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]

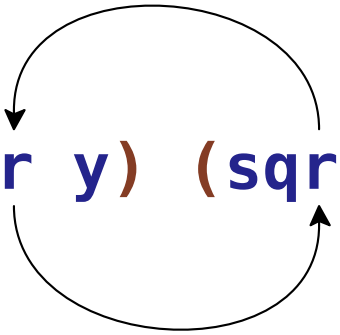
- Start: Concrete machine semantics
- Simple transforms: Put semantics in right form
- Analysis: finitize addresses

```
(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 ⟵ ∅
02   Seen, W ⟵ {(Ĩ(pr), ς̃̂(pr))}
03   while W ≠ ∅
04     remove (ς̃1, ς̃2) from W
05     switch ς̃2
06       case ς̃2 of Entry, CApply, Inner-CEval
07         for each ς̃3 in succ(ς̃2) Propagate(ς̃1, ς̃3)
08       case ς̃2 of Call
09         for each ς̃3 in succ(ς̃2)
10           Propagate(ς̃3, ς̃3)
11           insert (ς̃1, ς̃2, ς̃3) in Callers
12           for each (ς̃3, ς̃4) in Summary  Update(ς̃1, ς̃2, ς̃3, ς̃4)
13       case ς̃2 of Exit-CEval
14         if ς̃1 = Ĩ(pr) then
15           Final(ς̃2)
16         else
17           insert (ς̃1, ς̃2) in Summary
18           for each (ς̃3, ς̃4, ς̃1) in Callers  Update(ς̃3, ς̃4, ς̃1, ς̃2)
19           for each (ς̃3, ς̃4, ς̃1) in TCallers Propagate(ς̃3, ς̃2)
20       case ς̃2 of Exit-TC
21         for each ς̃3 in succ(ς̃2)
22           Propagate(ς̃3, ς̃3)
23           insert (ς̃1, ς̃2, ς̃3) in TCallers
24           for each (ς̃3, ς̃4) in Summary Propagate(ς̃1, ς̃4)

     Propagate(ς̃1, ς̃2) ≜
25     if (ς̃1, ς̃2) not in Seen then insert (ς̃1, ς̃2) in Seen and W

     Update(ς̃1, ς̃2, ς̃3, ς̃4) ≜
26     ς̃1 of the form  (⟦(λ_l1 (u1 k1) call1)⟧ , d̂1, h1)
27     ς̃2 of the form  (⟦(f e2 (λ_γ2 (u2) call2))^l2⟧, tf2, h2)
28     ς̃3 of the form  (⟦(λ_l3 (u3 k3) call3)⟧ , d̂3, h2)
29     ς̃4 of the form  (⟦(k4 e4)^γ4⟧, tf4, h4)
30     d̂ ⟵  Ã_u(e4, γ4, tf4, h4)
31     tf ⟵  { tf2[f ↦ {⟦(λ_l3 (u3 k3) call3)⟧}]   S?(l2, f)
             { tf2                                  H?(l2, f) ∨ Lam?(f)
32     ς̃ ⟵  (⟦(λ_γ2 (u2) call2)⟧, d̂, tf, h4)
33     Propagate(ς̃1, ς̃)

     Final(ς̃) ≜
34     ς̃ of the form  (⟦(k e)^γ⟧, tf, h)
35     insert  (halt, Ã_u(e, γ, tf, h), ∅, h) in Final
```

Figure 8: CFA2 workset algorithm

Pushdown Higher-Order Flow Analysis

```
01   Summary, Callers, TCallers, Final ⟵ ∅
02   Seen, W ⟵ {(Ĩ(pr), Ĩ(pr))}
03   while W ≠ ∅
04     remove (ζ̃1, ζ̃2) from W
05     switch ζ̃2
06       case ζ̃2 of Entry, CApply, Inner-CEval
07         for each ζ̃3 in succ(ζ̃2) Propagate(ζ̃1, ζ̃3)
08       case ζ̃2 of Call
09         for each ζ̃3 in succ(ζ̃2)
10           Propagate(ζ̃3, ζ̃3)
11           insert (ζ̃1, ζ̃2, ζ̃3) in Callers
12           for each (ζ̃3, ζ̃4) in Summary  Update(ζ̃1, ζ̃2, ζ̃3, ζ̃4)
13       case ζ̃2 of Exit-CEval
14         if ζ̃1 = Ĩ(pr) then
15           Final(ζ̃2)
16         else
17           insert (ζ̃1, ζ̃2) in Summary
18           for each (ζ̃3, ζ̃4, ζ̃1) in Callers  Update(ζ̃3, ζ̃4, ζ̃1, ζ̃2)
19           for each (ζ̃3, ζ̃4, ζ̃1) in TCallers Propagate(ζ̃3, ζ̃2)
20       case ζ̃2 of Exit-TC
21         for each ζ̃3 in succ(ζ̃2)
22           Propagate(ζ̃3, ζ̃3)
23           insert (ζ̃1, ζ̃2, ζ̃3) in TCallers
24           for each (ζ̃3, ζ̃4) in Summary Propagate(ζ̃1, ζ̃4)

     Propagate(ζ̃1, ζ̃2) ≜
25     if (ζ̃1, ζ̃2) not in Seen then insert (ζ̃1, ζ̃2) in Seen and W

     Update(ζ̃1, ζ̃2, ζ̃3, ζ̃4) ≜
26     ζ̃1 of the form ([(λ_l1 (u1 k1) call1)] , d̂1, h1)
27     ζ̃2 of the form ([(f e2 (λ_γ2 (u2) call2))^l2], tf2, h2)
28     ζ̃3 of the form ([(λ_l3 (u3 k3) call3)] , d̂3, h2)
29     ζ̃4 of the form ([(k4 e4)^γ4], tf4, h4)
30     d̂ ⟵ Ã_u(e4, γ4, tf4, h4)
31     tf ⟵ { tf2[f ↦ {[(λ_l3 (u3 k3) call3)]}]   S?(l2, f)
            { tf2                                  H?(l2, f) ∨ Lam?(f)
32     ζ̃ ⟵ ([(λ_γ2 (u2) call2)], d̂, tf, h4)
33     Propagate(ζ̃1, ζ̃)

     Final(ζ̃) ≜
34     ζ̃ of the form ([(k e)^γ], tf, h)
35     insert (halt, Ã_u(e, γ, tf, h), ∅, h) in Final
```

Figure 8: CFA2 workset algorithm

$$
\begin{aligned}
\mathcal{F}'(M) &= f, \text{ where} \\
M &= (Q, \Gamma, \delta, q_0) \\
f(G, G_\epsilon, \Delta G, \Delta H) &= (G', G'_\epsilon, \Delta G', \Delta H' - H), \text{ where} \\
(S, \Gamma, E, s_0) &= G \\
(S, H) &= G_\epsilon \\
(\Delta S, \Delta E) &= \Delta G \\
(\Delta E_0, \Delta H_0) &= \bigcup_{s \in \Delta S} sprout_M(s) \\
(\Delta E_1, \Delta H_1) &= \bigcup_{(s,\gamma_+,s') \in \Delta E} addPush_M(G, G_\epsilon)(s, \gamma_+, s') \\
(\Delta E_2, \Delta H_2) &= \bigcup_{(s,\gamma_-,s') \in \Delta E} addPop_M(G, G_\epsilon)(s, \gamma_-, s') \\
(\Delta E_3, \Delta H_3) &= \bigcup_{(s,\epsilon,s') \in \Delta E} addEmpty_M(G, G_\epsilon)(s, s') \\
(\Delta E_4, \Delta H_4) &= \bigcup_{(s,s') \in \Delta H} addEmpty_M(G, G_\epsilon)(s, s') \\
S' &= S \cup \Delta S \\
E' &= E \cup \Delta E \\
H' &= H \cup \Delta H \\
\Delta E' &= \Delta E_0 \cup \Delta E_1 \cup \Delta E_2 \cup \Delta E_3 \cup \Delta E_4 \\
\Delta S' &= \{s' : (s, g, s') \in \Delta E'\} \\
\Delta H' &= \Delta H_0 \cup \Delta H_1 \cup \Delta H_2 \cup \Delta H_3 \cup \Delta H_4 \\
G' &= (S \cup \Delta S, \Gamma, E', q_0) \\
G'_\epsilon &= (S', H') \\
\Delta G' &= (\Delta S' - S', \Delta E' - E').
\end{aligned}
$$

