CAV23

Incremental Dead State Detection in Logarithmic Time

Caleb Stanford and Margus Veanes

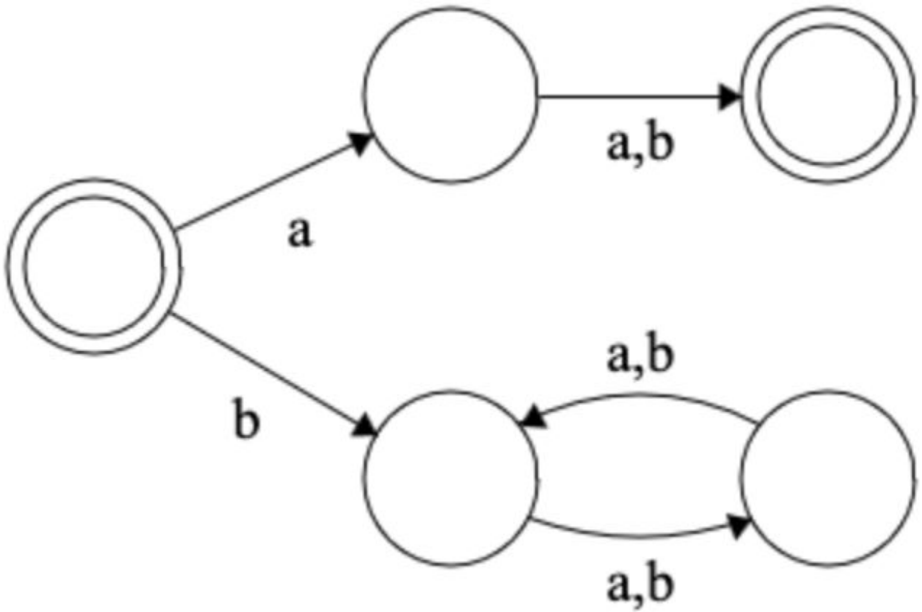
UC DAVIS

Microsoft Research

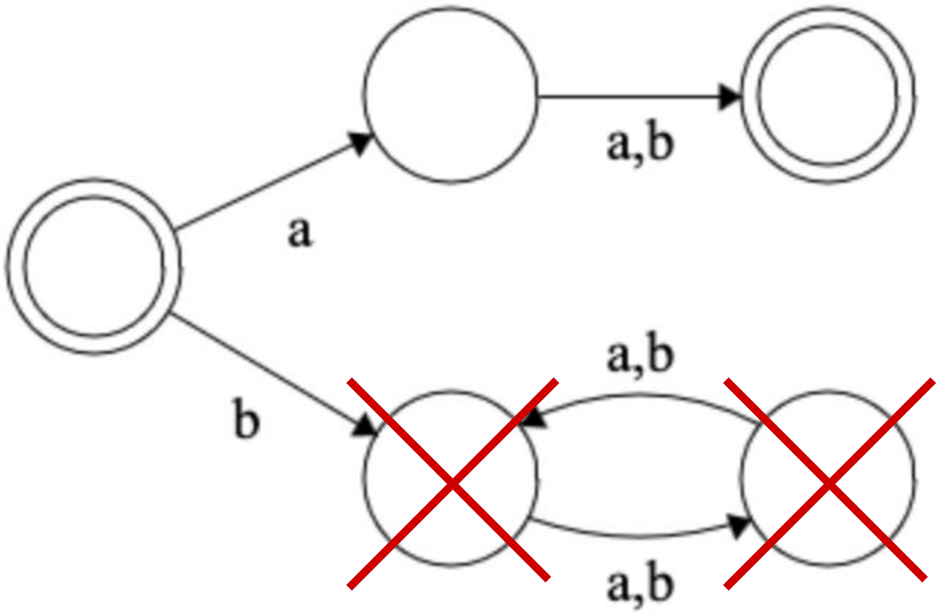
CAV Artifact Evaluation ★ Available

CAV Artifact Evaluation ★★★ Reusable

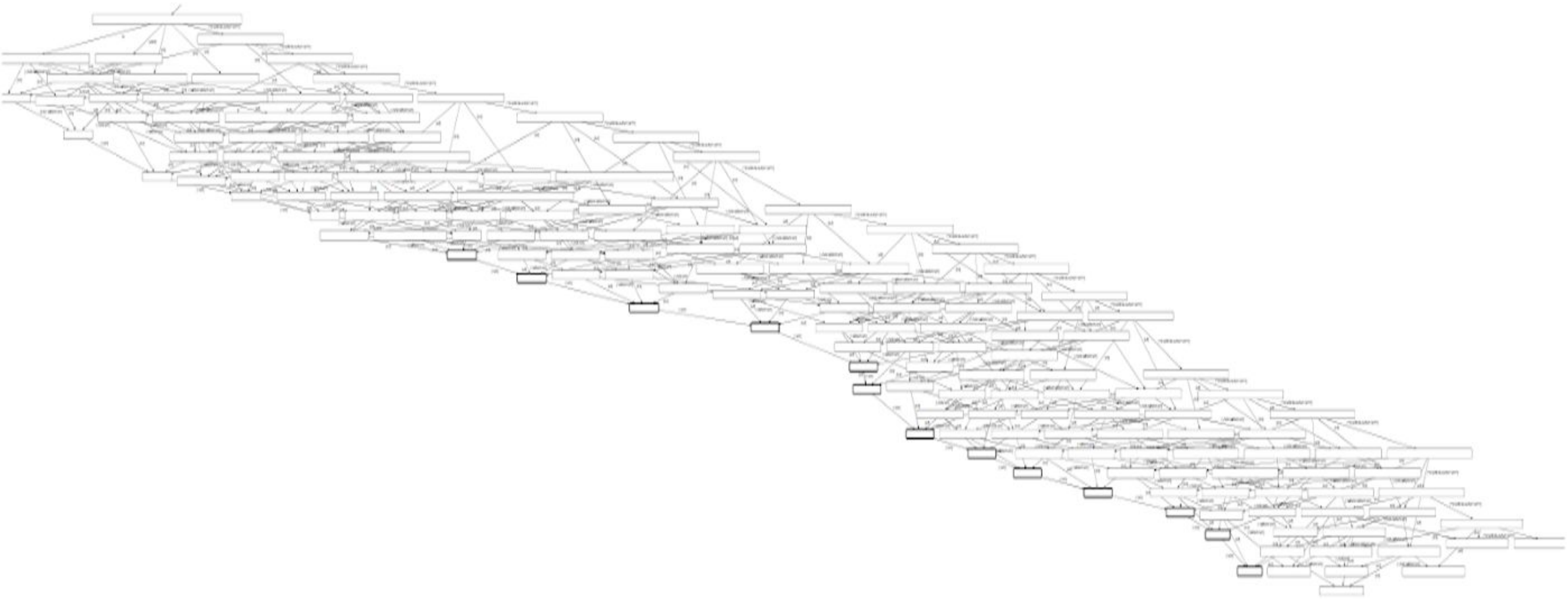
Dead State Detection in Automata



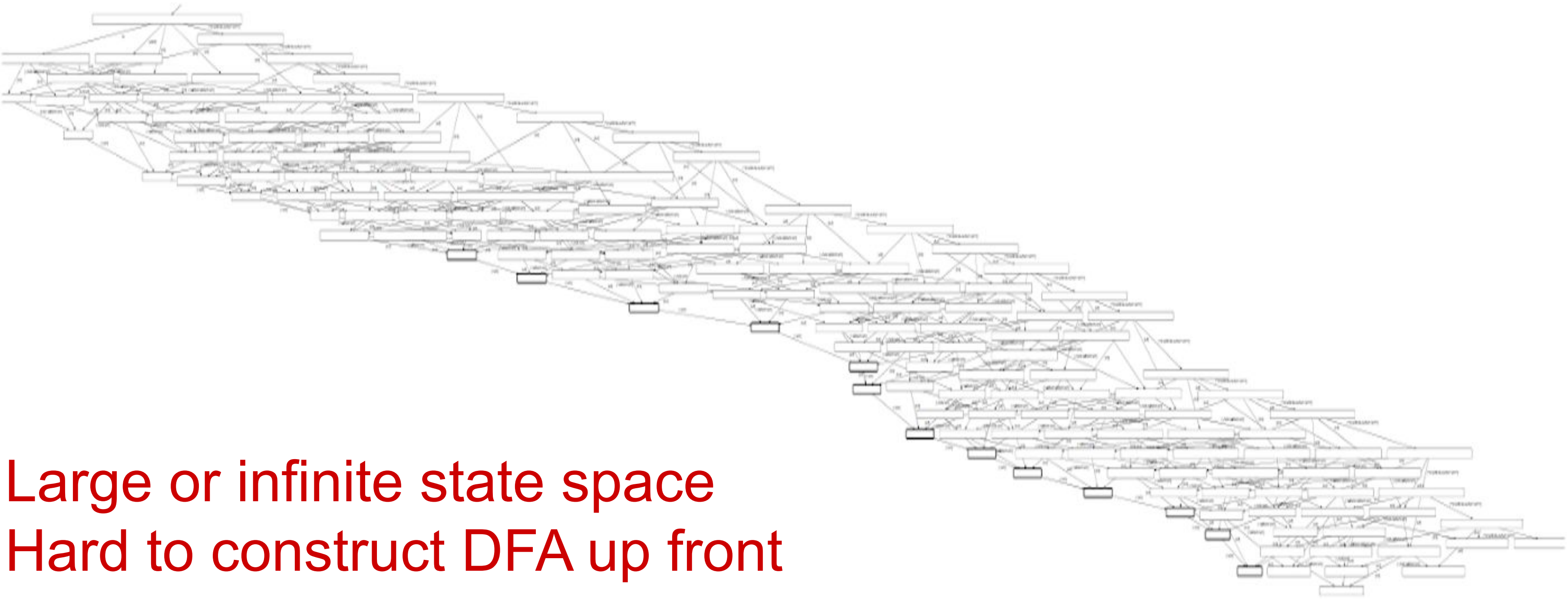
Dead State Detection in Automata



Dead State Detection in Practice

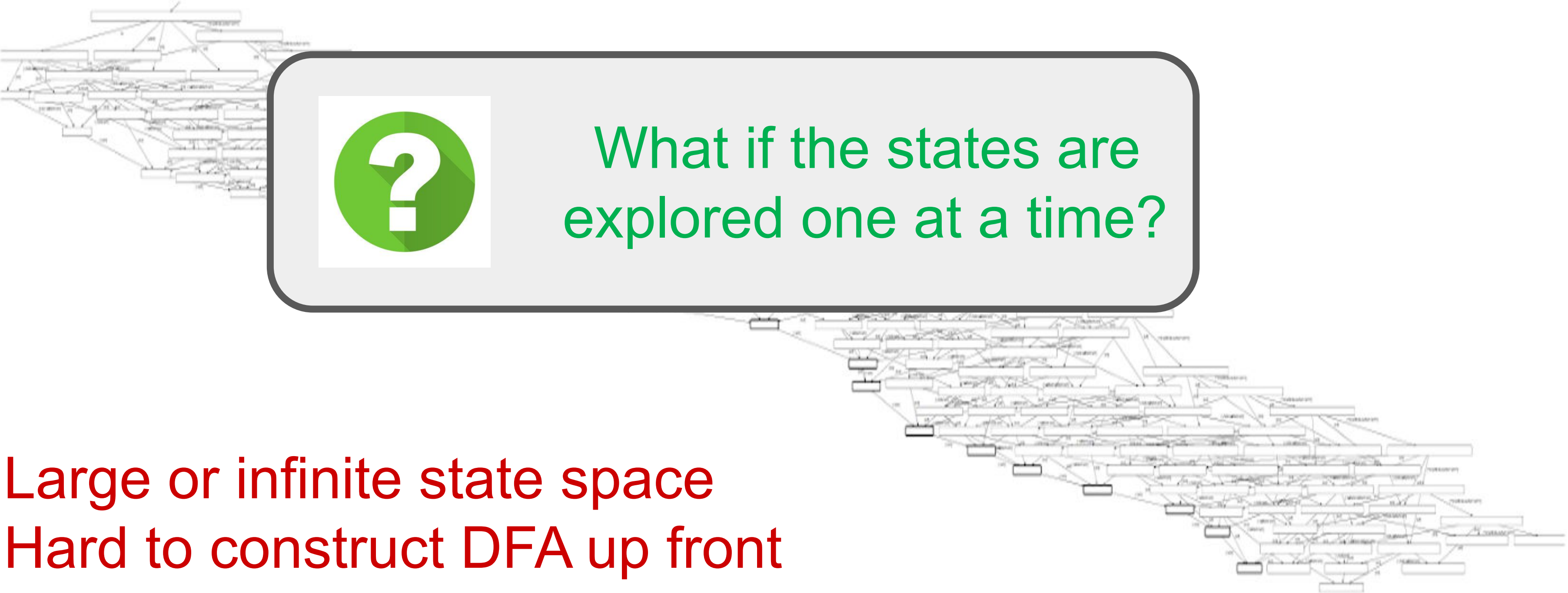


Dead State Detection in Practice



Large or infinite state space
Hard to construct DFA up front

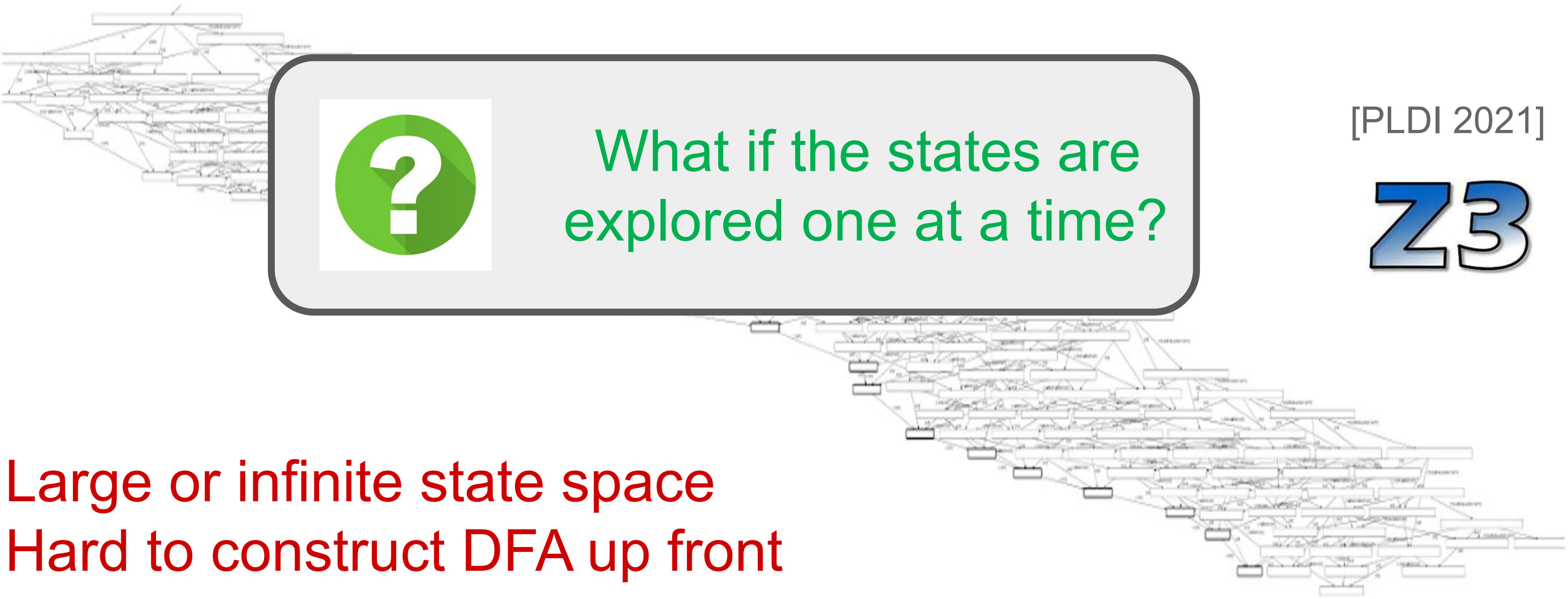
Dead State Detection in Practice



What if the states are explored one at a time?

Large or infinite state space
Hard to construct DFA up front

Dead State Detection in Practice



What if the states are explored one at a time?

[PLDI 2021]

Z3

Large or infinite state space
Hard to construct DFA up front

Existing Solutions

What if the states are explored one at a time?

Best result in online graph algorithms: **O(sqrt m) per edge**

