
Regular Model Checking Using Solver Technologies and Automata Learning

Daniel Neider Nils Jansen

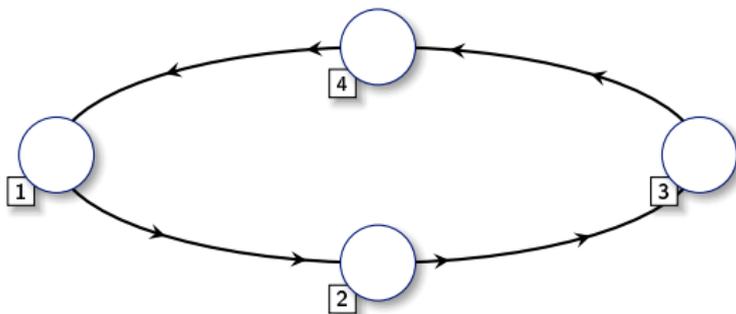
RWTH Aachen University, Germany

May 14th, 2013

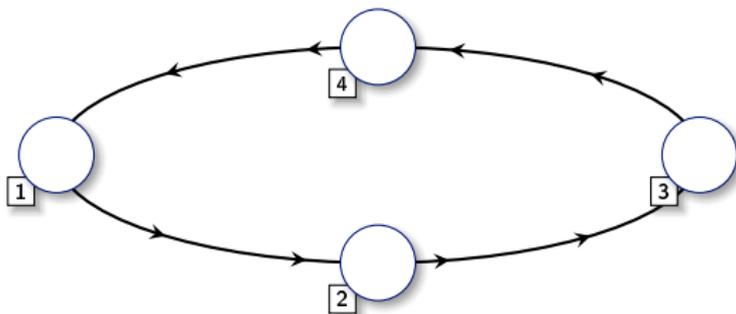
NASA Formal Methods Symposium

NASA Ames Research Center, Moffett Field, CA, USA

Token Ring Example



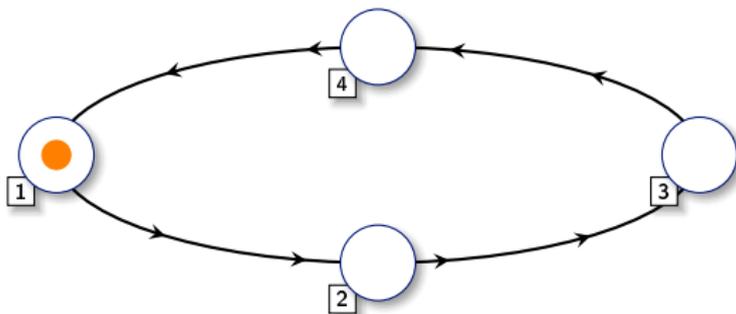
Token Ring Example



Modeling Programs

- Configurations:
- Transitions:

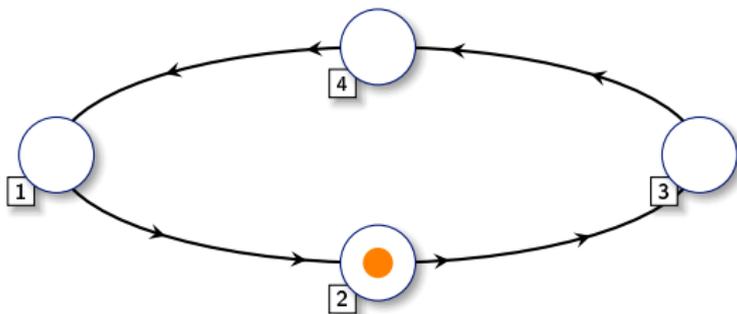
Token Ring Example



Modeling Programs

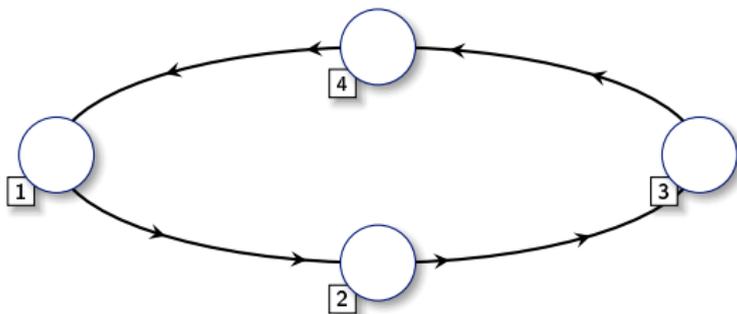
- Configurations: 1000
- Transitions:

Token Ring Example



Modeling Programs

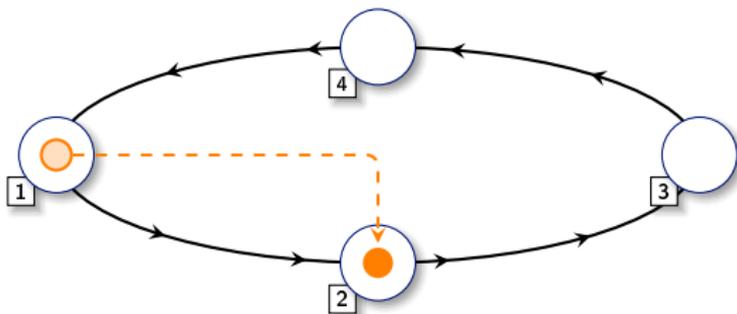
- Configurations: 1000, 0100
- Transitions:



Modeling Programs

- Configurations: 1000, 0100, 0010, 0001, 0000, 1100, ...
- Transitions:

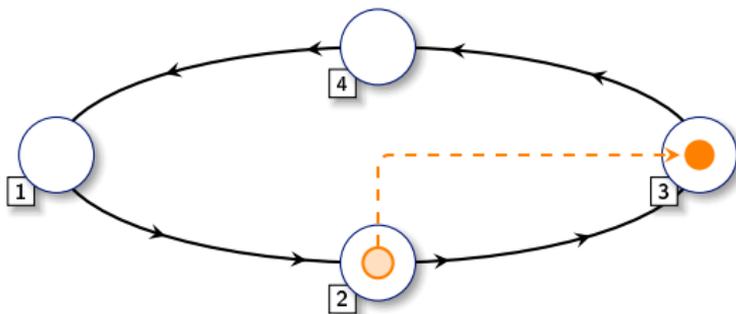
Token Ring Example



Modeling Programs

■ Configurations: 1000, 0100, 0010, 0001, 0000, 1100, ...

