Error-awareness Accelerates Active Automata Learning

Loes Kruger, Sebastian Junges, and *Jurriaan Rot*

STANP workshop, Paris, 2026

iCIS | Software Science
Radboud University

Overview

- Challenge: **scalability and efficiency** of automata learning
- Observation: lots of **irrelevant inputs words** leading to observable errors
- Our work: learning and testing with **(partial) knowledge of relevant inputs**

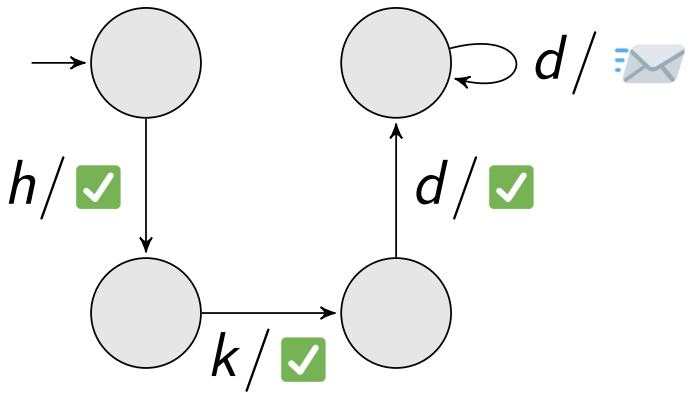
Overview

- Challenge: **scalability and efficiency** of automata learning
- Observation: lots of **irrelevant inputs words** leading to observable errors
- Our work: learning and testing with **(partial) knowledge of relevant inputs**

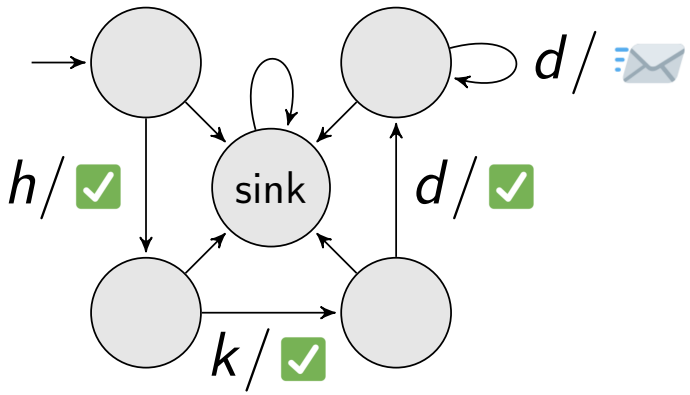
More details: paper at FM 2026

iCIS | Software Science
Radboud University

Automata Learning

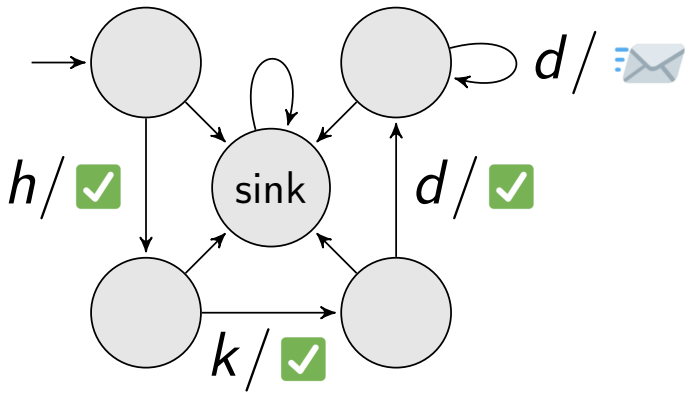


Automata Learning

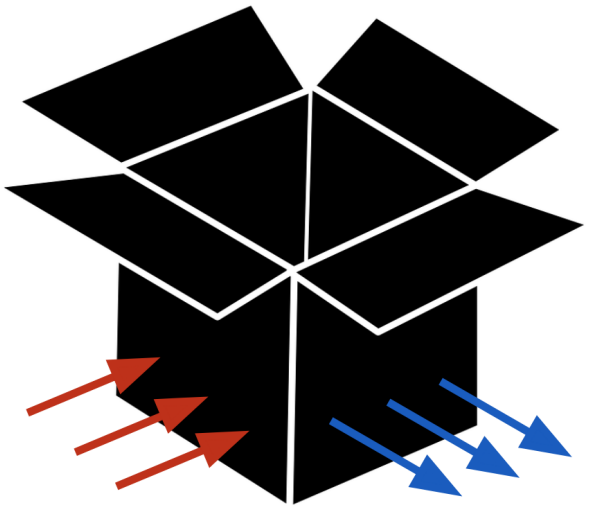


Mealy Machine

Automata Learning



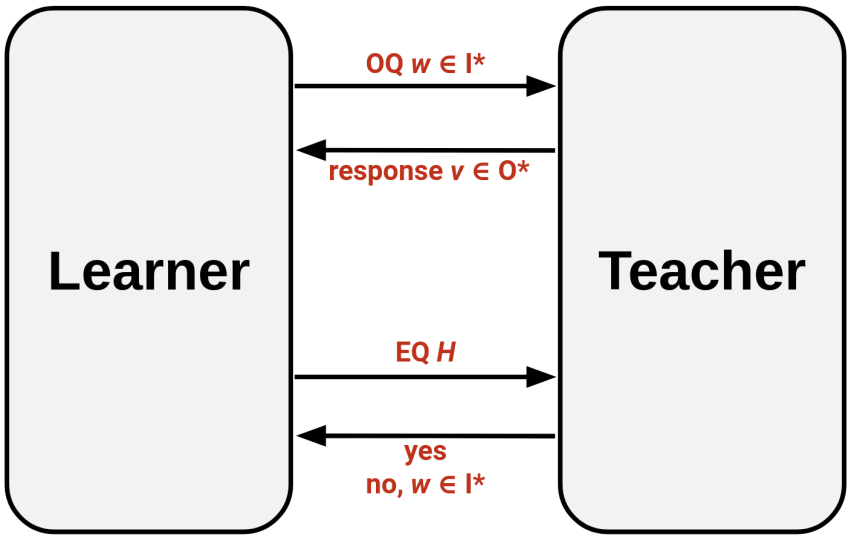
Mealy Machine

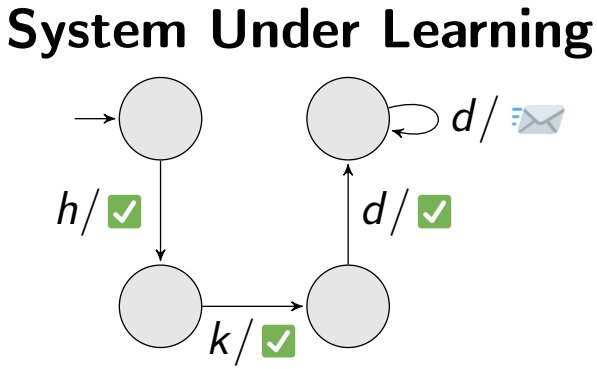


Black-box System

Active Automata Learning

The MAT framework proposed by Angluin (1987):