[Bender, Fineman, Gilbert, Tarjan 2015]

Existing Solutions

What if the states are explored one at a time?

Best result in online graph algorithms: **O(sqrt m) per edge**

[Bender, Fineman, Gilbert, Tarjan 2015]

Our main result: O(log m) per edge

Talk Outline

1. Motivation
2. Guided Incremental Digraphs (GID)
    - Broadly applicable data structure for dead state detection
3. Algorithms
4. Evaluation
    - **110-530x speedup** over [BFGT 2015]

[PLDI 2021]

Motivation in Z3

2020 internship

Microsoft Research

Boolean regex constraints:
s matches R1 and R2 and …
And does not match R4, R5, …

[PLDI 2021]

Motivation in Z3

Z3

2020 internship

Microsoft Research

Boolean regex constraints:
s matches R1 and R2 and …
And does not match R4, R5, …

aws Zelkova

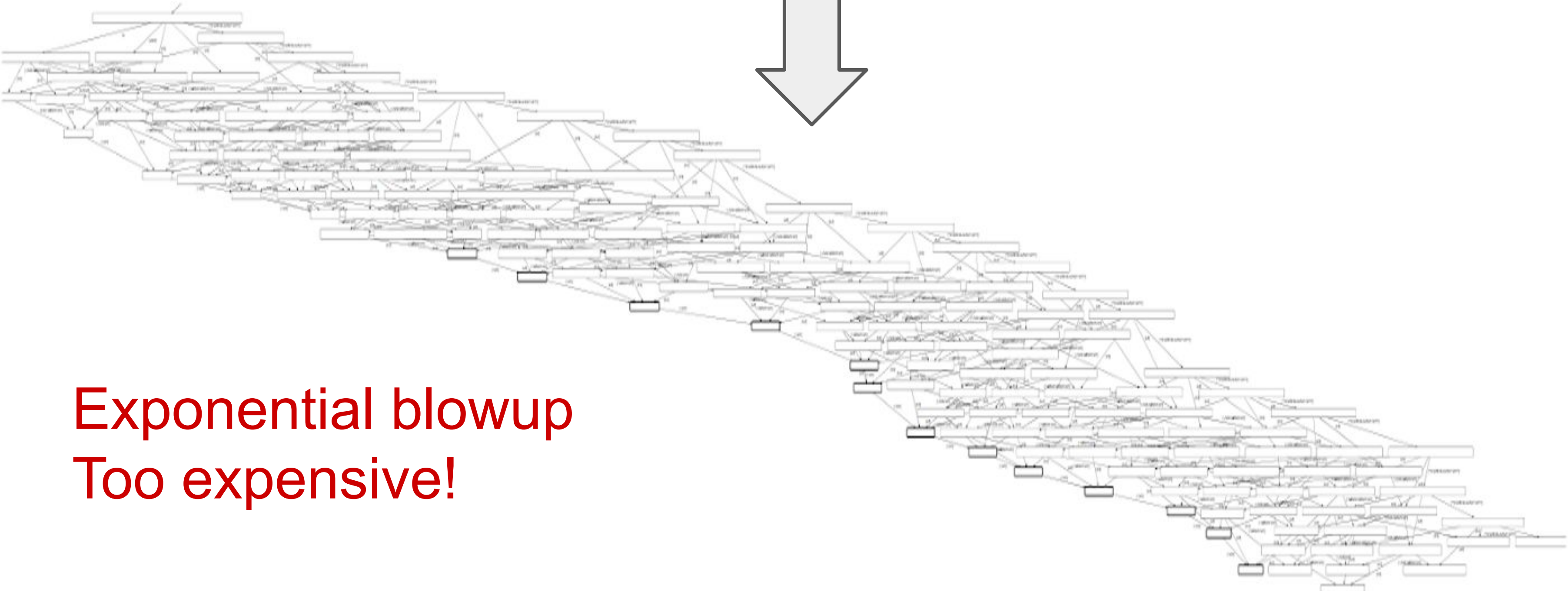
"The total number of invocations of Zelkova ranges from a few million to tens of millions in a single day"

⭐ + this year at CAV ⭐

[PLDI 2021]

Motivation in Z3

s matches R1 and R2 and …
And does not match R4, R5, …

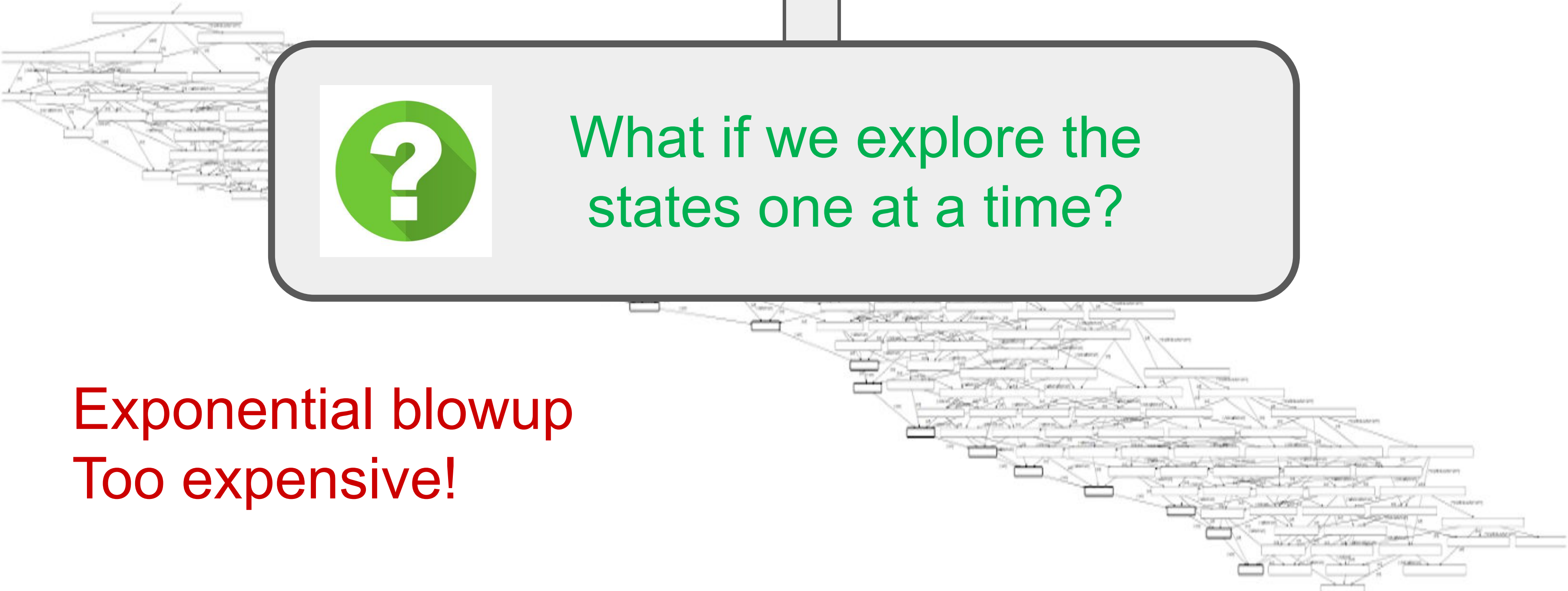


Exponential blowup
Too expensive!

