The Decidability and Complexity of Interleaved Bidirected Dyck Reachability

Adam Husted Kjelstrøm and **Andreas Pavlogiannis**

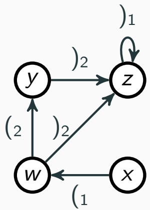
AARHUS UNIVERSITY

Dyck Reachability

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

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

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



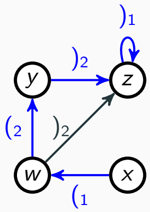
Dyck Reachability

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

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

$$P : x \rightsquigarrow z \quad \text{with} \quad \lambda(P) = (_1(_2)_2)_1$$

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



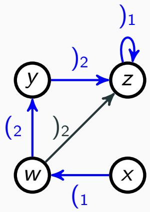
Dyck Reachability

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

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

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

$$P : x \rightsquigarrow z \quad \text{with} \quad \lambda(P) = (_1(_2)_2)_1$$



- Alias analysis
- Data-dependence analysis
- Data-flow analysis
- Databases
- ...

- Impact analysis
- Bloat analysis
- Program slicing
- Network analysis
- ...

Dyck Reachability

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

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

15:00 - 16:00	**Tutorials 1 at** Independence **Chair(s):** Ning Luo	TutorialFest
15:00 **60m** *Tutorial*	★ **Program Analysis via Graph Reachability: Past, Present, and Future [Part A]** Thomas Reps University of Wisconsin--Madison	REMOTE
16:20 - 17:50	**Tutorials 2 at** Independence **Chair(s):** Ning Luo	TutorialFest
16:20 **90m** *Tutorial*	★ **Program Analysis via Graph Reachability: Past, Present, and Future [Part B]** Qirun Zhang Georgia Institute of Technology, USA	INPERSON

- Data-dependence analysis
- Data-flow analysis
- Databases
- ...

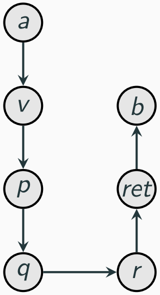
- Bloat analysis
- Program slicing
- Network analysis
- ...

Interleaved Dyck Reachability

```
void setX(Point p, int v){
  p.x = v;
}
int getX(Point r){
  return r.x;
}
...
int a,b;
Point q;
setX(q, a);
b=getX(q);
```

Interleaved Dyck Reachability

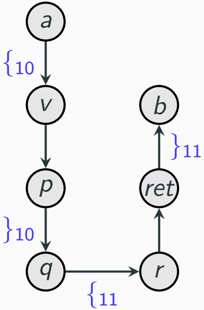
```
void setX(Point p, int v){
  p.x = v;
}
int getX(Point r){
  return r.x;
}
...
int a,b;
Point q;
setX(q, a);
b=getX(q);
```



Interleaved Dyck Reachability

```
void setX(Point p, int v){
  p.x = v;
}
int getX(Point r){
  return r.x;
}
...
int a,b;
Point q;
setX(q, a);
b=getX(q);
```

Context sensitivity

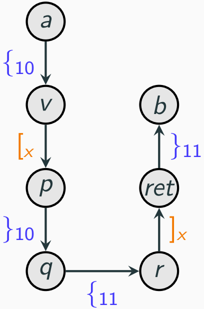


Interleaved Dyck Reachability

```
void setX(Point p, int v){
  p.x = v;
}
int getX(Point r){
  return r.x;
}
...
int a,b;
Point q;
setX(q, a);
b=getX(q);
```

Context sensitivity

Field sensitivity

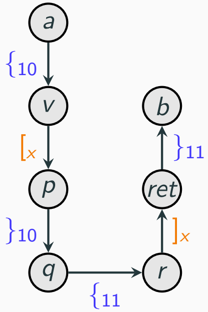


Interleaved Dyck Reachability

```
void setX(Point p, int v){
  p.x = v;
}
int getX(Point r){
  return r.x;
}
...
int a,b;
Point q;
setX(q, a);
b=getX(q);
```

Context sensitivity

Field sensitivity

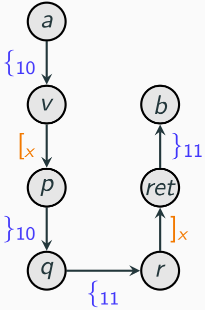


Interleaved Dyck Reachability

```
void setX(Point p, int v){
  p.x = v;
}
int getX(Point r){
  return r.x;
}
...
int a,b;
Point q;
setX(q, a);
b=getX(q);
```

Context sensitivity

Field sensitivity



- Reachability wrt two Dyck languages $\mathcal{D}_k^1$, $\mathcal{D}_k^2$
- $\{_{10}\}_{10}\{_{11}\}_{11} \in \mathcal{D}_k^1$ and $[_x]_x \in \mathcal{D}_k^2$
- Thus $\{_{10}[_x\}_{10}\{_{11}]_x\}_{11} \in \mathcal{D}_k^1 \odot \mathcal{D}_k^2$

Interleaved Dyck Reachability

- Interleaved Dyck Reachability has large modeling power in static analysis

Interleaved Dyck Reachability

- Interleaved Dyck Reachability has large modeling power in static analysis
- Perhaps "too" large

Theorem (Reps '00)

$\mathcal{D}_k \odot \mathcal{D}_k$ *reachability is undecidable*

Interleaved Dyck Reachability

- Interleaved Dyck Reachability has large modeling power in static analysis
- Perhaps "too" large

Theorem (Reps '00)

$\mathcal{D}_k \odot \mathcal{D}_k$ reachability is undecidable

- Still highly used in practice — approximations, e.g.,
  - Q. Zhang and Z. Su. "Context-Sensitive Data-Dependence Analysis via Linear Conjunctive Language Reachability". In: *SIGPLAN Not.* 1 (Jan. 2017), pp. 344–358.
  - J. Späth, K. Ali, and E. Bodden. "Context-, Flow-, and Field-Sensitive Data-Flow Analysis Using Synchronized Pushdown Systems". In: *Proc. ACM Program. Lang.* POPL (Jan. 2019).
  - K. Ferles, J. Stephens, and I. Dillig. "Verifying Correct Usage of Context-Free API Protocols". In: *Proc. ACM Program. Lang.* POPL (Jan. 2021).

  ...

