Automata Learning for Dynamic Software Product Lines

Mohammad Mousavi

mohammad.mousavi@kcl.ac.uk

SPLC 2023, Tokyo, Japan

29 August 2023

Acknowledgments

In this tutorial I have re-used slides by colleagues and co-authors, in particular:
Diego Damasceno, **Sophie Fortz**, **Faeze Labbaf**, and **Shaghayegh Tavassoli**.

The material presented in the slides are due to many researchers,
explicitly acknowledged at the respective slides.

My research has been partially funded by:
UKRI TAS Node on Verifiability and **EPSRC VSL-Q** Grants.

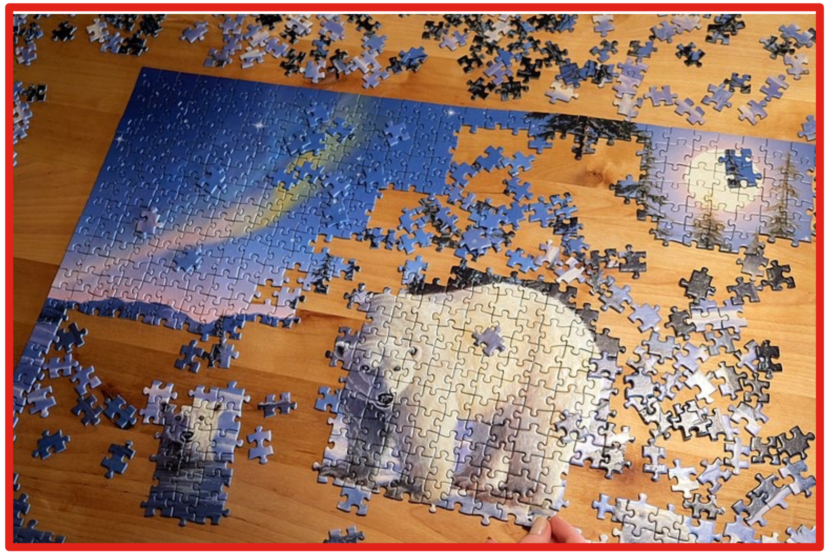
Engineering and Physical Sciences Research Council

