Designing Precise Pushdown Higher-Order Flow Analyses

J. Ian Johnson, David Van Horn and Olin Shivers
{ianj,dvanhorn,shivers}@ccs.neu.edu
Northeastern University
Boston, MA, USA

Flow analysis is indispensible

Flow analysis is indispensible

Every language should have one

Analysis is great

- Compiler optimizations

Analysis is great

- Compiler optimizations
- Type reconstruction

Analysis is great

- Compiler optimizations
- Type reconstruction
- Deadlock detection

Analysis is great

- Compiler optimizations
- Type reconstruction
- Deadlock detection
- Data race detection

Analysis is great

- Compiler optimizations
- Type reconstruction
- Deadlock detection
- Data race detection
- Code quality enforcement

Analysis is great

- Compiler optimizations
- Type reconstruction
- Deadlock detection
- Data race detection
- Code quality enforcement
- *Hundreds* more

Analysis is terrible

- Annoying false positives

Analysis is terrible

- Annoying false positives
- Takes too long

Analysis is terrible

- Annoying false positives
- Takes too long
- Hard to implement

Analysis is terrible

- Annoying false positives
- Takes too long
- Hard to implement
- Hard to show correct

Analysis is terrible

- Annoying false positives
- Takes too long
- Hard to implement
- Hard to show correct
- The good ones inherently first-order

Analysis is terrible

- Annoying false positives
- Takes too long
- Hard to implement
- Hard to show correct
- The good ones inherently first-order

We can fix this

Analysis is terrible

- Annoying false positives
- Takes too long
- Hard to implement
- Hard to show correct
- The good ones inherently first-order

We can fix this

First some history

Flow Graphs vs. Programs

Higher-order

First-order

(Hecht 77) CFG + lattice

Flow Graphs vs. Programs

Higher-order

(Jones & Muchnick 82) Program + set constraints

First-order

(Hecht 77) CFG + lattice

(Sharir & Pnueli 81) CFG + lattice + summaries

Flow Graphs vs. Programs

Higher-order

(Jones & Muchnick 82) Program + set constraints

(Shivers 91) CPS + set constraints

First-order

(Hecht 77) CFG + lattice

(Sharir & Pnueli 81) CFG + lattice + summaries

(Reps 95) PDS + idempotent semiring

Flow Graphs vs. Programs

First-order

(Hecht 77) CFG + lattice

(Sharir & Pnueli 81) CFG + lattice + summaries

(Reps 95) PDS + idempotent semiring

Higher-order

(Jones & Muchnick 82) Program + set constraints

(Shivers 91) CPS + set constraints

(Might 07) CPS + abstract small step semantics

Flow Graphs vs. Programs

First-order

(Hecht 77) CFG + lattice

(Sharir & Pnueli 81) CFG + lattice + summaries

(Reps 95) PDS + idempotent semiring

Higher-order

(Jones & Muchnick 82) Program + set constraints

(Shivers 91) CPS + set constraints

(Might 07) CPS + abstract small step semantics

(Might & Van Horn 10) Derived from concrete

Flow Graphs vs. Programs

First-order

(Hecht 77) CFG + lattice

(Sharir & Pnueli 81) CFG + lattice + summaries

(Reps 95) PDS + idempotent semiring

Higher-order

(Jones & Muchnick 82) Program + set constraints

(Shivers 91) CPS + set constraints

(Might 07) CPS + abstract small step semantics

(Might & Van Horn 10) Derived from concrete

(Vardoulakis & Shivers 10) CPS + summaries

(Earl et al. 10) ANF + Dyck state graph

Flow Graphs vs. Programs

First-order

(Hecht 77) CFG + lattice

(Sharir & Pnueli 81) CFG + lattice + summaries

(Reps 95) PDS + idempotent semiring

Higher-order

(Jones & Muchnick 82) Program + set constraints

(Shivers 91) CPS + set constraints

(Might 07) CPS + abstract small step semantics

(Might & Van Horn 10) Derived from concrete

(Vardoulakis & Shivers 10) CPS + summaries

(Earl et al. 10) ANF + Dyck state graph

Graphs are a bad abstraction

Flow Graphs vs. Programs

Higher-order

(Jones & Muchnick 82) Program + set constraints

(Shivers 91) CPS + set constraints

(Might 07) CPS + abstract small step semantics

(Might & Van Horn 10) Derived from concrete

(Vardoulakis & Shivers 10) CPS + summaries

(Earl et al. 10) ANF + Dyck state graph

First-order

(Hecht 77) CFG + lattice

(Sharir & Pnueli 81) CFG + lattice + summaries

(Reps 95) PDS + idempotent semiring

Graphs are a bad abstraction

Flow Graphs vs. Programs

First-order

(Hecht 77) CFG + lattice

(Sharir & Pnueli 81) CFG + lattice + summaries

(Reps 95) PDS + idempotent semiring

Higher-order

(Jones & Muchnick 82) Program + set constraints

(Shivers 91) CPS + set constraints

(Might 07) CPS + abstract small step semantics

(Might & Van Horn 10) Derived from concrete

(Vardoulakis & Shivers 10) CPS + summaries

(Earl et al. 10) ANF + Dyck state graph

Flow Graphs vs. Programs

Higher-order

(Jones & Muchnick 82) Program + set constraints

(Shivers 91) CPS + set constraints

(Might 07) CPS + abstract small step semantics

(Might & Van Horn 10) Derived from concrete

(Vardoulakis & Shivers 10) CPS + summaries

(Earl et al. 10) ANF + Dyck state graph

?

First-order

(Hecht 77) CFG + lattice

(Sharir & Pnueli 81) CFG + lattice + summaries

(Reps 95) PDS + idempotent semiring

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: pointwise-abstraction

Functions Without Stack

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

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

Functions Without Stack

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

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

Functions Without Stack

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

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

0CFA: Not even terminating!

Pushdown Higher-Order Flow Analysis

$Summary$, $Callers$, $TCallers$, $Final \longleftarrow \emptyset$
$Seen$, $W \longleftarrow \{(\tilde{\mathcal{I}}(pr),\ \tilde{\mathcal{I}}(pr))\}$
while $W \neq \emptyset$
  remove $(\tilde{\varsigma}_1, \tilde{\varsigma}_2)$ from W
  switch $\tilde{\varsigma}_2$
    case $\tilde{\varsigma}_2$ of Entry, CApply, Inner-CEval
      for each $\tilde{\varsigma}_3$ in $succ(\tilde{\varsigma}_2)$ Propagate$(\tilde{\varsigma}_1, \tilde{\varsigma}_3)$
    case $\tilde{\varsigma}_2$ of Call
      for each $\tilde{\varsigma}_3$ in $succ(\tilde{\varsigma}_2)$
        Propagate$(\tilde{\varsigma}_3, \tilde{\varsigma}_3)$
        insert $(\tilde{\varsigma}_1, \tilde{\varsigma}_2, \tilde{\varsigma}_3)$ in $Callers$
        for each $(\tilde{\varsigma}_3, \tilde{\varsigma}_4)$ in $Summary$ Update$(\tilde{\varsigma}_1, \tilde{\varsigma}_2, \tilde{\varsigma}_3, \tilde{\varsigma}_4)$
    case $\tilde{\varsigma}_2$ of Exit-CEval
      if $\tilde{\varsigma}_1 = \tilde{\mathcal{I}}(pr)$ then
        Final$(\tilde{\varsigma}_2)$
      else
        insert $(\tilde{\varsigma}_1, \tilde{\varsigma}_2)$ in $Summary$
        for each $(\tilde{\varsigma}_3, \tilde{\varsigma}_4, \tilde{\varsigma}_1)$ in $Callers$ Update$(\tilde{\varsigma}_3, \tilde{\varsigma}_4, \tilde{\varsigma}_1, \tilde{\varsigma}_2)$
        for each $(\tilde{\varsigma}_3, \tilde{\varsigma}_4, \tilde{\varsigma}_1)$ in $TCallers$ Propagate$(\tilde{\varsigma}_3, \tilde{\varsigma}_2)$
    case $\tilde{\varsigma}_2$ of Exit-TC
      for each $\tilde{\varsigma}_3$ in $succ(\tilde{\varsigma}_2)$
        Propagate$(\tilde{\varsigma}_3, \tilde{\varsigma}_3)$
        insert $(\tilde{\varsigma}_1, \tilde{\varsigma}_2, \tilde{\varsigma}_3)$ in $TCallers$
        for each $(\tilde{\varsigma}_3, \tilde{\varsigma}_4)$ in $Summary$ Propagate$(\tilde{\varsigma}_1, \tilde{\varsigma}_4)$

