Subcubic Control-Flow Analysis Algorithms

Jan Midtgaard and David Van Horn
Outline: Subcubic CFA

★ Inclusion-based CFA (0CFA)
★ History of the “Cubic bottleneck”
★ Indirect subcubic CFA algorithm
★ Cubic CFA algorithm
★ Fast sets
★ Direct subcubic CFA algorithm
★ Further improvements
Inclusion-based CFA
What is CFA?

An analysis to determine for each call site of a program, a set of functions which may be applied when the program is run. More generally, it is a computable approximation to evaluation.

Example: \((\lambda f. f f v)(\lambda x. x)\)
What is CFA?

An analysis to determine for each call site of a program, a set of functions which may be applied when the program is run.

More generally, it is a computable approximation to evaluation.

Example: \((\lambda f. f f v) (\lambda x. x)\)

\[\begin{align*}
\ast & \quad f \rightarrow \lambda x. x \\
\ast & \quad (\lambda f. f f v) (\lambda x. x) \rightarrow f f v
\end{align*}\]
What is CFA?

An analysis to determine for each call site of a program, a set of functions which may be applied when the program is run.

More generally, it is a computable approximation to evaluation.

Example: \((\lambda f. f f \, v) \, (\lambda x. \, x)\)

\[
\begin{align*}
\ast & \quad f \rightarrow \lambda x. \, x \\
\ast & \quad (\lambda f. f f \, v) \, (\lambda x. \, x) \rightarrow f f \, v \\
\ast & \quad x \rightarrow \lambda x. \, x \\
\ast & \quad f f \rightarrow x
\end{align*}
\]
What is CFA?

An analysis to determine for each call site of a program, a set of functions which may be applied when the program is run.

More generally, it is a computable approximation to evaluation.

Example: \((\lambda f. ff v) (\lambda x. x)\)

\[
\begin{align*}
&\star f \rightarrow \lambda x. x \\
&\star (\lambda f. ff v) (\lambda x. x) \rightarrow ff v \\
&\star x \rightarrow \lambda x. x \\
&\star ff \rightarrow x \\
&\star x \rightarrow v \\
&\star ff v \rightarrow x
\end{align*}
\]
What is CFA?

An analysis to determine for each call site of a program, a set of functions which may be applied when the program is run.

More generally, it is a computable approximation to evaluation.

Example: \((\lambda f. f f v) (\lambda x. x) \rightarrow \{(\lambda x. x), v\}\)

\[ \begin{align*}
\ast & \ f \rightarrow \lambda x. x \\
\ast & \ (\lambda f. f f v) (\lambda x. x) \rightarrow f f v \\
\ast & \ x \rightarrow \lambda x. x \\
\ast & \ f f \rightarrow x \\
\ast & \ x \rightarrow v \\
\ast & \ f f v \rightarrow x
\end{align*} \]
Language:

\[ e ::= x \mid \lambda x. e \mid e \; e \]  
(expressions)

Analysis:

\[
\begin{align*}
\text{input}(e) & \quad \quad \text{REFL} \quad \quad \text{input}(e_1 \; e_2) \quad e_1 \rightarrow \lambda x. e \\
\end{align*}
\]
\[
\begin{align*}
x \rightarrow e_2, \ e_1 \; e_2 \rightarrow e \\
\end{align*}
\]
\[
\begin{align*}
e_0 \rightarrow e_1 \quad e_1 \rightarrow e_2 \\
\end{align*}
\]
\[
\begin{align*}
e_0 \rightarrow e_2 \\
\end{align*}
\]  
(TRANS)
History of the “Cubic bottleneck”
Aho et al. (1968)

Time and tape complexity of pushdown automaton languages:

- Introduces the class “2NPDA”
- Gives a $O(n^3)$ algorithm for 2NPDA.
Flow analysis of lambda expressions:

★ Coined the term “control flow analysis” for the problem of approximating the control-flow of higher-order programs.
★ Formulated and proved sound CFA under CBV and CBN.
★ Precursor to what we now call 0CFA.
★ No algorithmic complexity analysis.
Jones (1981a,b)

Flow analysis of lambda expressions:

★ Coined the term “control flow analysis” for the problem of approximating the control-flow of higher-order programs.
★ Formulated and proved sound CFA under CBV and CBN.
★ Precursor to what we now call 0CFA.
★ No algorithmic complexity analysis.
Ayers (1992)

Efficient Closure Analysis with Reachability:

★ Gives an algorithm for CFA.
★ Proves it is $O(n^3)$. 
Heintze and McAllester (1997a)

On the Cubic Bottleneck in Subtyping and Flow Analysis:

★ Deciding “$e_1 \rightarrow e_2$” is complete for 2NPDA.

