京都大学
KYOTO UNIVERSITY

Specification-Based Testing of Cyber-Physical Systems via Black-Box Checking

Masaki Waga

Kyoto University

2025 December 15th, QuantFormal 2025

Joint work with Junya Shijubo, Tsubasa Matsumoto, Kazuki Watanabe, and Kohei Suenaga

Safety Critical CPS

NEWS

Home | Israel-Gaza war | War in Ukraine | Climate | Video | World | Asia | UK | More

Business | Tech

Flight disruption warning as Airbus requests modifications to 6,000 planes

29 November 2025

Tesla investigated over self-driving cars on wrong side of road

10 October 2025

Share Save

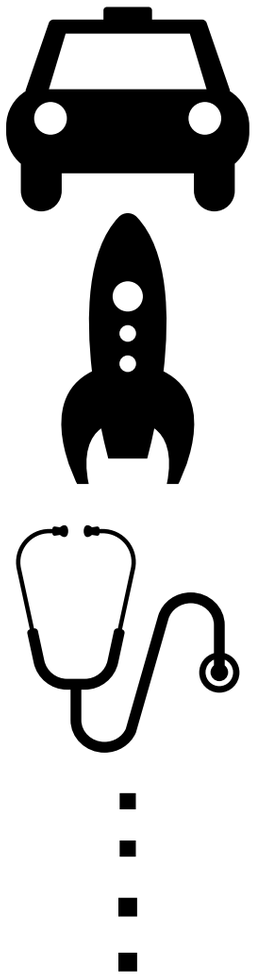
Imran Rahman-Jones

Technology reporter

Bloomberg via Getty Images

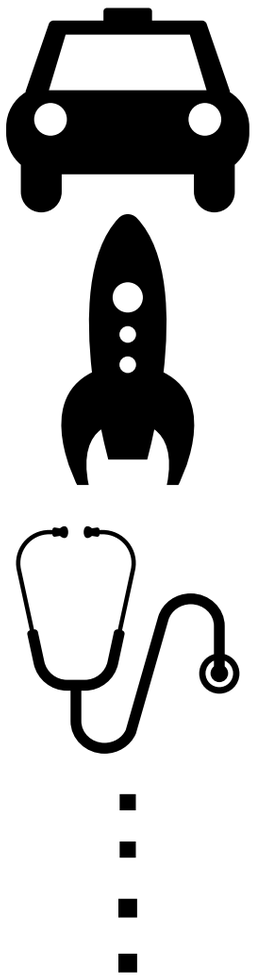
https://www.bbc.com/news/live/cvg4y6g74ert
https://www.bbc.com/news/articles/cvg02rdxxz7o

Q. How to Trust Safety Critical Systems?

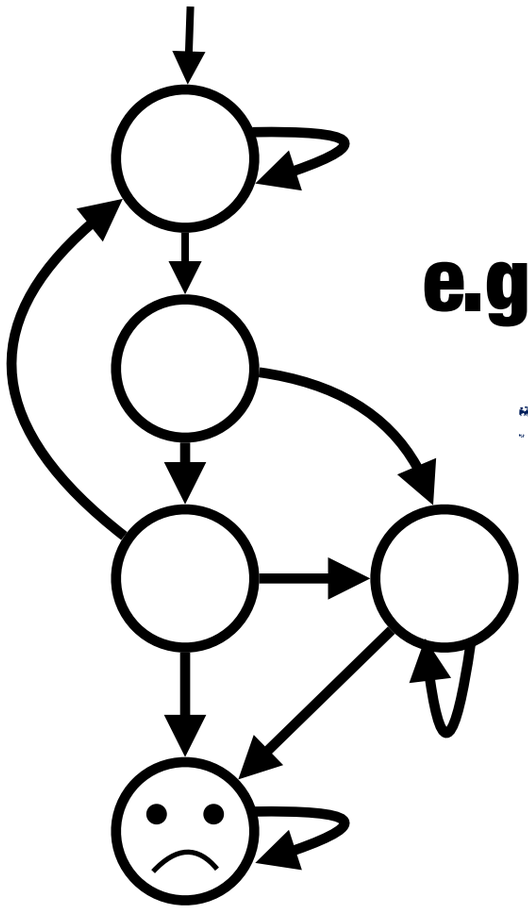


Q. How to Trust Safety Critical Systems?

Approach 1: Guarantee safety or find bugs via model checking



Formal Modeling



Verification
e.g. Model Checking

✔

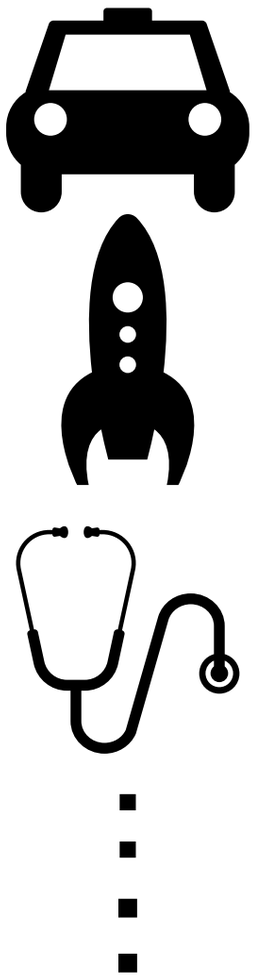
✘

?

Prob.
Param.
⋮

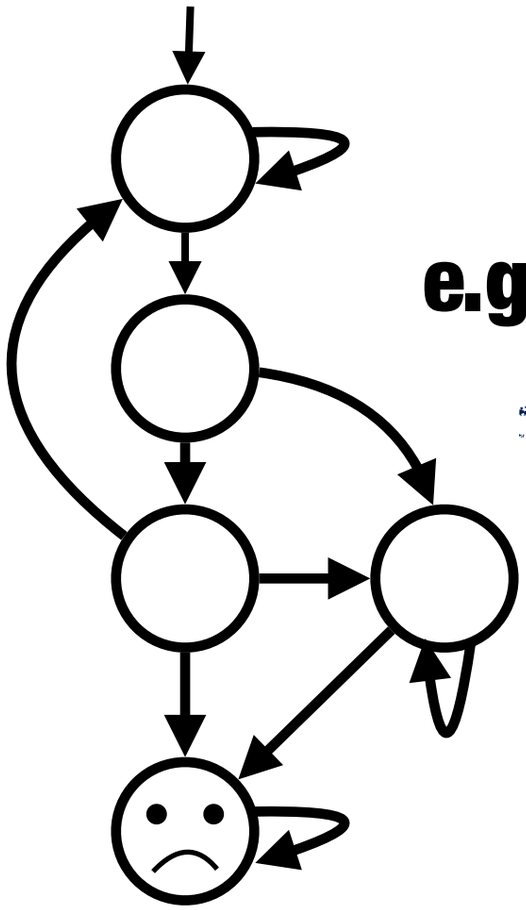
Q. How to Trust Black-box Safety Critical Systems?

Approach 1: Guarantee safety or find bugs via model checking



Formal Modeling

Modeling is not difficult!



Verification
e.g. Model Checking

✔

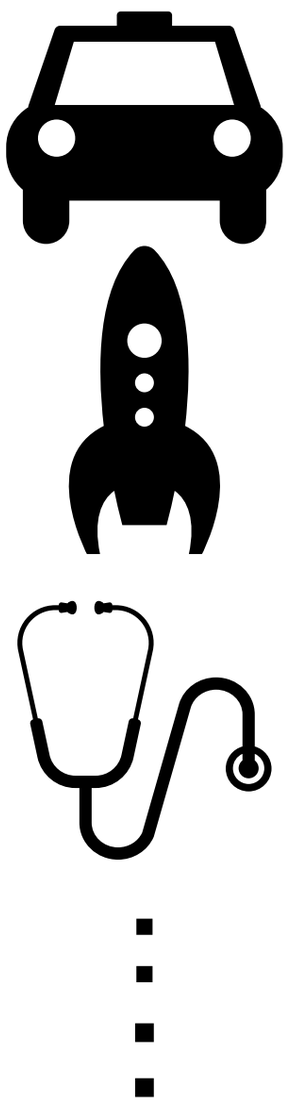
✘

?

Prob.
Param.
▪
▪
▪
▪

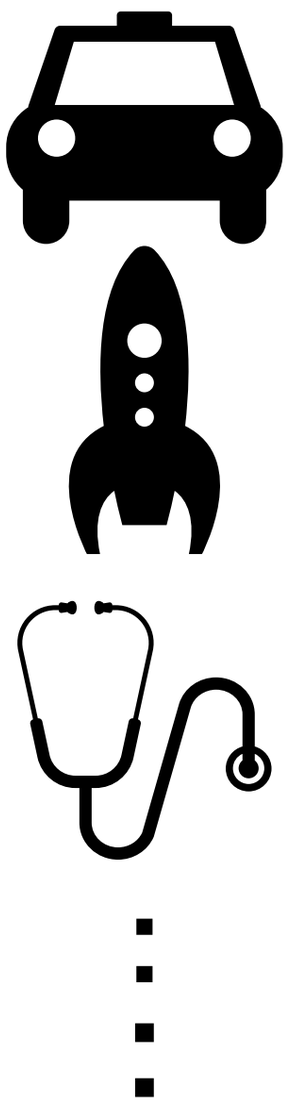
Testing

Approach 2: Sample inputs, feed it to the system, reveal bugs



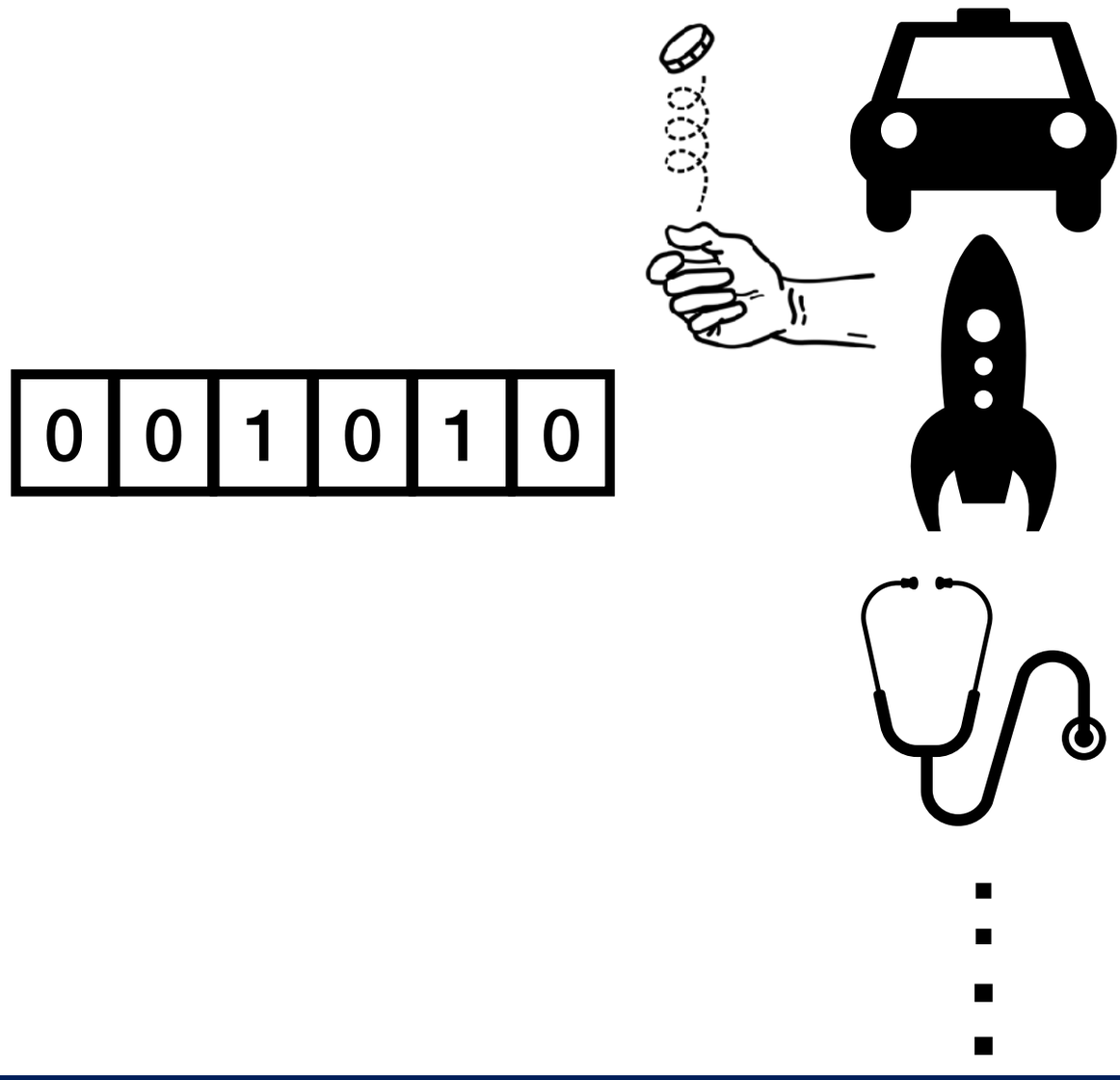
Sampling-based Testing

Approach 2: Sample inputs, feed it to the system, reveal bugs



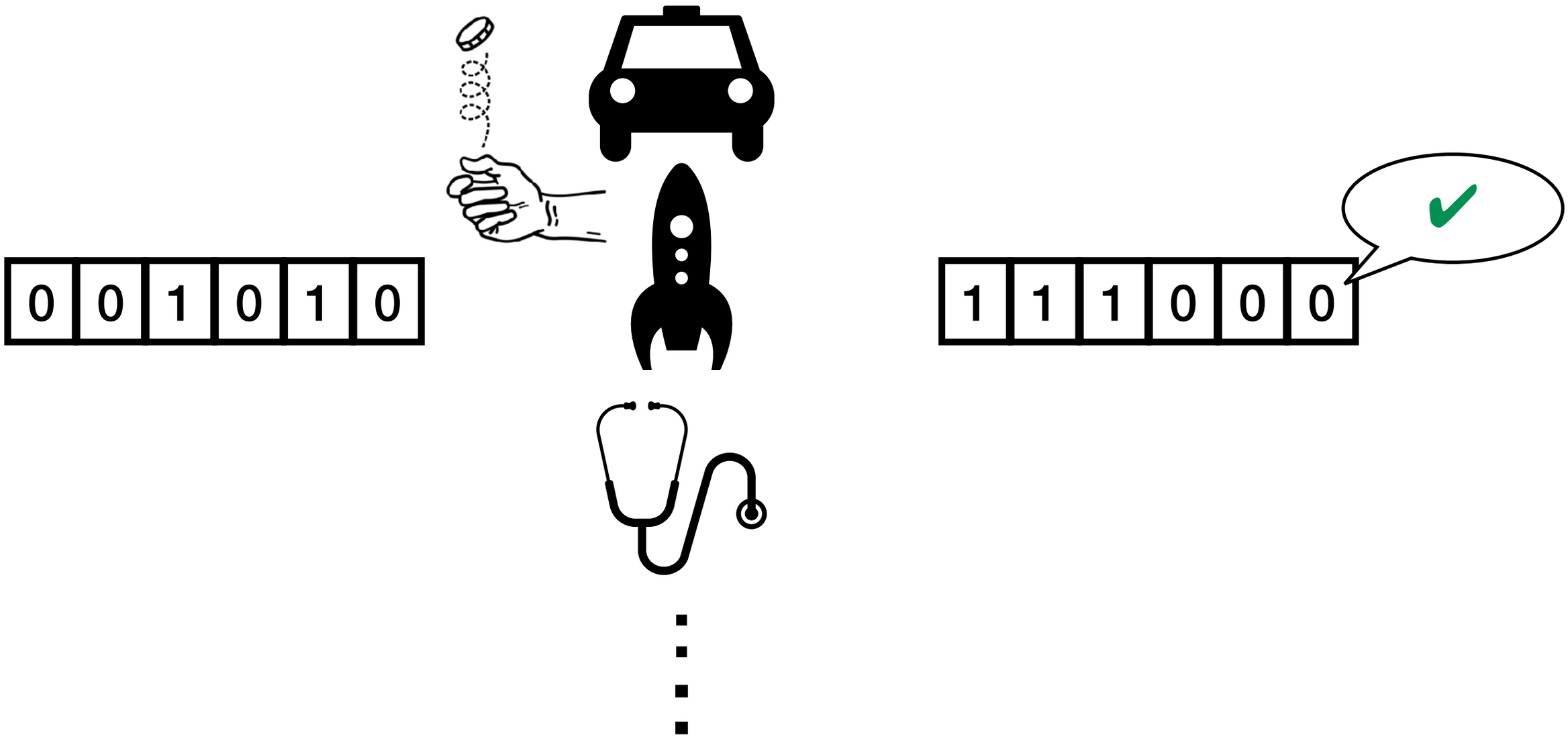
Sampling-based Testing

Approach 2: Sample inputs, feed it to the system, reveal bugs



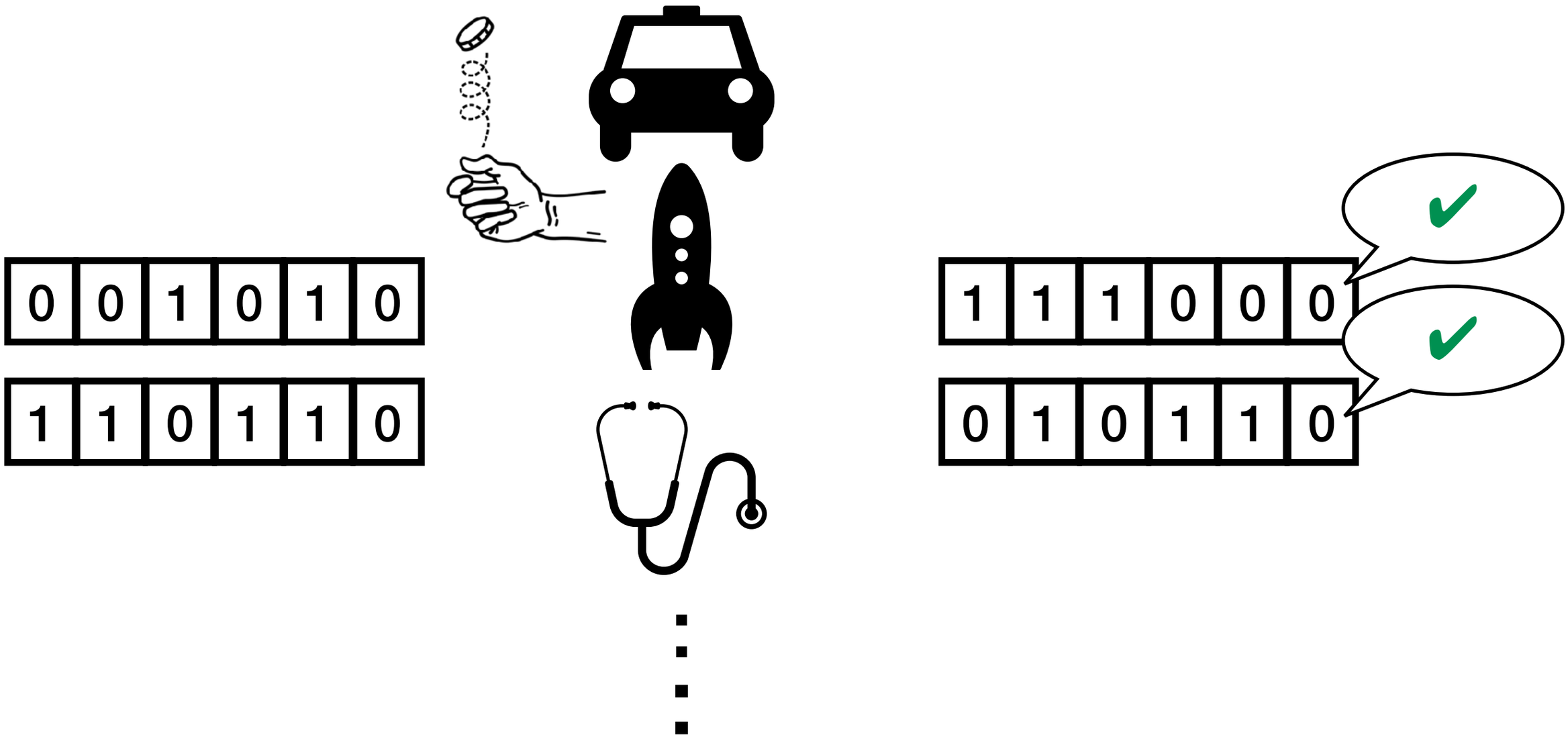
Sampling-based Testing

Approach 2: Sample inputs, feed it to the system, reveal bugs



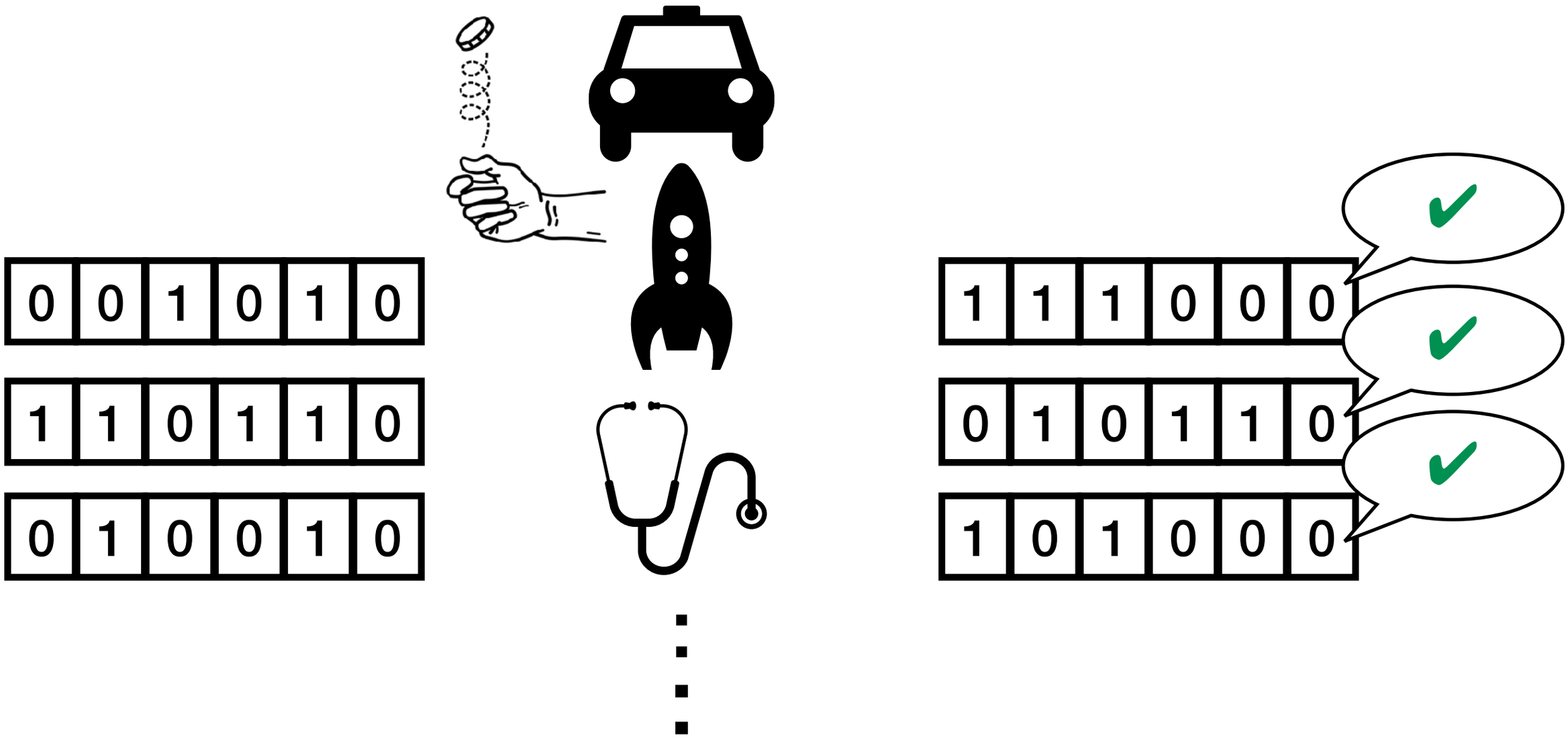
Sampling-based Testing

Approach 2: Sample inputs, feed it to the system, reveal bugs



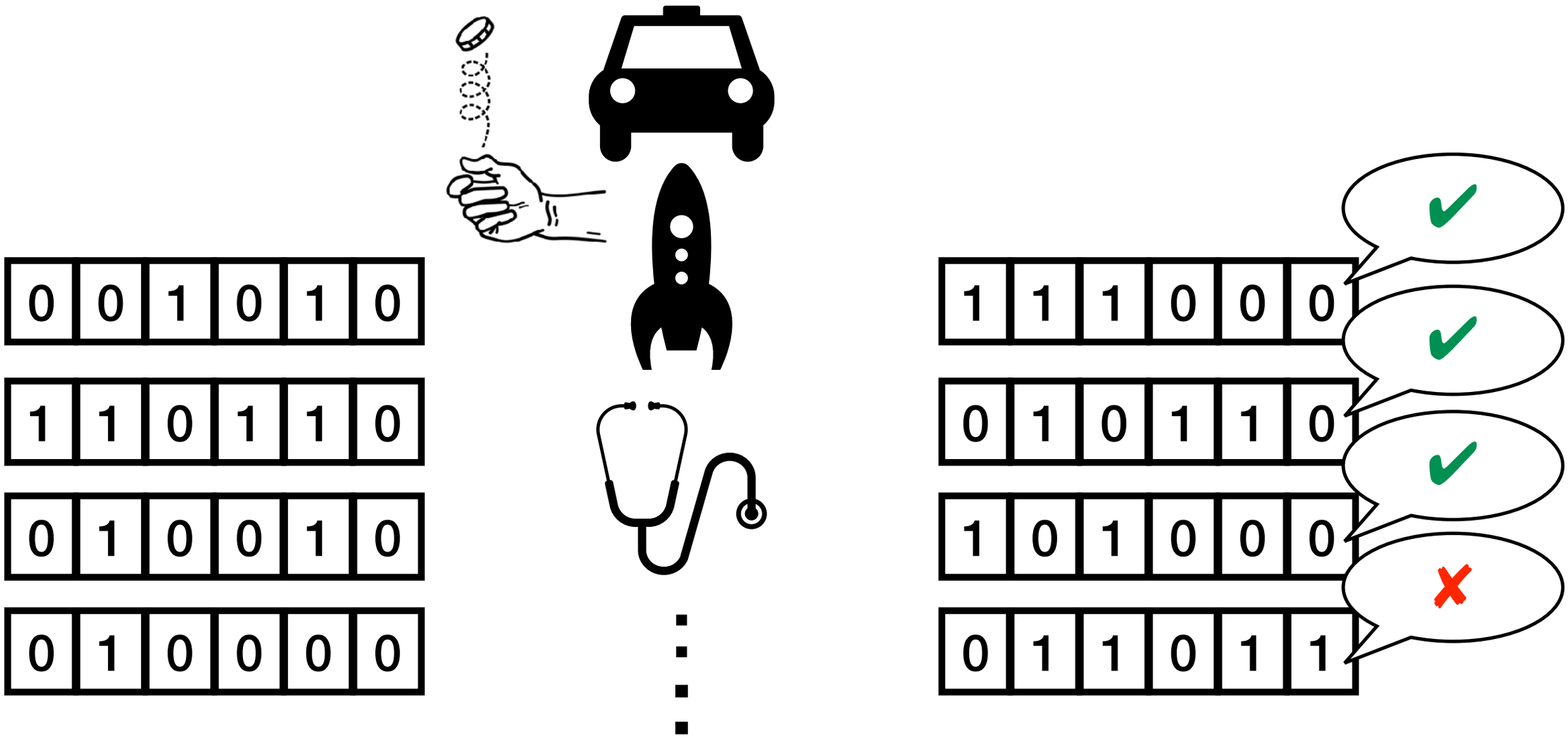
Sampling-based Testing

Approach 2: Sample inputs, feed it to the system, reveal bugs



Sampling-based Testing

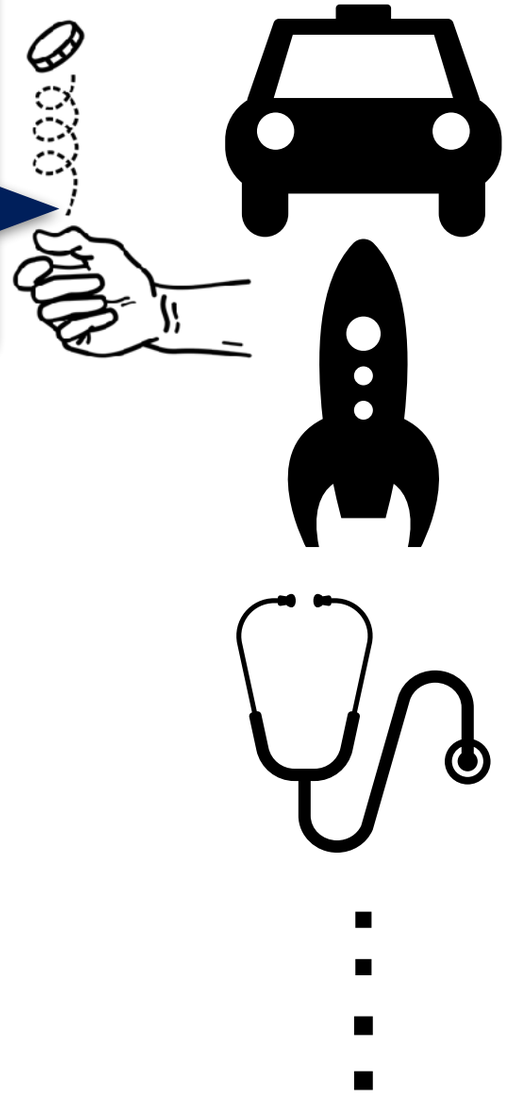
Approach 2: Sample inputs, feed it to the system, reveal bugs



