Small Test Suites for Model Learning

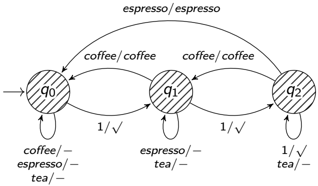
Loes Kruger
Testing Techniques
December 20, 2024

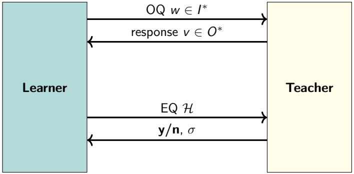
loes.kruger@ru.nl

iCIS | Software Science
Radboud University

Previous Lectures

- Model learning with L^* and $L^\#$
- Black-box testing





$$T = A \cdot I^{\leq k+1} \cdot C$$

Implementing the Equivalence Query

In practice, EQs are approximated

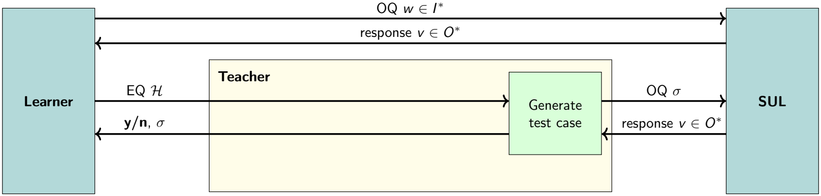
- Random walks

 loes.kruger@ru.nl

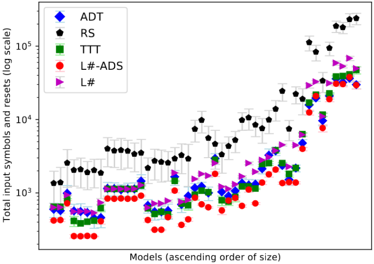
Implementing the Equivalence Query

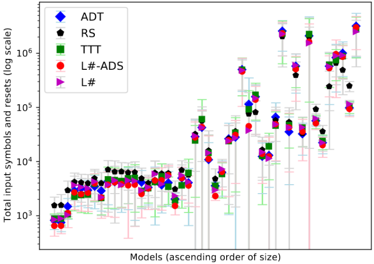
In practice, EQs are approximated

- Random walks
- W-method



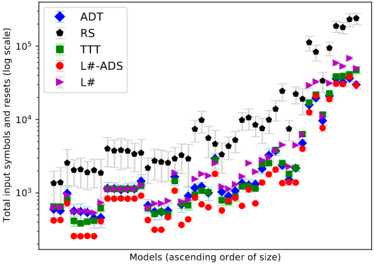
Testing as Bottleneck



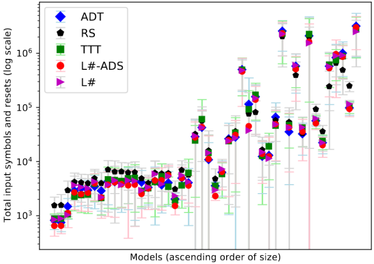


(a) Symbols used during learning phase (b) Symbols used both learning and testing

Testing as Bottleneck



(a) Symbols used during learning phase



(b) Symbols used both learning and testing

Rust demo!

Outline

Small Test Suites for Active Automata Learning
(Loes Kruger, Sebastian Junges and Jurriaan Rot)

- $L^{\#}$ + W-method example with a large test suite
- Ways to make test suites smaller
- Experimental results

Outline

Small Test Suites for Active Automata Learning
(Loes Kruger, Sebastian Junges and Jurriaan Rot)

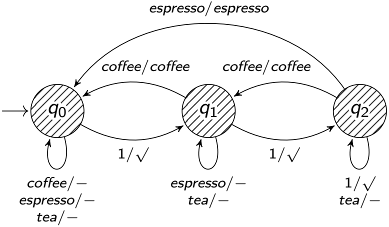
- $L^{\#}$ + W-method example with a large test suite
- Ways to make test suites smaller
- Experimental results

Learning goals

- Give some intuition about model learning in practice
- Testing is the bottleneck
- Active research field

System Under Learning

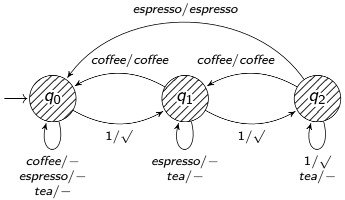
- $I = \{coffee, espresso, tea, 1\}$
- $O = \{coffee, espresso, -, \surd\}$



