The Fine-Grained And Parallel Complexity of Andersen's Pointer Analysis

Anders Alnor Mathiasen and **Andreas Pavlogiannis**

AARHUS UNIVERSITY

Static Pointer Analysis

Pointers make static analysis hard

```
...
*a=42;
*b=84;
c = *a;
//is c 42 or 84?
```

Static Pointer Analysis

Pointers make static analysis hard

```
...
*a=42;
*b=84;
c = *a;
//is c 42 or 84?
```

Pointer analysis is typically a prerequisite to other static analyses

- Must be fast
- Determines the quality of the static analysis

Andersen's Pointer Analysis (APA)

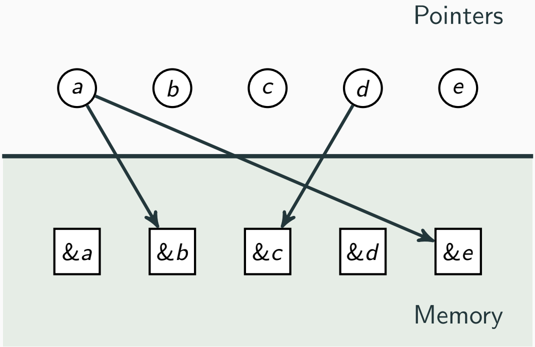
[PDF] Program analysis and specialization for the C programming language

LO Andersen - 1994 - pdfs.semanticscholar.org

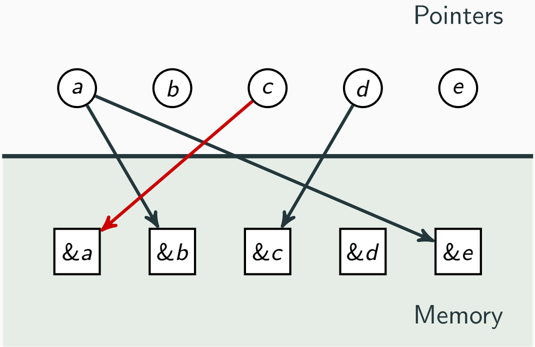
Software engineers are faced with a dilemma. They want to write general and wellstructured programs that are flexible and easy to maintain. On the other hand, generality has a price: efficiency. A specialized program solving a particular problem is often significantly faster than a general program. However, the development of specialized software is time-consuming, and is likely to exceed the production of today's programmers. New techniques are required to solve this so-called software crisis. Partial evaluation is a program …

- Flow insensitive
- No nested dereferences
- Inclusion based

Andersen's Pointer Analysis (APA)

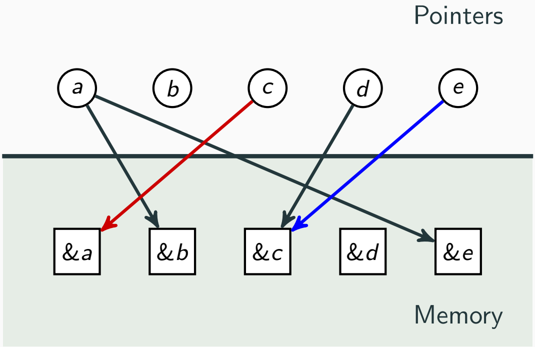


Andersen's Pointer Analysis (APA)



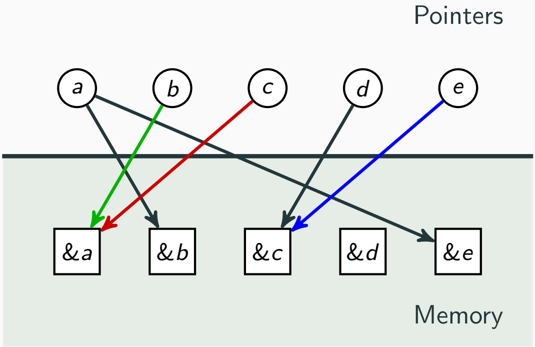
$c = \&a$

Andersen's Pointer Analysis (APA)



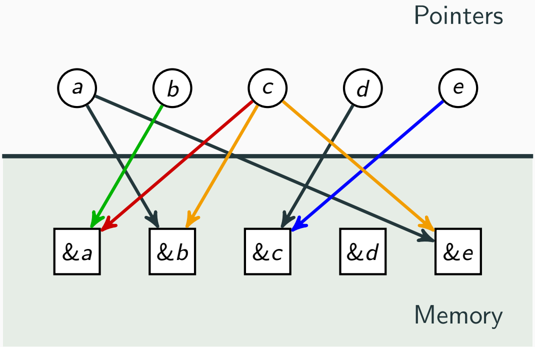
$c = \&a \qquad e = d$

Andersen's Pointer Analysis (APA)



$c = \&a$ $\quad e = d$ $\quad b = *e$

Andersen's Pointer Analysis (APA)



$c = \&a$ $\quad$ $e = d$ $\quad$ $b = *e$ $\quad$ $*d = a$

Andersen's Pointer Analysis (APA)