“The fact that [CFA problems] are hard for 2NPDA can be interpreted as evidence they can not be solved in sub-cubic time — the cubic time decision procedure for an arbitrary 2NPDA problem has not been improved since its discovery in 1968.”
Repeated Repeatedly

“The cubic time decision problem for 2NPDA has not been improved since its discovery in 1968.”

- Neal (1989)
- Heintze and McAllester (1997a), LICS
- Heintze and McAllester (1997b), PLDI
- Melski and Reps (2000), TCS
- McAllester (2002), JACM
- Van Horn and Mairson (2008), SAS
Unknown Knowns

“There are known knowns. There are known unknowns. But there are also unknown unknowns.”

— Donald Rumsfeld
Unknown Knowns

“There are known knowns. There are known unknowns. But there are also unknown unknowns.”

— Donald Rumsfeld

There are also unknown knowns.
Unknown Knowns: Rytter (1985)

“There are known knowns. There are known unknowns. But there are also unknown unknowns.”

— Donald Rumsfeld

There are also unknown knowns.

Fast recognition of pushdown automaton and context-free languages:

* Improved the 2NPDA algorithm to $O(n^3 / \lg n)$. 
Indirect subcubic CFA algorithm
Melski and Reps (2000)

Interconvertibility of a class of set constraints and context-free-language reachability:

★ Gives CFA-constraint to CFL-reachability reduction.
★ Gives a CFL-normalization algorithm.
★ CFL reduction preserves $O(n^3)$ solvability.
Subcubic algorithms for recursive state machines:

- Uses Rytter’s technique to obtain $O(n^3/\lg n)$ CFL-reachability algorithm.
Indirect subcubic CFA
Indirect subcubic CFA

☆ Reduce CFA to CFL (Melski and Reps 2000)
Indirect subcubic CFA

- Reduce CFA to CFL (Melski and Reps 2000)
- Normalize CFL grammar (ibid.)
Indirect subcubic CFA

★ Reduce CFA to CFL (Melski and Reps 2000)
★ Normalize CFL grammar (ibid.)
★ $O(n^3 / \lg n)$ CFL-reachability algorithm (Chaudhuri 2008)
Indirect subcubic CFA

★ Reduce CFA to CFL (Melski and Reps 2000)
★ Normalize CFL grammar (ibid.)
★ $O(n^3 / \lg n)$ CFL-reachability algorithm (Chaudhuri 2008)
★ Map results back to CFA.
Where’s the Beef?
Where’s the Beef?

What does all of this mean for CFA algorithms?
Cubic CFA algorithm
Cubic algorithm

CUBIC-CFA(e)

\[ Q, R \leftarrow \{ e \rightarrow e \mid \text{input}(e) \} \]

while \( Q \neq \emptyset \)

\[
\begin{align*}
\text{do } (e_i \rightarrow e_j) & \leftarrow \text{DELETE}(Q) \\
\text{for } e_k \rightarrow e_i \in R \text{ such that } (e_k \rightarrow e_j) & \notin R \\
\text{do } \text{INSERT}(e_k \rightarrow e_j, R) \\
& \text{INSERT}(e_k \rightarrow e_j, Q) \\
\text{for } e_j \rightarrow e_k \in R \text{ such that } (e_i \rightarrow e_k) & \notin R \\
\text{do } \text{INSERT}(e_i \rightarrow e_k, R) \\
& \text{INSERT}(e_i \rightarrow e_k, Q) \\
\text{if } e_j = \lambda x. e \text{ and } \text{input}(e_i e_w) \\
\text{then if } (x \rightarrow e_w) & \notin R \\
& \text{then } \text{INSERT}(x \rightarrow e_w, R) \\
& \text{INSERT}(x \rightarrow e_w, Q) \\
\text{if } (e_i e_w \rightarrow e) & \notin R \\
\text{then } \text{INSERT}(e_i e_w \rightarrow e, R) \\
& \text{INSERT}(e_i e_w \rightarrow e, Q)
\end{align*}
\]

return \( R \)
Cubic algorithm

\textbf{CUBIC-CFA}(e)

\[ Q, R \leftarrow \{ e \rightarrow e \mid \text{input}(e) \} \]