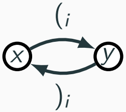
Bidirected graphs

$$\begin{tikzcd} x \arrow[r, bend left, "(_i"] & y \arrow[l, bend left, ")_i"] \end{tikzcd}$$

$$\forall u, v \in V : \ \lambda(u, v) = (_i \quad \Longleftrightarrow \quad \lambda(v, u) =)_i$$

Bidirected graphs

$$\forall u, v \in V : \ \lambda(u, v) = (_i \quad \Longleftrightarrow \quad \lambda(v, u) =)_i$$



- Used to handle mutable heap data
- Demand-driven alias analysis
- CFL formulation of pointer analysis

M. Sridharan et al. "Demand-driven Points-to Analysis for Java". In: *OOPSLA*. 2005.

J. Lu and J. Xue. "Precision-Preserving yet Fast Object-Sensitive Pointer Analysis with Partial Context Sensitivity". In: *Proc. ACM Program. Lang.* OOPSLA (Oct. 2019).

A. Milanova. "FlowCFL: Generalized Type-Based Reachability Analysis". In: *Proc. ACM Program. Lang.* OOPSLA (Nov. 2020).

T. Reps. "Program Analysis via Graph Reachability". In: *Proceedings of the 1997 International Symposium on Logic Programming*. ILPS. 1997, pp. 5–19.

...

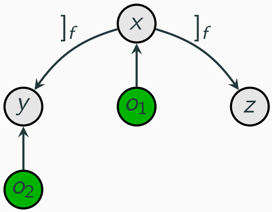
Inclusion-based Alias Analysis

- A Dyck-reachability formulation
- If heap object o Dyck-reaches variable x then $o \in \mathit{PointsToSet}(x)$

Inclusion-based Alias Analysis

- A Dyck-reachability formulation
- If heap object o Dyck-reaches variable x then $o \in \mathit{PointsToSet}(x)$

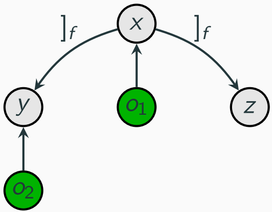
```
x = new O(); //Object o1
y = new O(); //Object o2
...
y = x.f;
z = x.f;
```



Inclusion-based Alias Analysis

- A Dyck-reachability formulation
- If heap object o Dyck-reaches variable x then $o \in PointsToSet(x)$

```
x = new O(); //Object o1
y = new O(); //Object o2
...
y = x.f;
z = x.f;
```



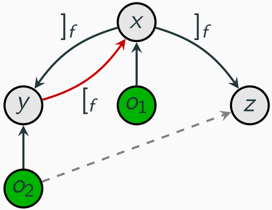
Inclusion based

If y may alias z and $o \in PointsToSet(y)$ then $o \in PointsToSet(z)$

Inclusion-based Alias Analysis

- A Dyck-reachability formulation
- If heap object o Dyck-reaches variable x then $o \in PointsToSet(x)$

```
x = new O(); //Object o1
y = new O(); //Object o2
...
y = x.f;
z = x.f;
```



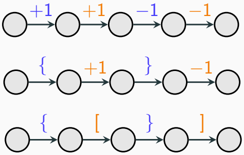
Inclusion based

If y may alias z and $o \in PointsToSet(y)$ then $o \in PointsToSet(z)$

Interleaved Bidirected Dyck Reachability

Interleaving + Bidirectedness, Variants →

1. $\mathcal{D}_1 \odot \mathcal{D}_1$
2. $\mathcal{D}_k \odot \mathcal{D}_1$
3. $\mathcal{D}_k \odot \mathcal{D}_k$



Interleaved Bidirected Dyck Reachability

Interleaving + Bidirectedness, Variants $\rightarrow$

1. $\mathcal{D}_1 \odot \mathcal{D}_1$ $\quad \bigcirc \xrightarrow{+1} \bigcirc \xrightarrow{+1} \bigcirc \xrightarrow{-1} \bigcirc \xrightarrow{-1} \bigcirc$
2. $\mathcal{D}_k \odot \mathcal{D}_1$ $\quad \bigcirc \xrightarrow{\{} \bigcirc \xrightarrow{+1} \bigcirc \xrightarrow{\}} \bigcirc \xrightarrow{-1} \bigcirc$
3. $\mathcal{D}_k \odot \mathcal{D}_k$ $\quad \bigcirc \xrightarrow{\{} \bigcirc \xrightarrow{[} \bigcirc \xrightarrow{\}} \bigcirc \xrightarrow{]} \bigcirc$

What do we know about this problem?

- $\mathcal{D}_1 \odot \mathcal{D}_1$ yields a 2-dimensional Vector Addition System with States (VASS)
- Reachability in NL

  M. Englert, R. Lazić, and P. Totzke. "Reachability in Two-Dimensional Unary Vector Addition Systems with States is NL-Complete". In: *LICS*. 2016.

- Bidirected $\mathcal{D}_1 \odot \mathcal{D}_1$ in $O(n^7)$
- Bidirected $\mathcal{D}_k \odot \mathcal{D}_k$ is NP-hard

  Y. Li, Q. Zhang, and T. Reps. "On the Complexity of Bidirected Interleaved Dyck-Reachability". In: *Proc. ACM Program. Lang.* POPL (Jan. 2021).

Main Results

	$\mathcal{D}_1 \odot \mathcal{D}_1$	$\mathcal{D}_k \odot \mathcal{D}_1$	$\mathcal{D}_k \odot \mathcal{D}_1$ (bounded counter)	$\mathcal{D}_k \odot \mathcal{D}_k$
Upper Bound				-
Lower Bound	-	-		

Main Results

	$\mathcal{D}_1 \odot \mathcal{D}_1$	$\mathcal{D}_k \odot \mathcal{D}_1$	$\mathcal{D}_k \odot \mathcal{D}_1$ (bounded counter)	$\mathcal{D}_k \odot \mathcal{D}_k$
Upper Bound	$O(n^3 \cdot \alpha(n))^\dagger$			-
Lower Bound	-	-		

† Improves over previous $O(n^7)$

Main Results

	$\mathcal{D}_1 \odot \mathcal{D}_1$	$\mathcal{D}_k \odot \mathcal{D}_1$	$\mathcal{D}_k \odot \mathcal{D}_1$ (bounded counter)	$\mathcal{D}_k \odot \mathcal{D}_k$
Upper Bound	$O(n^3 \cdot \alpha(n))^\dagger$	Decidable‡		-
Lower Bound	-	-		

† Improves over previous $O(n^7)$

‡ Without bidirectedness, decidability is open (PVASS)