Solve for inclusion constraints

Type	Statement	Inclusion Constraint
1	$a = b$	$PointsToSet(b) \subseteq PointsToSet(a)$
2	$a = \&b$	$b \in PointsToSet(a)$
3	$a = *b$	$\forall c \in PointsToSet(b)\colon PointsToSet(c) \subseteq PointsToSet(a)$
4	$*a = b$	$\forall c \in PointsToSet(a)\colon PointsToSet(b) \subseteq PointsToSet(c)$

Andersen's Pointer Analysis (APA)

Solve for inclusion constraints

Type	Statement	Inclusion Constraint
1	$a = b$	$\mathit{PointsToSet}(b) \subseteq \mathit{PointsToSet}(a)$
2	$a = \&b$	$b \in \mathit{PointsToSet}(a)$
3	$a = *b$	$\forall c \in \mathit{PointsToSet}(b) \colon \mathit{PointsToSet}(c) \subseteq \mathit{PointsToSet}(a)$
4	$*a = b$	$\forall c \in \mathit{PointsToSet}(a) \colon \mathit{PointsToSet}(b) \subseteq \mathit{PointsToSet}(c)$

Input:(A, S)

- A is a set of n pointers
- S is a set of m statements
    - $m \leq 4 \cdot n^2$ unique statements

Andersen's Pointer Analysis (APA)

Solve for inclusion constraints

Type	Statement	Inclusion Constraint
1	$a = b$	$\mathit{PointsToSet}(b) \subseteq \mathit{PointsToSet}(a)$
2	$a = \&b$	$b \in \mathit{PointsToSet}(a)$
3	$a = *b$	$\forall c \in \mathit{PointsToSet}(b)\colon \mathit{PointsToSet}(c) \subseteq \mathit{PointsToSet}(a)$
4	$*a = b$	$\forall c \in \mathit{PointsToSet}(a)\colon \mathit{PointsToSet}(b) \subseteq \mathit{PointsToSet}(c)$

Input: (A, S)

- A is a set of n pointers
- S is a set of m statements
    - $m \leq 4 \cdot n^2$ unique statements

Output:

- *Exhaustive:* Return $\mathit{PointsToSet}(a)$ for all pointers $a \in A$
- *On-demand:* Return if $b \in \mathit{PointsToSet}(a)$ for a specific pair $a, b \in A$

This Paper: How fast can we perform Andersen's Pointer Analysis?

- Part A: Overview of results
- Part B: D_1-Reachability in $O(n^{\omega} \cdot \log^2 n)$ time

Part A: Overview of results

Cubic complexity of APA

Cubic Complexity of APA

- (A, S), $|A| = n$, $|S| = m$
- m can be as large as $4 \cdot n^2$

APA is solvable in cubic time

Cubic means:

$O(m^3)$? $O(n^2 \cdot m)$? $O(n^4)$??

Cubic Complexity of APA

- (A, S), $|A| = n$, $|S| = m$
- m can be as large as $4 \cdot n^2$

APA is solvable in cubic time

Cubic means:

$O(m^3)$? $O(n^2 \cdot m)$? $O(n^4)$??

Theorem

APA *solvable in* $O(n^3)$ *time, regardless of* m.

Can we do it faster? Is n^3 tight?

Exhaustive vs On-demand

Exhaustive

- For all a, output $\mathit{PointsToSet}(a)$
- Output size $\Theta(n^2)$

On-demand

- Given a, b, is it that $b \in \mathit{PointsToSet}(a)$?
- Output size $O(1)$

Exhaustive vs On-demand

Exhaustive

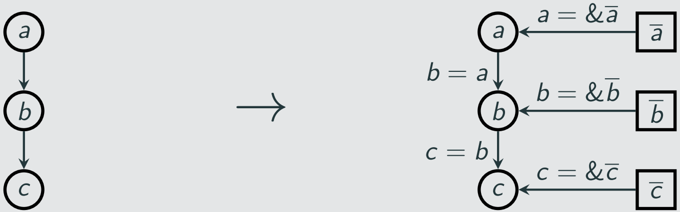
- For all a, output $PointsToSet(a)$
- Output size $\Theta(n^2)$

On-demand

- Given a, b, is it that $b \in PointsToSet(a)$?
- Output size $O(1)$

Cubic Hardness for Exhaustive APA

- Exhaustive APA can encode transitive closure of graph $G = (V, E)$



- $\implies$ Combinatorial cubic hardness for exhaustive APA

Cubic Bottleneck for APA

Theorem

On-demand APA *has no* $(n^{3-\epsilon})$ *time algorithm, for any* $\epsilon > 0$, *under the combinatorial MM hypothesis.*

- *On-demand not easier than exhaustive*
- *Hardness does not come from large outputs*

Cubic Bottleneck for APA

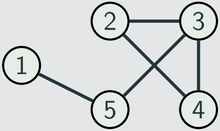
Theorem

On-demand APA *has no* $(n^{3-\epsilon})$ *time algorithm, for any* $\epsilon > 0$, *under the combinatorial MM hypothesis.*

- *On-demand not easier than exhaustive*
- *Hardness does not come from large outputs*

Proof.