\[ \textbf{while } Q \neq \emptyset \]
\[ \textbf{do } (e_i \rightarrow e_j) \leftarrow \text{DELETE}(Q) \]
\[ \textbf{for } e_k \rightarrow e_i \in R \textbf{ such that } (e_k \rightarrow e_j) \notin R \]
\[ \textbf{do } \text{INSERT}(e_k \rightarrow e_j, R) \]
\[ \text{INSERT}(e_k \rightarrow e_j, Q) \]
\[ \textbf{for } e_j \rightarrow e_k \in R \textbf{ such that } (e_i \rightarrow e_k) \notin R \]
\[ \textbf{do } \text{INSERT}(e_i \rightarrow e_k, R) \]
\[ \text{INSERT}(e_i \rightarrow e_k, Q) \]
\[ \textbf{if } e_j = \lambda x. e \textbf{ and } \text{input}(e_i e_w) \]
\[ \textbf{then if } (x \rightarrow e_w) \notin R \]
\[ \textbf{then } \text{INSERT}(x \rightarrow e_w, R) \]
\[ \text{INSERT}(x \rightarrow e_w, Q) \]
\[ \textbf{if } (e_i e_w \rightarrow e) \notin R \]
\[ \textbf{then } \text{INSERT}(e_i e_w \rightarrow e, R) \]
\[ \text{INSERT}(e_i e_w \rightarrow e, Q) \]

\[ \textbf{return } R \]

\begin{itemize}
  \item Initialization is \( O(n) \).
  \item Edges inserted \textit{simultaneously} into \( Q, R \).
  \item Once in \( R \), never inserted in \( Q \) again.
  \item So edges inserted into \( Q \) at most once.
  \item At most \( O(n^2) \) while-loop iterations.
\end{itemize}
Cubic algorithm

\texttt{CUBIC-CFA}(e)

\begin{align*}
Q, R & \leftarrow \{ e \rightarrow e \mid \text{input}(e) \} \\
\text{while } Q \neq \emptyset \\
& \quad \text{do} \ (e_i \rightarrow e_j) \leftarrow \text{DELETE}(Q) \\
& \quad \quad \text{for } e_k \rightarrow e_i \in R \textbf{ such that } (e_k \rightarrow e_j) \notin R \\
& \quad \quad \quad \text{do } \text{INSERT}(e_k \rightarrow e_j, R) \\
& \quad \quad \quad \text{INSERT}(e_k \rightarrow e_j, Q) \\
& \quad \quad \text{for } e_j \rightarrow e_k \in R \textbf{ such that } (e_i \rightarrow e_k) \notin R \\
& \quad \quad \quad \text{do } \text{INSERT}(e_i \rightarrow e_k, R) \\
& \quad \quad \quad \text{INSERT}(e_i \rightarrow e_k, Q) \\
& \quad \text{if } e_j = \lambda x. \ e \textbf{ and } \text{input}(e_i e_w) \\
& \quad \quad \text{then if } (x \rightarrow e_w) \notin R \\
& \quad \quad \quad \text{then } \text{INSERT}(x \rightarrow e_w, R) \\
& \quad \quad \quad \text{INSERT}(x \rightarrow e_w, Q) \\
& \quad \quad \text{if } (e_i e_w \rightarrow e) \notin R \\
& \quad \quad \quad \text{then } \text{INSERT}(e_i e_w \rightarrow e, R) \\
& \quad \quad \quad \text{INSERT}(e_i e_w \rightarrow e, Q) \\
\end{align*}

\text{return } R

Body dominated by for-loops.

Each loop can be a linear scan of a matrix column and row, resp.

\begin{itemize}
\item \text{Hence, } O(n^3).
\end{itemize}
Fast sets
Fast Sets (1/2)

Consider operations over a set \( \{0, \ldots, n - 1\} \)

Assume the RAM has \( \Theta(\log n) \) word size (standard).

Represent the set as a bit vector, cut into words:

\[
\begin{array}{c}
+--------------------------------------------+ \\
| 01\ldots0 | | | ... | | | \\
+--------------------------------------------+
\end{array}
\]

word
Fast Sets (2/2)

- Insert and membership is constant time.
- Improved set difference.