[PLDI 2021]

Motivation in Z3

s matches R1 and R2 and …
And does not match R4, R5, …



What if we explore the states one at a time?

Exponential blowup
Too expensive!

Regex Derivatives

Derivatives:

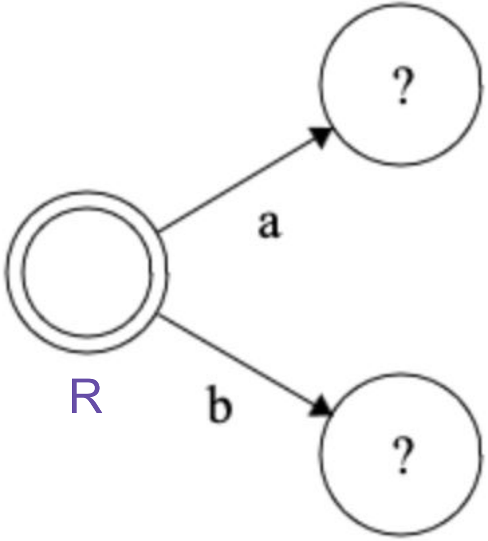
$R = a\Sigma \mid b((\Sigma\Sigma)^* \cap \Sigma(\Sigma\Sigma)^*)$

[PLDI 2021]

Regex Derivatives

Derivatives:

R = aΣ | b((ΣΣ)* ∩ Σ(ΣΣ)*)

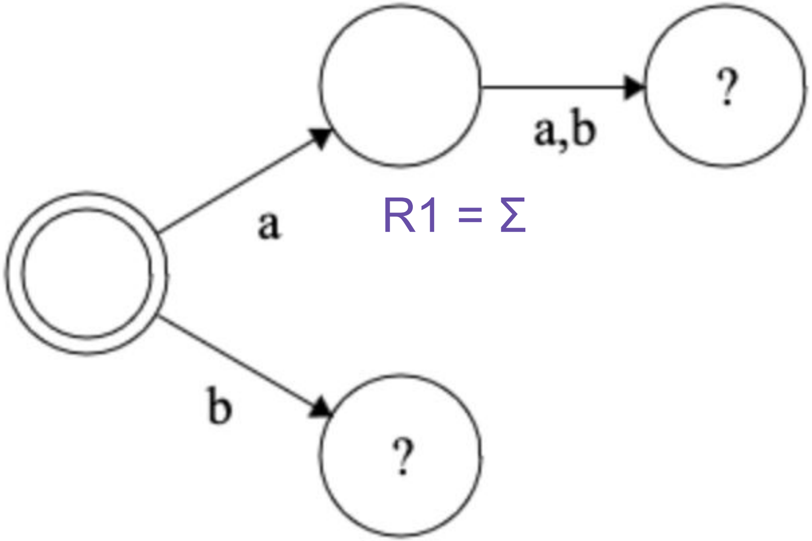


[PLDI 2021]

Regex Derivatives

Derivatives:

$R = a\Sigma \mid b((\Sigma\Sigma)^* \cap \Sigma(\Sigma\Sigma)^*)$



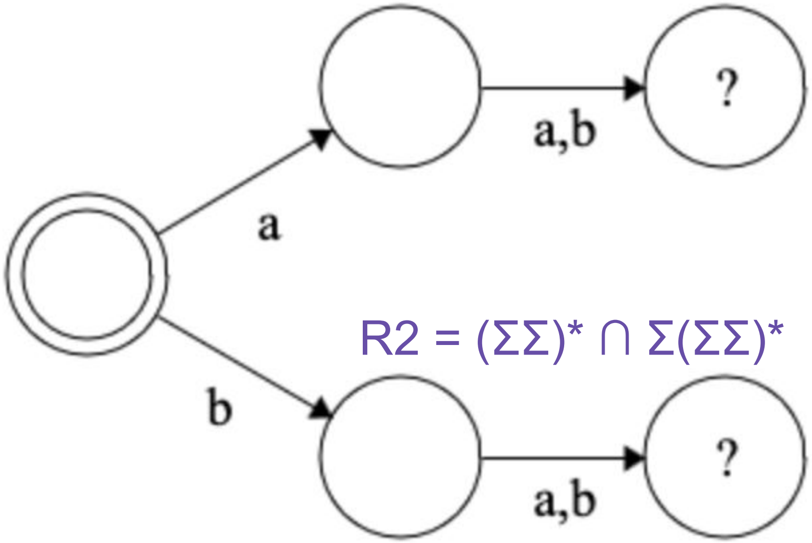
$R1 = \Sigma$

[PLDI 2021]

Regex Derivatives

Derivatives:

$R = a\Sigma \mid b((\Sigma\Sigma)^* \cap \Sigma(\Sigma\Sigma)^*)$



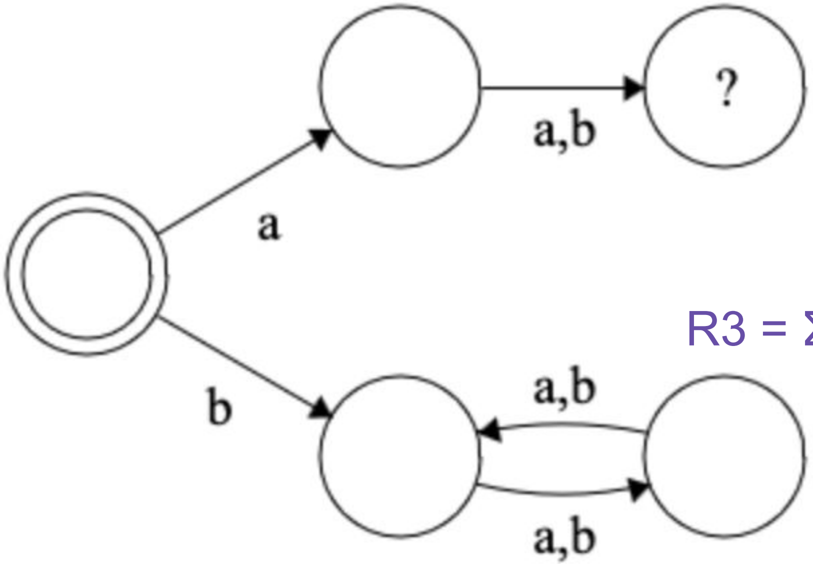
$R2 = (\Sigma\Sigma)^* \cap \Sigma(\Sigma\Sigma)^*$

[PLDI 2021]

Regex Derivatives

Derivatives:

R = aΣ | b((ΣΣ)* ∩ Σ(ΣΣ)*)

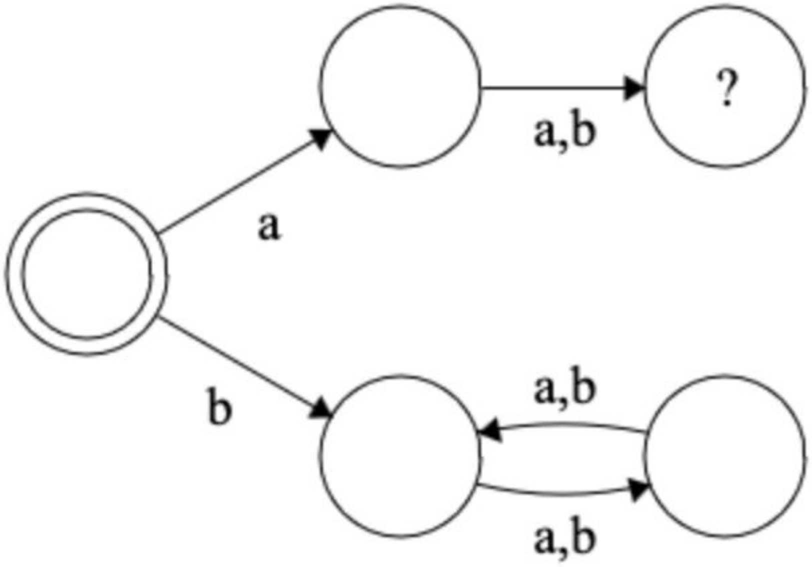


R3 = Σ(ΣΣ)* ∩ (ΣΣ)*

Regex Derivatives

Derivatives:

$R = a\Sigma \mid b((\Sigma\Sigma)^* \cap \Sigma(\Sigma\Sigma)^*)$



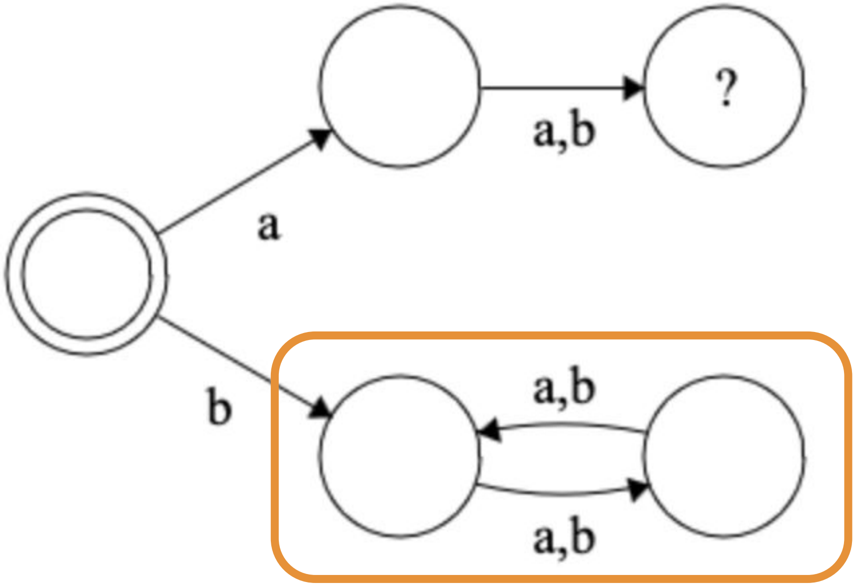
Lazy decision procedure – very fast in practice!