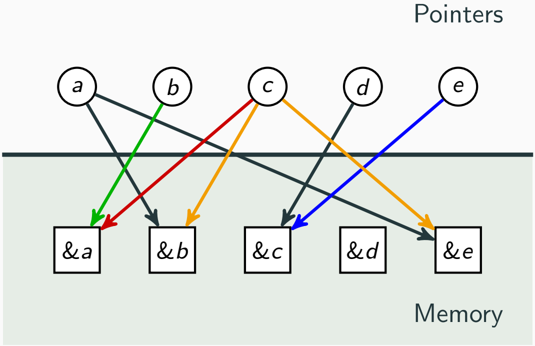
Fine-grained reduction from finding a triangle in an undirected graph.



□

Witness Bounding

Witness Bounding



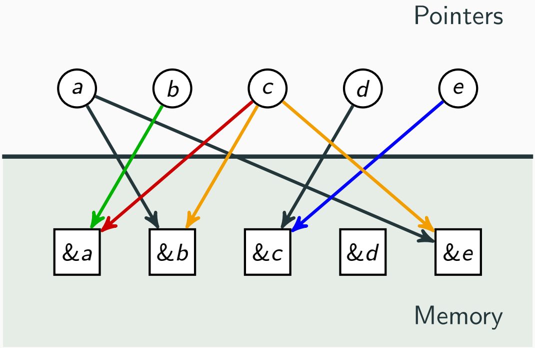
$c = \&a$ → n^2

$e = d$ → n^2

$b = *e$ → n^2

$*d = a$ → n^2

Witness Bounding



$c = \&a \;\downarrow\; n^2$

$e = d \;\downarrow\; n^2$

$b = *e \;\downarrow\; n^2$

$*d = a \;\downarrow\; \boxed{\log n}$

Bounded APA

(i,j)-**bounded** APA

At most j applications of rule i.

1. $a = b$
2. $a = \&b$
3. $a = *b$
4. $*a = b$

Bounded APA

(i, j)-**bounded** APA

At most j applications of rule i.

1. $a = b$
2. $a = \&b$
3. $a = *b$
4. $*a = b$

Theorem

$(4, \tilde{O}(1))$-bounded APA *solvable in $\tilde{O}(n^{\omega})$ time ($\omega < 2.372\ldots$).*

Only polylog n many applications of statements of the form $*a = b$

Strategy

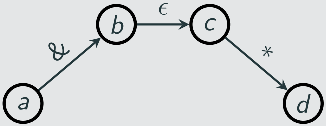
D$_1$-Reachability **captures** $(A, S \setminus S_4)$

$$b = \&a$$

$$c = b$$

$$d = *c$$

$$\longrightarrow$$



Strategy

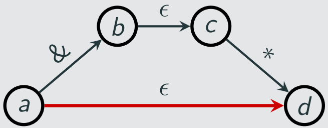
D$_1$-Reachability **captures** $(A, S \setminus S_4)$

$b = \&a$

$c = b$

$d = *c$

$\longrightarrow$



Algorithm for $(4, j)$-bounded APA

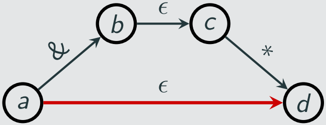
1. Take the Dyck graph $G = (A, E)$ representing $(A, S \setminus S_4)$
2. For $i \in \{1, \ldots, j\}$
   2.1 Solve D$_1$-Reachability in G
   2.2 Find all $x \xrightarrow{\&} a \overset{\epsilon}{\rightsquigarrow} d$
   2.3 For every statement $*d = y$, insert $y \xrightarrow{\epsilon} x$ in G

Strategy

D_1-Reachability captures $(A, S \setminus S_4)$

$$b = \&a$$
$$c = b$$
$$d = *c$$

$\longrightarrow$



Algorithm for $(4, j)$-bounded APA

1. Take the Dyck graph $G = (A, E)$ representing $(A, S \setminus S_4)$
2. For $i \in \{1, \ldots, j\}$
   2.1 Solve D_1-Reachability in G
   2.2 Find all $x \xrightarrow{\&} a \overset{\epsilon}{\rightsquigarrow} d$
   2.3 For every statement $*d = y$, insert $y \xrightarrow{\epsilon} x$ in G

Theorem

D_1-Reachability *in* $O(n^\omega \cdot \log^2 n)$ *time.*

Quadratic Lower Bound

Theorem

$(4, \tilde{O}(1))$*-bounded* APA *solvable in* $\tilde{O}(n^{\omega})$ *time.*

- $\omega < 2.372\ldots$
- Believed that $\omega = 2 + o(1)$

Can we do $O(n^{2-\epsilon})$ if we only look for logarithmic-length witnesses?