$L^{\#}$ Recap

The $L^{\#}$ algorithm has four rules:

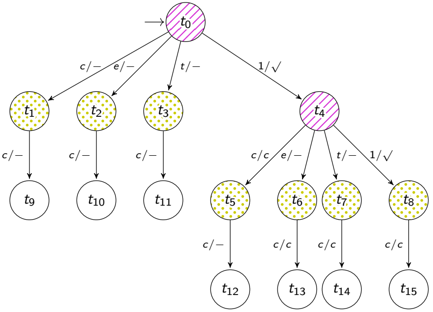
- **Promotion**: Move a frontier state to the basis if it is apart from all basis states
- **Extension**: Ensure each basis state has a transition with each input
- **Identification**: Identify all frontier states
- **Equivalence**: Build $\mathcal{H}$, ask EQ, process counterexample



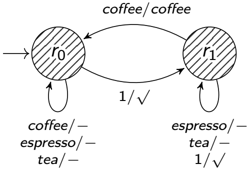
$L^{\#}$ Example

Example on the blackboard!

Example



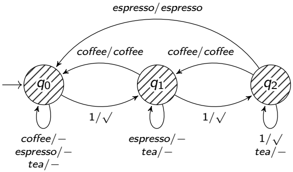
t_0
c/— e/— t/— 1/√
t_1 t_2 t_3 t_4
c/— c/— c/— c/c e/— t/— 1/√
t_9 t_{10} t_{11} t_5 t_6 t_7 t_8
c/— c/c c/c c/c
t_{12} t_{13} t_{14} t_{15}

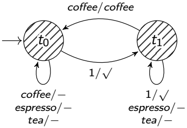


coffee/ coffee
r_0
r_1
1/√
coffee/—
espresso/—
tea/—
espresso/—
tea/—
1/√

Black-Box Testing

$$A \quad \cdot \quad I^{\leq k+1} \quad \cdot \quad C$$



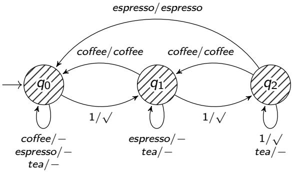


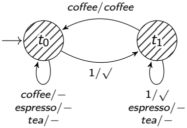
iCIS | Software Science
Radboud University

Black-Box Testing

$$A \quad \cdot \quad I^{\leq k+1} \quad \cdot \quad C$$

access · step · separate





Building the Characterization Set

Splitting tree example on the blackboard!

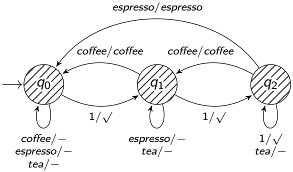
iCIS | Software Science
Radboud University

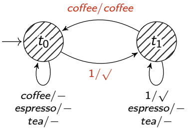
Black-Box Testing

$A \quad \cdot \quad I^{\leq k+1} \quad \cdot \quad C$

access · step · separate

$\{\varepsilon, 1\} \cdot \{coffee, 1, tea, espresso\}^{\leq k+1} \cdot \{coffee\}$

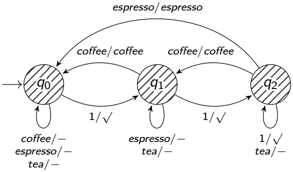




Black-Box Testing

$$A \quad \cdot \quad I^{\leq k+1} \quad \cdot \quad C$$

access · step · separate

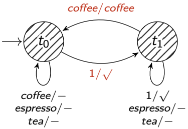


$\{\varepsilon, 1\} \cdot \{coffee, 1, tea, espresso\}^{\leq k+1} \cdot \{coffee\}$

Consider $k = 0$

$T = \{c, e, t, 1, cc, ec, tc, 1c, 1cc, 1ec, 1tc, 11c\}$

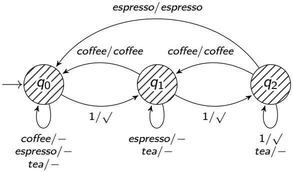
$|T| = 12$



Black-Box Testing

$$A \quad \cdot \quad I^{\leq k+1} \quad \cdot \quad C$$

access · step · separate

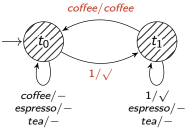


$\{\varepsilon, 1\} \cdot \{coffee, 1, tea, espresso\}^{\leq k+1} \cdot \{coffee\}$

Consider $k = 0$

$T = \{c, e, t, 1, cc, ec, tc, 1c, 1cc, 1ec, 1tc, 11c\}$

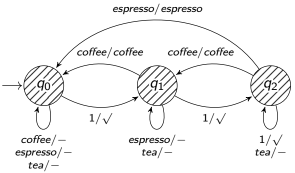
$|T| = 12$

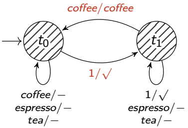


Counterexample $11cc$ not in T!