[PLDI 2021]

Regex Derivatives

Derivatives:

R = aΣ | b((ΣΣ)* ∩ Σ(ΣΣ)*)

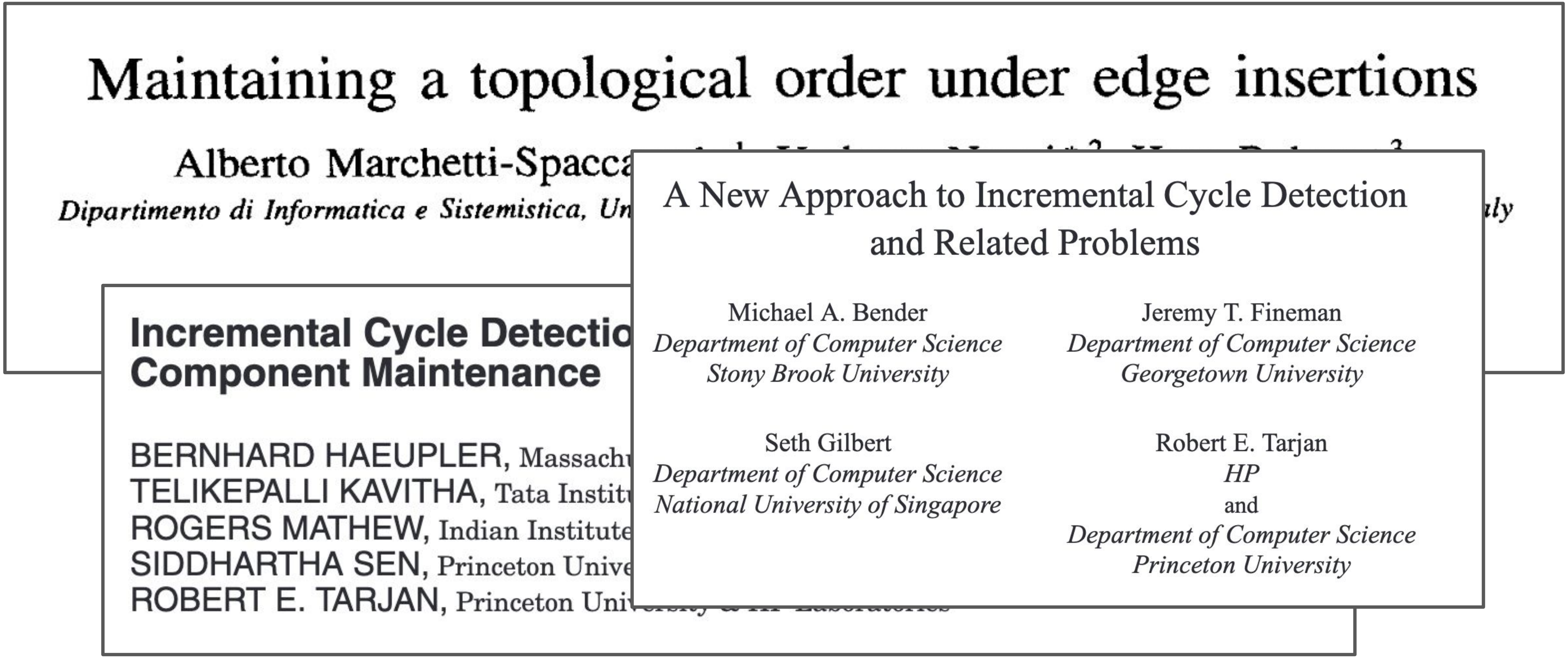


Lazy decision procedure – very fast in practice!

How do we detect dead states? 🤔

Dead State Detection

How do we detect dead states?


Maintaining a topological order under edge insertions

Alberto Marchetti-Spacca

Dipartimento di Informatica e Sistemistica, Un

Incremental Cycle Detectio
Component Maintenance

BERNHARD HAEUPLER, Massachu
TELIKEPALLI KAVITHA, Tata Instit
ROGERS MATHEW, Indian Institute
SIDDHARTHA SEN, Princeton Unive
ROBERT E. TARJAN, Princeton Uni

A New Approach to Incremental Cycle Detection and Related Problems

Michael A. Bender
Department of Computer Science
Stony Brook University

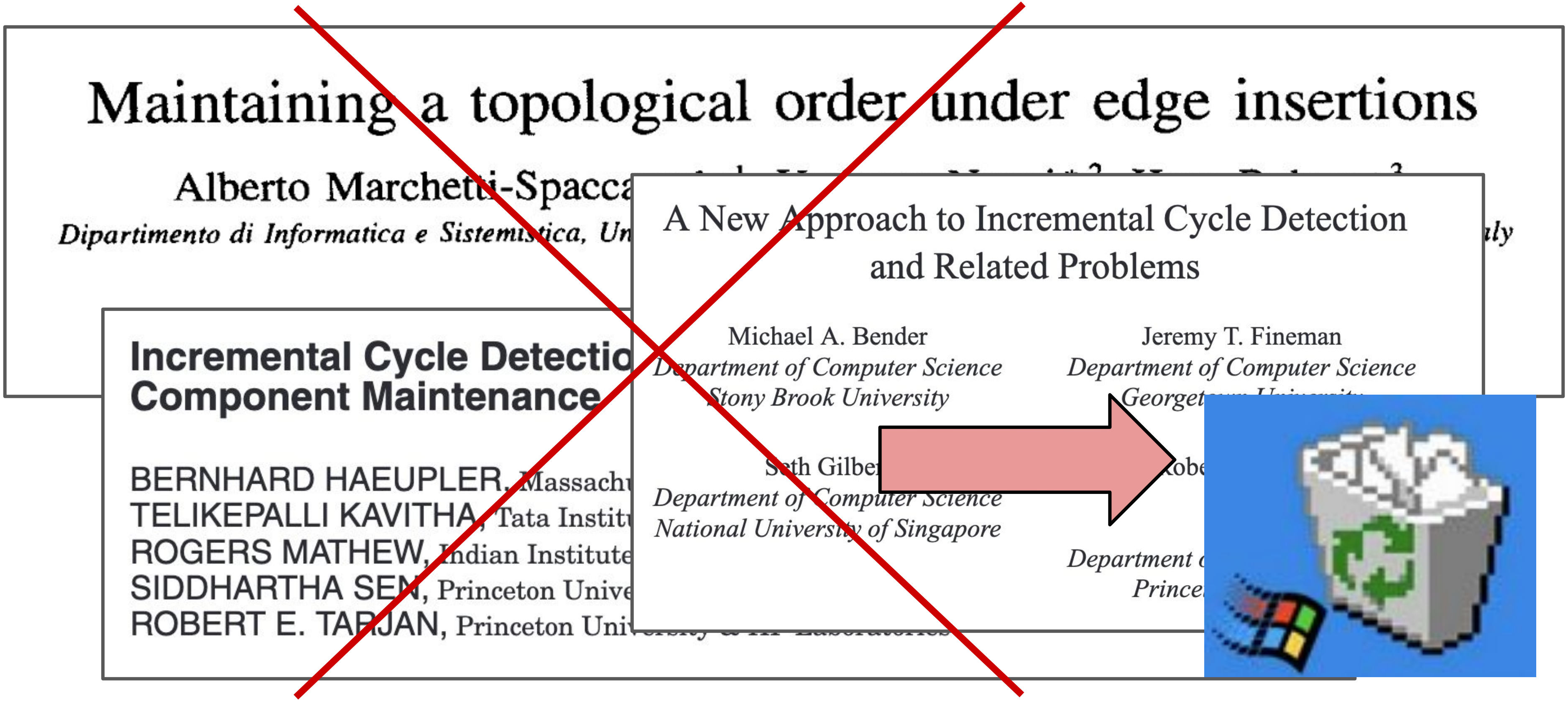
Jeremy T. Fineman
Department of Computer Science
Georgetown University

Seth Gilbert
Department of Computer Science
National University of Singapore

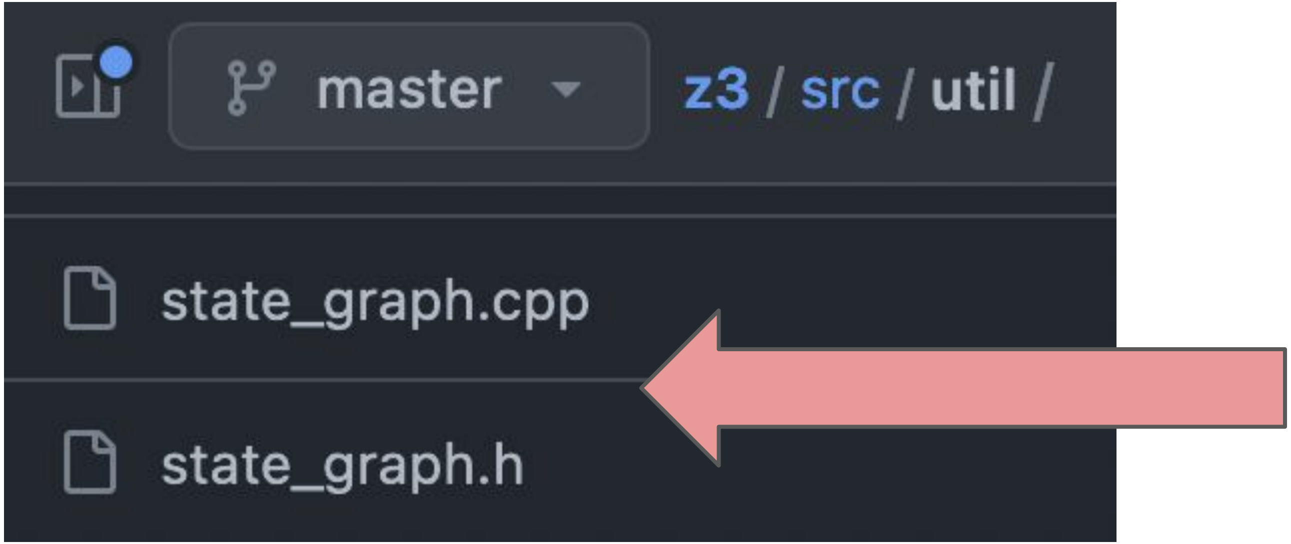
Robert E. Tarjan
HP
and
Department of Computer Science
Princeton University