<table>
<thead>
<tr>
<th>operation</th>
<th>type</th>
<th>time complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>∈</td>
<td>( \text{int} \times \text{fset} \rightarrow \text{bool} )</td>
<td>( O(1) )</td>
</tr>
<tr>
<td>INSERT</td>
<td>( \text{int} \times \text{fset} \rightarrow \text{unit} )</td>
<td>( O(1) )</td>
</tr>
<tr>
<td>DIFF</td>
<td>( \text{fset} \times \text{fset} \rightarrow \text{int list} )</td>
<td>( O(n/\log n + v) )</td>
</tr>
</tbody>
</table>
Direct subcubic CFA algorithm
Changes to Cubic algorithm

★ Represent $R$ as a sequence of rows and columns, each of which is a fast set.

★ Assume a numbering of all sub-expressions.

★ Assume the following operations:

\[
\text{COLUMN}(j) = \{k \mid k \to j \in R\}
\]
\[
\text{Row}(j) = \{k \mid j \to k \in R\}
\]

★ Implement $\text{INSERT}(e_i \to e_j, R)$ as:

$\text{INSERT}(i, \text{COLUMN}(j))$ and $\text{INSERT}(j, \text{Row}(i))$.

★ $e_i \to e_j \notin R$ becomes: $i \notin \text{COLUMN}(j)$ and $j \notin \text{Row}(i)$. 
Subcubic algorithm

Subcubic-CFA(e)

\[ Q, R \leftarrow \{ e \rightarrow e \mid \text{input}(e) \} \]

while \( Q \neq \emptyset \)

\[ (e_i \rightarrow e_j) \leftarrow \text{DELETE}(Q) \]

for \( k \in \text{DIFF} (\text{COLUMN}(i), \text{COLUMN}(j)) \)

\[ \text{do\ INSERT}^*(e_k \rightarrow e_j, R) \]

\[ \text{INSERT}(e_k \rightarrow e_j, Q) \]

for \( k \in \text{DIFF} (\text{ROW}(j), \text{ROW}(i)) \)

\[ \text{do\ INSERT}^*(e_i \rightarrow e_k, R) \]

\[ \text{INSERT}(e_i \rightarrow e_k, Q) \]

if \( e_j = \lambda x. e \\text{ and input}(e_i e_w) \)

\[ \text{then if } (x \rightarrow e_w) \notin R \]

\[ \text{then\ INSERT}^*(x \rightarrow e_w, R) \]

\[ \text{INSERT}(x \rightarrow e_w, Q) \]

if \( (e_i e_w \rightarrow e) \notin R \)

\[ \text{then\ INSERT}^*(e_i e_w \rightarrow e, R) \]

\[ \text{INSERT}(e_i e_w \rightarrow e, Q) \]

return \( R \)
Subcubic algorithm

\textsc{Subcubic-CFA}(e)

\begin{align*}
Q, R &\leftarrow \{ e \rightarrow e \mid \text{input}(e) \} \\
\textbf{while } & Q \neq \emptyset \\
\textbf{do } (e_i \rightarrow e_j) &\leftarrow \text{DELETE}(Q) \\
\textbf{for } k &\in \text{DIFF} (\text{COLUMN}(i), \text{COLUMN}(j)) \\
\textbf{do } &\text{INSERT}^*(e_k \rightarrow e_j, R) \\
&\text{INSERT}(e_k \rightarrow e_j, Q) \\
\textbf{for } k &\in \text{DIFF} (\text{ROW}(j), \text{ROW}(i)) \\
\textbf{do } &\text{INSERT}^*(e_i \rightarrow e_k, R) \\
&\text{INSERT}(e_i \rightarrow e_k, Q) \\
\textbf{if } &e_j = \lambda x. e \ \text{and } \text{input}(e_i e_w) \\
\textbf{then if } & (x \rightarrow e_w) \notin R \\
\textbf{then } &\text{INSERT}^*(x \rightarrow e_w, R) \\
&\text{INSERT}(x \rightarrow e_w, Q) \\
\textbf{if } & (e_i e_w \rightarrow e) \notin R \\
\textbf{then } &\text{INSERT}^*(e_i e_w \rightarrow e, R) \\
&\text{INSERT}(e_i e_w \rightarrow e, Q) \\

\text{return } R
\end{align*}

★ Still $O(n^2)$ while-iterations.
★ Body still dominated by for-loops.
★ \text{DIFF} can be done in $O(n/\lg n)$.
★ Each insertion executed once per edge.
★ Hence $O(n^3/\lg n)$.