Sampling-based Testing

Approach 2: Sample inputs, feed it to the system, reveal bugs

Can be biased to pick less safe inputs (cf. robustness of signal temporal logic [Fainekos & Pappas, TCS'09])



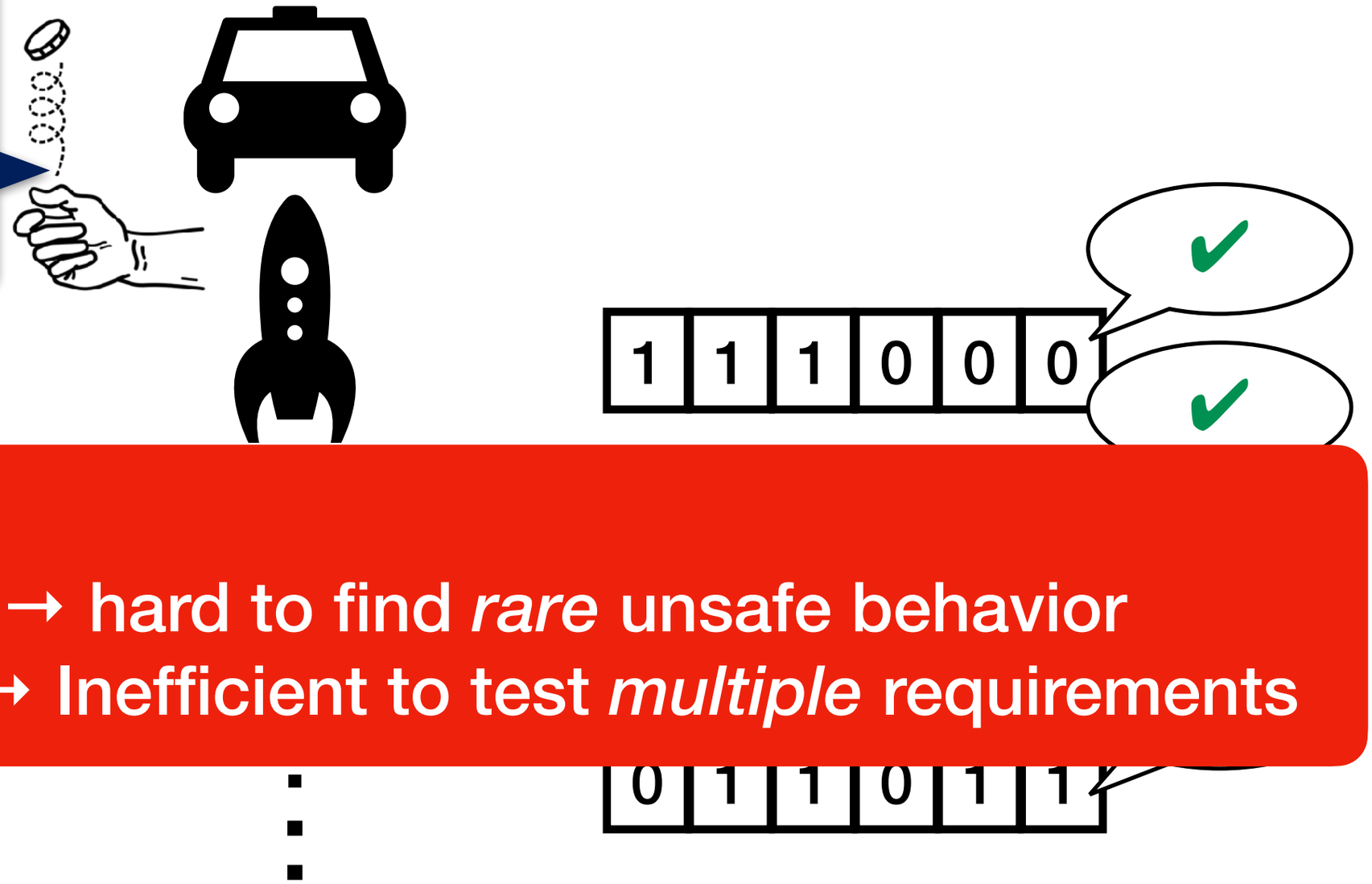
0	0	1	0	1	0
1	1	0	1	1	0
0	1	0	0	1	0
0	1	0	0	0	0

1	1	1	0	0	0	✔
0	1	0	1	1	0	✔
1	0	1	0	0	0	✔
0	1	1	0	1	1	✘

Sampling-based Testing

Approach 2: Sample inputs, feed it to the system, reveal bugs

Can be biased to pick less safe inputs (cf. robustness of signal temporal logic [Fainekos & Pappas, TCS'09])

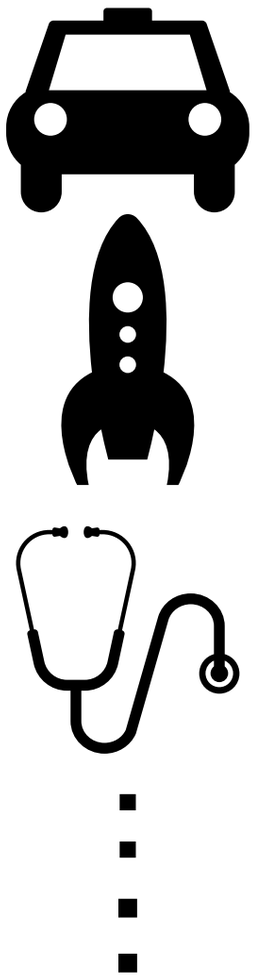


Limitation:
Uniform sampling → hard to find *rare* unsafe behavior
Biased sampling → Inefficient to test *multiple* requirements

Black-Box Checking (BBC)

[Peled et al., FORTE'99]

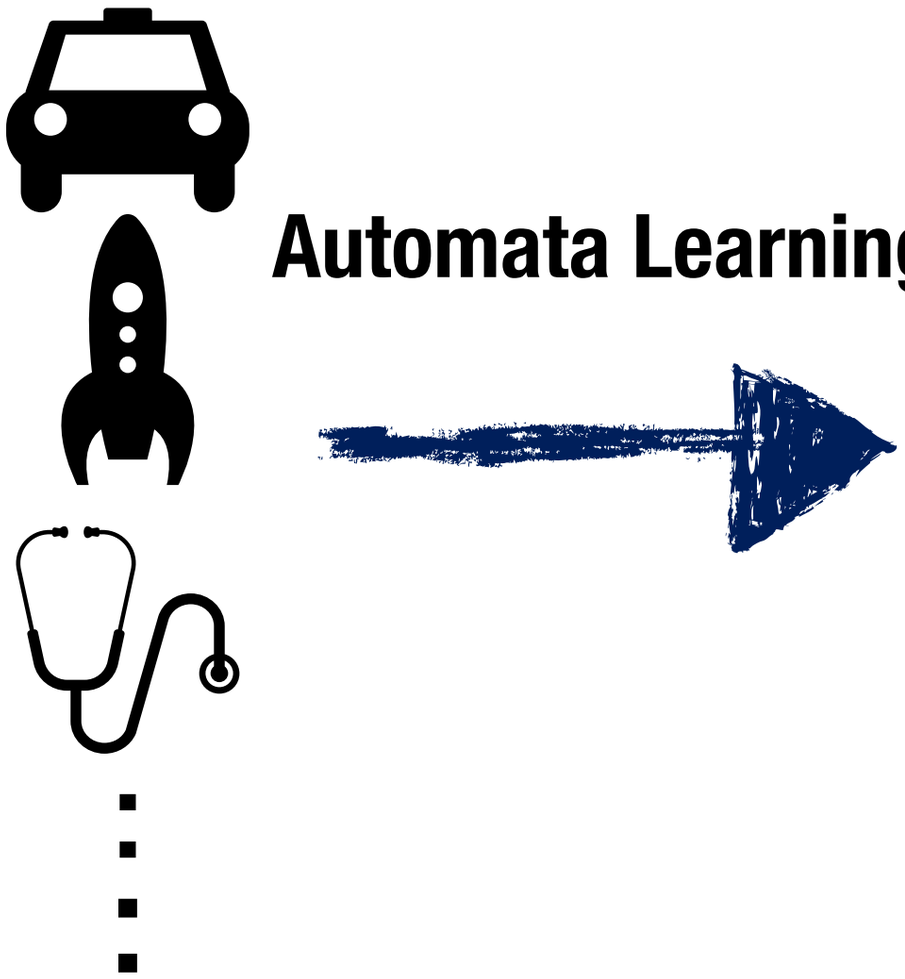
Approach 3: Automata learning → formal verification



Black-Box Checking (BBC)

[Peled et al., FORTE'99]

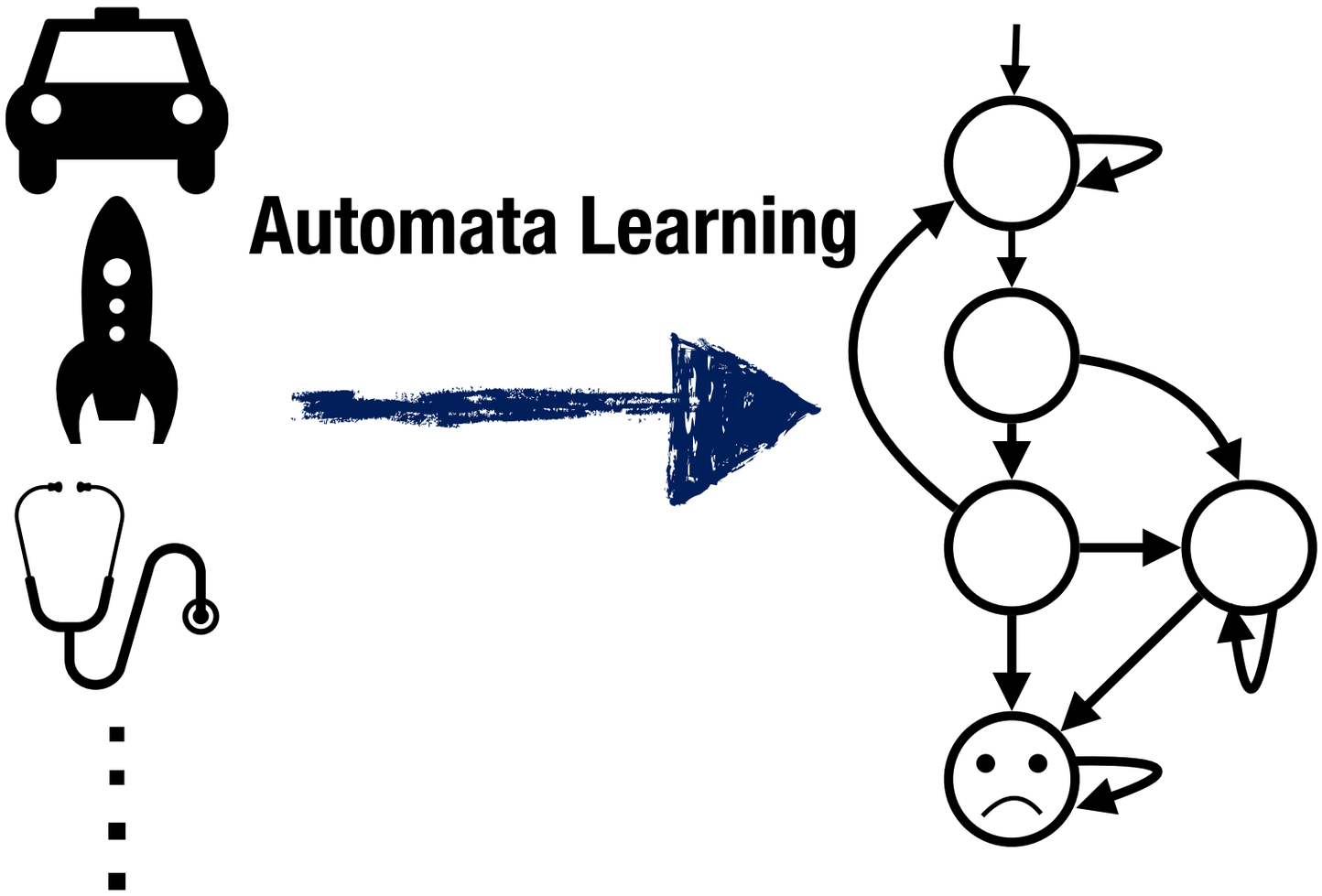
Approach 3: Automata learning → formal verification



Black-Box Checking (BBC)

[Peled et al., FORTE'99]

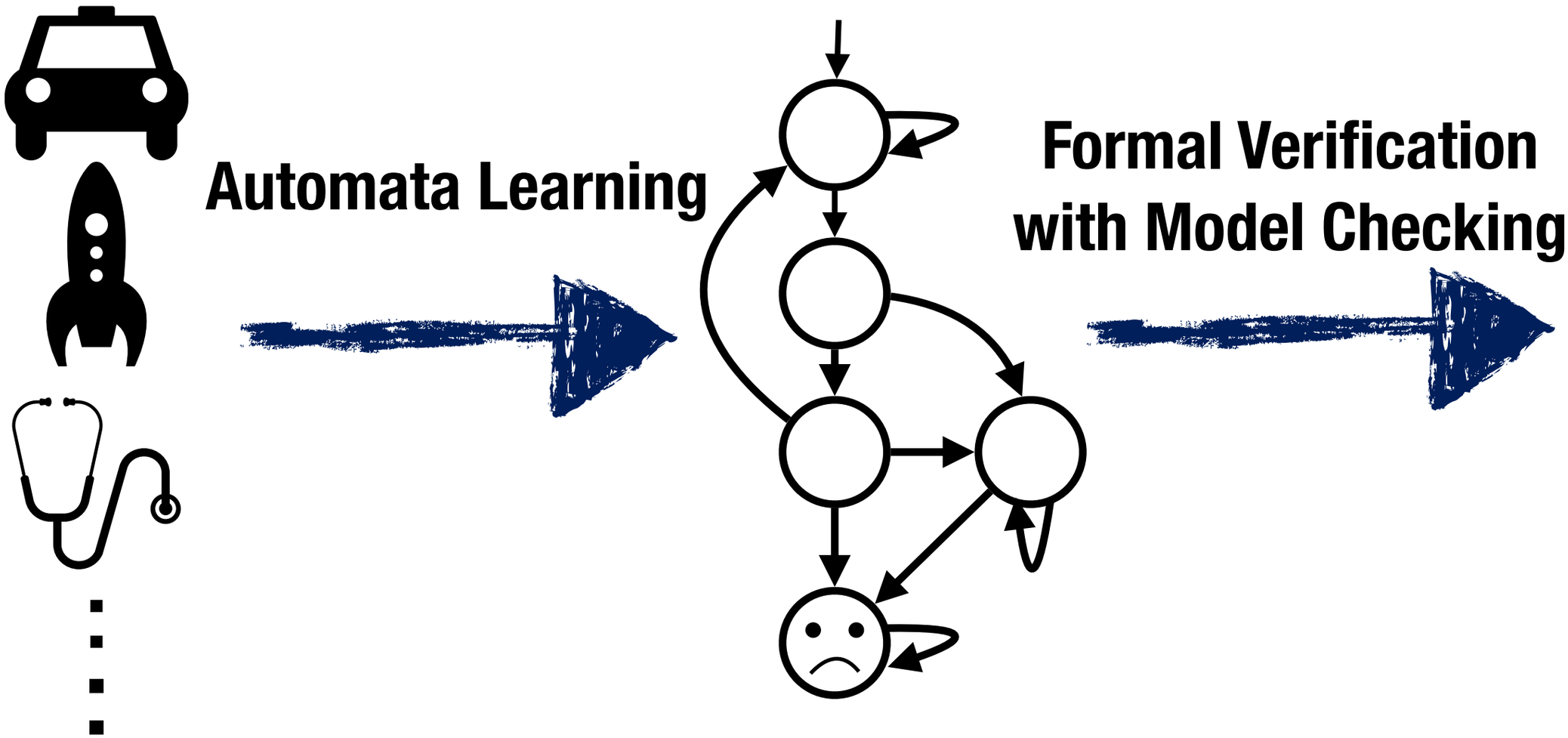
Approach 3: Automata learning → formal verification



Black-Box Checking (BBC)

[Peled et al., FORTE'99]

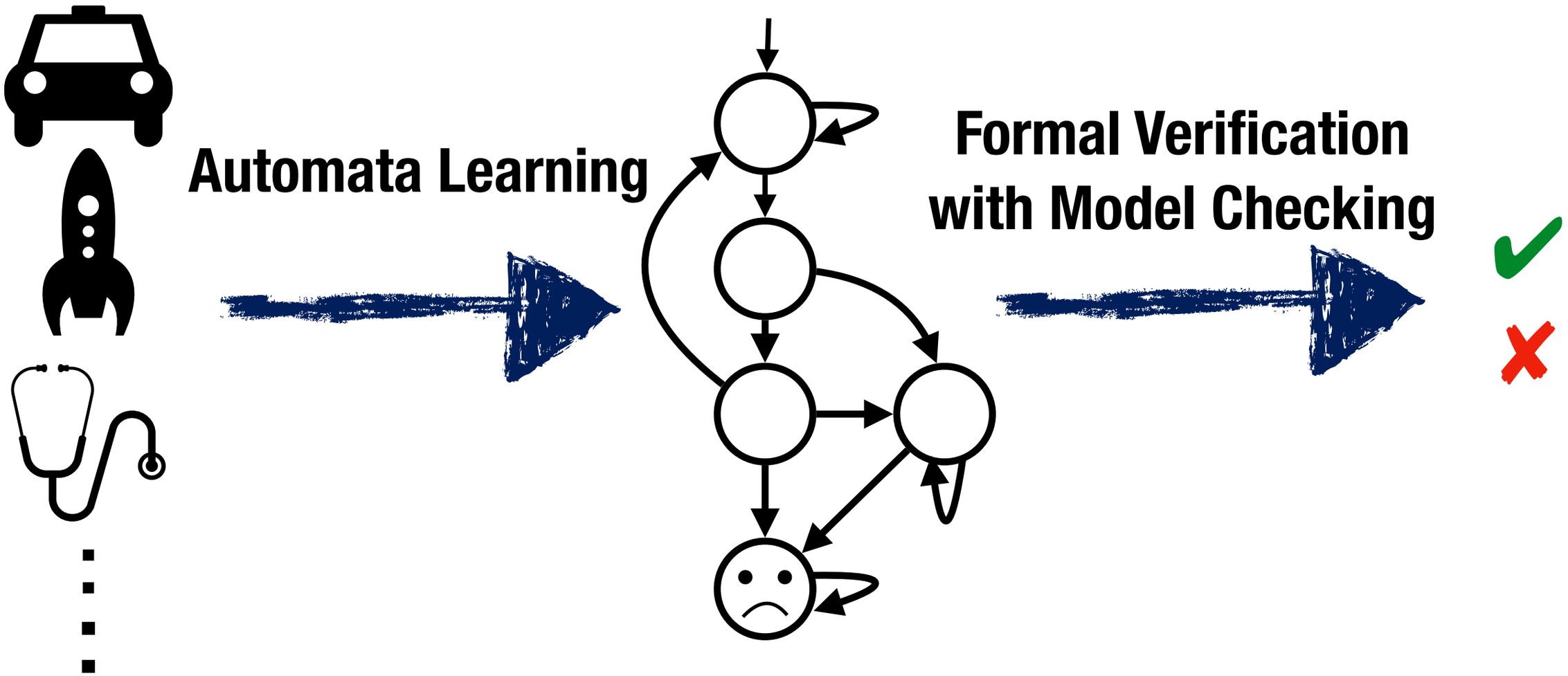
Approach 3: Automata learning → formal verification



Black-Box Checking (BBC)

[Peled et al., FORTE'99]

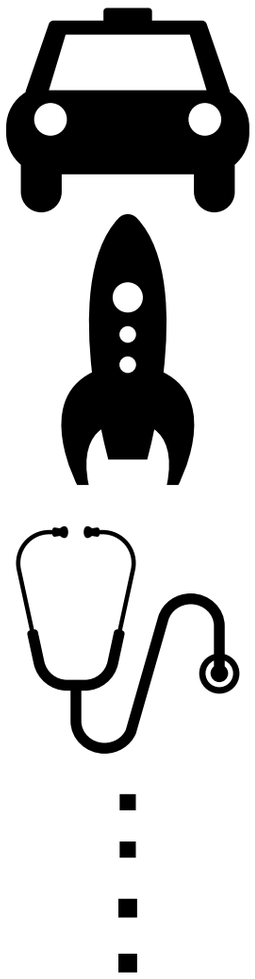
Approach 3: Automata learning → formal verification



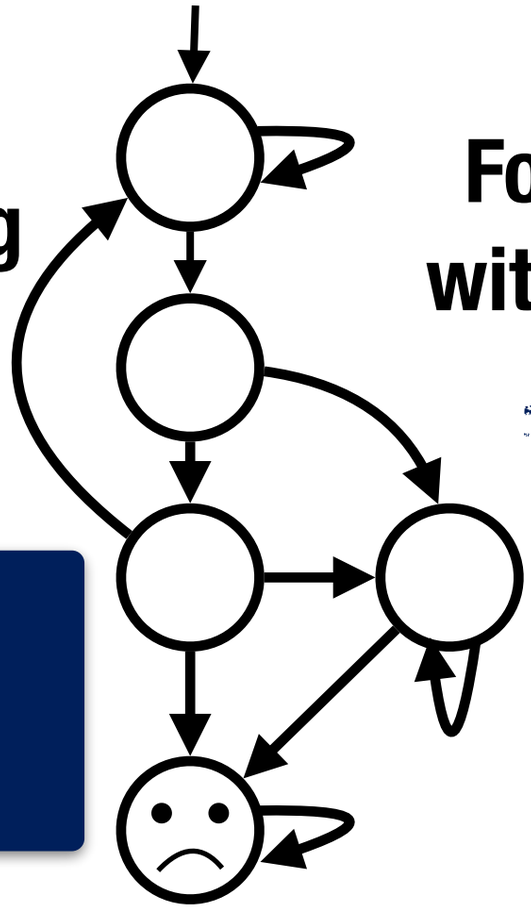
Black-Box Checking (BBC)

[Peled et al., FORTE'99]

Approach 3: Automata learning → formal verification



Automata Learning



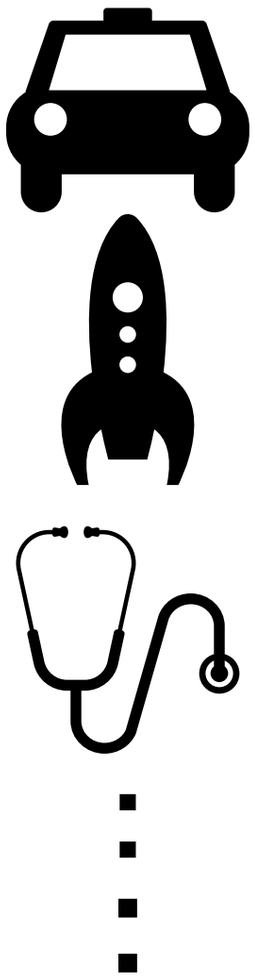
Formal Verification with Model Checking

Good theoretical properties, e.g., convergence

Black-Box Checking (BBC)