UKRI Verifiability Node

Active Automata Learning

Adaptive Learning

Product-Line Learning

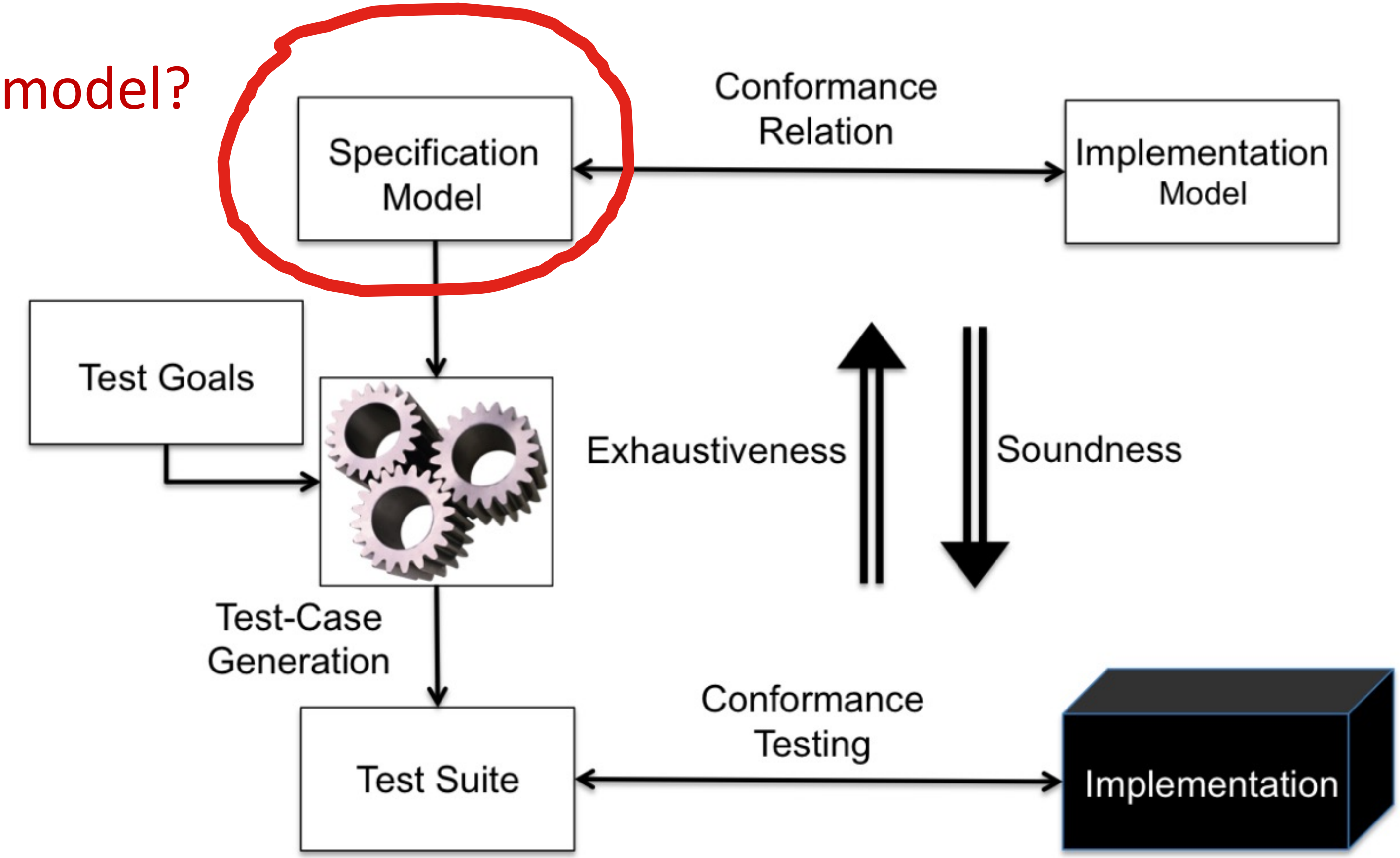


Compositional Learning

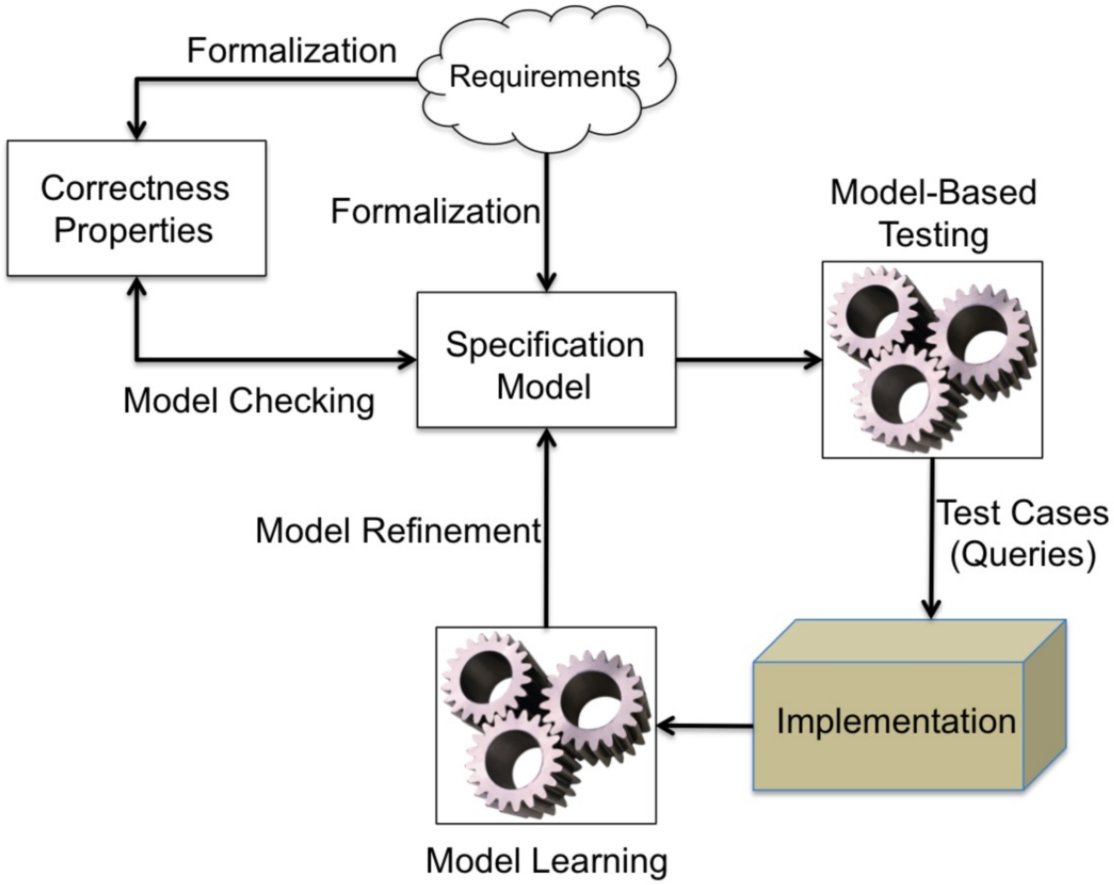
Active Automata Learning

Active Learning: Why?

Model? What model?

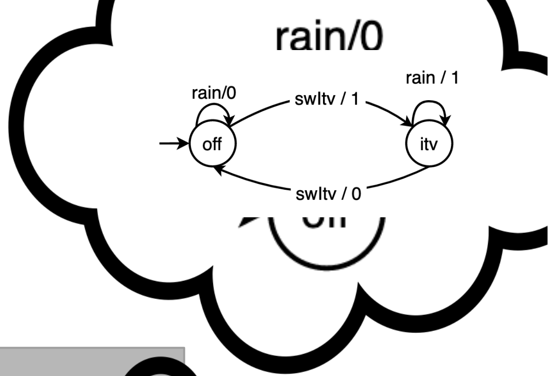


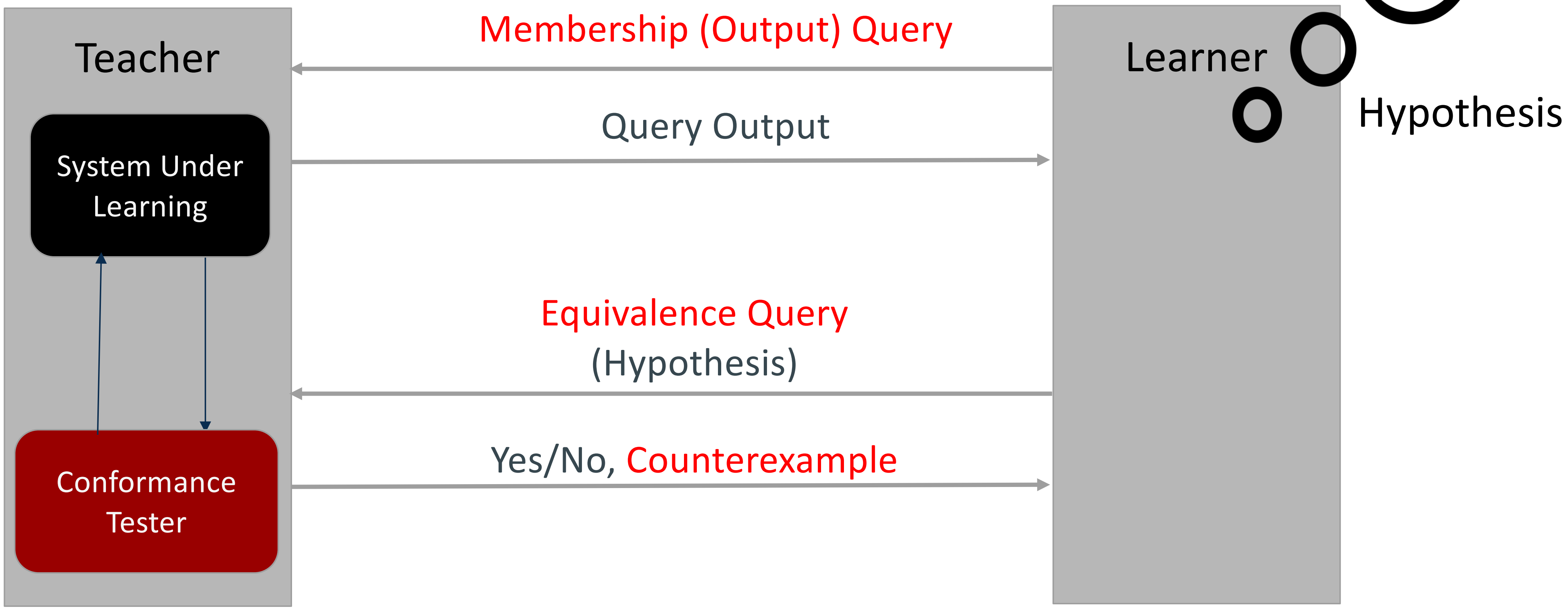
Active Learning: Why?



[Aichernig, et al. Model Learning and Model-Based Testing]
[Howar and Steffen. Active Automata Learning in Practice]
[Vaandrager. Model Learning]

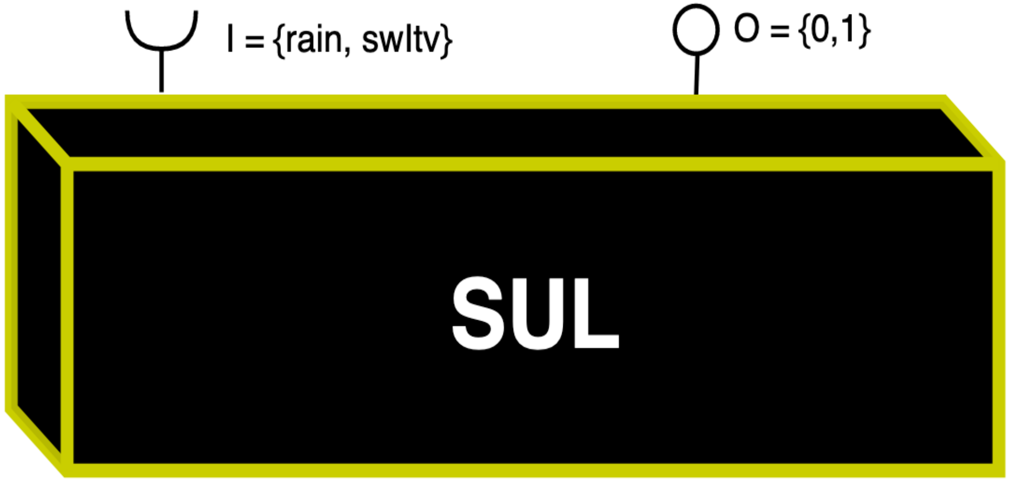
Active Learning: What?