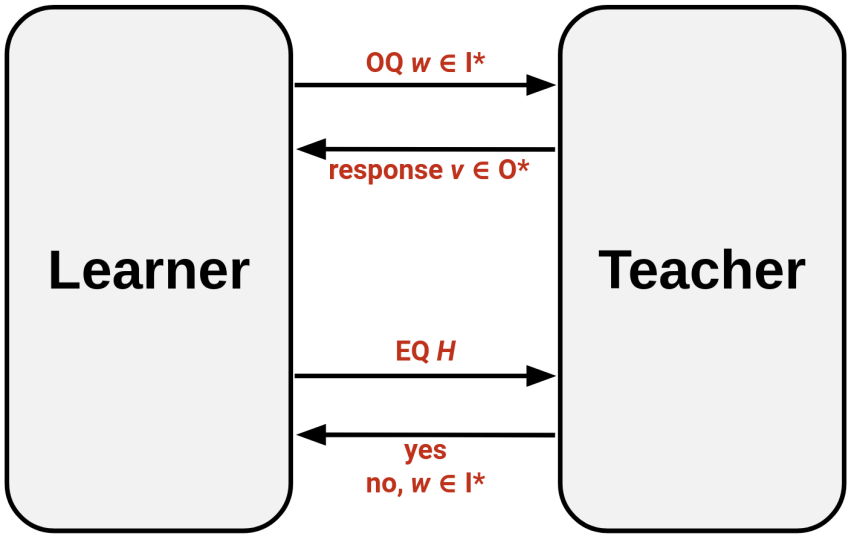




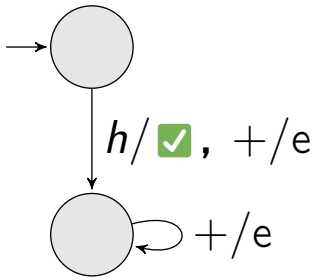
OQ: hh? e

Active Automata Learning

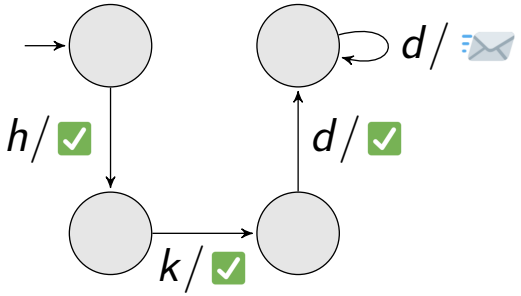
The MAT framework proposed by Angluin (1987):



Hypothesis



System Under Learning

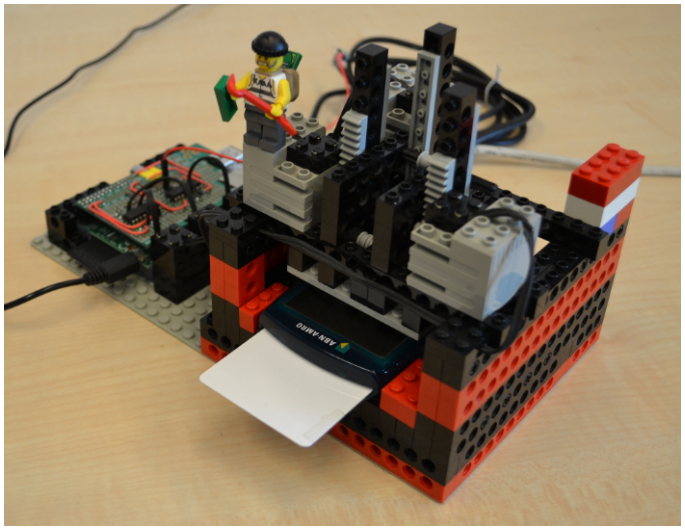


Counterexample hkh: ✓ee vs ✓✓e

iCIS | Software Science
Radboud University

Active Automata Learning Applications

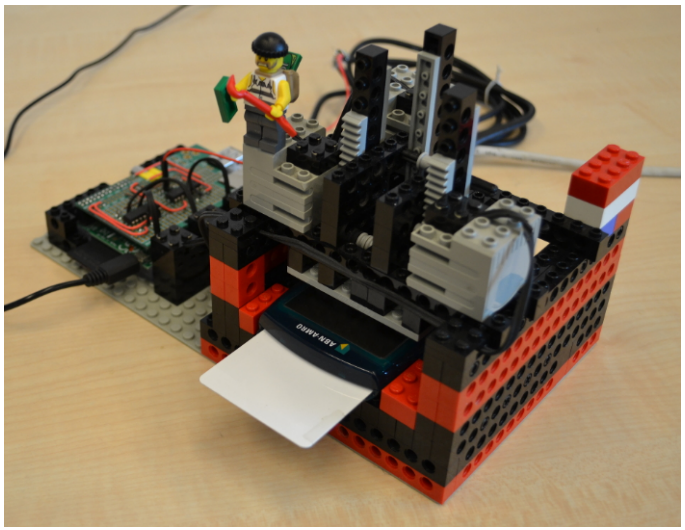
- Find bugs in protocols
- Fingerprint Bluetooth devices
- Reverse engineering of smartcard reader
- Learn an easy representation of a policy or monitor



iCIS | Software Science
Radboud University

Active Automata Learning Applications

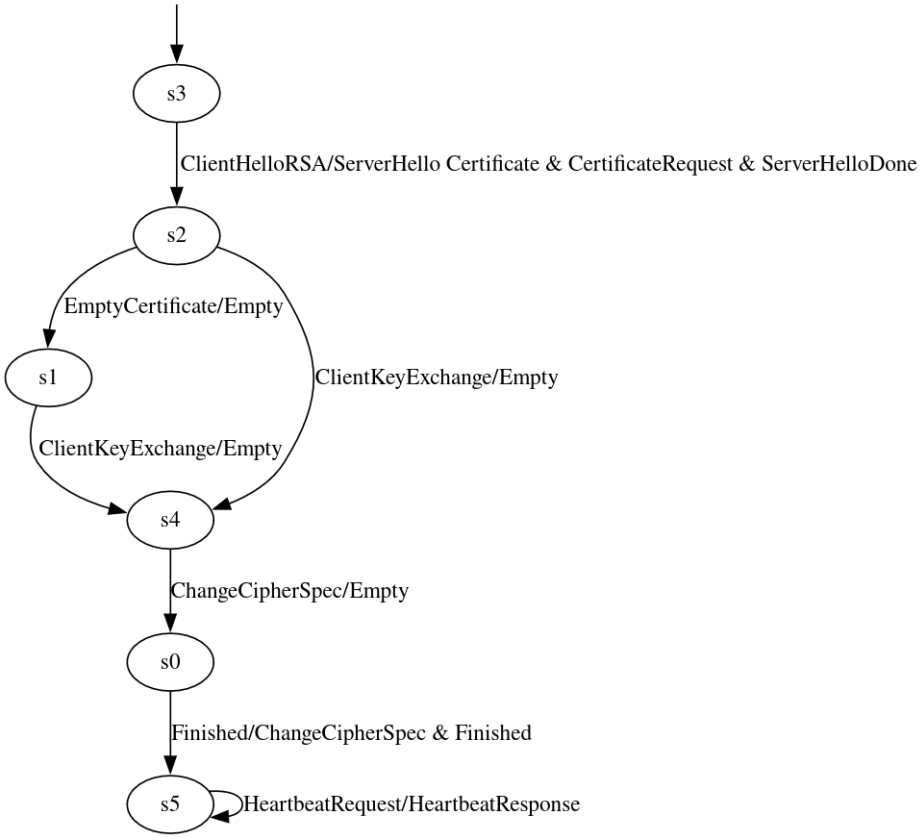
- Find bugs in protocols
- Fingerprint Bluetooth devices
- Reverse engineering of smartcard reader
- Learn an easy representation of a policy or monitor
- Learning complete models of real systems takes several days