■ Transitions: $\begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

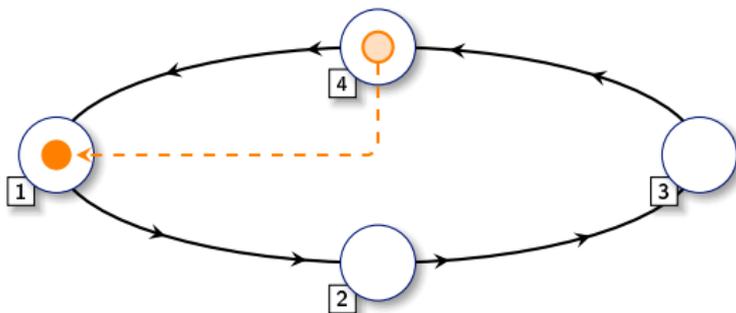


Modeling Programs

■ Configurations: 1000, 0100, 0010, 0001, 0000, 1100, ...

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

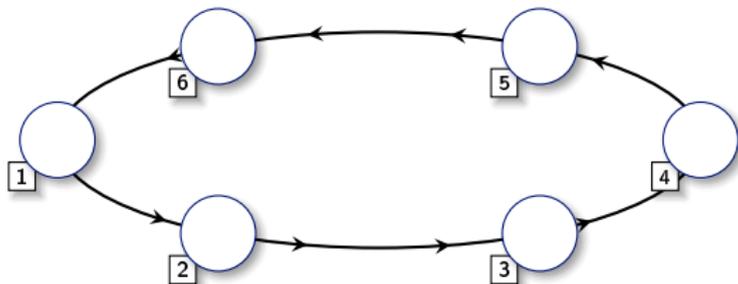
Token Ring Example



Modeling Programs

■ Configurations: 1000, 0100, 0010, 0001, 0000, 1100, ...

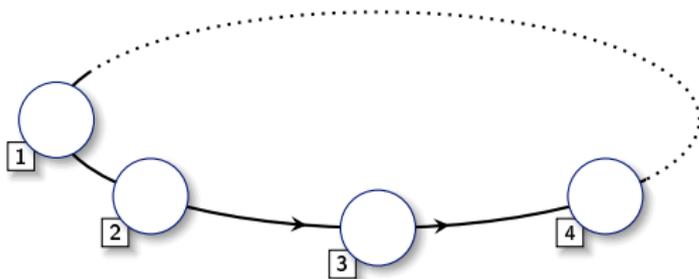
■ Transitions: $\begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix},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



Modeling Programs

■ Configurations: 100000, 010000, 001000, 000000, ...

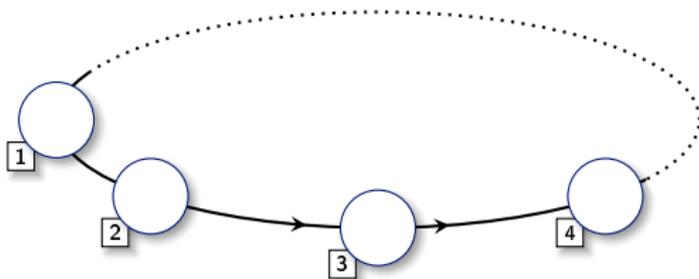
■ Transitions: $\begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \dots, \begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix}$



Modeling Programs

■ Configurations: $(0 + 1)^*$

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



The question we want to address

- Initial configurations $I = 10^*$
- Transitions $T = \dots$
- Bad configurations $B = 0^* + 0^*10^*1(0 + 1)^*$

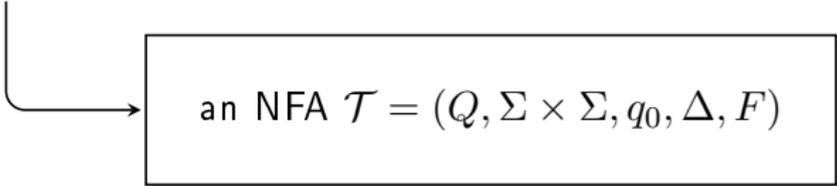
Is there a path from some $c \in I$ to some $c' \in B$ along T ?

Input

- A regular set of initial configurations I ,
- a regular set of bad configurations B , and
- a transducer \mathcal{T} defining the transitions.

Input

- A regular set of initial configurations I ,
- a regular set of bad configurations B , and
- a transducer \mathcal{T} defining the transitions.



an NFA $\mathcal{T} = (Q, \Sigma \times \Sigma, q_0, \Delta, F)$

Input

- A regular set of initial configurations I ,
- a regular set of bad configurations B , and
- a transducer \mathcal{T} defining the transitions.

Question

Does $\text{Reach}_{\mathcal{T}}(I) \cap B = \emptyset$ hold?

Input

- A regular set of initial configurations I ,
- a regular set of bad configurations B , and
- a transducer \mathcal{T} defining the transitions.

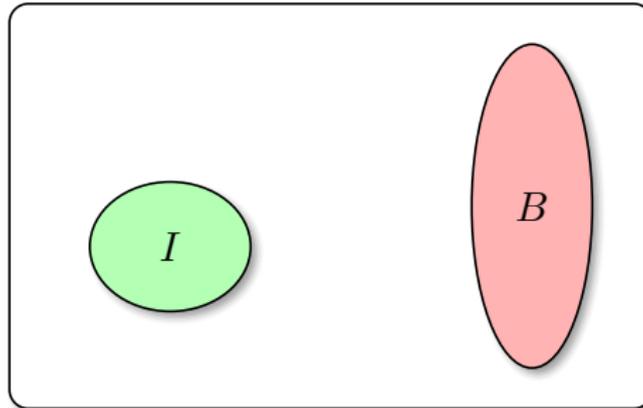
Question

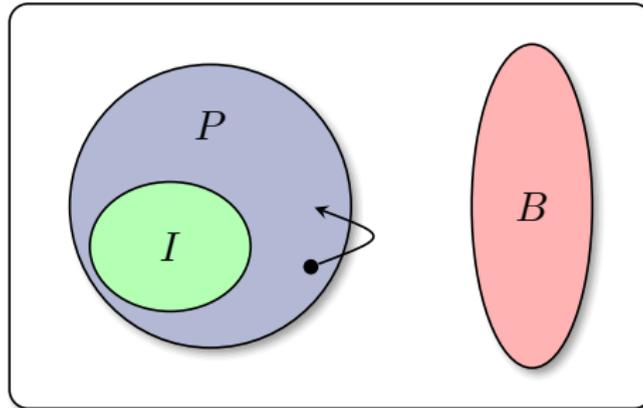
Does $\text{Reach}_{\mathcal{T}}(I) \cap B = \emptyset$ hold?

