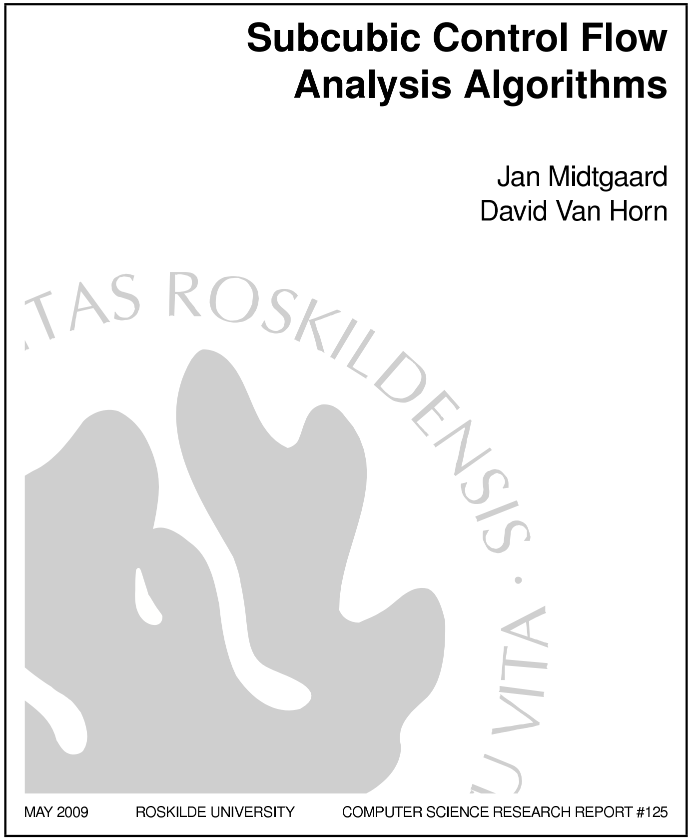
Subcubic Control-Flow Analysis Algorithms

Jan Midtgaard and David Van Horn

SOLIDUM PETIT IN PROFUNDIS · UNIVERSITAS ARHUSIENSIS ·

NORTHEASTERN UNIVERSITY
LVX VERITAS VIRTVS
FOUNDED AD 1898
BOSTON, MASSACHUSETTS

Technical Report



Subcubic Control Flow Analysis Algorithms

Jan Midtgaard
David Van Horn

MAY 2009 ROSKILDE UNIVERSITY COMPUTER SCIENCE RESEARCH REPORT #125

http://www.ruc.dk/dat_en/research/reports/125/

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

- $\star$ $f \longrightarrow \lambda x.\, x$
- $\star$ $(\lambda f.\, f\, f\, v)\, (\lambda x.\, x) \longrightarrow f\, f\, v$

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

- $\star$ $f \to \lambda x.\, x$
- $\star$ $(\lambda f.\, f\, f\, v)\, (\lambda x.\, x) \to f\, f\, v$
- $\star$ $x \to \lambda x.\, x$
- $\star$ $f f \to x$

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

- $\star$ $f \longrightarrow \lambda x.\, x$
- $\star$ $(\lambda f.\, f f\, v)\,(\lambda x.\, x) \longrightarrow f f\, v$
- $\star$ $x \longrightarrow \lambda x.\, x$
- $\star$ $f f \longrightarrow x$
- $\star$ $x \longrightarrow v$
- $\star$ $f f\, v \longrightarrow x$

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

- $\star$ $f \rightarrow \lambda x.\, x$
- $\star$ $(\lambda f.\, f\,f\,v)\,(\lambda x.\, x) \rightarrow f\,f\,v$
- $\star$ $x \rightarrow \lambda x.\, x$
- $\star$ $f\,f \rightarrow x$
- $\star$ $x \rightarrow v$
- $\star$ $f\,f\,v \rightarrow x$

CFA

Language:

$$e ::= x \mid \lambda x.\, e \mid e\; e \qquad \text{(expressions)}$$

Analysis:

$$\frac{input(e)}{e \rightarrow e}\ \text{(REFL)} \qquad \frac{input(e_1\; e_2) \qquad e_1 \rightarrow \lambda x.\, e}{x \rightarrow e_2,\ e_1\; e_2 \rightarrow e}\ \text{(APP)}$$

$$\frac{e_0 \rightarrow e_1 \qquad e_1 \rightarrow e_2}{e_0 \rightarrow e_2}\ \text{(TRANS)}$$

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.