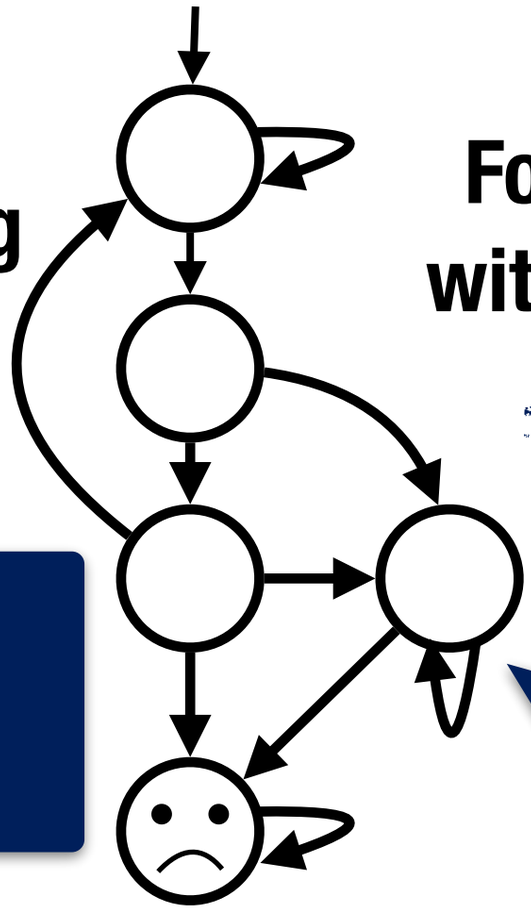
[Peled et al., FORTE'99]

Approach 3: Automata learning → formal verification



Automata Learning

Good theoretical properties, e.g., convergence



Formal Verification with Model Checking

✔

✘

Reusable for multiple requirements

Summary of This Talk

- Techniques for efficient black-box checking
  - Robustness-guided equivalence testing
  - Model checking of strengthened formulas
    - Specification-guided abstraction of system under test
- Java Toolkit FalCAuN for black-box checking
- Probabilistic extension of black-box checking

Outline

- Techniques for efficient black-box checking
  - Robustness-guided equivalence testing
  - Model checking of strengthened formulas
  - Specification-guided abstraction of system under test
- Java Toolkit FalCAuN for black-box checking
- Probabilistic extension of black-box checking

Outline

- Preliminaries
  - Active automata learning
  - Black-box checking
- Techniques for efficient black-box checking
  - Robustness-guided equivalence testing
  - Model checking of strengthened formulas
  - Specification-guided abstraction of system under test
- Java Toolkit FalCAuN for black-box checking
- Probabilistic extension of black-box checking

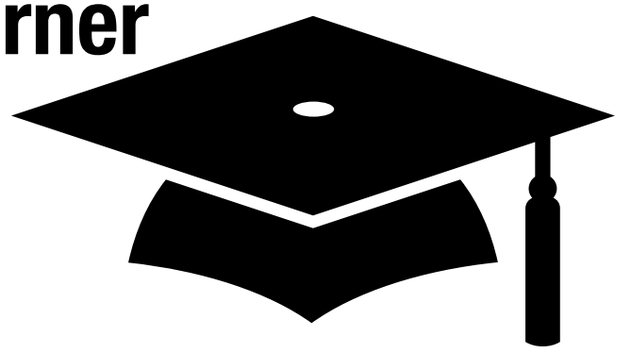
Active Automata Learning

[Angluin, Inf. Comput.'87]

Active Automata Learning

[Angluin, Inf. Comput.'87]

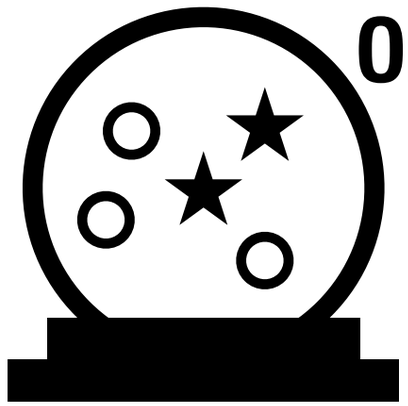
Learner



Active Automata Learning

[Angluin, Inf. Comput.'87]

Learner



Oracle

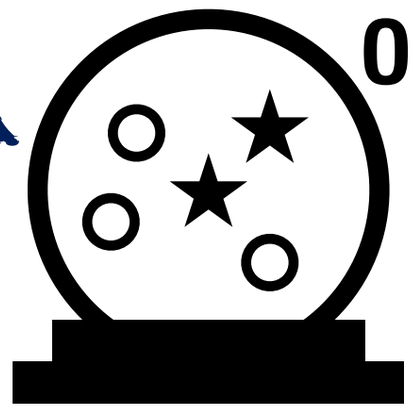
Active Automata Learning

[Angluin, Inf. Comput.'87]

Questions: $w \overset{?}{\in} L_{\text{target}}$, $L(\mathscr{A}_{\text{hyp}}) \overset{?}{=} L_{\text{target}}$

Learner

Oracle



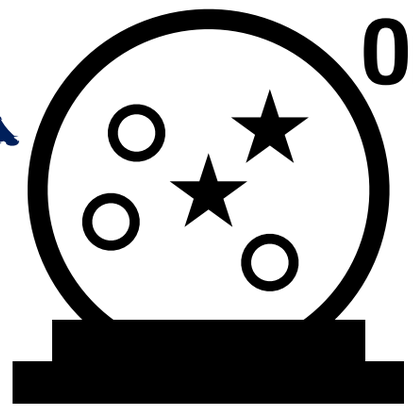
Active Automata Learning

[Angluin, Inf. Comput.'87]

Membership

Questions: $w \overset{?}{\in} L_{\text{target}}$, $L(\mathscr{A}_{\text{hyp}}) \overset{?}{=} L_{\text{target}}$

Learner



Oracle

Active Automata Learning

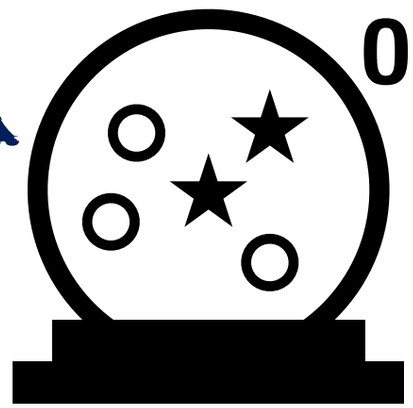
[Angluin, Inf. Comput.'87]

Membership

Equivalence

Questions: $w \overset{?}{\in} L_{\text{target}}$, $L(\mathscr{A}_{\text{hyp}}) \overset{?}{=} L_{\text{target}}$

Learner



Oracle

Active Automata Learning

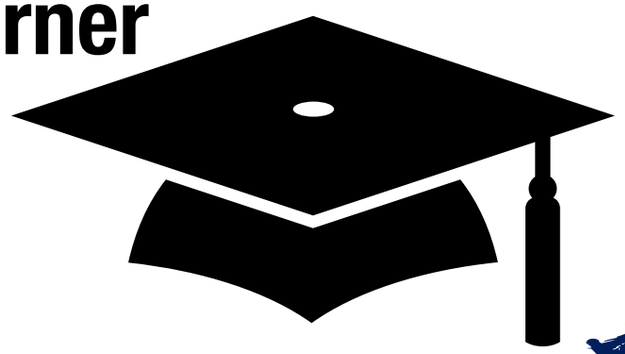
[Angluin, Inf. Comput.'87]

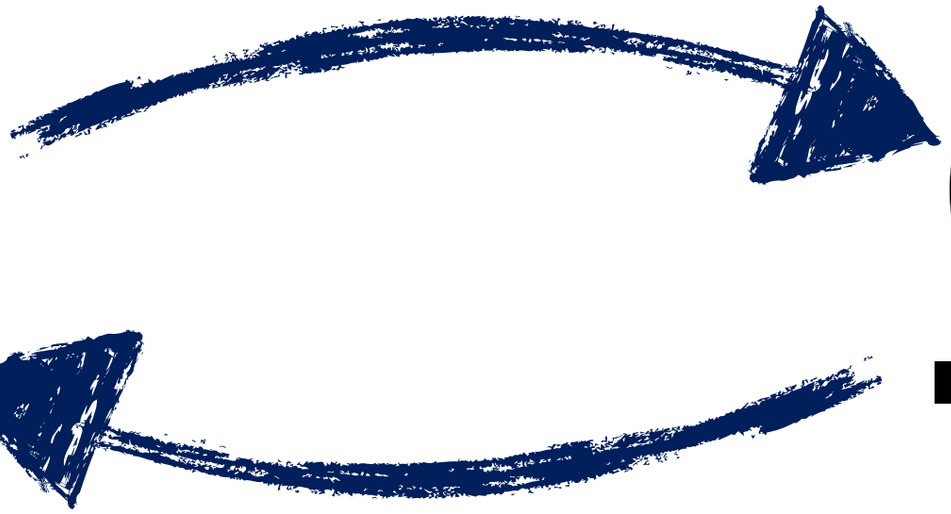
Membership

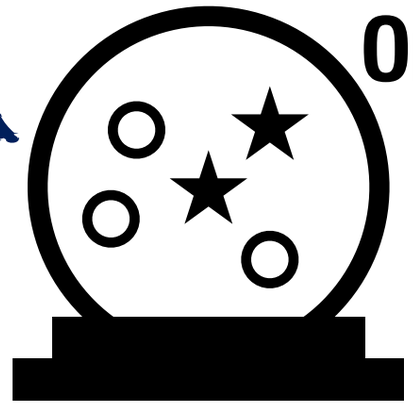
Equivalence

Questions: $w \overset{?}{\in} L_{\text{target}}$, $L(\mathscr{A}_{\text{hyp}}) \overset{?}{=} L_{\text{target}}$

Learner







Oracle

Answers: Yes, No, Evidence $w \in L(\mathscr{A}_{\text{hyp}}) \triangle L_{\text{target}}$

Active Automata Learning

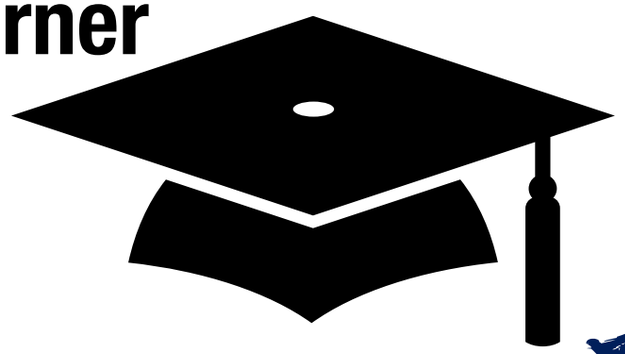
[Angluin, Inf. Comput.'87]

Membership

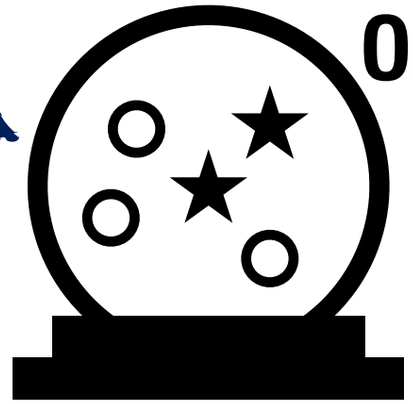
Equivalence

Questions: $w \overset{?}{\in} L_{\text{target}}$, $L(\mathscr{A}_{\text{hyp}}) \overset{?}{=} L_{\text{target}}$

Learner

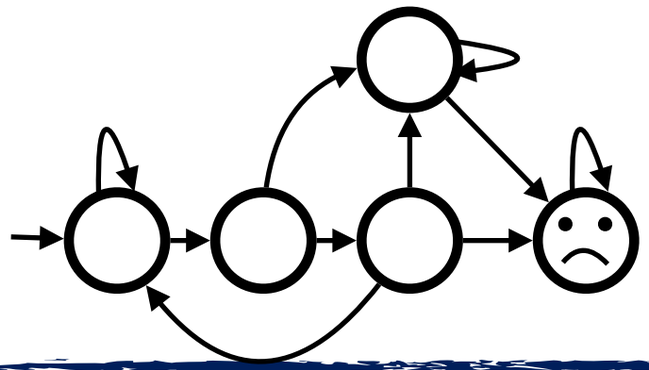


Oracle



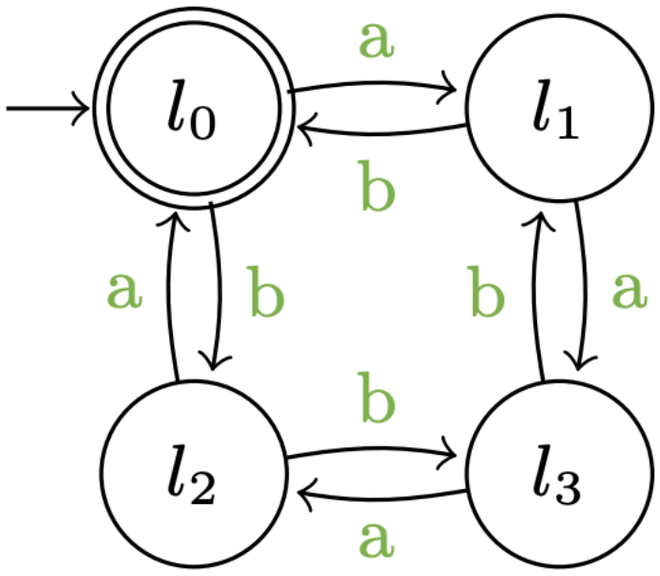
Answers: Yes, No, Evidence $w \in L(\mathscr{A}_{\text{hyp}}) \triangle L_{\text{target}}$

is modeled by



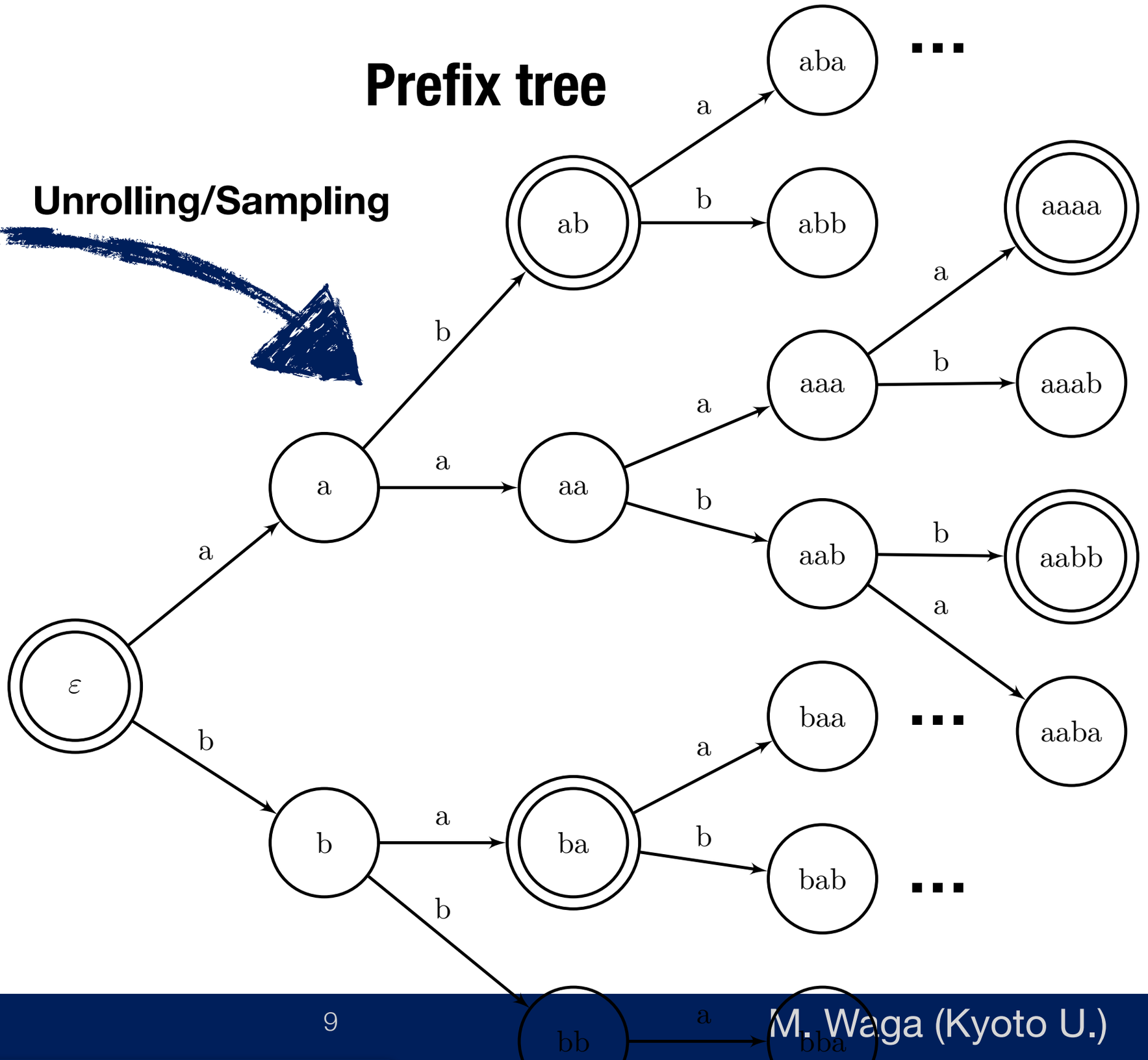
Idea: Find Good Prefixes/Suffixes

Target DFA



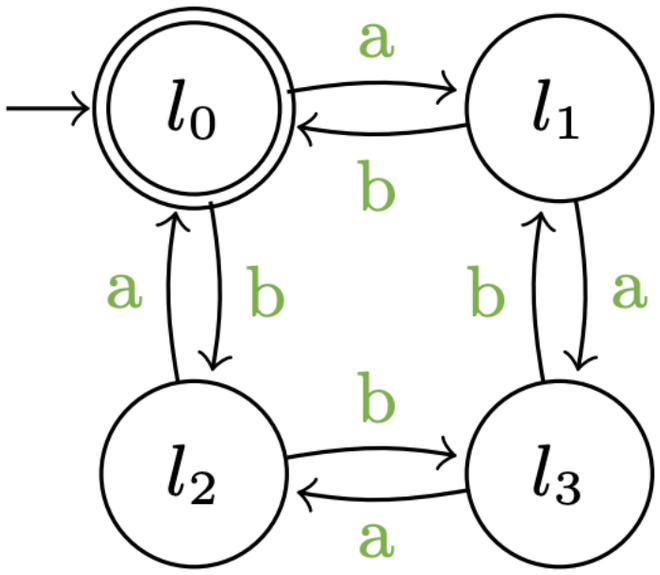
Unrolling/Sampling

Prefix tree



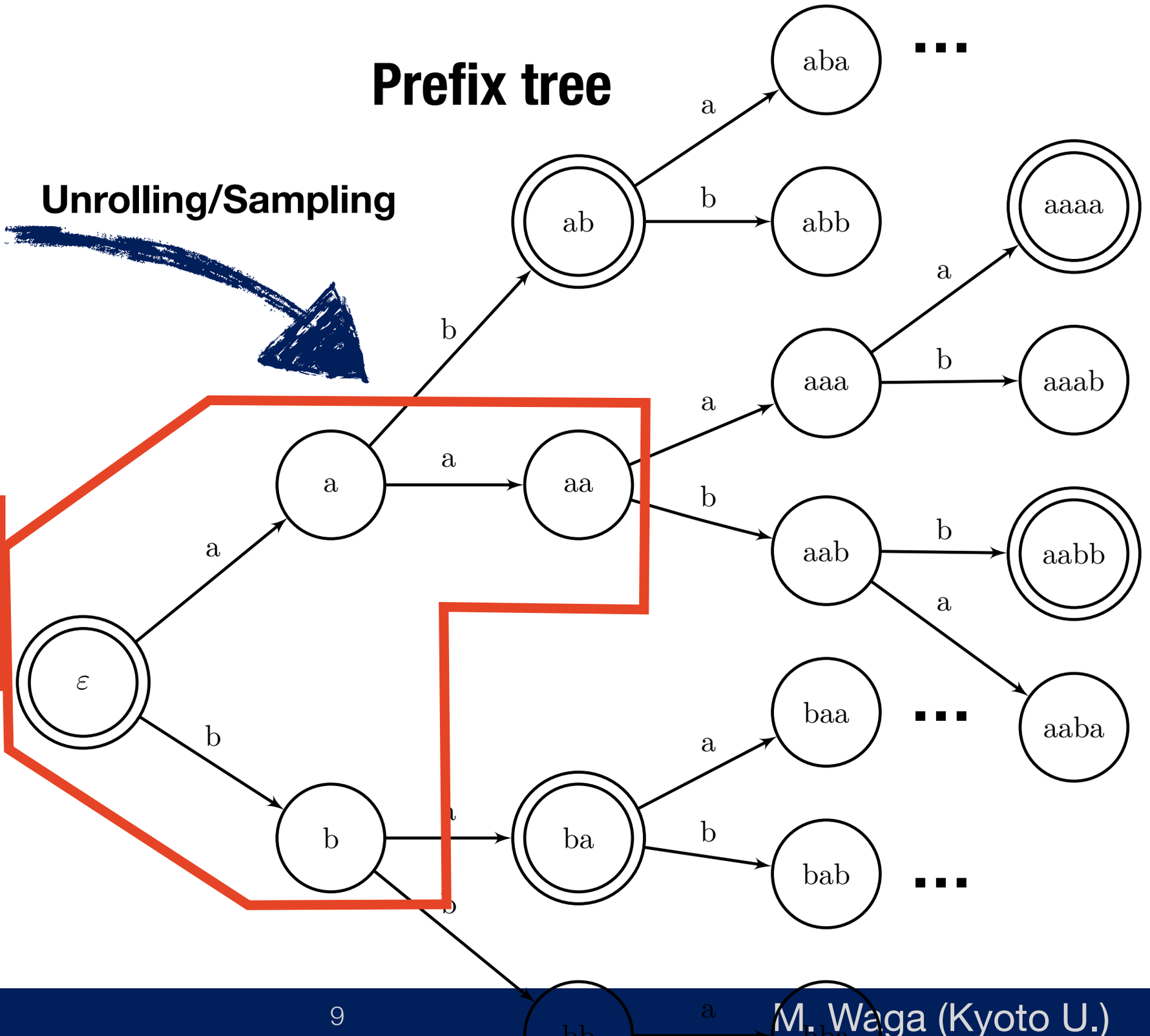
Idea: Find Good Prefixes/Suffixes

Target DFA



Unrolling/Sampling

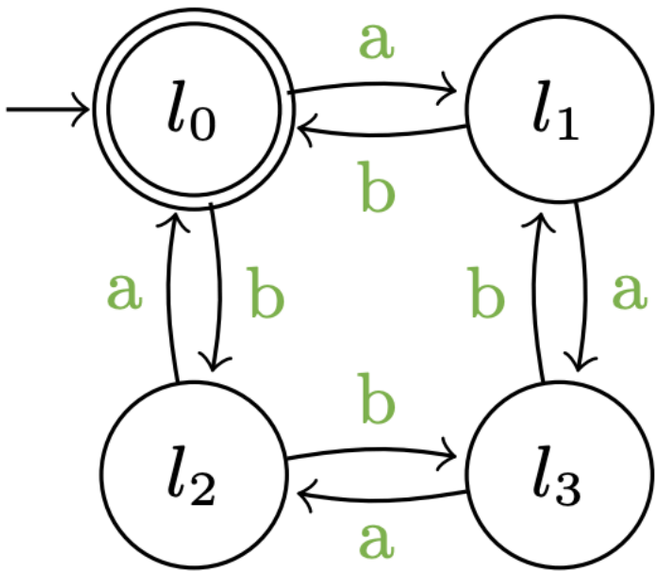
Prefix tree



Prefixes P to cover states

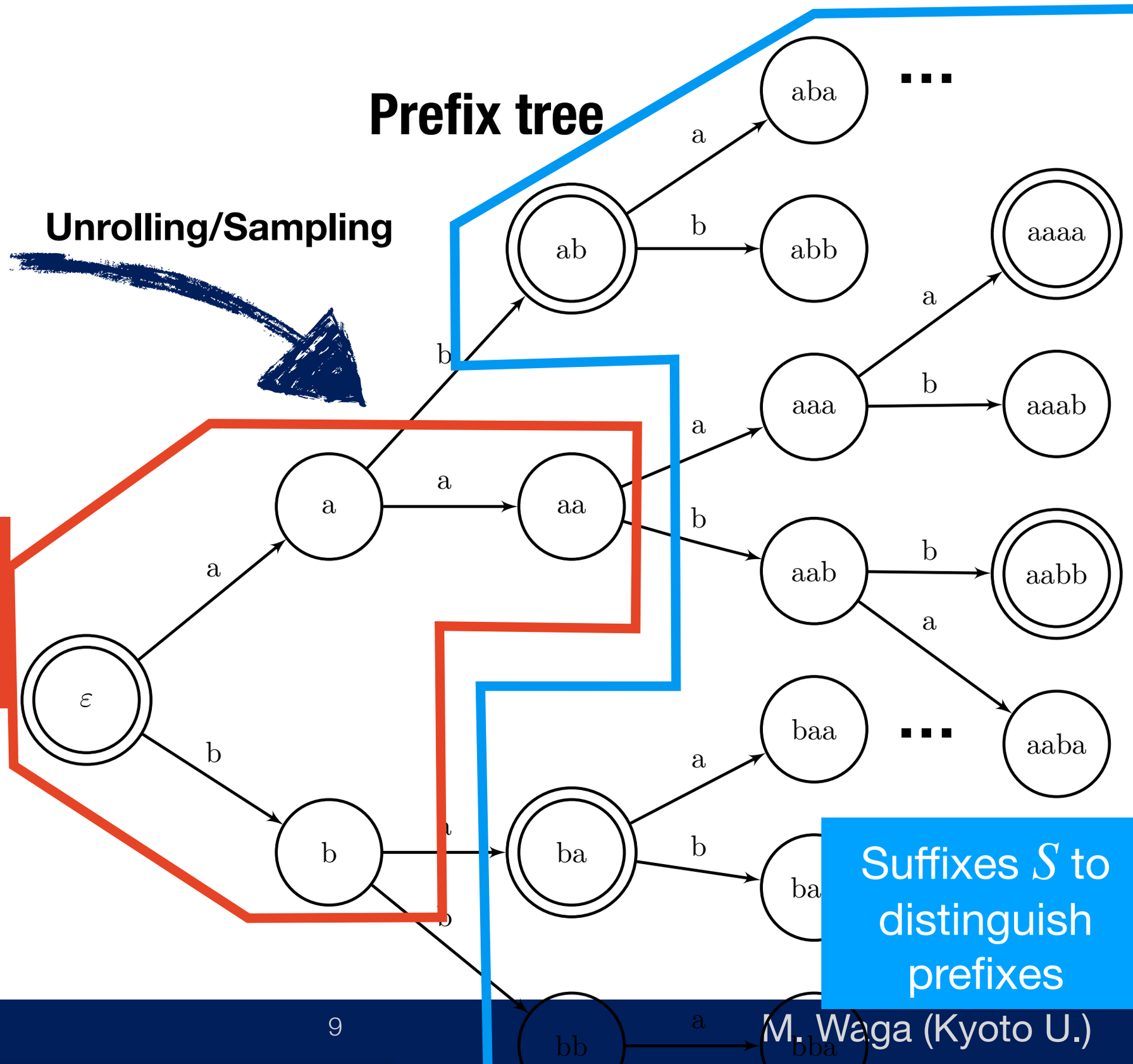
Idea: Find Good Prefixes/Suffixes

Target DFA



Unrolling/Sampling

Prefix tree



Prefixes P to cover states

Suffixes S to distinguish prefixes

Outline of Active Automata Learning

1. Initialize $P \leftarrow \{\varepsilon\}, S \leftarrow \{\varepsilon\}$

2. Increase P and S to satisfy certain conditions,
   e.g., all the successors are constructed

3. Construct a hypothesis DFA $\mathscr{A}_{\mathrm{hyp}}$ and ask eq. query
   → if $L(\mathscr{A}_{\mathrm{hyp}}) = L_{\mathrm{tgt}}$, the learning finishes

4. Refine P and S using $w \in L(\mathscr{A}_{\mathrm{hyp}}) \triangle L_{\mathrm{tgt}}$
   → go back to 2.

Black-Box Checking (BBC)

[Peled et al., FORTE'99]

Inputs

Deterministic

- Black-box system under test $\mathscr{M} : \Sigma^\infty \to (2^{\mathrm{AP}})^\infty$
  - $\forall \sigma \in \Sigma^\infty.\ |\sigma| = |\mathscr{M}(\sigma)|$

Finite or infinite sequence

  - $\mathscr{M}$ preserves prefixes: $\forall w \in \Sigma^\omega, i.\ \mathscr{M}(w|_{[1,\ldots,i]}) = \mathscr{M}(w)|_{[1,\ldots,i]}$
- (Safety) ω regular language $\varphi \subseteq (2^{\mathrm{AP}})^\omega$

Output (informal): One of the following

- "Likely $\mathscr{M} \models \varphi$" + Mealy machine $\mathscr{M}'$ approximating $\mathscr{M}$ s.t. $\forall \sigma \in \Sigma^\omega.\ \mathscr{M}'(\sigma) \in \varphi$
- $\sigma \in \Sigma^*$ s.t. $\mathscr{M}(\sigma) \in (2^{\mathrm{AP}})^*$ witnesses the violation of φ

Black-Box Checking (BBC)

[Peled et al., FORTE'99]

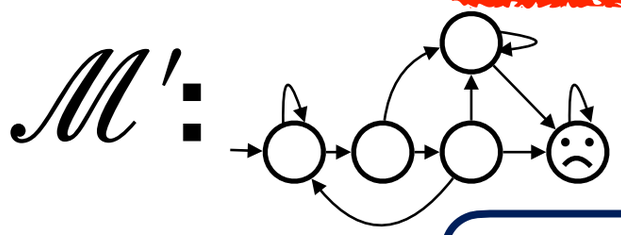
$\mathscr{M}$: 

φ: Spec.