Test Suite Size

$\{\varepsilon, 1\} \cdot \{\mathit{coffee}, 1, \mathit{tea}, \mathit{espresso}\}^{\leq k+\mathbf{1}} \cdot \{\mathit{coffee}\}$

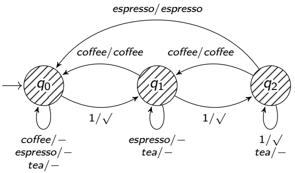


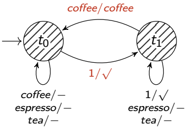


Test Suite Size

$\{\varepsilon, 1\} \cdot \{\textit{coffee}, 1, \textit{tea}, \textit{espresso}\}^{\leq k+1} \cdot \{\textit{coffee}\}$

$T_{k=0} = \{c, e, t, 1, cc, ec, tc, 1c, 1cc, 1ec, 1tc, 11c\}$
$|T_{k=0}| = 12$



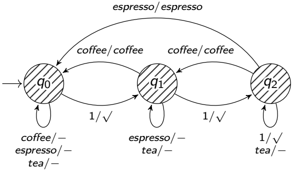


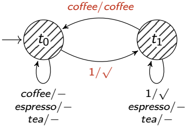
Test Suite Size

$\{\varepsilon, 1\} \cdot \{\mathit{coffee}, 1, \mathit{tea}, \mathit{espresso}\}^{\leq k+1} \cdot \{\mathit{coffee}\}$

$T_{k=0} = \{c, e, t, 1, cc, ec, tc, 1c, 1cc, 1ec, 1tc, 11c\}$
$|T_{k=0}| = 12$

$T_{k=1} = T_{k=0} + \{ccc, cec, ctc, c1c, ecc, eec, etc, e1c, tcc, tec, ttc, t1c, 1cc, 1ec, 1tc, 11c, ...\}$
$|T_{k=1}| = 12 + 32 = 40$

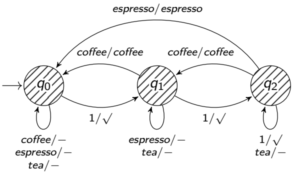




iCIS | Software Science
Radboud University

Test Suite Size

$\{\varepsilon, 1\} \cdot \{coffee, 1, tea, espresso\}^{\leq k+1} \cdot \{coffee\}$



$T_{k=0} = \{c, e, t, 1, cc, ec, tc, 1c, 1cc, 1ec, 1tc, 11c\}$

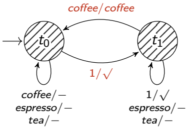
$|T_{k=0}| = 12$

$T_{k=1} = T_{k=0} + \{ccc, cec, ctc, c1c, ecc, eec, etc,$
$e1c, tcc, tec, ttc, t1c, 1cc, 1ec, 1tc, 11c, ...\}$

$|T_{k=1}| = 12 + 32 = 40$

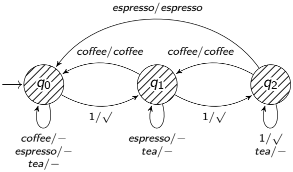
$T_{k=2} = T_{k=0} + T_{k=1} +$
$\{cccc, ccec, cctc, cc1c, cecc, ctcc, c1cc, ceec, ...\}$

$|T_{k=2}| = 12 + 32 + 128 = 168$



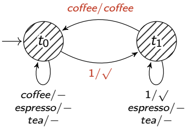
iCIS | Software Science
Radboud University

Smaller Test Suite



$\{\varepsilon, 1\} \cdot \{coffee, 1, tea, espresso\}^{\leq k+1} \cdot \{coffee\}$

- Inputs *espresso* and *tea* self-loop in $\mathcal{H}$
- Possible counterexample $11cc$
- Use $I_{sub}^{\leq k+1}$ for $I^{\leq k+1}$
- Test suite size for $k = 2$ only 30!