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 ⟵ {(Ĩ(pr), Ĩ(pr))}
03  while W ≠ ∅
04    remove (ζ̃1, ζ̃2) from W
05    switch ζ̃2
06      case ζ̃2 of Entry, CApply, Inner-CEval
07        for each ζ̃3 in succ(ζ̃2) Propagate(ζ̃1, ζ̃3)
08      case ζ̃2 of Call
09        for each ζ̃3 in succ(ζ̃2)
10          Propagate(ζ̃3, ζ̃3)
11          insert (ζ̃1, ζ̃2, ζ̃3) in Callers
12          for each (ζ̃3, ζ̃4) in Summary  Update(ζ̃1, ζ̃2, ζ̃3, ζ̃4)
13      case ζ̃2 of Exit-CEval
14        if ζ̃1 = Ĩ(pr) then
15          Final(ζ̃2)
16        else
17          insert (ζ̃1, ζ̃2) in Summary
18          for each (ζ̃3, ζ̃4, ζ̃1) in Callers  Update(ζ̃3, ζ̃4, ζ̃1, ζ̃2)
19          for each (ζ̃3, ζ̃4, ζ̃1) in TCallers Propagate(ζ̃3, ζ̃2)
20      case ζ̃2 of Exit-TC
21        for each ζ̃3 in succ(ζ̃2)
22          Propagate(ζ̃3, ζ̃3)
23          insert (ζ̃1, ζ̃2, ζ̃3) in TCallers
24          for each (ζ̃3, ζ̃4) in Summary Propagate(ζ̃1, ζ̃4)

    Propagate(ζ̃1, ζ̃2) ≜
25    if (ζ̃1, ζ̃2) not in Seen then insert (ζ̃1, ζ̃2) in Seen and W

    Update(ζ̃1, ζ̃2, ζ̃3, ζ̃4) ≜
26    ζ̃1 of the form  (⟦(λ_l1 (u1 k1) call1)⟧ , d̂1, h1)
27    ζ̃2 of the form  (⟦(f e2 (λ_γ2 (u2) call2))^l2⟧, tf2, h2)
28    ζ̃3 of the form  (⟦(λ_l3 (u3 k3) call3)⟧ , d̂3, h2)
29    ζ̃4 of the form  (⟦(k4 e4)^γ4⟧, tf4, h4)
30    d̂ ⟵  Ã_u(e4, γ4, tf4, h4)
31    tf ⟵  { tf2[f ↦ {⟦(λ_l3 (u3 k3) call3)⟧}]   S?(l2, f)
            { tf2                                  H?(l2, f) ∨ Lam?(f)
32    ζ̃ ⟵  (⟦(λ_γ2 (u2) call2)⟧, d̂, tf, h4)
33    Propagate(ζ̃1, ζ̃)

    Final(ζ̃) ≜
34    ζ̃ of the form  (⟦(k e)^γ⟧, tf, h)
35    insert  (halt, Ã_u(e, γ, tf, h), ∅, h) in Final
```

Figure 8: CFA2 workset algorithm

$$sprout_{(Q,\Gamma,\delta)}(s) = (\Delta E, \Delta H), \text{where}$$

$$\Delta E = \left\{ s \overset{\epsilon}{\mapsto} q : s \overset{\epsilon}{\mapsto} q \in \delta \right\} \cup \left\{ s \overset{\gamma_+}{\mapsto} q : s \overset{\gamma_+}{\mapsto} q \in \delta \right\}$$

$$\Delta H = \left\{ s \mapsto q : s \overset{\epsilon}{\mapsto} q \in \delta \right\}.$$

$$(\Delta S, \Delta E) = \Delta G$$

$$(\Delta E_0, \Delta H_0) = \bigcup_{s \in \Delta S} sprout_M(s)$$

$$(\Delta E_1, \Delta H_1) = \bigcup_{(s,\gamma_+,s') \in \Delta E} addPush_M(G, G_\epsilon)(s, \gamma_+, s')$$

$$(\Delta E_2, \Delta H_2) = \bigcup_{(s,\gamma_-,s') \in \Delta E} addPop_M(G, G_\epsilon)(s, \gamma_-, s')$$

$$(\Delta E_3, \Delta H_3) = \bigcup_{(s,\epsilon,s') \in \Delta E} addEmpty_M(G, G_\epsilon)(s, s')$$

$$(\Delta E_4, \Delta H_4) = \bigcup_{(s,s') \in \Delta H} addEmpty_M(G, G_\epsilon)(s, s')$$

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

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

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

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

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

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

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

$$G'_\epsilon = (S', H')$$

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

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 ⟵ {(Ĩ(pr), Ĩ(pr))}
03   while W ≠ ∅
04     remove (ζ̃1, ζ̃2) from W
05     switch ζ̃2
06       case ζ̃2 of Entry, CApply, Inner-CEval
07         for each ζ̃3 in succ(ζ̃2) Propagate(ζ̃1, ζ̃3)
08       case ζ̃2 of Call
09         for each ζ̃3 in succ(ζ̃2)
10           Propagate(ζ̃3, ζ̃3)
11           insert (ζ̃1, ζ̃2, ζ̃3) in Callers
12           for each (ζ̃3, ζ̃4) in Summary  Update(ζ̃1, ζ̃2, ζ̃3, ζ̃4)
13       case ζ̃2 of Exit-CEval
14         if ζ̃1 = Ĩ(pr) then
15           Final(ζ̃2)
16         else
17           insert (ζ̃1, ζ̃2) in Summary
18           for each (ζ̃3, ζ̃4, ζ̃1) in Callers  Update(ζ̃3, ζ̃4, ζ̃1, ζ̃2)
19           for each (ζ̃3, ζ̃4, ζ̃1) in TCallers Propagate(ζ̃3, ζ̃2)
20       case ζ̃2 of Exit-TC
21         for each ζ̃3 in succ(ζ̃2)
22           Propagate(ζ̃3, ζ̃3)
23           insert (ζ̃1, ζ̃2, ζ̃3) in TCallers
24           for each (ζ̃3, ζ̃4) in Summary Propagate(ζ̃1, ζ̃4)

     Propagate(ζ̃1, ζ̃2) ≜
25     if (ζ̃1, ζ̃2) not in Seen then insert (ζ̃1, ζ̃2) in Seen and W

     Update(ζ̃1, ζ̃2, ζ̃3, ζ̃4) ≜
26     ζ̃1 of the form  (⟦(λ_l1 (u1 k1) call1)⟧ , d̂1, h1)
27     ζ̃2 of the form  (⟦(f e2 (λ_γ2 (u2) call2))^l2⟧, tf2, h2)
28     ζ̃3 of the form  (⟦(λ_l3 (u3 k3) call3)⟧ , d̂3, h2)
29     ζ̃4 of the form  (⟦(k4 e4)^γ4⟧, tf4, h4)
30     d̂ ⟵  Ã_u(e4, γ4, tf4, h4)
31     tf ⟵  { tf2[f ↦ {⟦(λ_l3 (u3 k3) call3)⟧}]   S?(l2, f)
             { tf2                                  H?(l2, f) ∨ Lam?(f)
32     ζ̃ ⟵  (⟦(λ_γ2 (u2) call2)⟧, d̂, tf, h4)
33     Propagate(ζ̃1, ζ̃)

     Final(ζ̃) ≜
34     ζ̃ of the form  (⟦(k e)^γ⟧, tf, h)
35     insert  (halt, Ã_u(e, γ, tf, h), ∅, h) in Final
```