Propagate$(\tilde{\varsigma}_1, \tilde{\varsigma}_2) \triangleq$
  if $(\tilde{\varsigma}_1, \tilde{\varsigma}_2)$ not in $Seen$ then insert $(\tilde{\varsigma}_1, \tilde{\varsigma}_2)$ in $Seen$ and W

Update$(\tilde{\varsigma}_1, \tilde{\varsigma}_2, \tilde{\varsigma}_3, \tilde{\varsigma}_4) \triangleq$
  $\tilde{\varsigma}_1$ of the form $([\![(\lambda_{l_1}(u_1\ k_1)\ call_1)]\!],\ \hat{\mathrm{d}}_1,\ h_1)$
  $\tilde{\varsigma}_2$ of the form $([\![(f\ e_2\ (\lambda_{\gamma_2}\ (u_2)\ call_2))^{l_2}]\!],\ tf_2,\ h_2)$
  $\tilde{\varsigma}_3$ of the form $([\![(\lambda_{l_3}(u_3\ k_3)\ call_3)]\!],\ \hat{\mathrm{d}}_3,\ h_2)$
  $\tilde{\varsigma}_4$ of the form $([\![(k_4\ e_4)^{\gamma_4}]\!],\ tf_4,\ h_4)$
  $\hat{\mathrm{d}} \longleftarrow \hat{\mathcal{A}}_u(e_4,\ \gamma_4,\ tf_4,\ h_4)$
  $tf \longleftarrow \begin{cases} tf_2[f \mapsto \{[\![(\lambda_{l_3}(u_3\ k_3)\ call_3)]\!]\}] & S_?(l_2, f) \\ tf_2 & H_?(l_2, f) \vee Lam_?(f) \end{cases}$
  $\tilde{\varsigma} \longleftarrow ([\![(\lambda_{\gamma_2}(u_2)\ call_2)]\!],\ \hat{\mathrm{d}},\ tf,\ h_4)$
  Propagate$(\tilde{\varsigma}_1, \tilde{\varsigma})$

Final$(\tilde{\varsigma}) \triangleq$
  $\tilde{\varsigma}$ of the form $([\![(k\ e)^{\gamma}]\!],\ tf,\ h)$
  insert $(halt,\ \hat{\mathcal{A}}_u(e,\ \gamma,\ tf,\ h),\ \emptyset,\ h)$ in $Final$

Figure 8: CFA2 workset algorithm

Pushdown Higher-Order Flow Analysis

$Summary$, $Callers$, $TCallers$, $Final \longleftarrow \emptyset$
$Seen$, $W \longleftarrow \{(\tilde{\mathcal{I}}(pr),\ \tilde{\mathcal{I}}(pr))\}$
`while` $W \neq \emptyset$
  `remove` $(\tilde{\varsigma}_1, \tilde{\varsigma}_2)$ `from` W
  `switch` $\tilde{\varsigma}_2$
    `case` $\tilde{\varsigma}_2$ `of Entry, CApply, Inner-CEval`
      `for each` $\tilde{\varsigma}_3$ `in` $succ(\tilde{\varsigma}_2)$ `Propagate(`$\tilde{\varsigma}_1, \tilde{\varsigma}_3$`)`
    `case` $\tilde{\varsigma}_2$ `of Call`
      `for each` $\tilde{\varsigma}_3$ `in` $succ(\tilde{\varsigma}_2)$
        `Propagate(`$\tilde{\varsigma}_3, \tilde{\varsigma}_3$`)`
        `insert` $(\tilde{\varsigma}_1, \tilde{\varsigma}_2, \tilde{\varsigma}_3)$ `in` $Callers$
        `for each` $(\tilde{\varsigma}_3, \tilde{\varsigma}_4)$ `in` $Summary$ `Update(`$\tilde{\varsigma}_1, \tilde{\varsigma}_2, \tilde{\varsigma}_3, \tilde{\varsigma}_4$`)`
    `case` $\tilde{\varsigma}_2$ `of Exit-CEval`
      `if` $\tilde{\varsigma}_1 = \tilde{\mathcal{I}}(pr)$ `then`
        `Final(`$\tilde{\varsigma}_2$`)`
      `else`
        `insert` $(\tilde{\varsigma}_1, \tilde{\varsigma}_2)$ `in` $Summary$
        `for each` $(\tilde{\varsigma}_3, \tilde{\varsigma}_4, \tilde{\varsigma}_1)$ `in` $Callers$ `Update(`$\tilde{\varsigma}_3, \tilde{\varsigma}_4, \tilde{\varsigma}_1, \tilde{\varsigma}_2$`)`
        `for each` $(\tilde{\varsigma}_3, \tilde{\varsigma}_4, \tilde{\varsigma}_1)$ `in` $TCallers$ `Propagate(`$\tilde{\varsigma}_3, \tilde{\varsigma}_2$`)`
    `case` $\tilde{\varsigma}_2$ `of Exit-TC`
      `for each` $\tilde{\varsigma}_3$ `in` $succ(\tilde{\varsigma}_2)$
        `Propagate(`$\tilde{\varsigma}_3, \tilde{\varsigma}_3$`)`
        `insert` $(\tilde{\varsigma}_1, \tilde{\varsigma}_2, \tilde{\varsigma}_3)$ `in` $TCallers$
        `for each` $(\tilde{\varsigma}_3, \tilde{\varsigma}_4)$ `in` $Summary$ `Propagate(`$\tilde{\varsigma}_1, \tilde{\varsigma}_4$`)`

`Propagate(`$\tilde{\varsigma}_1, \tilde{\varsigma}_2$`)` $\triangleq$
  `if` $(\tilde{\varsigma}_1, \tilde{\varsigma}_2)$ `not in` $Seen$ `then insert` $(\tilde{\varsigma}_1, \tilde{\varsigma}_2)$ `in` $Seen$ `and` W

`Update(`$\tilde{\varsigma}_1, \tilde{\varsigma}_2, \tilde{\varsigma}_3, \tilde{\varsigma}_4$`)` $\triangleq$
  $\tilde{\varsigma}_1$ `of the form` $([\![(\lambda_{l_1}(u_1\ k_1)\ call_1)]\!],\ \hat{\mathrm{d}}_1,\ h_1)$
  $\tilde{\varsigma}_2$ `of the form` $([\![(f\ e_2\ (\lambda_{\gamma_2}\ (u_2)\ call_2))^{l_2}]\!],\ tf_2,\ h_2)$
  $\tilde{\varsigma}_3$ `of the form` $([\![(\lambda_{l_3}(u_3\ k_3)\ call_3)]\!],\ \hat{\mathrm{d}}_3,\ h_2)$
  $\tilde{\varsigma}_4$ `of the form` $([\![(k_4\ e_4)^{\gamma_4}]\!],\ tf_4,\ h_4)$
  $\hat{\mathrm{d}} \longleftarrow \hat{\mathcal{A}}_u(e_4,\ \gamma_4,\ tf_4,\ h_4)$
  $tf \longleftarrow \begin{cases} tf_2[f \mapsto \{[\![(\lambda_{l_3}(u_3\ k_3)\ call_3)]\!]\}] & S_?(l_2, f) \\ tf_2 & H_?(l_2, f) \vee Lam_?(f) \end{cases}$
  $\tilde{\varsigma} \longleftarrow ([\![(\lambda_{\gamma_2}(u_2)\ call_2)]\!],\ \hat{\mathrm{d}},\ tf,\ h_4)$
  `Propagate(`$\tilde{\varsigma}_1, \tilde{\varsigma}$`)`