Compromising k-completeness

- k-completeness:
  1. SUL has at most k additional states,
  2. Test suite passed

  $\longrightarrow$ SUL and $\mathcal{H}$ equivalent

iCIS | Software Science
Radboud University

Compromising k-completeness

- k-completeness:
  1. SUL has at most k additional states,
  2. Test suite passed

  $\longrightarrow$ SUL and $\mathcal{H}$ equivalent
- Test suite too large to be completely executed
- Especially when learning on an actual system!
- Shift towards find counterexamples faster
- *From ZULU to RERS: Lessons Learned in the ZULU Challenge (Howar et al, 2011)*

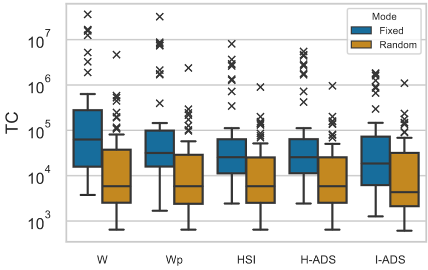
Side Note on k-completeness

- Using fault-domains with at most k extra states is quite arbitrary
- Based on work by Moore, Hennie, and Chow
- Restricted fault-domain based on the specification
- k-A-completeness where A is a set of common scenarios
- *Completeness of FSM Test Suites Reconsidered (Vaandrager et al, 2024)*

iCIS | Software Science
Radboud University

Finding Counterexamples Fast

- Random walk in the infix
- No guarantees
- Rust demo!



An Experimental Evaluation of Conformance Testing Techniques in Active Automata Learning (Garhewal and Damasceno, 2023)

Approach

Ask more promising tests first (based on important inputs)

Approach

Ask more promising tests first (based on important inputs)

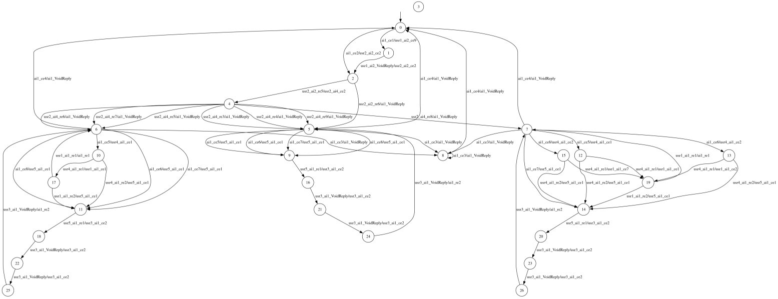
1. Define experts to generate subalphabet based on $\mathcal{H}$
2. Generate expert test suite
3. Combine test suites using multi-armed bandits

 loes.kruger@ru.nl

iCIS | Software Science
Radboud University

ASML Model 199

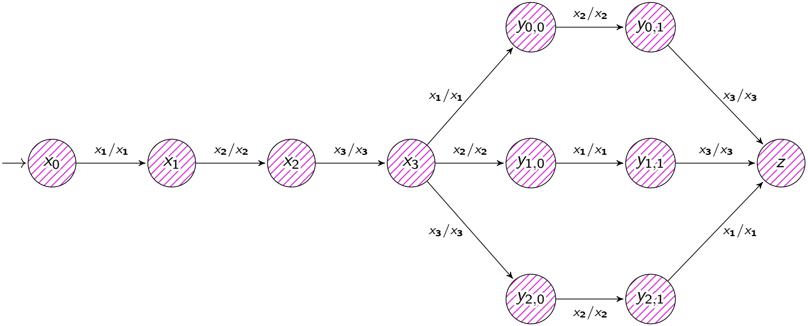
- 27 states, 56 inputs but only 24 useful
- Originally a partial model



The RERS 2017 challenge (Jasper et al, 2017)