Figure 8: CFA2 workset algorithm

$$sprout_{(Q,\Gamma,\delta)}(s) = (\Delta E, \Delta H), \text{ where}$$

$$\Delta E = \left\{ s \overset{\epsilon}{\mapsto} q : s \overset{\epsilon}{\mapsto} q \in \delta \right\} \cup \left\{ s \overset{\gamma+}{\mapsto} q : s \overset{\gamma+}{\mapsto} q \in \delta \right\}$$

$$\Delta H = \left\{ s \mapsto q : s \overset{\epsilon}{\mapsto} q \in \delta \right\}.$$

$$addPush_{(Q,\Gamma,\delta)}(G, G_\epsilon)(s \overset{\gamma+}{\mapsto} q) = (\Delta E, \Delta H), \text{ where}$$

$$\Delta E = \left\{ q' \overset{\gamma-}{\mapsto} q'' : q' \in \overrightarrow{G}_\epsilon[q] \text{ and } q' \overset{\gamma-}{\mapsto} q'' \in \delta \right\}$$

$$\Delta H = \left\{ s \mapsto q'' : q' \in \overrightarrow{G}_\epsilon[q] \text{ and } q' \overset{\gamma-}{\mapsto} q'' \in \delta \right\}.$$

$$\bigcup_{(s,\gamma_-,s') \in \Delta E}$$

$$(\Delta E_3, \Delta H_3) = \bigcup_{(s,\epsilon,s') \in \Delta E} addEmpty_M(G, G_\epsilon)(s, s')$$

$$(\Delta E_4, \Delta H_4) = \bigcup_{(s,s') \in \Delta H} addEmpty_M(G, G_\epsilon)(s, s')$$

$$S' = S \cup \Delta S$$
$$E' = E \cup \Delta E$$
$$H' = H \cup \Delta H$$
$$\Delta E' = \Delta E_0 \cup \Delta E_1 \cup \Delta E_2 \cup \Delta E_3 \cup \Delta E_4$$
$$\Delta S' = \left\{ s' : (s, g, s') \in \Delta E' \right\}$$
$$\Delta H' = \Delta H_0 \cup \Delta H_1 \cup \Delta H_2 \cup \Delta H_3 \cup \Delta H_4$$
$$G' = (S \cup \Delta S, \Gamma, E', q_0)$$
$$G'_\epsilon = (S', H')$$
$$\Delta G' = (\Delta S' - S', \Delta E' - E').$$

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 ⟵ {(Ĩ(pr), Ĩ(pr))}
03   while W ≠ ∅
04     remove (ζ̃1, ζ̃2) from W
05     switch ζ̃2
06       case ζ̃2 of Entry, CApply, Inner-CEval
07         for each ζ̃3 in succ(ζ̃2) Propagate(ζ̃1, ζ̃3)
08       case ζ̃2 of Call
09         for each ζ̃3 in succ(ζ̃2)
10           Propagate(ζ̃3, ζ̃3)
11           insert (ζ̃1, ζ̃2, ζ̃3) in Callers
12           for each (ζ̃3, ζ̃4) in Summary  Update(ζ̃1, ζ̃2, ζ̃3, ζ̃4)
13       case ζ̃2 of Exit-CEval
14         if ζ̃1 = Ĩ(pr) then
15           Final(ζ̃2)
16         else
17           insert (ζ̃1, ζ̃2) in Summary
18           for each (ζ̃3, ζ̃4, ζ̃1) in Callers  Update(ζ̃3, ζ̃4, ζ̃1, ζ̃2)
19           for each (ζ̃3, ζ̃4, ζ̃1) in TCallers Propagate(ζ̃3, ζ̃2)
20       case ζ̃2 of Exit-TC
21         for each ζ̃3 in succ(ζ̃2)
22           Propagate(ζ̃3, ζ̃3)
23           insert (ζ̃1, ζ̃2, ζ̃3) in TCallers
24           for each (ζ̃3, ζ̃4) in Summary Propagate(ζ̃1, ζ̃4)

     Propagate(ζ̃1, ζ̃2) ≜
25     if (ζ̃1, ζ̃2) not in Seen then insert (ζ̃1, ζ̃2) in Seen and W

     Update(ζ̃1, ζ̃2, ζ̃3, ζ̃4) ≜
26     ζ̃1 of the form  (⟦(λ_l1 (u1 k1) call1)⟧ , d̂1, h1)
27     ζ̃2 of the form  (⟦(f e2 (λ_γ2 (u2) call2))^l2⟧, tf2, h2)
28     ζ̃3 of the form  (⟦(λ_l3 (u3 k3) call3)⟧ , d̂3, h2)
29     ζ̃4 of the form  (⟦(k4 e4)^γ4⟧, tf4, h4)
30     d̂ ⟵  Ã_u(e4, γ4, tf4, h4)
31     tf ⟵  { tf2[f ↦ {⟦(λ_l3 (u3 k3) call3)⟧}]   S?(l2, f)
             { tf2                                  H?(l2, f) ∨ Lam?(f)
32     ζ̃ ⟵  (⟦(λ_γ2 (u2) call2)⟧, d̂, tf, h4)
33     Propagate(ζ̃1, ζ̃)

     Final(ζ̃) ≜
34     ζ̃ of the form  (⟦(k e)^γ⟧, tf, h)
35     insert  (halt, Ã_u(e, γ, tf, h), ∅, h) in Final
```

Figure 8: CFA2 workset algorithm

$$sprout_{(Q,\Gamma,\delta)}(s) = (\Delta E, \Delta H), \text{ where}$$

$$\Delta E = \left\{ s \overset{\epsilon}{\mapsto} q : s \overset{\epsilon}{\mapsto} q \in \delta \right\} \cup \left\{ s \overset{\gamma+}{\mapsto} q : s \overset{\gamma+}{\mapsto} q \in \delta \right\}$$

$$\Delta H = \left\{ s \mapsto q : s \overset{\epsilon}{\mapsto} q \in \delta \right\}.$$

$$addPush_{(Q,\Gamma,\delta)}(G, G_\epsilon)(s \overset{\gamma+}{\mapsto} q) = (\Delta E, \Delta H), \text{ where}$$

$$\Delta E = \left\{ q' \overset{\gamma-}{\mapsto} q'' : q' \in \overrightarrow{G}_\epsilon[q] \text{ and } q' \overset{\gamma-}{\mapsto} q'' \in \delta \right\}$$

$$\Delta H = \left\{ s \mapsto q'' : q' \in \overrightarrow{G}_\epsilon[q] \text{ and } q' \overset{\gamma-}{\mapsto} q'' \in \delta \right\}.$$

$$addEmpty_{(Q,\Gamma,\delta)}(G, G_\epsilon)(s'' \mapsto s''') = (\Delta E, \Delta H), \text{ where}$$

$$\Delta E = \{ s'''' \overset{\gamma-}{\mapsto} q : s' \in \overleftarrow{G}_\epsilon[s''] \text{ and } s'''' \in \overrightarrow{G}_\epsilon[s'''] \text{ and } s \overset{\gamma+}{\mapsto} s' \in G \}$$

$$\Delta H = \{ s \mapsto q : s' \in \overleftarrow{G}_\epsilon[s''] \text{ and } s'''' \in \overrightarrow{G}_\epsilon[s'''] \text{ and } s \overset{\gamma+}{\mapsto} s' \in G \} \cup \left\{ s' \mapsto s''' : s' \in \overleftarrow{G}_\epsilon[s''] \right\} \cup \left\{ s'' \mapsto s'''' : s'''' \in \overrightarrow{G}_\epsilon[s'''] \right\} \cup \left\{ s' \mapsto s'''' : s' \in \overleftarrow{G}_\epsilon[s''] \text{ and } s'''' \in \overrightarrow{G}_\epsilon[s'''] \right\}.$$

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

$$G'_\epsilon = (S', H')$$

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

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 ⟵ {(Ĩ(pr), Ĩ(pr))}
03   while W ≠ ∅
04     remove (ζ̃1, ζ̃2) from W
05     switch ζ̃2
06       case ζ̃2 of Entry, CApply, Inner-CEval
07         for each ζ̃3 in succ(ζ̃2) Propagate(ζ̃1, ζ̃3)
08       case ζ̃2 of Call
09         for each ζ̃3 in succ(ζ̃2)
10           Propagate(ζ̃3, ζ̃3)
11           insert (ζ̃1, ζ̃2, ζ̃3) in Callers
12           for each (ζ̃3, ζ̃4) in Summary  Update(ζ̃1, ζ̃2, ζ̃3, ζ̃4)
13       case ζ̃2 of Exit-CEval
14         if ζ̃1 = Ĩ(pr) then
15           Final(ζ̃2)
16         else
17           insert (ζ̃1, ζ̃2) in Summary
18           for each (ζ̃3, ζ̃4, ζ̃1) in Callers  Update(ζ̃3, ζ̃4, ζ̃1, ζ̃2)
19           for each (ζ̃3, ζ̃4, ζ̃1) in TCallers Propagate(ζ̃3, ζ̃2)
20       case ζ̃2 of Exit-TC
21         for each ζ̃3 in succ(ζ̃2)
22           Propagate(ζ̃3, ζ̃3)
23           insert (ζ̃1, ζ̃2, ζ̃3) in TCallers
24           for each (ζ̃3, ζ̃4) in Summary Propagate(ζ̃1, ζ̃4)

     Propagate(ζ̃1, ζ̃2) ≜
25     if (ζ̃1, ζ̃2) not in Seen then insert (ζ̃1, ζ̃2) in Seen and W

     Update(ζ̃1, ζ̃2, ζ̃3, ζ̃4) ≜
26     ζ̃1 of the form  (⟦(λ_l1 (u1 k1) call1)⟧ , d̂1, h1)
27     ζ̃2 of the form  (⟦(f e2 (λ_γ2 (u2) call2))^l2⟧, tf2, h2)
28     ζ̃3 of the form  (⟦(λ_l3 (u3 k3) call3)⟧ , d̂3, h2)
29     ζ̃4 of the form  (⟦(k4 e4)^γ4⟧, tf4, h4)
30     d̂ ⟵  Ã_u(e4, γ4, tf4, h4)
31     tf ⟵  { tf2[f ↦ {⟦(λ_l3 (u3 k3) call3)⟧}]   S?(l2, f)
             { tf2                                  H?(l2, f) ∨ Lam?(f)
32     ζ̃ ⟵  (⟦(λ_γ2 (u2) call2)⟧, d̂, tf, h4)
33     Propagate(ζ̃1, ζ̃)

     Final(ζ̃) ≜
34     ζ̃ of the form  (⟦(k e)^γ⟧, tf, h)
35     insert  (halt, Ã_u(e, γ, tf, h), ∅, h) in Final
```