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.

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.

Chaudhuri (2008)

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 ← {e → e | input(e)}
  while Q ≠ ∅
      do (e_i → e_j) ← DELETE(Q)
         for e_k → e_i ∈ R such that (e_k → e_j) ∉ R
             do INSERT(e_k → e_j, R)
                INSERT(e_k → e_j, Q)
         for e_j → e_k ∈ R such that (e_i → e_k) ∉ R
             do INSERT(e_i → e_k, R)
                INSERT(e_i → e_k, Q)
         if e_j = λx. e and input(e_i e_w)
           then if (x → e_w) ∉ R
                  then INSERT(x → e_w, R)
                       INSERT(x → e_w, Q)
                if (e_i e_w → e) ∉ R
                  then INSERT(e_i e_w → e, R)
                       INSERT(e_i e_w → e, Q)
  return R
```

Cubic algorithm

CUBIC-CFA(e)

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

- ⋆ Initialization is $O(n)$.
- ⋆ Edges inserted *simultaneously* into Q, R.
- ⋆ Once in R, never inserted in Q again.
- ⋆ So edges inserted into Q at most once.
- ⋆ At most $O(n^2)$ while-loop iterations.

Cubic algorithm

CUBIC-CFA(e)

$Q, R \leftarrow \{e \to e \mid input(e)\}$
while $Q \neq \emptyset$
  do $(e_i \to e_j) \leftarrow$ DELETE(Q)
    for $e_k \to e_i \in R$ **such that** $(e_k \to e_j) \notin R$
      do INSERT$(e_k \to e_j, R)$
        INSERT$(e_k \to e_j, Q)$
    for $e_j \to e_k \in R$ **such that** $(e_i \to e_k) \notin R$
      do INSERT$(e_i \to e_k, R)$
        INSERT$(e_i \to e_k, Q)$
    if $e_j = \lambda x.\, e$ **and** $input(e_i\ e_w)$
      then if $(x \to e_w) \notin R$
        then INSERT$(x \to e_w, R)$
          INSERT$(x \to e_w, Q)$
      if $(e_i\ e_w \to e) \notin R$
        then INSERT$(e_i\ e_w \to e, R)$
          INSERT$(e_i\ e_w \to e, Q)$
return R

- ⋆ Body dominated by for-loops.
- ⋆ Each loop can be a linear scan of a matrix column and row, resp.
- ⋆ Hence, $O(n^3)$.

Fast sets

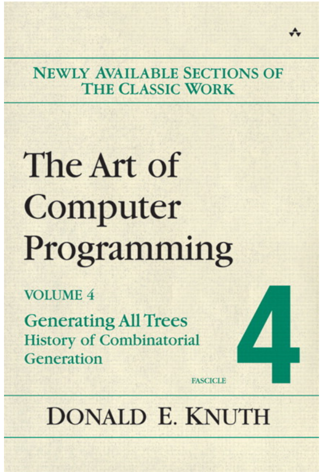
Fast Sets (1/2)

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:

```
+----------------------------------------+
|01...0|     |     |  ...   |     |     |
+----------------------------------------+
 \    /
  word
```



Fast Sets (2/2)

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

operation	type	time complexity
$\in$	$int \times fset \to bool$	$O(1)$
INSERT	$int \times fset \to unit$	$O(1)$
DIFF	$fset \times fset \to int\ list$	$O(n/\log n + v)$

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:

$$\begin{aligned} \textsc{Column}(j) &= \{k \mid k \rightarrow j \in R\} \\ \textsc{Row}(j) &= \{k \mid j \rightarrow k \in R\} \end{aligned}$$

- Implement $\textsc{Insert}(e_i \rightarrow e_j, R)$ as: $\textsc{Insert}(i, \textsc{Column}(j))$ and $\textsc{Insert}(j, \textsc{Row}(i))$.
- $e_i \rightarrow e_j \notin R$ becomes: $i \notin \textsc{Column}(j)$ **and** $j \notin \textsc{Row}(i)$.

Subcubic algorithm

