Learning to Prove Safety over Parameterised Concurrent Systems

Yu-Fang Chen[1] Chih-Duo Hong[2] Anthony W. Lin[2]
Philipp Pümmer[3]

[1]Academia Sinica, Taiwan

[2]University of Oxford, UK

[3]Uppsala University, Sweden

September 11, 2017

Overview

Parameterised concurrent systems

Infinite families $\mathcal{F}$ of finite-state concurrent systems parameterised by the number of processes.

$$\mathcal{F} = \{\text{systems with } n \text{ processes} : n \in \mathbb{N}\}$$

Overview

Parameterised concurrent systems

Infinite families $\mathcal{F}$ of finite-state concurrent systems parameterised by the number of processes.

$$\mathcal{F} = \{\text{systems with } n \text{ processes} : n \in \mathbb{N}\}$$

Checking safety for parameterised systems is undecidable in general.

Overview

Parameterised concurrent systems

Infinite families $\mathcal{F}$ of finite-state concurrent systems parameterised by the number of processes.

$$\mathcal{F} = \{\text{systems with } n \text{ processes} : n \in \mathbb{N}\}$$

Checking safety for parameterised systems is undecidable in general.

In this talk, we will introduce a simple but effective heuristic to verify safety of parameterised systems based on automata learning.

Symbolic Framework

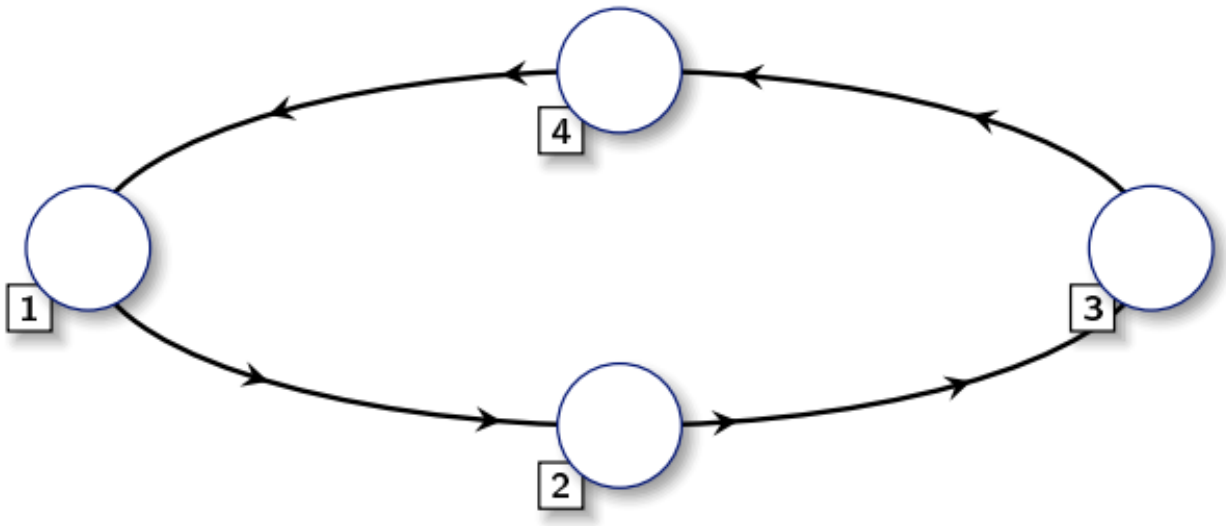
Modelling parameterised systems

Configurations: represented as finite words

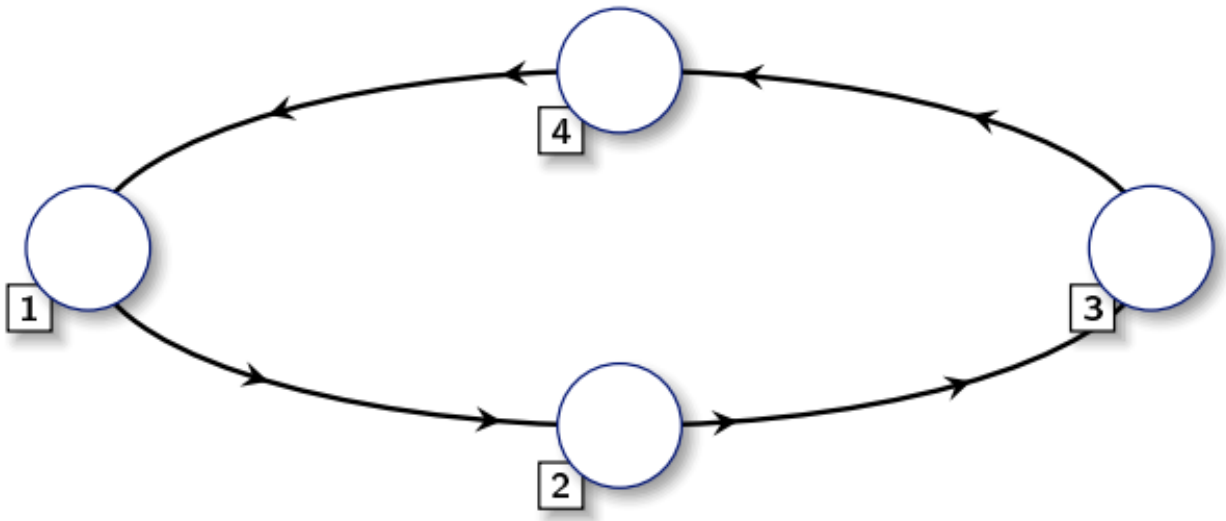
Sets of configurations: represented as finite automata