Figure 8: CFA2 workset algorithm

$$\hat{f}((\hat{P},\hat{E}),\hat{H},\hat{\sigma}) = ((\hat{P}',\hat{E}'),\hat{H}',\hat{\sigma}''), \text{where}$$

$$\hat{T}_+ = \left\{(\hat{\psi} \overset{\hat{\phi}_+}{\rightarrowtail} \hat{\psi}', \hat{\sigma}') : \hat{\psi} \underset{\hat{\phi}_+}{\overset{\hat{\sigma}}{\rightarrow}} (\hat{\psi}', \hat{\sigma}')\right\}$$

$$\hat{T}_\epsilon = \left\{(\hat{\psi} \overset{\epsilon}{\rightarrowtail} \hat{\psi}', \hat{\sigma}') : \hat{\psi} \underset{\epsilon}{\overset{\hat{\sigma}}{\rightarrow}} (\hat{\psi}', \hat{\sigma}')\right\}$$

$$\hat{T}_- = \{(\hat{\psi}'' \overset{\hat{\phi}_-}{\rightarrowtail} \hat{\psi}''', \hat{\sigma}') : \hat{\psi}'' \underset{\hat{\phi}_-}{\overset{\hat{\sigma}}{\rightarrow}} (\hat{\psi}''', \hat{\sigma}') \text{ and } \hat{\psi} \overset{\hat{\phi}_+}{\rightarrowtail} \hat{\psi}' \in \hat{E} \text{ and } \hat{\psi}' \rightarrowtail \hat{\psi}'' \in \hat{H}\}$$

$$\hat{T}' = \hat{T}_+ \cup \hat{T}_\epsilon \cup \hat{T}_-$$

$$\hat{E}' = \left\{\hat{e} : (\hat{e}, \_) \in \hat{T}'\right\}$$

$$\hat{\sigma}'' = \bigsqcup\left\{\hat{\sigma}' : (\_, \hat{\sigma}') \in \hat{T}'\right\}$$

$$\hat{H}_\epsilon = \left\{\hat{\psi} \rightarrowtail \hat{\psi}'' : \hat{\psi} \rightarrowtail \hat{\psi}' \in \hat{H} \text{ and } \hat{\psi}' \rightarrowtail \hat{\psi}'' \in \hat{H}\right\}$$

$$\hat{H}_{+-} = \{\hat{\psi} \rightarrowtail \hat{\psi}''' : \hat{\psi} \overset{\hat{\phi}_+}{\rightarrowtail} \hat{\psi}' \in \hat{E} \text{ and } \hat{\psi}' \rightarrowtail \hat{\psi}'' \in \hat{H} \text{ and } \hat{\psi}'' \overset{\hat{\phi}_-}{\rightarrowtail} \hat{\psi}''' \in \hat{E}\}$$

$$\hat{H}' = \hat{H}_\epsilon \cup \hat{H}_{+-}$$

$$\hat{P}' = \hat{P} \cup \left\{\hat{\psi}' : \hat{\psi} \overset{g}{\rightarrowtail} \hat{\psi}'\right\}.$$

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

$$G'_\epsilon = (S', H')$$

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

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 ⟵ {(Ĩ(pr), Ĩ(pr))}
03    while W ≠ ∅
04      remove (ζ̃1, ζ̃2) from W
05      switch ζ̃2
06        case ζ̃2 of Entry, CApply, Inner-CEval
07          for each ζ̃3 in succ(ζ̃2) Propagate(ζ̃1, ζ̃3)
08        case ζ̃2 of Call
09          for each ζ̃3 in succ(ζ̃2)
10            Propagate(ζ̃3, ζ̃3)
11            insert (ζ̃1, ζ̃2, ζ̃3) in Callers
12            for each (ζ̃3, ζ̃4) in Summary  Update(ζ̃1, ζ̃2, ζ̃3, ζ̃4)
13        case ζ̃2 of Exit-CEval
14          if ζ̃1 = Ĩ(pr) then
15            Final(ζ̃2)
16          else
17            insert (ζ̃1, ζ̃2) in Summary
18            for each (ζ̃3, ζ̃4, ζ̃1) in Callers  Update(ζ̃3, ζ̃4, ζ̃1, ζ̃2)
19            for each (ζ̃3, ζ̃4, ζ̃1) in TCallers Propagate(ζ̃3, ζ̃2)
20        case ζ̃2 of Exit-TC
21          for each ζ̃3 in succ(ζ̃2)
22            Propagate(ζ̃3, ζ̃3)
23            insert (ζ̃1, ζ̃2, ζ̃3) in TCallers
24            for each (ζ̃3, ζ̃4) in Summary Propagate(ζ̃1, ζ̃4)

      Propagate(ζ̃1, ζ̃2) ≜
25      if (ζ̃1, ζ̃2) not in Seen then insert (ζ̃1, ζ̃2) in Seen and W

      Update(ζ̃1, ζ̃2, ζ̃3, ζ̃4) ≜
26      ζ̃1 of the form  (⟦(λ_{l1}(u1 k1) call1)⟧ , d̂1, h1)
27      ζ̃2 of the form  (⟦(f e2 (λ_{γ2} (u2) call2))^{l2}⟧, tf2, h2)
28      ζ̃3 of the form  (⟦(λ_{l3}(u3 k3) call3)⟧ , d̂3, h2)
29      ζ̃4 of the form  (⟦(k4 e4)^{γ4}⟧, tf4, h4)
30      d̂ ⟵  Ã_u(e4, γ4, tf4, h4)
31      tf ⟵  { tf2[f ↦ {⟦(λ_{l3}(u3 k3) call3)⟧}]   S?(l2, f)
              { tf2                                  H?(l2, f) ∨ Lam?(f)
32      ζ̃ ⟵  (⟦(λ_{γ2}(u2) call2)⟧, d̂, tf, h4)
33      Propagate(ζ̃1, ζ̃)