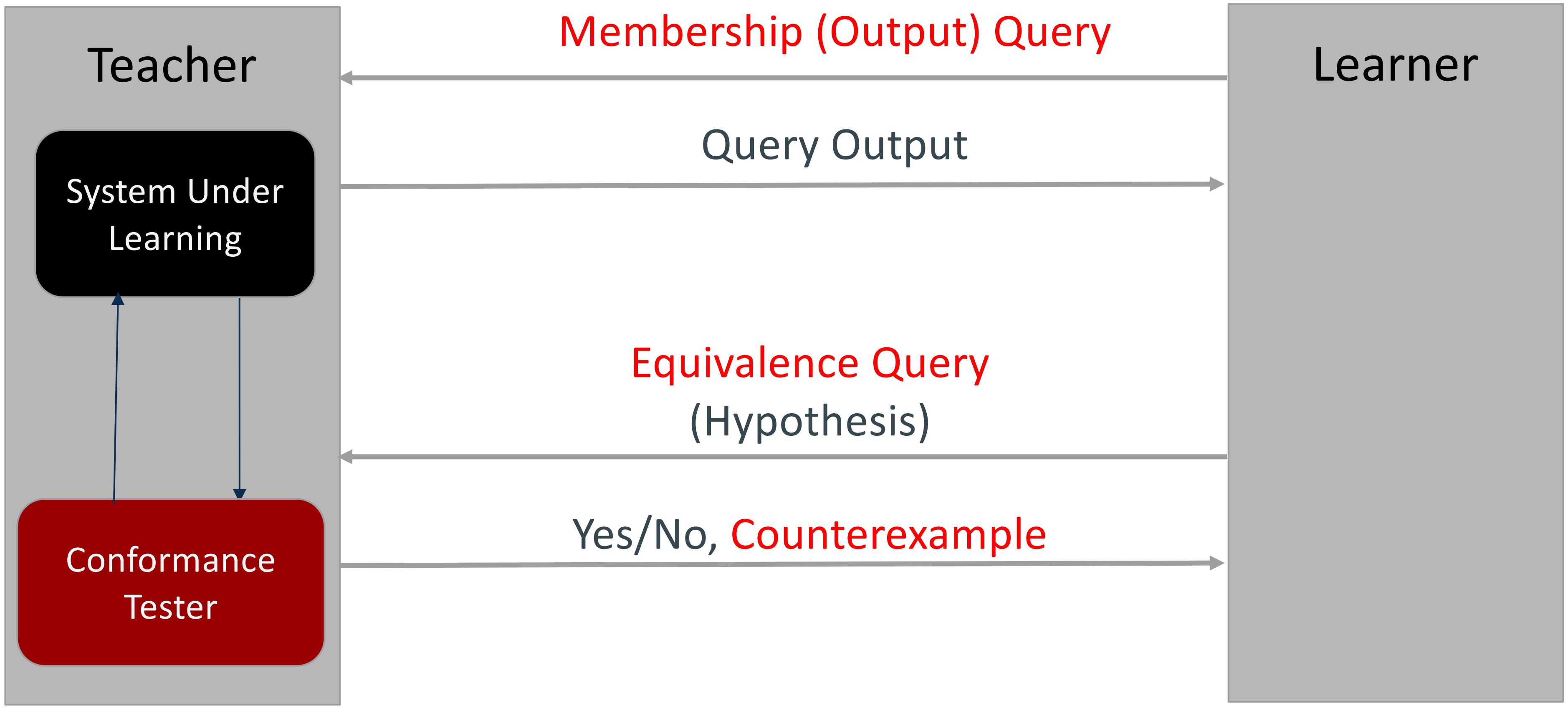


[Dana Angluin. Learning regular sets from queries and counterexamples. I&C. 1987]

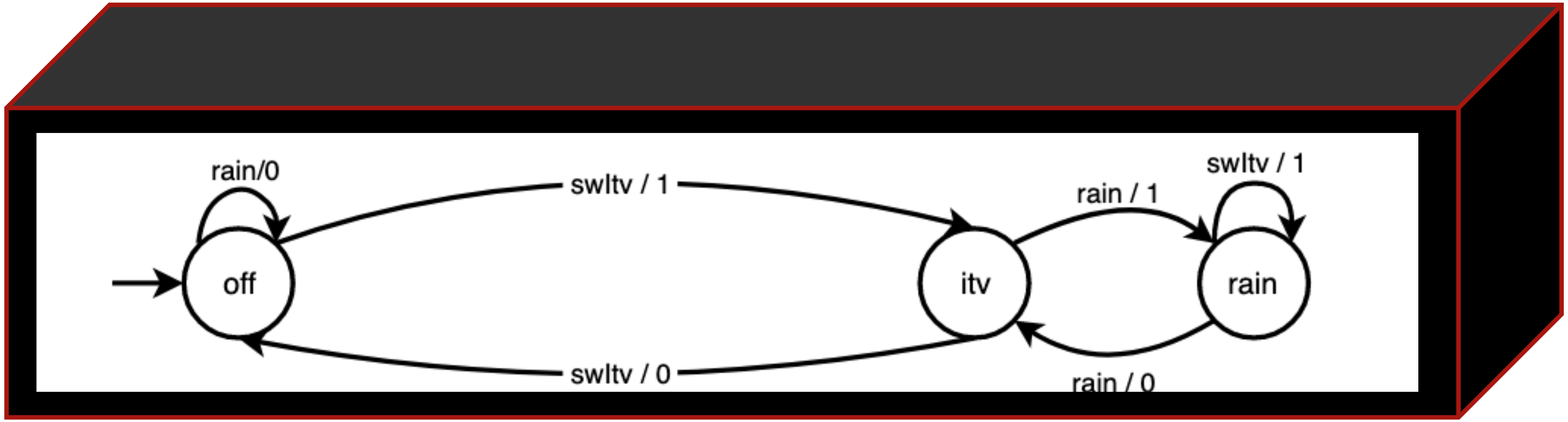
Active Learning: How?

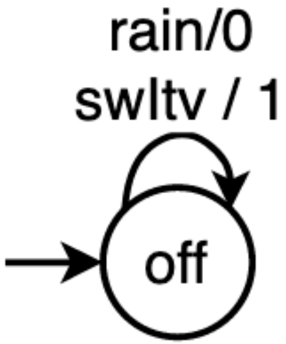


Active Learning: What?