```
SUBCUBIC-CFA(e)
    Q, R ← {e → e | input(e)}
    while Q ≠ ∅
        do (e_i → e_j) ← DELETE(Q)
            for k ∈ DIFF(COLUMN(i), COLUMN(j))
                do INSERT*(e_k → e_j, R)
                   INSERT(e_k → e_j, Q)
            for k ∈ DIFF(ROW(j), ROW(i))
                do INSERT*(e_i → e_k, R)
                   INSERT(e_i → e_k, Q)
            if e_j = λx. e and input(e_i e_w)
              then if (x → e_w) ∉ R
                     then INSERT*(x → e_w, R)
                          INSERT(x → e_w, Q)
                   if (e_i e_w → e) ∉ R
                     then INSERT*(e_i e_w → e, R)
                          INSERT(e_i e_w → e, Q)
    return R
```

Subcubic algorithm

SUBCUBIC-CFA(e)

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

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

Subcubic algorithm

SUBCUBIC-CFA(e)

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

Further improvements

Recall CFA

$$\frac{input(e)}{e \longrightarrow e} \text{ (REFL)} \qquad \frac{input(e_1\ e_2) \qquad e_1 \longrightarrow \lambda x.\, e}{x \longrightarrow e_2,\ e_1\ e_2 \longrightarrow e} \text{ (APP)}$$

$$\frac{e_0 \longrightarrow e_1 \qquad e_1 \longrightarrow e_2}{e_0 \longrightarrow e_2} \text{ (TRANS)}$$

Consider the program: $\lambda x_0.\, ((\lambda x.\, x)\ v)$

- ⋆ $x \longrightarrow v$
- ⋆ But, what happens when you run it?

Incorporating Reachability

$$\frac{\textsc{scope}(\lambda x.\, e) \in \text{Reach}}{\lambda x.\, e \rightarrow \lambda x.\, e} \ (\text{REFL-LAM})$$

$$\frac{e_1 \rightarrow \lambda x.\, e \qquad \textsc{scope}(e_1\ e_2) \in \text{Reach}}{\textsc{scope}(e) \in \text{Reach},\ x \rightarrow e_2,\ e_1\ e_2 \rightarrow e} \ (\text{APP})$$

$$\frac{e_0 \rightarrow e_1 \qquad e_1 \rightarrow e_2}{e_0 \rightarrow e_2} \ (\text{TRANS})$$

A known improvement to the precision of CFA.

Recall Recalling CFA

$$\frac{input(e)}{e \longrightarrow e} \text{ (REFL)} \qquad \frac{input(e_1\ e_2) \qquad e_1 \longrightarrow \lambda x.\, e}{x \longrightarrow e_2,\ e_1\ e_2 \longrightarrow e} \text{ (APP)}$$

$$\frac{e_0 \longrightarrow e_1 \qquad e_1 \longrightarrow e_2}{e_0 \longrightarrow e_2} \text{ (TRANS)}$$

Consider the program: $((\lambda x.\, x)\, \Omega)$

- $\star$ $x \longrightarrow \Omega$