iCIS | Software Science
Radboud University

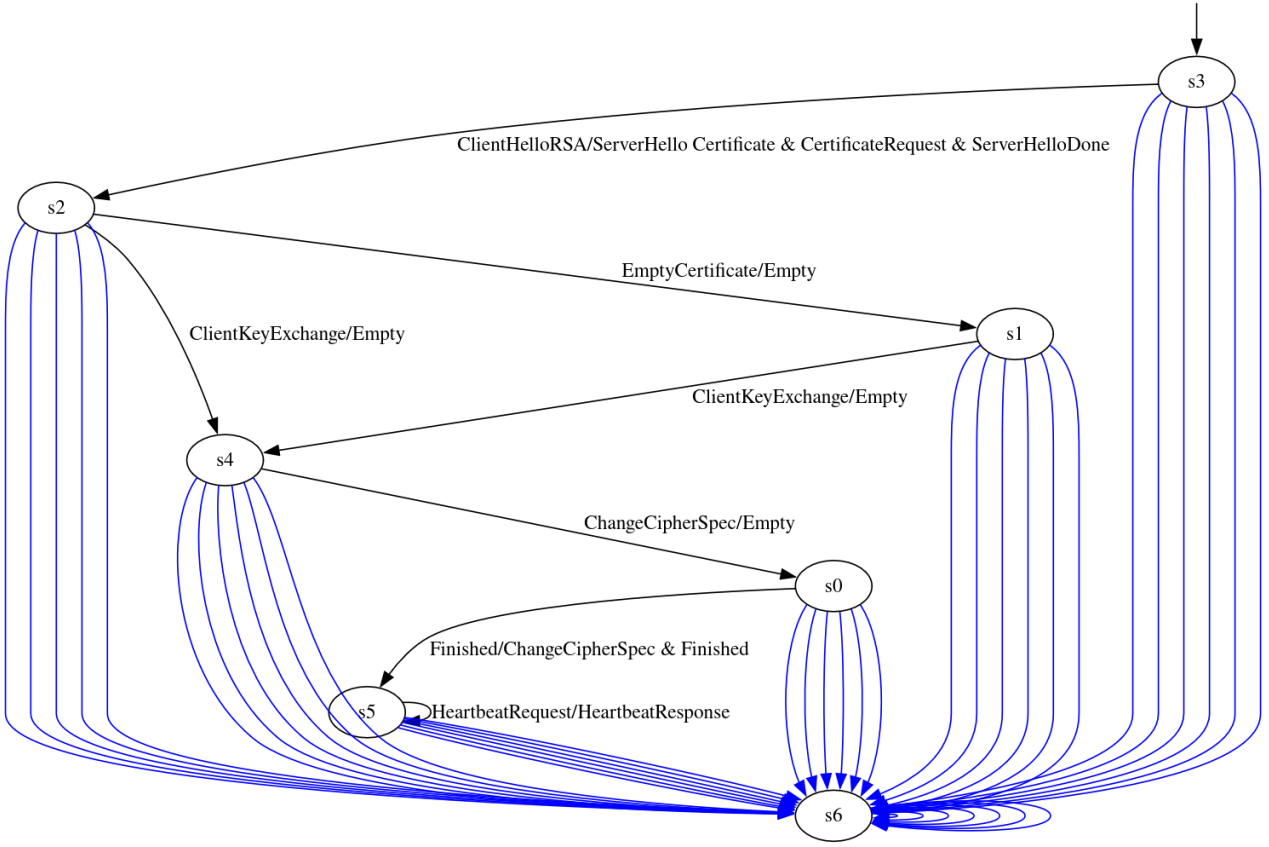
Application 1: Protocol Implementations

Protocol messages should be in a particular order



Application 1: Protocol Implementations

Protocol messages should be in a particular order



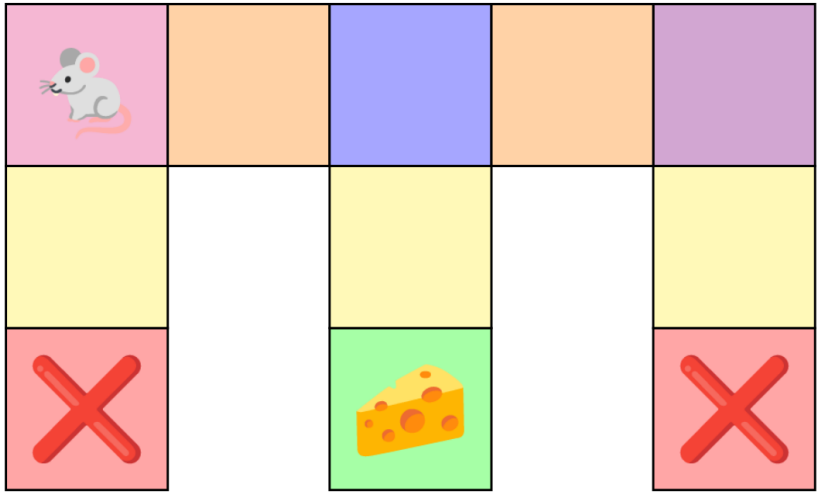
Application 2: Grid worlds

Given a sequence of color observations..

Policy : what is the best next action?

Monitor : is the current state dangerous?

Not all color observation sequences are valid



iCIS | Software Science
Radboud University

Error-aware Active Automata Learning

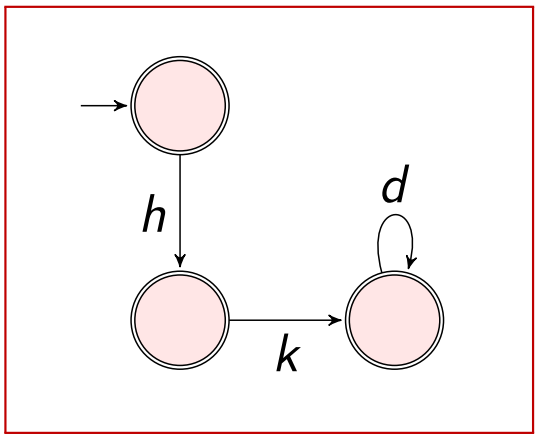
Most learning algorithms consider every input at every state . . .

Error-aware Active Automata Learning

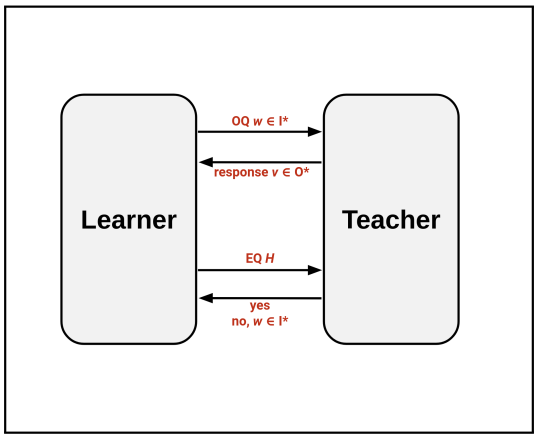
Most learning algorithms consider every input at every state ...