Transition relation: represented as a transducer

Token Ring Example



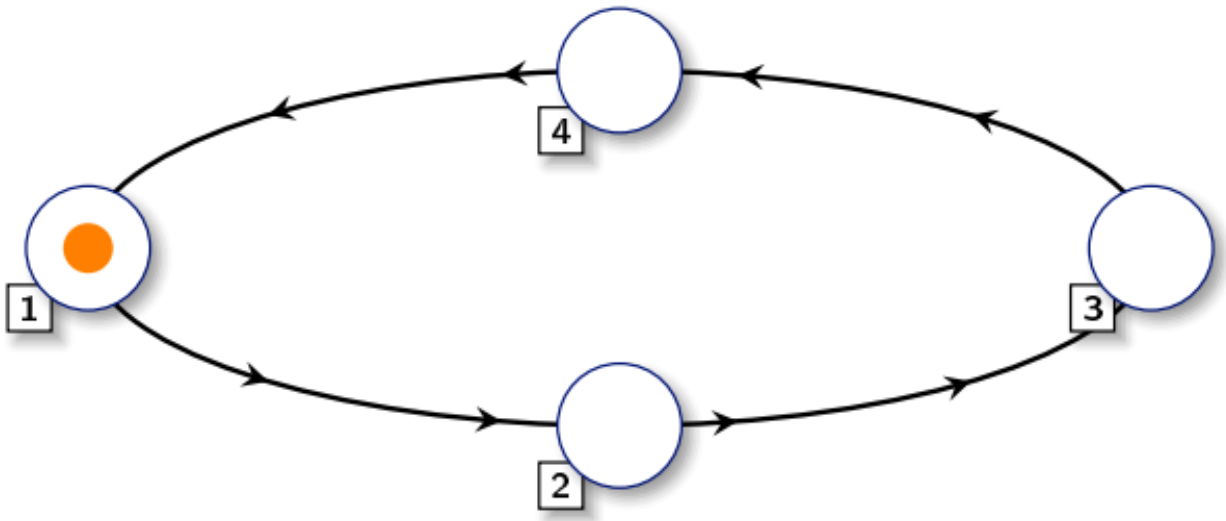
Token Ring Example



Configurations: 1000, 0100, 0010, 0001

Transitions: $\begin{bmatrix}1\\0\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}0\\0\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}0\\0\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix}.$

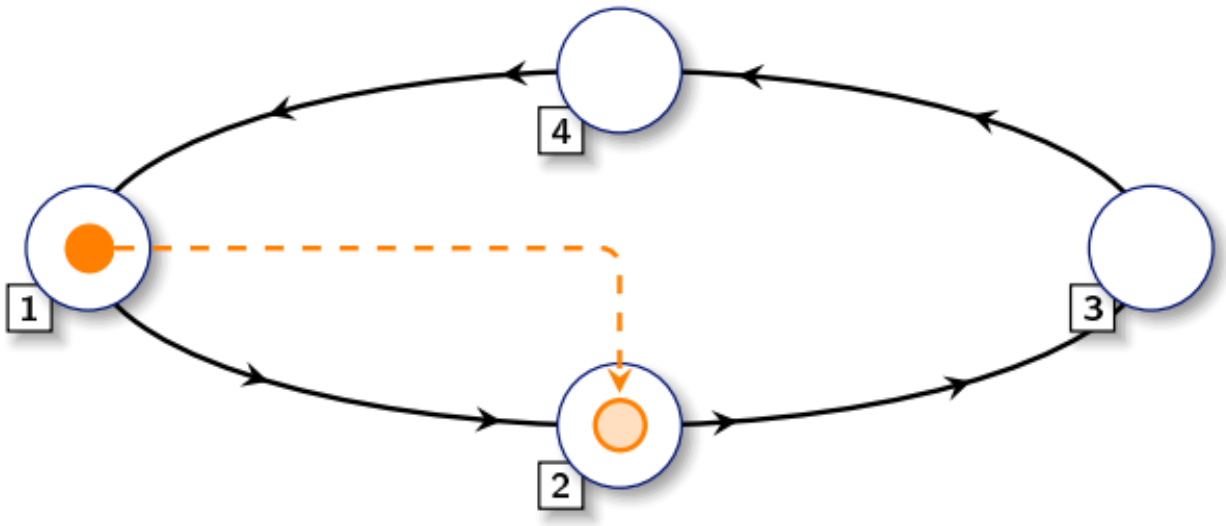
Token Ring Example



Configurations: **1000**, 0100, 0010, 0001

Transitions: $\begin{bmatrix}1\\0\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}0\\0\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}0\\0\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix}.$

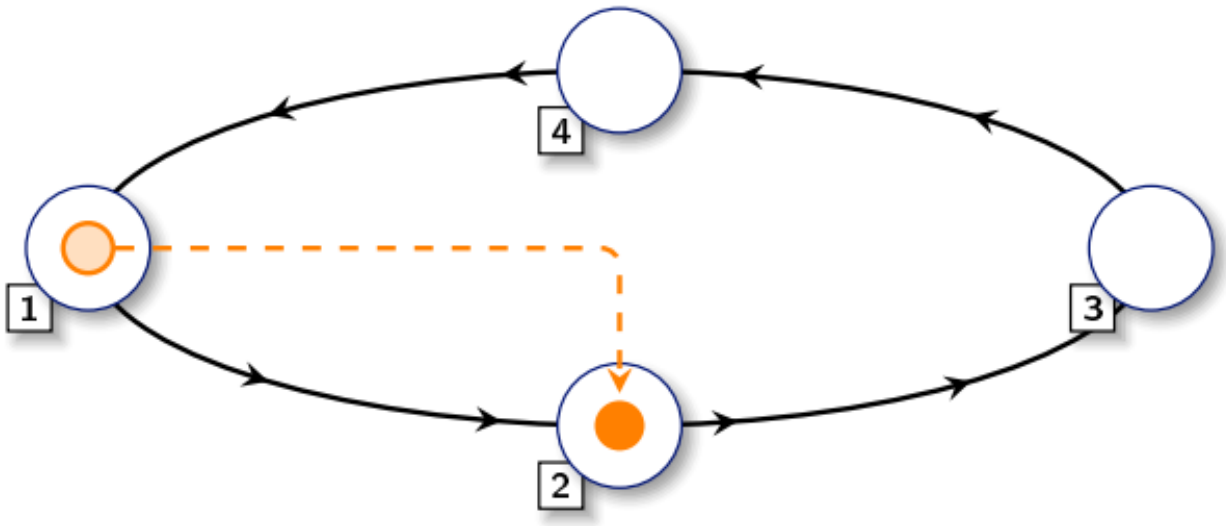
Token Ring Example



Configurations: **1000**, 0100, 0010, 0001

Transitions: $\begin{bmatrix}\mathbf{1}\\0\end{bmatrix}\begin{bmatrix}\mathbf{0}\\1\end{bmatrix}\begin{bmatrix}\mathbf{0}\\0\end{bmatrix}\begin{bmatrix}\mathbf{0}\\0\end{bmatrix}, \begin{bmatrix}0\\0\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}0\\0\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix}.$