Found σ s.t.
$\mathscr{M}(\sigma) \neq \mathscr{M}'(\sigma)$

✔
→ $\mathscr{M}(\sigma) \neq \mathscr{M}'(\sigma)$

Active Automata Learning, e.g. L*

$\mathscr{M}'$: 

Verify $\mathscr{M}'$ against φ

Find evidence of $\mathscr{M} \neq \mathscr{M}'$

Test $\mathscr{M}$ w/ input σ

Not Found ✔

✘ w/ input σ

✘

Likely ✔ (φ seems satisfied)

✘ (φ is violated)

Active Learning of Mealy Machine $\mathscr{M}'$

1. Initialize $P \leftarrow \{\varepsilon\}, S \leftarrow \{\varepsilon\}$

2. Increase P and S to satisfy certain conditions,
   e.g., all the successors are constructed

3. Construct a Mealy machine $\mathscr{M}'$ and ask eq. query
   → if $\forall w \in \Sigma^* \,.\, \mathscr{M}(w) = \mathscr{M}'(w)$, learning finishes

4. Refine P and S using $\sigma \in \Sigma^*$ s.t. $\mathscr{M}(\sigma) \neq \mathscr{M}'(\sigma)$
   → go back to 2

Construction of Mealy Machine $\mathscr{M}'$ in BBC

1. Initialize $P \leftarrow \{\varepsilon\}, S \leftarrow \{\varepsilon\}$

2. Increase P and S to satisfy certain conditions, e.g., all the successors are constructed

3. Construct a Mealy machine $\mathscr{M}'$ and ask eq. query
   → if ~~$\forall w \in \Sigma^* : \mathscr{M}(w) = \mathscr{M}'(w)$, learning finishes~~

4. Refine P and S using $\sigma \in \Sigma^*$ s.t. $\mathscr{M}(\sigma) \neq \mathscr{M}'(\sigma)$
   → go back to 2

Used for approx. autom. construction

Black-Box Checking (BBC)

[Peled et al., FORTE'99]

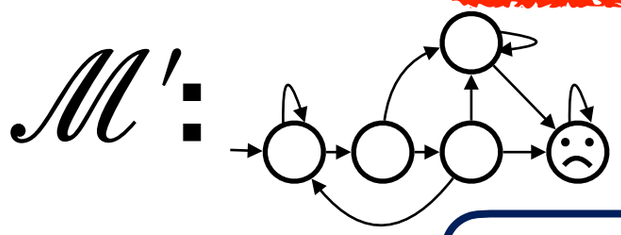
$\mathscr{M}$: 

φ: Spec.

Found σ s.t.
$\mathscr{M}(\sigma) \neq \mathscr{M}'(\sigma)$

✔
→ $\mathscr{M}(\sigma) \neq \mathscr{M}'(\sigma)$

Active Automata Learning, e.g. L*

$\mathscr{M}'$: 

Verify $\mathscr{M}'$ against φ

Find evidence of $\mathscr{M} \neq \mathscr{M}'$

Test $\mathscr{M}$ w/ input σ

Not Found ✔

✘ w/ input σ

✘

Likely ✔ (φ seems satisfied)

✘ (φ is violated)

Black-Box Checking (BBC)

[Peled et al., FORTE'99]

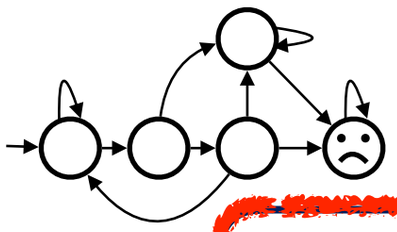
$\mathscr{M}$: 

φ: Spec.

Found σ s.t.
$\mathscr{M}(\sigma) \neq \mathscr{M}'(\sigma)$

✔
→ $\mathscr{M}(\sigma) \neq \mathscr{M}'(\sigma)$

Active Automata Learning, e.g. L*

$\mathscr{M}'$: 

Verify $\mathscr{M}'$ against φ

Find evidence of $\mathscr{M} \neq \mathscr{M}'$

Test $\mathscr{M}$ w/ input σ

Not Found ✔

✘ w/ input σ

✘

Likely ✔ (φ seems satisfied)

✘ (φ is violated)

Model Checking in BBC

We want a finite witness of violation
→ focus on safety properties

Properties with finite witness of violation

Def (safety property).
An ω-regular property φ represents a *safety property* if for any $w \in (2^{\mathrm{AP}})^{\omega}$ violating φ, there is a prefix $u \in (2^{\mathrm{AP}})^{*}$ of w such that for any $v \in (2^{\mathrm{AP}})^{\omega}$, $u \cdot v$ also violates φ.

Examples (in LTL): $\Box p$, $\Box (p \rightarrow \bigcirc q)$, and $pU(q \vee G \Box p)$

Assumption: Model checkers return a finite witness of a violation

Black-Box Checking (BBC)

[Peled et al., FORTE'99]

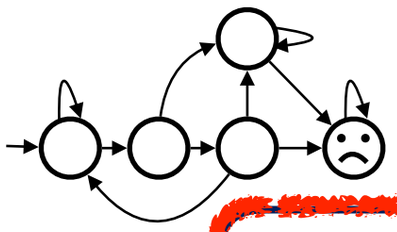
$\mathscr{M}$: 

φ: Spec.

Found σ s.t.
$\mathscr{M}(\sigma) \neq \mathscr{M}'(\sigma)$

✔
→ $\mathscr{M}(\sigma) \neq \mathscr{M}'(\sigma)$

Active Automata Learning, e.g. L*

$\mathscr{M}'$: 

Verify $\mathscr{M}'$ against φ

Find evidence of $\mathscr{M} \neq \mathscr{M}'$

Test $\mathscr{M}$ w/ input σ

Not Found ✔

✘ w/ input σ

✘

Likely ✔ (φ seems satisfied)

✘ (φ is violated)

Black-Box Checking (BBC)

[Peled et al., FORTE'99]

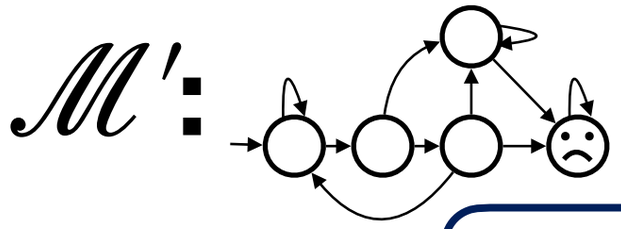
$\mathscr{M}$: 

φ: Spec.

Found σ s.t.
$\mathscr{M}(\sigma) \neq \mathscr{M}'(\sigma)$

✔
→ $\mathscr{M}(\sigma) \neq \mathscr{M}'(\sigma)$

Active Automata Learning, e.g. L*

$\mathscr{M}'$: 

Verify $\mathscr{M}'$ against φ

Find evidence of $\mathscr{M} \neq \mathscr{M}'$

Test $\mathscr{M}$ w/ input σ

Not Found ✔

✘ w/ input σ

✘

Likely ✔ (φ seems satisfied)

✘ (φ is violated)

Test $\mathscr{M}$ with the Witness σ

- Model checker returns $\sigma \in \Sigma^*$ s.t. $\forall u \in (2^{\mathrm{AP}})^\omega$. $\mathscr{M}'(\sigma) \cdot u \notin \varphi$
- We check if we also have $\forall u \in (2^{\mathrm{AP}})^\omega$. $\mathscr{M}(\sigma) \cdot u \notin \varphi$
  - **Note**: We can do this because φ is safety and ω regular
  - Yes → we finish the testing
  - No → σ is an evidence of $\mathscr{M} \neq \mathscr{M}'$
- We use σ to refine P and S if we have $\mathscr{M}(\sigma) \neq \mathscr{M}'(\sigma)$

Black-Box Checking (BBC)

[Peled et al., FORTE'99]

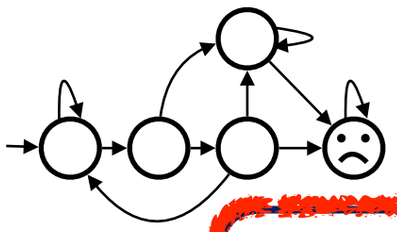
$\mathscr{M}$: 

φ: Spec.

Found σ s.t.
$\mathscr{M}(\sigma) \neq \mathscr{M}'(\sigma)$

✔
→ $\mathscr{M}(\sigma) \neq \mathscr{M}'(\sigma)$

Active Automata Learning, e.g. L*

$\mathscr{M}'$: 

Verify $\mathscr{M}'$ against φ

Find evidence of $\mathscr{M} \neq \mathscr{M}'$

Test $\mathscr{M}$ w/ input σ

Not Found ✔

✘ w/ input σ

✘

Likely ✔ (φ seems satisfied)

✘ (φ is violated)

Black-Box Checking (BBC)

[Peled et al., FORTE'99]

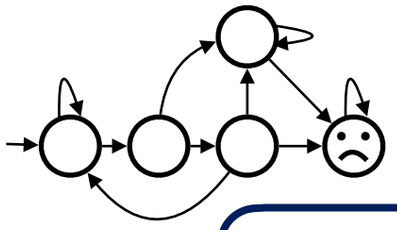
$\mathscr{M}$: 

φ: Spec.

Found σ s.t.
$\mathscr{M}(\sigma) \neq \mathscr{M}'(\sigma)$

✔
→ $\mathscr{M}(\sigma) \neq \mathscr{M}'(\sigma)$

Active Automata Learning, e.g. L*

$\mathscr{M}'$: 

Verify $\mathscr{M}'$ against φ

Find evidence of $\mathscr{M} \neq \mathscr{M}'$

Test $\mathscr{M}$ w/ input σ

Not Found ✔

✘ w/ input σ

✘

Likely ✔ (φ seems satisfied)

✘ (φ is violated)

Equivalence Testing

Goal: find $\sigma \in \Sigma^*$ s.t. $\mathscr{M}(\sigma) \neq \mathscr{M}'(\sigma)$ to refine S

Theoretically: Automata-based conformance test
e.g., W-method [Chow, TSE'78], Wp-method [Fujiwara+, TSE'91]

- 😀 Equivalence can be guaranteed
- ☹️ # of states is necessary for soundness

Practically: Random sampling of σ

$\mathscr{M}$ and $\mathscr{M}'$ are likely equivalent for most of σ

- 😀 PAC guarantee can be obtained [Angluin, Inf. Comput. 87]
- ☹️ Not good at "rare" cex

Black-Box Checking (BBC)

[Peled et al., FORTE'99]

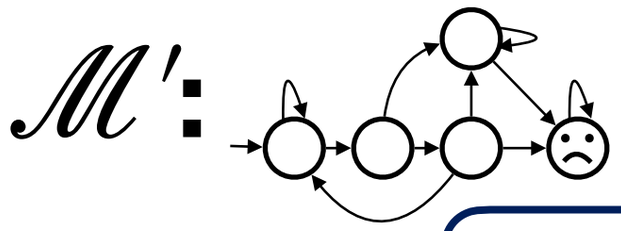
$\mathscr{M}$: 

φ: Spec.

Found σ s.t.
$\mathscr{M}(\sigma) \neq \mathscr{M}'(\sigma)$

✔
→ $\mathscr{M}(\sigma) \neq \mathscr{M}'(\sigma)$

Active Automata Learning, e.g. L*

$\mathscr{M}'$: 

Verify $\mathscr{M}'$ against φ

Find evidence of $\mathscr{M} \neq \mathscr{M}'$

Test $\mathscr{M}$ w/ input σ

Not Found ✔

✘ w/ input σ

✘

Likely ✔ (φ seems satisfied)

✘ (φ is violated)

Black-Box Checking (BBC)

[Peled et al., FORTE'99]

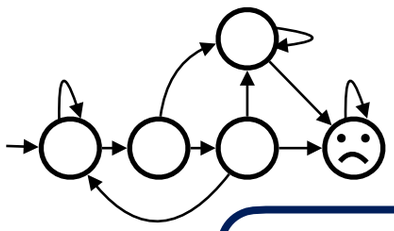
$\mathscr{M}$: 

φ: Spec.

Found σ s.t.
$\mathscr{M}(\sigma) \neq \mathscr{M}'(\sigma)$

✔
→ $\mathscr{M}(\sigma) \neq \mathscr{M}'(\sigma)$

Active Automata Learning, e.g. L*

$\mathscr{M}'$: 

Verify $\mathscr{M}'$ against φ

Difficult

Find evidence of $\mathscr{M} \neq \mathscr{M}'$

Test $\mathscr{M}$ w/ input σ

Not Found ✔

✘ w/ input σ

✘

Likely ✔ (φ seems satisfied)

✘ (φ is violated)

Outline

- Preliminaries
  - Active automata learning
  - Black-box checking
- Techniques for efficient black-box checking
  - Robustness-guided equivalence testing
  - Model checking of strengthened formulas
  - Specification-guided abstraction of system under test
- Java Toolkit FalCAuN for black-box checking
- Probabilistic extension of black-box checking

Equivalence Test Dedicated to BBC?

Eq. test for general automata learning: test overall equivalence

- **Conformance test**: Ensure the equivalence for all the transitions/states

assuming the size of $\mathscr{M}$

- **Random test**: Ensure the equivalence for many input words

Observation: BBC's goal is to show that $\mathscr{M}$ violates φ

→ How about focusing on "less safe" parts in eq. test?

Refinement of $\mathscr{M}'$ focusing on "less safe" parts of $\mathscr{M}$

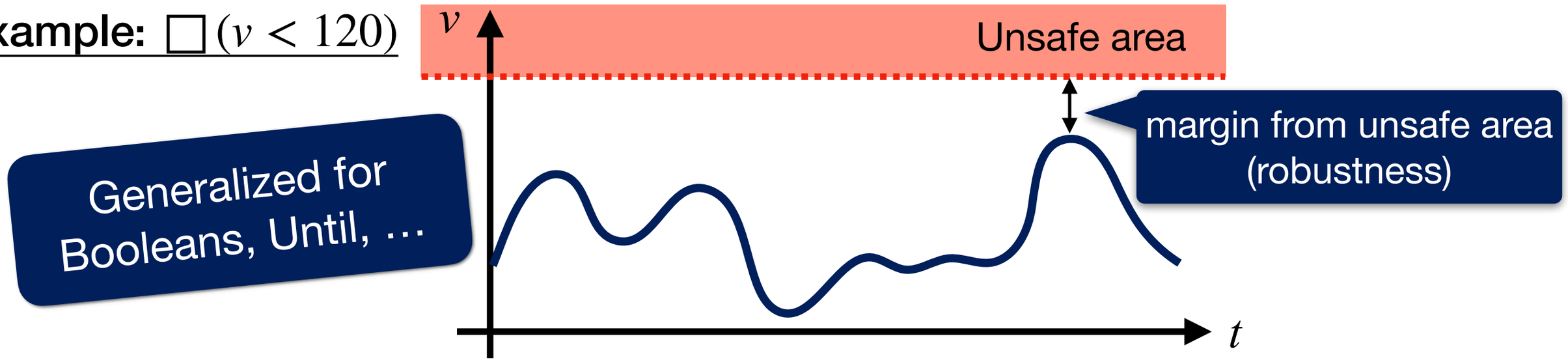
Robustness-Guided Equivalence Test

[Waga, HSCC'20]

Idea: sample "less safe" inputs by minimizing robustness

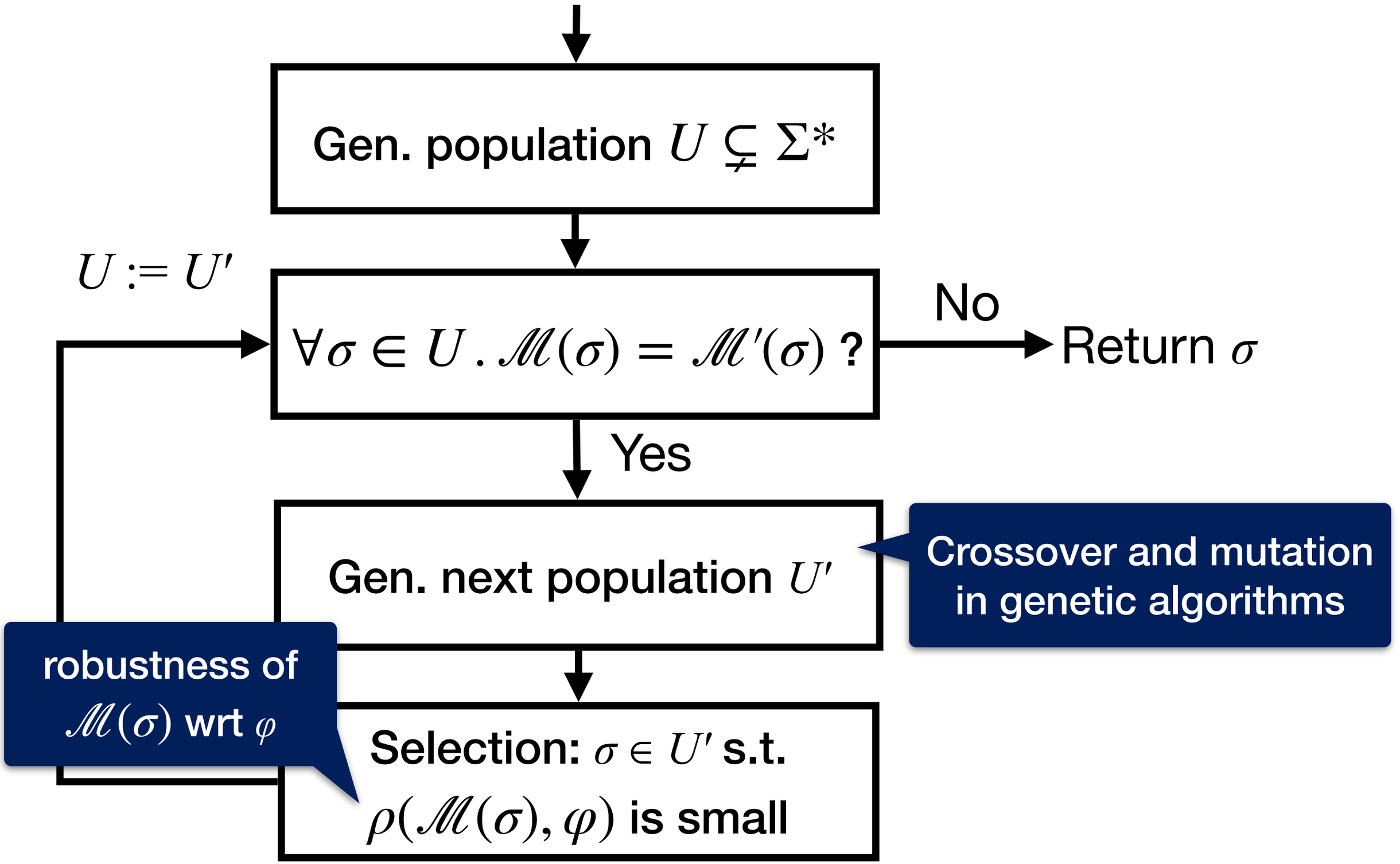
Robustness [Fainekos & Pappas, TCS'09]: "distance" from unsafe area

Example: $\square(v < 120)$



- Sample $\sigma \in \Sigma^*$ with small robustness in equivalence test
  - Black-box optimization can be used
  - e.g. genetic algorithms

Robustness-Guided Equivalence Test

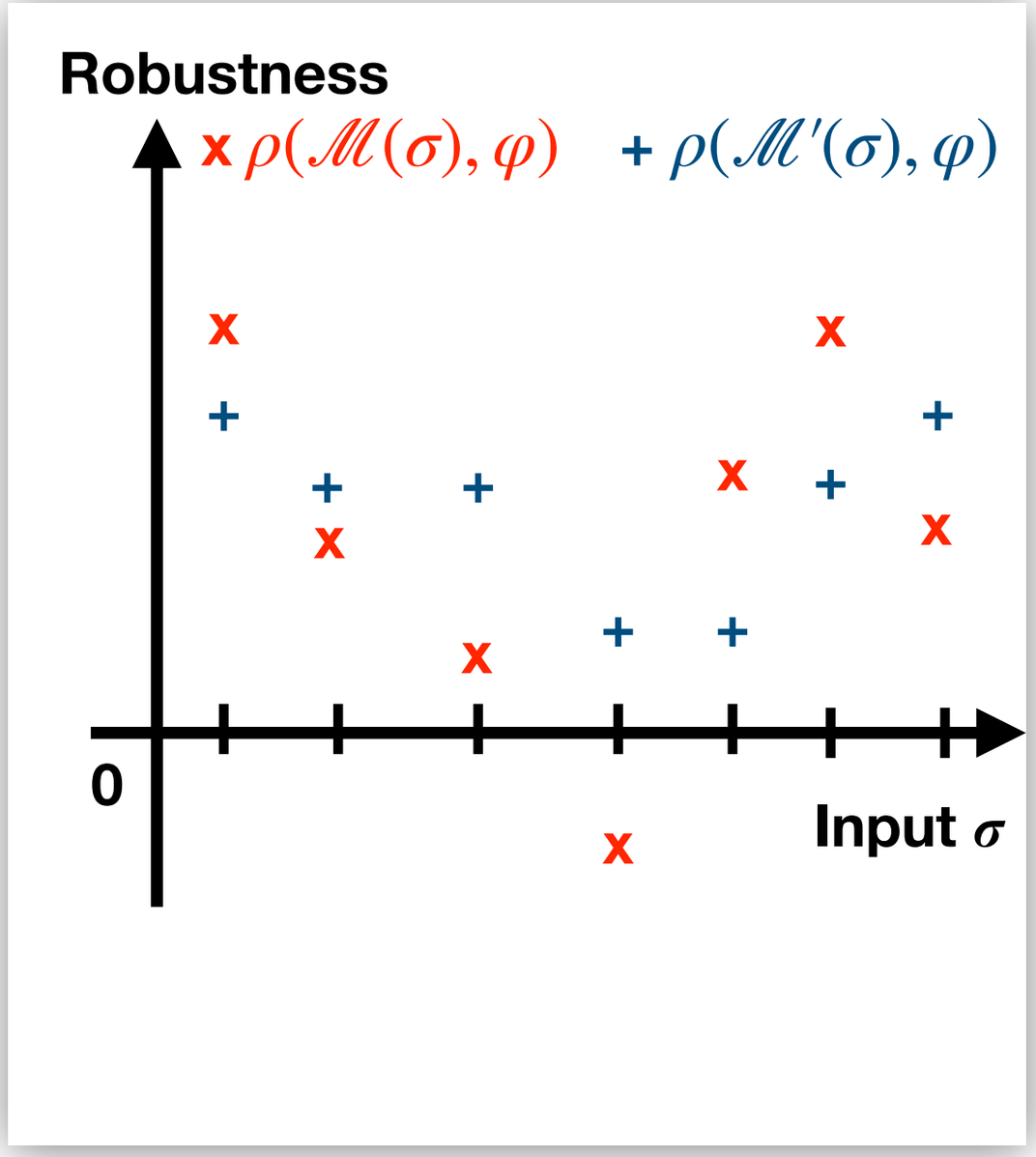


Why $\rho(\mathscr{M}(\sigma), \varphi)$ as Fitness?