Main Results

	$\mathcal{D}_1 \odot \mathcal{D}_1$	$\mathcal{D}_k \odot \mathcal{D}_1$	$\mathcal{D}_k \odot \mathcal{D}_1$ (bounded counter)	$\mathcal{D}_k \odot \mathcal{D}_k$
Upper Bound	$O(n^3 \cdot \alpha(n))^\dagger$	Decidable‡	$O(n^2 \cdot \alpha(n))$	-
Lower Bound	-	-		

† Improves over previous $O(n^7)$

‡ Without bidirectedness, decidability is open (PVASS)

Main Results

	$\mathcal{D}_1 \odot \mathcal{D}_1$	$\mathcal{D}_k \odot \mathcal{D}_1$	$\mathcal{D}_k \odot \mathcal{D}_1$ (bounded counter)	$\mathcal{D}_k \odot \mathcal{D}_k$
Upper Bound	$O(n^3 \cdot \alpha(n))^\dagger$	Decidable‡	$O(n^2 \cdot \alpha(n))$	-
Lower Bound	-	-	OV-hard$^\diamond$	

† Improves over previous $O(n^7)$

‡ Without bidirectedness, decidability is open (PVASS)

$\diamond$ $O(n^2)$ is tight (wrt polynomial improvements)

Main Results

	$\mathcal{D}_1 \odot \mathcal{D}_1$	$\mathcal{D}_k \odot \mathcal{D}_1$	$\mathcal{D}_k \odot \mathcal{D}_1$ (bounded counter)	$\mathcal{D}_k \odot \mathcal{D}_k$
Upper Bound	$O(n^3 \cdot \alpha(n))^\dagger$	Decidable‡	$O(n^2 \cdot \alpha(n))$	-
Lower Bound	-	-	OV-hard$^\diamond$	Undecidable*

- $\dagger$ Improves over previous $O(n^7)$
- $\ddagger$ Without bidirectedness, decidability is open (PVASS)
- $\diamond$ $O(n^2)$ is tight (wrt polynomial improvements)
- $*$ Improves over previous NP-hard

Main Results

	$\mathcal{D}_1 \odot \mathcal{D}_1$	$\mathcal{D}_k \odot \mathcal{D}_1$	$\mathcal{D}_k \odot \mathcal{D}_1$ (bounded counter)	$\mathcal{D}_k \odot \mathcal{D}_k$
Upper Bound	$\mathbf{O(n^3 \cdot \alpha(n))}^\dagger$	Decidable‡	$O(n^2 \cdot \alpha(n))$	-
Lower Bound	-	-	OV-hard$^\diamond$	**Undecidable***

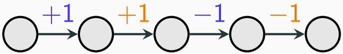
$\dagger$ Improves over previous $O(n^7)$

$\ddagger$ Without bidirectedness, decidability is open (PVASS)

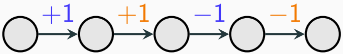
$\diamond$ $O(n^2)$ is tight (wrt polynomial improvements)

$*$ Improves over previous NP-hard

$\mathcal{D}_1 \odot \mathcal{D}_1$



$\mathcal{D}_1 \odot \mathcal{D}_1$

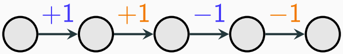


Theorem

Bidirected $\mathcal{D}_1 \odot \mathcal{D}_1$ reachability can be solved in $O(n^3 \cdot \alpha(n))$ time.

- *n is the number of nodes*
- *$\alpha(\cdot)$ is the inverse Ackermann function (practically constant)*

$\mathcal{D}_1 \odot \mathcal{D}_1$



Theorem

Bidirected $\mathcal{D}_1 \odot \mathcal{D}_1$ reachability can be solved in $O(n^3 \cdot \alpha(n))$ time.

- *n is the number of nodes*
- *$\alpha(\cdot)$ is the inverse Ackermann function (practically constant)*

Lemma

Without loss of generality, both counters along any witness path $P: u \rightsquigarrow v$ remain bounded by $O(n^2)$.

One Key Idea

Bidirectedness $\implies$ Boundedness

If u reaches v then there is a witness path where the counters are bounded.

$\implies$ limits the search space for witness paths

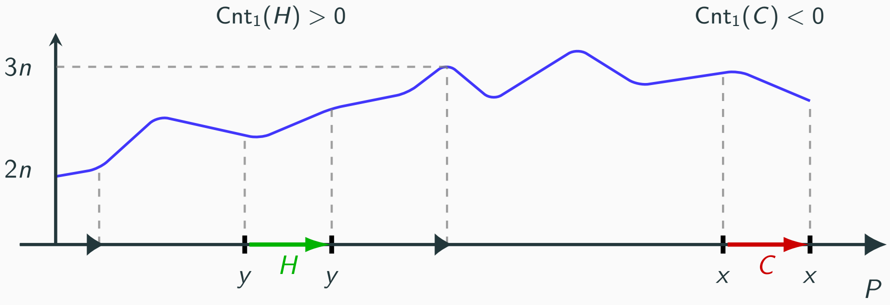
One Key Idea

Bidirectedness $\implies$ Boundedness

If u reaches v then there is a witness path where the counters are bounded.

$\implies$ limits the search space for witness paths

If we had just 1 counter (instead of 2):



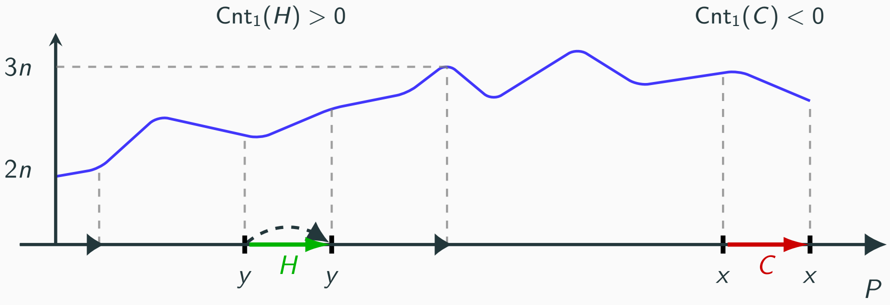
One Key Idea

Bidirectedness $\implies$ Boundedness

If u reaches v then there is a witness path where the counters are bounded.