Remark

The Regular Model Checking Problem is undecidable.

Thus, all algorithms are necessarily semi-algorithms.

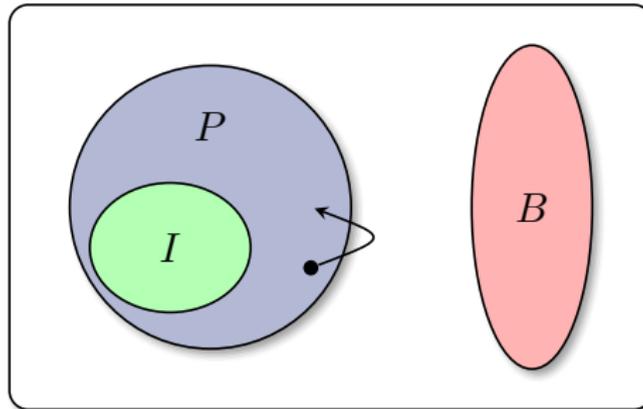




Proof

A (regular) proof is a (regular) set P with

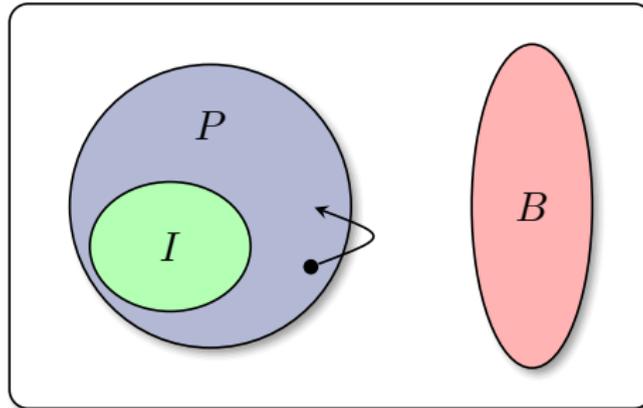
- $I \subseteq P$,
- $B \cap P = \emptyset$, and
- P is inductive, i.e., $u \in P$ and $(u, u') \in L(\mathcal{T})$ implies $u' \in P$.



Tools for Regular Model Checking

- Faster and
- T(O)RMC

Problem: The performance is poor for large representations of I !



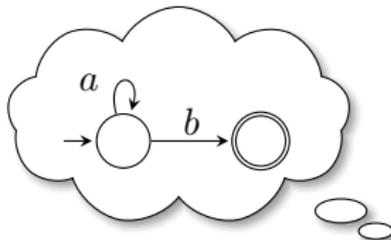
What we do

- We approximate I and B with finite sets.
- We use sampling strategies known from automata learning.
- We compute smallest proofs using logic solvers.