Key idea: use available domain knowledge about errors to avoid redundant queries!

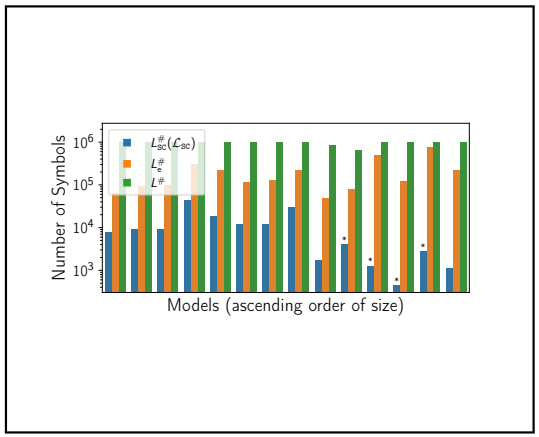
Outline



Types of Domain Knowledge



Adaptations to the MAT-framework



Experiments

iCIS | Software Science
Radboud University

Error-persistent Systems

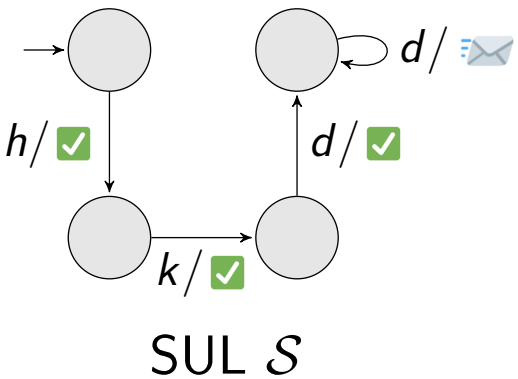
- Assumption: observable errors through some error output symbol e
  - Fatal alert
  - Closed connection
  - ...

Error-persistent Systems

- Assumption: observable errors through some error output symbol e
  - Fatal alert
  - Closed connection
  - ...
- A system is error-persistent if error outputs are followed only by error outputs

Reference Language

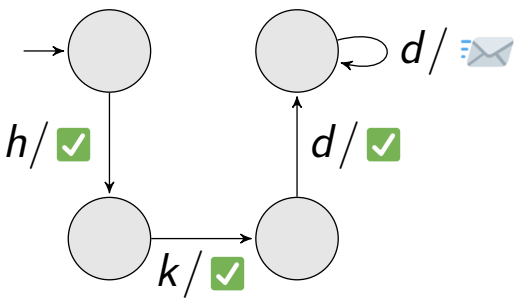
Reference language $\mathcal{L}$ for SUL $\mathcal{S}$: represents valid inputs (no error)



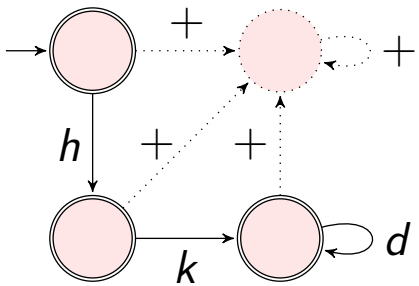
Reference Language

Reference language $\mathcal{L}$ for SUL $\mathcal{S}$: represents valid inputs (no error)

1. Sound and complete: word σ is not in $\mathcal{L}$ *iff* σ produces an error in $\mathcal{S}$



SUL $\mathcal{S}$

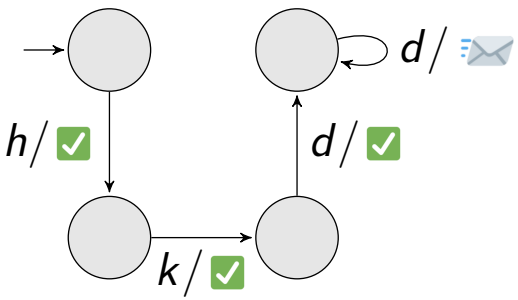


Sound and Complete

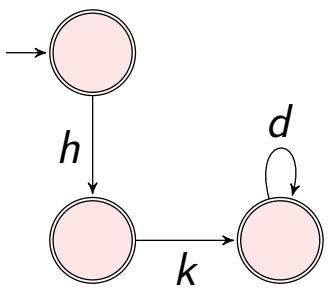
Reference Language

Reference language $\mathcal{L}$ for SUL $\mathcal{S}$: represents valid inputs (no error)

1. Sound and complete: word σ is not in $\mathcal{L}$ *iff* σ produces an error in $\mathcal{S}$



SUL $\mathcal{S}$

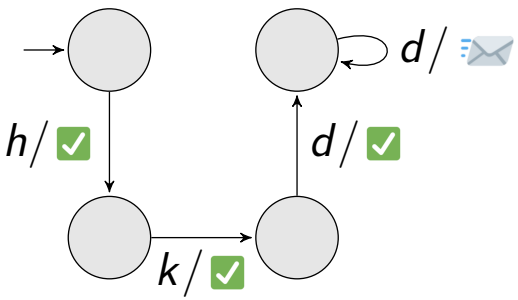


Sound and Complete

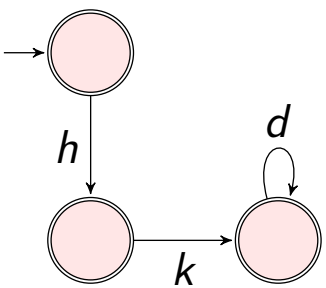
Reference Language

Reference language $\mathcal{L}$ for SUL $\mathcal{S}$: represents valid inputs (no error)

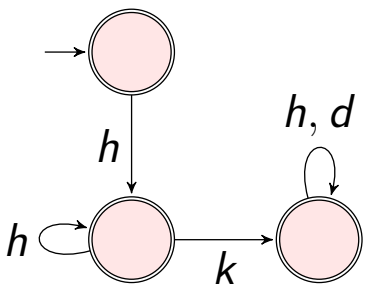
1. Sound and complete: word σ is not in $\mathcal{L}$ *iff* σ produces an error in $\mathcal{S}$
2. Sound: word σ is not in $\mathcal{L}$ *implies* σ produces an error in $\mathcal{S}$



SUL $\mathcal{S}$



Sound and Complete

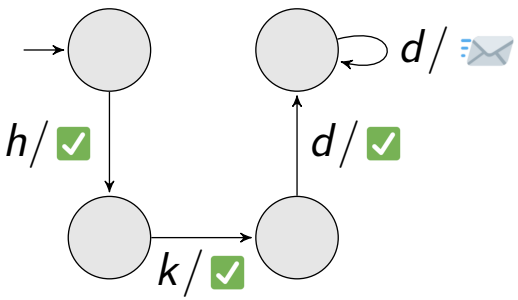


Sound

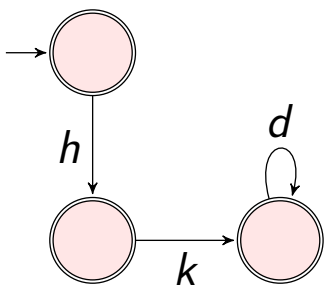
Reference Language

Reference language $\mathcal{L}$ for SUL $\mathcal{S}$: represents valid inputs (no error)