      Final(ζ̃) ≜
34      ζ̃ of the form  (⟦(k e)^γ⟧, tf, h)
35      insert  (halt, Ã_u(e, γ, tf, h), ∅, h) in Final
```

Figure 8: CFA2 workset algorithm

$$\hat{f}((\hat{P},\hat{E}),\hat{H},\hat{\sigma}) = ((\hat{P}',\hat{E}'),\hat{H}',\hat{\sigma}''), \text{where}$$

$$\hat{T}_+ = \left\{(\hat{\psi} \overset{\hat{\phi}_+}{\rightarrowtail} \hat{\psi}', \hat{\sigma}') : \hat{\psi} \underset{\hat{\phi}_+}{\overset{\hat{\sigma}}{\rightarrow}} (\hat{\psi}', \hat{\sigma}')\right\}$$

$$\hat{T}_\epsilon = \left\{(\hat{\psi} \overset{\epsilon}{\rightarrowtail} \hat{\psi}', \hat{\sigma}') : \hat{\psi} \underset{\epsilon}{\overset{\hat{\sigma}}{\rightarrow}} (\hat{\psi}', \hat{\sigma}')\right\}$$

$$\hat{T}_- = \{(\hat{\psi}'' \overset{\hat{\phi}_-}{\rightarrowtail} \hat{\psi}''', \hat{\sigma}') : \hat{\psi}'' \underset{\hat{\phi}_-}{\overset{\hat{\sigma}}{\rightarrow}} (\hat{\psi}''', \hat{\sigma}') \text{ and } \hat{\psi} \overset{\hat{\phi}_+}{\rightarrowtail} \hat{\psi}' \in \hat{E} \text{ and } \hat{\psi}' \rightarrowtail \hat{\psi}'' \in \hat{H}\}$$

$$\hat{T}' = \hat{T}_+ \cup \hat{T}_\epsilon \cup \hat{T}_-$$

$$\hat{E}' = \left\{\hat{e} : (\hat{e}, \_) \in \hat{T}'\right\}$$

$$\hat{\sigma}'' = \bigsqcup \left\{\hat{\sigma}' : (\_, \hat{\sigma}') \in \hat{T}'\right\}$$

$$\hat{H}_\epsilon = \left\{\hat{\psi} \rightarrowtail \hat{\psi}'' : \hat{\psi} \rightarrowtail \hat{\psi}' \in \hat{H} \text{ and } \hat{\psi}' \rightarrowtail \hat{\psi}'' \in \hat{H}\right\}$$

$$\hat{H}_{+-} = \{\hat{\psi} \rightarrowtail \hat{\psi}''' : \hat{\psi} \overset{\hat{\phi}_+}{\rightarrowtail} \hat{\psi}' \in \hat{E} \text{ and } \hat{\psi}' \rightarrowtail \hat{\psi}'' \in \hat{H} \text{ and } \hat{\psi}'' \overset{\hat{\phi}_-}{\rightarrowtail} \hat{\psi}''' \in \hat{E}\}$$

$$\hat{H}' = \hat{H}_\epsilon \cup \hat{H}_{+-}$$

$$\hat{P}' = \hat{P} \cup \left\{\hat{\psi}' : \hat{\psi} \overset{g}{\rightarrowtail} \hat{\psi}'\right\}.$$

$$addPop_{(Q,\Gamma,s)}(G, G_\epsilon)(s'' \overset{\gamma_-}{\rightarrowtail} q) = (\Delta E, \Delta H), \text{where}$$

$$\Delta E = \emptyset \text{ and } \Delta H = \left\{s \rightarrowtail q : s' \in \overleftarrow{G}_\epsilon[s''] \text{ and } s \overset{\gamma_+}{\rightarrowtail} s' \in G\right\}$$

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

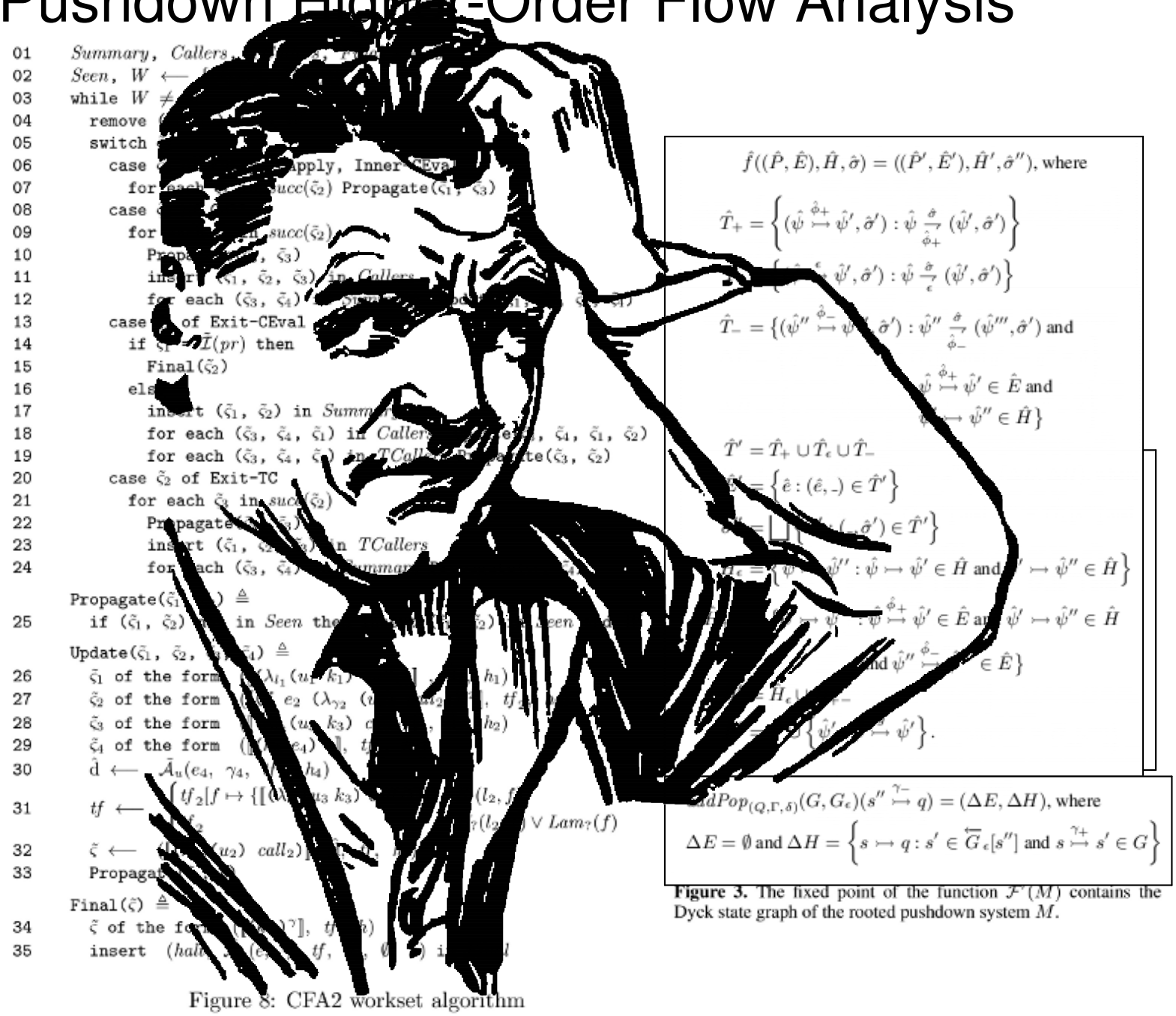


Figure 8: CFA2 workset algorithm

$$\hat{f}((\hat{P},\hat{E}),\hat{H},\hat{\sigma}) = ((\hat{P}',\hat{E}'),\hat{H}',\hat{\sigma}''), \text{ where}$$

$$\hat{T}_+ = \left\{(\hat{\psi} \overset{\hat{\phi}_+}{\mapsto} \hat{\psi}', \hat{\sigma}') : \hat{\psi} \underset{\hat{\phi}_+}{\overset{\hat{\sigma}}{\rightarrow}} (\hat{\psi}', \hat{\sigma}')\right\}$$

$$\left\{(\hat{\psi} \overset{\hat{\epsilon}}{\mapsto} \hat{\psi}', \hat{\sigma}') : \hat{\psi} \underset{\epsilon}{\overset{\hat{\sigma}}{\rightarrow}} (\hat{\psi}', \hat{\sigma}')\right\}$$

$$\hat{T}_- = \{(\hat{\psi}'' \overset{\hat{\phi}_-}{\mapsto} \hat{\psi}', \hat{\sigma}') : \hat{\psi}'' \underset{\hat{\phi}_-}{\overset{\hat{\sigma}}{\rightarrow}} (\hat{\psi}''', \hat{\sigma}') \text{ and}$$

$$\hat{\psi} \overset{\hat{\phi}_+}{\mapsto} \hat{\psi}' \in \hat{E} \text{ and}$$

$$\hat{\psi} \overset{\hat{\epsilon}}{\mapsto} \hat{\psi}'' \in \hat{H}\}$$

$$\hat{T}' = \hat{T}_+ \cup \hat{T}_\epsilon \cup \hat{T}_-$$

$$\hat{E}' = \left\{\hat{e} : (\hat{e}, \_) \in \hat{T}'\right\}$$

$$\hat{H}' = \bigsqcup \left\{(\_, \hat{\sigma}') \in \hat{T}'\right\}$$

$$\hat{H}_\epsilon = \left\{\hat{\psi} \overset{\hat{\epsilon}}{\mapsto} \hat{\psi}'' : \hat{\psi} \overset{\hat{\epsilon}}{\mapsto} \hat{\psi}' \in \hat{H} \text{ and } \hat{\psi}' \overset{\hat{\epsilon}}{\mapsto} \hat{\psi}'' \in \hat{H}\right\}$$

$$\cup \left\{\hat{\psi} \overset{\hat{\epsilon}}{\mapsto} \hat{\psi}'' : \hat{\psi} \overset{\hat{\phi}_+}{\mapsto} \hat{\psi}' \in \hat{E} \text{ and } \hat{\psi}' \overset{\hat{\epsilon}}{\mapsto} \hat{\psi}'' \in \hat{H}\right.$$

$$\left. \text{and } \hat{\psi}'' \overset{\hat{\phi}_-}{\mapsto} \hat{\psi}' \in \hat{E}\right\}$$

$$\hat{H}' = \hat{H} \cup \hat{H}_\epsilon \cup \hat{H}_-$$

$$\hat{H}_- = \left\{\hat{\psi}'' \overset{\hat{\epsilon}}{\mapsto} \hat{\psi}'\right\}.$$

$$AddPop_{(Q,\Gamma,s)}(G, G_\epsilon)(s'' \overset{\gamma_-}{\mapsto} q) = (\Delta E, \Delta H), \text{ where}$$

$$\Delta E = \emptyset \text{ and } \Delta H = \left\{s \mapsto q : s' \in \overleftarrow{G}_\epsilon[s''] \text{ and } s \overset{\gamma_+}{\mapsto} s' \in G\right\}$$

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

Pushdown Higher-Order Flow Analysis