`Final(`$\tilde{\varsigma}$`)` $\triangleq$
  $\tilde{\varsigma}$ `of the form` $([\![(k\ e)^{\gamma}]\!],\ tf,\ h)$
  `insert` $(halt,\ \hat{\mathcal{A}}_u(e,\ \gamma,\ tf,\ h),\ \emptyset,\ h)$ `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

```
Summary, Callers, TCallers, Final ⟵ ∅
Seen, W ⟵ {(Ĩ(pr), Ĩ(pr))}
while W ≠ ∅
  remove (ς̃1, ς̃2) from W
  switch ς̃2
    case ς̃2 of Entry, CApply, Inner-CEval
      for each ς̃3 in succ(ς̃2) Propagate(ς̃1, ς̃3)
    case ς̃2 of Call
      for each ς̃3 in succ(ς̃2)
        Propagate(ς̃3, ς̃3)
        insert (ς̃1, ς̃2, ς̃3) in Callers
        for each (ς̃3, ς̃4) in Summary  Update(ς̃1, ς̃2, ς̃3, ς̃4)
    case ς̃2 of Exit-CEval
      if ς̃1 = Ĩ(pr) then
        Final(ς̃2)
      else
        insert (ς̃1, ς̃2) in Summary
        for each (ς̃3, ς̃4, ς̃1) in Callers  Update(ς̃3, ς̃4, ς̃1, ς̃2)
        for each (ς̃3, ς̃4, ς̃1) in TCallers Propagate(ς̃3, ς̃2)
    case ς̃2 of Exit-TC
      for each ς̃3 in succ(ς̃2)
        Propagate(ς̃3, ς̃3)
        insert (ς̃1, ς̃2, ς̃3) in TCallers
        for each (ς̃3, ς̃4) in Summary Propagate(ς̃1, ς̃4)

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

Update(ς̃1, ς̃2, ς̃3, ς̃4) ≜
  ς̃1 of the form  ([[(λl1 (u1 k1) call1)]] , d̂1, h1)
  ς̃2 of the form  ([[(f e2 (λγ2 (u2) call2))^l2]], tf2, h2)
  ς̃3 of the form  ([[(λl3 (u3 k3) call3)]] , d̂3, h2)
  ς̃4 of the form  ([[(k4 e4)^γ4]], tf4, h4)
  d̂ ⟵  Âu(e4, γ4, tf4, h4)
  tf ⟵ { tf2[f ↦ {[[(λl3 (u3 k3) call3)]]}]   S?(l2, f)
       { tf2                                   H?(l2, f) ∨ Lam?(f)
  ς̃ ⟵ ([[(λγ2 (u2) call2)]], d̂, tf, h4)
  Propagate(ς̃1, ς̃)

Final(ς̃) ≜
  ς̃ of the form  ([[(k e)^γ]], tf, h)
  insert  (halt, Âu(e, γ, tf, h), ∅, h) in Final
```

Figure 8: CFA2 workset algorithm

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

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

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

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

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

$Summary$, $Callers$, $TCallers$, $Final$ $\longleftarrow \emptyset$
$Seen$, W $\longleftarrow \{(\tilde{\mathcal{I}}(pr), \tilde{\mathcal{I}}(pr))\}$
while $W \neq \emptyset$
  remove $(\tilde{\varsigma}_1, \tilde{\varsigma}_2)$ from W
  switch $\tilde{\varsigma}_2$
    case $\tilde{\varsigma}_2$ of Entry, CApply, Inner-CEval
      for each $\tilde{\varsigma}_3$ in $succ(\tilde{\varsigma}_2)$ Propagate($\tilde{\varsigma}_1$, $\tilde{\varsigma}_3$)
    case $\tilde{\varsigma}_2$ of Call
      for each $\tilde{\varsigma}_3$ in $succ(\tilde{\varsigma}_2)$
        Propagate($\tilde{\varsigma}_3$, $\tilde{\varsigma}_3$)
        insert $(\tilde{\varsigma}_1, \tilde{\varsigma}_2, \tilde{\varsigma}_3)$ in $Callers$
        for each $(\tilde{\varsigma}_3, \tilde{\varsigma}_4)$ in $Summary$ Update($\tilde{\varsigma}_1$, $\tilde{\varsigma}_2$, $\tilde{\varsigma}_3$, $\tilde{\varsigma}_4$)
    case $\tilde{\varsigma}_2$ of Exit-CEval
      if $\tilde{\varsigma}_1 = \tilde{\mathcal{I}}(pr)$ then
        Final($\tilde{\varsigma}_2$)
      else
        insert $(\tilde{\varsigma}_1, \tilde{\varsigma}_2)$ in $Summary$
        for each $(\tilde{\varsigma}_3, \tilde{\varsigma}_4, \tilde{\varsigma}_1)$ in $Callers$ Update($\tilde{\varsigma}_3$, $\tilde{\varsigma}_4$, $\tilde{\varsigma}_1$, $\tilde{\varsigma}_2$)
        for each $(\tilde{\varsigma}_3, \tilde{\varsigma}_4, \tilde{\varsigma}_1)$ in $TCallers$ Propagate($\tilde{\varsigma}_3$, $\tilde{\varsigma}_2$)
    case $\tilde{\varsigma}_2$ of Exit-TC
      for each $\tilde{\varsigma}_3$ in $succ(\tilde{\varsigma}_2)$
        Propagate($\tilde{\varsigma}_3$, $\tilde{\varsigma}_3$)
        insert $(\tilde{\varsigma}_1, \tilde{\varsigma}_2, \tilde{\varsigma}_3)$ in $TCallers$
        for each $(\tilde{\varsigma}_3, \tilde{\varsigma}_4)$ in $Summary$ Propagate($\tilde{\varsigma}_1$, $\tilde{\varsigma}_4$)

Propagate($\tilde{\varsigma}_1$, $\tilde{\varsigma}_2$) $\triangleq$
  if $(\tilde{\varsigma}_1, \tilde{\varsigma}_2)$ not in $Seen$ then insert $(\tilde{\varsigma}_1, \tilde{\varsigma}_2)$ in $Seen$ and W

Update($\tilde{\varsigma}_1$, $\tilde{\varsigma}_2$, $\tilde{\varsigma}_3$, $\tilde{\varsigma}_4$) $\triangleq$
  $\tilde{\varsigma}_1$ of the form $([\![(\lambda_{l_1}(u_1\ k_1)\ call_1)]\!],\ \hat{\mathrm{d}}_1,\ h_1)$
  $\tilde{\varsigma}_2$ of the form $([\![(f\ e_2\ (\lambda_{\gamma_2}\ (u_2)\ call_2))^{l_2}]\!],\ tf_2,\ h_2)$
  $\tilde{\varsigma}_3$ of the form $([\![(\lambda_{l_3}(u_3\ k_3)\ call_3)]\!],\ \hat{\mathrm{d}}_3,\ h_2)$
  $\tilde{\varsigma}_4$ of the form $([\![(k_4\ e_4)^{\gamma_4}]\!],\ tf_4,\ h_4)$
  $\hat{\mathrm{d}} \longleftarrow \hat{\mathcal{A}}_u(e_4,\ \gamma_4,\ tf_4,\ h_4)$
  $tf \longleftarrow \begin{cases} tf_2[f \mapsto \{[\![(\lambda_{l_3}(u_3\ k_3)\ call_3)]\!]\}] & S_?(l_2, f) \\ tf_2 & H_?(l_2, f) \vee Lam_?(f) \end{cases}$
  $\tilde{\varsigma} \longleftarrow ([\![(\lambda_{\gamma_2}(u_2)\ call_2)]\!],\ \hat{\mathrm{d}},\ tf,\ h_4)$
  Propagate($\tilde{\varsigma}_1$, $\tilde{\varsigma}$)

Final($\tilde{\varsigma}$) $\triangleq$
  $\tilde{\varsigma}$ of the form $([\![(k\ e)^{\gamma}]\!],\ tf,\ h)$
  insert $(halt,\ \hat{\mathcal{A}}_u(e,\ \gamma,\ tf,\ h),\ \emptyset,\ h)$ in $Final$