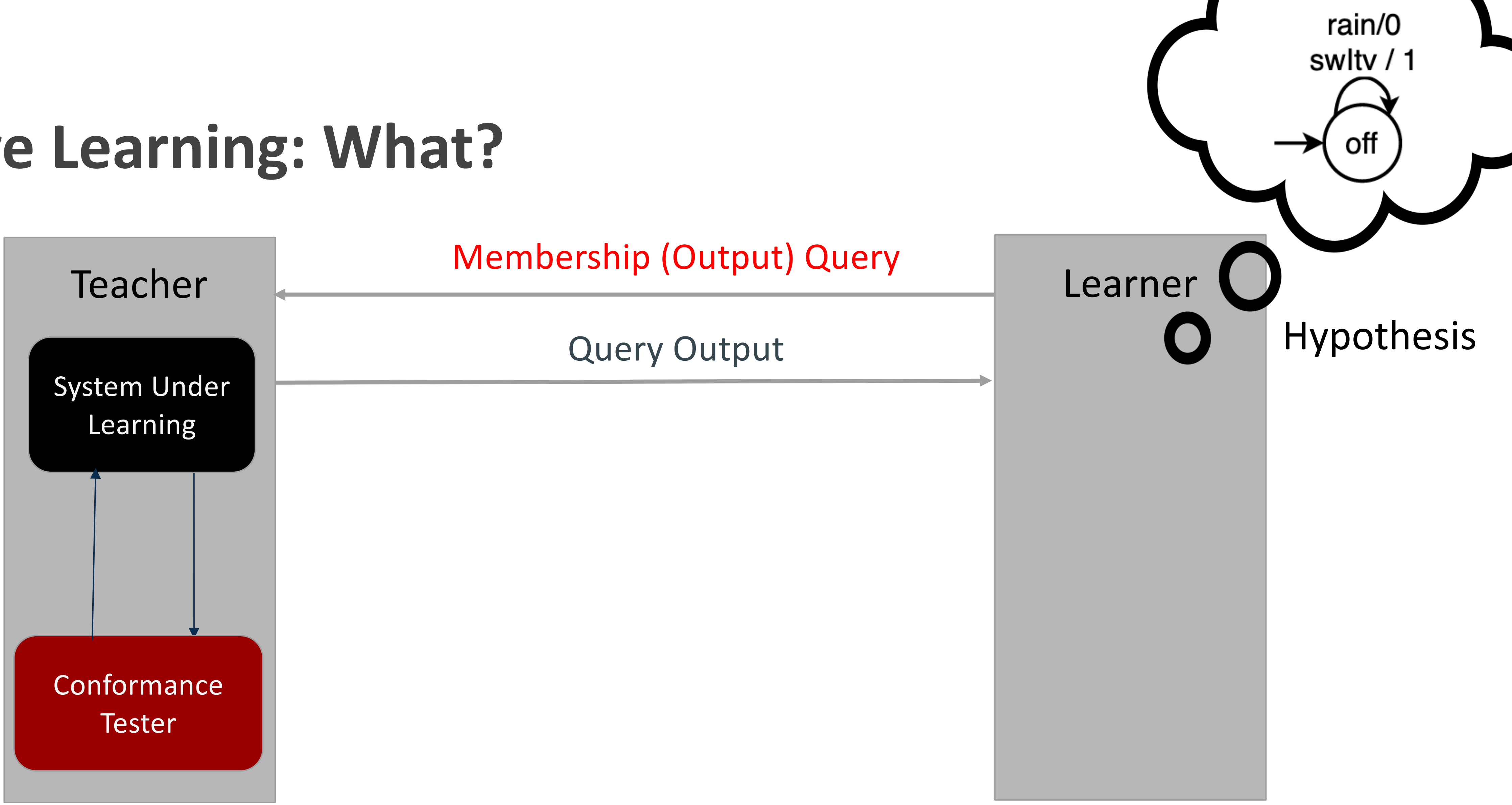
Active Learning: How?





		rain	swItv
S	ϵ	0	1

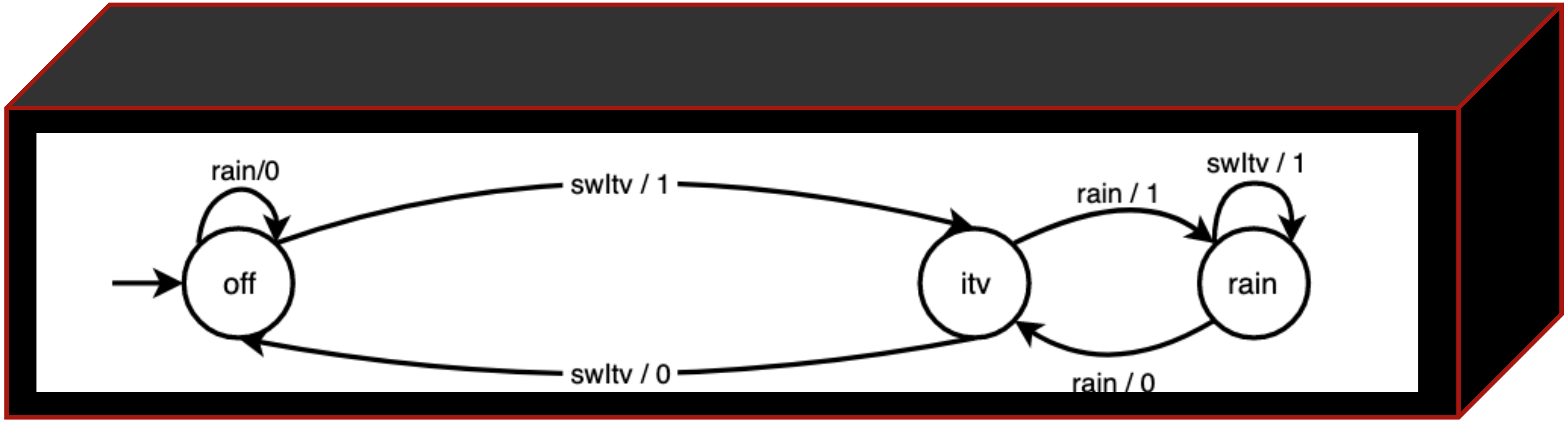
Active Learning: What?

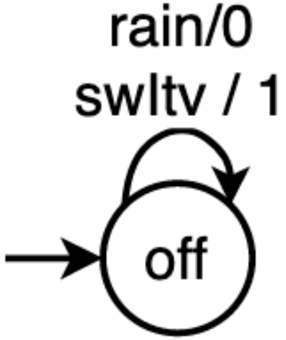


Good hypothesis?

Can I check that before asking the teacher an equivalence query?

Active Learning: How?