- $\star$ But, what happens when you run it?

Incorporating Valuability

$$\frac{\textsc{scope}(\lambda x.\ e) \in \mathsf{Reach}}{\lambda x.\ e \rightarrow \lambda x.\ e,\ \lambda x.\ e \in \mathsf{HasVals}}\ (\textsc{Refl-Lam})$$

$$\frac{e_1 \rightarrow \lambda x.\ e \qquad \textsc{scope}(e_1\ e_2) \in \mathsf{Reach} \qquad e_2 \in \mathsf{HasVals}}{\textsc{scope}(e) \in \mathsf{Reach},\ x \rightarrow e_2,\ e_1\ e_2 \rightarrow e}\ (\textsc{App})$$

$$\frac{e_0 \rightarrow e_1 \qquad e_1 \rightarrow e_2}{e_0 \rightarrow e_2}\ (\textsc{Trans})$$

$$\frac{e_0 \rightarrow e_1 \qquad e_1 \in \mathsf{HasVals}}{e_0 \in \mathsf{HasVals}}\ (\textsc{Has-Val})$$

A known improvement to the precision of CFA.

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

- ⋆ Montenyohl and Wand (1991), Sci. Comp. Program.
- ⋆ 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.

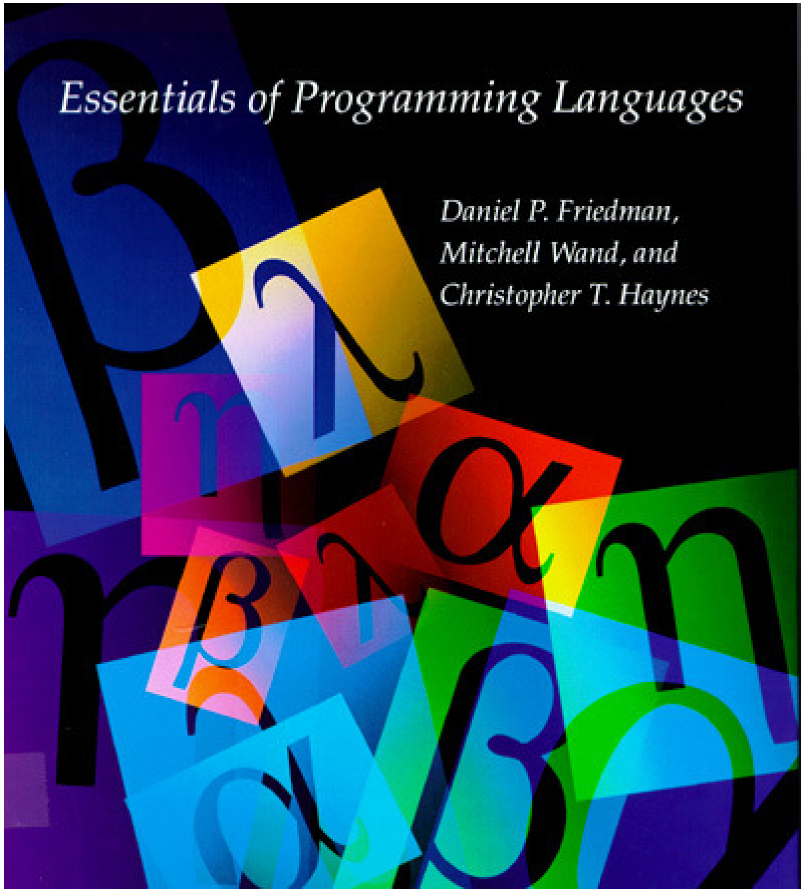
Wand’s Influence on CFA

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

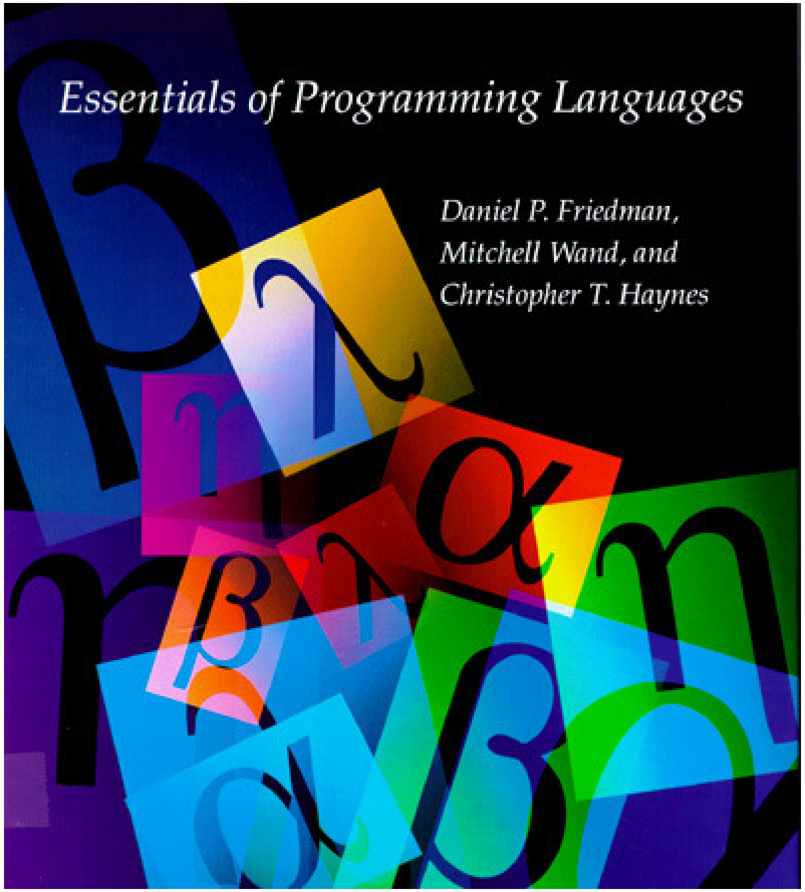
. . . just to name a few.

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

Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. Time and tape complexity of pushdown automaton languages. *Information and Control*, 13(3):186–206, 1968.

A. E. Ayers. Efficient closure analysis with reachability. In *WSA*, pages 126–134, 1992.

Swarat Chaudhuri. Subcubic algorithms for recursive state machines. In *POPL '08: Proceedings of the 35th annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 159–169, New York, New York, USA, 2008. ACM. ISBN 978-1-59593-689-9. doi: http://doi.acm.org/10.1145/1328438.1328460.

Daniel Damian and Olivier Danvy. CPS transformation of flow information, part ii: administrative reductions. *J. Funct. Program.*, 13(5):925–933, 2003. ISSN 0956-7968. doi: http://dx.doi.org/10.1017/S0956796803004702.

Olivier Danvy and Lasse R. Nielsen. A first-order one-pass CPS transformation. *Theor. Comput. Sci.*, 308(1-3):239–257, 2003. ISSN 0304-3975. doi: http://dx.doi.org/10.1016/S0304-3975(02)00733-8.

Nevin Heintze and David McAllester. On the cubic bottleneck in subtyping and flow analysis. In *LICS '97: Proceedings of* *the 12th Annual IEEE Symposium on Logic in Computer Science*, page 342, Washington, DC, USA, 1997a. IEEE Computer Society. ISBN 0-8186-7925-5.