Figure 8: CFA2 workset algorithm

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

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

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

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

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

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

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

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

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

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

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

```
Summary, Callers, TCallers, Final ⟵ ∅
Seen, W ⟵ {(Ĩ(pr), Ĩ(pr))}
while W ≠ ∅
  remove (ς̃₁, ς̃₂) from W
  switch ς̃₂
    case ς̃₂ of Entry, CApply, Inner-CEval
      for each ς̃₃ in succ(ς̃₂) Propagate(ς̃₁, ς̃₃)
    case ς̃₂ of Call
      for each ς̃₃ in succ(ς̃₂)
        Propagate(ς̃₃, ς̃₃)
        insert (ς̃₁, ς̃₂, ς̃₃) in Callers
        for each (ς̃₃, ς̃₄) in Summary  Update(ς̃₁, ς̃₂, ς̃₃, ς̃₄)
    case ς̃₂ of Exit-CEval
      if ς̃₁ = Ĩ(pr) then
        Final(ς̃₂)
      else
        insert (ς̃₁, ς̃₂) in Summary
        for each (ς̃₃, ς̃₄, ς̃₁) in Callers  Update(ς̃₃, ς̃₄, ς̃₁, ς̃₂)
        for each (ς̃₃, ς̃₄, ς̃₁) in TCallers Propagate(ς̃₃, ς̃₂)
    case ς̃₂ of Exit-TC
      for each ς̃₃ in succ(ς̃₂)
        Propagate(ς̃₃, ς̃₃)
        insert (ς̃₁, ς̃₂, ς̃₃) in TCallers
        for each (ς̃₃, ς̃₄) in Summary Propagate(ς̃₁, ς̃₄)

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

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

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

Figure 8: CFA2 workset algorithm

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

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

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

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

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

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

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

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

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

$$G' = (S \cup \Delta S, \Gamma, E', q_0)$$
$$G'_\epsilon = (S', H')$$
$$\Delta G' = (\Delta S' - S', \Delta E' - E').$$

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

$Summary$, $Callers$, $TCallers$, $Final$ $\longleftarrow \emptyset$
$Seen$, W $\longleftarrow \{(\tilde{\mathcal{I}}(pr),\ \tilde{\mathcal{I}}(pr))\}$
`while` $W \neq \emptyset$
  `remove` $(\tilde{\varsigma}_1,\ \tilde{\varsigma}_2)$ `from` W
  `switch` $\tilde{\varsigma}_2$
    `case` $\tilde{\varsigma}_2$ `of Entry, CApply, Inner-CEval`
      `for each` $\tilde{\varsigma}_3$ `in` $succ(\tilde{\varsigma}_2)$ `Propagate`$(\tilde{\varsigma}_1,\ \tilde{\varsigma}_3)$
    `case` $\tilde{\varsigma}_2$ `of Call`
      `for each` $\tilde{\varsigma}_3$ `in` $succ(\tilde{\varsigma}_2)$
        `Propagate`$(\tilde{\varsigma}_3,\ \tilde{\varsigma}_3)$
        `insert` $(\tilde{\varsigma}_1,\ \tilde{\varsigma}_2,\ \tilde{\varsigma}_3)$ `in` $Callers$
        `for each` $(\tilde{\varsigma}_3,\ \tilde{\varsigma}_4)$ `in` $Summary$ `Update`$(\tilde{\varsigma}_1,\ \tilde{\varsigma}_2,\ \tilde{\varsigma}_3,\ \tilde{\varsigma}_4)$
    `case` $\tilde{\varsigma}_2$ `of Exit-CEval`
      `if` $\tilde{\varsigma}_1 = \tilde{\mathcal{I}}(pr)$ `then`
        `Final`$(\tilde{\varsigma}_2)$
      `else`
        `insert` $(\tilde{\varsigma}_1,\ \tilde{\varsigma}_2)$ `in` $Summary$
        `for each` $(\tilde{\varsigma}_3,\ \tilde{\varsigma}_4,\ \tilde{\varsigma}_1)$ `in` $Callers$ `Update`$(\tilde{\varsigma}_3,\ \tilde{\varsigma}_4,\ \tilde{\varsigma}_1,\ \tilde{\varsigma}_2)$
        `for each` $(\tilde{\varsigma}_3,\ \tilde{\varsigma}_4,\ \tilde{\varsigma}_1)$ `in` $TCallers$ `Propagate`$(\tilde{\varsigma}_3,\ \tilde{\varsigma}_2)$
    `case` $\tilde{\varsigma}_2$ `of Exit-TC`
      `for each` $\tilde{\varsigma}_3$ `in` $succ(\tilde{\varsigma}_2)$
        `Propagate`$(\tilde{\varsigma}_3,\ \tilde{\varsigma}_3)$
        `insert` $(\tilde{\varsigma}_1,\ \tilde{\varsigma}_2,\ \tilde{\varsigma}_3)$ `in` $TCallers$
        `for each` $(\tilde{\varsigma}_3,\ \tilde{\varsigma}_4)$ `in` $Summary$ `Propagate`$(\tilde{\varsigma}_1,\ \tilde{\varsigma}_4)$

`Propagate`$(\tilde{\varsigma}_1,\ \tilde{\varsigma}_2) \triangleq$
  `if` $(\tilde{\varsigma}_1,\ \tilde{\varsigma}_2)$ `not in` $Seen$ `then insert` $(\tilde{\varsigma}_1,\ \tilde{\varsigma}_2)$ `in` $Seen$ `and` W

`Update`$(\tilde{\varsigma}_1,\ \tilde{\varsigma}_2,\ \tilde{\varsigma}_3,\ \tilde{\varsigma}_4) \triangleq$
  $\tilde{\varsigma}_1$ `of the form` $([\![(\lambda_{l_1}(u_1\ k_1)\ call_1)]\!],\ \hat{\mathrm{d}}_1,\ h_1)$
  $\tilde{\varsigma}_2$ `of the form` $([\![(f\ e_2\ (\lambda_{\gamma_2}\ (u_2)\ call_2))^{l_2}]\!],\ tf_2,\ h_2)$
  $\tilde{\varsigma}_3$ `of the form` $([\![(\lambda_{l_3}(u_3\ k_3)\ call_3)]\!],\ \hat{\mathrm{d}}_3,\ h_2)$
  $\tilde{\varsigma}_4$ `of the form` $([\![(k_4\ e_4)^{\gamma_4}]\!],\ tf_4,\ h_4)$
  $\hat{\mathrm{d}} \longleftarrow \hat{\mathcal{A}}_u(e_4,\ \gamma_4,\ tf_4,\ h_4)$
  $tf \longleftarrow \begin{cases} tf_2[f \mapsto \{[\![(\lambda_{l_3}(u_3\ k_3)\ call_3)]\!]\}] & S_?(l_2, f) \\ tf_2 & H_?(l_2, f) \vee Lam_?(f) \end{cases}$
  $\tilde{\varsigma} \longleftarrow ([\![(\lambda_{\gamma_2}(u_2)\ call_2)]\!],\ \hat{\mathrm{d}},\ tf,\ h_4)$
  `Propagate`$(\tilde{\varsigma}_1,\ \tilde{\varsigma})$

`Final`$(\tilde{\varsigma}) \triangleq$
  $\tilde{\varsigma}$ `of the form` $([\![(k\ e)^{\gamma}]\!],\ tf,\ h)$
  `insert` $(halt,\ \hat{\mathcal{A}}_u(e,\ \gamma,\ tf,\ h),\ \emptyset,\ h)$ `in` $Final$

Figure 8: CFA2 workset algorithm

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

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

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

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

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

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

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

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

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

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

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

$$G' = (S \cup \Delta S, \Gamma, E', q_0)$$
$$G'_\epsilon = (S', H')$$
$$\Delta G' = (\Delta S' - S', \Delta E' - E').$$

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

$Summary$, $Callers$, $TCallers$, $Final \longleftarrow \emptyset$
$Seen$, $W \longleftarrow \{(\tilde{\mathcal{I}}(pr),\ \tilde{\mathcal{I}}(pr))\}$
while $W \neq \emptyset$
  remove $(\tilde\varsigma_1, \tilde\varsigma_2)$ from W
  switch $\tilde\varsigma_2$
    case $\tilde\varsigma_2$ of Entry, CApply, Inner-CEval
      for each $\tilde\varsigma_3$ in $succ(\tilde\varsigma_2)$ Propagate($\tilde\varsigma_1$, $\tilde\varsigma_3$)
    case $\tilde\varsigma_2$ of Call
      for each $\tilde\varsigma_3$ in $succ(\tilde\varsigma_2)$
        Propagate($\tilde\varsigma_3$, $\tilde\varsigma_3$)
        insert $(\tilde\varsigma_1, \tilde\varsigma_2, \tilde\varsigma_3)$ in $Callers$
        for each $(\tilde\varsigma_3, \tilde\varsigma_4)$ in $Summary$ Update($\tilde\varsigma_1$, $\tilde\varsigma_2$, $\tilde\varsigma_3$, $\tilde\varsigma_4$)
    case $\tilde\varsigma_2$ of Exit-CEval
      if $\tilde\varsigma_1 = \tilde{\mathcal{I}}(pr)$ then
        Final($\tilde\varsigma_2$)
      else
        insert $(\tilde\varsigma_1, \tilde\varsigma_2)$ in $Summary$
        for each $(\tilde\varsigma_3, \tilde\varsigma_4, \tilde\varsigma_1)$ in $Callers$ Update($\tilde\varsigma_3$, $\tilde\varsigma_4$, $\tilde\varsigma_1$, $\tilde\varsigma_2$)
        for each $(\tilde\varsigma_3, \tilde\varsigma_4, \tilde\varsigma_1)$ in $TCallers$ Propagate($\tilde\varsigma_3$, $\tilde\varsigma_2$)
    case $\tilde\varsigma_2$ of Exit-TC