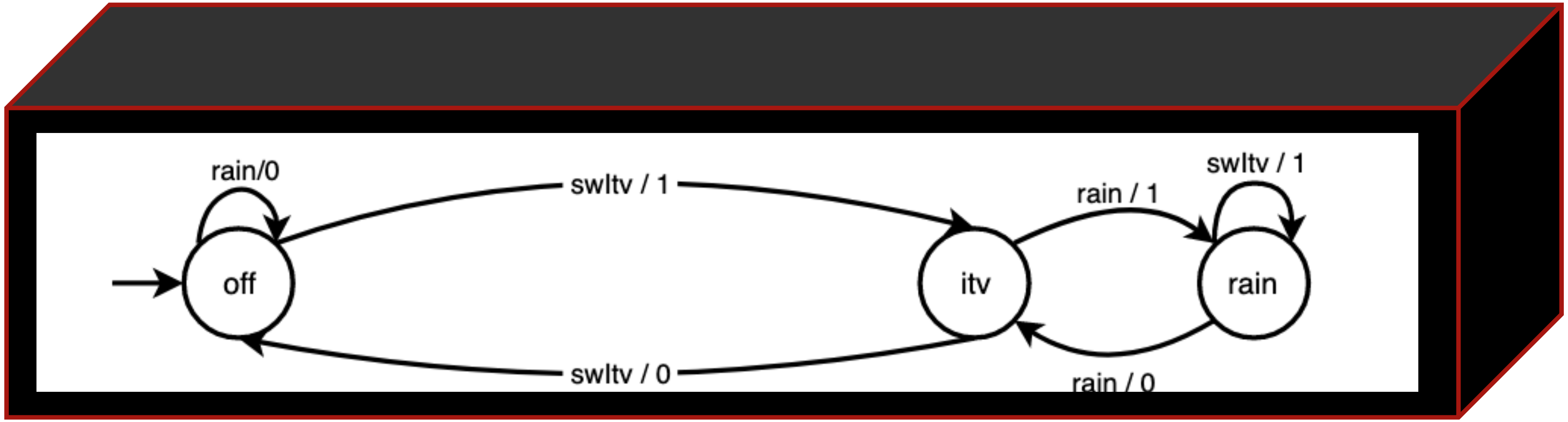


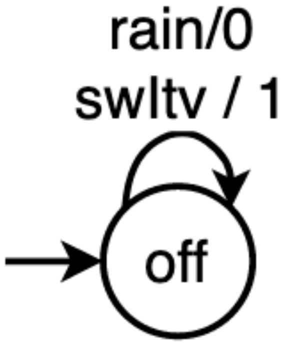


		rain	swItv
S	ϵ	0	1

Consistent: $\forall p, p' \in S_r \cdot p \cong p' \Rightarrow \forall i \in I\ p.i \cong p'.i$

Active Learning: How?

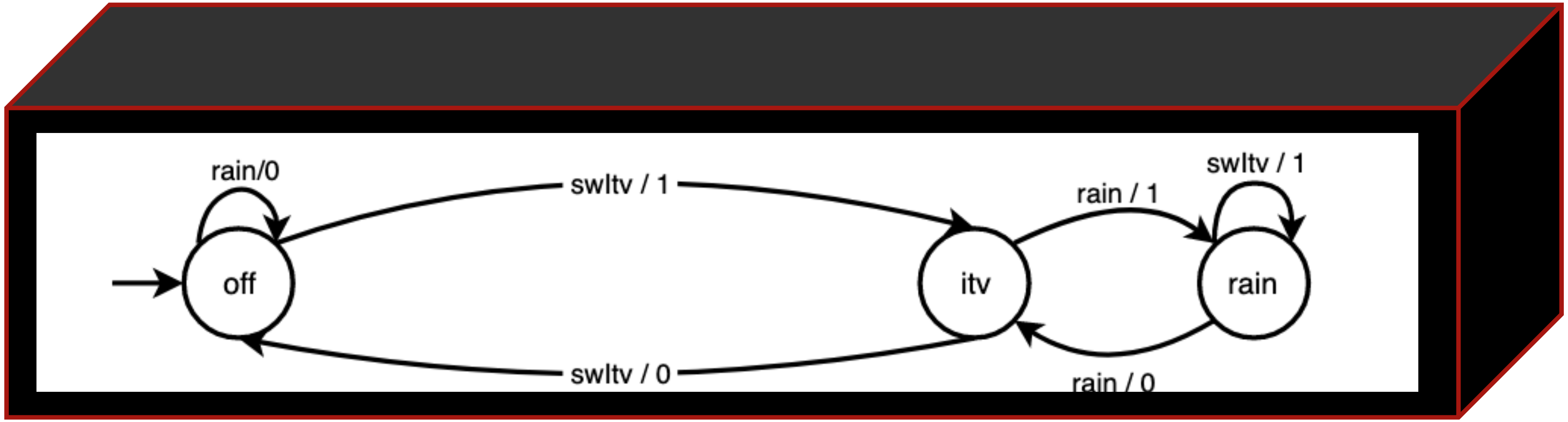


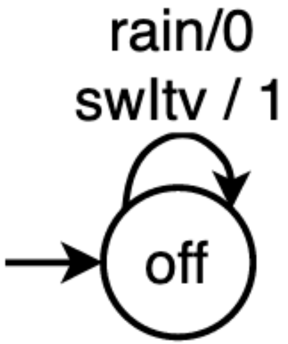


		rain	swItv
S	ϵ	0	1
$S \cdot I$	rain	0	1
	swItv	1	0

Consistent: $\forall p, p' \in S_r \cdot p \cong p' \Rightarrow \forall i \in I\ p.i \cong p'.i$

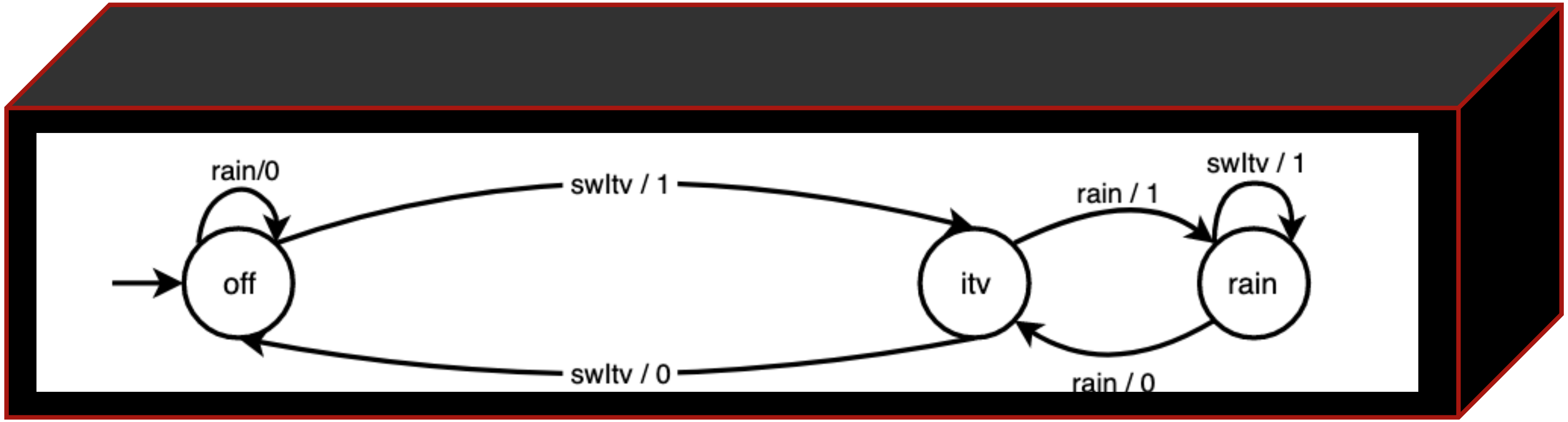
Active Learning: How?