Nevin Heintze and David McAllester. Linear-time subtransitive control flow analysis. In *PLDI '97: Proceedings of the ACM SIGPLAN 1997 Conference on Programming Language Design and Implementation*, pages 261–272, New York, New York, USA, 1997b. ACM Press. ISBN 0-89791-907-6. doi: http://doi.acm.org/10.1145/258915.258939.

Fritz Henglein. Efficient type inference for higher-order binding-time analysis. In *Proceedings of the 5th ACM Conference on Functional Programming Languages and Computer Architecture*, pages 448–472, London, UK, 1991. Springer-Verlag. ISBN 3-540-54396-1.

Neil D. Jones. Flow analysis of lambda expressions (preliminary version). In *Proceedings of the 8th Colloquium on Automata, Languages, and Programming*, pages 114–128. Springer-Verlag, 1981a.

Neil D. Jones. Flow analysis of lambda expressions. Technical Report PB-128, Aarhus University, Aarhus, Denmark, 1981b.

David McAllester. On the complexity analysis of static analyses. *J. ACM*, 49(4):512–537, 2002. ISSN 0004-5411. doi: http://doi.acm.org/10.1145/581771.581774.

David Melski and Thomas Reps. Interconvertibility of a class of set constraints and context-free-language reachability. *Theor. Comput. Sci.*, 248(1-2):29–98, 2000. ISSN 0304-3975. doi: http://dx.doi.org/10.1016/S0304-3975(00)00049-9.

Philippe Meunier, Robert Bruce Findler, Paul Steckler, and Mitchell Wand. Selectors make set-based analysis too hard. *Higher Order Symbol. Comput.*, 18(3-4):245–269, 2005. ISSN 1388-3690. doi: http://dx.doi.org/10.1007/s10990-005-4876-5.

Philippe Meunier, Robert Bruce Findler, and Matthias Felleisen. Modular set-based analysis from contracts. In *POPL '06: Conference record of the 33rd ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 218–231, New York, NY, USA, 2006. ACM. ISBN 1-59593-027-2. doi: http://doi.acm.org/10.1145/1111037.1111057.

Margaret Montenyohl and Mitchell Wand. Correctness of static flow analysis in continuation semantics. *Sci. Comput. Program.*, 16(1):1–18, 1991. ISSN 0167-6423. doi: http://dx.doi.org/10.1016/0167-6423(91)90021-O.

Radford Neal. The computational complexity of taxonomic inference, December 1989. Unpublished manuscript. `ftp://ftp.cs.utoronto.ca/pub/radford/taxc`