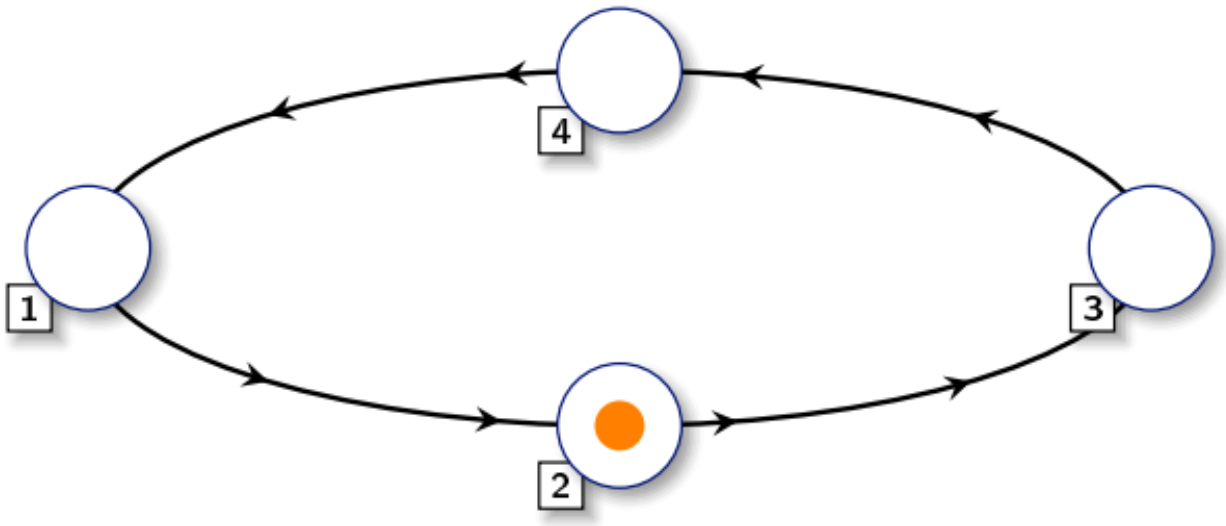
Token Ring Example



Configurations: **1000**, 0100, 0010, 0001

Transitions: $\begin{bmatrix}1\\\mathbf{0}\end{bmatrix}\begin{bmatrix}0\\\mathbf{1}\end{bmatrix}\begin{bmatrix}0\\\mathbf{0}\end{bmatrix}\begin{bmatrix}0\\\mathbf{0}\end{bmatrix}, \begin{bmatrix}0\\0\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}0\\0\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix}.$

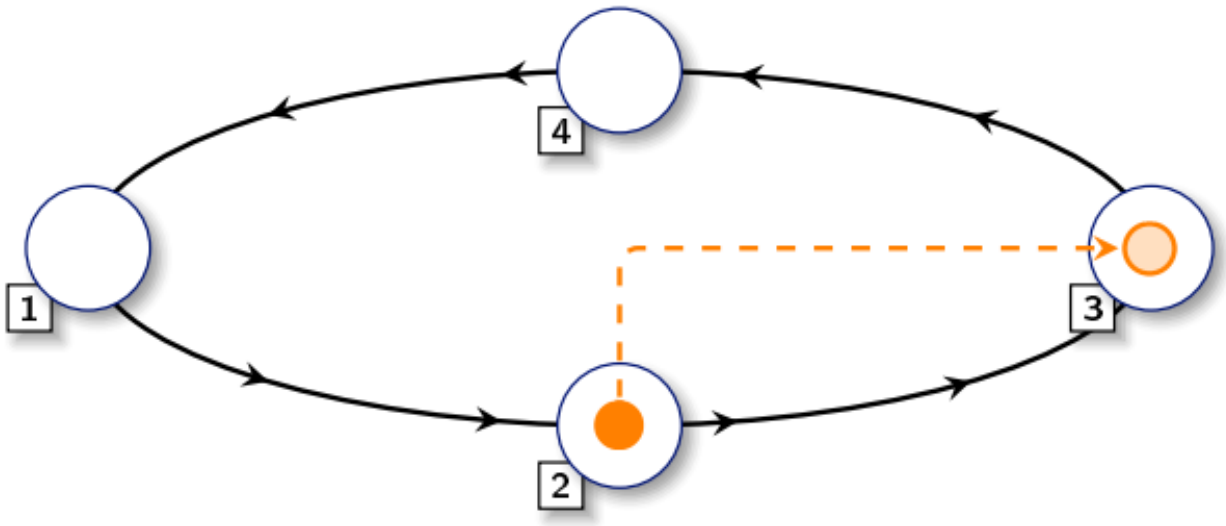
Token Ring Example



Configurations: 1000, **0100**, 0010, 0001

Transitions: $\begin{bmatrix}1\\0\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}0\\0\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}0\\0\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix}.$