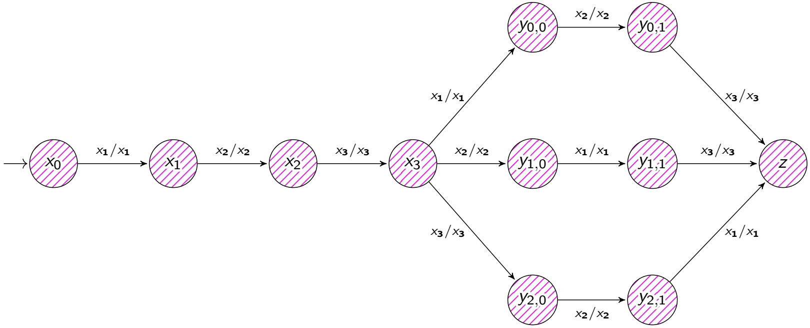
Removing Uninformative Inputs

$I = \{x_1, x_2, x_3\}$



Removing Uninformative Inputs

$I = \{x_1, x_2, x_3\}$

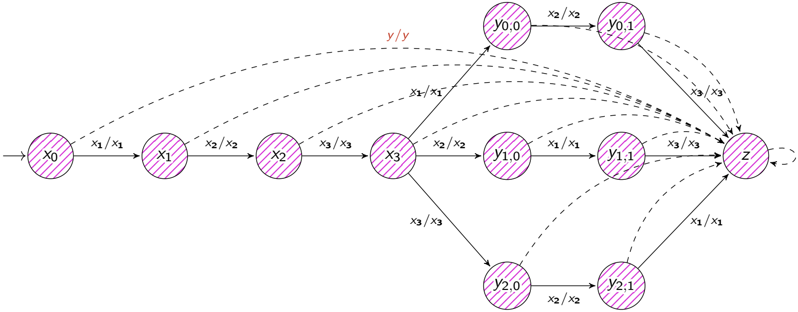


k	I
2	674
4	6071
6	54659

loes.kruger@ru.nl

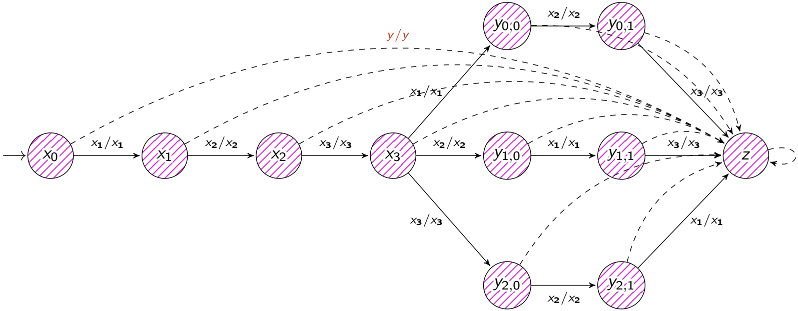
Removing Uninformative Inputs

$I = \{x_1, x_2, x_3\}$ and uninformative y



Removing Uninformative Inputs

$I = \{x_1, x_2, x_3\}$ and uninformative y



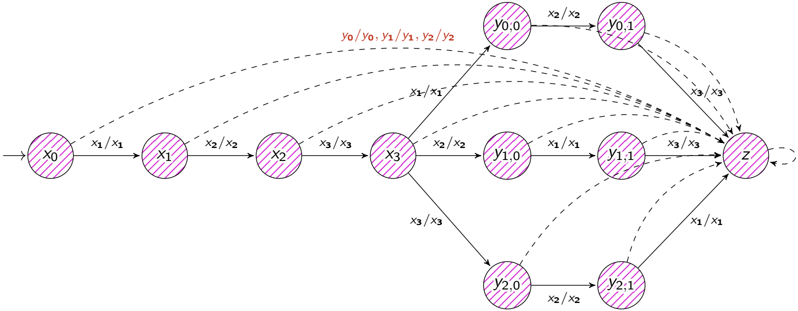
k	I	$I \cup \{y\}$
2	674	1775
4	6071	28409
6	54659	454638

 loes.kruger@ru.nl

iCIS | Software Science
Radboud University

Removing Uninformative Inputs

$I = \{x_1, x_2, x_3\} \cup \{y_0, y_1, y_2\}$

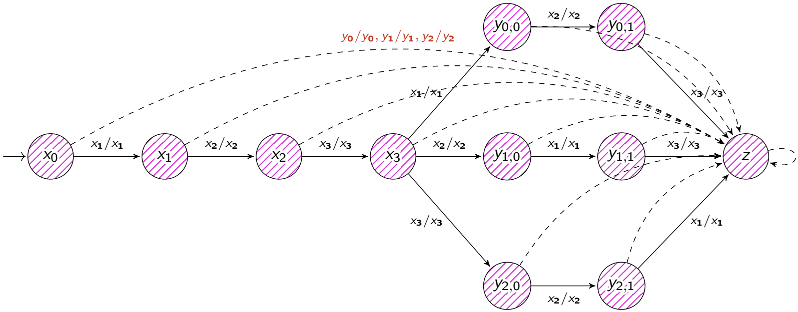


k	I	$I \cup \{y\}$	$I \cup \{y_0, y_1, y_2\}$
2	674	1775	6587
4	6071	28409	237161
6	54659	454638	8538030