      for each $\tilde\varsigma_3$ in $succ(\tilde\varsigma_2)$
        Propagate($\tilde\varsigma_3$, $\tilde\varsigma_3$)
        insert $(\tilde\varsigma_1, \tilde\varsigma_2, \tilde\varsigma_3)$ in $TCallers$
        for each $(\tilde\varsigma_3, \tilde\varsigma_4)$ in $Summary$ Propagate($\tilde\varsigma_1$, $\tilde\varsigma_4$)

Propagate($\tilde\varsigma_1$, $\tilde\varsigma_2$) $\triangleq$
  if $(\tilde\varsigma_1, \tilde\varsigma_2)$ not in $Seen$ then insert $(\tilde\varsigma_1, \tilde\varsigma_2)$ in $Seen$ and W

Update($\tilde\varsigma_1$, $\tilde\varsigma_2$, $\tilde\varsigma_3$, $\tilde\varsigma_4$) $\triangleq$
  $\tilde\varsigma_1$ of the form $([\![(\lambda_{l_1}(u_1\ k_1)\ call_1)]\!],\ \hat{\mathrm{d}}_1,\ h_1)$
  $\tilde\varsigma_2$ of the form $([\![(f\ e_2\ (\lambda_{\gamma_2}\ (u_2)\ call_2))^{l_2}]\!],\ tf_2,\ h_2)$
  $\tilde\varsigma_3$ of the form $([\![(\lambda_{l_3}(u_3\ k_3)\ call_3)]\!],\ \hat{\mathrm{d}}_3,\ h_2)$
  $\tilde\varsigma_4$ of the form $([\![(k_4\ e_4)^{\gamma_4}]\!],\ tf_4,\ h_4)$
  $\hat{\mathrm{d}} \longleftarrow \hat{\mathcal{A}}_u(e_4,\ \gamma_4,\ tf_4,\ h_4)$
  $tf \longleftarrow \begin{cases} tf_2[f \mapsto \{[\![(\lambda_{l_3}(u_3\ k_3)\ call_3)]\!]\}] & S_?(l_2, f) \\ tf_2 & H_?(l_2, f) \vee Lam_?(f) \end{cases}$
  $\tilde\varsigma \longleftarrow ([\![(\lambda_{\gamma_2}(u_2)\ call_2)]\!],\ \hat{\mathrm{d}},\ tf,\ h_4)$
  Propagate($\tilde\varsigma_1$, $\tilde\varsigma$)

Final($\tilde\varsigma$) $\triangleq$
  $\tilde\varsigma$ of the form $([\![(k\,e)^{\gamma}]\!],\ tf,\ h)$
  insert $(halt,\ \hat{\mathcal{A}}_u(e,\ \gamma,\ tf,\ h),\ \emptyset,\ h)$ in $Final$

Figure 8: CFA2 workset algorithm

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

$$\hat T_+ = \left\{ (\hat\psi \overset{\hat\phi_+}{\rightarrowtail} \hat\psi', \hat\sigma') : \hat\psi \underset{\hat\phi_+}{\overset{\hat\sigma}{\rightarrow}} (\hat\psi', \hat\sigma') \right\}$$

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

$$\hat T_- = \{ (\hat\psi'' \overset{\hat\phi_-}{\rightarrowtail} \hat\psi''', \hat\sigma') : \hat\psi'' \underset{\hat\phi_-}{\overset{\hat\sigma}{\rightarrow}} (\hat\psi''', \hat\sigma') \text{ and } \hat\psi \overset{\hat\phi_+}{\rightarrowtail} \hat\psi' \in \hat E \text{ and } \hat\psi' \rightarrowtail \hat\psi'' \in \hat H \}$$

$$\hat T' = \hat T_+ \cup \hat T_\epsilon \cup \hat T_-$$

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

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

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

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

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

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

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

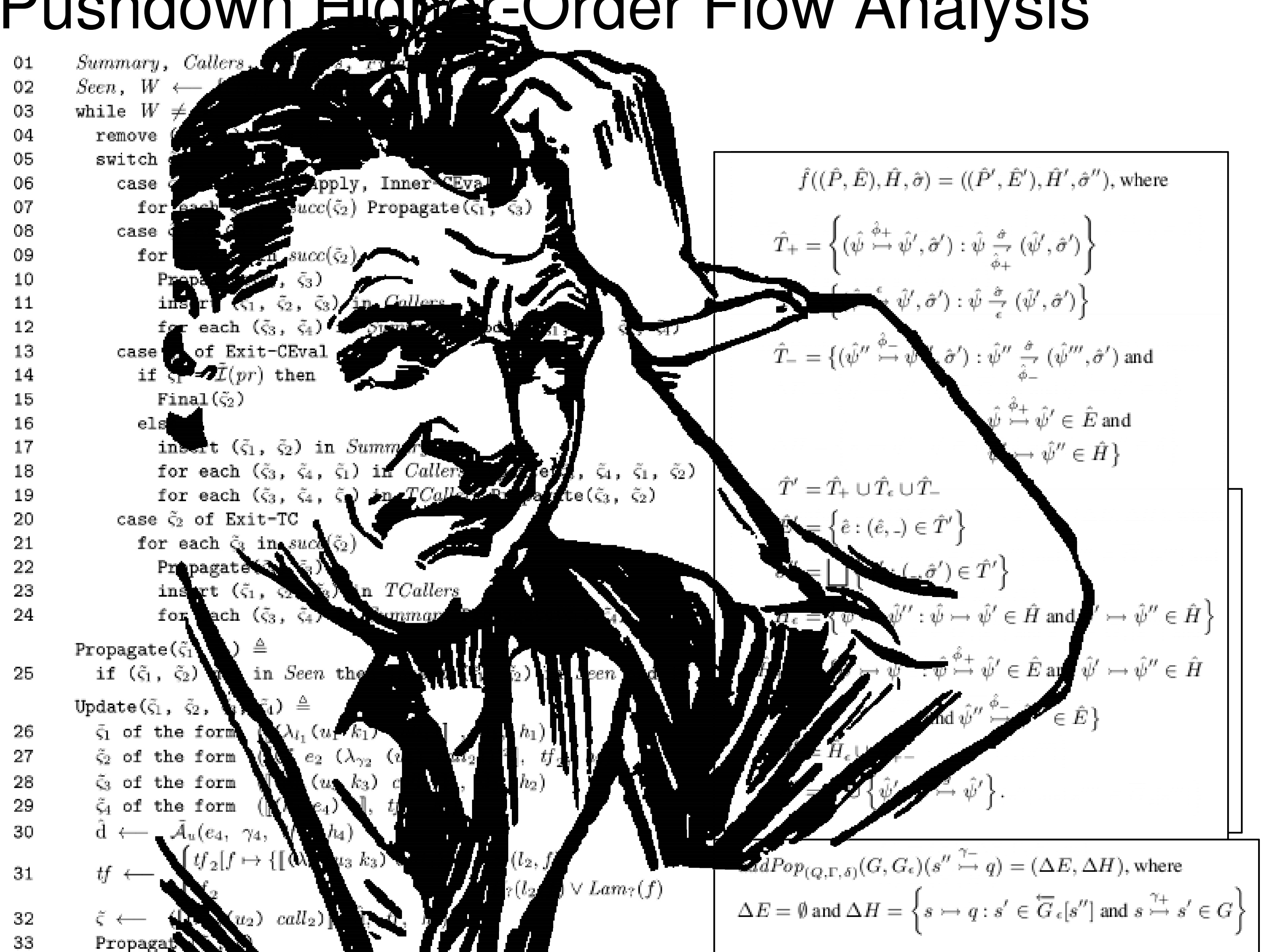

Figure 3. The fixed point of the function $\mathcal{F}(M)$ contains the Dyck state graph of the rooted pushdown system M.

Figure 8: CFA2 workset algorithm

Pushdown Higher-Order Flow Analysis

$Summary$, $Callers$, $TCallers$, $Final \longleftarrow \emptyset$
$Seen$, $W \longleftarrow \{(\tilde{\mathcal{I}}(pr),\ \tilde{\mathcal{I}}(pr))\}$
while $W \neq \emptyset$
  remove $(\tilde{\varsigma}_1, \tilde{\varsigma}_2)$ from W
  switch $\tilde{\varsigma}_2$
    case $\tilde{\varsigma}_2$ of Entry, CApply, Inner-CEval
      for each $\tilde{\varsigma}_3$ in $succ(\tilde{\varsigma}_2)$ Propagate($\tilde{\varsigma}_1$, $\tilde{\varsigma}_3$)
    case $\tilde{\varsigma}_2$ of Call
      for each $\tilde{\varsigma}_3$ in $succ(\tilde{\varsigma}_2)$
        Propagate($\tilde{\varsigma}_3$, $\tilde{\varsigma}_3$)
        insert $(\tilde{\varsigma}_1, \tilde{\varsigma}_2, \tilde{\varsigma}_3)$ in $Callers$
        for each $(\tilde{\varsigma}_3, \tilde{\varsigma}_4)$ in $Summary$ Update($\tilde{\varsigma}_1$, $\tilde{\varsigma}_2$, $\tilde{\varsigma}_3$, $\tilde{\varsigma}_4$)
    case $\tilde{\varsigma}_2$ of Exit-CEval
      if $\tilde{\varsigma}_1 = \tilde{\mathcal{I}}(pr)$ then
        Final($\tilde{\varsigma}_2$)
      else
        insert $(\tilde{\varsigma}_1, \tilde{\varsigma}_2)$ in $Summary$
        for each $(\tilde{\varsigma}_3, \tilde{\varsigma}_4, \tilde{\varsigma}_1)$ in $Callers$ Update($\tilde{\varsigma}_3$, $\tilde{\varsigma}_4$, $\tilde{\varsigma}_1$, $\tilde{\varsigma}_2$)
        for each $(\tilde{\varsigma}_3, \tilde{\varsigma}_4, \tilde{\varsigma}_1)$ in $TCallers$ Propagate($\tilde{\varsigma}_3$, $\tilde{\varsigma}_2$)
    case $\tilde{\varsigma}_2$ of Exit-TC
      for each $\tilde{\varsigma}_3$ in $succ(\tilde{\varsigma}_2)$
        Propagate($\tilde{\varsigma}_3$, $\tilde{\varsigma}_3$)
        insert $(\tilde{\varsigma}_1, \tilde{\varsigma}_2, \tilde{\varsigma}_3)$ in $TCallers$
        for each $(\tilde{\varsigma}_3, \tilde{\varsigma}_4)$ in $Summary$ Propagate($\tilde{\varsigma}_1$, $\tilde{\varsigma}_4$)

Propagate($\tilde{\varsigma}_1$, $\tilde{\varsigma}_2$) $\triangleq$
  if $(\tilde{\varsigma}_1, \tilde{\varsigma}_2)$ not in $Seen$ then insert $(\tilde{\varsigma}_1, \tilde{\varsigma}_2)$ in $Seen$ and W

Update($\tilde{\varsigma}_1$, $\tilde{\varsigma}_2$, $\tilde{\varsigma}_3$, $\tilde{\varsigma}_4$) $\triangleq$
  $\tilde{\varsigma}_1$ of the form $([\![(\lambda_{l_1}(u_1\ k_1)\ call_1)]\!],\ \hat{d}_1,\ h_1)$
  $\tilde{\varsigma}_2$ of the form $([\![(f\ e_2\ (\lambda_{\gamma_2}\ (u_2)\ call_2))^{l_2}]\!],\ tf_2,\ h_2)$
  $\tilde{\varsigma}_3$ of the form $([\![(\lambda_{l_3}(u_3\ k_3)\ call_3)]\!],\ \hat{d}_3,\ h_2)$
  $\tilde{\varsigma}_4$ of the form $([\![(k_4\ e_4)^{\gamma_4}]\!],\ tf_4,\ h_4)$
  $\hat{d} \longleftarrow \hat{\mathcal{A}}_u(e_4,\ \gamma_4,\ tf_4,\ h_4)$
  $tf \longleftarrow \begin{cases} tf_2[f \mapsto \{[\![(\lambda_{l_3}(u_3\ k_3)\ call_3)]\!]\}] & S_?(l_2, f) \\ tf_2 & H_?(l_2, f) \vee Lam_?(f) \end{cases}$
  $\tilde{\varsigma} \longleftarrow ([\![(\lambda_{\gamma_2}(u_2)\ call_2)]\!],\ \hat{d},\ tf,\ h_4)$
  Propagate($\tilde{\varsigma}_1$, $\tilde{\varsigma}$)

Final($\tilde{\varsigma}$) $\triangleq$
  $\tilde{\varsigma}$ of the form $([\![(k\ e)^{\gamma}]\!],\ tf,\ h)$
  insert $(halt,\ \hat{\mathcal{A}}_u(e,\ \gamma,\ tf,\ h),\ \emptyset,\ h)$ in $Final$

Figure 8: CFA2 workset algorithm

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

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

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

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

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

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

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

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

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

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

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

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

Analysis is ~~terrible~~ salvagable

- Annoying false positives 
- ~~Takes too long~~ 
- Hard to implement 
- Hard to show correct 
- ~~The good ones inherently first-order~~ 

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
- Simple transform: heap-allocate data

Deriving Pushdown Analyses

- Start: Concrete machine semantics
- Simple transform: memoize functions
- Simple transform: store functions’ calling contexts
- Simple transform: heap-allocate data
- Analysis: pointwise-abstraction

Deriving Pushdown Analyses

- Start: Concrete machine semantics
- Simple transform: memoize functions
- Simple transform: store functions' calling contexts
- Simple transform: heap-allocate data
- Analysis: pointwise-abstraction

Gives semantic account of summaries

Analysis is ~~terrible~~ salvagable

- Annoying false positives
- ~~Takes too long~~
- ~~Hard to implement~~
- ~~Hard to show correct~~
- ~~The good ones inherently first-order~~

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

Environment:

```
id   ...
app  ...
```

Memo

Contexts

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

Environment:

```
id  ...
f   id
y   1
```

Memo

Contexts

```
⟨app id 1⟩ (let* (... [n1 •] ...) ...)
```

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

Environment:

`x 1`

Memo

Contexts

```
⟨app id 1⟩  (let* (... [n1 •] ...) ...)
⟨id 1⟩      (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))
```

Environment:

Memo

⟨id 1⟩ 1

Contexts

⟨app id 1⟩ (let* (... [n1 •] ...) ...)

⟨id 1⟩ (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))
```

Environment:

Memo

```
⟨id 1⟩      1
⟨app id 1⟩  1
```

Contexts

```
⟨app id 1⟩  (let* (... [n1 •] ...) ...)
⟨id 1⟩      (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))
```

Environment:

```
id   ...
app  ...
n1   1
```

Memo

```
⟨id 1⟩      1
⟨app id 1⟩  1
```

Contexts

```
⟨app id 1⟩  (let* (... [n1 •] ...) ...)
⟨id 1⟩      (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))
```

Environment:

id	...
f	id
y	2

Memo

⟨id 1⟩	1
⟨app id 1⟩	1

Contexts

⟨app id 1⟩	(let* (... [n1 •] ...) ...)
⟨id 1⟩	(let* (... [app (λ (f y) •)] ...) ...)
⟨app id 2⟩	(let* (... [n2 •]) ...)