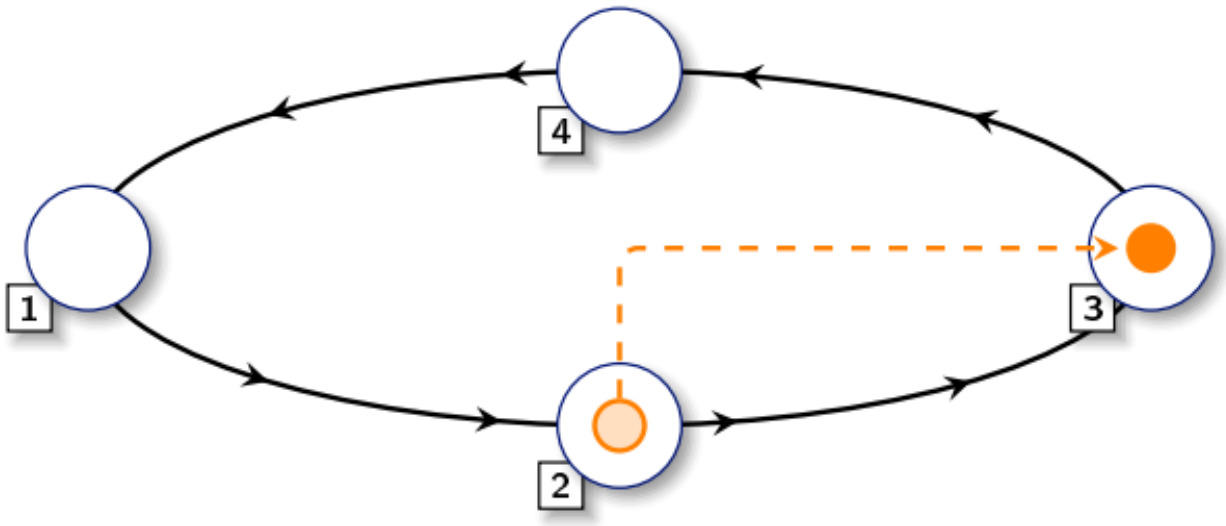
Token Ring Example



Configurations: 1000, **0100**, 0010, 0001

Transitions: $\begin{bmatrix}1\\0\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}\mathbf{0}\\0\end{bmatrix}\begin{bmatrix}\mathbf{1}\\0\end{bmatrix}\begin{bmatrix}\mathbf{0}\\1\end{bmatrix}\begin{bmatrix}\mathbf{0}\\0\end{bmatrix}, \begin{bmatrix}0\\0\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix}.$

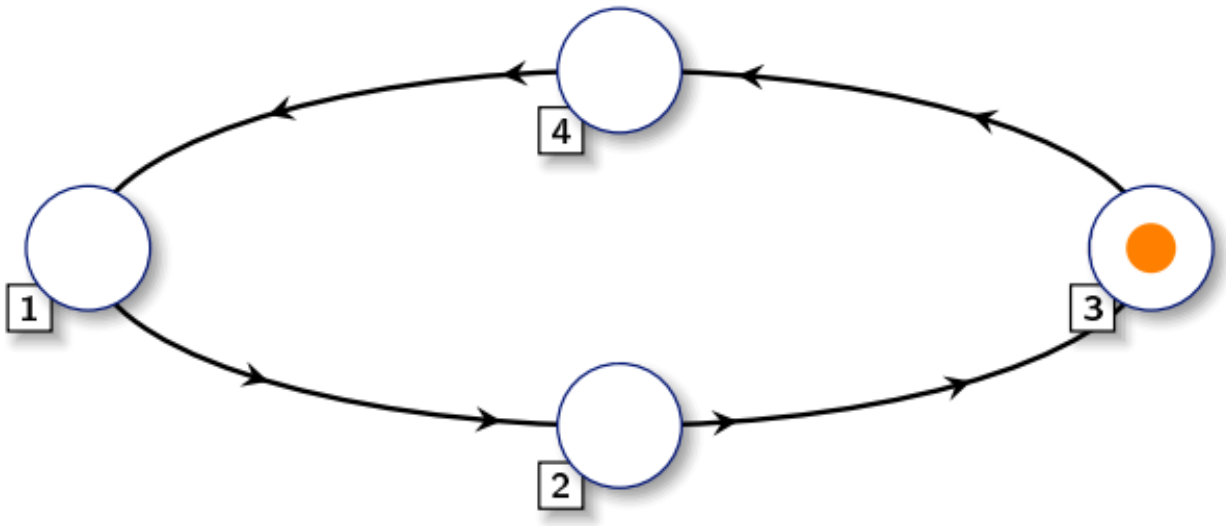
Token Ring Example



Configurations: 1000, **0100**, 0010, 0001

Transitions: $\begin{bmatrix}1\\0\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}0\\\mathbf{0}\end{bmatrix}\begin{bmatrix}1\\\mathbf{0}\end{bmatrix}\begin{bmatrix}0\\\mathbf{1}\end{bmatrix}\begin{bmatrix}0\\\mathbf{0}\end{bmatrix}, \begin{bmatrix}0\\0\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix}.$

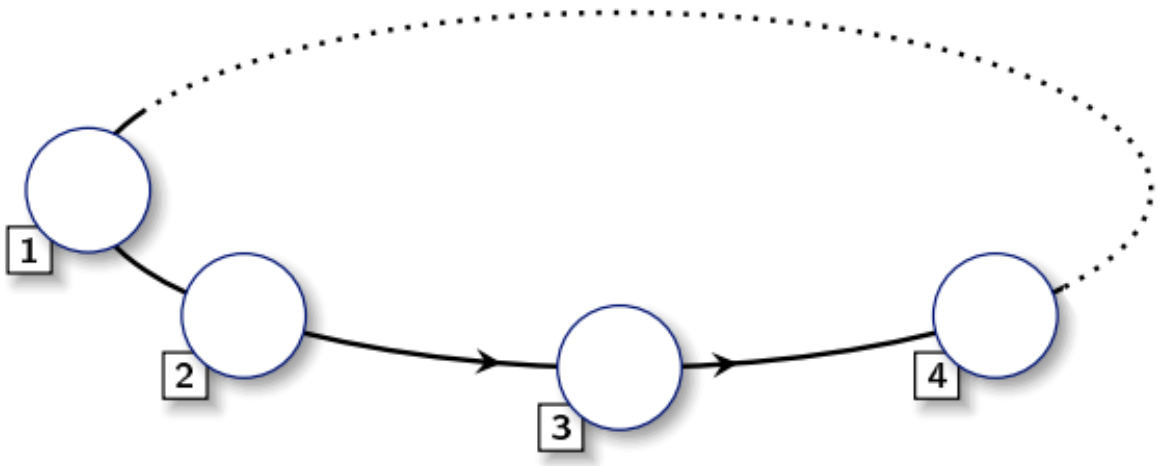
Token Ring Example



Configurations: 1000, 0100, **0010**, 0001

Transitions: $\begin{bmatrix}1\\0\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}0\\0\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}0\\0\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix}.$