Dead State Detection

How do we detect dead states? 🤔

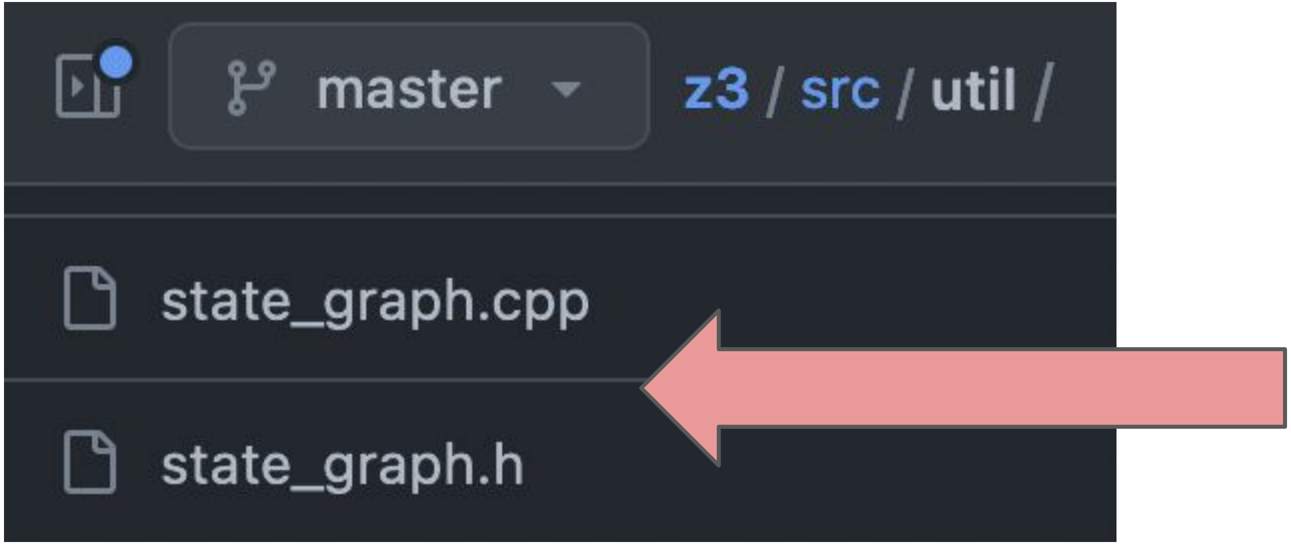


Dead State Detection



Basically a
naive BFS/DFS
O(m) per edge

Dead State Detection



Basically a naive BFS/DFS
O(m) per edge

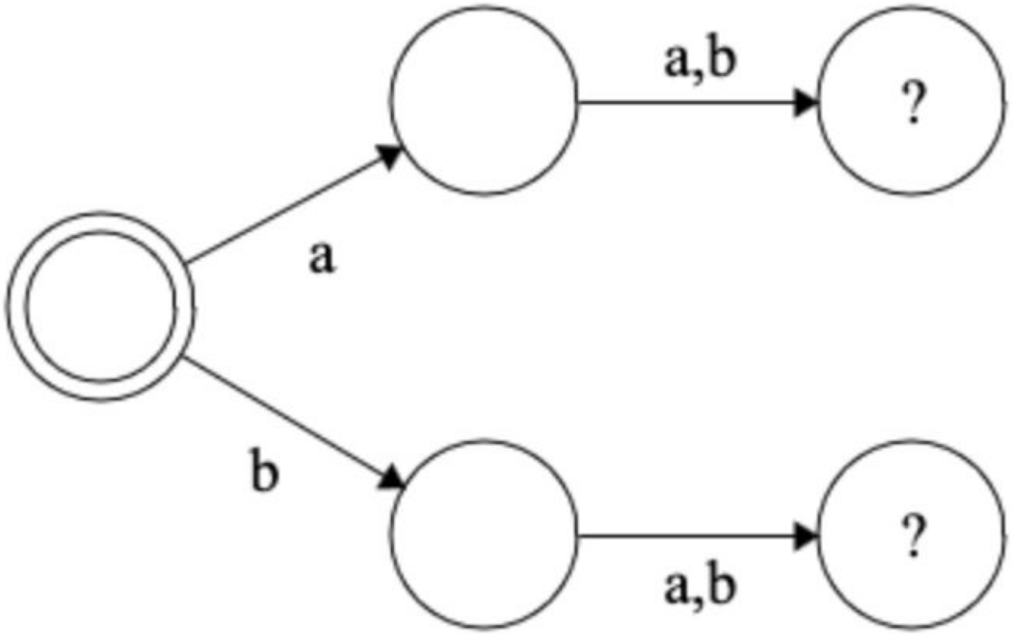
Fast forward 3 years…

Defining the Problem

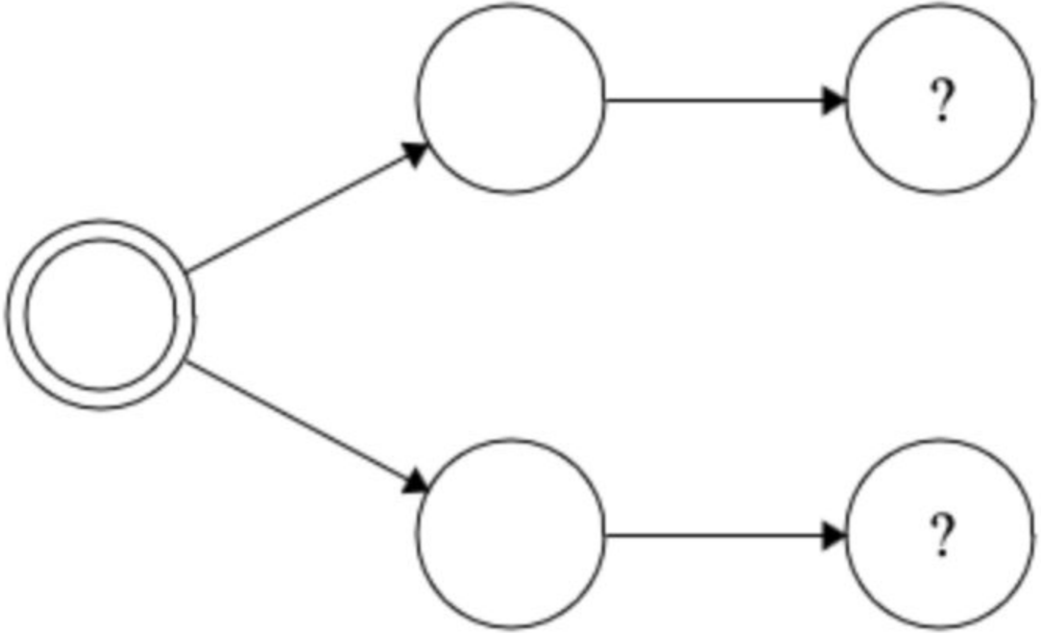
Almost every problem that you come across is befuddled with all kinds of extraneous data of one sort or another; and if you can bring this problem down into the main issues, you can see more clearly what you're trying to do.