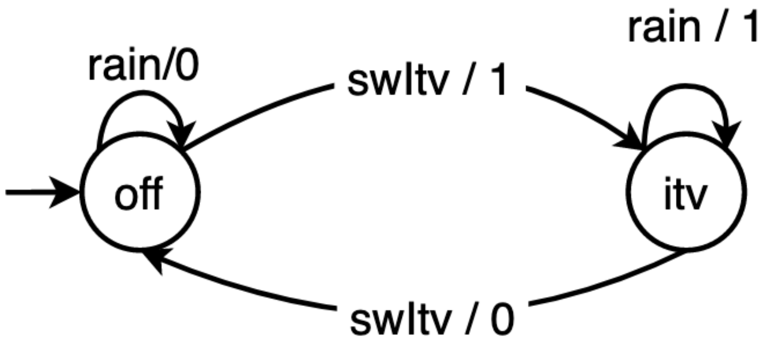




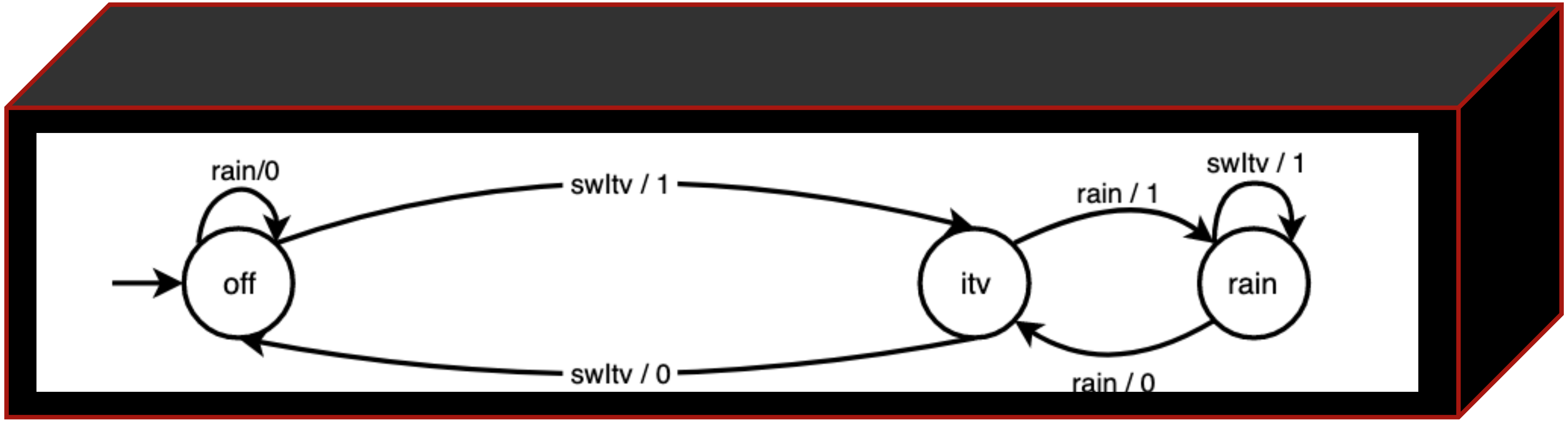
		rain	swItv
S	ϵ	0	1
$S \cdot I$	rain	0	1
	swItv	1	0

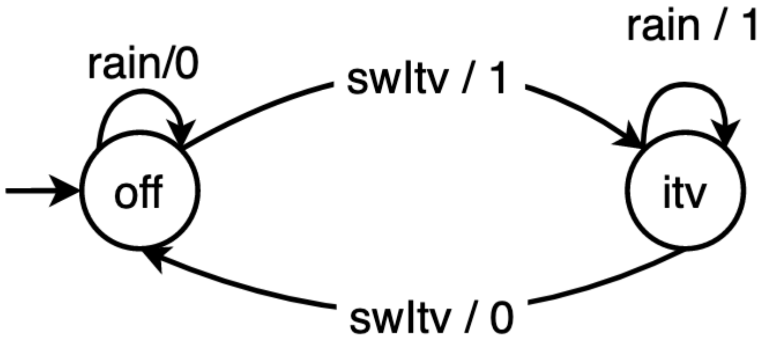
Active Learning: How?





		rain	swItv
S	ϵ $(\{\epsilon, \texttt{rain}\})$	0	1
	swItv	1	0



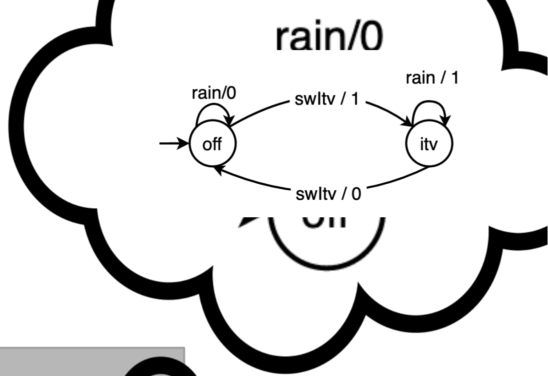


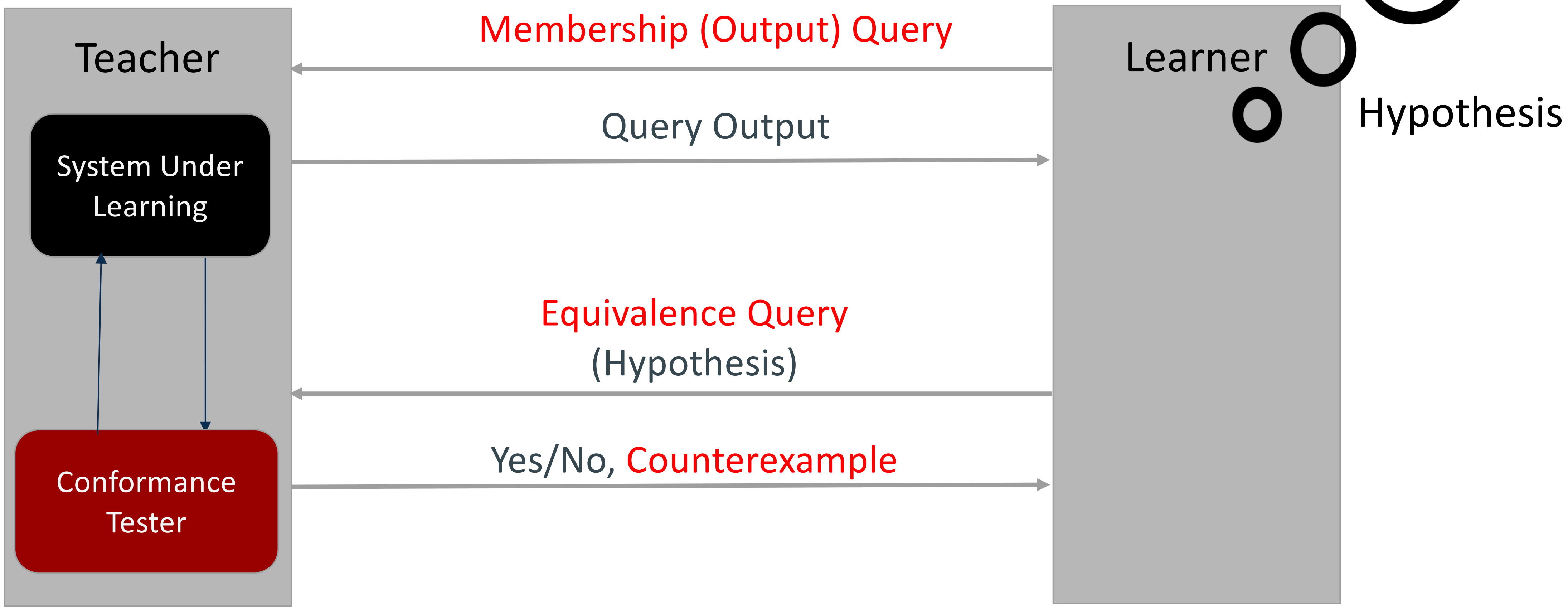
		rain	swItv
S	ϵ $(\{\epsilon, \texttt{rain}\})$	0	1
	swItv	1	0
$S \cdot I$	rain	0	1
	swItv · rain	0	1
	swItv · swItv	0	1

