Introspective Pushdown Analysis

Chris Earl
University of Utah
cwearl.com

Matt Might
University of Utah
matt.might.net
@mattmight

Ilya Sergey
KU Leuven
@ilyasergey

Dave Van Horn
Northeastern University
lambda-calcul.us
@lambda_calculus
“grad-school Vietnam”

“charred remains”

“would-be Ph.D.s”

-Olin Shivers
“academic War on Terror”

“unwinnable”

“never-ending”

-Me, to my grad students
What is flow analysis?
What is flow analysis?

What is wrong with it?
What is flow analysis?

What is wrong with it?

How do we fix it?
How do we fix the fixes?
We use fixed points.
What is flow analysis?
What is control-flow analysis?
(f x)
What is $f$?
Why not run the program?
CEK

(Felleisen and Friedman, 1986)
e
What is $f$, here?
What is \( f \) here?
$e$
Make it terminate?
Make it finite.
Make it finite.

(Might, SAS 2010)   (Van Horn and Might, ICFP 2010)
CEK
E = V \rightarrow \lambda \times E

СЕК
$E = V \rightarrow \lambda \times E$

CEK
E = V \rightarrow A

CEK
CEK
CEK
CESK
CESK

(Felleisen and Friedman, 1987)
CESK

$S = A \rightarrow \lambda \times E$
CESK

\[ S = A \rightarrow \lambda \times E \]
CESK
CESK*
CESK*
K* ⊂ A

CESK*

S = A → λ × E + K
CESK*
CESK*
$e$
What is $f$, here?
What’s wrong with flow analysis?
Control-flow forks.
Data-flow merges.
Why?
S = A → D
S = \hat{A} \rightarrow D
S = \hat{A} \rightarrow D
S = \hat{A} \rightarrow \hat{D}
\[ \hat{S} = \hat{A} \rightarrow P(\hat{D}) \]
(f x)
What is $f$?
f is or or
f is O or O or •
Problem?
Finite store.
Solution?
Toss garbage.
Toss garbage.

(Might & Shivers, ICFP 2006)
e, \hat{\rho}, \hat{\sigma}, \hat{\kappa}
$e, \hat{\rho}, \hat{\sigma}, \hat{\kappa}$
$e, \hat{\rho}, \hat{\sigma}, \hat{\kappa}$
$e, \hat{\rho}, \hat{\sigma}', \hat{\kappa}$
Control-flow forks.
Return-flow forks.
(foo)

(define (foo)
  ...)

(foo)
(foo)

(define (foo)
  ...)

(foo)
(define (foo)
  ...
)
(define (foo)
...
(foo)
(define (foo)
...
(foo)
(foo))
\( \hat{S} = \hat{A} \rightarrow P(\hat{D}) \)
\[ \hat{S} = \hat{A} \rightarrow \mathcal{P}(\lambda x \hat{E} + \hat{K}) \]
CESK*
CESK*
CES K
control state
control state + stack = pushdown
control state + stack = pushdown

(Vardoulakis and Shivers, CFA2)
(define (foo)
...
(foobar))
(define (foo) ...)

(foo)

(foo)
GC & pushdown?
Halt!
\[ \delta : Q \times \Delta \Gamma \rightarrow \mathcal{P}(Q) \]
\[ \delta : Q \times \Delta \Gamma \rightarrow \mathcal{P}(Q) \]
\[ \delta : Q \times \Delta \Gamma \times \Gamma^* \rightarrow \mathcal{P}(Q) \]
\[ \delta : Q \times \Delta \Gamma \times \Gamma^* \to \mathcal{P}(Q) \]
\[ \delta : Q \times \Delta \Gamma \times \mathcal{P}(\Gamma^*) \rightarrow \mathcal{P}(Q) \]
\[ \delta : Q \times \Delta \Gamma \times \mathcal{P}(\Gamma^*) \rightarrow \mathcal{P}(Q) \]
How do we fix the fixes?
\[\delta : Q \times \Delta \Gamma \times \mathcal{P}(\Gamma^*) \rightarrow \mathcal{P}(Q)\]
\( \mathcal{P}(\Gamma^*) \)
\mathcal{P}(\Gamma^*)
Result?
confused by non-tail-recursive loop structure. With both techniques
Figure 1.