Quadratic Lower Bound

Theorem

$(4, \tilde{O}(1))$*-bounded* APA *solvable in* $\tilde{O}(n^{\omega})$ *time.*

- $\omega < 2.372\ldots$
- Believed that $\omega = 2 + o(1)$

Can we do $O(n^{2-\epsilon})$ if we only look for logarithmic-length witnesses?

Theorem

$(\mathrm{All}, \tilde{O}(1))$*-bounded on-demand* APA *has no* $O(n^{2-\epsilon})$ *time algorithm, for any* $\epsilon > 0$*, under the Orthogonal Vectors Hypothesis.*

Orthogonal Vectors

Input: Sets $A, B \subseteq \{0,1\}^d$ with $|A| = |B| = n$ and $d = \Omega(\log n)$

Output: $(a, b) \in A \times B$ such that $a \perp b$

Hypothesis: No $O(n^{2-\epsilon})$-time algorithm

Is APA parallelizable?

Can we solve it really fast with polynomial resources?

APA **is not Parallelizable**

Theorem

APA *is P-complete.*

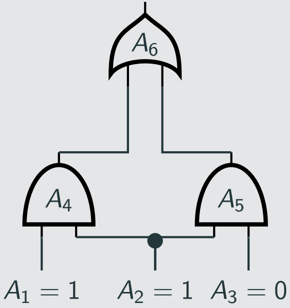
APA **is not Parallelizable**

Theorem

APA *is P-complete.*

Proof.

Log-space reduction from Monotone Circuit Value Problem (CVP).



□

Bounded APA **is Parallelizable**

The class NC

Recall: given $i \in \mathbb{N}^+$, NC^i is the class of problems solvable
(a) in parallel time $O(\log^i n)$ and (b) with polynomially many of processors

$$\mathsf{NC} = \bigcup_i \mathsf{NC}^i$$

E.g., matrix multiplication $\in \mathsf{NC}^1$, graph reachability $\in \mathsf{NC}^2$

Bounded APA is Parallelizable

The class NC

Recall: given $i \in \mathbb{N}^+$, NC^i is the class of problems solvable
(a) in parallel time $O(\log^i n)$ and (b) with polynomially many of processors

$$\mathsf{NC} = \bigcup_i \mathsf{NC}^i$$

E.g., matrix multiplication $\in \mathsf{NC}^1$, graph reachability $\in \mathsf{NC}^2$

Theorem

$(4, \log^i n)$-*bounded* APA *is in* NC^{i+2}.

Witnesses with $\leq \log^i n$ applications of $*a = b$ helps

Conclusion

- Andersen's Pointer Analysis is one of the most popular static pointer analyses
- Its running time is very important, yet its complexity, so far, poorly understood

Conclusion

- Andersen's Pointer Analysis is one of the most popular static pointer analyses
- Its running time is very important, yet its complexity, so far, poorly understood

The complexity of Andersen's Pointer Analysis

- Cubic upper and lower bounds
- Bounding number of $*a = b$ statements helps
  - $\tilde{O}(n^{\omega})$ upper bound (D_1-Reachability solvable in this bound)
  - $\Omega(n^2)$ lower bound
- P-complete
- Bounding number of $*a = b$ statements brings it in NC

Conclusion

- Andersen's Pointer Analysis is one of the most popular static pointer analyses
- Its running time is very important, yet its complexity, so far, poorly understood

The complexity of Andersen's Pointer Analysis

- Cubic upper and lower bounds
- Bounding number of $*a = b$ statements helps
  - $\tilde{O}(n^{\omega})$ upper bound (D_1-Reachability solvable in this bound)
  - $\Omega(n^2)$ lower bound
- P-complete
- Bounding number of $*a = b$ statements brings it in NC

Thank you!

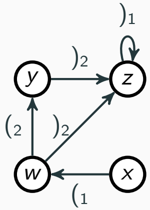
Part B: D_1-Reachability in $O\left(n^{\omega} \cdot \log^2 n\right)$ time

Dyck Graphs

$$\Sigma = \{(_1,)_1, \ldots (_k,)_k\} \cup \{\epsilon\}$$

$$\mathcal{S} \to \mathcal{S}\ \mathcal{S} \mid (_1\ \mathcal{S}\ )_1 \mid \ldots \mid (_k\ \mathcal{S}\ )_k \mid \epsilon$$

$$G = (V, E, \lambda : E \to \Sigma)$$

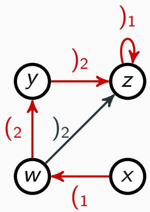


Dyck Graphs

$\Sigma = \{(_1,)_1, \dots (_k,)_k\} \cup \{\epsilon\}$

$\mathcal{S} \rightarrow \mathcal{S}\ \mathcal{S} \mid (_1\ \mathcal{S}\ )_1 \mid \dots \mid (_k\ \mathcal{S}\ )_k \mid \epsilon$

$G = (V, E, \lambda : E \rightarrow \Sigma)$



D_1-Reachability

D_k-Reachability: use k parenthesis types $(_i,)_i$

Computational problem

Given a Dyck graph G, compute all-pairs D_k-Reachability.