```
01   Summary, Callers, TCallers, Final ⟵ ∅
02   Seen, W ⟵ {(Ĩ(pr), ĩ(pr))}
03   while W ≠ ∅
04     remove (ζ̃1, ζ̃2) from W
05     switch ζ̃2
06       case ζ̃2 of Entry, CApply, Inner-CEval
07         for each ζ̃3 in succ(ζ̃2) Propagate(ζ̃1, ζ̃3)
08       case ζ̃2 of Call
09         for each ζ̃3 in succ(ζ̃2)
10           Propagate(ζ̃3, ζ̃3)
11           insert (ζ̃1, ζ̃2, ζ̃3) in Callers
12           for each (ζ̃3, ζ̃4) in Summary  Update(ζ̃1, ζ̃2, ζ̃3, ζ̃4)
13       case ζ̃2 of Exit-CEval
14         if ζ̃1 = Ĩ(pr) then
15           Final(ζ̃2)
16         else
17           insert (ζ̃1, ζ̃2) in Summary
18           for each (ζ̃3, ζ̃4, ζ̃1) in Callers  Update(ζ̃3, ζ̃4, ζ̃1, ζ̃2)
19           for each (ζ̃3, ζ̃4, ζ̃1) in TCallers Propagate(ζ̃3, ζ̃2)
20       case ζ̃2 of Exit-TC
21         for each ζ̃3 in succ(ζ̃2)
22           Propagate(ζ̃3, ζ̃3)
23           insert (ζ̃1, ζ̃2, ζ̃3) in TCallers
24           for each (ζ̃3, ζ̃4) in Summary Propagate(ζ̃1, ζ̃4)

     Propagate(ζ̃1, ζ̃2) ≜
25     if (ζ̃1, ζ̃2) not in Seen then insert (ζ̃1, ζ̃2) in Seen and W

     Update(ζ̃1, ζ̃2, ζ̃3, ζ̃4) ≜
26     ζ̃1 of the form  (⟦(λ_l1 (u1 k1) call1)⟧ , d̂1, h1)
27     ζ̃2 of the form  (⟦(f e2 (λ_γ2 (u2) call2))^l2⟧, tf2, h2)
28     ζ̃3 of the form  (⟦(λ_l3 (u3 k3) call3)⟧ , d̂3, h2)
29     ζ̃4 of the form  (⟦(k4 e4)^γ4⟧, tf4, h4)
30     d̂ ⟵  Âu(e4, γ4, tf4, h4)
31     tf ⟵  { tf2[f ↦ {⟦(λ_l3 (u3 k3) call3)⟧}]   S?(l2, f)
             { tf2                                  H?(l2, f) ∨ Lam?(f)
32     ζ̃ ⟵  (⟦(λ_γ2 (u2) call2)⟧, d̂, tf, h4)
33     Propagate(ζ̃1, ζ̃)