Token Ring Example



Configurations: $(0+1)^*$

Transitions: $\begin{bmatrix}0\\0\end{bmatrix}^*\begin{bmatrix}1\\0\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix}^* + \begin{bmatrix}0\\1\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix}^*\begin{bmatrix}1\\0\end{bmatrix}$

Regular Model Checking

System model

A parameterised system is modelled by a tuple (I, T), where

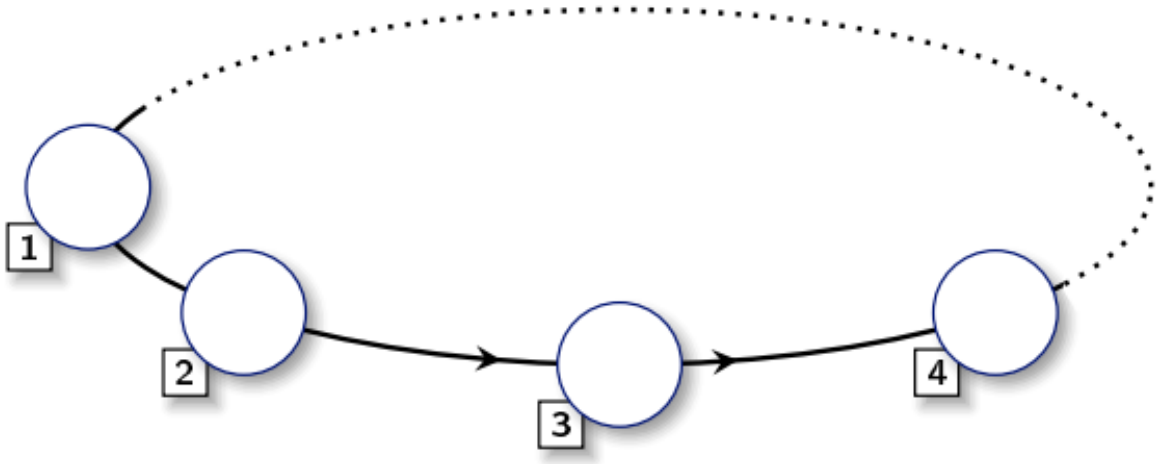
configurations are represented by words in Σ^*

$I \subseteq \Sigma^*$ is represented by a finite automaton

$T \subseteq \Sigma^* \times \Sigma^*$ is represented by a length-preserving transducer

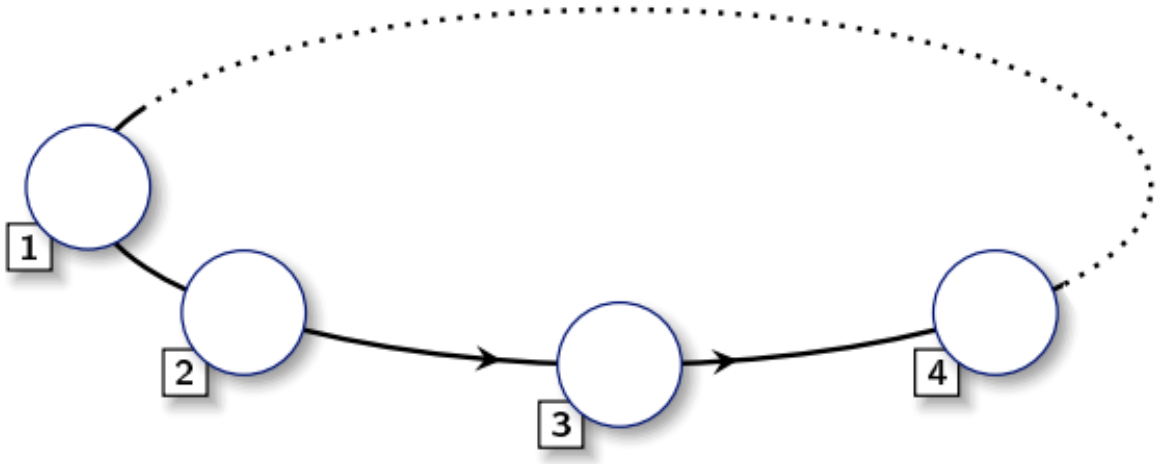
$T^*(I) \equiv \bigcup_{n \geq 0} T^n(I)$ is called the reachability set of (I, T)

Regular Model Checking



I: 0^*10^*, T: $\begin{bmatrix}0\\0\end{bmatrix}^*\begin{bmatrix}1\\0\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix}^* + \begin{bmatrix}0\\1\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix}^*\begin{bmatrix}1\\0\end{bmatrix}$, B: $0^*10^*10^*$

Regular Model Checking



I: 0^*10^*, T: $\begin{bmatrix}0\\0\end{bmatrix}^*\begin{bmatrix}1\\0\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix}^* + \begin{bmatrix}0\\1\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix}^*\begin{bmatrix}1\\0\end{bmatrix}$, B: $0^*10^*10^*$

Safety property: $T^*(I) \cap B = \emptyset$

Regular Model Checking

Safety verification

Given I, T, and B, does $T^*(I) \cap B = \emptyset$ hold?

Regular Model Checking

Safety verification

Given I, T, and B, does $T^*(I) \cap B = \emptyset$ hold?

Proof rules

A regular set A is called a (regular) proof for safety iff

- $I \subseteq A$
- $A \cap B = \emptyset$
- $T(A) \subseteq A$

Regular Model Checking

Safety verification

Given I, T, and B, does $T^*(I) \cap B = \emptyset$ hold?

Proof rules

A regular set A is called a (regular) proof for safety iff

- $I \subseteq A$
- $A \cap B = \emptyset$
- $T(A) \subseteq A$

We exploit these proof rules and the L* learning algorithm to synthesise a regular proof.