D_1-Reachability

D_k-Reachability: use k parenthesis types $(_i,)_i$

Computational problem

Given a Dyck graph G, compute all-pairs D_k-Reachability.

How fast can we solve it?

- $k = 0$: standard graph reachability, $O(n^\omega)$
    - Standard transitive closure $O(n^3)$
    - $\omega \simeq 2.37$
- $k \geq 2$: $O(n^3)$, believed to be tight

D_1-Reachability

D_k-Reachability: use k parenthesis types $(_i,)_i$

Computational problem

Given a Dyck graph G, compute all-pairs D_k-Reachability.

How fast can we solve it?

- $k = 0$: standard graph reachability, $O(n^\omega)$
  - Standard transitive closure $O(n^3)$
  - $\omega \simeq 2.37$
- $k \geq 2$: $O(n^3)$, believed to be tight

Theorem

All-pairs D_1-Reachability *can be solved in* $O(n^\omega \cdot \log^2 n)$ *time.*

Length of Witness Paths

- For $k = 1$, the stack alphabet is unary
- Just a counter

Length of Witness Paths

- For $k = 1$, the stack alphabet is unary
- Just a counter

$\underline{k = 2}$

Lemma

Distances can be exponential

- *at least* $\Omega(2^n)$

Length of Witness Paths

- For $k = 1$, the stack alphabet is unary
- Just a counter

$\underline{k = 2}$

Lemma

Distances can be exponential

- *at least* $\Omega(2^n)$

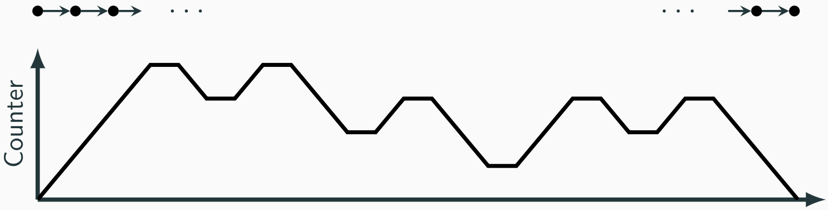
$\underline{k = 1}$

Lemma

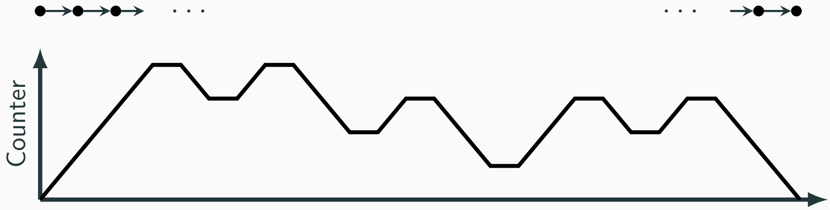
Distances are polynomial

- *at most* $O(n^2)$

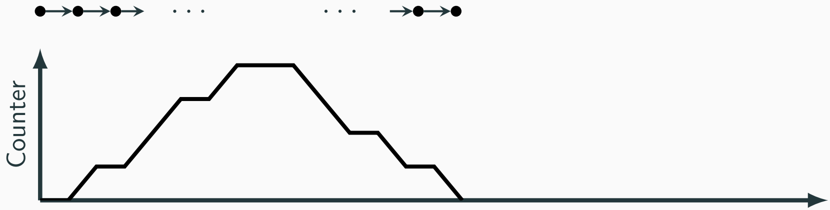
Bell-Shape Paths



Bell-Shape Paths



Bell-shape

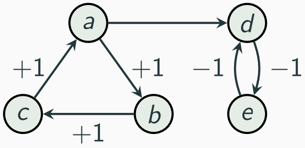


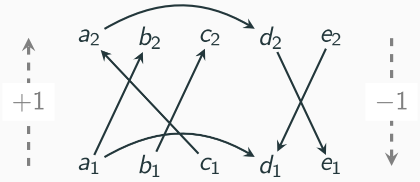
Bell-Shape Reachability

- What if we focus on paths with counter ≤ 1?
- At most one $+1$

Bell-Shape Reachability

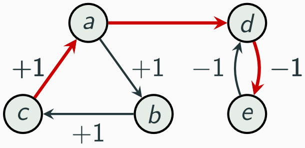
- What if we focus on paths with counter ≤ 1?
- At most one $+1$

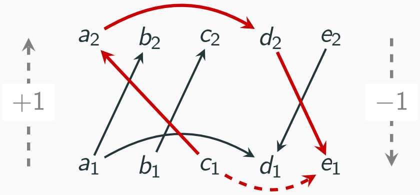




Bell-Shape Reachability

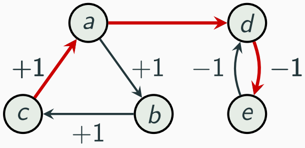
- What if we focus on paths with counter ≤ 1?
- At most one $+1$