$\implies$ limits the search space for witness paths

If we had just 1 counter (instead of 2):



With 2 counters, more involved, gives $O(n^2)$ bound instead

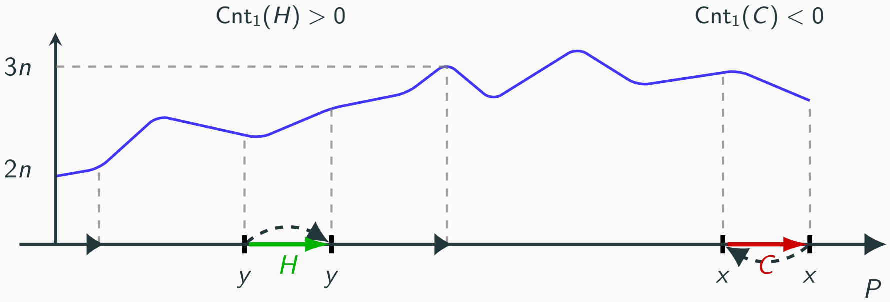
One Key Idea

Bidirectedness $\implies$ Boundedness

If u reaches v then there is a witness path where the counters are bounded.

$\implies$ limits the search space for witness paths

If we had just 1 counter (instead of 2):



With 2 counters, more involved, gives $O(n^2)$ bound instead

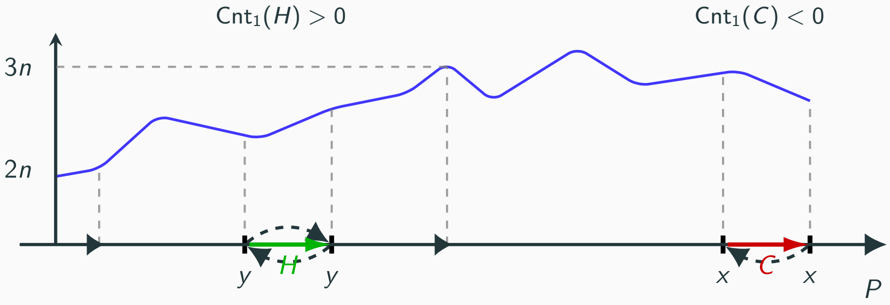
One Key Idea

Bidirectedness $\implies$ Boundedness

If u reaches v then there is a witness path where the counters are bounded.

$\implies$ limits the search space for witness paths

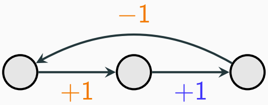
If we had just 1 counter (instead of 2):



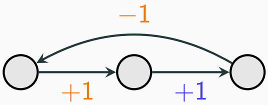
With 2 counters, more involved, gives $O(n^2)$ bound instead

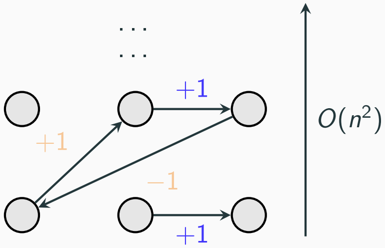
Using the $O(n^2)$ Counter Bound

Using the $O(n^2)$ Counter Bound

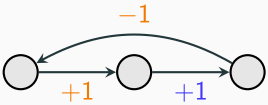


Using the $O(n^2)$ Counter Bound

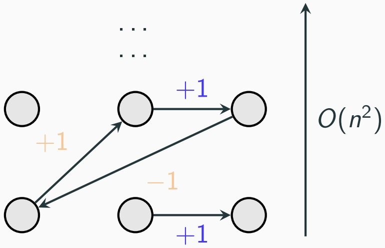




Using the $O(n^2)$ Counter Bound

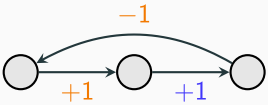


$\mathcal{D}_1 \odot \mathcal{D}_1$ on G
with n nodes

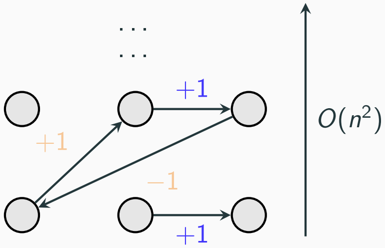


$\mathcal{D}_1$ on G'
with n^3 nodes

Using the $O(n^2)$ Counter Bound



$\mathcal{D}_1 \odot \mathcal{D}_1$ on G
with n nodes

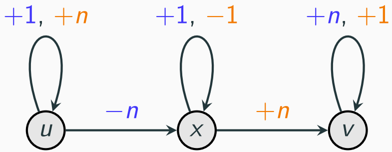


$\mathcal{D}_1$ on G'
with n^3 nodes

K. Chatterjee, B. Choudhary, and A. Pavlogiannis. "Optimal Dyck Reachability for Data-Dependence and Alias Analysis". In: *Proc. ACM Program. Lang.* POPL (Dec. 2018).

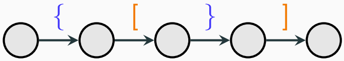
Total time $O(n^3\alpha(n))$

The Counter Bound $O(n^2)$ is Tight

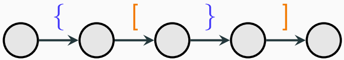


Both counters reach a quadratic value

Bidirected $\mathcal{D}_k \odot \mathcal{D}_k$



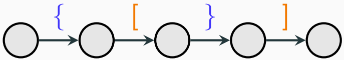
Bidirected $\mathcal{D}_k \odot \mathcal{D}_k$



Theorem

Bidirected $\mathcal{D}_k \odot \mathcal{D}_k$ reachability is undecidable.

Bidirected $\mathcal{D}_k \odot \mathcal{D}_k$



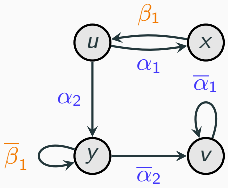
Theorem

Bidirected $\mathcal{D}_k \odot \mathcal{D}_k$ reachability is undecidable.