Assumption: $\mathscr{M} \not\models \varphi$
i.e. $\exists \sigma . \rho(\mathscr{M}(\sigma), \varphi) \leq 0$

Fact: $\mathscr{M}' \models \varphi$ i.e.
$\forall \sigma . \rho(\mathscr{M}'(\sigma), \varphi) \geq 0$
by model checking

Heuristic: Find σ s.t.
$\mathscr{M}(\sigma) \neq \mathscr{M}'(\sigma)$ in
$\rho(\mathscr{M}(\sigma), \varphi)$ is small

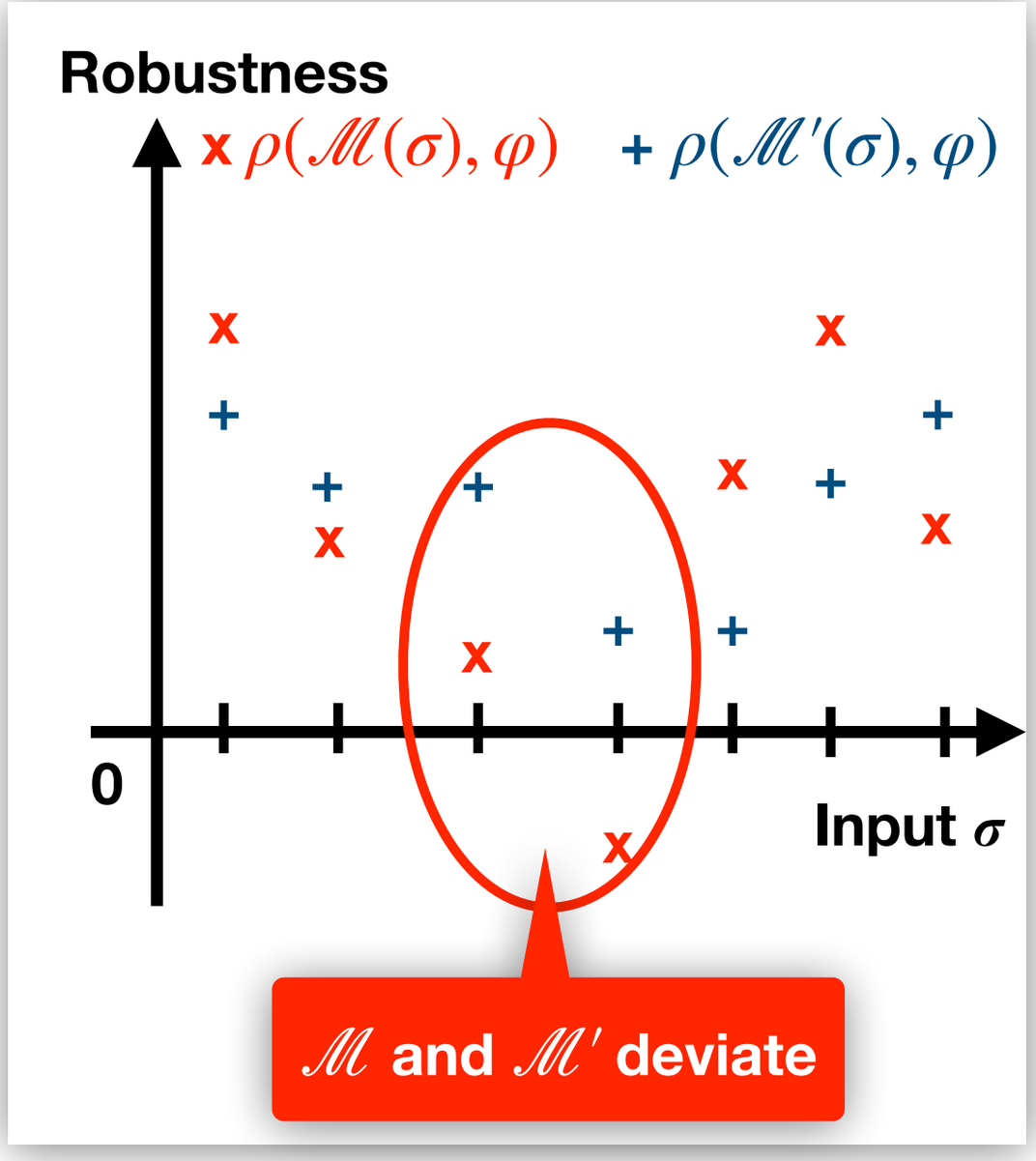


Why $\rho(\mathscr{M}(\sigma), \varphi)$ as Fitness?

Assumption: $\mathscr{M} \not\models \varphi$
i.e. $\exists \sigma . \rho(\mathscr{M}(\sigma), \varphi) \leq 0$

Fact: $\mathscr{M}' \models \varphi$ i.e.
$\forall \sigma . \rho(\mathscr{M}'(\sigma), \varphi) \geq 0$
by model checking

Heuristic: Find σ s.t.
$\mathscr{M}(\sigma) \neq \mathscr{M}'(\sigma)$ in
$\rho(\mathscr{M}(\sigma), \varphi)$ is small



Outline

- Preliminaries
  - Active automata learning
  - Black-box checking
- Techniques for efficient black-box checking
  - Robustness-guided equivalence testing
  - Model checking of strengthened formulas
  - Specification-guided abstraction of system under test
- Java Toolkit FalCAuN for black-box checking
- Probabilistic extension of black-box checking

Black-Box Checking (BBC)

[Peled et al., FORTE'99]

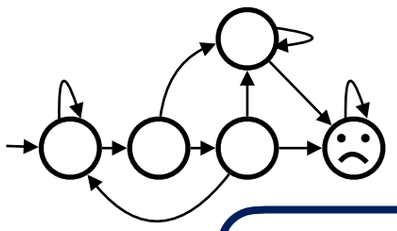
$\mathscr{M}$: 

φ: Spec.

Found σ s.t.
$\mathscr{M}(\sigma) \neq \mathscr{M}'(\sigma)$

✔
→ $\mathscr{M}(\sigma) \neq \mathscr{M}'(\sigma)$

Active Automata Learning, e.g. L*

$\mathscr{M}'$: 

Verify $\mathscr{M}'$ against φ

Find evidence of $\mathscr{M} \neq \mathscr{M}'$

Test $\mathscr{M}$ w/ input σ

Not Found ✔

✘ w/ input σ

✘

Likely ✔ (φ seems satisfied)

✘ (φ is violated)

Black-Box Checking (BBC)

[Peled et al., FORTE'99]

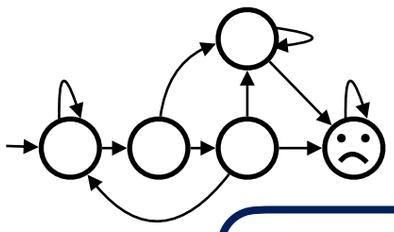
$\mathscr{M}$: 

φ: Spec.

Found σ s.t.
$\mathscr{M}(\sigma) \neq \mathscr{M}'(\sigma)$

✔
→ $\mathscr{M}(\sigma) \neq \mathscr{M}'(\sigma)$

Active Automata Learning, e.g. L*

$\mathscr{M}'$: 

Verify $\mathscr{M}'$ against φ

Difficult

Find evidence of $\mathscr{M} \neq \mathscr{M}'$

Test $\mathscr{M}$ w/ input σ

Not Found ✔

✘ w/ input σ

✘

Likely ✔ (φ seems satisfied)

✘ (φ is violated)

Finding a Deviation via Model checking of Strengthened Spec.

Idea: find "near unsafe" inputs via model checking

Observation 1: model checking of $\mathscr{M}'$ is typically more efficient than equivalent testing

Running $\mathscr{M}$ for many times

- Particularly when running $\mathscr{M}$ is time consuming

Observation 2: $\sigma \in \Sigma^*$ s.t. $\mathscr{M}(\sigma) \neq \mathscr{M}'(\sigma)$ and $\mathscr{M}'(\sigma) \notin \psi$ for a bit stronger ψ than φ is often useful for BBC

- *Stronger*: $\psi \subsetneq \varphi$
- If ψ is much stronger than φ, we cannot bias the learning

BBC + Model checking with Strengthened Spec.

$\mathscr{M}$:

Found σ s.t.

$\mathscr{M}(\sigma) \neq \mathscr{M}'(\sigma)$

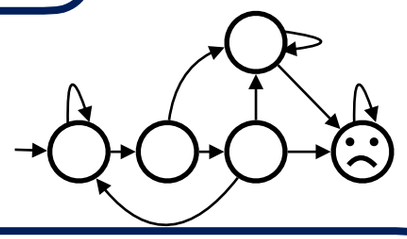
φ: Spec. [Shijubo et al., RV'21]

✔

$\rightarrow \mathscr{M}(\sigma) \neq \mathscr{M}'(\sigma)$

Active Automata Learning, e.g. L*

$\mathscr{M}'$:



✘

Test if $\mathscr{M}(\sigma) = \mathscr{M}'(\sigma)$

Verify $\mathscr{M}'$ against φ

✔

✘ w/ input σ

✔

✘ w/ input σ

Difficult

Find evidence of $\mathscr{M} \neq \mathscr{M}'$

✔

Verify $\mathscr{M}'$ against ψ

Test $\mathscr{M}$ w/ input σ

Not Found

✘

Likely ✔ (φ seems satisfied)

✘ (φ is violated)

BBC + Model checking with Strengthened Spec.

$\mathscr{M}$: 

Found σ s.t.

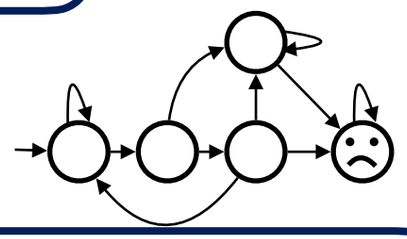
$\mathscr{M}(\sigma) \neq \mathscr{M}'(\sigma)$

φ: Spec. [Shijubo et al., RV'21]

✔

$\rightarrow \mathscr{M}(\sigma) \neq \mathscr{M}'(\sigma)$

Active Automata Learning, e.g. L*

$\mathscr{M}'$: 

Verify $\mathscr{M}'$ against φ

✘

New

Test if $\mathscr{M}(\sigma) = \mathscr{M}'(\sigma)$

✔

✘ w/ input σ

✔

✘ w/ input σ

Difficult

Find evidence of $\mathscr{M} \neq \mathscr{M}'$

✔

Verify $\mathscr{M}'$ against ψ

New

Test $\mathscr{M}$ w/ input σ

Not Found

✘

Likely ✔ (φ seems satisfied)

✘ (φ is violated)

BBC + Model checking with Strengthened Spec.

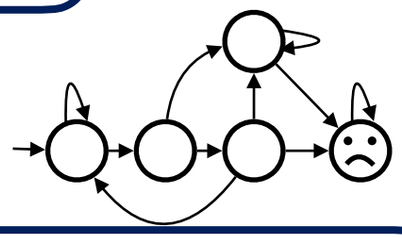
$\mathscr{M}$: 

φ: Spec. [Shijubo et al., RV'21]

Found σ s.t.
$\mathscr{M}(\sigma) \neq \mathscr{M}'(\sigma)$

✔ → $\mathscr{M}(\sigma) \neq \mathscr{M}'(\sigma)$

Active Automata Learning, e.g. L*

$\mathscr{M}'$: 

Verify $\mathscr{M}'$ against φ

✘

New

Test if $\mathscr{M}(\sigma) = \mathscr{M}'(\sigma)$

✔

✘ w/ input σ

✔

✘ w/ input σ

Difficult

Find evidence of $\mathscr{M} \neq \mathscr{M}'$

✔

Verify $\mathscr{M}'$ against ψ

New

Test $\mathscr{M}$ w/ input σ

Not Found

✘

Likely ✔ (φ seems satisfied)

✘ (φ is violated)

BBC + Model checking with Strengthened Spec.

[Shijubo et al., RV'21]

1. Construct ψ that is (a bit) stronger than φ

2. Model check $\mathscr{M}'$ using ψ

   - If $\mathscr{M}'$ satisfies ψ, we move to equivalence test

3. Check if $\mathscr{M}(\sigma) = \mathscr{M}'(\sigma)$, where $\mathscr{M}'(\sigma) \notin \psi$

   - If $\mathscr{M}(\sigma) \neq \mathscr{M}'(\sigma)$, we use σ to refine $\mathscr{M}'$
   - Otherwise, we have $\mathscr{M}(\sigma) \notin \psi$.
     → we deem ψ too strong and generate a weaker ψ

Syntactic Strengthening for LTL

Syntactic rules to derive stronger LTL formulas

1. *For any* $\mu, \nu \in \mathbf{LTL}$, *we have* $(\mu \vee \nu) \rightarrowtail (\mu \wedge \nu)$.
2. *For any* $\mu \in \mathbf{LTL}$, *we have* $\Diamond\mu \rightarrowtail \Box\Diamond\mu$.
3. *For any* $\mu \in \mathbf{LTL}$, *we have* $\Box\Diamond\mu \rightarrowtail \Diamond\Box\mu$.
4. *For any* $\mu \in \mathbf{LTL}$, *we have* $\Diamond\Box\mu \rightarrowtail \Box\mu$.
5. *For any* $\mu \in \mathbf{LTL}$ *and for any indices* $i, j \in \mathbb{N} \cup$ *have* $\Diamond_{[i,j)}\mu \rightarrowtail \Box_{[i,j)}\mu$.
6. *For any* $\mu, \nu \in \mathbf{LTL}$, *we have* $(\mu \,\mathcal{U}\, \nu) \rightarrowtail (\Box\mu \wedge \sqsubset$

$\vdots$

Syntactic Strengthening for LTL

Syntactic rules to derive stronger LTL formulas

$\Box\Diamond\mu$ is stronger than $\Diamond\mu$

1. *For a* ... *ave* $(\mu \vee \nu) \rightarrowtail (\mu \wedge \nu)$.
2. *For any* $\mu \in \mathbf{LTL}$, *we have* $\Diamond\mu \rightarrowtail \Box\Diamond\mu$.
3. *For any* $\mu \in \mathbf{LTL}$, *we have* $\Box\Diamond\mu \rightarrowtail \Diamond\Box\mu$.
4. *For any* $\mu \in \mathbf{LTL}$, *we have* $\Diamond\Box\mu \rightarrowtail \Box\mu$.
5. *For any* $\mu \in \mathbf{LTL}$ *and for any indices* $i, j \in \mathbb{N} \cup$ *have* $\Diamond_{[i,j)}\mu \rightarrowtail \Box_{[i,j)}\mu$.
6. *For any* $\mu, \nu \in \mathbf{LTL}$, *we have* $(\mu \,\mathcal{U}\, \nu) \rightarrowtail (\Box\mu \wedge \Box$
   $\vdots$

Syntactic Strengthening for LTL

Syntactic rules to derive stronger LTL formulas

1. *For a* $\Box\Diamond\mu$ is stronger than $\Diamond\mu$ *ave* $(\mu \vee \nu) \rightarrowtail (\mu \wedge \nu)$.
2. *For any* $\mu \in \mathbf{LTL}$, *we have* $\Diamond\mu \rightarrowtail \Box\Diamond\mu$.
3. *For any* $\mu \in \mathbf{LTL}$, *we have* $\Box\Diamond\mu \rightarrowtail \Diamond\Box\mu$.
4. *For any* $\mu \in \mathbf{LTL}$, *we have* $\Diamond\Box\mu \rightarrowtail \Box\mu$.
5. *For any* $\mu \in \mathbf{LTL}$ *and for any indices* $i, j \in \mathbb{N} \cup$ *have* $\Diamond_{[i,j)}\mu \rightarrowtail \Box_{[i,j)}\mu$.
6. *For any* μ $\nu \in \mathbf{LTL}$, *we have* $(\mu \,\mathcal{U}\, \nu) \rightarrowtail (\Box\mu \wedge \Box$:

Strengthening by changing timing intervals

Outline

- Preliminaries
  - Active automata learning
  - Black-box checking
- Techniques for efficient black-box checking
  - Robustness-guided equivalence testing
  - Model checking of strengthened formulas
  - Specification-guided abstraction of system under test
- Java Toolkit FalCAuN for black-box checking
- Probabilistic extension of black-box checking

Black-Box Checking (BBC)

[Peled et al., FORTE'99]

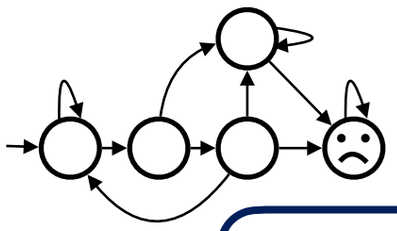
$\mathscr{M}$: 

φ: Spec.

Found σ s.t.
$\mathscr{M}(\sigma) \neq \mathscr{M}'(\sigma)$

✔
→ $\mathscr{M}(\sigma) \neq \mathscr{M}'(\sigma)$

Active Automata Learning, e.g. L*

$\mathscr{M}'$: 

Verify $\mathscr{M}'$ against φ

Find evidence of $\mathscr{M} \neq \mathscr{M}'$

Test $\mathscr{M}$ w/ input σ

Not Found ✔

✘ w/ input σ

✘

Likely ✔ (φ seems satisfied)

✘ (φ is violated)

Black-Box Checking (BBC)

[Peled et al., FORTE'99]

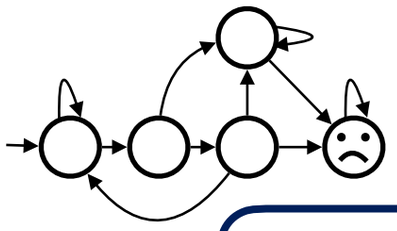
$\mathscr{M}$: 

φ: Spec.

Found σ s.t.
$\mathscr{M}(\sigma) \neq \mathscr{M}'(\sigma)$

✔
→ $\mathscr{M}(\sigma) \neq \mathscr{M}'(\sigma)$

Active Automata Learning, e.g. L*

$\mathscr{M}'$: 

Observation: an abstraction of $\mathscr{M}$ is enough if the result of model checking is same

Verify $\mathscr{M}'$ against φ

Find evidence of $\mathscr{M} \neq \mathscr{M}'$

Test $\mathscr{M}$ w/ input σ

Not Found ✔

✘ w/ input σ

✘

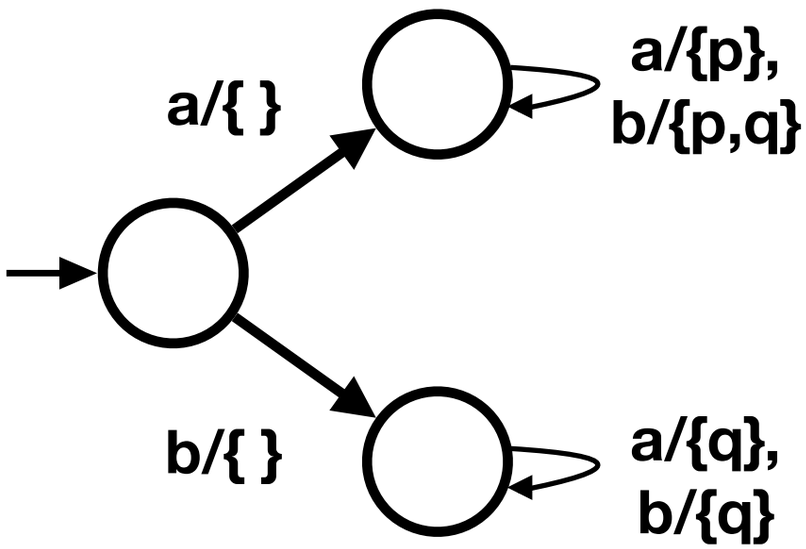
Likely ✔ (φ seems satisfied)

✘ (φ is violated)

Output Abstraction wrt Spec.

[Matsumoto et al., EMSOFT'25]

Spec φ: $\Diamond p \vee \Diamond q$
(Eventually p holds or eventually q holds)

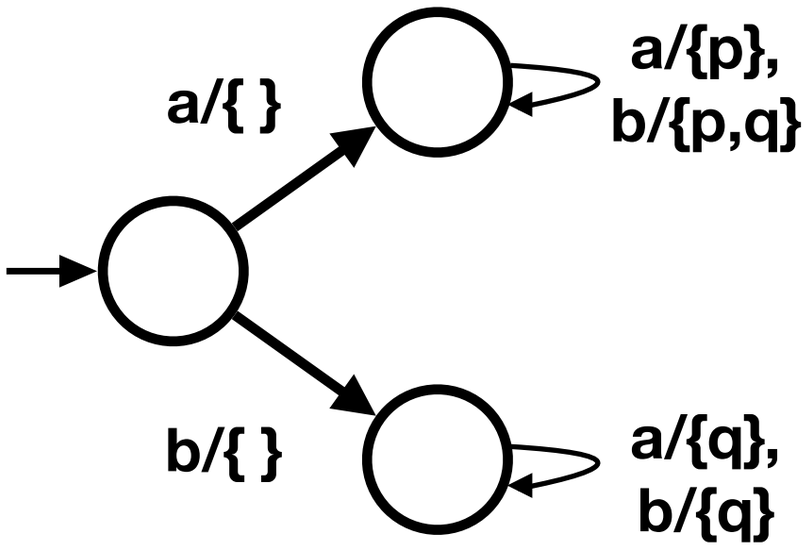


Output Abstraction wrt Spec.

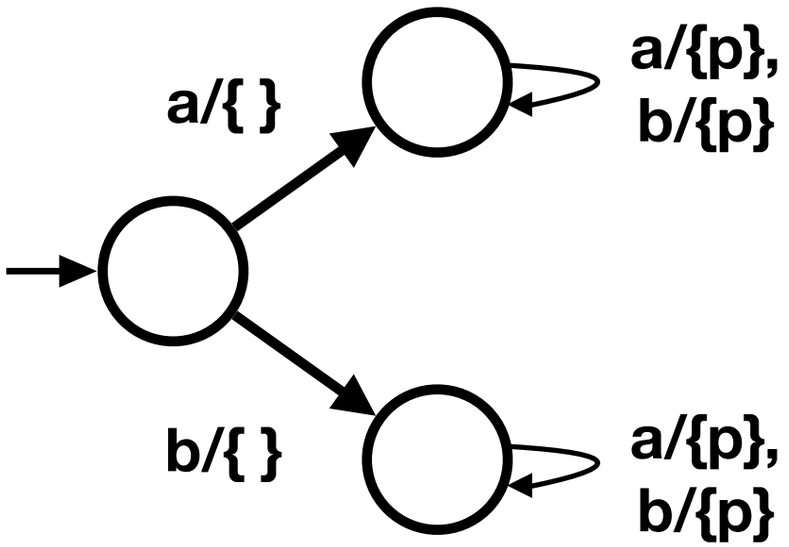
[Matsumoto et al., EMSOFT'25]

Spec φ: $\Diamond p \vee \Diamond q$
(Eventually p holds or eventually q holds)

Equivalent for the satisfaction of φ
(p vs q is irrelevant for φ)



$\approx_{\varphi}$

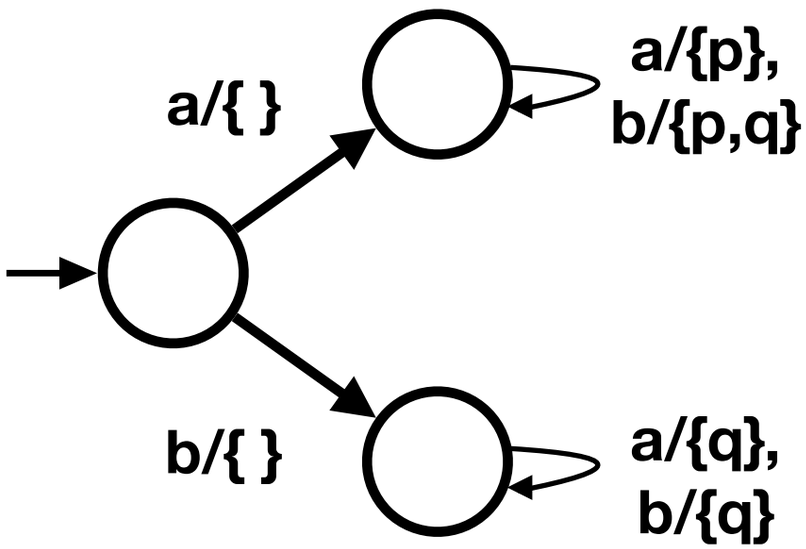


Output Abstraction wrt Spec.

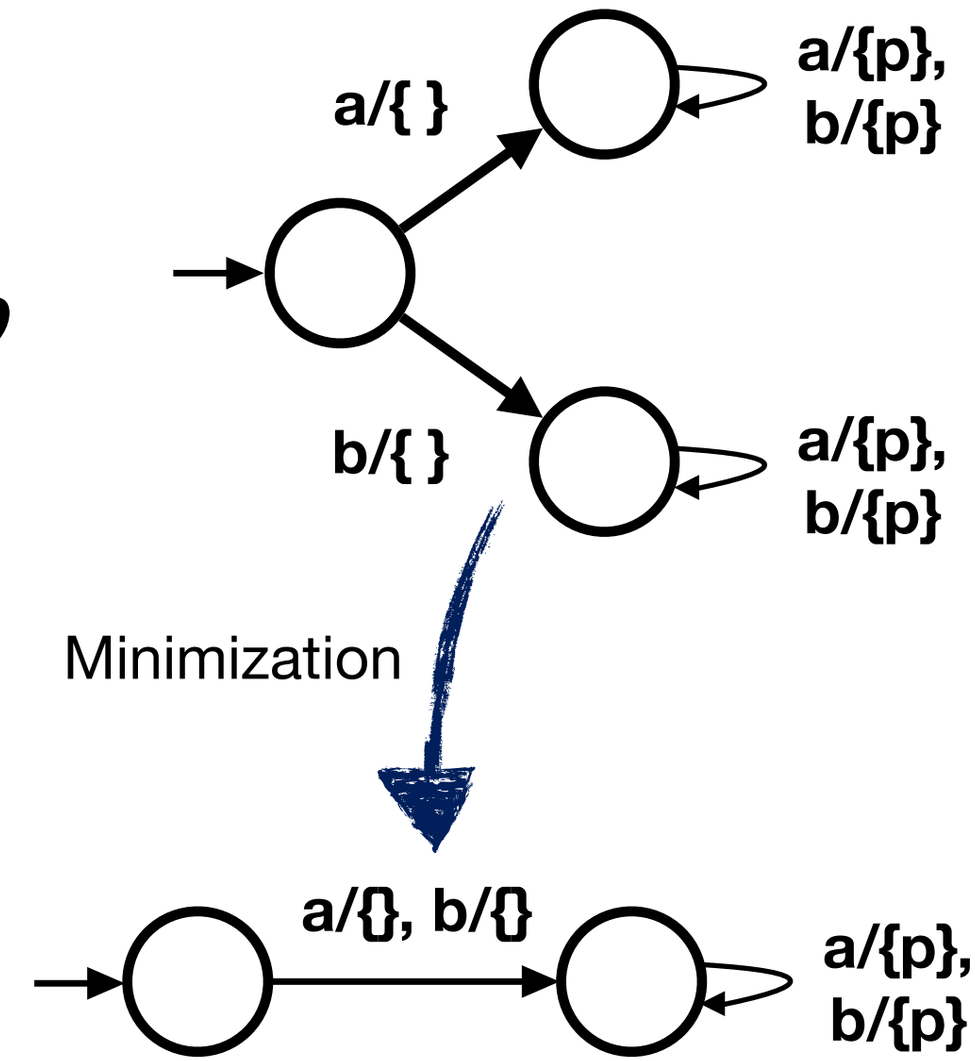
[Matsumoto et al., EMSOFT'25]

Spec φ: $\Diamond p \vee \Diamond q$
(Eventually p holds or eventually q holds)

Equivalent for the satisfaction of φ
(p vs q is irrelevant for φ)



$\approx_{\varphi}$



Specification-guided Abstraction

One can control the granularity of abstraction

- For $v, v' \in 2^{\mathrm{AP}}$, $v \approx_\varphi v'$ only if for any $\sigma \in (2^{\mathrm{AP}})^\omega$, we have $v \cdot \sigma \vDash \psi \iff v' \cdot \sigma \vDash \psi$ for any subfml ψ of φ
- $\mathscr{M}/\approx_\varphi$ is defined by mapping the outputs of $\mathscr{M}$ with $\alpha : 2^{\mathrm{AP}} \to \Gamma$, where $\Gamma = 2^{\mathrm{AP}}/\approx_\varphi$

Violation (or satisfaction) is witnessed by a finite trace

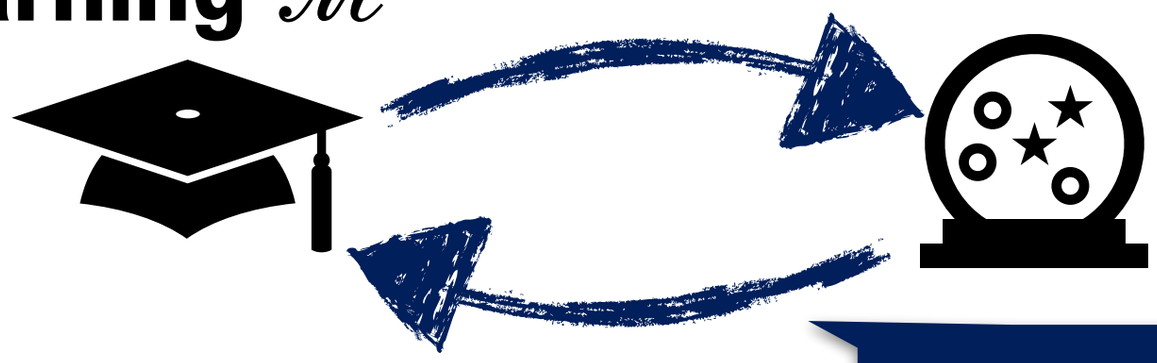
Theorem

We have $\mathscr{M} \vDash \varphi \iff \mathscr{M}/\approx_\varphi \vDash \varphi$ for any safety or co-safety property

Learning of Abstracted System $\mathscr{M}/\approx_\varphi$

Abstracted system can be leaned by changing outputs!