1. Automata Learning
2. Regular Model Checking via Automata Learning
3. Experiments
4. Conclusion

1. Automata Learning
 2. Regular Model Checking via Automata Learning
 3. Experiments
 4. Conclusion
-

Angluin's Learning Framework



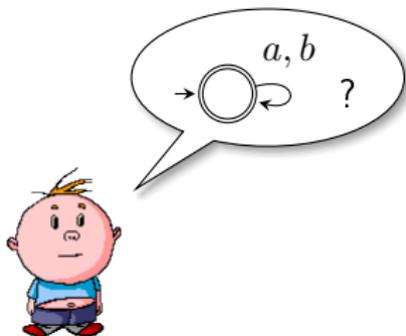
Angluin's Learning Framework



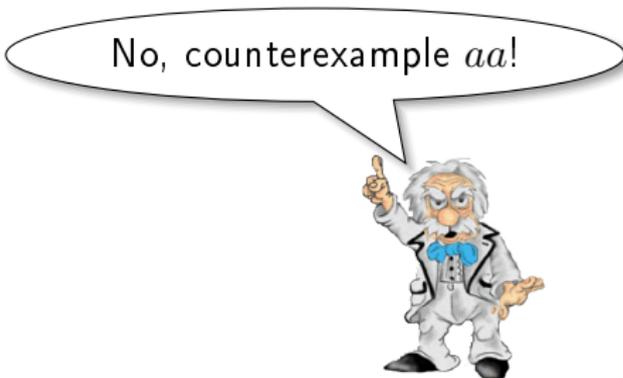
Angluin's Learning Framework



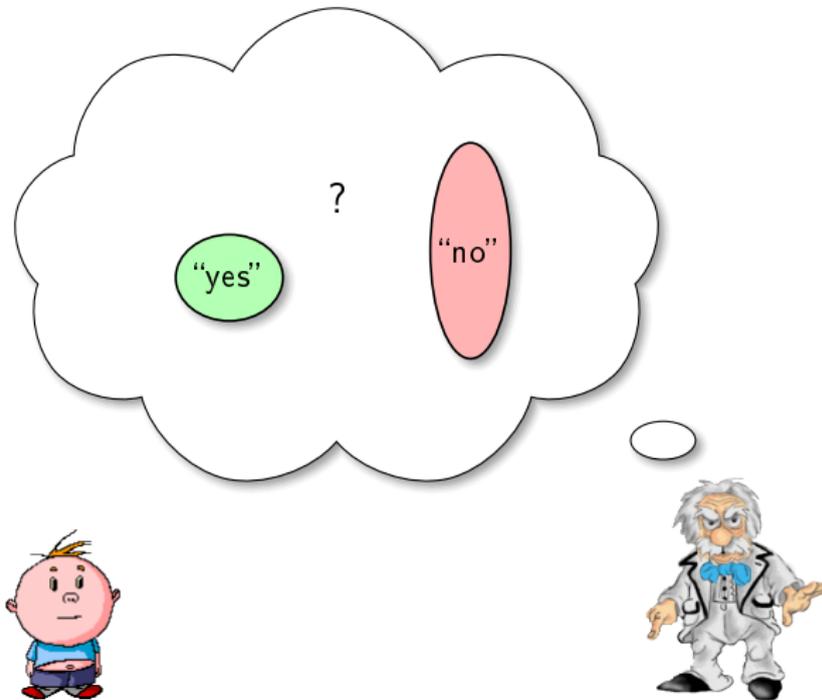
Angluin's Learning Framework



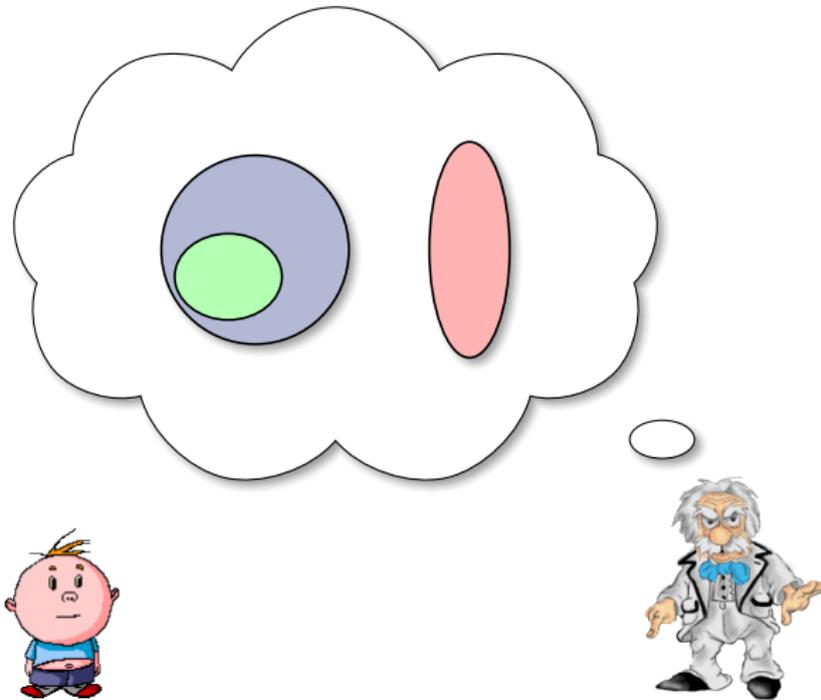
Angluin's Learning Framework



Our Learning Framework

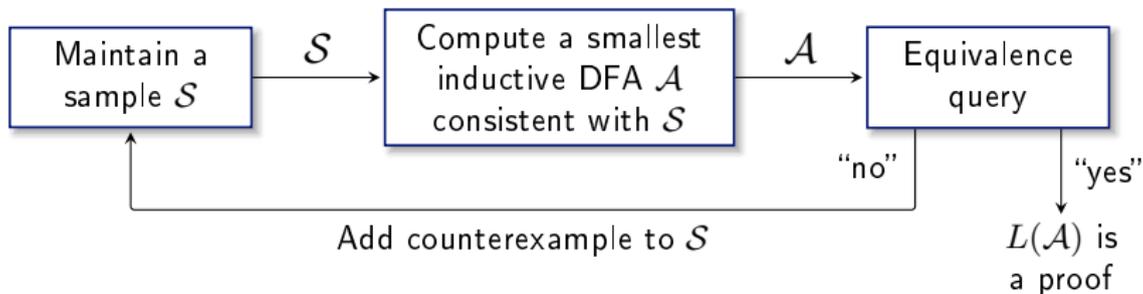


Our Learning Framework

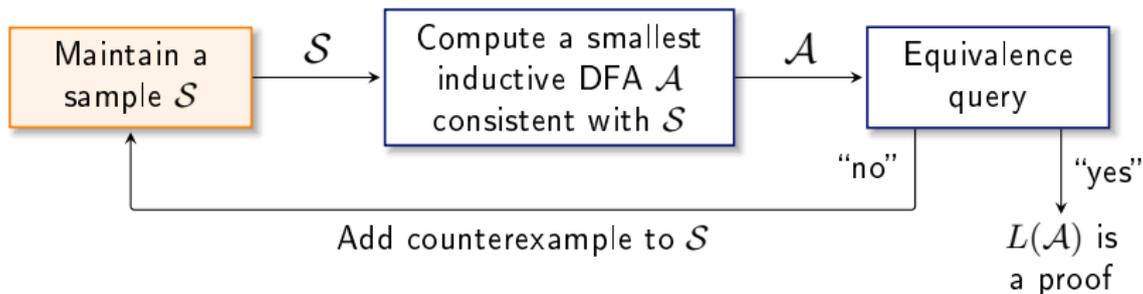


1. Automata Learning
 2. Regular Model Checking via Automata Learning
 3. Experiments
 4. Conclusion
-

CEGAR-style Regular Model Checking

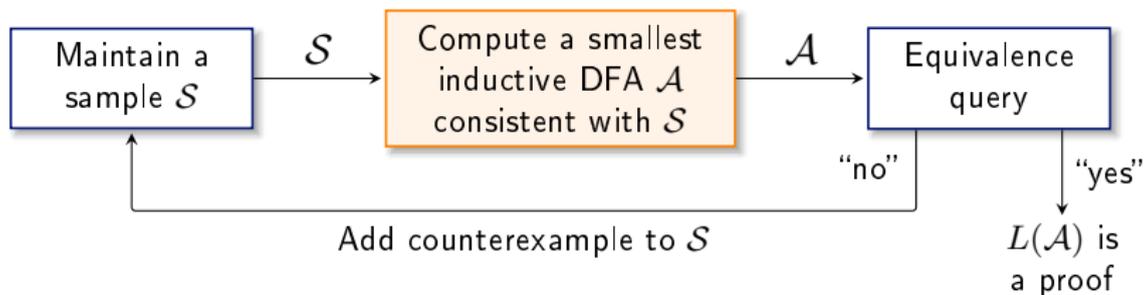


CEGAR-style Regular Model Checking



1. Sample $\mathcal{S} = (S_+, S_-)$ where $S_+, S_- \subseteq \Sigma^*$ are finite.

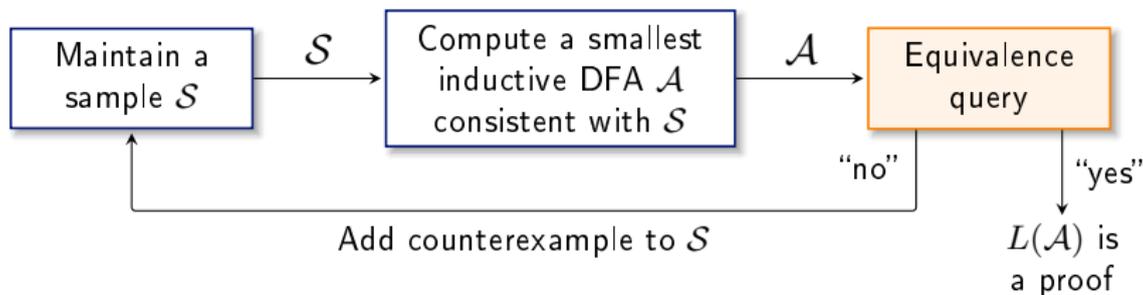
CEGAR-style Regular Model Checking



1. Sample $\mathcal{S} = (S_+, S_-)$ where $S_+, S_- \subseteq \Sigma^*$ are finite.
2. Compute a smallest inductive DFA \mathcal{A} consistent with \mathcal{S} , i.e.,

$$S_+ \subseteq L(\mathcal{A}) \text{ and } S_- \cap L(\mathcal{A}) = \emptyset.$$