—Claude Shannon

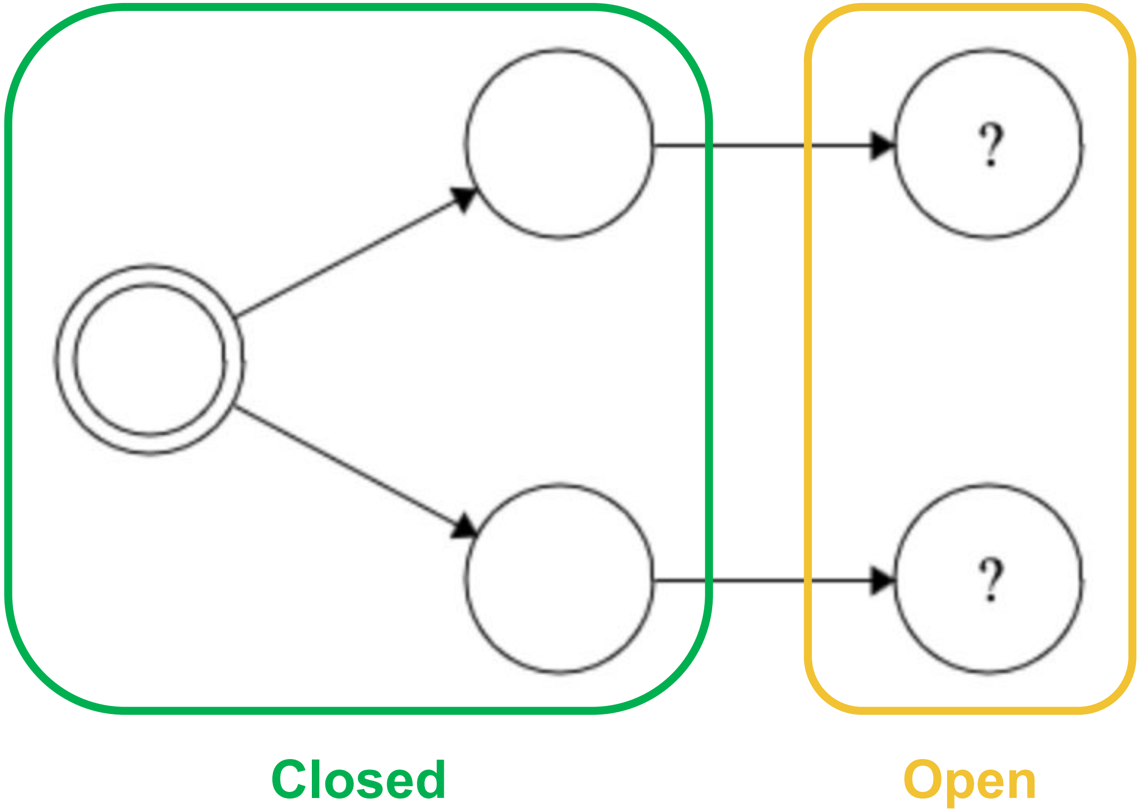
Simplifying…



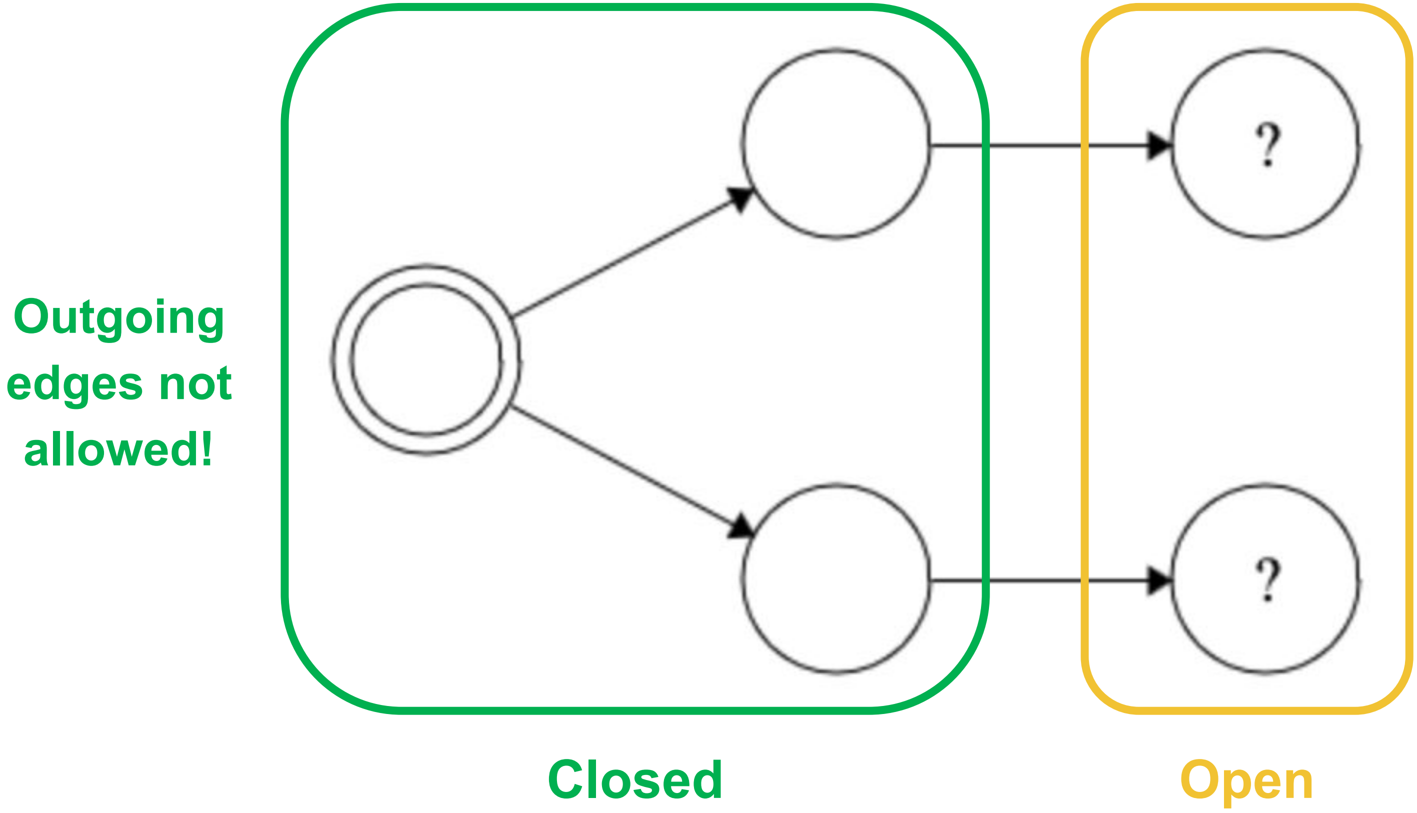
Simplifying…



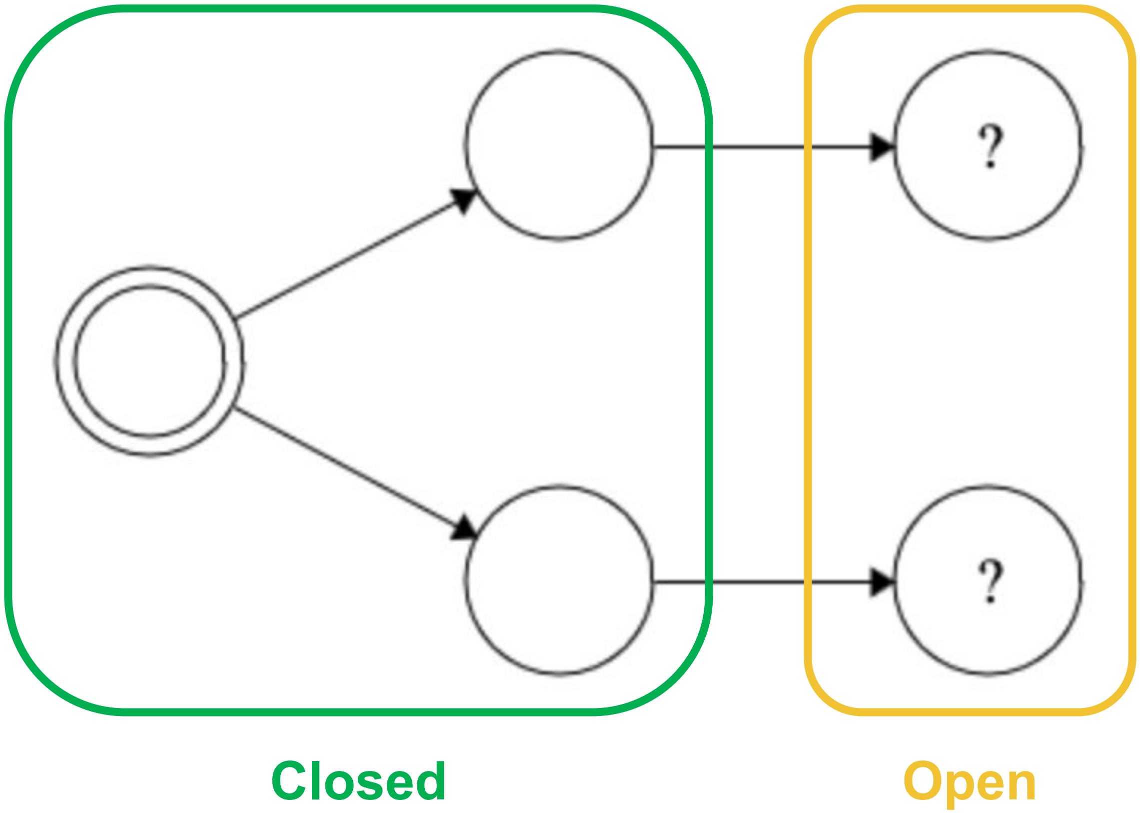
Simplifying…



Guided Incremental Digraph (GID)



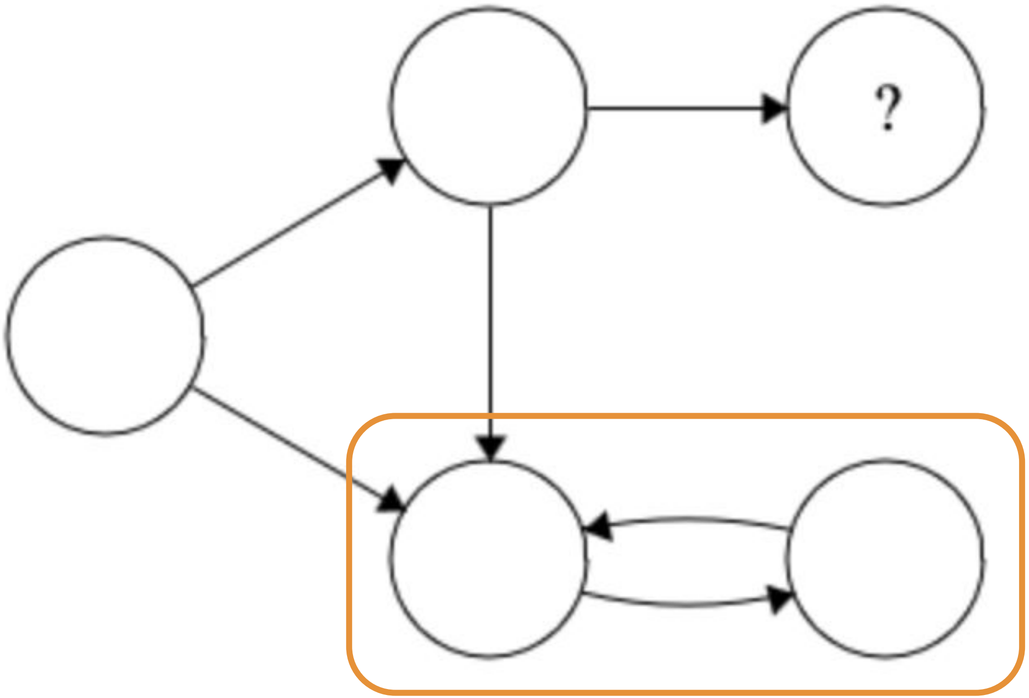
Guided Incremental Digraph (GID)



Live: can reach a terminal state

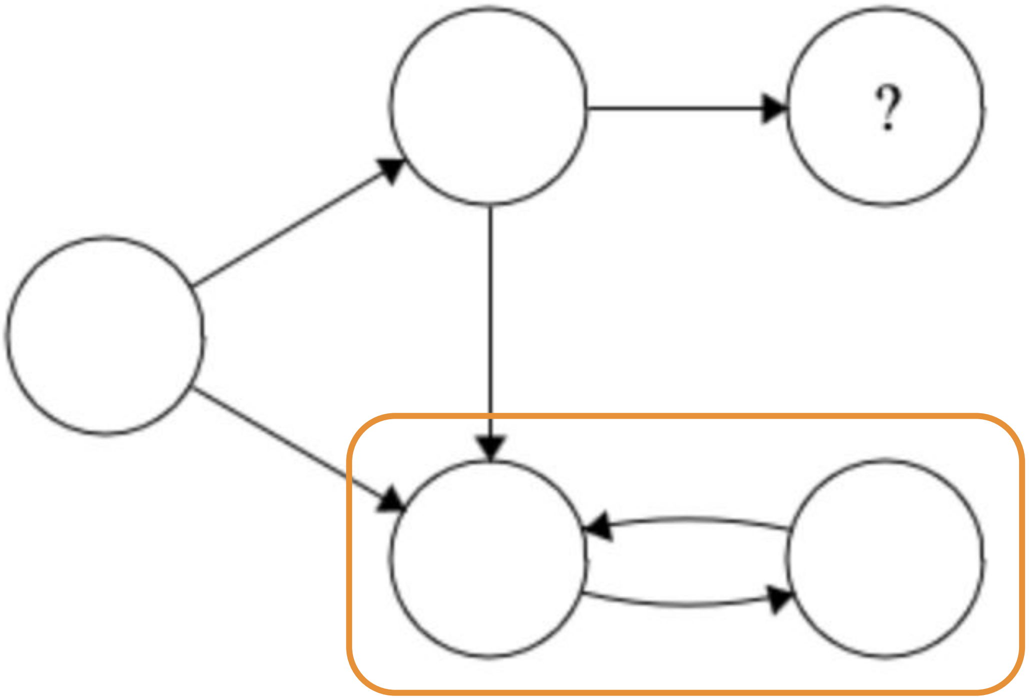
Dead: not live **and all reachable states are closed**

Solving the Problem



Solving the Problem

BFS/DFS: **O(m) per update**

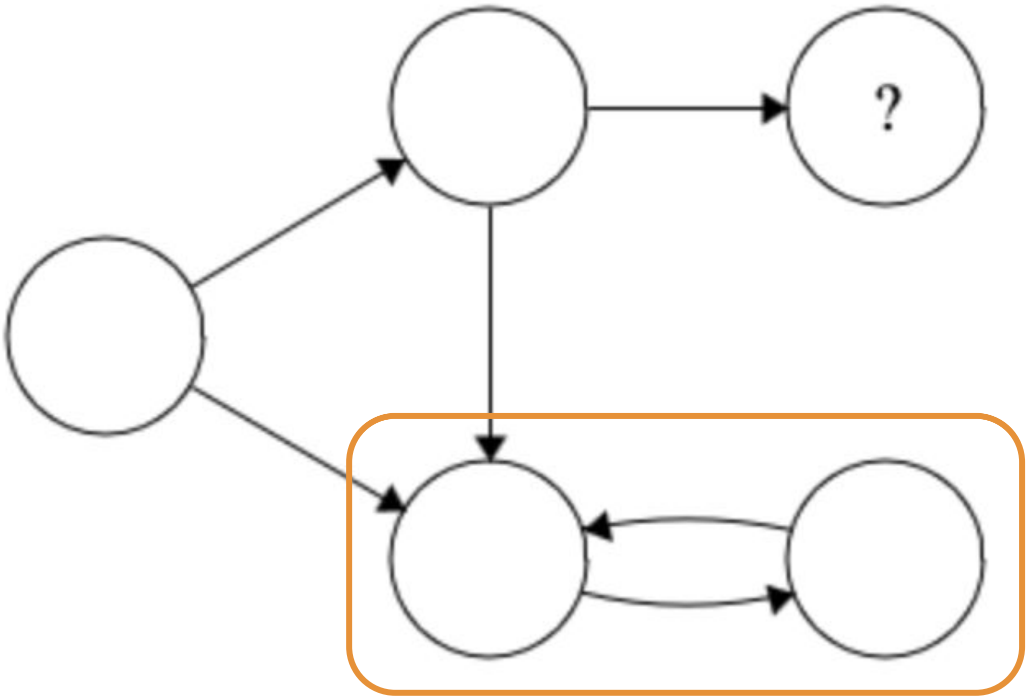


Solving the Problem

BFS/DFS: **O(m) per update**

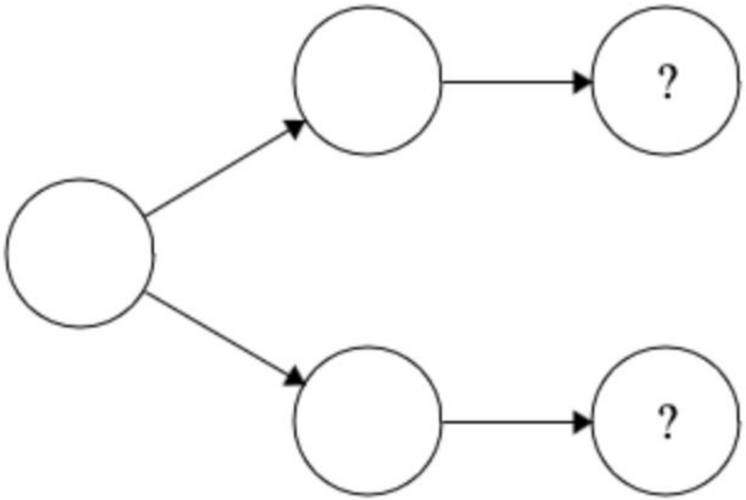
Maintain the graph as a set of SCCs [BFGT2015]

- **O(sqrt(m)) per update**



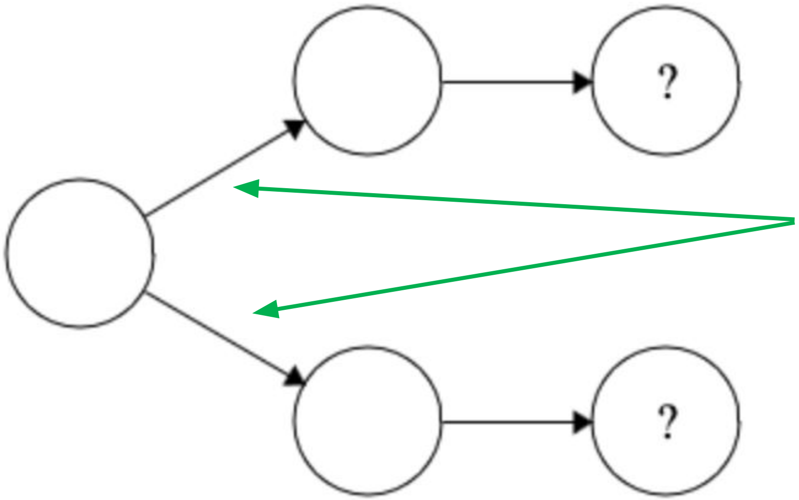
Key insight

What information do we need for non-dead states?