Learning $\mathscr{M}$



Returning $\mathscr{M}(\sigma) \in (2^{\mathrm{AP}})^*$ for $\sigma \in \Sigma^*$

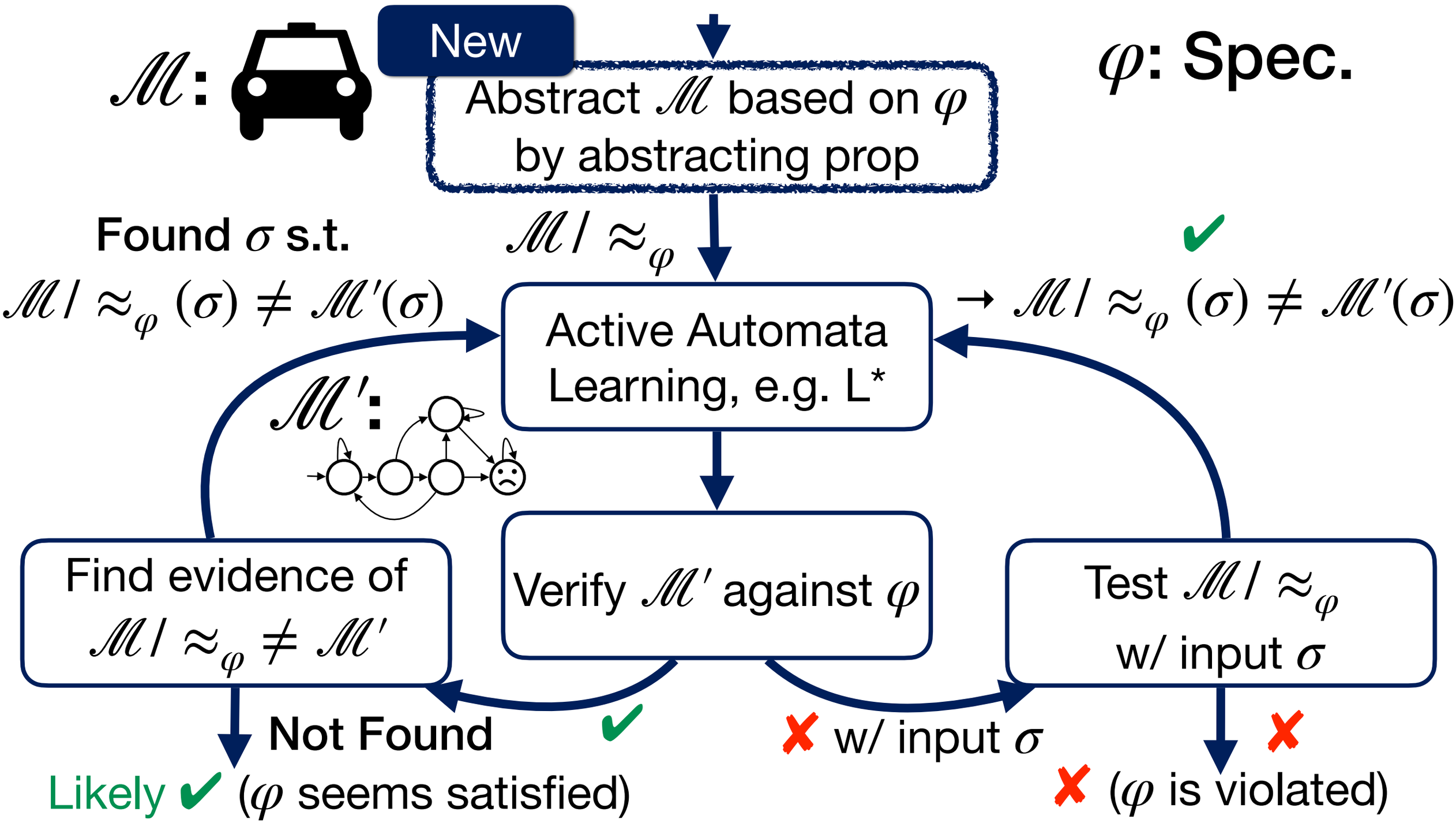
Learning $\mathscr{M}/\approx_\varphi$

Output Abstraction
$\alpha : 2^{\mathrm{AP}} \to \Gamma$

Return
$\mathscr{M}/\approx_\varphi (\sigma) = \alpha^*(\mathscr{M}(\sigma)) \in \Gamma^*$
for $\sigma \in \Sigma^*$

Return $\mathscr{M}(\sigma) \in (2^{\mathrm{AP}})^*$
for $\sigma \in \Sigma^*$

BBC w/ Specification-guided Abstraction



Outline

- Preliminaries
  - Active automata learning
  - Black-box checking
- Techniques for efficient black-box checking
  - Robustness-guided equivalence testing
  - Model checking of strengthened formulas
  - Specification-guided abstraction of system under test
- Java Toolkit FalCAuN for black-box checking
- Probabilistic extension of black-box checking

FalCAuN: A Toolkit for Black-Box Checking

Java library for Black-box checking
→ Can be used from JVM languages

Systems: MATLAB/Simulink, Python, Java

Spec: (discrete-time) STL/LTL

Implementing all three methods to enhance the performance of BBC

They are orthogonal
→ We can use all of them!

Case Study: Insulin Pump

- **System**: Type-1 Diabetes Simulator + RL-based Controller of Insulin Pump
  - **Simulator**: simglucose implemented in Python
- **Requirements**:
  - $\Box bg > 55$
  - $(meal \wedge Xmeal) R (bg > 180 \Rightarrow \Diamond_{[0,1]} bg < 180)$
  - $(meal \wedge Xmeal) R (\text{insulin} > 0.5 \Rightarrow \Diamond_{[0,1]} bg < 180)$
- Executed on a workstation w/
  CPU: Intel i9-10980XE,
  RAM: 128484MiB,
  OS: Ubuntu 22.04

Blood glucose level should not be too low

Blood glucose level should not be too high for a while unless eating a lot

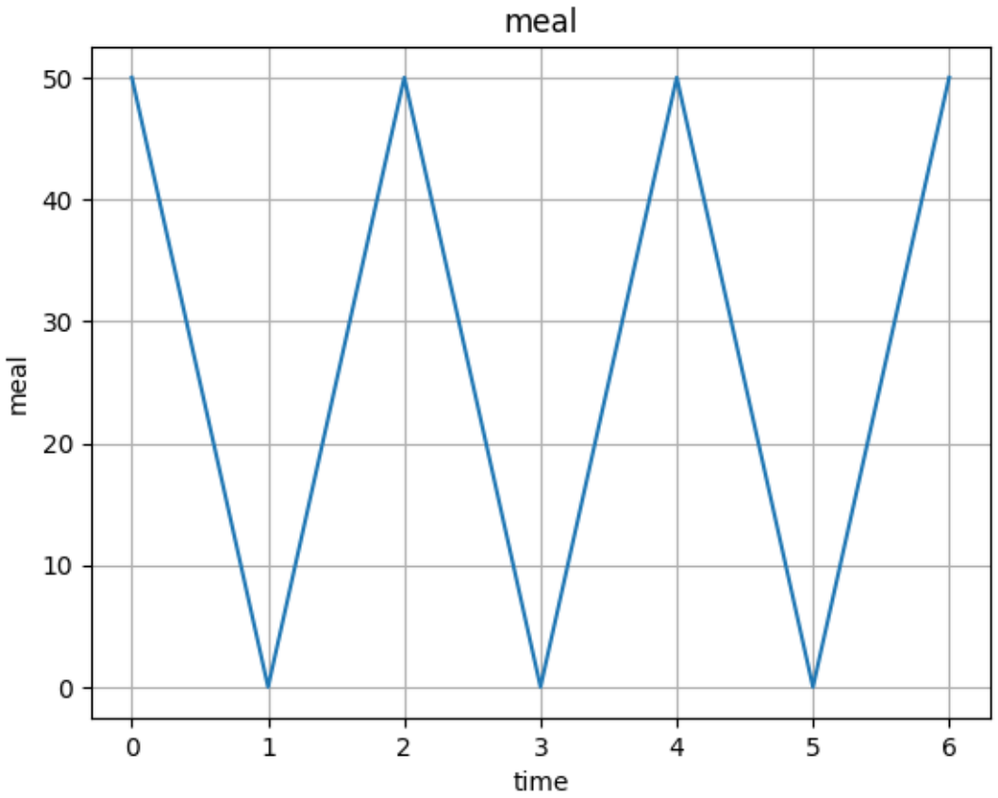
Blood glucose level should not be too high for a while after feeding insulin unless eating a lot

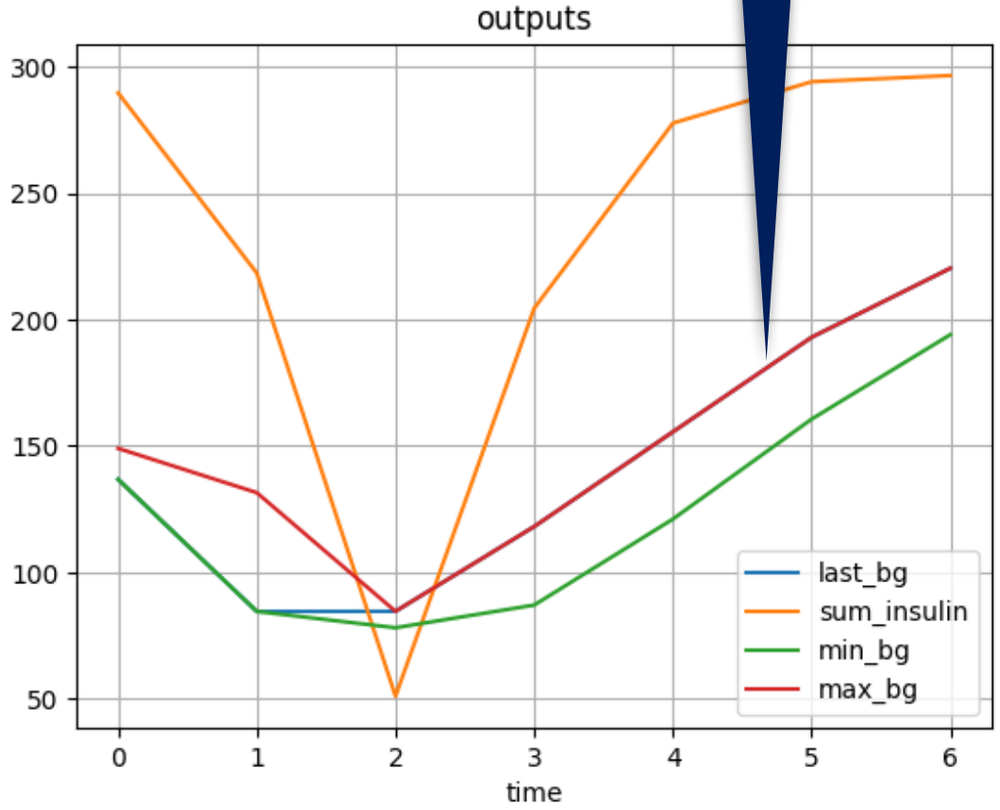
Result of the Case Study

Execution Time: 116.8727 sec

Counter example of $(meal \wedge Xmeal)R(\text{insulin} > 0.5 \Rightarrow \Diamond_{[0,1]} bg < 180)$

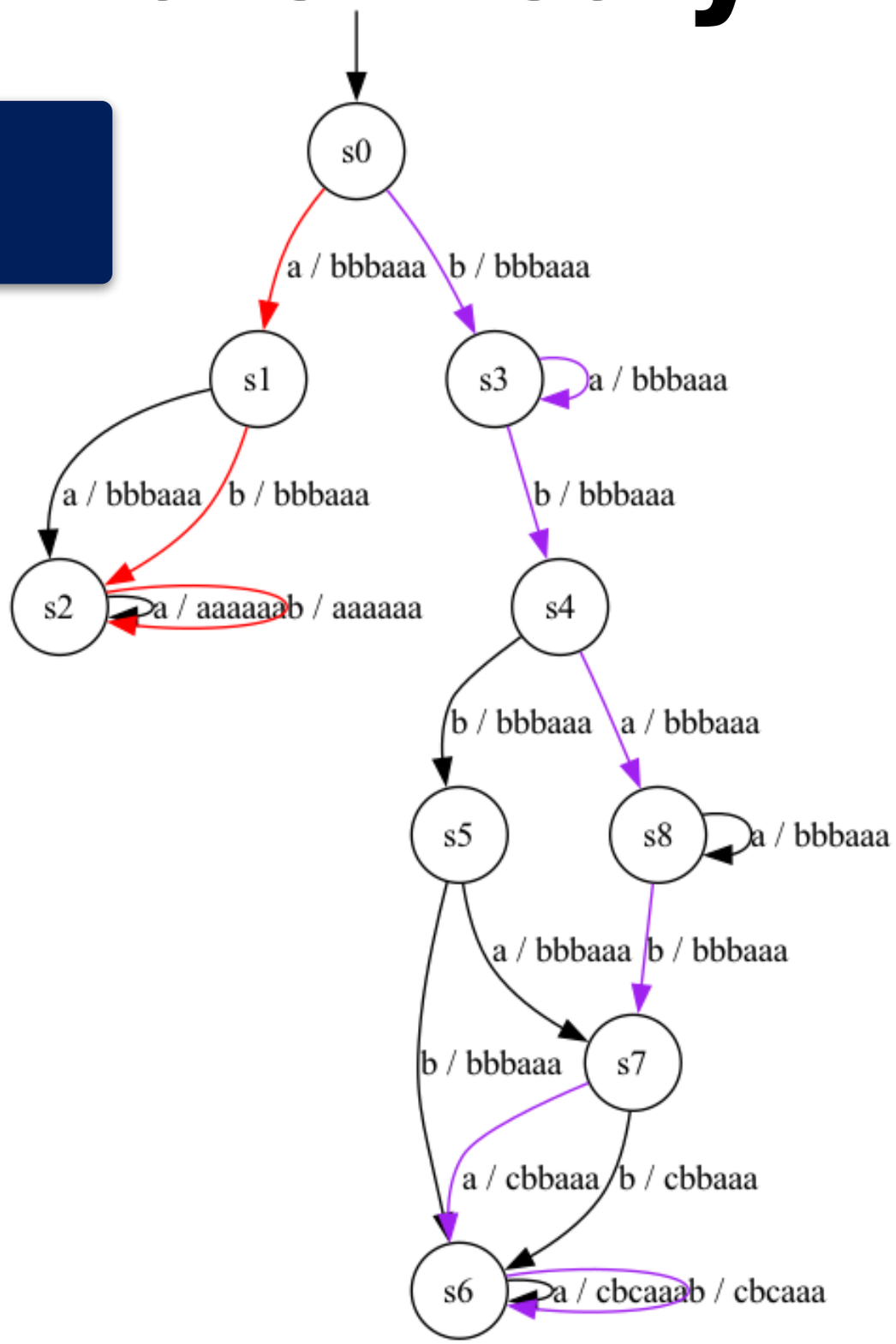
Blood glucose level goes high by "zig-zag eating”





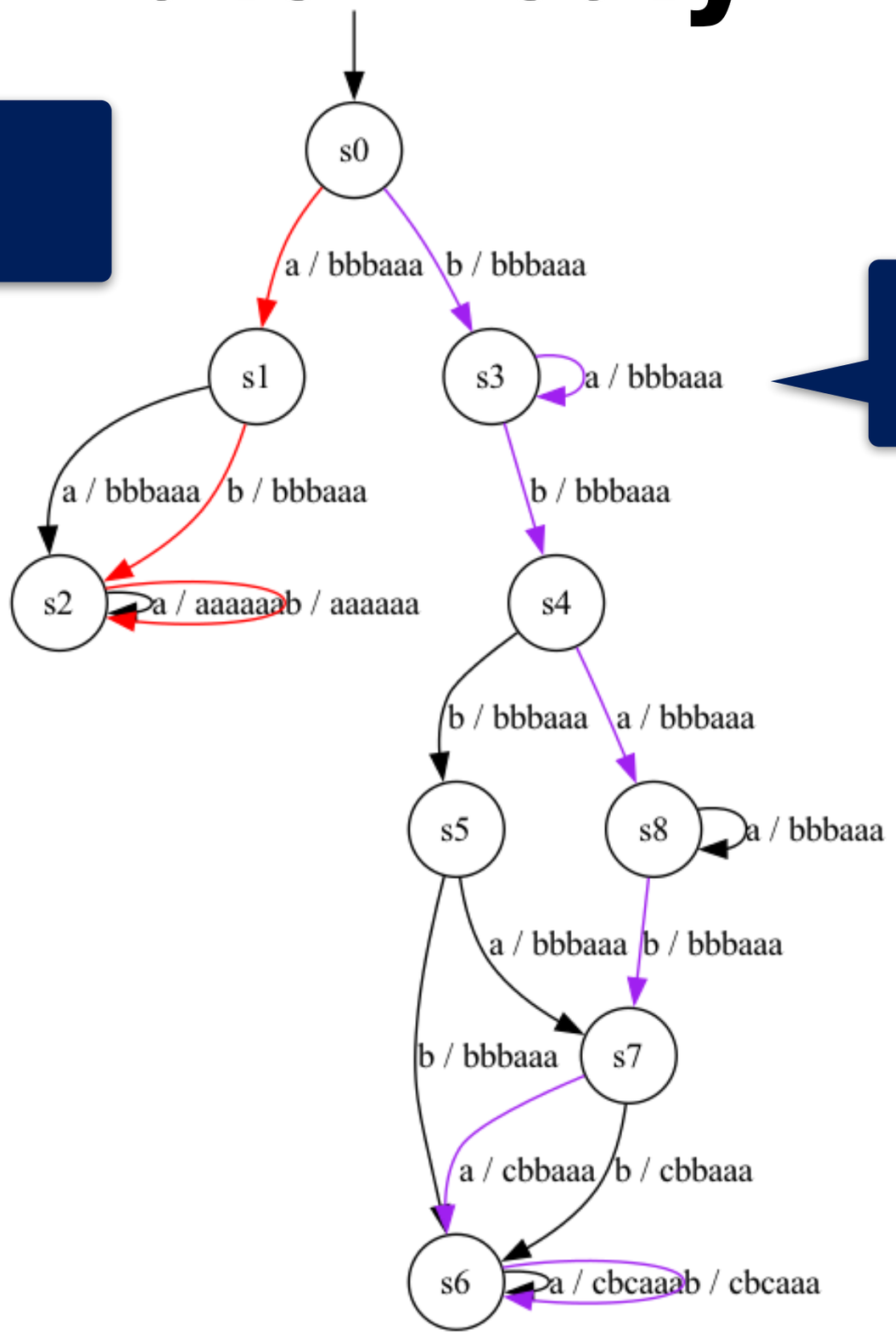
Approximate Mealy Machine $\mathscr{M}'$

a: no meal
b: meal



Approximate Mealy Machine $\mathscr{M}'$

a: no meal
b: meal



not eating meal does not change the state

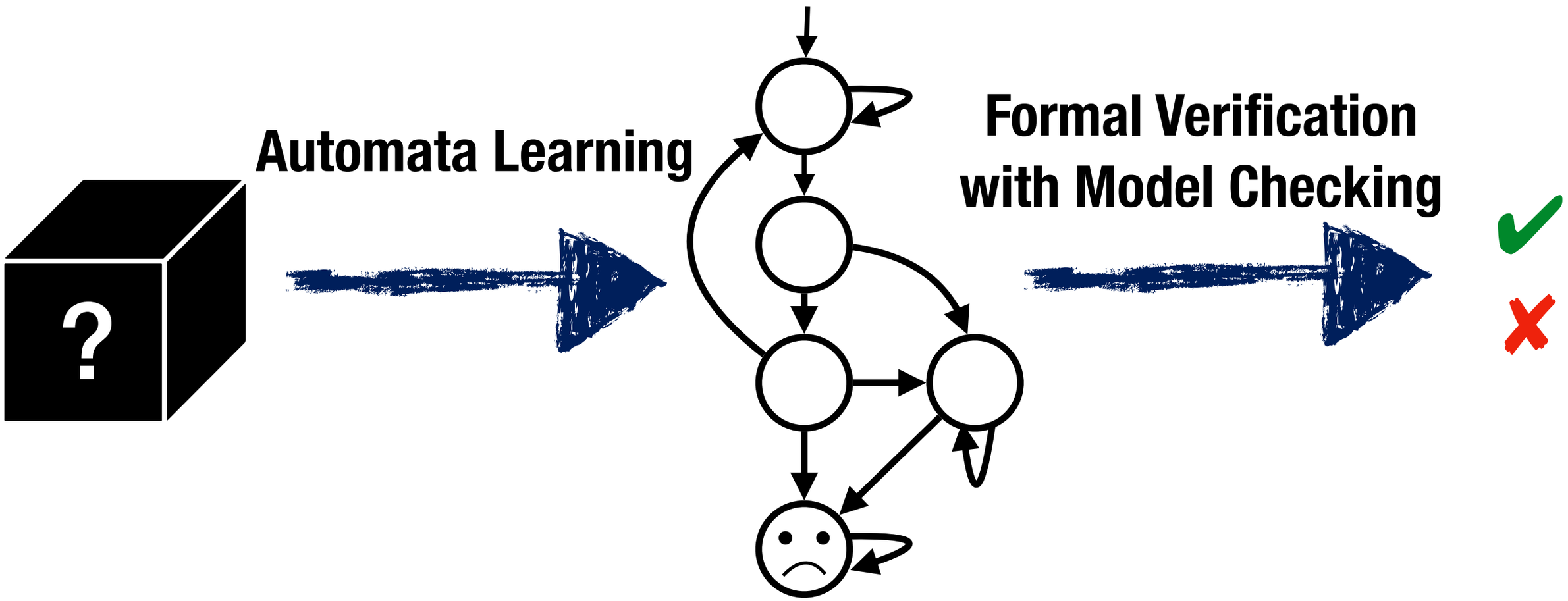
Outline

- Preliminaries
  - Active automata learning
  - Black-box checking
- Techniques for efficient black-box checking
  - Robustness-guided equivalence testing
  - Model checking of strengthened formulas
  - Specification-guided abstraction of system under test
- Java Toolkit FalCAuN for black-box checking
- Probabilistic extension of black-box checking

Black-box Checking

[Peled et al., PSTV & FORTE'99]

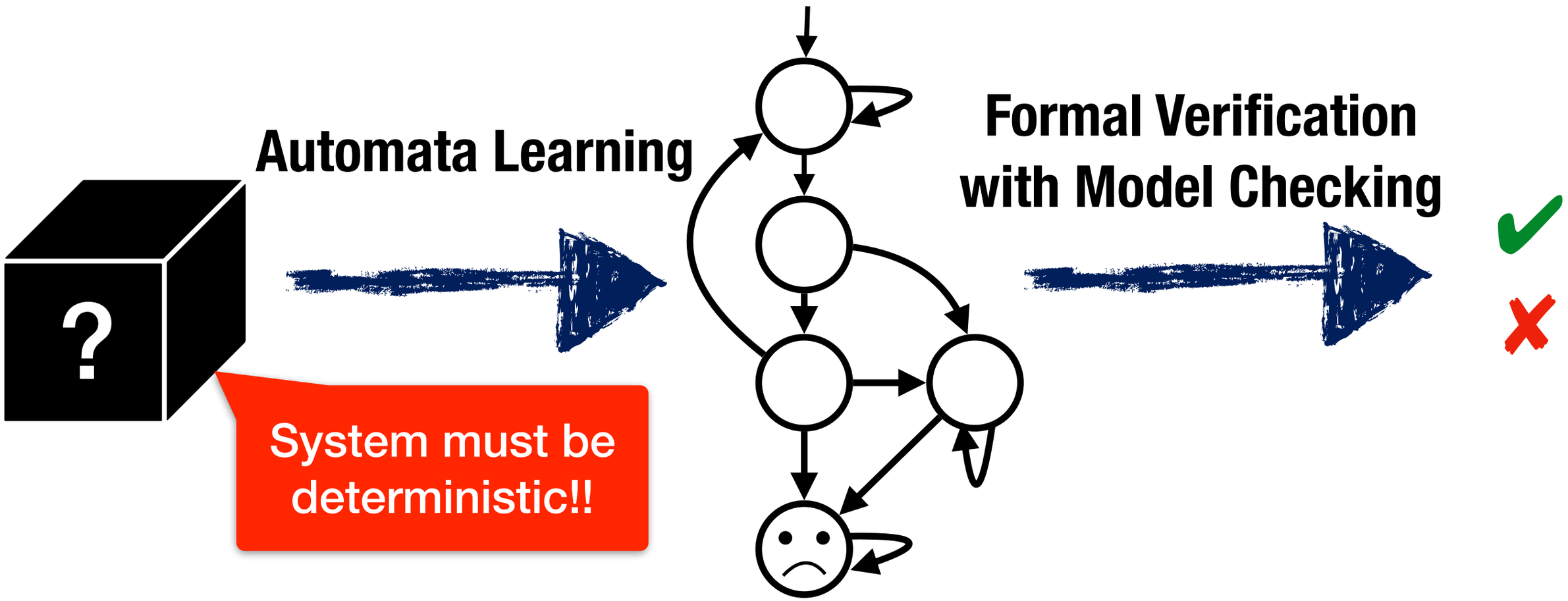
Idea: Automata learning → formal verification!