\textcopyright{} Subcubic Control-Flow Analysis Algorithms, Symposium in Honor of Mitchell Wand, Northeastern University, August, 2009 – p.25/3
Subcubic algorithm

**Subcubic-CFA**

\( Q, R \leftarrow \{ e \rightarrow e \mid \text{input}(e) \} \)

**while** \( Q \neq \emptyset \)

**do** \((e_i \rightarrow e_j) \leftarrow \text{DELETE}(Q)\)

**for** \( k \in \text{DIFF(COLUMN}(i) , \text{COLUMN}(i))\)

**do** \(\text{INSERT}^*(e_k \rightarrow e_j, R)\)

\(\text{INSERT}(e_k \rightarrow e_j, Q)\)

**for** \( k \in \text{DIFF(ROW}(j) , \text{ROW}(i))\)

**do** \(\text{INSERT}^*(e_i \rightarrow e_k, R)\)

\(\text{INSERT}(e_i \rightarrow e_k, Q)\)

**if** \( e_j = \lambda x. e \text{ and } \text{input}(e_i e_w) \)

**then** \( (x \rightarrow e_w) \notin R \)

**then** \(\text{INSERT}^*(x \rightarrow e_w, R)\)

\(\text{INSERT}(x \rightarrow e_w, Q)\)

**if** \( (e_i e_w \rightarrow e) \notin R \)

**then** \(\text{INSERT}^*(e_i e_w \rightarrow e, R)\)

\(\text{INSERT}(e_i e_w \rightarrow e, Q)\)

**return** \( R \)
Further improvements
Consider the program: $\lambda x_0. ((\lambda x. x) v)$

$\star x \rightarrow v$

$\star$ But, what happens when you run it?
Incorporating Reachability

\[
\text{SCOPE}(\lambda x. e) \in \text{Reach} \\
\lambda x. e \rightarrow \lambda x. e \quad \text{(REFL-LAM)}
\]

\[
e_1 \rightarrow \lambda x. e \quad \text{SCOPE}(e_1 e_2) \in \text{Reach} \\
\text{SCOPE}(e) \in \text{Reach}, \; x \rightarrow e_2, \; e_1 e_2 \rightarrow e \quad \text{(APP)}
\]

\[
e_0 \rightarrow e_1 \quad e_1 \rightarrow e_2 \quad e_0 \rightarrow e_2 \quad \text{(TRANS)}
\]

A known improvement to the precision of CFA.
Consider the program: \(((\lambda x. x) \Omega)\)

- $x \rightarrow \Omega$
- But, what happens when you run it?
**Incorporating Valuability**

\[
\text{SCOPE}(\lambda x. e) \in \text{Reach} \\
\lambda x. e \rightarrow \lambda x. e, \lambda x. e \in \text{HasVals} \quad \text{(REFL-LAM)}
\]

\[
e_1 \rightarrow \lambda x. e \quad \text{SCOPE}(e_1 e_2) \in \text{Reach} \quad e_2 \in \text{HasVals} \quad \text{(APP)}
\]

\[
\text{SCOPE}(e) \in \text{Reach}, \ x \rightarrow e_2, \ e_1 e_2 \rightarrow e
\]

\[
e_0 \rightarrow e_1 \\
e_1 \rightarrow e_2 \\
\hline
\text{(TRANS)}
\]

\[
e_0 \rightarrow e_1 \\
e_1 \in \text{HasVals} \quad \text{(HAS-VAL)}
\]

A known improvement to the precision of CFA.

Subcubic Control-Flow Analysis Algorithms, Symposium in Honor of Mitchell Wand, Northeastern University, August, 2009 – p.30/35
What to Take Away

★ The cubic bottleneck can be broken.
★ Simple, direct algorithms can be written using known (textbook) techniques.
★ Known improvements to the precision of CFA can also be done in less than cubic time.
The End

Thank you.
Wand’s Work on CFA

- Steckler and Wand (1994, 1997), POPL, TOPLAS.
- Wand and Siveroni (1999), POPL.
- Wand (2002).
- Wand and Williamson (2002), ESOP.
- Palsberg and Wand (2003), JFP.
- Meunier et al. (2005), HOSC.
Wand’s Influence on CFA

- Henglein (1991), FPCA
- Palsberg (1995), TOPLAS.
- Steensgaard (1996), POPL.
- Damian and Danvy (2003), JFP.
- Danvy and Nielsen (2003), TCS.
- Meunier et al. (2006), POPL.

...just to name a few.
Wand’s Influence on Me
Wand’s Influence on Me
Wand’s Influence on Me

“Thank you for making this day necessary.”

— Yogi Berra
References