Key insight

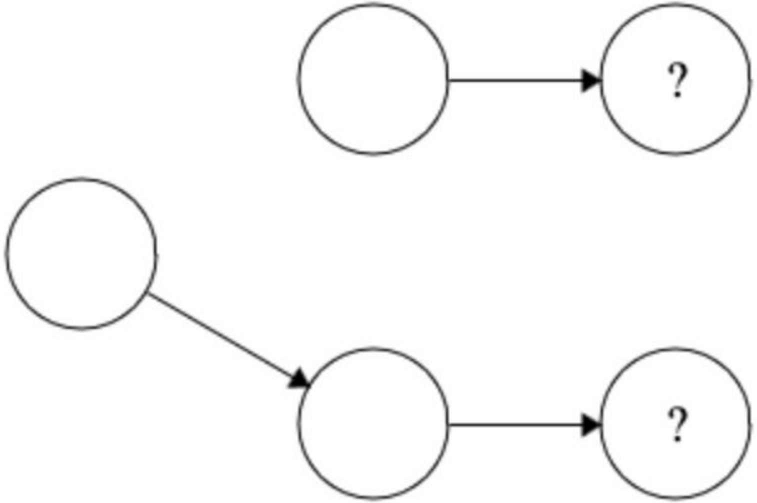
What information do we need for non-dead states?



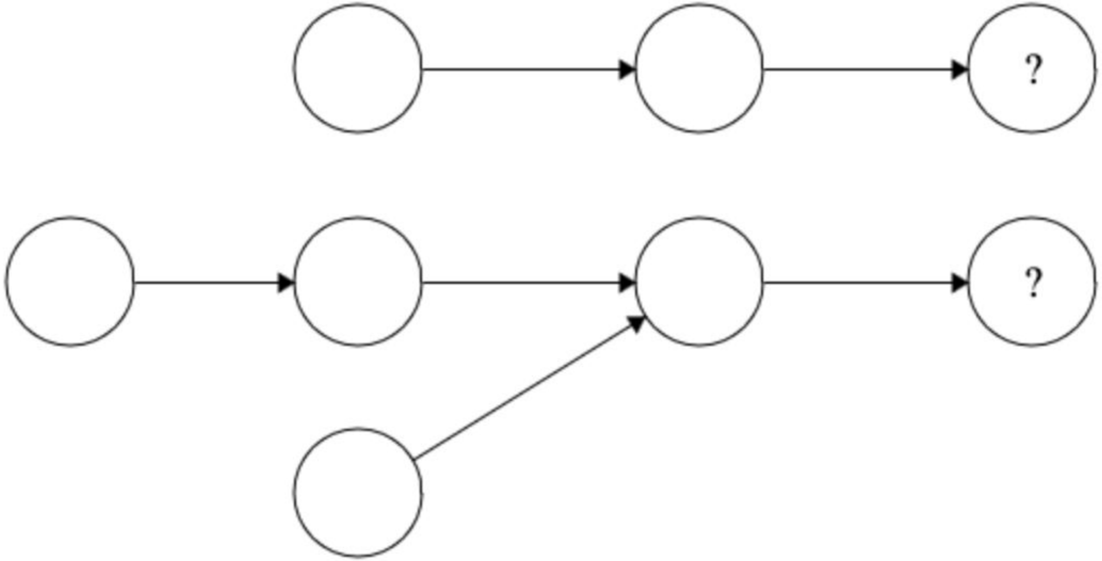
Only need one of these two paths!

Key insight

What information do we need for non-dead states?

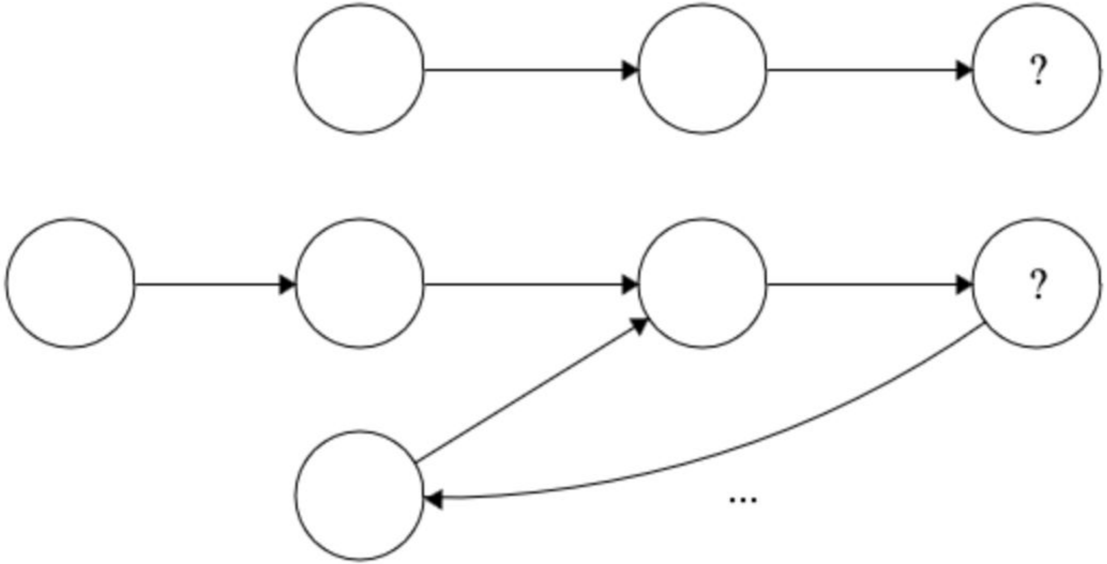


Key insight

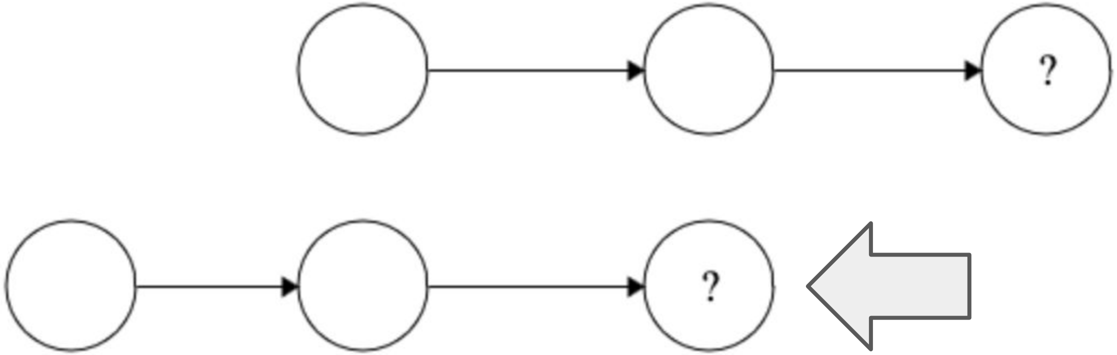


Directed rooted forest

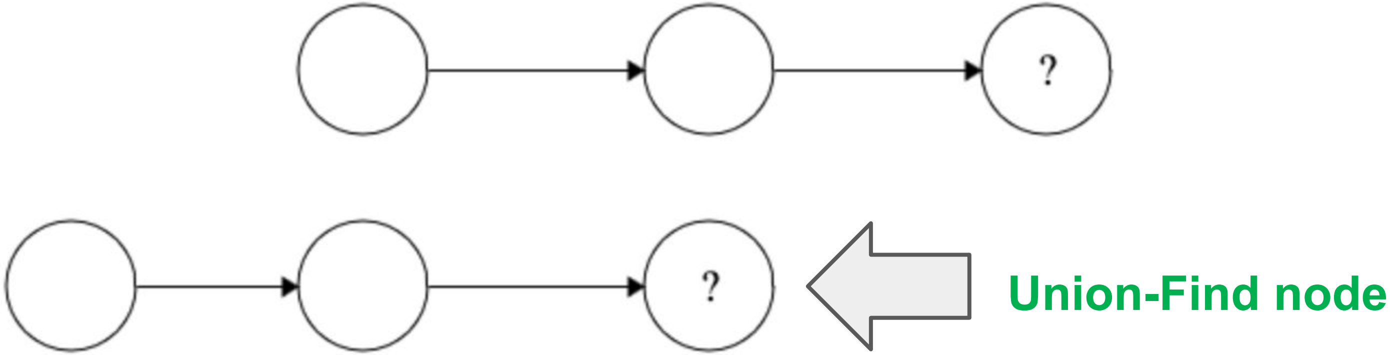
Dealing with cycles?



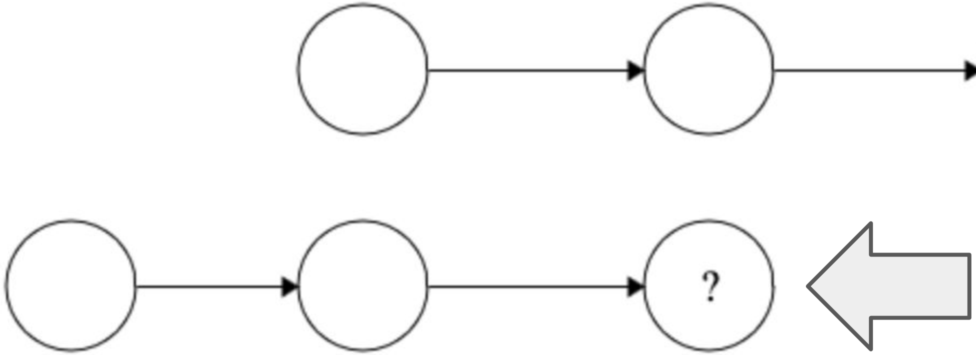
Dealing with cycles?



Dealing with cycles?



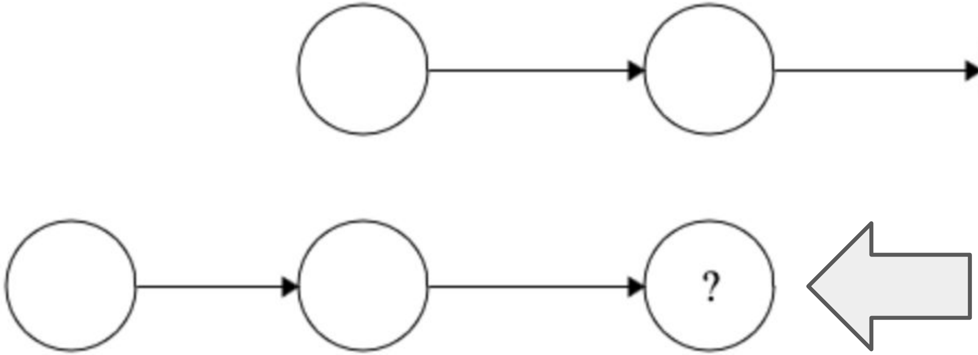
Dealing with cycles?