Consistent: $\forall p, p' \in S_r \cdot p \cong p' \Rightarrow \forall i \in I\ p.i \cong p'.i$

Closed (Complete): $\forall p \in S_r . I_r\ \exists p' \in S_r \cdot p \cong p'$

Active Learning: What?



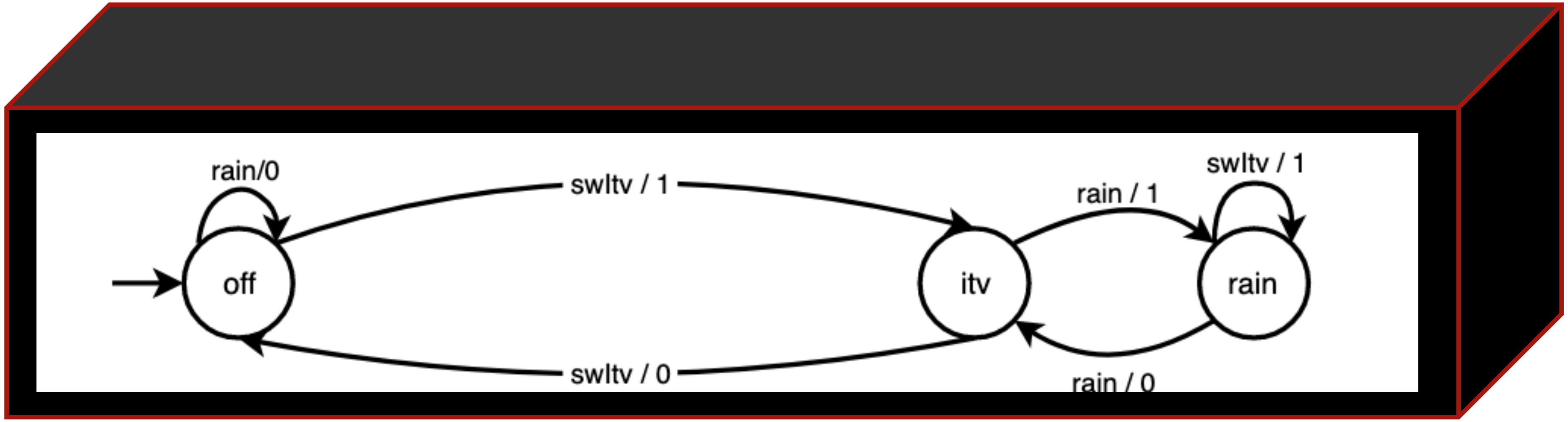


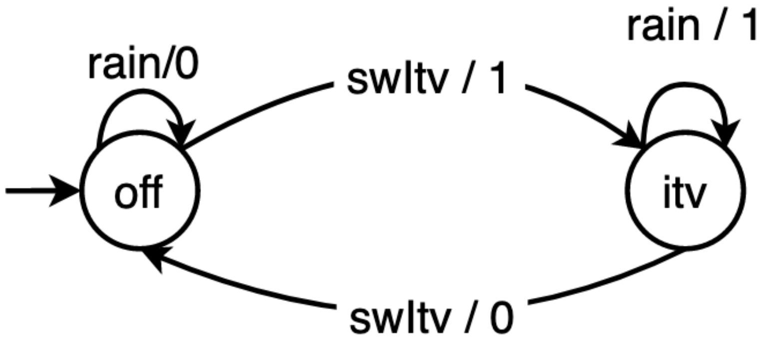
Equivalence Queries

- Random walk: surprisingly efficient, no guarantees

- Complete model-based tests (W Method, WP Method):
  - Two major phases:
    - establishing a tour of hypothesis states in the SUL
    - Testing all pairs of states and inputs,
      checking for the correct output and target state
  - Proven guarantee of detecting all differences wrt. a given fault model

[Broy, Jonsson, Katoen, Leucker, and Pretschner. *Model-Based Testing of Reactive Systems*]