- Even bidirected formalisms of context + field sensitivity are undecidable
- Need coarser approximations
- Or just techniques that work well in practice

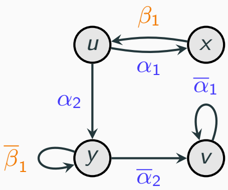
Undecidability - Sketch

Directed

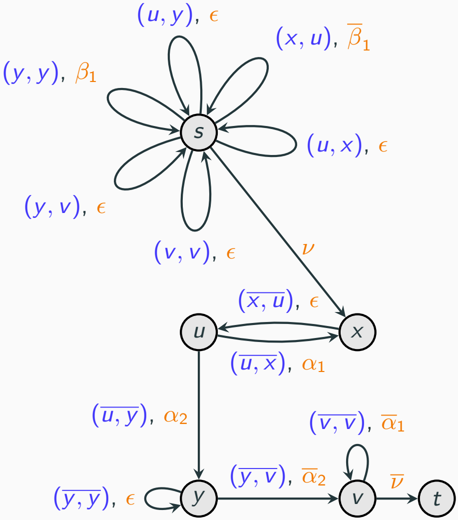


Undecidability - Sketch

Directed

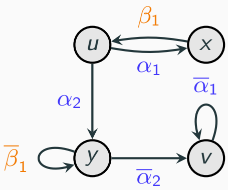


Bidirected

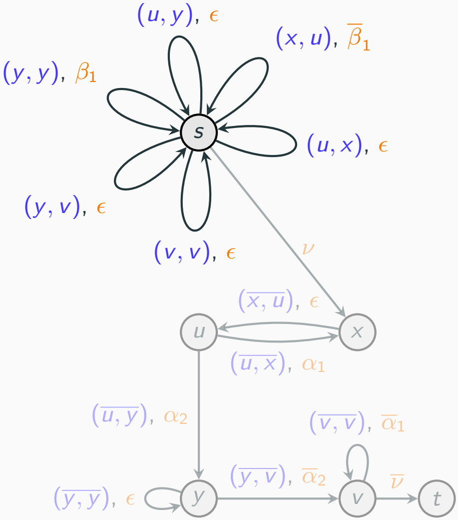


Undecidability - Sketch

Directed

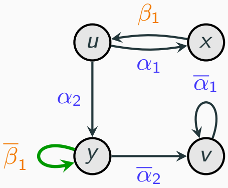


Bidirected

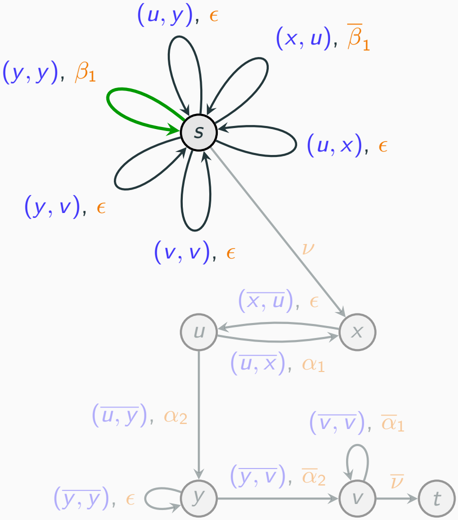


Undecidability - Sketch

Directed

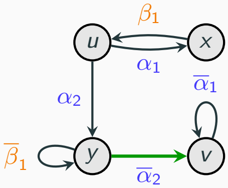


Bidirected

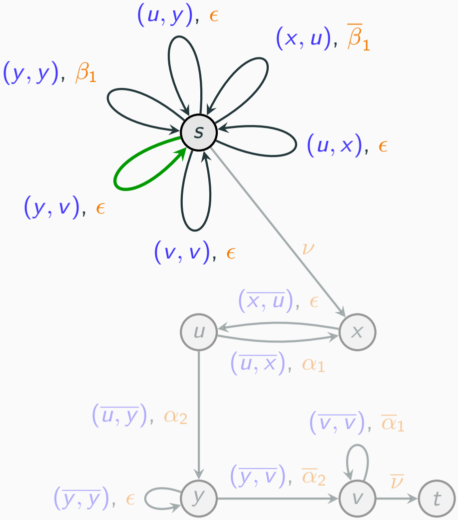


Undecidability - Sketch

Directed

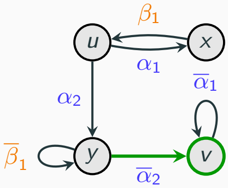


Bidirected

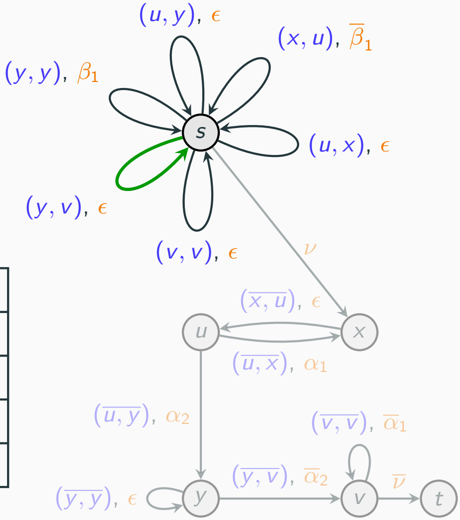


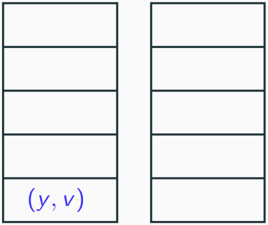
Undecidability - Sketch

Directed



Bidirected

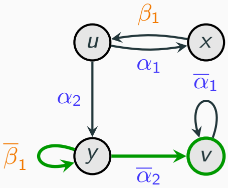




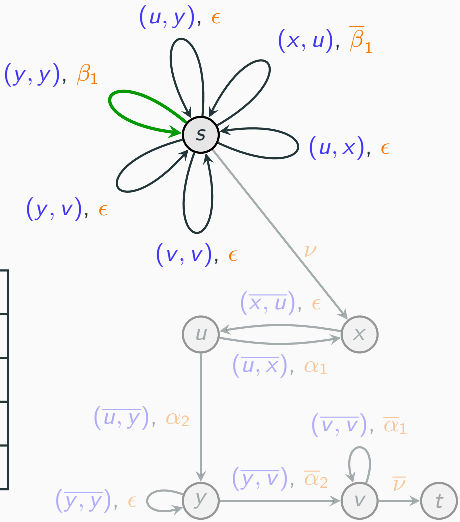
Stack 1 Stack 2

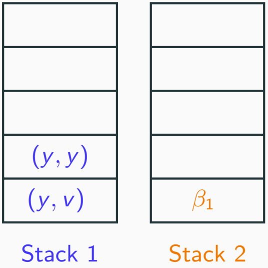
Undecidability - Sketch

Directed



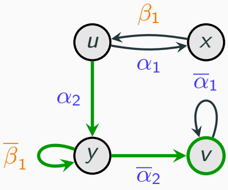
Bidirected

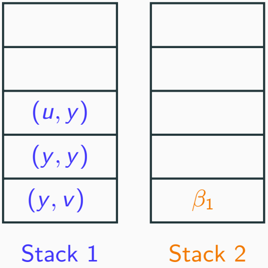




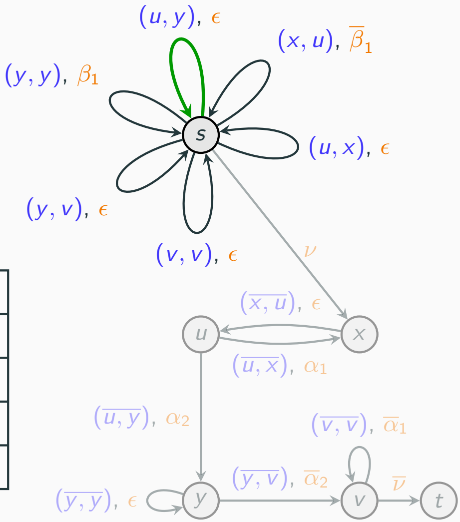