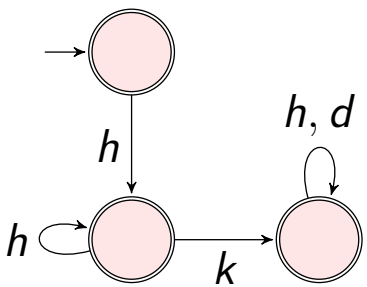
1. Sound and complete: word σ is not in $\mathcal{L}$ *iff* σ produces an error in $\mathcal{S}$
2. Sound: word σ is not in $\mathcal{L}$ *implies* σ produces an error in $\mathcal{S}$
3. Arbitrary: no formal relation



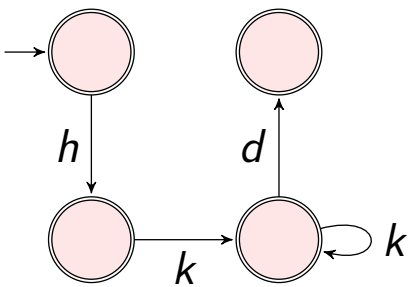
SUL $\mathcal{S}$



Sound and Complete



Sound



Arbitrary

iCIS | Software Science
Radboud University

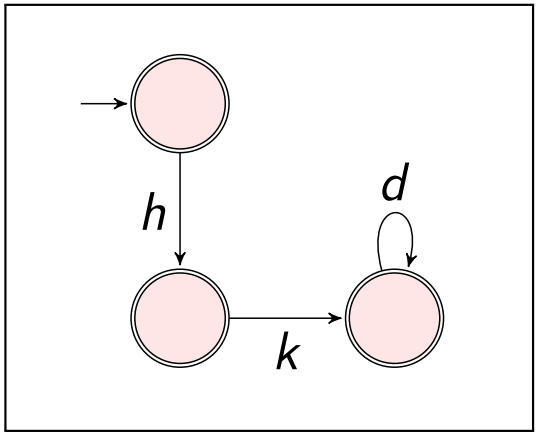
Problem Statement

Error-aware AAL with a Reference

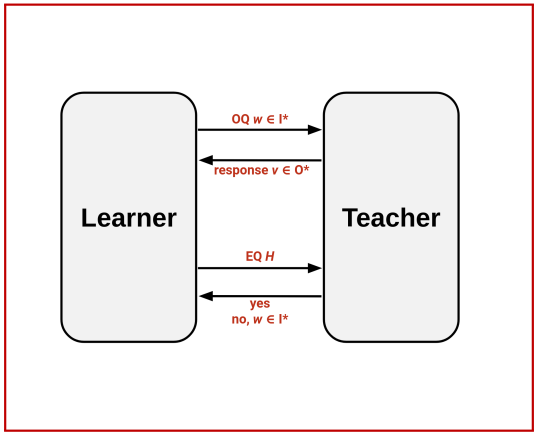
Given an e-persistent SUL $\mathcal{S}$, a sound and complete/sound/arbitrary reference language $\mathcal{L}$, efficiently learn a Mealy machine $\mathcal{H}$ s.t. $\mathcal{S} \approx \mathcal{H}$.

iCIS | Software Science
Radboud University

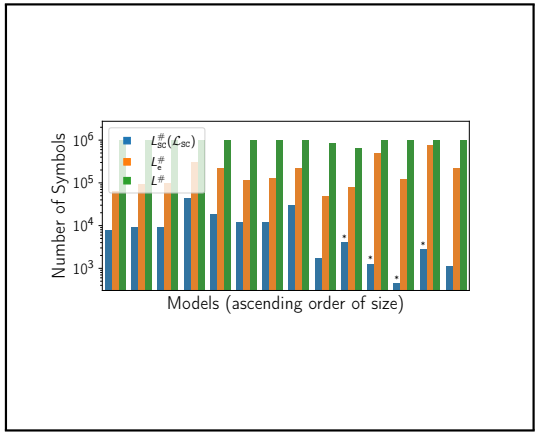
Outline



Types of Domain Knowledge



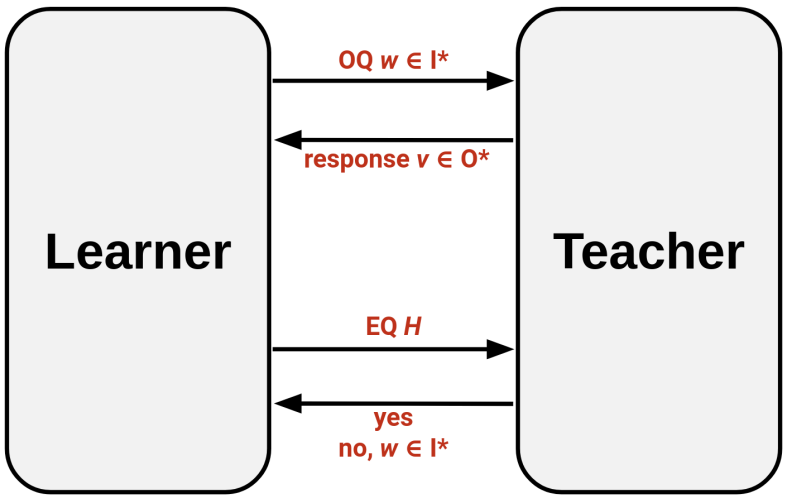
Adaptations to the MAT-framework



Experiments

Conformance Testing

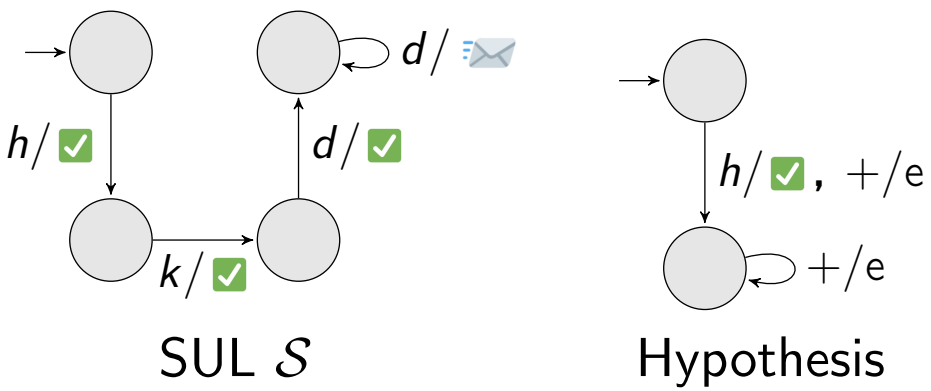
- EQs are approximated using conformance testing
- Construct a test suite from hypothesis $\mathcal{H}$
- Goal: Find a counterexample



iCIS | Software Science
Radboud University

Conformance Testing

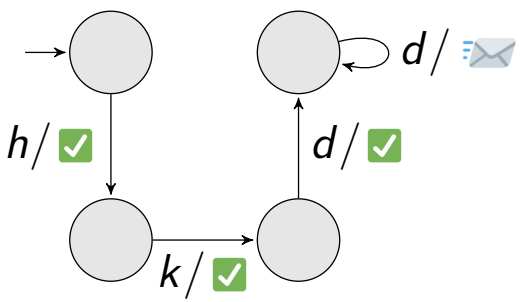
- EQs are approximated using conformance testing
- Construct a test suite from hypothesis $\mathcal{H}$
- Goal: Find a counterexample
- $T = \{kh, dh, ch, hhh, hkh, hdh, hch\}$