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

Environment:

x 2

Memo

⟨id 1⟩	1
⟨app id 1⟩	1

Contexts

⟨app id 1⟩	`(let* (... [n1 •] ...) ...)`
⟨id 1⟩	`(let* (... [app (λ (f y) •)] ...) ...)`
⟨app id 2⟩	`(let* (... [n2 •]) ...)`
⟨id 2⟩	`(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))
```

Environment:

Memo

⟨id 1⟩	1
⟨app id 1⟩	1
⟨id 2⟩	2

Contexts

⟨app id 1⟩	(let* (... [n1 •] ...) ...)
⟨id 1⟩	(let* (... [app (λ (f y) •)] ...) ...)
⟨app id 2⟩	(let* (... [n2 •]) ...)
⟨id 2⟩	(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))
```

Environment:

Memo

⟨id 1⟩	1
⟨app id 1⟩	1
⟨id 2⟩	2
⟨app id 2⟩	2

Contexts

⟨app id 1⟩	(let* (... [n1 •] ...) ...)
⟨id 1⟩	(let* (... [app (λ (f y) •)] ...) ...)
⟨app id 2⟩	(let* (... [n2 •]) ...)
⟨id 2⟩	(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))
```

Environment:

```
id   ...
app  ...
n1   1
n2   2
```

Memo

```
⟨id 1⟩       1
⟨app id 1⟩   1
⟨id 2⟩       2
⟨app id 2⟩   2
```

Contexts

```
⟨app id 1⟩  (let* (... [n1 •] ...) ...)
⟨id 1⟩      (let* (... [app (λ (f y) •)] ...) ...)
⟨app id 2⟩  (let* (... [n2 •]) ...)
⟨id 2⟩      (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_0

Memo

Start Store: σ_0

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	id
y	1

Memo

Start Store: σ_1

Contexts

⟨app σ_1 ⟩ ((let* (... [n1 •] ...) ...) σ_0)

```
(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	id
y	1
x	1

Memo

Start Store: σ_2

Contexts

⟨app σ_1 ⟩ ((let* (... [n1 •] ...) ...) σ_0)

⟨id σ_2 ⟩ ((let* (... [app (λ (f y) •)] ...) ...) σ_1)

```
(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	id
y	1
x	1

Memo

⟨id σ_2⟩ (1 σ_2)

Start Store: σ_2

Contexts

⟨app σ_1 ⟩ ((let* (... [n1 •] ...) ...) σ_0)

⟨id σ_2 ⟩ ((let* (... [app (λ (f y) •)] ...) ...) σ_1)

```
(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	id
y	1
x	1

Memo

⟨id σ_2⟩ (1 σ_2)

⟨app σ_1⟩ (1 σ_2)

Start Store: σ_1

Contexts

⟨app σ_1 ⟩ ((let* (... [n1 •] ...) ...) σ_0)

⟨id σ_2 ⟩ ((let* (... [app (λ (f y) •)] ...) ...) σ_1)

```
(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	id
y	1
x	1
n1	1

Memo

⟨id σ_2⟩ (1 σ_2)

⟨app σ_1⟩ (1 σ_2)

Start Store: σ_0

Contexts

⟨app σ_1 ⟩ ((let* (... [n1 •] ...) ...) σ_0)

⟨id σ_2 ⟩ ((let* (... [app (λ (f y) •)] ...) ...) σ_1)

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

Store: σ4

f	id
y	(1 2)
x	1
n1	1

Memo

⟨id σ2⟩ (1 σ2)

⟨app σ1⟩ (1 σ2)

Start Store: σ4

Contexts

⟨app σ1 ⟩ ((let* (... [n1 •] ...) ...) σ0)

⟨id σ2 ⟩ ((let* (... [app (λ (f y) •)] ...) ...) σ1)

⟨app σ4 ⟩ ((let* (... [n2 •]) ...) σ0)

```
(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	id
y	(1 2)
x	(1 2)
n1	1

Memo

⟨id σ_2⟩ (1 σ_2)

⟨app σ_1⟩ (1 σ_2)

Start Store: σ_5

Contexts

⟨app σ_1 ⟩ ((let* (... [n1 •] ...) ...) σ_0)

⟨id σ_2 ⟩ ((let* (... [app (λ (f y) •)] ...) ...) σ_1)

⟨app σ_4 ⟩ ((let* (... [n2 •]) ...) σ_0)

⟨id σ_5 ⟩ ((let* (... [app (λ (f y) •)] ...) ...) σ_4)

```
(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	id
y	(1 2)
x	(1 2)
n1	1

Memo

⟨id σ_2⟩	(1 σ_2)
⟨app σ_1⟩	(1 σ_2)
⟨id σ_5⟩	(2 σ_5)

Start Store: σ_5

Contexts

⟨app σ_1 ⟩	((let* (... [n1 •] ...) ...) σ_0)
⟨id σ_2 ⟩	((let* (... [app (λ (f y) •)] ...) ...) σ_1)
⟨app σ_4 ⟩	((let* (... [n2 •]) ...) σ_0)
⟨id σ_5 ⟩	((let* (... [app (λ (f y) •)] ...) ...) σ_4)

```
(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	id
y	(1 2)
x	(1 2)
n1	1

Start Store: σ_4

Memo

$\langle$id $\sigma_2\rangle$	(1 σ_2)
$\langle$app $\sigma_1\rangle$	(1 σ_2)
$\langle$id $\sigma_5\rangle$	(2 σ_5)
$\langle$app $\sigma_4\rangle$	(2 σ_5)

Contexts

$\langle$app σ_1 $\rangle$	((let* (... [n1 •] ...) ...) σ_0)
$\langle$id σ_2 $\rangle$	((let* (... [app (λ (f y) •)] ...) ...) σ_1)
$\langle$app σ_4 $\rangle$	((let* (... [n2 •]) ...) σ_0)
$\langle$id σ_5 $\rangle$	((let* (... [app (λ (f y) •)] ...) ...) σ_4)

```
(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	id
y	(1 2)
x	(1 2)
n1	1
n2	(1 2)

Start Store: σ_0

Memo

$\langle$id $\sigma_2\rangle$	(1 σ_2)
$\langle$app $\sigma_1\rangle$	(1 σ_2)
$\langle$id $\sigma_5\rangle$	(2 σ_5)
$\langle$app $\sigma_4\rangle$	(2 σ_5)

Contexts

$\langle$app σ_1 $\rangle$	((let* (... [n1 •] ...) ...) σ_0)
$\langle$id σ_2 $\rangle$	((let* (... [app (λ (f y) •)] ...) ...) σ_1)
$\langle$app σ_4 $\rangle$	((let* (... [n2 •]) ...) σ_0)
$\langle$id σ_5 $\rangle$	((let* (... [app (λ (f y) •)] ...) ...) σ_4)

Heaps are imprecise

```
(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	id
y	(1 2)
x	(1 2)
n1	1
n2	(1 2)

Start Store: σ_0

Memo

⟨id σ_2⟩	(1 σ_2)
⟨app σ_1⟩	(1 σ_2)
⟨id σ_5⟩	(2 σ_5)
⟨app σ_4⟩	(2 σ_5)

Contexts

⟨app σ_1 ⟩	((let* (... [n1 •] ...) ...) σ_0)
⟨id σ_2 ⟩	((let* (... [app (λ (f y) •)] ...) ...) σ_1)
⟨app σ_4 ⟩	((let* (... [n2 •]) ...) σ_0)
⟨id σ_5 ⟩	((let* (... [app (λ (f y) •)] ...) ...) σ_4)

Fixed[Vardoulakis & Shivers 10]

Local storage (stack frames)