Undecidability - Sketch

Directed



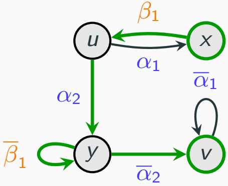


Bidirected

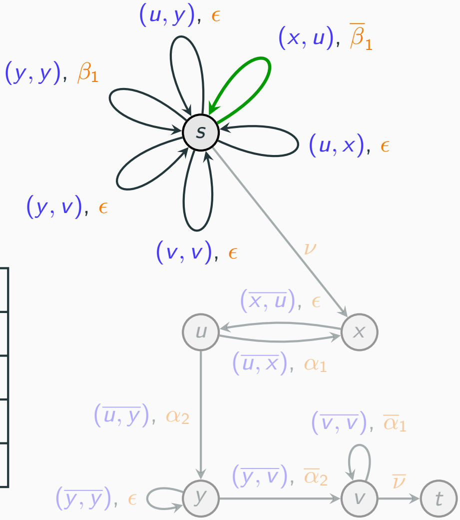


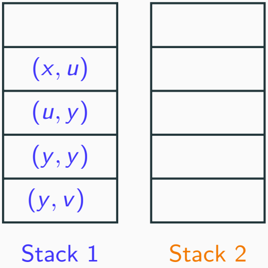
Undecidability - Sketch

Directed

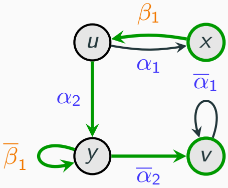


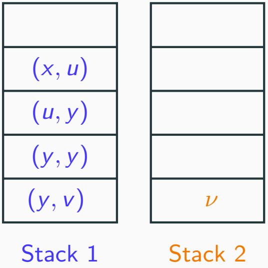
Bidirected



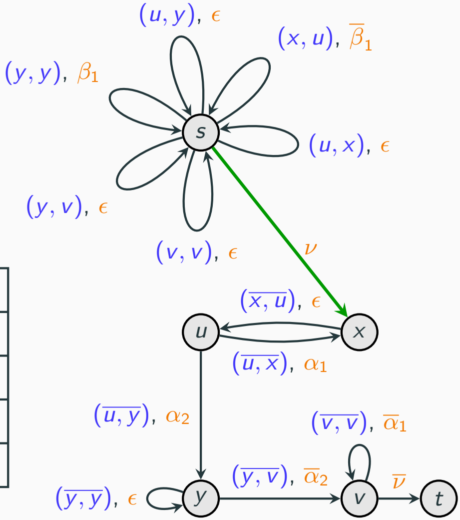


Directed



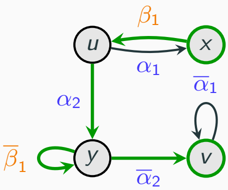


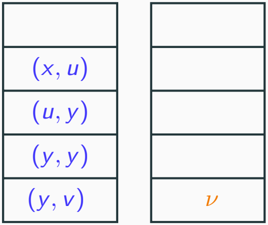
Bidirected



Undecidability - Sketch

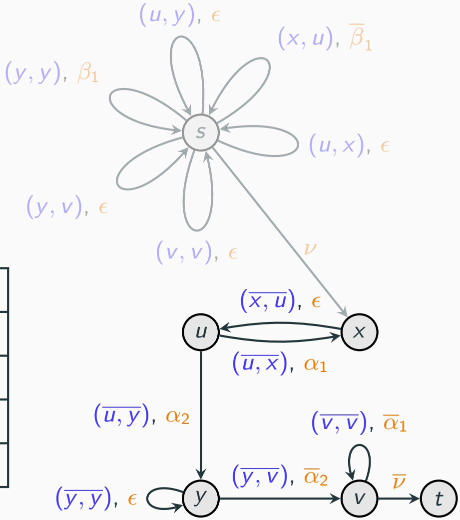
Directed





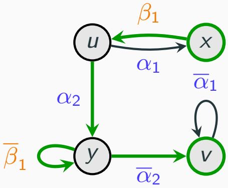
Stack 1 Stack 2

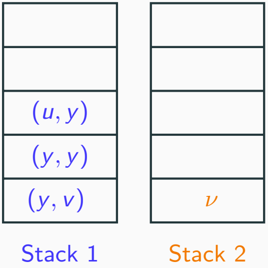
Bidirected



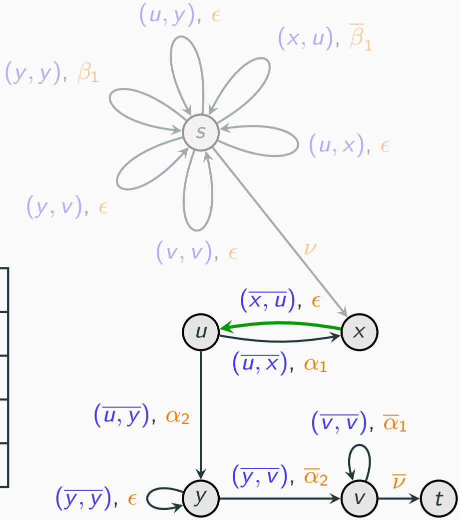
Undecidability - Sketch

Directed

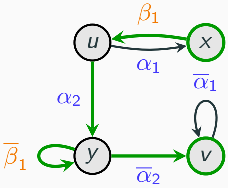


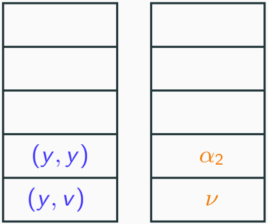


Bidirected



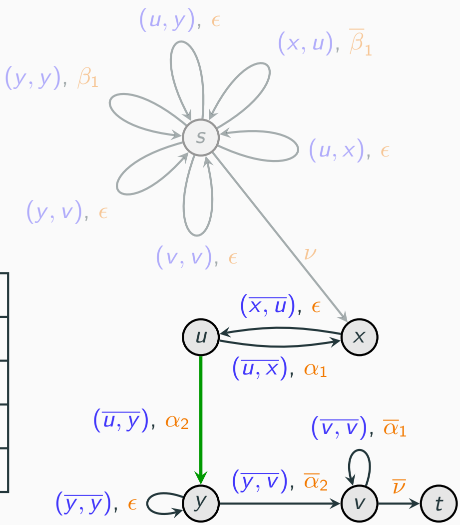
Directed





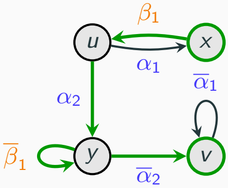
Stack 1 Stack 2

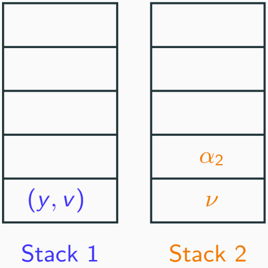
Bidirected



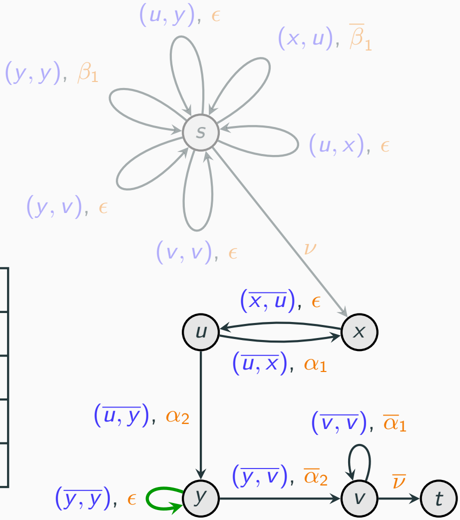
Undecidability - Sketch

Directed



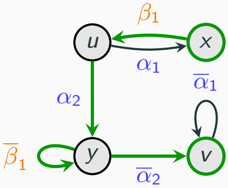


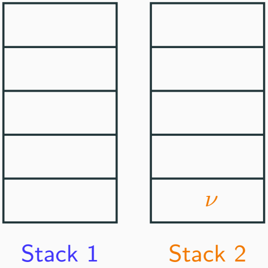
Bidirected



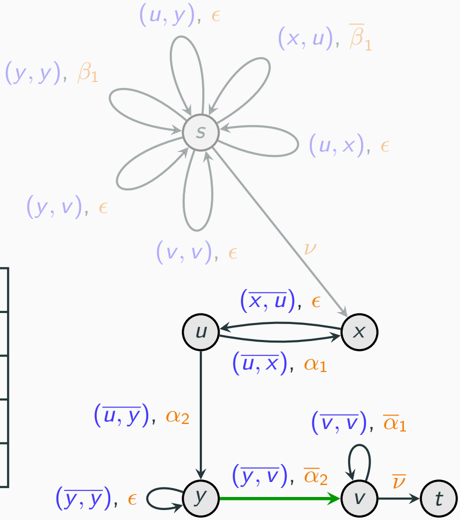
Undecidability - Sketch