Figure 1.

Figure 1.

Figure 1.

Figure 1.

Figure 1.

Figure 1.

Figure 1.

Figure 1.

Figure 1.

Figure 1.

Figure 1.

Figure 1.

Figure 1.

Figure 1.

Indeed with pushdown analysis and abstract GC: 77 states

It is therefore possible to fuse the full benefits of abstract garbage collection with pushdown analysis. The dramatic reduction in ab-

...in the abstract semantics, which are in turn, phrased in

(define (f n)
  (cond ((= n 0) 1)
        (else (+ (* n n) (g (- n 1))))))

Fortunately, abstract garbage collection does not need to arbitrarily

...define (id x) x)

...directly into the abstract semantics, which are in turn, phrased in

...define (id x) x)

...define (id x) x)

...define (id x) x)

...define (id x) x)

...define (id x) x)

...define (id x) x)

...define (id x) x)

...define (id x) x)

...define (id x) x)

...define (id x) x)

...define (id x) x)

...define (id x) x)
Abstract GC, the abstract transition graph shrinks by an order of magnitude with pushdown only: 139 states. For short durations, the abstract-GC-only is used to prove the decidability of control-state reachability.}

There are three strong secondary motivations: introspective pushdown systems, and recast abstract garbage collection systems. A small example to illuminate the strengths and weaknesses of both pushdown analysis and abstract garbage collection.
A pushdown flow analysis can be construed as computing the

\[ O(\|M\|^2 \cdot (\|\Gamma\|)^3) = O(\|M\|^3) \]

\[ A\text{ddr} = \text{Var} \times \text{Exp} \]

\[ \text{allocate}(v, (\{ f \text{ vars} \}, \rho, \delta)) = \begin{cases} (v, \{ f \text{ vars} \}) & \text{if } f \text{ is let-bound} \\ \text{otherwise} \end{cases} \]

\[ \delta \subseteq \delta' \text{ iff } \delta(\overline{a}) \subseteq \delta'(\overline{a}) \text{ for all } \overline{a} \in \text{dom}(\delta); \]

\[ (\lambda, \rho) \subseteq (\lambda, \rho') \text{ iff } \rho \subseteq \rho'; \]

\[ (v, e, \rho) \subseteq (v, e, \rho') \text{ iff } \rho \subseteq \rho'; \]

\[ \langle \dot{\phi}_1, \ldots, \dot{\phi}_n \rangle \subseteq \langle \dot{\phi}_1, \ldots, \dot{\phi}_n \rangle \text{ iff } \dot{\phi}_i \subseteq \dot{\phi}_i; \]

\[ \langle \dot{\phi}_1, \ldots, \dot{\phi}_n \rangle \subseteq \langle \dot{\phi}_1, \ldots, \dot{\phi}_n \rangle \text{ iff } \dot{\phi}_i \subseteq \dot{\phi}_i; \]

\[ \text{Theorem 4.1. If:} \]

\[ \alpha(c) \subseteq \varepsilon \text{ and } c \Rightarrow \varepsilon', \]

\[ \text{then there must exist } \varepsilon'' \in \text{Conf} \text{ such that:} \]

\[ \alpha(c') \subseteq \varepsilon'' \text{ and } c' \sim \varepsilon''. \]

\[ \text{Reachable}(e, \rho, \sigma, \delta) = \text{range}(\rho) \cup \text{StackRoot}(\delta) \]

\[ \text{Root}(e, \rho, \sigma, \delta) = \text{range}(\rho) \cup \text{StackRoot}(\delta) \]

\[ \text{StackRoot}((v_1, e_1, \rho_1), \ldots, (v_n, e_n, \rho_n)) = \bigcup \text{range}(\rho_i), \]