```
(define (foldr f b lst)
  (letrec
    ([loop
      (λ (lst)
       (cond [(empty? lst) b]
             [else (cons (f (car lst))
                         (loop (cdr lst)))]))])
    (loop lst)))
```

```
(define (foldr f b lst)
  (letrec
    ([loop
      (λ (lst)
       (cond [(empty? lst) b]
             [else (cons (f (car lst))
                         (loop (cdr lst)))]))])
    (loop lst)))
```

```
(define (foldr f b lst)
  (letrec
    ([loop
      (λ (lst)
       (cond [(empty? lst) b]
             [else (cons (f (car lst))
                         (loop (cdr lst)))]))])
    (loop lst)))
```

Fixed

Instrument your escape analysis

Escape Analysis

Which addresses outlive their creator's context?

Escape Analysis

Which addresses outlive their creator's context?

We introduce three components:

- Escaped addresses $\mathcal{E}$
- Addresses owned by current function $\mathscr{O}$
- The random access stack Ξ

Escape Analysis

Which addresses outlive their creator's context?

We introduce three components:

- Escaped addresses $\mathcal{E}$
- Addresses owned by current function $\mathscr{O}$
- The random access stack Ξ

Lookup is now defined with

$$\mathcal{L}(a,\sigma,\Xi,\mathcal{E}) = a \in \mathcal{E} \rightarrow \sigma(a),\ \Xi(a)$$

$$\langle (x, \rho), \sigma, \Xi, \kappa \rangle \longmapsto \langle v, \sigma, \Xi, \kappa \rangle$$

where $v \in \mathcal{L}(\rho(x), \sigma, \Xi, \mathcal{E})$

$$\langle (x, \rho), \sigma, \Xi, \kappa \rangle \longmapsto \langle v, \sigma, \Xi, \kappa \rangle$$

where $v \in \mathcal{L}(\rho(x), \sigma, \Xi, \mathcal{E})$

$$\langle v, \sigma, \Xi, \mathrm{fn}((\lambda\, x.e, \rho), \kappa) \rangle \longmapsto \langle c, \sigma next, \Xi[a \mapsto v], []^{c} \rangle$$

where $c = (e, \rho[x \mapsto a])$

$\sigma next = \sigma[a \mapsto v]$

$\mathcal{E}' = \mathcal{E} \cap \mathcal{R}(\mathscr{T}(c), \sigma next)$

$$\langle (x, \rho), \sigma, \Xi, \kappa \rangle \longmapsto \langle v, \sigma, \Xi, \kappa \rangle$$

where $v \in \mathcal{L}(\rho(x), \sigma, \Xi, \mathcal{E})$

$$\langle v, \sigma, \Xi, \mathbf{fn}((\lambda\, x.e, \rho), \kappa) \rangle \longmapsto \langle c, \sigma next, \Xi[a \mapsto v], []^c \rangle$$

where $c = (e, \rho[x \mapsto a])$

$\sigma next = \sigma[a \mapsto v]$

$\mathcal{E}' = \mathcal{E} \cap \mathcal{R}(\mathscr{T}(c), \sigma next)$

$\mathscr{O}' = \{a\}$

$$\langle (x, \rho), \sigma, \Xi, \kappa \rangle \longmapsto \langle v, \sigma, \Xi, \kappa \rangle$$

where $v \in \mathcal{L}(\rho(x), \sigma, \Xi, \mathcal{E})$

$$\langle v, \sigma, \Xi, \mathrm{fn}((\lambda\, x.e, \rho), \kappa) \rangle \longmapsto \langle c, \sigma next, \Xi[a \mapsto v], []^{c} \rangle$$

where $c = (e, \rho[x \mapsto a])$

$\sigma next \;=\; \sigma[a \mapsto v]$

$\mathcal{E}' = \mathcal{E} \cap \mathcal{R}(\mathscr{T}(c), \sigma next)$

$\mathscr{O}' = \{a\}$

$L' \;=\; L \sqcup [(c, \sigma next) \mapsto (\kappa, \sigma cur, \Xi, \mathcal{E}, \mathscr{O})]$

$$\langle v,\sigma,\Xi,[\,]^c\rangle \longmapsto \langle v,\sigma,\Xi',\kappa\rangle$$

$$\text{if } (\kappa, \sigma next, \Xi', \mathcal{E}', \mathcal{O}') \in L(c, \sigma cur)$$

$$M' = M[(c, \sigma cur) \mapsto (v, \sigma)]$$

$$\mathcal{E}'' = \mathcal{E}' \cup ((\mathcal{E} \cup \mathcal{O}) \cap \mathcal{R}(\mathcal{T}(v),\sigma))$$

Analysis is ~~terrible~~ salvagable

- ~~Annoying false positives~~
- Takes too long
- ~~Hard to implement~~
- ~~Hard to show correct~~
- ~~The good ones inherently first-order~~

We're not done yet

Too context-sensitive

```
(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	id
y	(1 2)
x	(1 2)
n1	1
n2	(1 2)

Start Store: σ_0

Memo

⟨id σ_2⟩	(1 σ_2)
⟨app σ_1⟩	(1 σ_2)
⟨id σ_5⟩	(2 σ_5)
⟨app σ_4⟩	(2 σ_5)

Contexts

⟨app σ_1 ⟩	((let* (... [n1 •] ...) ...) σ_0)
⟨id σ_2 ⟩	((let* (... [app (λ (f y) •)] ...) ...) σ_1)
⟨app σ_4 ⟩	((let* (... [n2 •]) ...) σ_0)
⟨id σ_5 ⟩	((let* (... [app (λ (f y) •)] ...) ...) σ_4)

First-Order Influence

First-order

(Hecht 77) CFG + lattice

(Sharir & Pnueli 81) CFG + lattice + summaries

(Reps 95) PDS + idempotent semiring

Higher-order

(Jones & Muchnick 82) Program + set constraints

(Shivers 91) CPS + set constraints

(Might 07) CPS + abstract small step semantics

(Might & Van Horn 10) Derived from concrete

(Vardoulakis & Shivers 10) CPS + summaries

(Earl et al. 10) ANF + Dyck state graph

Today's talk

?

First-Order Influence

Higher-order

(Jones & Muchnick 82) Program + set constraints

(Shivers 91) CPS + set constraints

(Might 07) CPS + abstract small step semantics

(Might & Van Horn 10) Derived from concrete

(Vardoulakis & Shivers 10) CPS + summaries

(Earl et al. 10) ANF + Dyck state graph

Today's talk

?

First-order

(Hecht 77) CFG + lattice

(Sharir & Pnueli 81) CFG + lattice + summaries

(Reps 95) PDS + idempotent semiring

(Oh et al. 12) CFG + lattice + bypassing

Higher-order

(Jones & Muchnick 82) Program + set constraints

(Shivers 91) CPS + set constraints

(Might 07) CPS + abstract small step semantics

(Might & Van Horn 10) Derived from concrete

(Vardoulakis & Shivers 10) CPS + summaries

(Earl et al. 10) ANF + Dyck state graph

Today's talk

?

First-order

(Hecht 77) CFG + lattice

(Sharir & Pnueli 81) CFG + lattice + summaries

(Reps 95) PDS + idempotent semiring

(Oh et al. 12) CFG + lattice + bypassing

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
- Precise analyses: Stack allocation $\Longrightarrow$ precision

To Conclude

- Design: Model abstract mechanisms concretely
- Pushdown: Memo and local continuation tables
- Precise analyses: Stack allocation $\Longrightarrow$ precision
- (Not shown) works for control operators / GC

To Conclude

- Design: Model abstract mechanisms concretely
- Pushdown: Memo and local continuation tables
- Precise analyses: Stack allocation $\Longrightarrow$ precision
- (Not shown) works for control operators / GC
- Future work: sparse analysis via time travel

To Conclude

- Design: Model abstract mechanisms concretely
- Pushdown: Memo and local continuation tables
- Precise analyses: Stack allocation $\Longrightarrow$ precision
- (Not shown) works for control operators / GC
- Future work: sparse analysis via time travel

Thank you