Learning Automata via Queries

L* learning algorithm

Proposed by Dana Angluin in 1987 to infer regular sets via querying.

To infer a regular set $\mathcal{R}$, L* makes two types of queries to an oracle:

- Membership query for a word w: Is w in $\mathcal{R}$?
- Equivalence query for a DFA $\mathcal{A}$: Is $L(\mathcal{A}) = \mathcal{R}$?

  If the answer is NO, L* will ask for a word $w \in L(\mathcal{A}) \ominus \mathcal{R}$.

Learning Automata via Queries

L* learning algorithm

Proposed by Dana Angluin in 1987 to infer regular sets via querying.

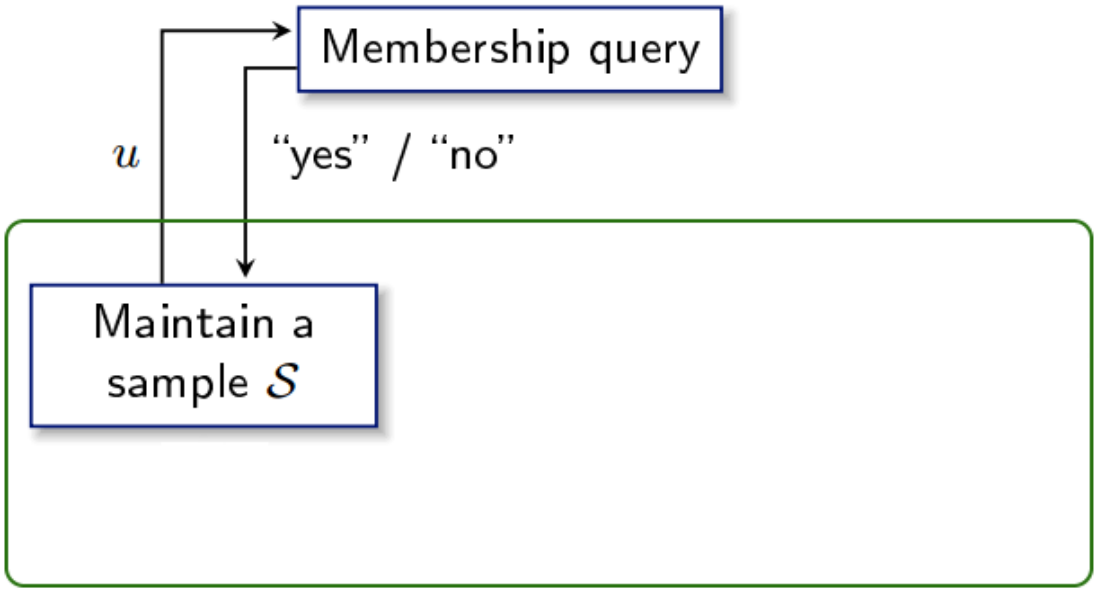
To infer a regular set $\mathcal{R}$, L* makes two types of queries to an oracle:

- Membership query for a word w: Is w in $\mathcal{R}$?
- Equivalence query for a DFA $\mathcal{A}$: Is $L(\mathcal{A}) = \mathcal{R}$?

  If the answer is NO, L* will ask for a word $w \in L(\mathcal{A}) \ominus \mathcal{R}$.

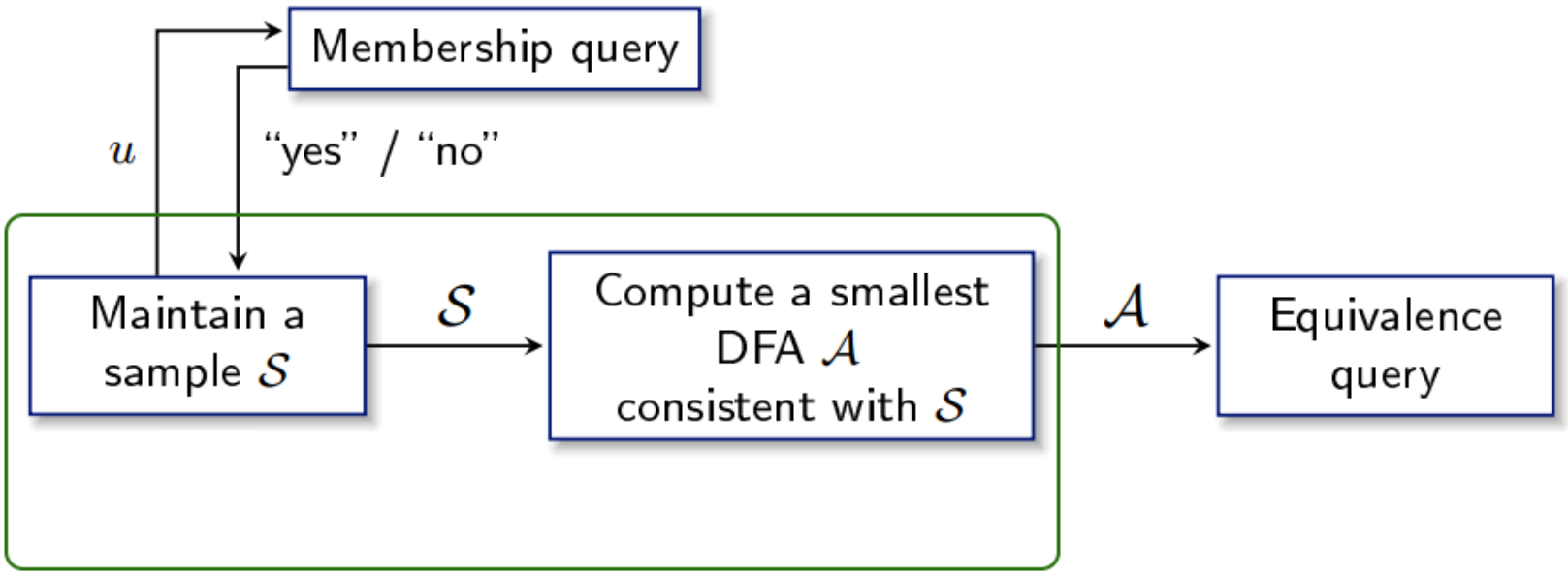
Guaranteed to learn a minimal DFA $\mathcal{A}$ for $\mathcal{R}$ with a polynomial number of queries (in the size of $\mathcal{A}$ and the returned words).