CEGAR-style Regular Model Checking



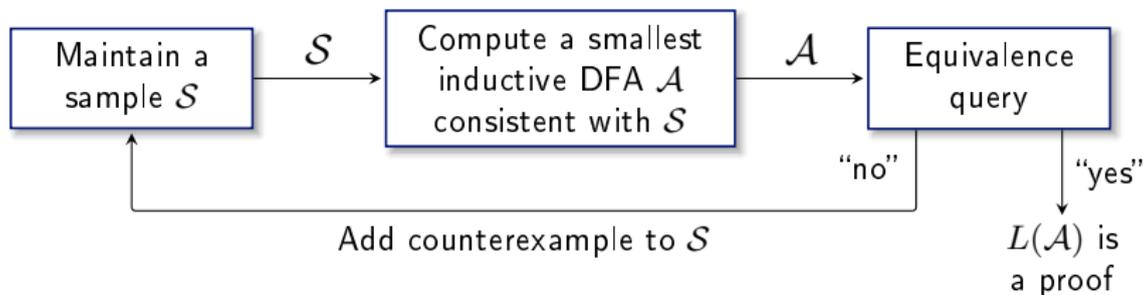
1. Sample $\mathcal{S} = (S_+, S_-)$ where $S_+, S_- \subseteq \Sigma^*$ are finite.
2. Compute a smallest inductive DFA \mathcal{A} consistent with \mathcal{S} , i.e.,

$$S_+ \subseteq L(\mathcal{A}) \text{ and } S_- \cap L(\mathcal{A}) = \emptyset.$$

3. Equivalence query: if \mathcal{A} is inductive, it is enough to check

$$I \subseteq L(\mathcal{A}) \text{ and } B \cap L(\mathcal{A}) = \emptyset.$$

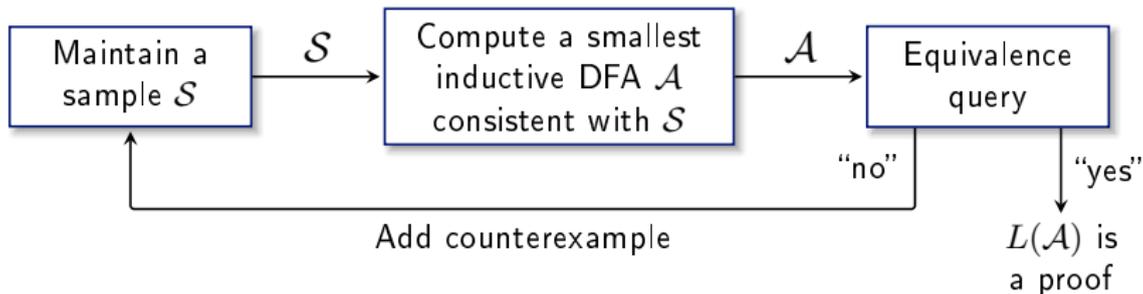
CEGAR-style Regular Model Checking



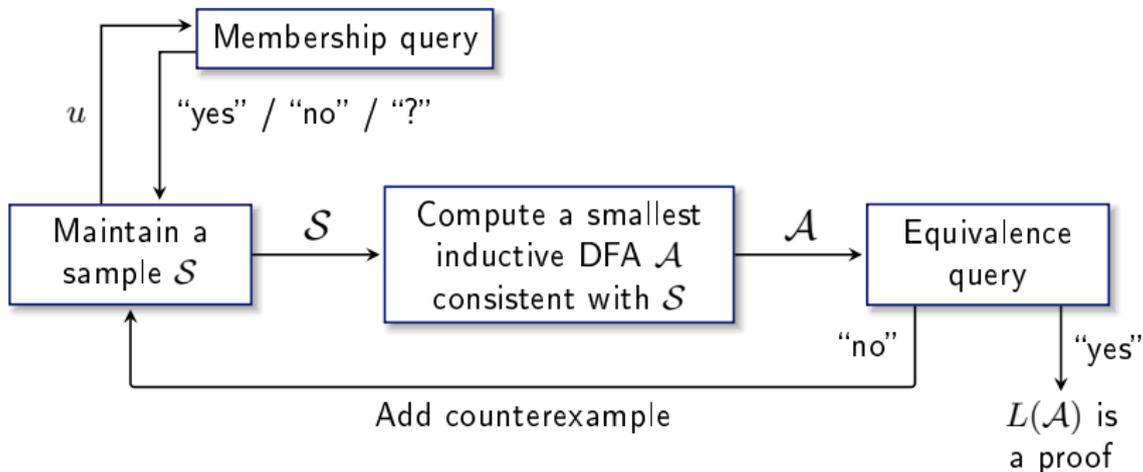
Correctness

- The algorithm terminates once $L(\mathcal{A})$ is a proof.
- Successive DFAs are different and the size of successive DFAs increases monotonically.
- Computing minimal DFAs guarantees termination if a proof exists.

Angluin-style Regular Model Checking



Angluin-style Regular Model Checking



Given a sample \mathcal{S} and a transducer \mathcal{T} .

Main Idea

Construct a formula $\varphi_n^{\mathcal{S}, \mathcal{T}}$ such that

$\varphi_n^{\mathcal{S}, \mathcal{T}}$ is satisfiable

\Leftrightarrow

there exists a DFA \mathcal{A} with n states such that \mathcal{A} is consistent with \mathcal{S} and inductive with respect to \mathcal{T} .

Given a sample \mathcal{S} and a transducer \mathcal{T} .