Jens Palsberg. Closure analysis in constraint form. *ACM Trans. Program. Lang. Syst.*, 17(1):47–62, 1995. ISSN 0164-0925. doi: http://doi.acm.org/10.1145/200994.201001.

Jens Palsberg and Mitchell Wand. CPS transformation of flow information. *J. Funct. Program.*, 13(5):905–923, 2003. ISSN 0956-7968. doi: http://dx.doi.org/10.1017/S0956796802004513.

Wojciech Rytter. Fast recognition of pushdown automaton and context-free languages. *Inf. Control*, 67(1-3):12–22, 1985. ISSN 0019-9958. doi: http://dx.doi.org/10.1016/S0019-9958(85)80024-3.

Paul A. Steckler and Mitchell Wand. Lightweight closure conversion. *ACM Trans. Program. Lang. Syst.*, 19(1):48–86, 1997. ISSN 0164-0925. doi: http://doi.acm.org/10.1145/239912.239915.

Paul A. Steckler and Mitchell Wand. Selective and lightweight closure conversion. In *POPL '94: Proceedings of the 21st ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 435–445, New York, NY, USA, 1994. ACM. ISBN 0-89791-636-0. doi: http://doi.acm.org/10.1145/174675.178044.

Bjarne Steensgaard. Points-to analysis in almost linear time. In *POPL '96: Proceedings of the 23rd ACM*

SIGPLAN-SIGACT Symposium on Principles of Programming Languages, pages 32–41, New York, NY, USA, 1996. ACM. ISBN 0-89791-769-3. doi: http://doi.acm.org/10.1145/237721.237727.

David Van Horn and Harry G. Mairson. Flow analysis, linearity, and PTIME. In María Alpuente and Germán Vidal, editors, *SAS*, volume 5079 of *Lecture Notes in Computer Science*, pages 255–269. Springer, 2008. ISBN 978-3-540-69163-1.

Mitchell Wand. Analyses that distinguish different evaluation orders, or, unsoundness results in control-flow analysis, July 2002. Unpublished manuscript.

Mitchell Wand and Igor Siveroni. Constraint systems for useless variable elimination. In *POPL '99: Proceedings of the 26th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 291–302, New York, NY, USA, 1999. ACM. ISBN 1-58113-095-3. doi: http://doi.acm.org/10.1145/292540.292567.

Mitchell Wand and Galen B. Williamson. A modular, extensible proof method for small-step flow analyses. In *ESOP '02: Proceedings of the 11th European Symposium on Programming Languages and Systems*, pages 213–227, London, UK, 2002. Springer-Verlag. ISBN 3-540-43363-5.