loes.kruger@ru.nl

Removing Uninformative Inputs

Only consider active inputs

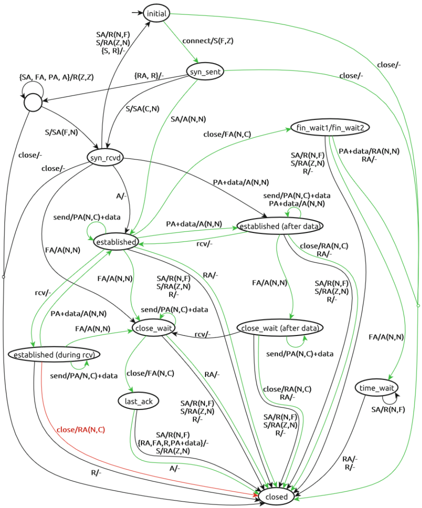


k	I	$I \cup \{y\}$	$I \cup \{y_0, y_1, y_2\}$	I_{act}
2	674	1775	6587	3347
4	6071	28409	237161	30341
6	54659	454638	8538030	284220

$$A \cdot I_{act}^{\leq k+1} \cdot C$$

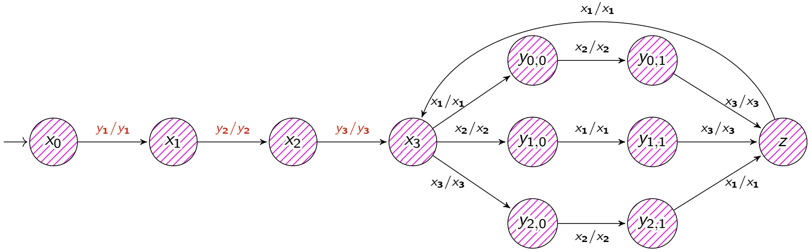
TCP models

1. Establish connection
2. Send data



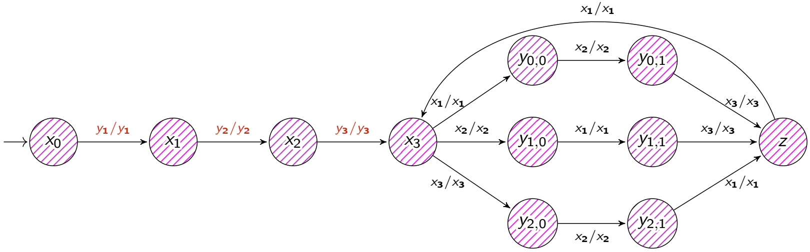
Combining Model Learning and Model Checking to Analyze TCP Implementations (Fiterău-Broştean et al., 2017)

Remove Inputs Uninformative In The Future



loes.kruger@ru.nl

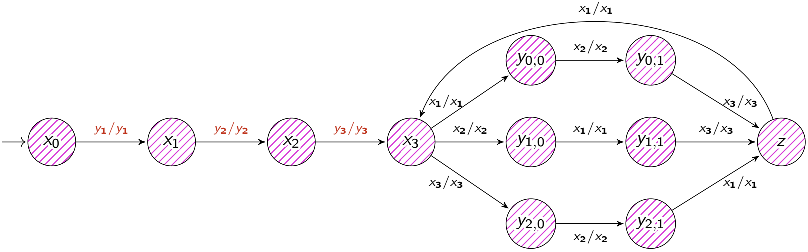
Remove Inputs Uninformative In The Future



k	full
2	13137
4	474294
6	17075979

iCIS | Software Science
Radboud University

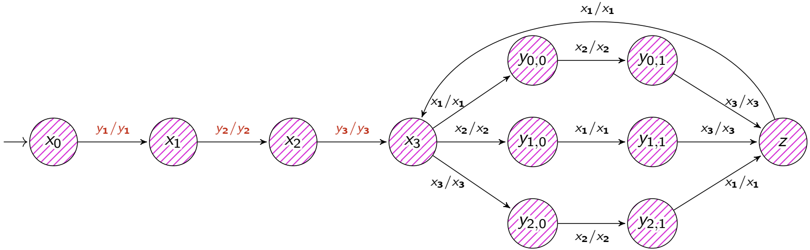
Remove Inputs Uninformative In The Future