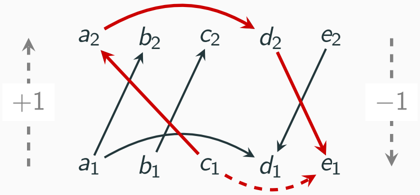




Bell-Shape Reachability

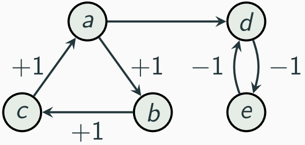
- What if we focus on paths with counter ≤ 1?
- At most one $+1$

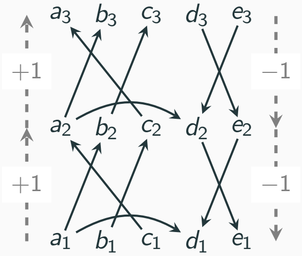




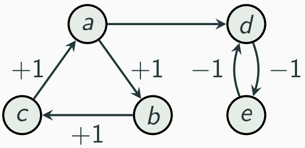
Larger counter values? Iterative doubling

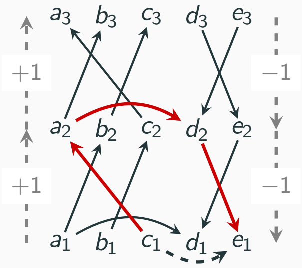
Bell-Shape Reachability



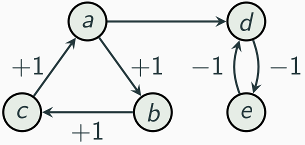


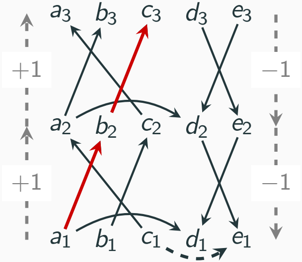
Bell-Shape Reachability



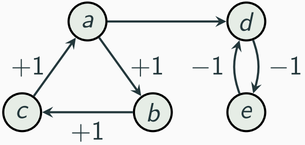


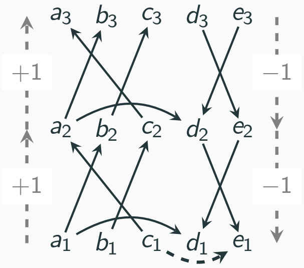
Bell-Shape Reachability

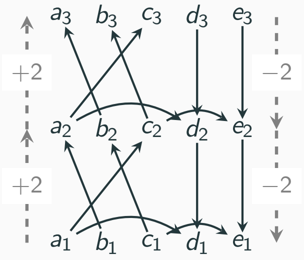




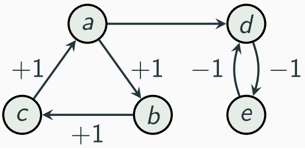
Bell-Shape Reachability

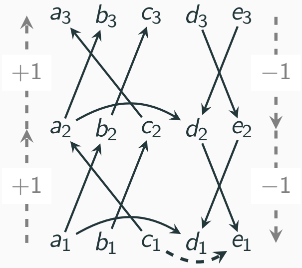


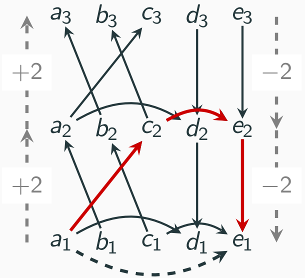




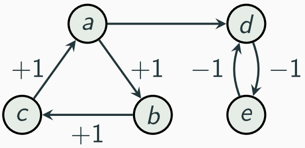
Bell-Shape Reachability

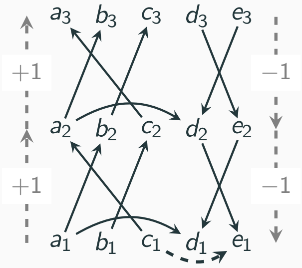


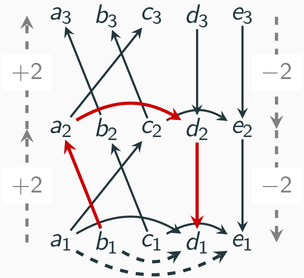




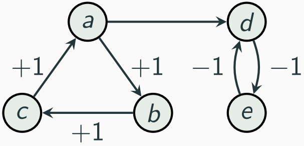
Bell-Shape Reachability