Directed



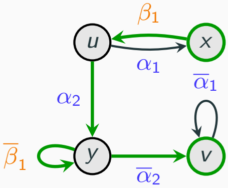


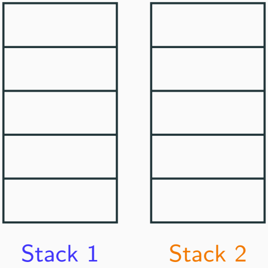
Bidirected



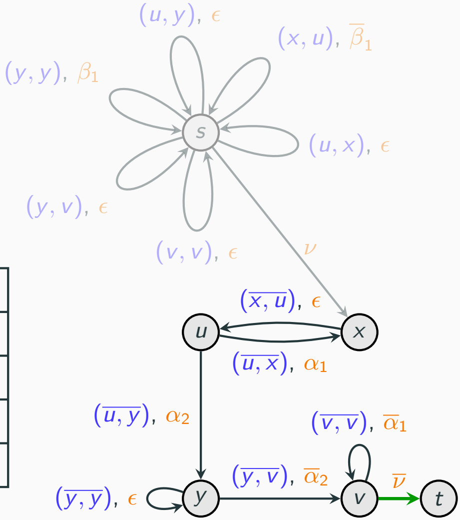
Undecidability - Sketch

Directed





Bidirected



Experiments

Experiments

Setup

- DaCapo benchmarks
- (field + context)-sensitive alias analysis $\rightarrow$ Bidirected $\mathcal{D}_k \odot \mathcal{D}_k$ reachability
- $\mathcal{D}_1 \odot \mathcal{D}_1$ and bounded $\mathcal{D}_k \odot \mathcal{D}_1$ reachability by abstracting on one language

Experiments

Setup

- DaCapo benchmarks
- (field + context)-sensitive alias analysis $\rightarrow$ Bidirected $\mathcal{D}_k \odot \mathcal{D}_k$ reachability
- $\mathcal{D}_1 \odot \mathcal{D}_1$ and bounded $\mathcal{D}_k \odot \mathcal{D}_1$ reachability by abstracting on one language

Summary

- Previously $\mathcal{D}_1 \odot \mathcal{D}_1$ took more than 2 days
- Now the whole dataset takes $\sim$5 mins on a laptop for each case $\mathcal{D}_1 \odot \mathcal{D}_1$ and $\mathcal{D}_k \odot \mathcal{D}_1$
- Times are usable!

Thank you!

A. H. Kjelstrøm and A. Pavlogiannis. "The Decidability and Complexity of Interleaved Bidirected Dyck Reachability". In: *Proc. ACM Program. Lang.* POPL (Jan. 2022).