Black-box Checking

[Peled et al., PSTV & FORTE'99]

Idea: Automata learning → formal verification!



Probabilistic Black-box Checking

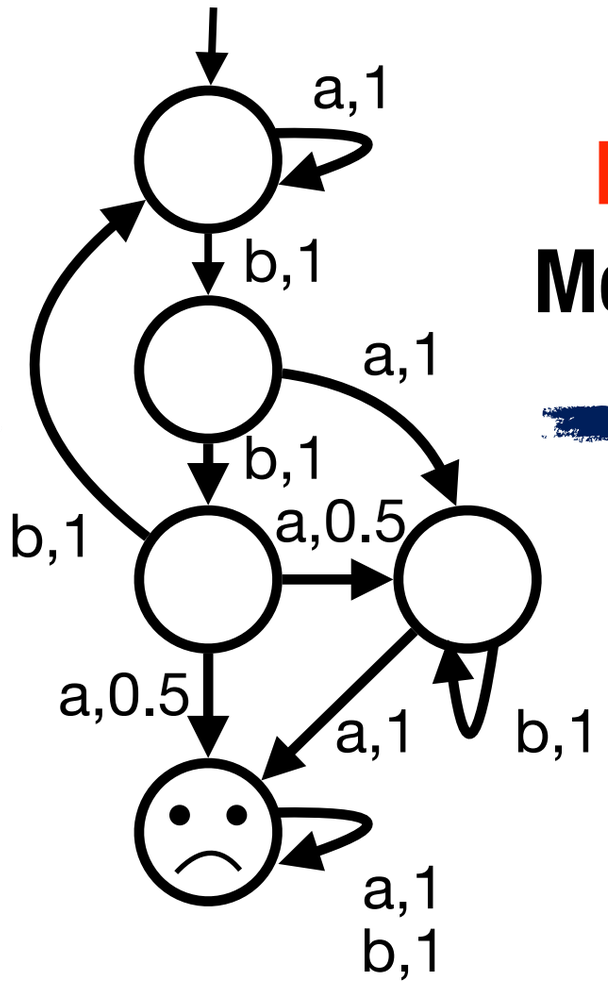
[Shijubo et al., EMSOFT'23]

Idea: MDP learning → formal verification!

MDP Learning

e.g. L^*_{MDP}

[Tappler et al., FAOC'21]



Probabilistic Model Checking

Max./Min. Satisfaction Prob.

Probabilistic Black-box Checking

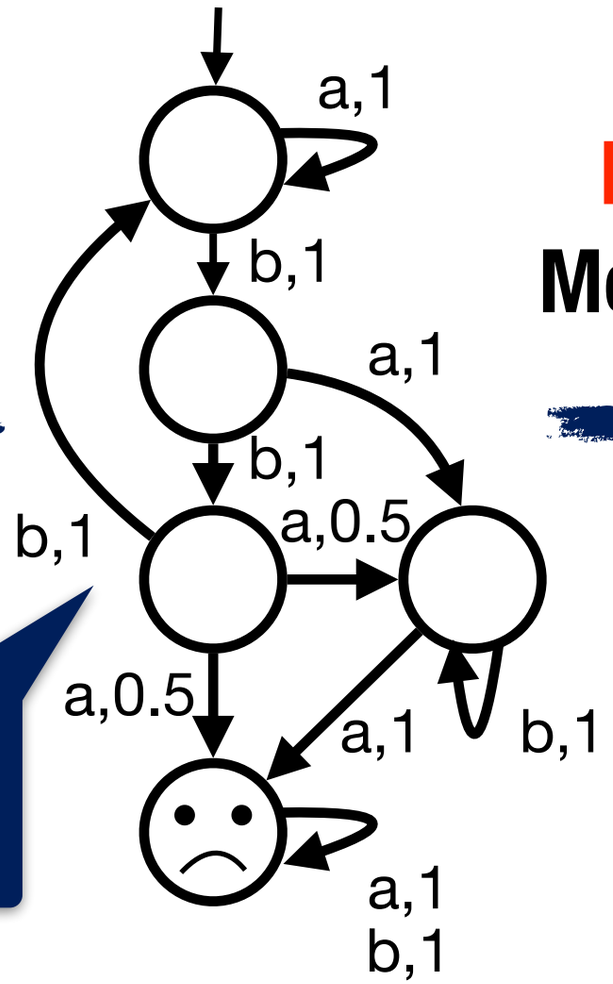
[Shijubo et al., EMSOFT'23]

Idea: MDP learning → formal verification!

MDP Learning

e.g. L^*_{MDP}

[Tappler et al., FAOC'21]



MDP: actions + trans. prob.

Probabilistic Model Checking

Max./Min. Satisfaction Prob.

Probabilistic Black-box Checking

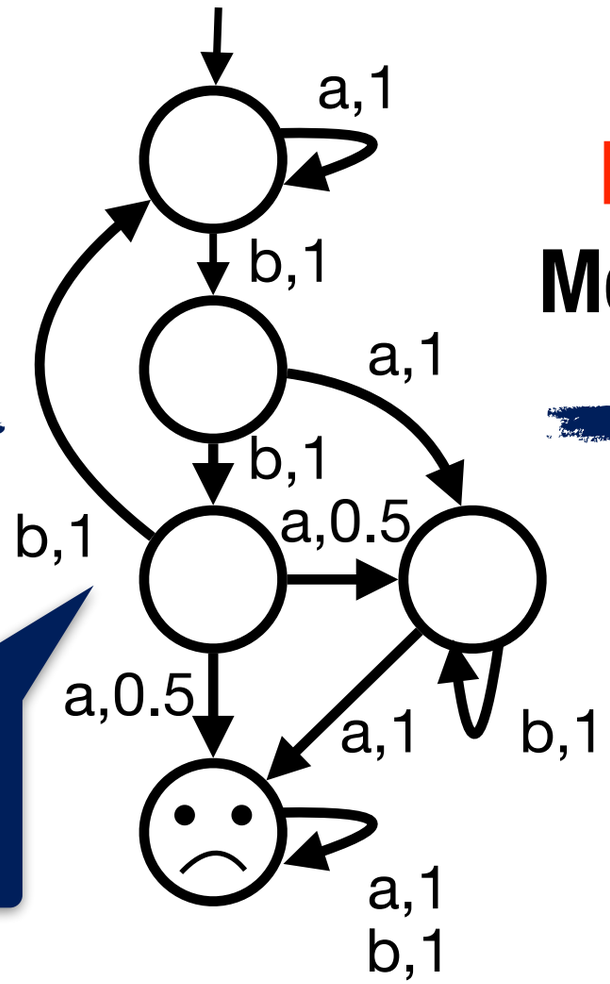
[Shijubo et al., EMSOFT'23]

Idea: MDP learning → formal verification!

MDP Learning

e.g. L^*_{MDP}

[Tappler et al., FAOC'21]



MDP: actions + trans. prob.

Probabilistic Model Checking

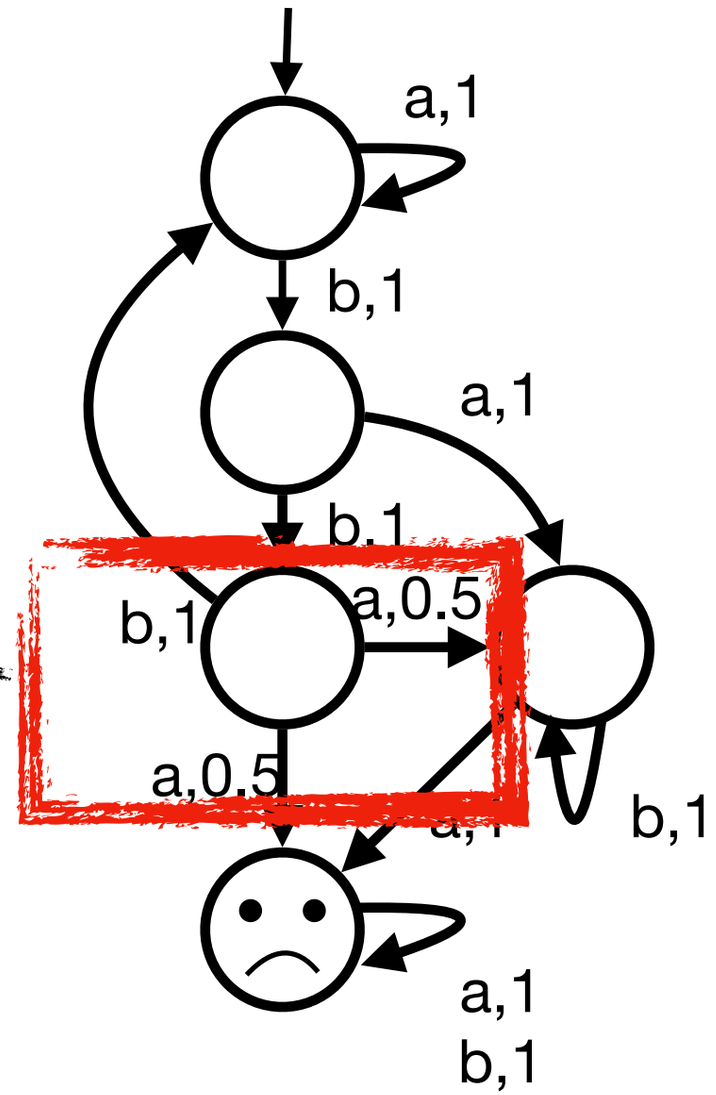
Max./Min. Satisfaction Prob.

Estimate "safe" probability

What is MDP?

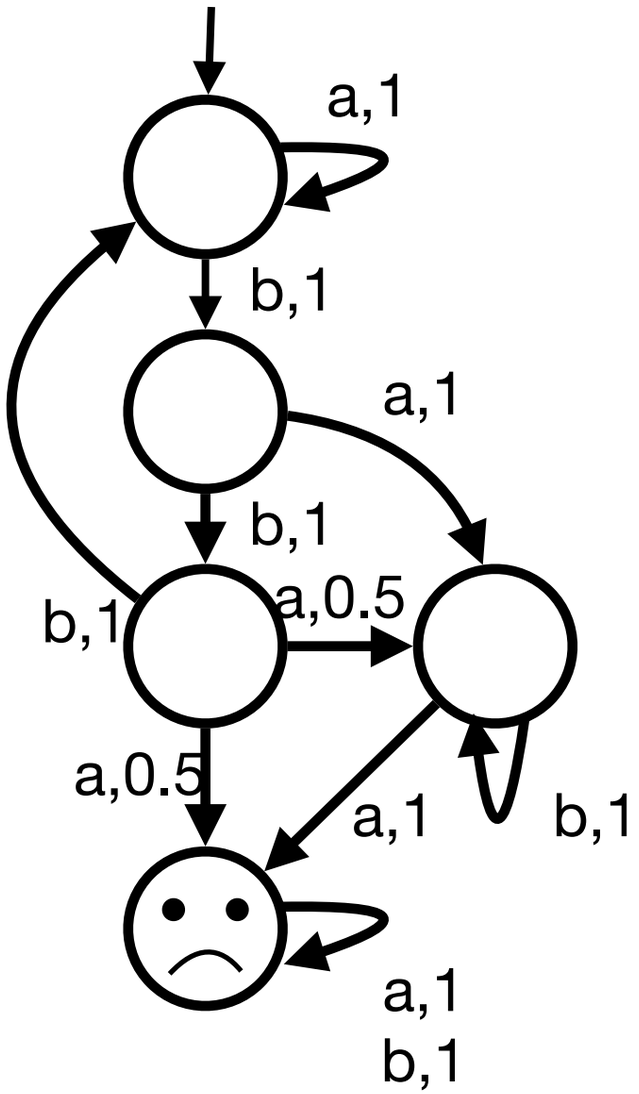
Model of stochastic system accepting inputs

Input sym. and prob. of transition



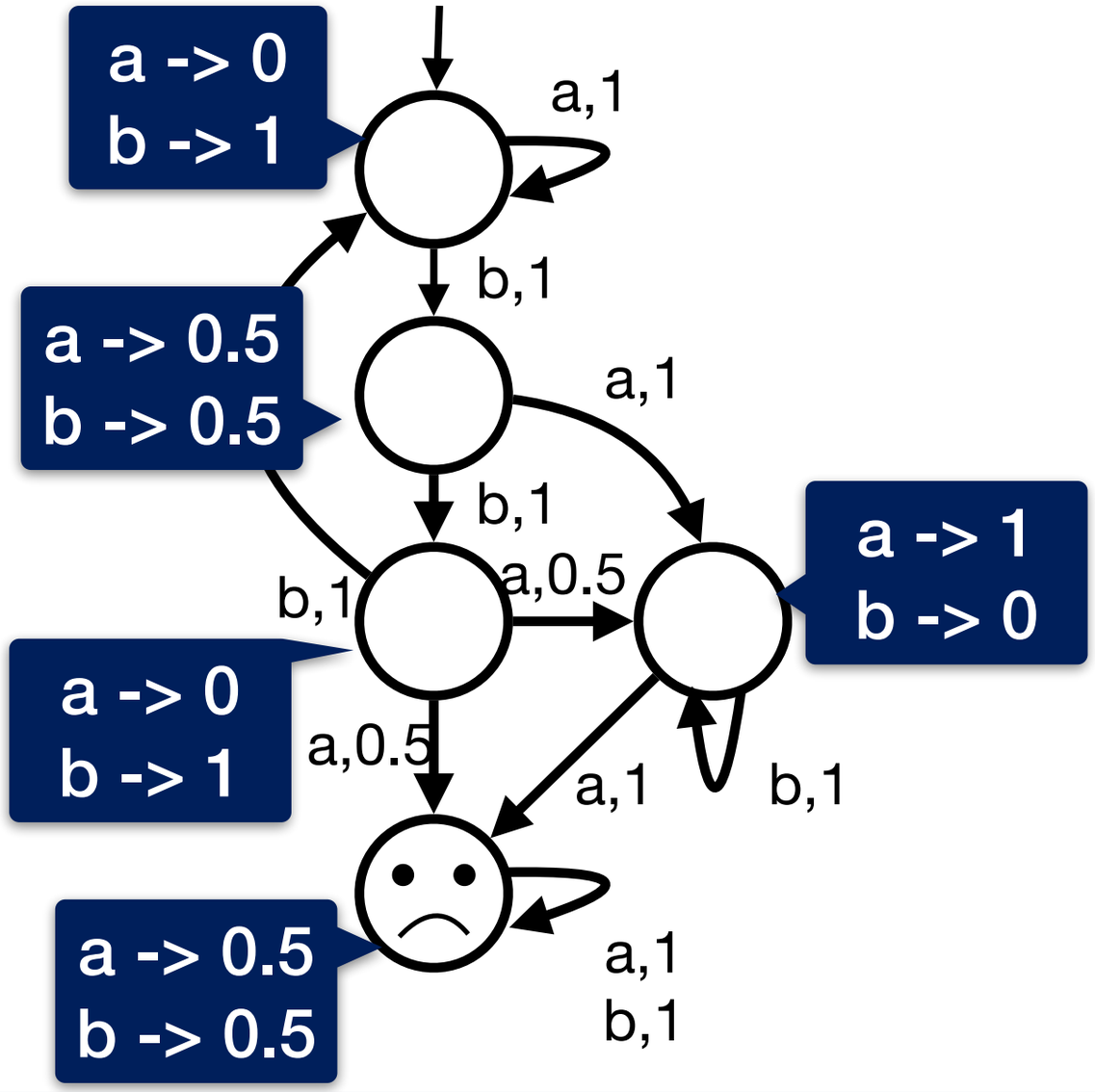
Strategy

Prob. distr. over input symbols at each state



Strategy

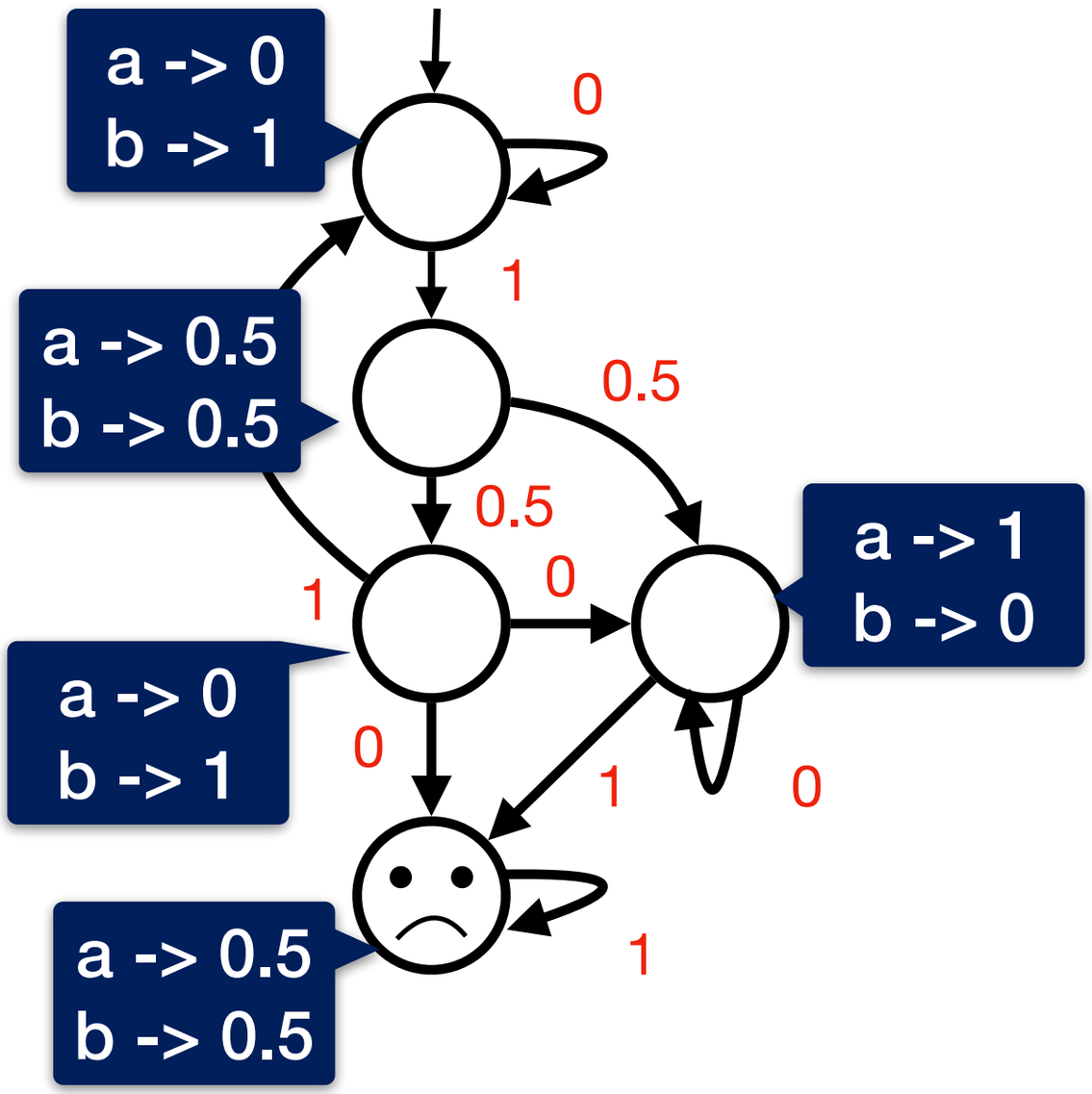
Prob. distr. over input symbols at each state



Strategy

Prob. distr. over input symbols at each state

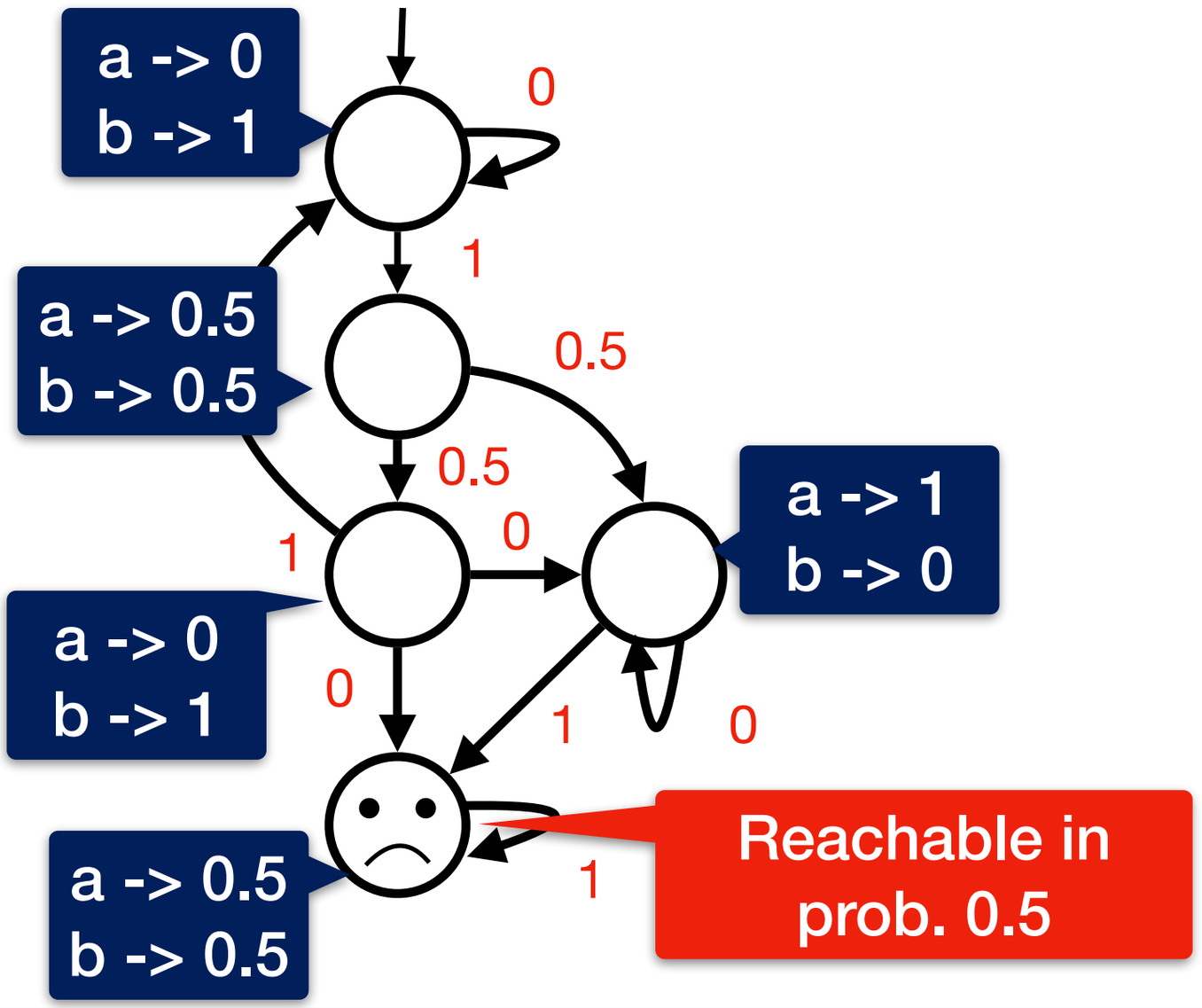
Fixing a strategy turns MDP to Markov chain



Strategy

Prob. distr. over input symbols at each state

Fixing a strategy turns MDP to Markov chain

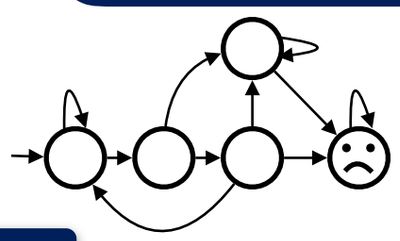


Probabilistic Black-box Checking

[Shijubo et al., EMSOFT'23]

$\mathscr{M}$:  ···· φ: Spec.