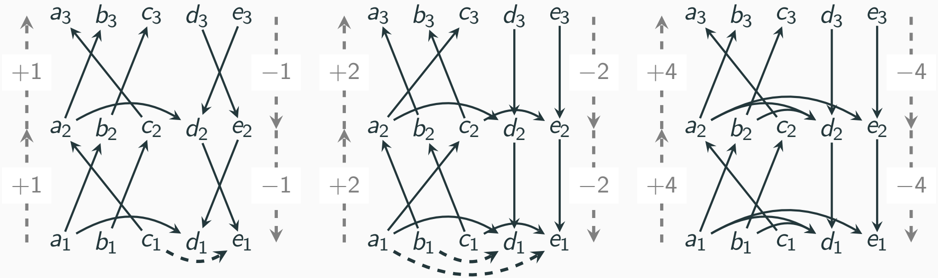




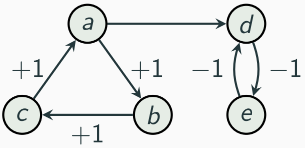


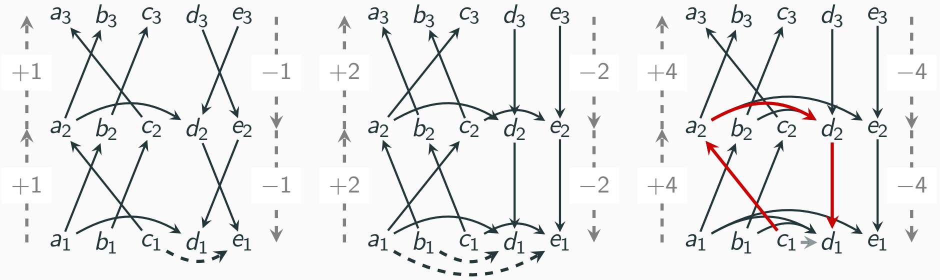
Bell-Shape Reachability



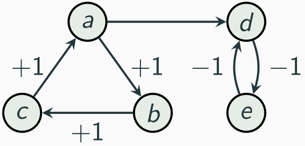


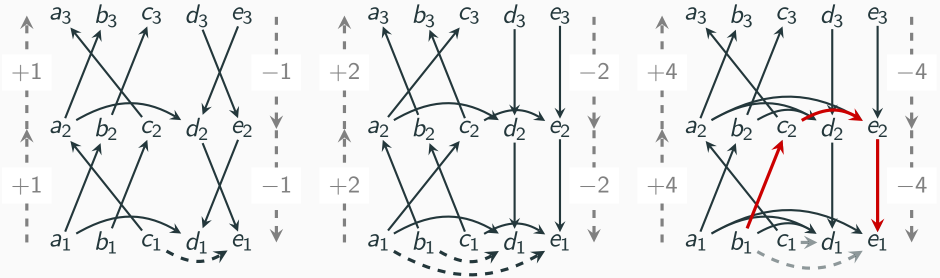
Bell-Shape Reachability





Bell-Shape Reachability





Bell-Shape Reachability

How long did it take?

In each iteration

- Standard transitive closure
- $O(n^{\omega})$ time
- How many iterations?

How long did it take?

In each iteration

- Standard transitive closure
- $O(n^{\omega})$ time
- How many iterations?

- Every iteration doubles the counter value
- Max counter value $\leq$ path length $\leq n^2$
- Only $2 \log n$ iterations

Bell-Shape Reachability

How long did it take?

In each iteration

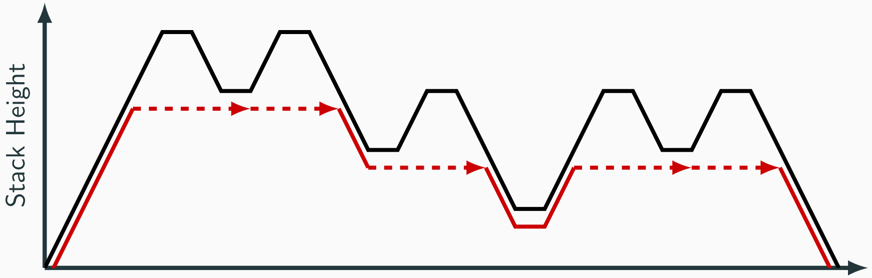
- Standard transitive closure
- $O(n^\omega)$ time
- How many iterations?

- Every iteration doubles the counter value
- Max counter value $\leq$ path length $\leq n^2$
- Only $2 \log n$ iterations