Appendix

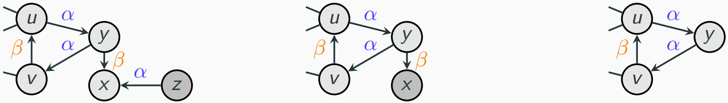
Experiments

Benchmark	n	$\mathcal{D}_1 \odot \mathcal{D}_1$		$\mathcal{D}_k \odot \mathcal{D}_1$ (bounded counter)	
		ID-CCs	**Time (s)**	**ID-CCs**	**Time (s)**
antlr	29831	26793	40.5	27014	9.1
bloat	36181	32693	16.2	32946	12.8
chart	67535	60787	32.8	61158	65.1
eclipse	30981	27812	16.0	27999	12.3
fop	61016	54671	29.8	55012	58.4
hsqldb	27494	24584	15.5	24775	8.8
jython	36162	31811	26.3	32060	21.5
luindex	28595	25610	15.6	25809	7.9
lusearch	29530	26417	17.8	26655	9.1
pmd	31333	28064	18.2	28296	9.9
xalan	27358	24498	15.2	24689	7.4

Quick Heuristics

1. Double-self-loops

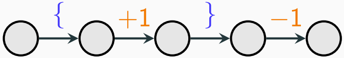
2. Trimming



3. CFL underapproximation

1. Treat $\mathcal{D}_k \odot \mathcal{D}_k$ as one Dyck language over the union alphabet
2. Perform reachability and collapse components
3. Solve $\mathcal{D}_k \odot \mathcal{D}_k$ on the quotient graph

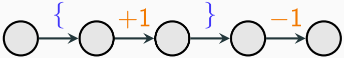
$\mathcal{D}_k \odot \mathcal{D}_1$



Coverability

u covers (v, k) if u can reach v with an empty stack and counter at least k

$\mathcal{D}_k \odot \mathcal{D}_1$



Coverability

u covers (v, k) if u can reach v with an empty stack and counter at least k

Theorem (LST '15)

Coverability in PVASS is decidable.

J. Leroux, G. Sutre, and P. Totzke. "On the Coverability Problem for Pushdown Vector Addition Systems in One Dimension". In: *Automata, Languages, and Programming*. Springer Berlin Heidelberg, 2015, pp. 324–336.

$\mathcal{D}_k \odot \mathcal{D}_1$ Algorithm

1. u covers $(v, 0)$ and v covers $(u, 0)$?

$\mathcal{D}_k \odot \mathcal{D}_1$ Algorithm

1. u covers $(v, 0)$ and v covers $(u, 0)$ ✓

$\mathcal{D}_k \odot \mathcal{D}_1$ Algorithm

1. u covers $(v, 0)$ and v covers $(u, 0)$ ✓
2. u covers $(v, 1)$ and v covers $(u, 1)$?

$\mathcal{D}_k \odot \mathcal{D}_1$ Algorithm

1. u covers $(v, 0)$ and v covers $(u, 0)$ ✓
2. u covers $(v, 1)$ and v covers $(u, 1)$ ✓

$\mathcal{D}_k \odot \mathcal{D}_1$ Algorithm

1. u covers $(v, 0)$ and v covers $(u, 0)$ ✓
2. u covers $(v, 1)$ and v covers $(u, 1)$ ✓
3. Derive a stack height bound $= \max(n^2, \text{height of coverability witnesses})$

$\mathcal{D}_k \odot \mathcal{D}_1$ Algorithm

1. u covers $(v, 0)$ and v covers $(u, 0)$ ✓
2. u covers $(v, 1)$ and v covers $(u, 1)$ ✓
3. Derive a stack height bound $= \max(n^2, \text{height of coverability witnesses})$

$\mathcal{D}_k \odot \mathcal{D}_1$ Algorithm

1. u covers $(v, 0)$ and v covers $(u, 0)$ ✓
2. u covers $(v, 1)$ and v covers $(u, 1)$ ✓
3. Derive a stack height bound $= \max(n^2, \text{height of coverability witnesses})$

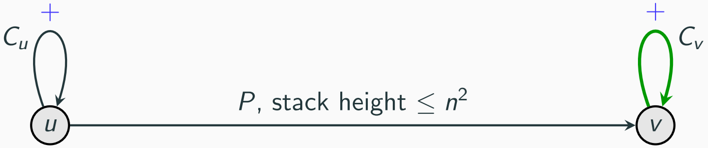
$\mathcal{D}_k \odot \mathcal{D}_1$ Algorithm

1. u covers $(v, 0)$ and v covers $(u, 0)$ ✓
2. u covers $(v, 1)$ and v covers $(u, 1)$ ✓
3. Derive a stack height bound $= \max(n^2$, height of coverability witnesses$)$

$C_u^* \circ P$

$\mathcal{D}_k \odot \mathcal{D}_1$ Algorithm

1. u covers $(v, 0)$ and v covers $(u, 0)$ ✓
2. u covers $(v, 1)$ and v covers $(u, 1)$ ✓
3. Derive a stack height bound $= \max(n^2,$ height of coverability witnesses$)$



$C_u^* \circ P \circ C_v^*$

$\mathcal{D}_k \odot \mathcal{D}_1$ Algorithm

1. u covers $(v, 0)$ and v covers $(u, 0)$ ✓
2. u covers $(v, 1)$ and v covers $(u, 1)$ ✓
3. Derive a stack height bound $= \max(n^2,$ height of coverability witnesses$)$

$C_u^* \circ P \circ C_v^* \circ \overline{P}$

$\mathcal{D}_k \odot \mathcal{D}_1$ Algorithm

1. u covers $(v, 0)$ and v covers $(u, 0)$ ✓
2. u covers $(v, 1)$ and v covers $(u, 1)$ ✓
3. Derive a stack height bound $= \max(n^2, \text{height of coverability witnesses})$

$$C_u^* \circ P \circ C_v^* \circ \overline{P} \circ \overline{C_u^*}$$

$\mathcal{D}_k \odot \mathcal{D}_1$ Algorithm

1. u covers $(v, 0)$ and v covers $(u, 0)$ ✓
2. u covers $(v, 1)$ and v covers $(u, 1)$ ✓
3. Derive a stack height bound $= \max(n^2, \text{height of coverability witnesses})$

$$C_u^* \circ P \circ C_v^* \circ \overline{P} \circ \overline{C_u^*} \circ P$$

$\mathcal{D}_k \odot \mathcal{D}_1$ Algorithm

1. u covers $(v, 0)$ and v covers $(u, 0)$ ✓
2. u covers $(v, 1)$ and v covers $(u, 1)$ ✓
3. Derive a stack height bound $=$ max(n^2, height of coverability witnesses)

$$C_u^* \circ P \circ C_v^* \circ \overline{P} \circ \overline{C_u^*} \circ P \circ \overline{C_v^*}$$