Main Idea

Construct a formula $\varphi_n^{\mathcal{S}, \mathcal{T}}$ such that

$\varphi_n^{\mathcal{S}, \mathcal{T}}$ is satisfiable

\Leftrightarrow

there exists a DFA \mathcal{A} with n states such that \mathcal{A} is consistent with \mathcal{S} and inductive with respect to \mathcal{T} .

Theorem

By increasing the value of n , we will find a smallest DFA consistent with \mathcal{S} and inductive with respect to \mathcal{T} if one exists.

$$\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$$

If Q , Σ , and q_0 are fixed, then every DFA is completely defined by δ and F .

$$\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$$

If Q , Σ , and q_0 are fixed, then every DFA is completely defined by δ and F .

- Use Boolean variables $d_{p,a,q}$ with the meaning:
if $d_{p,a,q} \equiv \text{true}$, then $\delta(p, a) = q$.
- Use Boolean variables f_q with the meaning:
if $f_q \equiv \text{true}$, then $q \in F$.

$$\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$$

If Q , Σ , and q_0 are fixed, then every DFA is completely defined by δ and F .

- Use Boolean variables $d_{p,a,q}$ with the meaning:
if $d_{p,a,q} \equiv \text{true}$, then $\delta(p, a) = q$.
- Use Boolean variables f_q with the meaning:
if $f_q \equiv \text{true}$, then $q \in F$.

Use constraints

$$\neg d_{p,a,q} \vee \neg d_{p,a,q'}$$
$$\bigvee_{q \in Q} d_{p,a,q}$$

Let φ_n^{DFA} be the conjunction of these constraints.

Construction of a DFA from a Model

Let $\mathfrak{M} \models \varphi_n^{\text{DFA}}$. Then, we construct a DFA

$\mathcal{A}_{\mathfrak{M}} = (\{q_0, \dots, q_{n-1}\}, \Sigma, q_0, \delta, F)$ as follows:

- $\delta(p, a) = q$ for the unique $q \in Q$ such that $\mathfrak{M}(d_{p,a,q}) = \text{true}$.
- $q \in F$ if and only if $\mathfrak{M}(f_q) = \text{true}$.

Construction of a DFA from a Model

Let $\mathfrak{M} \models \varphi_n^{\text{DFA}}$. Then, we construct a DFA

$\mathcal{A}_{\mathfrak{M}} = (\{q_0, \dots, q_{n-1}\}, \Sigma, q_0, \delta, F)$ as follows:

- $\delta(p, a) = q$ for the unique $q \in Q$ such that $\mathfrak{M}(d_{p,a,q}) = \text{true}$.
- $q \in F$ if and only if $\mathfrak{M}(f_q) = \text{true}$.

Impose restrictions on the behavior of $\mathcal{A}_{\mathfrak{M}}$ by introducing the two formulas

- $\varphi_n^{\mathcal{S}}$ that enforces that $\mathcal{A}_{\mathfrak{M}}$ is consistent with \mathcal{S} , and
- $\varphi_n^{\mathcal{T}}$ that enforces that $\mathcal{A}_{\mathfrak{M}}$ is inductive with respect to \mathcal{T} .

Construction of a DFA from a Model

Let $\mathfrak{M} \models \varphi_n^{\text{DFA}}$. Then, we construct a DFA

$\mathcal{A}_{\mathfrak{M}} = (\{q_0, \dots, q_{n-1}\}, \Sigma, q_0, \delta, F)$ as follows:

- $\delta(p, a) = q$ for the unique $q \in Q$ such that $\mathfrak{M}(d_{p,a,q}) = \text{true}$.
- $q \in F$ if and only if $\mathfrak{M}(f_q) = \text{true}$.

Impose restrictions on the behavior of $\mathcal{A}_{\mathfrak{M}}$ by introducing the two formulas

- $\varphi_n^{\mathcal{S}}$ that enforces that $\mathcal{A}_{\mathfrak{M}}$ is consistent with \mathcal{S} , and
- $\varphi_n^{\mathcal{T}}$ that enforces that $\mathcal{A}_{\mathfrak{M}}$ is inductive with respect to \mathcal{T} .

$\varphi_n^{\mathcal{S}, \mathcal{T}} := \varphi_n^{\text{DFA}} \wedge \varphi_n^{\mathcal{S}} \wedge \varphi_n^{\mathcal{T}}$ is the desired formula.

Let $\mathcal{S} = (S_+, S_-)$. Establish $S_+ \subseteq L(\mathcal{A}_M)$ and $S_- \cap L(\mathcal{A}_M) = \emptyset$!

Let $S = (S_+, S_-)$. Establish $S_+ \subseteq L(\mathcal{A}_M)$ and $S_- \cap L(\mathcal{A}_M) = \emptyset$!

Introduce variables $x_{u,q}$ for $u \in \text{Pref}(S_+ \cup S_-)$ with the meaning:

if $\mathcal{A}_M: q_0 \xrightarrow{u} q$, then $x_{u,q} \equiv \text{true}$.

Let $S = (S_+, S_-)$. Establish $S_+ \subseteq L(\mathcal{A}_M)$ and $S_- \cap L(\mathcal{A}_M) = \emptyset$!

Introduce variables $x_{u,q}$ for $u \in \text{Pref}(S_+ \cup S_-)$ with the meaning:

if $\mathcal{A}_M: q_0 \xrightarrow{u} q$, then $x_{u,q} \equiv \text{true}$.

Introduce constraints:

$$x_{\varepsilon, q_0}$$

$$(x_{u,p} \wedge d_{p,a,q}) \rightarrow x_{ua,q} \quad \text{for } ua \in \text{Pref}(S_+ \cup S_-)$$

$$x_{u,q} \rightarrow f_q \quad \text{for } u \in S_+$$

$$x_{u,q} \rightarrow \neg f_q \quad \text{for } u \in S_-$$

Let $S = (S_+, S_-)$. Establish $S_+ \subseteq L(\mathcal{A}_M)$ and $S_- \cap L(\mathcal{A}_M) = \emptyset$!

Introduce variables $x_{u,q}$ for $u \in \text{Pref}(S_+ \cup S_-)$ with the meaning:

if $\mathcal{A}_M: q_0 \xrightarrow{u} q$, then $x_{u,q} \equiv \text{true}$.