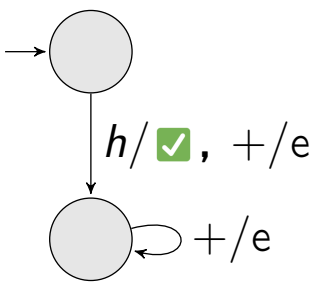
SUL $\mathcal{S}$ Hypothesis

Conformance Testing with e-persistence

- e-persistent system: any e output is followed only be e
- If $\mathcal{H}$ and $\mathcal{S}$ e-persistent, there are *no* counterexamples after seeing e
- Truncate a test sequence after observing the first e
- $T = \{kh, dh, ch, hhh, hkh, hdh, hch\}$
- $T_e = \{k, d, c, hh, hk, hd, hc\}$



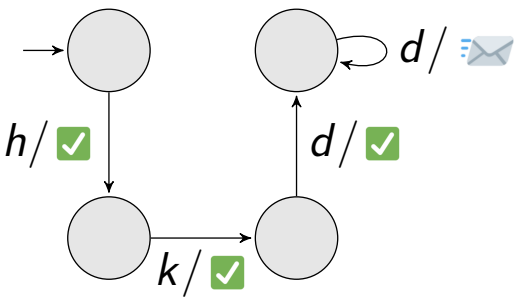
SUL $\mathcal{S}$



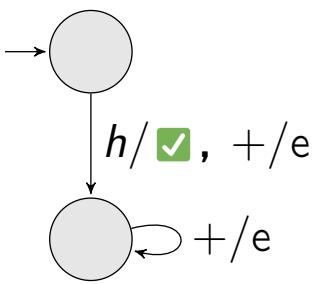
Hypothesis

Conformance Testing with a Sound Reference

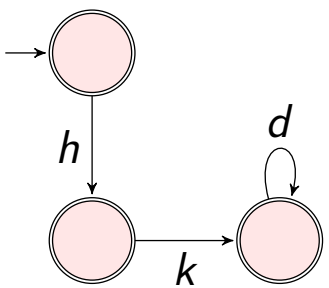
- Sound: if σ is not in $\mathcal{L}$, then σ produces an error in $\mathcal{S}$
- Truncate a test sequence before observing the first e
- $T = \{kh, dh, ch, hhh, hkh, hdh, hch\}$
- $T_{\mathrm{e}} = \{k, d, c, hh, hk, hd, hc\}$
- $T_{\mathrm{S}} = \{hk\}$



SUL $\mathcal{S}$



Hypothesis



Sound Reference $\mathcal{L}$

Testing & Learning Adaptations

Domain Knowledge	Algorithm	Explanation
e-persistence	T_{e}	Truncate to after the error
Sound (Complete) Ref	T_{S}	Truncate to before the error
Arbitrary Ref	MoE($T_{\mathrm{e}}, T_{\mathrm{S}}$)	Combine T_{e} and T_{S}

Testing & Learning Adaptations

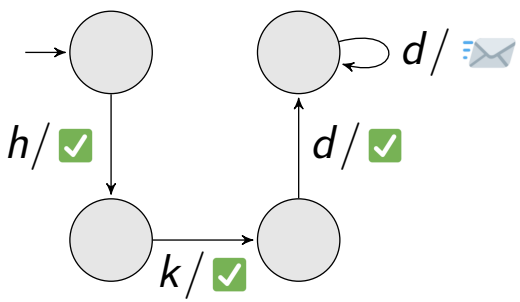
Domain Knowledge	Algorithm	Explanation
e-persistence	T_{e}	Truncate to after the error
Sound (Complete) Ref	T_{S}	Truncate to before the error
Arbitrary Ref	MoE($T_{\mathrm{e}}, T_{\mathrm{S}}$)	Combine T_{e} and T_{S}

Domain Knowledge	Algorithm	Explanation
e-persistence	$L^{\#}_{\mathrm{e}}$	All e transitions go to a sink
Arbitrary Ref	$AL^{\#}_{\mathrm{e}}$	Adaptive learning with rebuilding
Sound Ref	$L^{\#}_{\mathrm{S}}$	S-apartness
Sound Complete Ref	$L^{\#}_{\mathrm{SC}}$	SC-apartness and rebuilding

iCIS | Software Science
Radboud University

$L^{\#}$ Algorithm

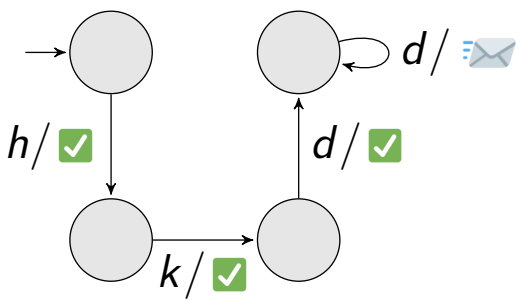
- Based on $L^{\#}$ (Vaandrager et al., 2022)
- Separates states based on apartness, $h \vdash p \# q$



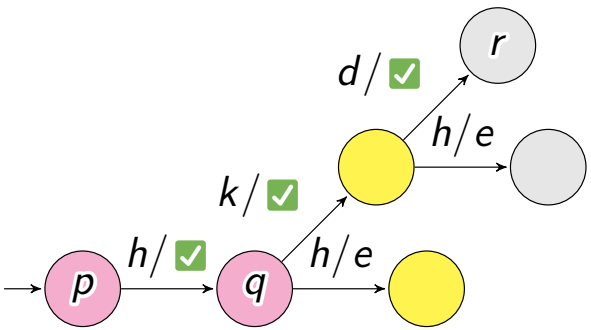
SUL $\mathcal{S}$

$L^\#$ Algorithm

- Based on $L^\#$ (Vaandrager et al., 2022)
- Separates states based on apartness, $h \vdash p \# q$
- Maintains an observation tree
- Basis: distinct states in the hidden SUL
- Frontier: Successors of basis states
- Goal: make basis complete and identify all frontier states



SUL $\mathcal{S}$

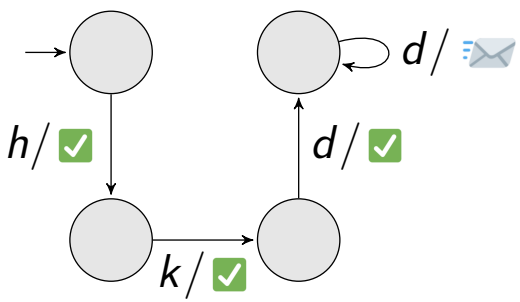


Observation tree

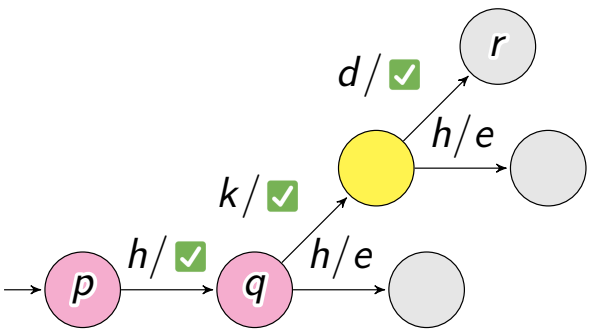
iCIS | Software Science
Radboud University