Learning Automata via Queries



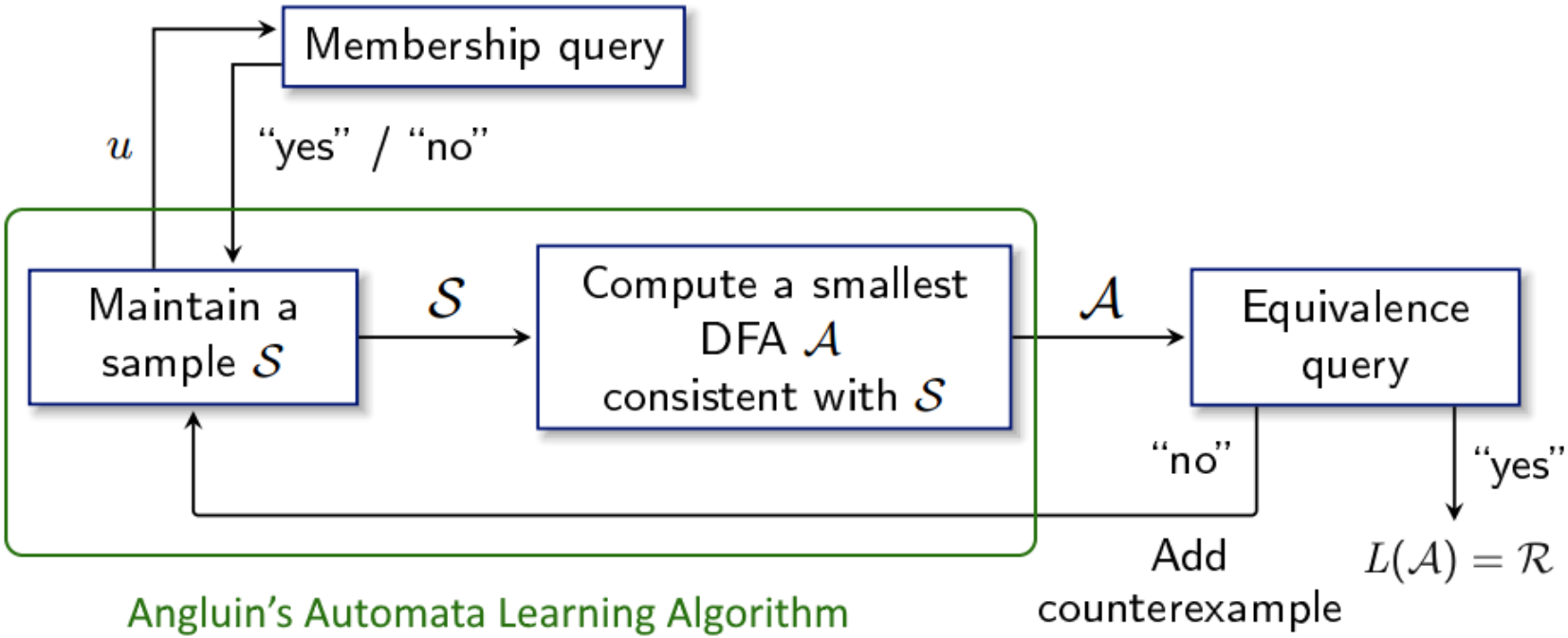
Angluin's Automata Learning Algorithm

Learning Automata via Queries

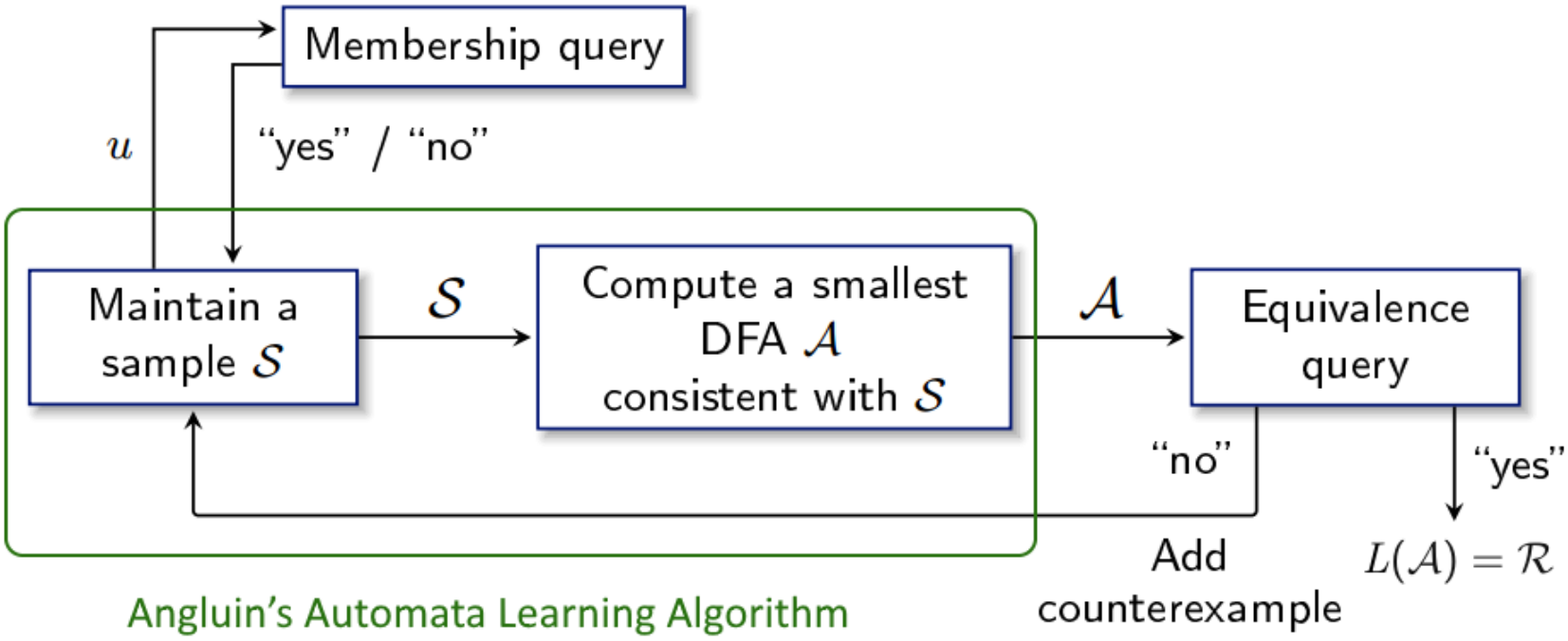


Angluin's Automata Learning Algorithm

Learning Automata via Queries

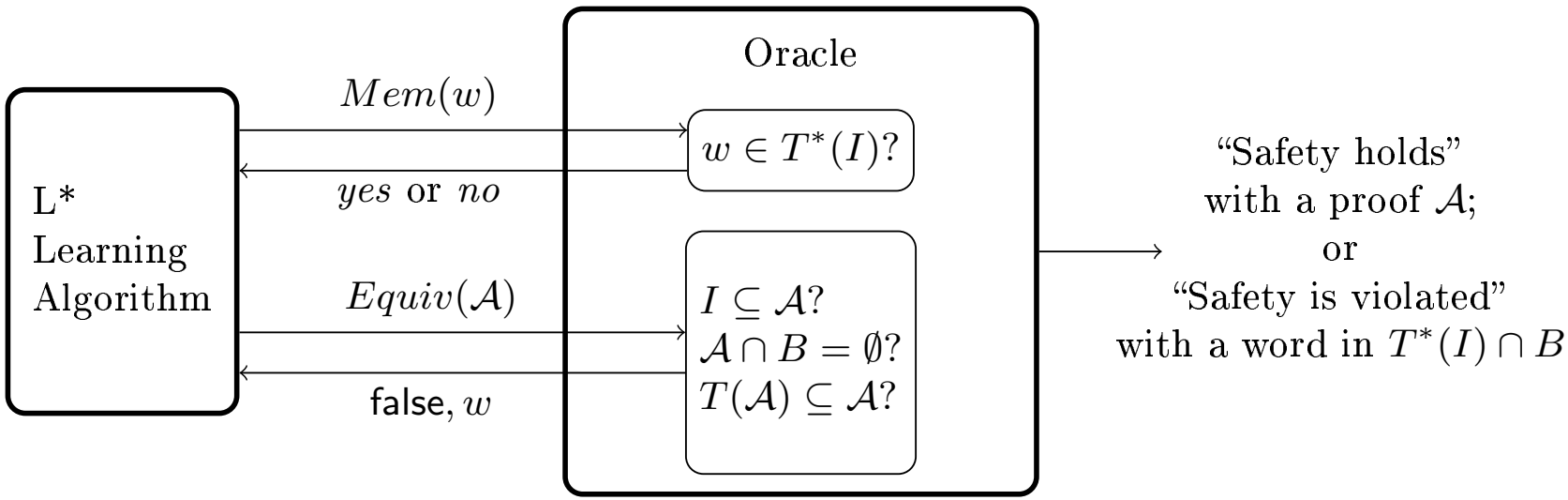


Learning Automata via Queries



We propose an oracle for L* to learn a regular proof for safety.

An overview of Oracle



Comparisons with Other Methods

Methodology	Complete Subclass	Transition Relation
Learning-based synthesis[1]	$T^*(I)$ is regular	l.-p. / rational*
SAT-based refinement[2]	regular proof exists	rational
Widening / Accelerating[3]	unknown	length-preserving
Predicate abs. refinement[4]	unknown	rational

1. Chen et al.'17, Vardhan'04, Habermehl and Vojnar'05
2. Neider and Jansen'13, Lin and Rümmer'16
3. Nilsson'00, Legay'08
4. Bouajjani et al.'06

$\star$ Without termination guarantee [Vardhan, Habermehl and Vojnar]

Comparisons with Other Methods

RMC problems	Learning-based			SAT refinement			Widening			PAR
Name	Time	S_{inv}	T_{inv}	Time	S_{inv}	T_{inv}	Time	S_{inv}	T_{inv}	Time