k	full	active
2	13137	13137
4	474294	474294
6	17075979	17075979

Remove Inputs Uninformative In The Future

Only consider inputs active in the future

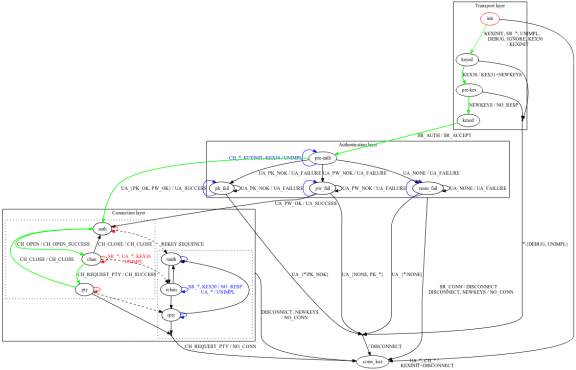


k	full	active	future
2	13137	13137	6135
4	474294	474294	135015
6	17075979	17075979	2917455

iCIS | Software Science
Radboud University

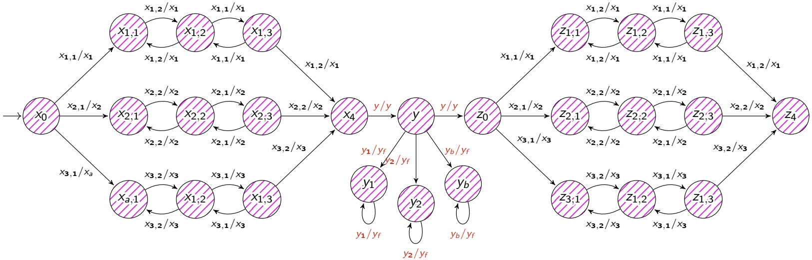
SSH models

Groups of states repeat



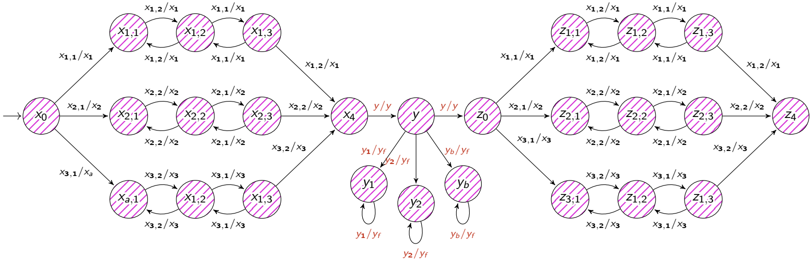
Model Learning and Model Checking of SSH Implementations (Fiterău-Broştean et al, 2017)

Select Inputs Based on Components



iCIS | Software Science
Radboud University

Select Inputs Based on Components

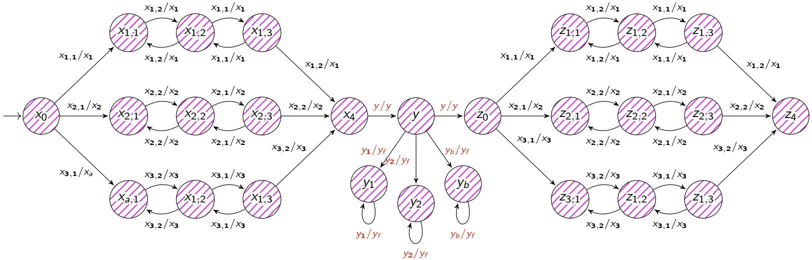


k	full	active	future
2	234783	234783	62933
4	23499548	23499548	4521248

loes.kruger@ru.nl

iCIS | Software Science
Radboud University

Select Inputs Based on Components



k	full	active	future	component
2	234783	234783	62933	106106
4	23499548	23499548	4521248	1743082

loes.kruger@ru.nl

Expert Selection

Expert: reduced test suite generator: $A \cdot I_{sub}^{\leq k+1} \cdot C$

- Remove uninformative inputs
- Remove future uninformative inputs
- Select inputs based on components
- (Use full alphabet)

 loes.kruger@ru.nl

iCIS | Software Science
Radboud University

Expert Selection

Expert: reduced test suite generator: $A \cdot I_{sub}^{\leq k+1} \cdot C$

- Remove uninformative inputs
- Remove future uninformative inputs
- Select inputs based on components
- (Use full alphabet)

Which expert should be used when?