     Final(ζ̃) ≜
34     ζ̃ of the form  (⟦(k e)^γ⟧, tf, h)
35     insert  (halt, Âu(e, γ, tf, h), ∅, h) in Final
```

Figure 8: CFA2 workset algorithm

$$\hat{f}((\hat{P},\hat{E}),\hat{H},\hat{\sigma}) = ((\hat{P}',\hat{E}'),\hat{H}',\hat{\sigma}''), \text{ where}$$

$$\hat{T}_+ = \left\{(\hat{\psi} \overset{\hat{\phi}_+}{\rightarrowtail} \hat{\psi}', \hat{\sigma}') : \hat{\psi} \underset{\hat{\phi}_+}{\overset{\hat{\sigma}}{\rightarrow}} (\hat{\psi}', \hat{\sigma}')\right\}$$

$$\hat{T}_\epsilon = \left\{(\hat{\psi} \overset{\epsilon}{\rightarrowtail} \hat{\psi}', \hat{\sigma}') : \hat{\psi} \underset{\epsilon}{\overset{\hat{\sigma}}{\rightarrow}} (\hat{\psi}', \hat{\sigma}')\right\}$$

$$\hat{T}_- = \{(\hat{\psi}'' \overset{\hat{\phi}_-}{\rightarrowtail} \hat{\psi}''', \hat{\sigma}') : \hat{\psi}'' \underset{\hat{\phi}_-}{\overset{\hat{\sigma}}{\rightarrow}} (\hat{\psi}''', \hat{\sigma}') \text{ and } \hat{\psi} \overset{\hat{\phi}_+}{\rightarrowtail} \hat{\psi}' \in \hat{E} \text{ and } \hat{\psi}' \rightarrowtail \hat{\psi}'' \in \hat{H}\}$$

$$\hat{T}' = \hat{T}_+ \cup \hat{T}_\epsilon \cup \hat{T}_-$$

$$\hat{E}' = \left\{\hat{e} : (\hat{e}, \_) \in \hat{T}'\right\}$$

$$\hat{\sigma}'' = \bigsqcup\left\{\hat{\sigma}' : (\_, \hat{\sigma}') \in \hat{T}'\right\}$$

$$\hat{H}_\epsilon = \left\{\hat{\psi} \rightarrowtail \hat{\psi}'' : \hat{\psi} \rightarrowtail \hat{\psi}' \in \hat{H} \text{ and } \hat{\psi}' \rightarrowtail \hat{\psi}'' \in \hat{H}\right\}$$

$$\hat{H}_{+-} = \{\hat{\psi} \rightarrowtail \hat{\psi}''' : \hat{\psi} \overset{\hat{\phi}_+}{\rightarrowtail} \hat{\psi}' \in \hat{E} \text{ and } \hat{\psi}' \rightarrowtail \hat{\psi}'' \in \hat{H} \text{ and } \hat{\psi}'' \overset{\hat{\phi}_-}{\rightarrowtail} \hat{\psi}''' \in \hat{E}\}$$

$$\hat{H}' = \hat{H}_\epsilon \cup \hat{H}_{+-}$$

$$\hat{P}' = \hat{P} \cup \left\{\hat{\psi}' : \hat{\psi} \overset{g}{\rightarrowtail} \hat{\psi}'\right\}.$$

$$addPop_{(Q,\Gamma,s)}(G, G_\epsilon)(s'' \overset{\gamma_-}{\rightarrowtail} q) = (\Delta E, \Delta H), \text{ where}$$

$$\Delta E = \emptyset \text{ and } \Delta H = \left\{s \rightarrowtail q : s' \in \overleftarrow{G}_\epsilon[s''] \text{ and } s \overset{\gamma_+}{\rightarrowtail} s' \in G\right\}$$

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 : Context → $\wp$(Value)

CESK + M

- M : Context → $\wp$(Value)
- Context = CES

CESK + M

- M : Context → $\wp$(Value)
- Context = CES
- New continuation frame: rt(ctx : Context)

CESK + M

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

CESK + M

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

CESKM + Ξ

- Ξ : Context → $\wp$(Kont)

CESKM + Ξ

- Ξ : Context → $\wp$(Kont)
- Store up to rt/mt in Ξ on call

CESKM + Ξ

- Ξ : Context → $\wp$(Kont)
- Store up to rt/mt in Ξ on call
- Return to all calling contexts after memoizing

CESKM + Ξ

- Ξ : Context → $\wp$(Kont)
- Store up to rt/mt in Ξ 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: σ_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: σ_1

f_0	`id`
y_0	`1`

Memo

Store in rt: σ_1

Contexts

$\langle$`(f y) ρ1 σ1`$\rangle$ `(let* (... [n1 •] ...) ...)`

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

Store:σ_2

f_0	id
y_0	1
x_0	1

Memo

Store in rt:σ_2

Contexts

$\langle$(f y) ρ_1 $\sigma_1\rangle$ (let* (... [n1 •] ...) ...)

$\langle$x ρ_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:σ_2

f_0	`id`
y_0	`1`
x_0	`1`

Memo

$\langle$`(f y)` ρ_1 $\sigma_1\rangle$ `1`

Store in rt:σ_2

Contexts

$\langle$`(f y)` ρ_1 $\sigma_1\rangle$ `(let* (... [n1 •] ...) ...)`

$\langle$`x` ρ_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:σ_2

f_0	`id`
y_0	`1`
x_0	`1`

Memo

$\langle$`(f y)` ρ_1 $\sigma_1\rangle$	`1`
$\langle$`x` ρ_1 $\sigma_2\rangle$	`1`

Store in rt:σ_1

Contexts

$\langle$`(f y)` ρ_1 $\sigma_1\rangle$	`(let* (... [n1 •] ...) ...)`
$\langle$`x` ρ_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: σ_3

f_0	id
y_0	1
x_0	1
$n1_0$	1

Memo

$\langle$(f y) ρ_1 $\sigma_1\rangle$ 1

$\langle$x ρ_1 $\sigma_2\rangle$ 1

Store in rt: N/A

Contexts

$\langle$(f y) ρ_1 $\sigma_1\rangle$ (let* (... [n1 •] ...) ...)

$\langle$x ρ_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:σ_4

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

Memo

$\langle$(f y) ρ_1 $\sigma_1\rangle$	1
$\langle$x ρ_1 $\sigma_2\rangle$	1

Store in rt:σ_4

Contexts

$\langle$(f y) ρ_1 $\sigma_1\rangle$	(let* (... [n1 •] ...) ...)
$\langle$x ρ_1 $\sigma_2\rangle$	(let* (... [app (λ (f y) •)] ...) ...)
$\langle$(f y) ρ_4 $\sigma_4\rangle$	(let* (... [n2 •]) ...)

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

Store:σ_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) ρ_1 $\sigma_1\rangle$	1
$\langle$x ρ_1 $\sigma_2\rangle$	1

Store in rt:σ_5

Contexts

$\langle$(f y) ρ_1 $\sigma_1\rangle$	(let* (... [n1 •] ...) ...)
$\langle$x ρ_1 $\sigma_2\rangle$	(let* (... [app (λ (f y) •)] ...) ...)
$\langle$(f y) ρ_4 $\sigma_4\rangle$	(let* (... [n2 •]) ...)
$\langle$x ρ_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:σ_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) ρ_1 $\sigma_1\rangle$	1
$\langle$x ρ_1 $\sigma_2\rangle$	1
$\langle$(f y) ρ_4 $\sigma_4\rangle$	2

Store in rt:σ_5

Contexts

$\langle$(f y) ρ_1 $\sigma_1\rangle$	(let* (... [n1 •] ...) ...)
$\langle$x ρ_1 $\sigma_2\rangle$	(let* (... [app (λ (f y) •)] ...) ...)
$\langle$(f y) ρ_4 $\sigma_4\rangle$	(let* (... [n2 •]) ...)
$\langle$x ρ_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:σ_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) ρ_1 $\sigma_1\rangle$	1
$\langle$x ρ_1 $\sigma_2\rangle$	1
$\langle$(f y) ρ_4 $\sigma_4\rangle$	2
$\langle$x ρ_5 $\sigma_5\rangle$	2

Store in rt:σ_4

Contexts

$\langle$(f y) ρ_1 $\sigma_1\rangle$	(let* (... [n1 •] ...) ...)
$\langle$x ρ_1 $\sigma_2\rangle$	(let* (... [app (λ (f y) •)] ...) ...)
$\langle$(f y) ρ_4 $\sigma_4\rangle$	(let* (... [n2 •]) ...)
$\langle$x ρ_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: σ_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

Memo

$\langle$(f y) ρ_1 $\sigma_1\rangle$	1
$\langle$x ρ_1 $\sigma_2\rangle$	1
$\langle$(f y) ρ_4 $\sigma_4\rangle$	2
$\langle$x ρ_5 $\sigma_5\rangle$	2

Contexts

$\langle$(f y) ρ_1 $\sigma_1\rangle$	(let* (... [n1 •] ...) ...)
$\langle$x ρ_1 $\sigma_2\rangle$	(let* (... [app (λ (f y) •)] ...) ...)
$\langle$(f y) ρ_4 $\sigma_4\rangle$	(let* (... [n2 •]) ...)
$\langle$x ρ_5 $\sigma_5\rangle$	(let* (... [app (λ (f y) •)] ...) ...)

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[(reset F[(shift k e)])]
```

↦

```
E[e{k:=(λ (x) (reset F[x]))}]
```

(F doesn't contain a reset)