Active MDP Learning, e.g. L^*_{MDP}

$\mathscr{M}'$: 

Compute max. sat. prob. of φ in $\mathscr{M}'$

True prob. by prob. model checking

p w/ *strategy* σ

Estimate sat. prob. of φ in $\mathscr{M}$ w/ σ

Not one input

stat. hypo. testing

Far from p → $\mathscr{M}(\sigma) \neq \mathscr{M}'(\sigma)$

Found σ' s.t. $\mathscr{M}(\sigma') \neq \mathscr{M}'(\sigma')$

Find evidence of $\mathscr{M} \neq \mathscr{M}'$

Not far from p

Not Found

Likely p

Correctness in the Limit

Eventually returns the true max. prob. + exact MDP with prob. 1

Theorem (convergence):

e.g. finite states

If black-box system $\mathscr{M}$ satisfies several conditions, with probability 1,

- learned MDP $\mathscr{M}'$ converges to $\mathscr{M}$ with finite iterations
- the estimated probability p converges to the true value

Setting of Experiments

- Implemented in Python
- Show the results for 7 benchmarks mostly from literature
  - The other results are in the paper
- Baseline: **ProbBlackReach** [Aichernig & Tappler, FMSD'19]
  - Similar but using passive learning algorithm with ε-greedy sampling
  - Java implementation
- Google Cloud Platform c2-standard-4 instance (4 vCPU, 16GB RAM) with Debian 11

Summary of the Results

Estimate max. prob.
→ Larger is better

	Ground Truth	Our Method	ProbBlackReach
Slot machine	0.510	0.507	0.480
Slot machine with limited observation	0.510	0.509	0.448
MQTT	0.815	0.808	0.815
TCP	0.771	0.768	0.771
GridWorld Small	0.618	0.617	0.569
GridWorld Large	0.671	0.672	0.0683
SharedCoin	0.250	0.251	0.218

Summary of the Results

Estimate max. prob.
→ Larger is better

	Ground Truth	Our Method	ProbBlackReach
Slot machine	0.510	0.507	0.480
Slot machine with limited observation	0.510	0.509	0.448
MQTT	0.815	0.808	0.815
TCP	0.771	0.768	0.771
GridWorld Small	0.618	0.617	0.569
GridWorld Large	0.671	0.672	0.0683
SharedCoin	0.250	0.251	0.218

Always close to ground truth

Summary of the Results

Estimate max. prob.
→ Larger is better

	Ground Truth	Our Method	ProbBlackReach
Slot machine	0.510	0.507	0.480
Slot machine with limited observation	0.510	0.509	0.448
MQTT	0.815	0.808	0.815
TCP	0.771	0.768	0.771
GridWorld Small	0.618	0.617	0.569
GridWorld Large	0.671	0.672	0.0683
SharedCoin	0.250	0.251	0.218

Always close to ground truth

Not much close to ground truth

Summary of the Results

Estimate max. prob.
→ Larger is better

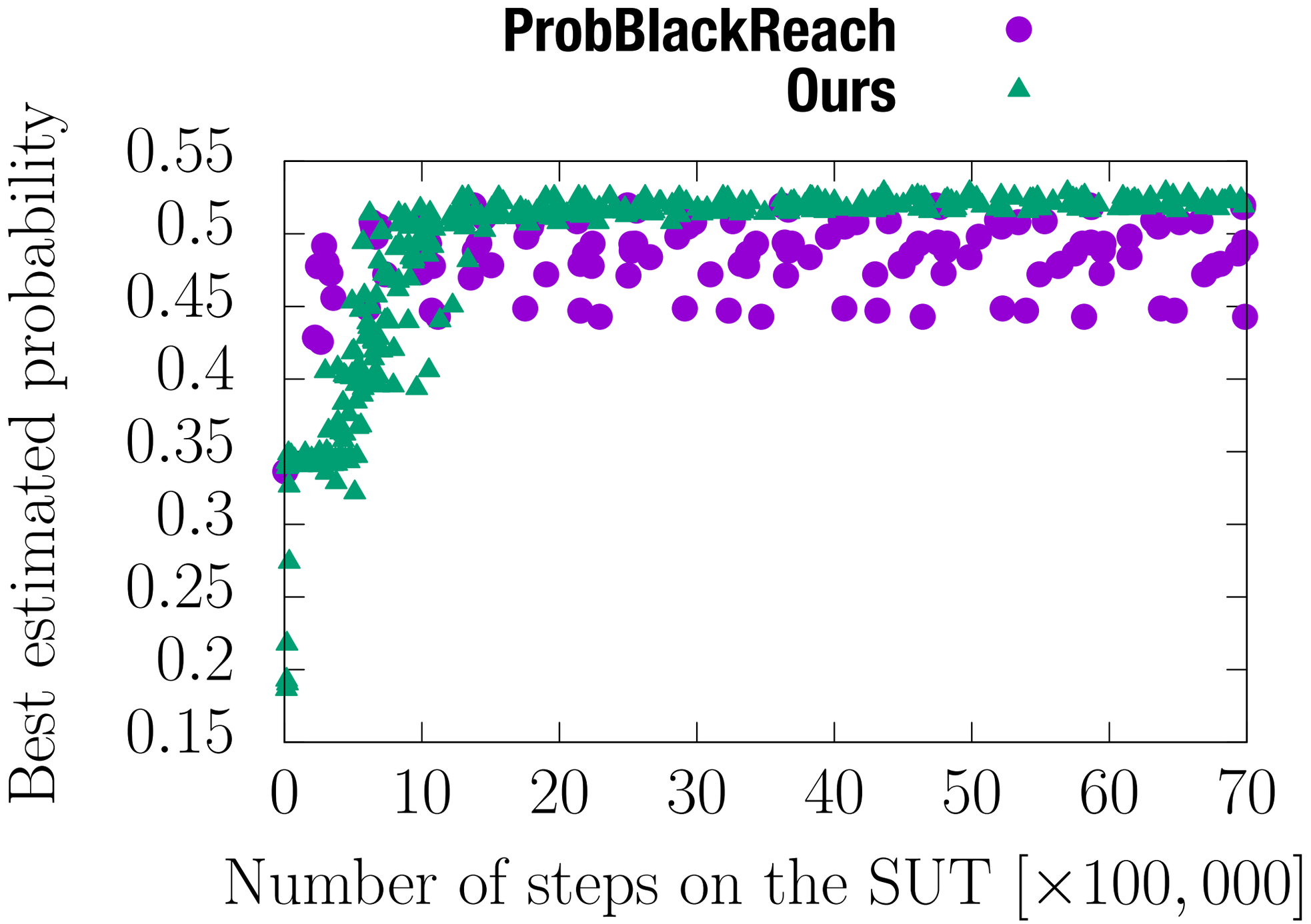
	Ground Truth	Our Method	ProbBlackReach
Slot machine	0.510	0.507	0.480
Slot machine with limited observation	0.510	0.509	0.448
MQTT	0.815	0.808	0.815
TCP	0.771	0.768	0.771
GridWorld Small	0.618	0.617	0.569
GridWorld Large	0.671	0.672	0.0683
SharedCoin	0.250	0.251	0.218

Always close to ground truth

Not much close to ground truth

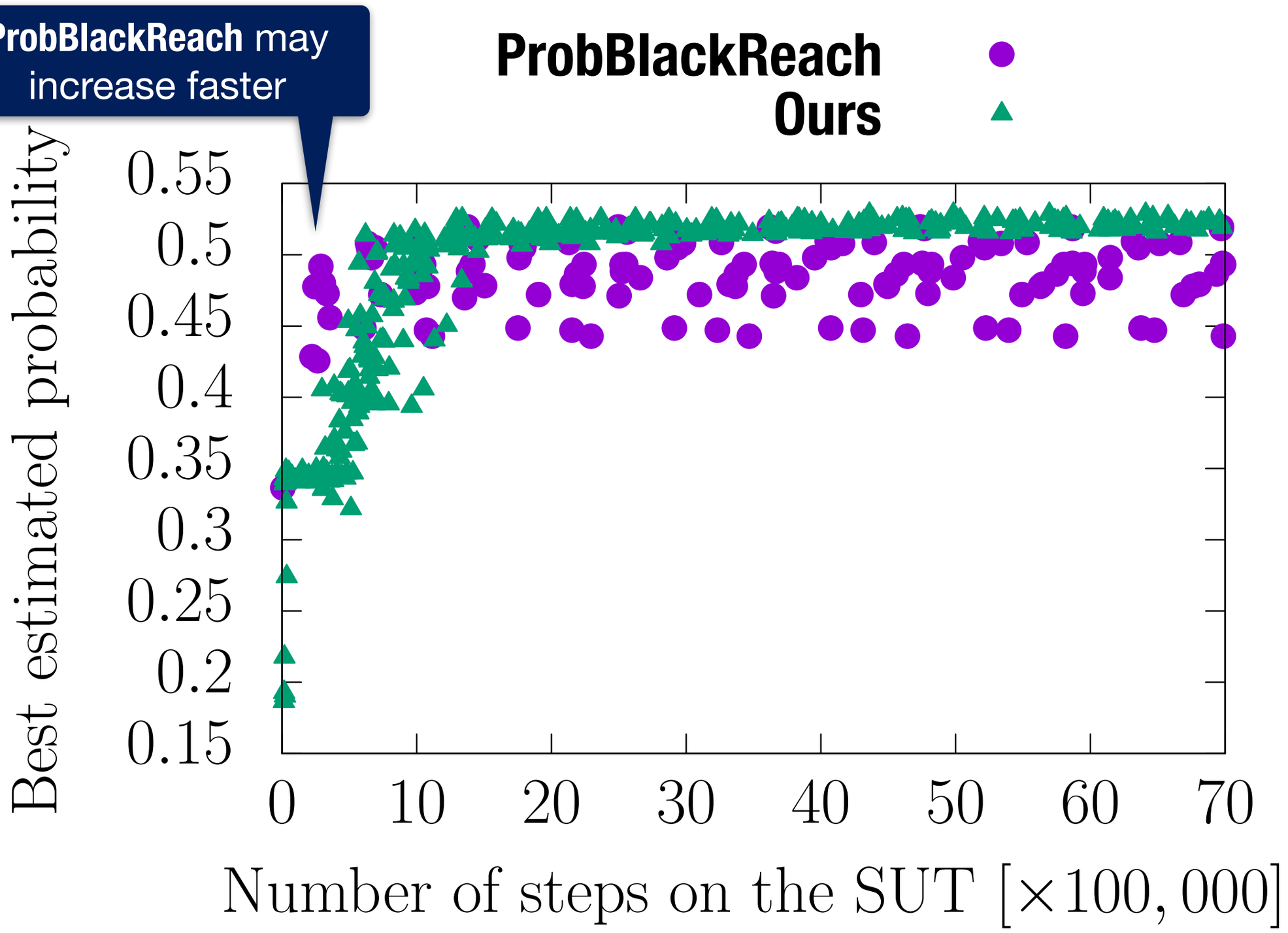
Far from ground truth

Summary of the Results



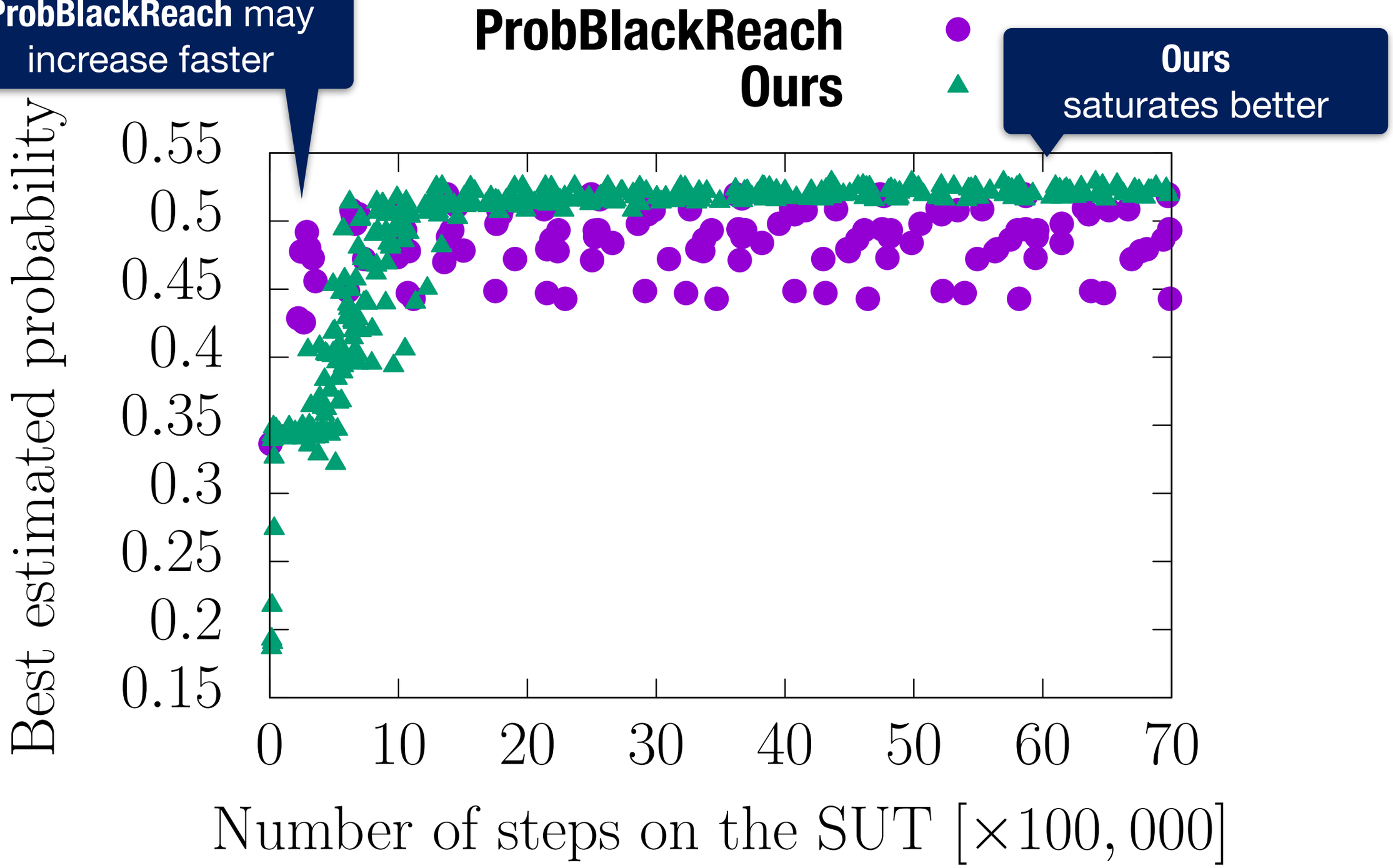
Summary of the Results

ProbBlackReach may increase faster



Summary of the Results

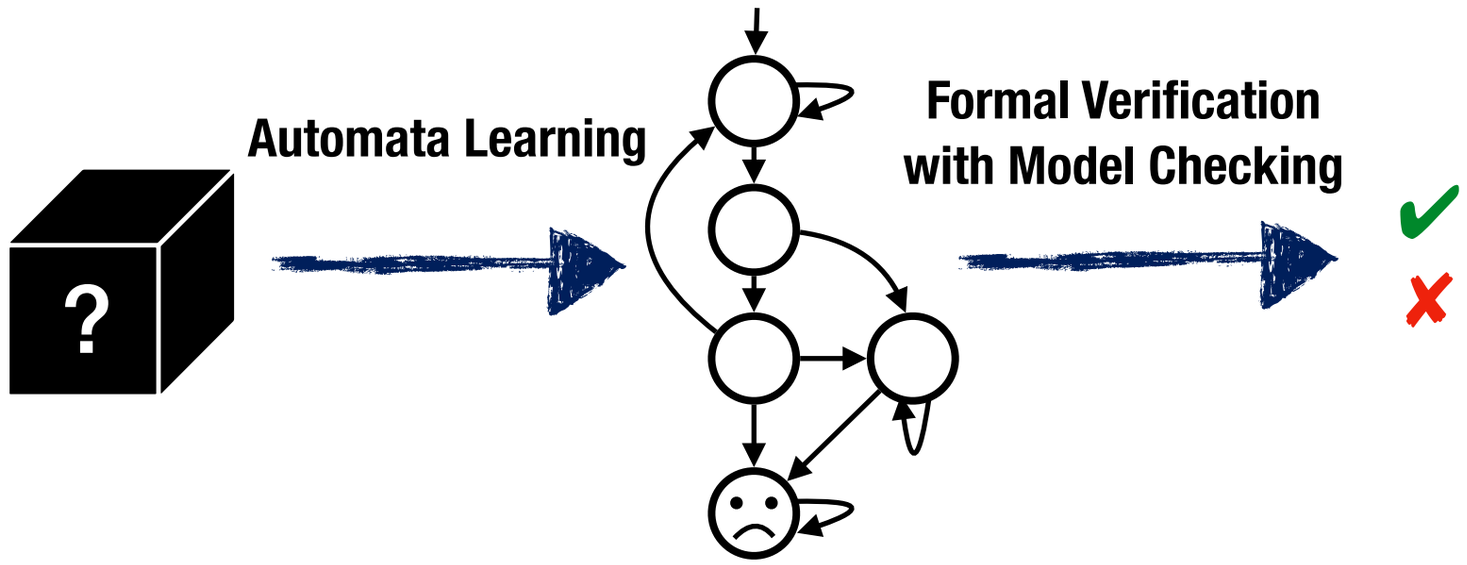
ProbBlackReach may increase faster



Ours saturates better

Conclusions

- One can systematically test black-box systems with black-box checking
  - **Idea**: Automata learning + model checking
- We proposed several techniques to enhance the efficiency of black-box checking
- A probabilistic extension is also available



Some Future Directions

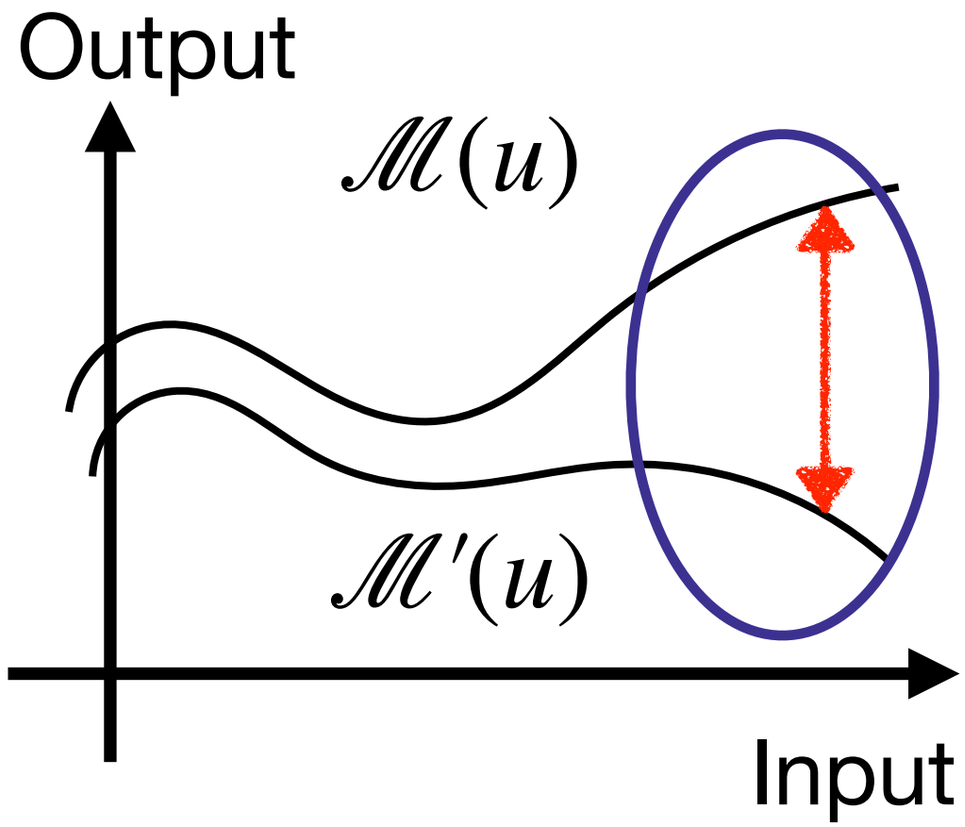
- Extension of black-box checking for, e.g.
  - continuous-time systems
    - Timed automata
    - Hybrid automata
  - Infinite alphabet
    - Symbolic automata
    - Nominal automata
  - More expressive automata
    - Visibly pushdown automata
- Use of white-box information of the system for gray-box systems
- More practical case study

Appendix

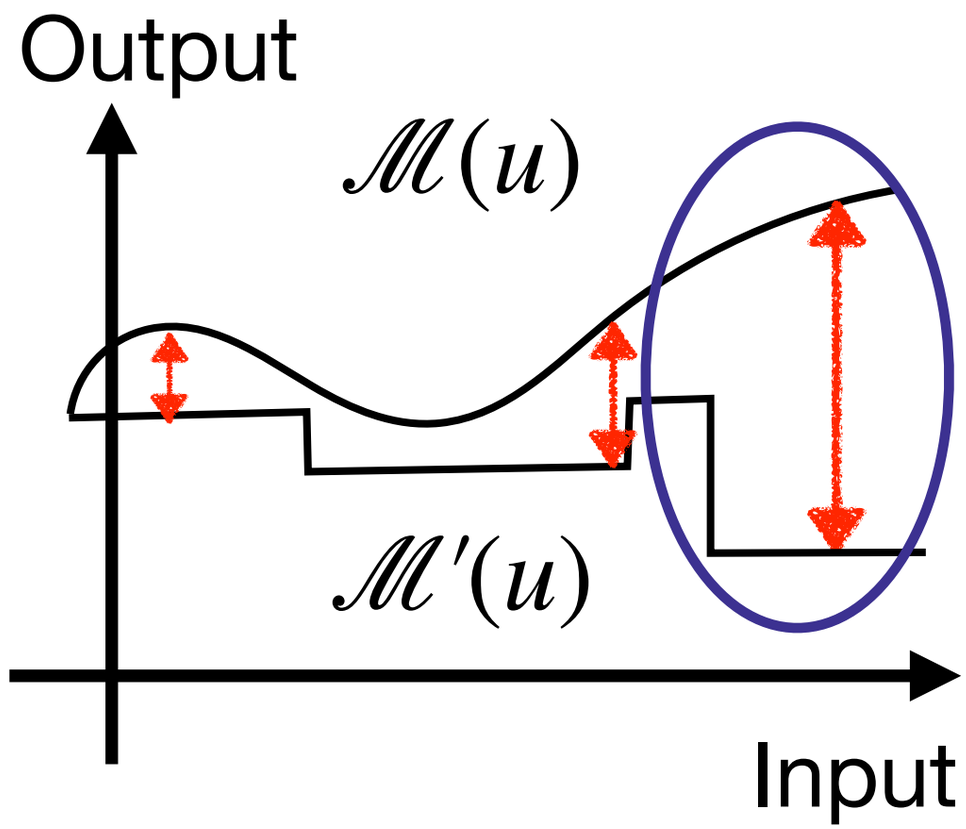
How About Direct Comparison?

- Direct comparison is also available e.g. [Abbas+, MEMOCODE'14]
- **Potential Issue**: more local maxima

Cont. vs Cont.



Cont. vs Disc.



Example on Jupyter w/ Kotlin

jupyter ATS1-step-5 Last Checkpoint: 13 days ago

File Edit View Run Kernel Settings Help

Not Trusted

Markdown

Kotlin

Use Kotlin to load the system, write spec, visualize the results, …

```
val initScript = """
versionString = version('-release');
oldpath = path;
path(strcat(userpath, '/Examples/R', versionString, '/simulink_automotive/ModelingAnAutomaticTransmissionControllerExample/'), oldpath);

mdl = 'Autotrans_shift';
load_system(mdl);
"""
val paramNames = listOf("throttle", "brake")
val signalStep = 5.0
val simulinkSimulationStep = 0.0025
```

[3]:
```
// Load the automatic transmission model. This must be manually closed!!
val sul = SimulinkSUL(initScript, paramNames, signalStep, simulinkSimulationStep)
```

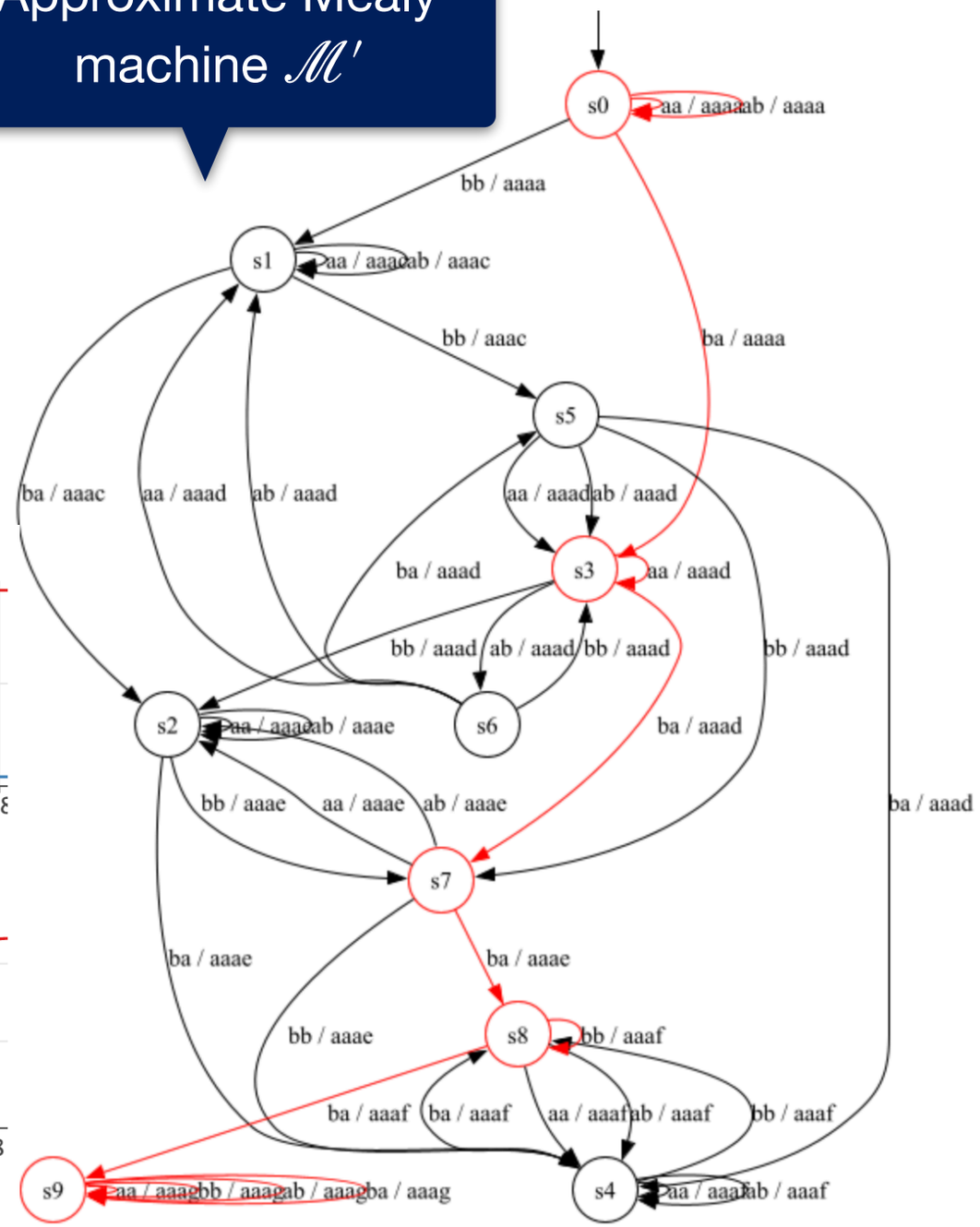
Definition of the STL properties

[4]:
```
import java.io.BufferedReader
import java.io.StringReader

// Define the input and output mappers
val throttleValues = listOf(0.0, 100.0)
val brakeValues = listOf(0.0, 325.0)
val inputMapper = InputMapperReader.make(listOf(throttleValues, brakeValues))
val ignoreValues = listOf(null)
val velocityValues = listOf(20.0, 40.0, 60.0, 80.0, 100.0, 120.0, null)
val accelerationValues = listOf(null)
val gearValues = listOf(null)
val outputMapperReader = OutputMapperReader(listOf(ignoreValues, accelerationValues, gearValues, velocityValues))
outputMapperReader.parse()
val mapperString = listOf("previous_max_output(0)").joinToString("\n")
val signalMapper: ExtendedSignalMapper = ExtendedSignalMapper.parse(BufferedReader(StringReader(mapperString)))
assert(signalMapper.size() == 1)
val mapper =
    NumericSULMapper(inputMapper, outputMapperReader.largest, outputMapperReader.outputMapper, signalMapper)
```

Example on Jupyter w/ Kotlin

Approximate Mealy machine $\mathscr{M}'$



Input signal to make the velocity of a car too large