Introduce constraints:

$$\begin{aligned}
 & x_{\varepsilon, q_0} \\
 & (x_{u,p} \wedge d_{p,a,q}) \rightarrow x_{ua,q} \quad \text{for } ua \in \text{Pref}(S_+ \cup S_-) \\
 & x_{u,q} \rightarrow f_q \quad \text{for } u \in S_+ \\
 & x_{u,q} \rightarrow \neg f_q \quad \text{for } u \in S_-
 \end{aligned}$$

Let φ_n^S be the conjunction of these constraints. Then,
 $M \models \varphi_n^{\text{DFA}} \wedge \varphi_n^S$ implies $S_+ \subseteq L(\mathcal{A}_M)$ and $S_- \cap L(\mathcal{A}_M) = \emptyset$.

Let $\mathcal{T} = (Q^T, \Sigma \times \Sigma, q_0^T, \Delta^T, F^T)$. Establish
 $u \in L(\mathcal{A}_{\mathcal{M}}) \wedge (u, u') \in L(\mathcal{T}) \Rightarrow u' \in L(\mathcal{A}_{\mathcal{M}})!$

Let $\mathcal{T} = (Q^T, \Sigma \times \Sigma, q_0^T, \Delta^T, F^T)$. Establish
 $u \in L(\mathcal{A}_{\mathcal{M}}) \wedge (u, u') \in L(\mathcal{T}) \Rightarrow u' \in L(\mathcal{A}_{\mathcal{M}})!$

Introduce variables $y_{q,q',q''}$ with with the meaning:

$$\begin{array}{l} \mathcal{A}_{\mathcal{M}}: \quad q_0 \xrightarrow{u} q \\ \text{if there are } u, u' \in \Sigma^* \text{ such that } \mathcal{T} : \quad q_0^T \xrightarrow{(u,u')} q' , \\ \mathcal{A}_{\mathcal{M}}: \quad q_0 \xrightarrow{u'} q'' \end{array}$$

then $y_{q,q',q''} \equiv \text{true}$.

Introduce constraints:

$$y_{q_0, q_0^T, q_0}$$

$$(y_{p, p', p''} \wedge d_{p, a, q} \wedge d_{p'', b, q''}) \rightarrow z_{q, q', q''} \quad \text{for } (p', (a, b), q') \in \Delta^T$$

$$(y_{q, q', q''} \wedge f_q) \rightarrow f_{q''} \quad \text{for } q' \in F^T$$

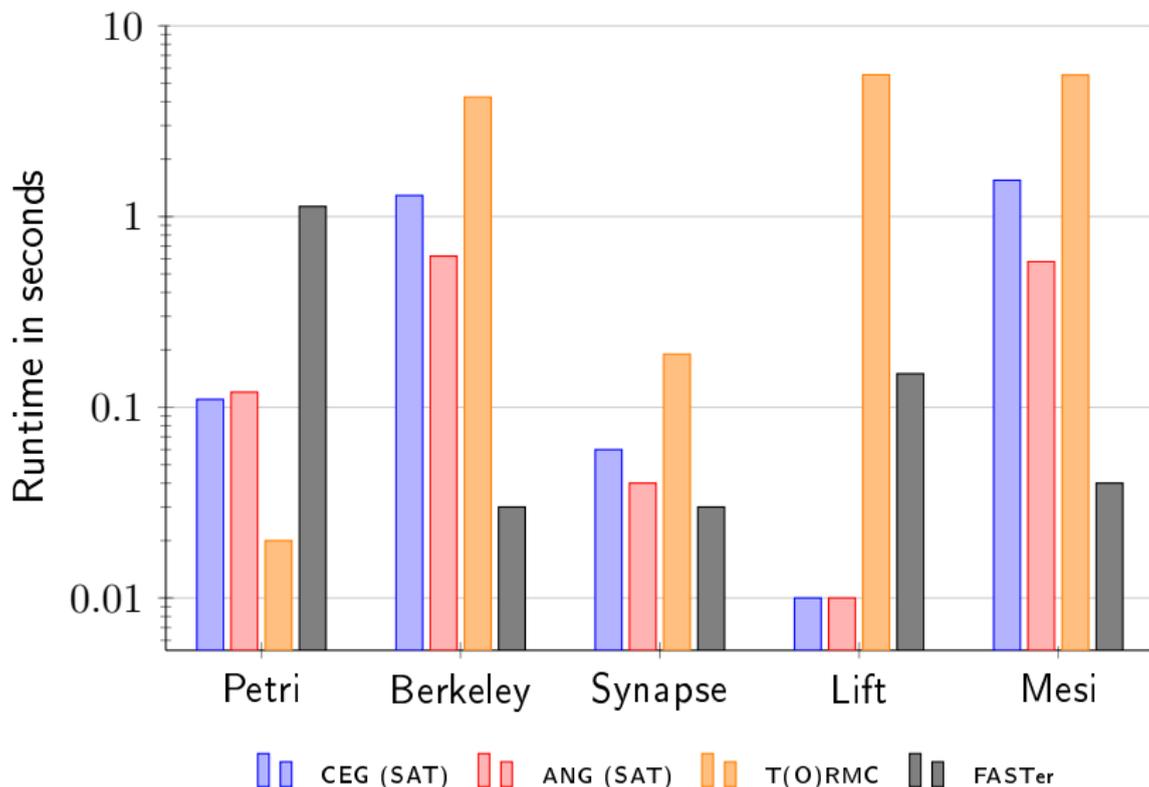
Let φ_n^T be the conjunction of these constraints.