Metacontinuations

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

Metacontinuations

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

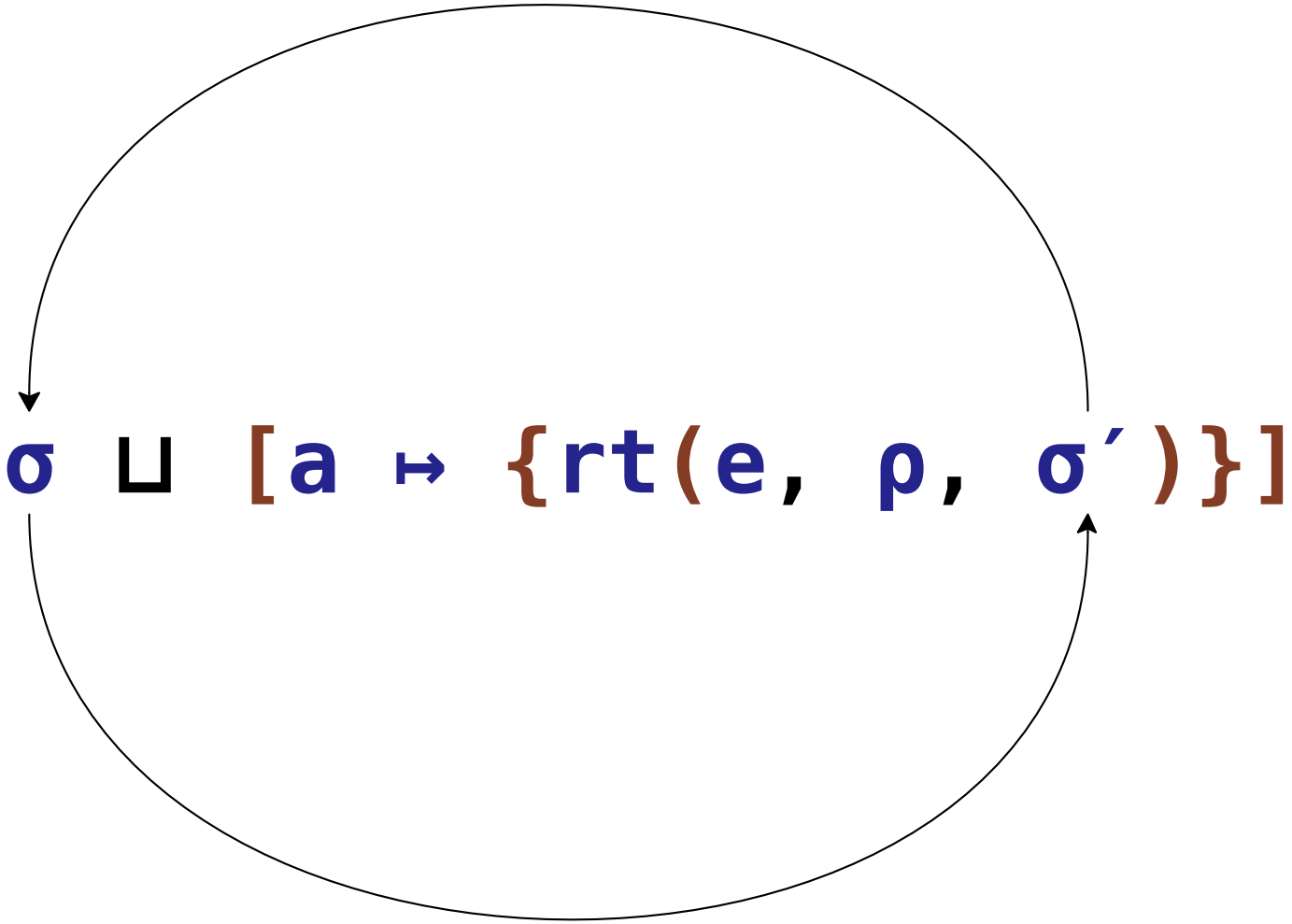
Metacontinuations

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

Naive doesn’t cut it

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

Naive doesn't cut it



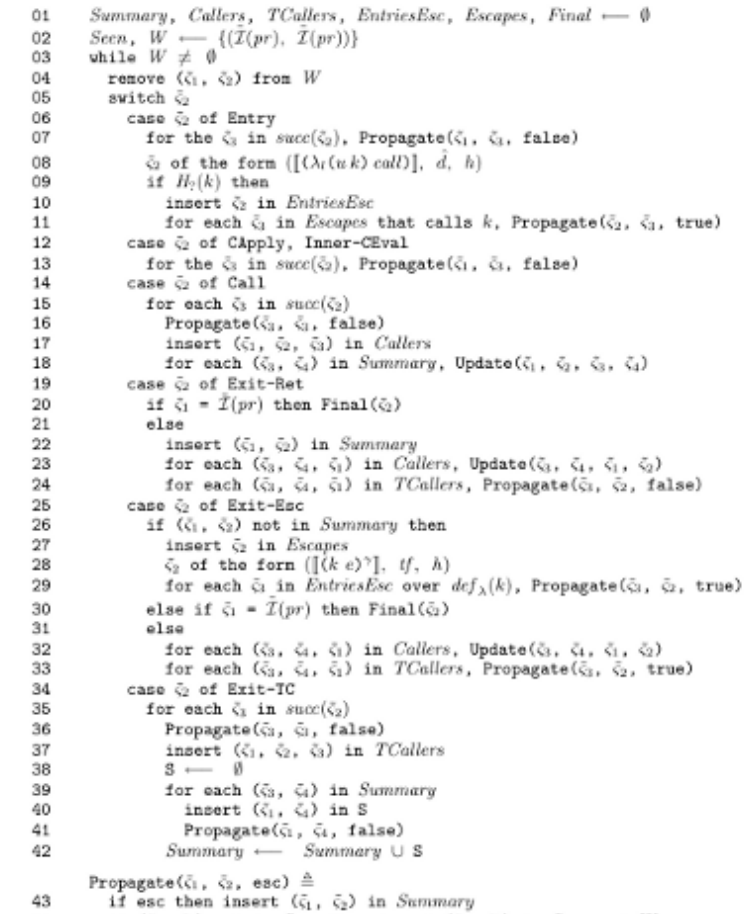


Figure 8: CFA2 workset algorithm

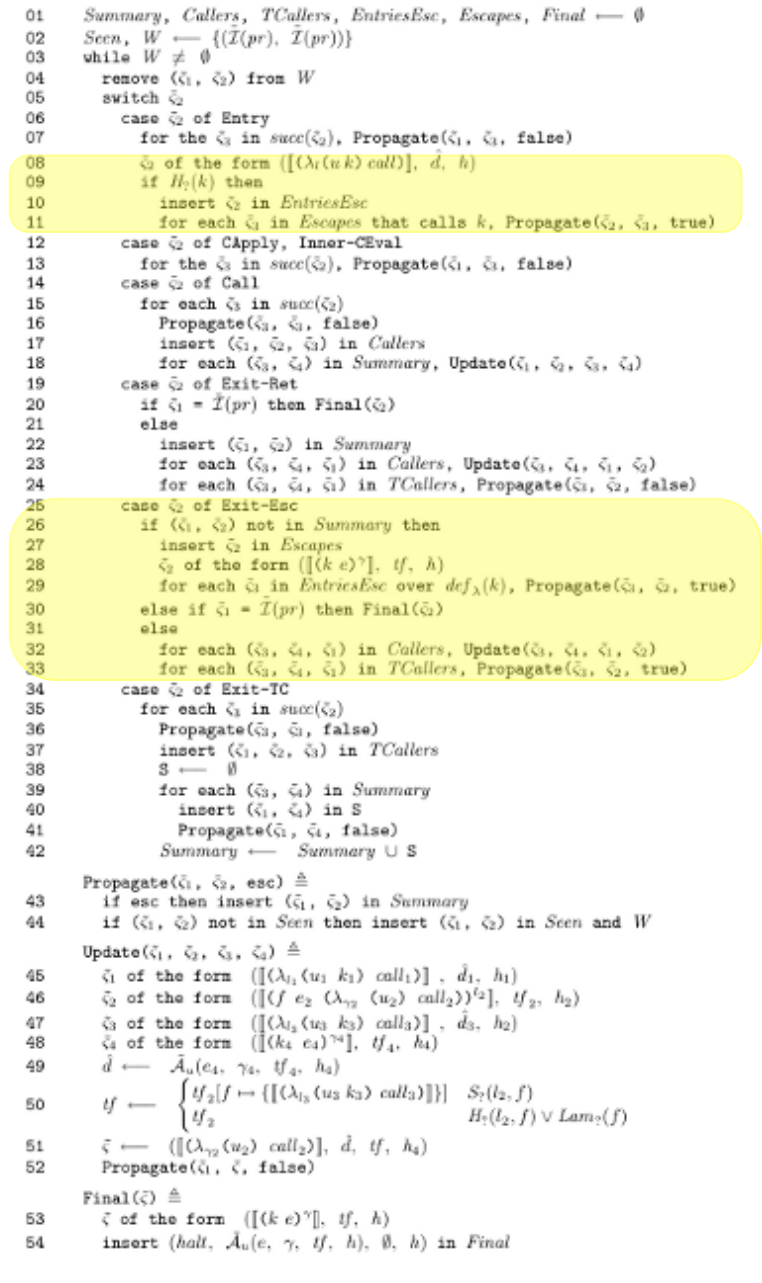


Figure 8: CFA2 workset algorithm

The AAM way

Break circularity with indirection

Approximation tuning: **alloc**

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

Approximation tuning: **alloc**

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

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

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

Approximation tuning: **alloc**

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

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

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

K

Metacontinuations

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

Metacontinuations

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

$$C ::= \mathbf{halt} \mid \sharp(ctx)$$

Metacontinuations

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

$$C ::= \mathbf{halt} \mid \sharp(ctx)$$

$$ctx ::= \langle e, \rho, \sigma \rangle \mid \langle \kappa, v, \sigma \rangle$$

The new context

- Prompts treated just like function calls

The new context

- Prompts treated just like function calls

$$\mathtt{call}\langle \kappa,\ v,\ \sigma,\ \kappa,\ C\rangle$$

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

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 Ξ

Fix up memoization and returns

Approximate contexts mean all instantiations from Ξ

$\langle \mathtt{rt(e, \rho, a), C, v} \rangle$

Fix up memoization and returns

Approximate contexts mean all instantiations from Ξ

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

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

Fix up memoization and returns

Approximate contexts mean all instantiations from Ξ

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

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

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

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 κ through Ξ

$$\mathscr{T}(\mathtt{rt}(\mathtt{e}, \rho, \sigma)) = \bigcup\{\mathscr{T}(\kappa) : \kappa \in \Xi(\mathtt{e}, \rho, \sigma)\}$$