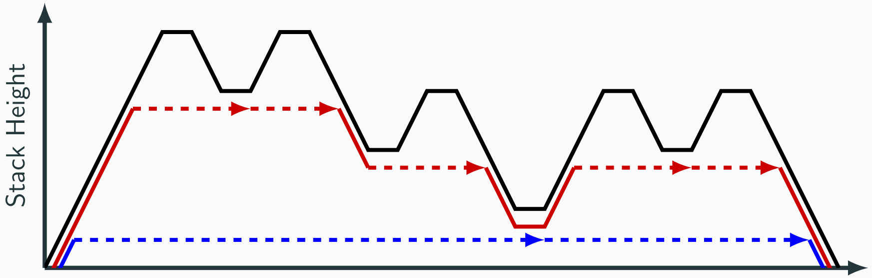
Total time $O(n^\omega \cdot \log n)$

General Case

General Case

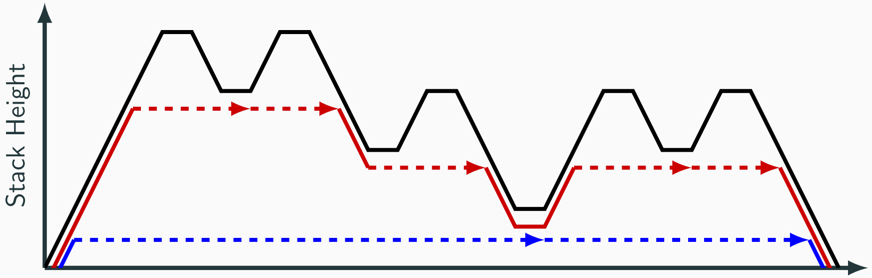


General Case



Repeat!

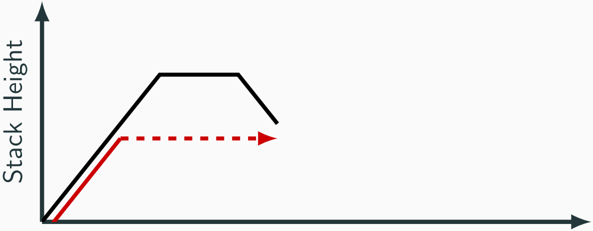
General Case



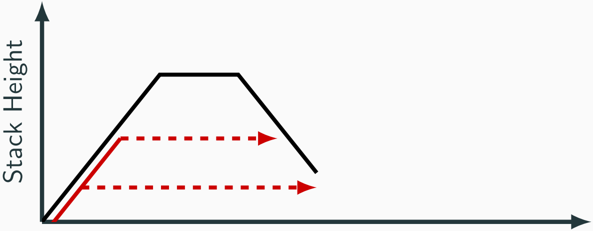
Repeat!

How many iterations?

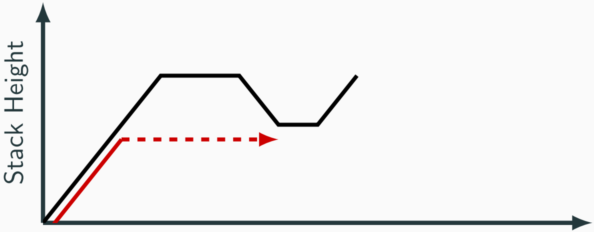
Counting Bells



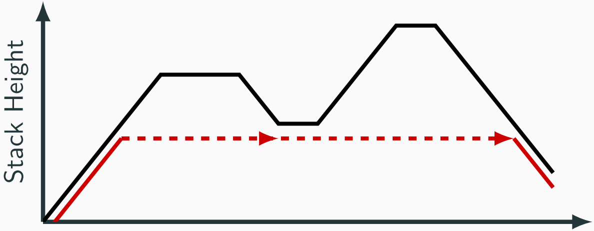
Counting Bells



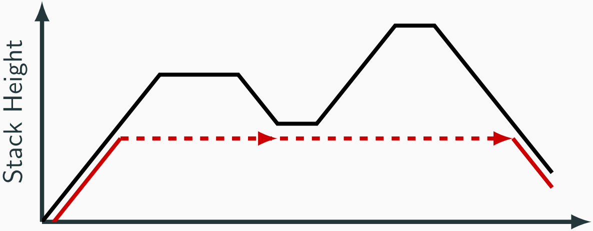
Counting Bells



Counting Bells

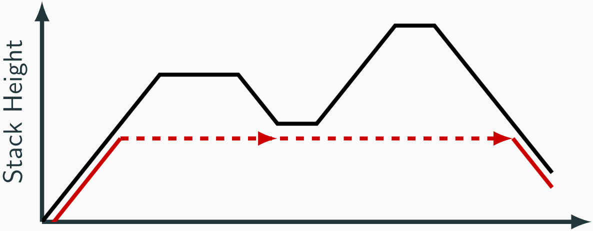


Counting Bells



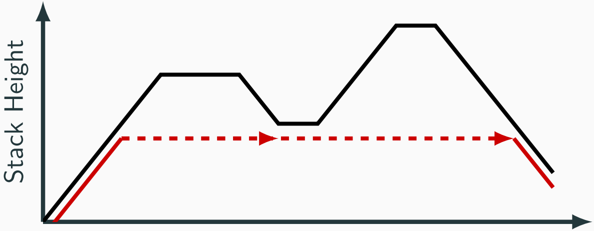
- #bells halves in each iteration
- initial #bells $\leq$ path length $= O(n^2)$

Counting Bells



- #bells halves in each iteration
- initial #bells $\leq$ path length $= O(n^2)$
- $O(\log n)$ iterations
- $O(n^\omega \cdot \log n)$ per iteration

Counting Bells



- #bells halves in each iteration
- initial #bells $\leq$ path length $= O(n^2)$
- $O(\log n)$ iterations
- $O(n^\omega \cdot \log n)$ per iteration

Total time $n^\omega \cdot \log^2 n$