$L^{\#}$ with e-persistence

- Frontier: Successors of basis states that are not reached by errors
- Redirect e-transitions to a sink state



SUL $\mathcal{S}$

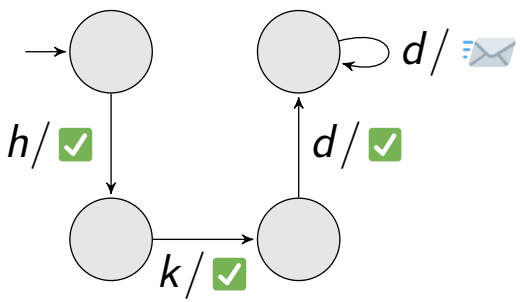


Observation tree

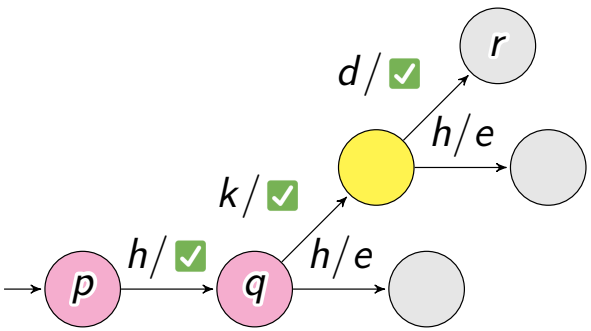
$L^\#$ with a Sound Reference

A sound reference can be used to show apartness

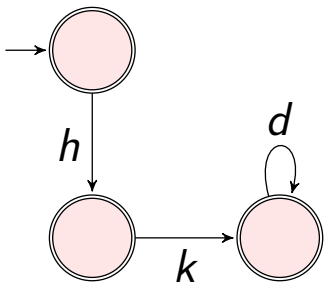
- $p \xrightarrow{h/\checkmark}$
- State r reached by hkd and $hkd \cdot h \notin \mathcal{L}$
- According to $\mathcal{L}$, $r \xrightarrow{h/e}$



SUL $\mathcal{S}$



Observation tree



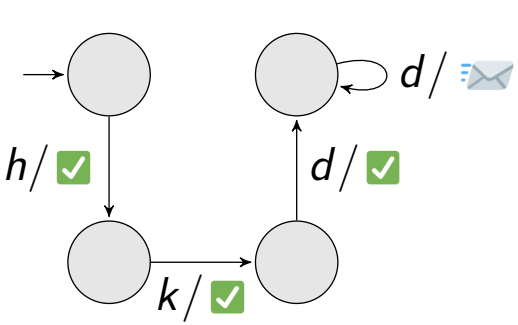
Sound Reference $\mathcal{L}$

iCIS | Software Science
Radboud University

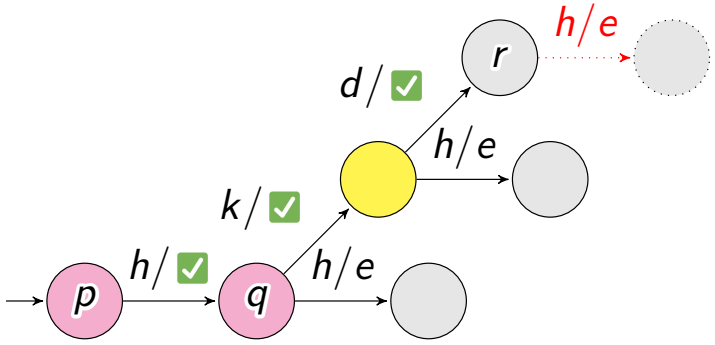
$L^{\#}$ with a Sound Reference

A sound reference can be used to show apartness

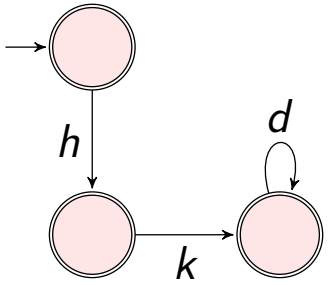
- $p \xrightarrow{h/\checkmark}$
- State r reached by hkd and $hkd \cdot h \notin \mathcal{L}$
- According to $\mathcal{L}$, $r \xrightarrow{h/e}$
- $h \vdash p \# r$



SUL $\mathcal{S}$



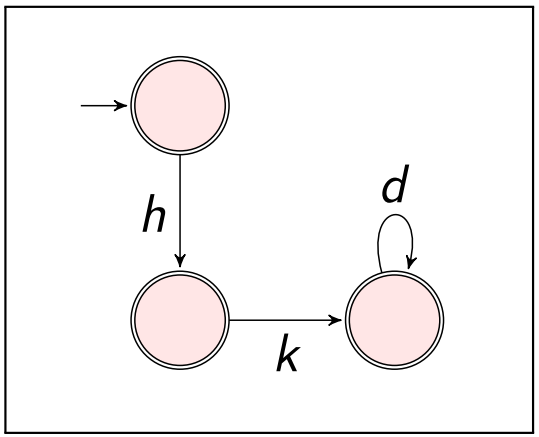
Observation tree



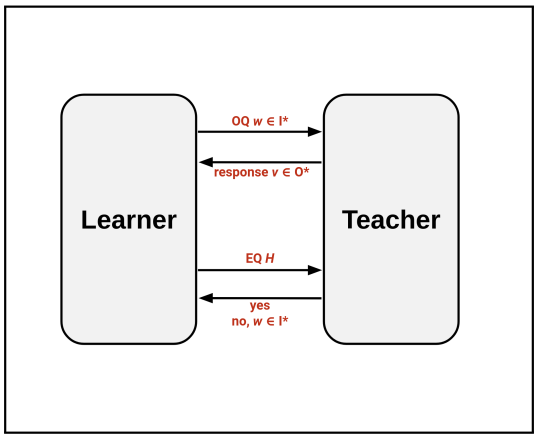
Sound Reference $\mathcal{L}$

iCIS | Software Science
Radboud University

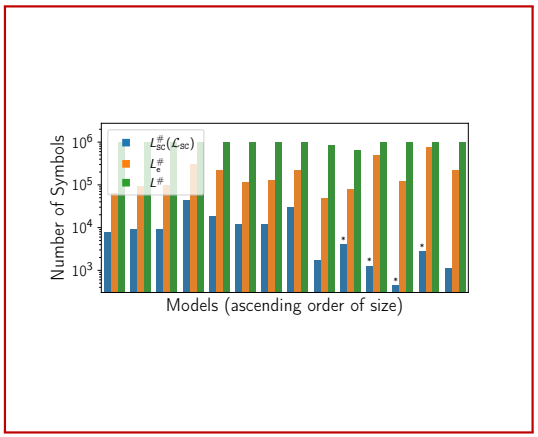
Outline



Types of Domain Knowledge



Adaptations to the MAT-framework



Experiments

iCIS | Software Science
Radboud University

Benchmarks

Target systems for learning: TLS, monitors, ASML models

Reference models obtained by:

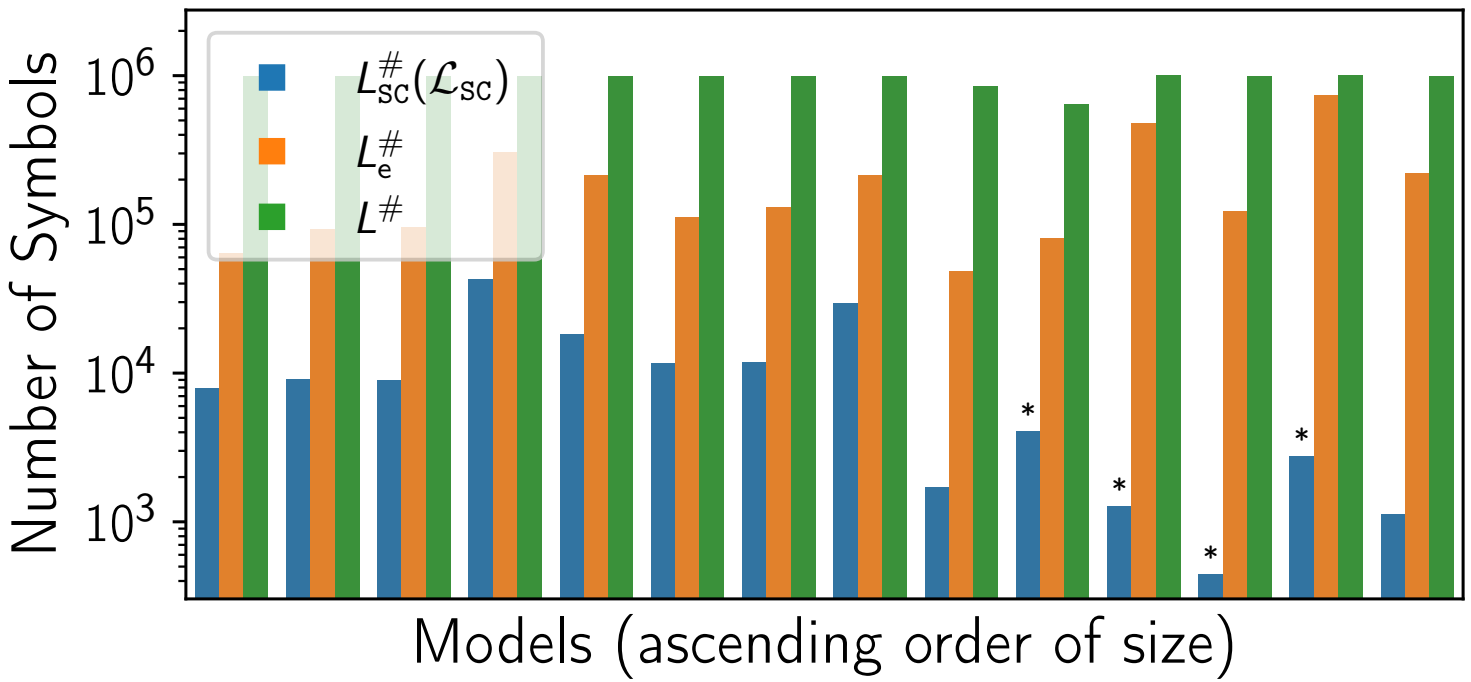
- Sound references from expert (analysing TLS specification)
- chatGPT-generated references from specification
- Passive learning
- Monitors: from underlying hidden Markov model
- Knowing the black box ...

Experiment 1

Baseline ($L^{\#}$) vs error-aware ($L^{\#}_{\mathrm{e}}$) vs sound & complete ($L^{\#}_{\mathrm{SC}}$)

Benchmark: monitors with 21-75 states, 18-325 inputs

References: Learned or inferred from underlying model



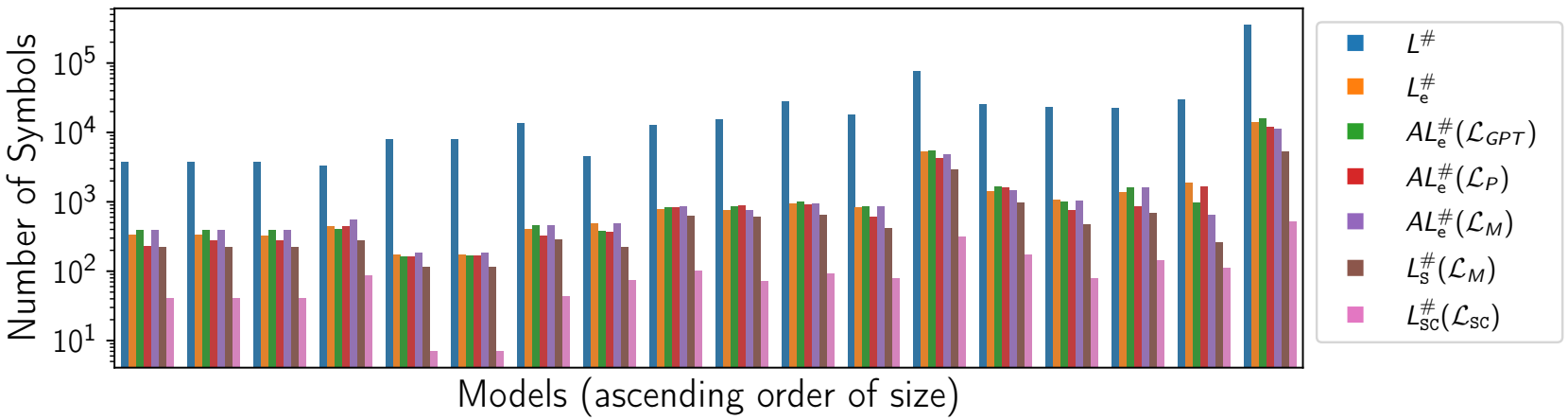
iCIS | Software Science
Radboud University

Experiment 2

Error-aware ($L_e^\#$) vs Arbitrary ($AL_e^\#$) vs Sound ($L_S^\#$)

Benchmark: TLS with 6-16 states and 8-11 inputs

References: Manual inspection specification, chatGPT or passive learning



iCIS | Software Science
Radboud University

Conclusion

Take home message: Use domain knowledge about errors!

Future work:

- Different definitions of errors
- Investigate sources of (sound) error-aware references

Domain Knowledge	Algorithm	Explanation
e-persistence	T_{e}	Truncate to after the error
Sound (Complete) Ref	T_{S}	Truncate to before the error
Arbitrary Ref	MoE(T_{e}, T_{S})	Combine T_{e} and T_{S}
e-persistence	$L^{\#}_{\mathrm{e}}$	All e transitions go to a sink
Arbitrary Ref	$AL^{\#}_{\mathrm{e}}$	Adaptive learning
Sound Ref	$L^{\#}_{\mathrm{S}}$	S-apartness
Sound Complete Ref	$L^{\#}_{\mathrm{SC}}$	SC-apartness and rebuilding

Paper

Artifact

iCIS | Software Science
Radboud University