text{ctx})$

# Metacontinuations

$$C ::= \text{halt} \mid \kappa \circ C$$

$$C ::= \text{halt} \mid \#(\text{ctx})$$
$$\text{ctx} ::= \langle e, \rho, \sigma \rangle \mid \langle \kappa, v, \sigma \rangle$$

# The new context

- Prompts treated just like function calls

# The new context

- Prompts treated just like function calls

`call` $\langle \kappa, v, \sigma, \kappa, C \rangle$



$\langle \kappa, \#(ctx), v, \sigma \rangle$

with  $ctx = (\kappa, v, \sigma)$

$\Xi' = \Xi \sqcup [ctx \mapsto \{\langle \kappa, C \rangle\}]$

# Fix up memoization and returns

Approximate contexts mean all instantiations from  $\Xi$



# Fix up memoization and returns

Approximate contexts mean all instantiations from  $\Xi$

$\langle \text{rt}(\mathbf{e}, \mathbf{p}, \mathbf{a}), \mathbf{C}, \mathbf{v} \rangle$

# Fix up memoization and returns

Approximate contexts mean all instantiations from  $\Xi$

$$\langle \text{rt}(\mathbf{e}, \mathbf{p}, \mathbf{a}), \mathbf{C}, \mathbf{v} \rangle \mapsto \langle \mathbf{k}, \mathbf{C}, \mathbf{v} \rangle$$

where  $\mathbf{k} \in \bigcup \{ \Xi(\mathbf{e}, \mathbf{p}, \sigma) : \sigma \in \Xi(\mathbf{a}) \}$

# Fix up memoization and returns

Approximate contexts mean all instantiations from  $\Xi$

$$\langle \text{rt}(\mathbf{e}, \mathbf{\rho}, \mathbf{a}), \mathbf{C}, \mathbf{v} \rangle \mapsto \langle \mathbf{\kappa}, \mathbf{C}, \mathbf{v} \rangle$$

$$\text{where } \mathbf{\kappa} \in \bigcup \{ \Xi(\mathbf{e}, \mathbf{\rho}, \mathbf{\sigma}) : \mathbf{\sigma} \in \Xi(\mathbf{a}) \}$$

$$\mathbf{M}' = \mathbf{M} \sqcup \bigsqcup_{\mathbf{\sigma} \in \Xi(\mathbf{a})} [(\mathbf{e}, \mathbf{\rho}, \mathbf{\sigma}) \mapsto \{\mathbf{v}\}]$$

# To Conclude

- Design: Model abstract mechanisms concretely

# To Conclude

- Design: Model abstract mechanisms concretely
- Pushdown: Memo and local continuation tables

# To Conclude

- Design: Model abstract mechanisms concretely
- Pushdown: Memo and local continuation tables
- Works for control operators / GC (not shown)

# To Conclude

- Design: Model abstract mechanisms concretely
- Pushdown: Memo and local continuation tables
- Works for control operators / GC (not shown)

<https://github.com/ianj/pushdown-shift-reset>

# Thank you

# Garbage collection

Read root addresses of  $\kappa$  through  $\Xi$

$$\mathcal{R}(\text{rt}(\mathbf{e}, \boldsymbol{\rho}, \boldsymbol{\sigma})) = \bigcup \{ \mathcal{R}(\kappa) : \kappa \in \Xi(\mathbf{e}, \boldsymbol{\rho}, \boldsymbol{\sigma}) \}$$