\[ \text{q}^{\bullet} \text{if there exists } \overline{q} \text{ such that: } q_{\overline{q}} \subseteq q \text{ and } \langle \rho, g, q', q' \rangle \in \delta, \]

\[ \text{where } q_{\overline{q}} \subseteq q \text{ if } \langle \overline{q}_1, e_1, \rho_1 \rangle \subseteq \ldots \subseteq \langle \overline{q}_n, e_n, \rho_n \rangle \subseteq \langle q, g, q', q' \rangle. \]

\[ \text{Theorem 7.1. DSg}(M) = \text{lip}(F(M)). \]
<table>
<thead>
<tr>
<th>Program</th>
<th>Exp</th>
<th>Var</th>
<th>( k )</th>
<th>( k )-CFA</th>
<th>( k )-PDCFA</th>
<th>( k )-CFA + GC</th>
</tr>
</thead>
<tbody>
<tr>
<td>mj09</td>
<td>19</td>
<td>8</td>
<td>0</td>
<td>83</td>
<td>107</td>
<td>4</td>
</tr>
<tr>
<td>eta</td>
<td>21</td>
<td>8</td>
<td>0</td>
<td>63</td>
<td>812</td>
<td>4</td>
</tr>
<tr>
<td>kcfa2</td>
<td>20</td>
<td>10</td>
<td>0</td>
<td>1</td>
<td>74</td>
<td>1</td>
</tr>
<tr>
<td>kcfa3</td>
<td>20</td>
<td>10</td>
<td>0</td>
<td>63</td>
<td>812</td>
<td>4</td>
</tr>
<tr>
<td>blur</td>
<td>25</td>
<td>13</td>
<td>1</td>
<td>194</td>
<td>33</td>
<td>4</td>
</tr>
<tr>
<td>loop2</td>
<td>40</td>
<td>13</td>
<td>0</td>
<td>327</td>
<td>236</td>
<td>8</td>
</tr>
<tr>
<td>sat</td>
<td>63</td>
<td>14</td>
<td>0</td>
<td>970</td>
<td>236</td>
<td>8</td>
</tr>
</tbody>
</table>

As expected, the fused analysis does at least as well as the best benchmarks are available: abstract garbage collection or both. Our implementation source and transitions/DSG edges computed during the analysis (for both less is better). Inequalities for some results denote the case when the analysis did not finish within 30 minutes. For such cases we can only report an upper bound of singleton variables as this number can only decrease.

![Figure 6.](image.png)

Program optimizability) and better than both in some cases. Also worthy of note is the dramatic reduction in the size of control states, i.e., how many variables have a single lambda flow to them (more is better).

![Figure 7.](image.png)

Interrupted due to the an execution time greater than 30 minutes. For each of the four analyses the left column denotes the values obtained with no abstract collection, and the right one—with GC on. The results of the analyses are compared to those of the pushdown analysis.

![Figure 8.](image.png)

Program Optimizability) and better than both in some cases. Also worthy of note is the dramatic reduction in the size of control states, i.e., how many variables have a single lambda flow to them (more is better).

![Figure 9.](image.png)

Given the already substantial reductions in analysis times provided...
http://github.com/ilyasergey/reachability
Finitization is double-edged.

Progress attacks finitization.

We can skirt inside decidability.
Tak.
Complexity?
• Monovariant, global store? Polynomial.

• Polyvariant? Exponential.

• Flat environments, global store? Polynomial.
Alternative?
\[ \delta : Q \times \Delta \Gamma \times \mathcal{P}(\Gamma^*) \rightarrow \mathcal{P}(Q) \]
\[ \delta : Q \times \Delta \Gamma \times \mathcal{P}(\Delta \Gamma) \rightarrow \mathcal{P}(Q) \]
\[ \delta : Q \times \mathcal{P}(\Delta \Gamma) \times \Delta \Gamma \rightarrow \mathcal{P}(Q) \]
control states

\[\delta : Q \times \mathcal{P}(\Delta \Gamma) \times \Delta \Gamma \rightarrow \mathcal{P}(Q)\]