- Static distribution

iCIS | Software Science
Radboud University

Expert Selection

Expert: reduced test suite generator: $A \cdot I_{sub}^{\leq k+1} \cdot C$

- Remove uninformative inputs
- Remove future uninformative inputs
- Select inputs based on components
- (Use full alphabet)

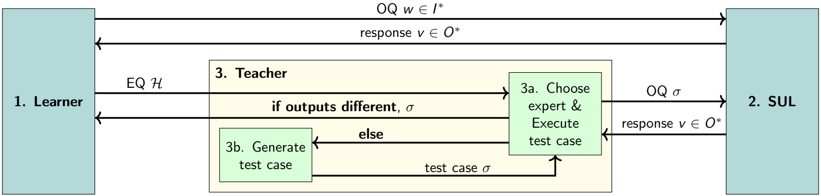
Which expert should be used when?

- Static distribution
- Multi-armed bandits to dynamically update distribution over experts
  - EXP3 algorithm
  - Exploration vs exploitation
  - More emphasis on recent test cases

 loes.kruger@ru.nl

iCIS | Software Science
Radboud University

MAT framework



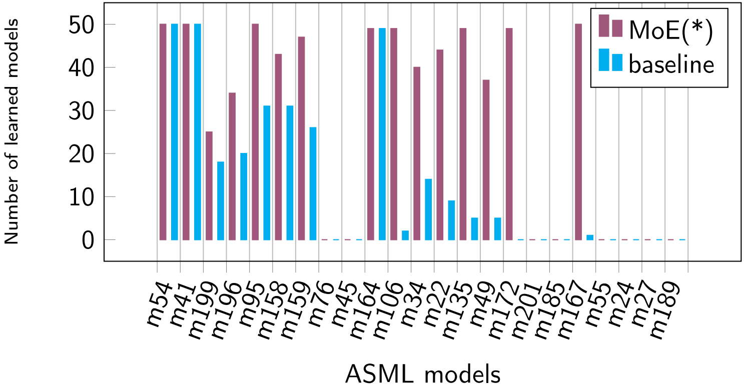
ASML models

- 23 models, 25-289 states, 10-177 inputs
- Maximal inputs budget of 10^8
- 50 seeds

 loes.kruger@ru.nl

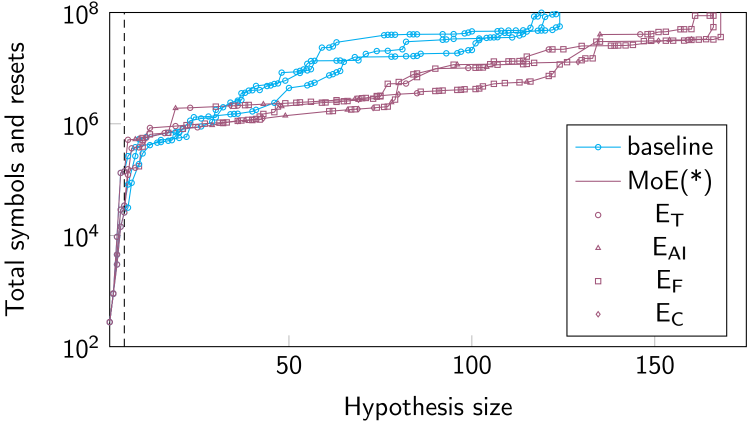
iCIS | Software Science
Radboud University

Results



Number of learned models
50
40
30
20
10
0
MoE(*)
baseline
m54
m41
m199
m196
m95
m158
m159
m76
m45
m164
m106
m34
m22
m135
m49
m172
m201
m185
m167
m55
m24
m27
m189
ASML models

Run Analysis



289 states and 138 inputs

Conclusion

- Testing is the bottleneck in active automata learning
- Test suites can be extremely large
- Based on $\mathcal{H}$, experts can make smaller test suites
- Future expert seems to work well, but more experts can be explored!

Related Model Learning Research

- Adaptive learning for evolving systems
- Test case prioritization based on apartness
- $L^{\#}$ for different types of automata
- Categorical framework for automata learning
- Fuzzing for network protocols (recently FTP)
- ...