Union-Find node

O(m) to check for cycle

Dealing with cycles?



In the paper: O(log m) with Euler-Tour Trees

Clever reduction to **undirected reachability** for undirected forests

Are we done?

Asymptotic complexity: **O(log m)** amortized per graph update

Are we done?

Asymptotic complexity: **O(log m)** amortized per graph update

Asymptotic complexity is not enough in practice

- Complex data structures are difficult to implement
- …*and* they impose data structure overheads

Euler Tour Trees	1510
	LoC

Are we done?

Asymptotic complexity: **O(log m)** amortized per graph update

Asymptotic complexity is not enough in practice

- Complex data structures are difficult to implement
- …*and* they impose data structure overheads

Euler Tour Trees	1510
	LoC

Solution: A second, *lazy* algorithm – efficient in practice

Evaluation

Evaluation

How does it perform compared to state-of-the-art online graph algorithms?

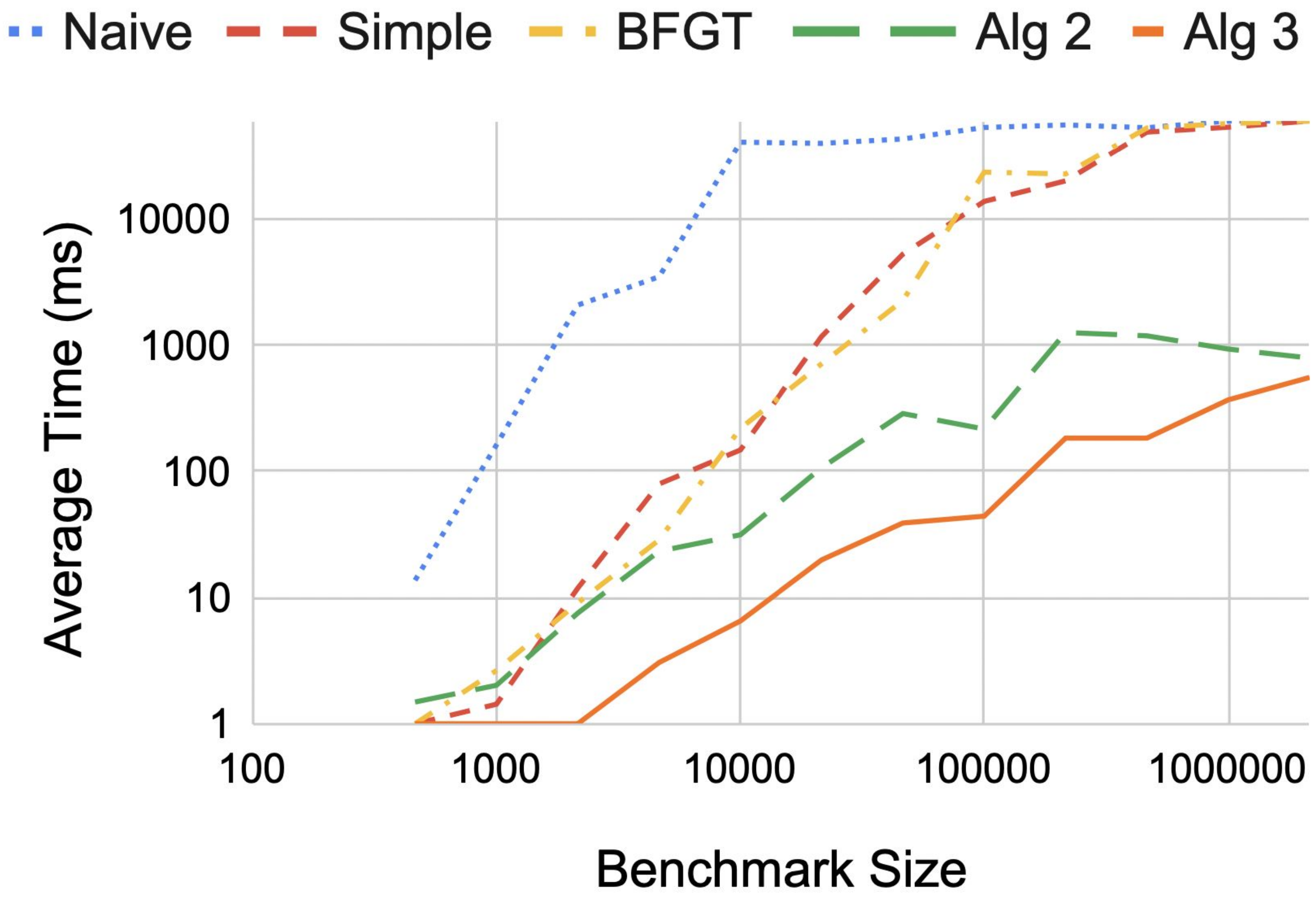
In the paper:

- How does performance change with the graph class?
- How does it perform on graphs from the Z3 regex application?

Evaluation

Green: Log(m) algorithm

Orange: Lazy algorithm



High-level takeaways

- Online graph algorithms are useful in formal methods
- Incremental dead state detection is a natural problem that arises in practical verification tools
- Asymptotic complexity is not always enough

Summary

Guided Incremental Digraphs

- *Closed* states: no more outgoing edges

New algorithms: for dead state detection

- In log(m) time
- + practical improvements

Publicly available on GitHub and crates.io

- https://github.com/cdstanford/gid

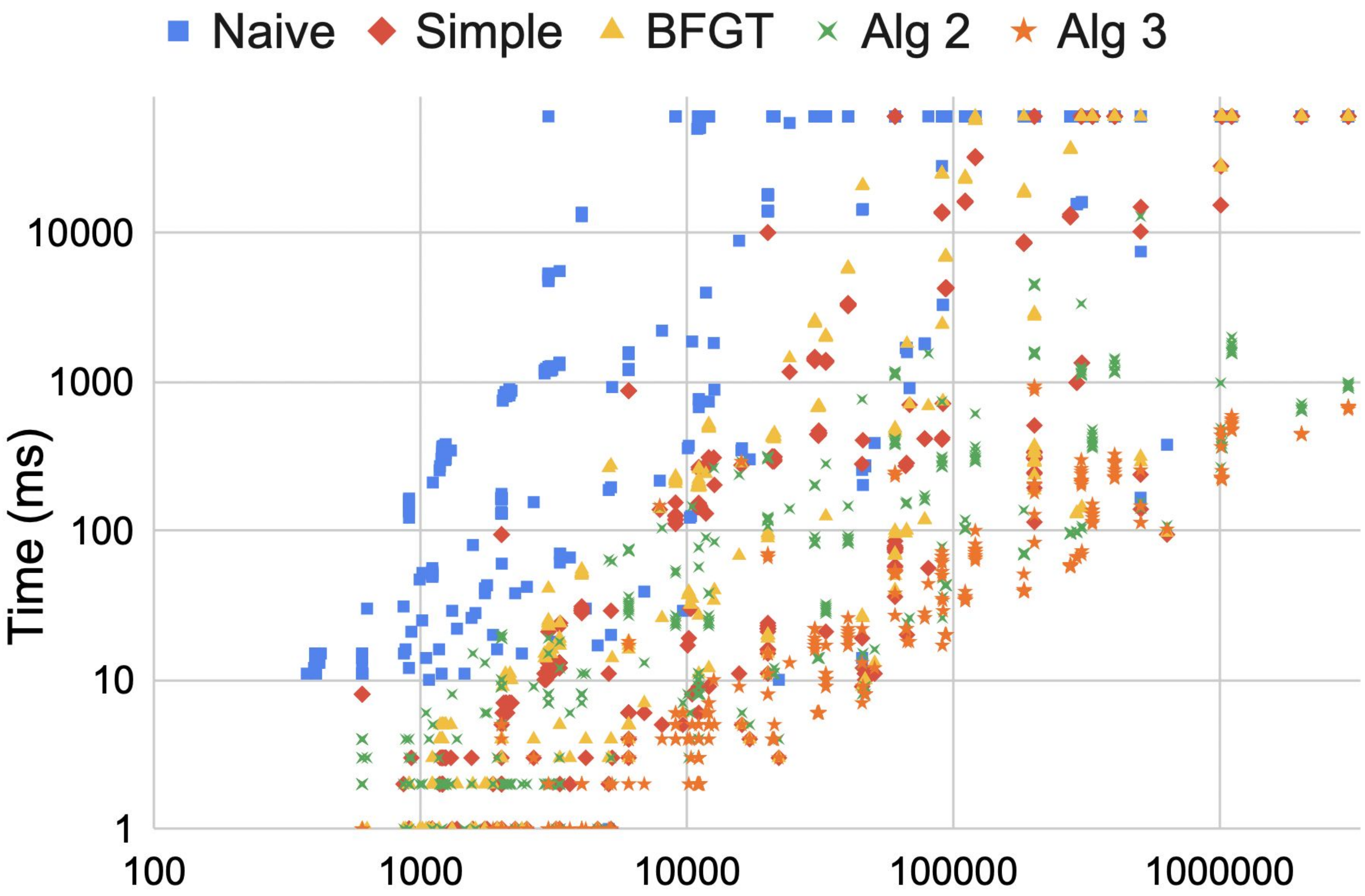
Future Work

Does this generalize to other problems like minimization?

Lazy decision procedures for other contexts

(e.g., LTL and Büchi automata)

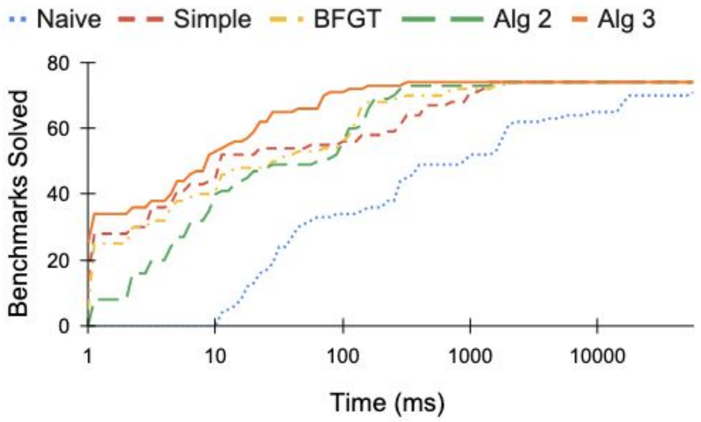
Results



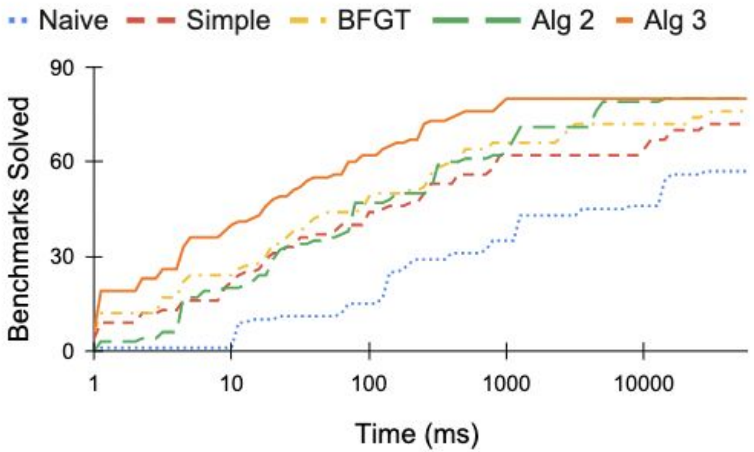
Naive
Simple
BFGT
Alg 2
Alg 3
Time (ms)
10000
1000
100
10
1
100
1000
10000
100000
1000000

Results – Q2 and Q3

Regex



Basic



Random