Bakery	0.0s	6	18	0.5s	2	5	0.0s	6	11	0.0s
Burns	0.2s	8	96	1.1s	2	10	0.1s	7	38	0.0s
Szymanski	0.3s	43	473	1.6s	2	21	2.0s	51	102	0.1s
German	4.8s	14	8134	TO	-	-	TO	-	-	10s
Dijkstra	0.1s	9	378	1.7s	2	24	6.1s	8	83	0.3s
Dijkstra, ring	1.4s	22	264	0.9s	2	14	TO	-	-	0.1s
Dining Crypto.	0.1s	32	448	TO	-	-	TO	-	-	7.2s
Coffee Can	0.0s	3	18	0.2s	2	7	0.1s	6	13	0.0s
Herman, linear	0.0s	2	4	0.2s	2	4	0.0s	2	4	0.0s
Herman, ring	0.0s	2	4	0.4s	2	4	0.0s	2	4	0.0s
Israeli-Jalfon	0.0s	4	8	0.1s	2	4	0.0s	4	8	0.0s
Lehmann-Rabin	0.1s	8	48	0.5s	2	11	0.8s	19	105	0.0s
LR Dining Philo.	0.0s	4	16	0.2s	2	6	0.1s	7	18	0.0s
Mux Array	0.0s	5	30	0.4s	2	7	0.2s	4	14	0.0s
Res. Allocator	0.0s	5	15	0.0s	1	3	0.0s	4	9	0.0s
Kanban	TO	-	-	TO	-	-	TO	-	-	3.5s
Water Jugs	0.1s	24	264	TO	-	-	TO	-	-	0.0s

Timeout: 60 seconds

Summary

Automatic method for proving safety for parameterised systems

Regular model checking as symbolic framework

Automata learning to synthesise "regular" proofs

Simple but effective (50-line Java code based on existing learning and automata libraries)

Full paper can be found at FMCAD'17