Then, $\mathfrak{M} \models \varphi_n^{\text{DFA}} \wedge \varphi_n^T$ implies

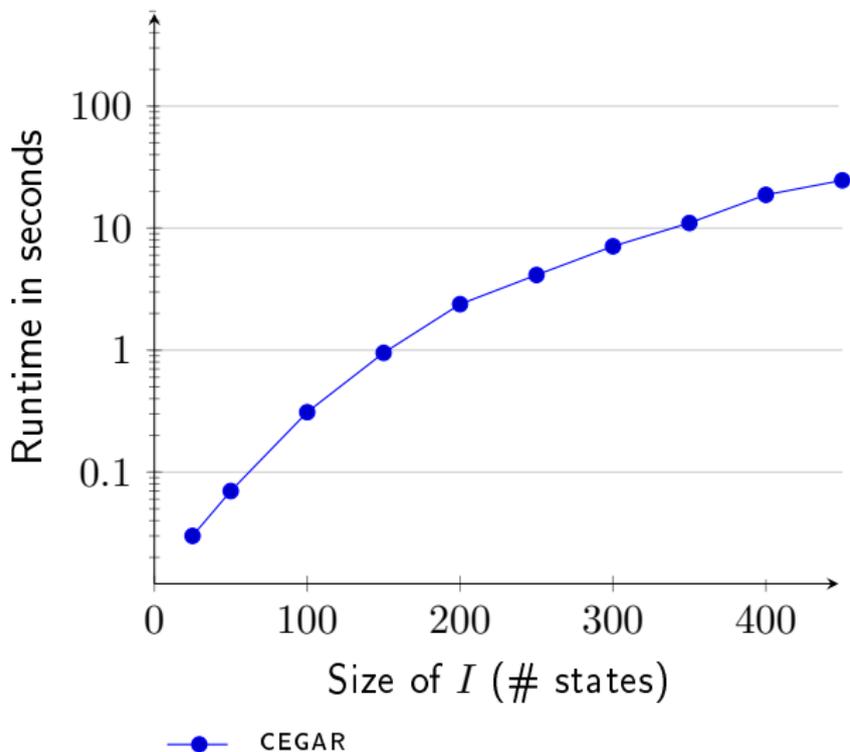
$$u \in L(\mathcal{A}_{\mathfrak{M}}) \wedge (u, u') \in L(\mathcal{T}) \Rightarrow u' \in L(\mathcal{A}_{\mathfrak{M}}).$$

1. Automata Learning
 2. Regular Model Checking via Automata Learning
 - 3. Experiments**
 4. Conclusion
-

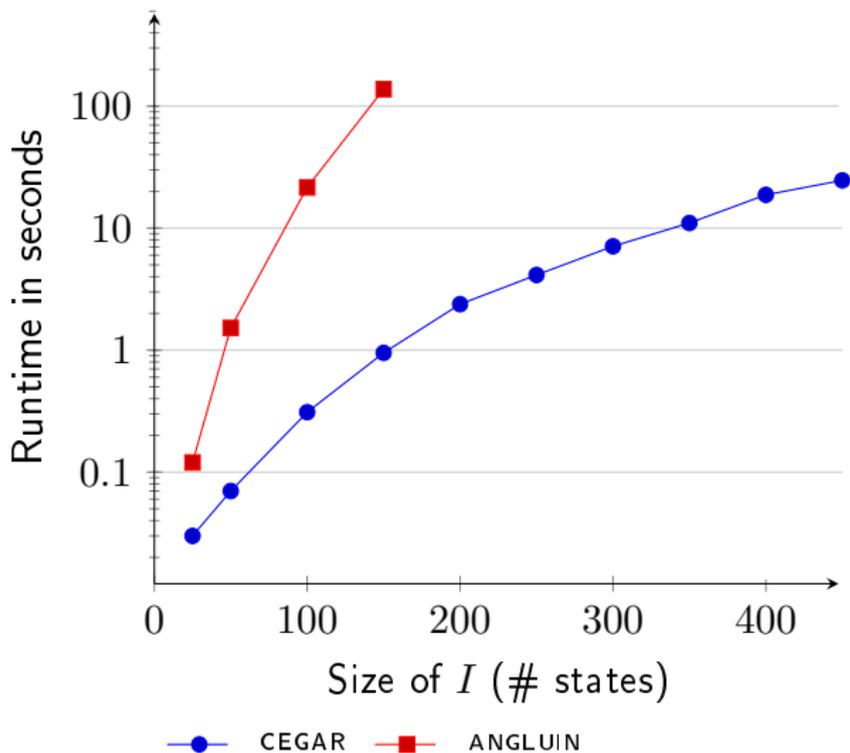
Experiments - Integer Linear Systems



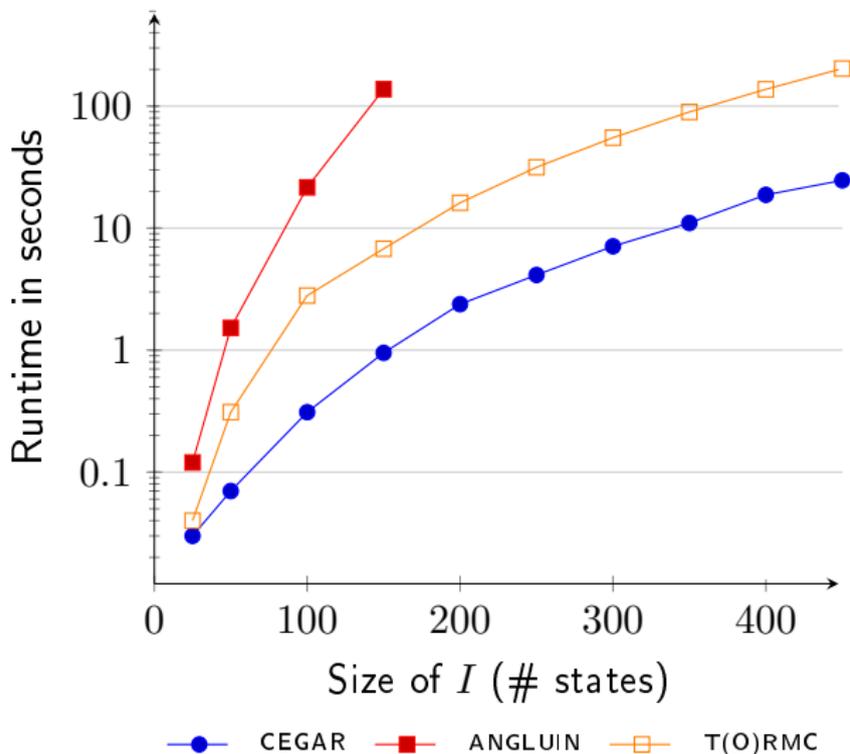
Experiments - Token Ring



Experiments - Token Ring



Experiments - Token Ring



1. Automata Learning
 2. Regular Model Checking via Automata Learning
 3. Experiments
 - 4. Conclusion**
-

Summary

- We presented a new technique for Regular Model Checking based upon automata learning and logic solver.
- Large sets of initial and bad configurations are approximated.
- Experiments show competitiveness to other available tools.

Summary

- We presented a new technique for Regular Model Checking based upon automata learning and logic solver.
- Large sets of initial and bad configurations are approximated.
- Experiments show competitiveness to other available tools.

Further Research

- Possible directions of future research are
 - suitable non-regular representations of I and B ,
 - nondeterministic automata as proofs, and
 - an incremental SAT approach.
- An interesting extension would be to also approximate the transducer.