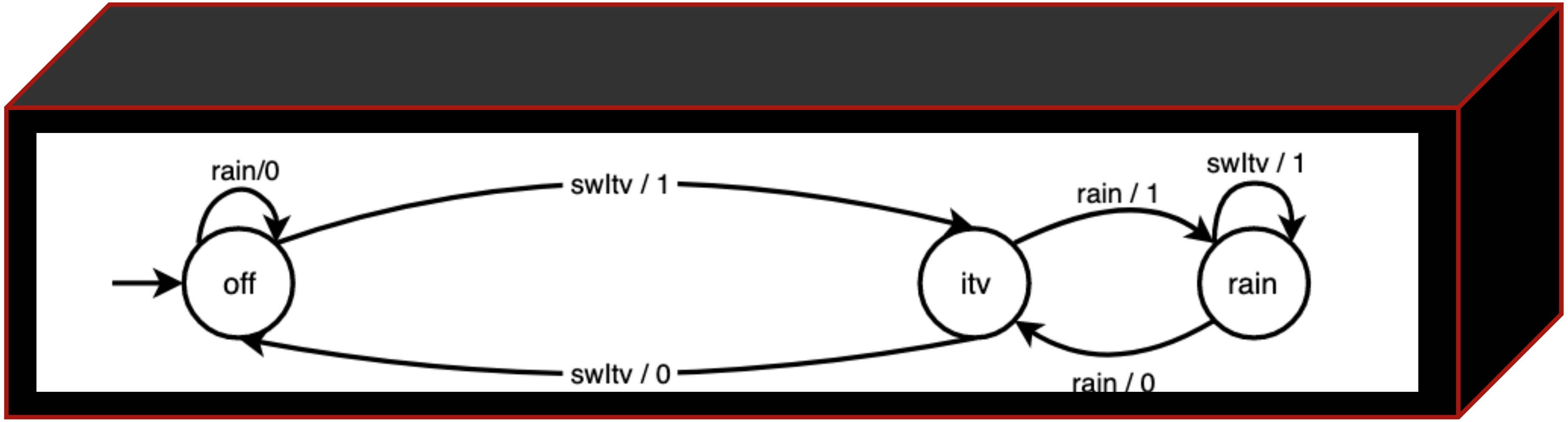


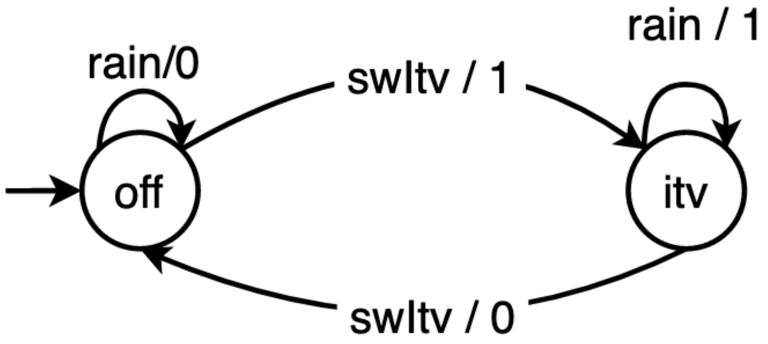


		rain	swItv
S	ϵ $(\{\epsilon, \texttt{rain}\})$	0	1
	swItv	1	0
$S \cdot I$	rain	0	1
	$\texttt{swItv} \cdot \texttt{rain}$	0	1
	$\texttt{swItv} \cdot \texttt{swItv}$	0	1

$\mathrm{CE} : \texttt{swItv} \cdot \texttt{rain} \cdot \texttt{rain} \cdot \texttt{rain}$

How should we process CE?





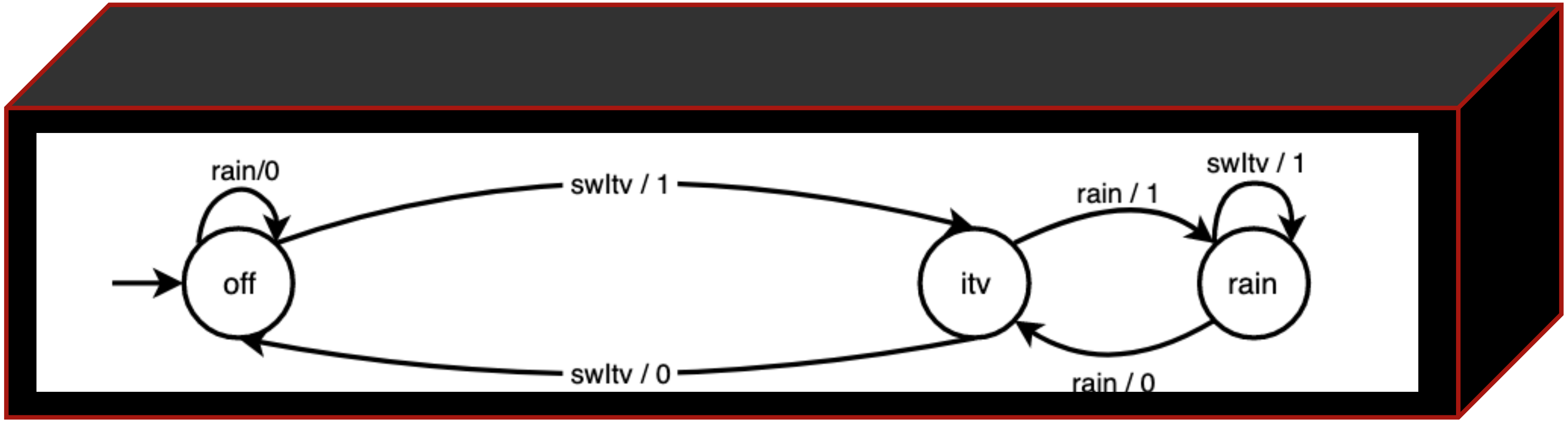
		$\mathtt{rain}$	$\mathtt{swItv}$
S	$\epsilon \quad (\{\epsilon, \mathtt{rain}\})$	0	1
	$\mathtt{swItv}$	1	0
$S \cdot I$	$\mathtt{rain}$	0	1
	$\mathtt{swItv} \cdot \mathtt{rain}$	0	1
	$\mathtt{swItv} \cdot \mathtt{swItv}$	0	1

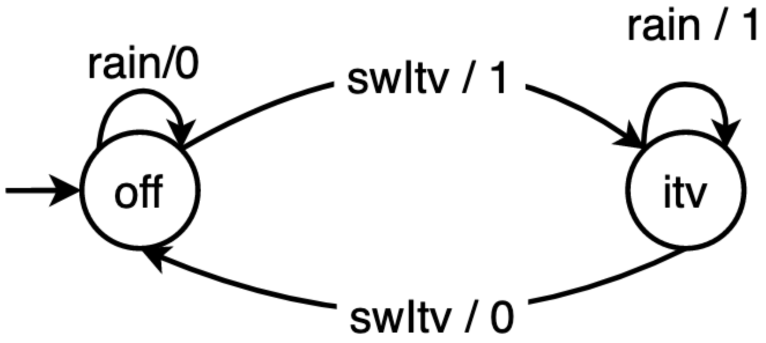
$$\mathrm{EQ} = \mathtt{swItv} \cdot \mathtt{rain} \cdot \mathit{rain} \cdot \mathit{rain}$$

$$1 \cdot 1 \cdot 1 \cdot 1 \neq 1 \cdot 1 \cdot 0 \cdot 1$$

[Rivest and Schapire. Inference of Finite Automata Using Homing Sequences. I&C 1993]

Counter-Example Processing





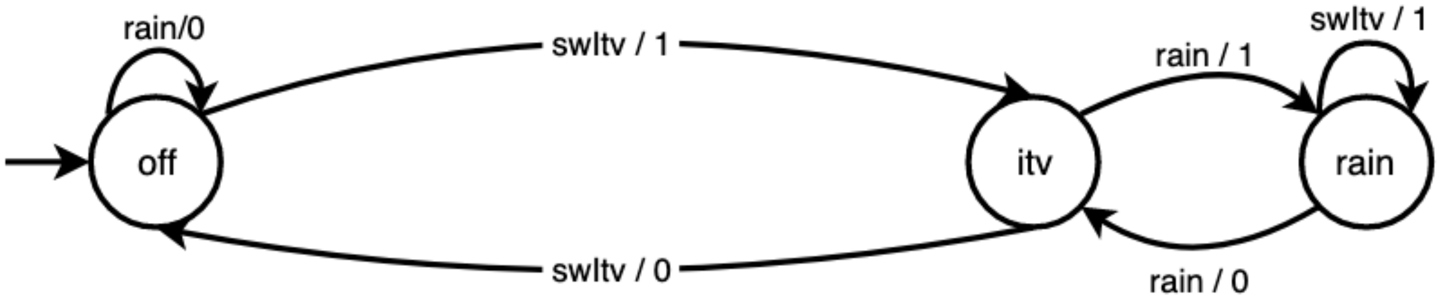
$$\mathtt{EQ} = \mathtt{swItv} \cdot \mathtt{rain} \cdot \boxed{\mathit{rain} \cdot \mathit{rain}}$$

$$1 \cdot 1 \cdot \boxed{1 \cdot 1} \neq 1 \cdot 1 \cdot \boxed{0 \cdot 1}$$

[Rivest and Schapire. Inference of finite automata using homing sequence. I&C. 1993]

[Irfan, Oriat, Groz. Model Inference and Testing. Adv. Comp. 2013]

Counter-Example Processing

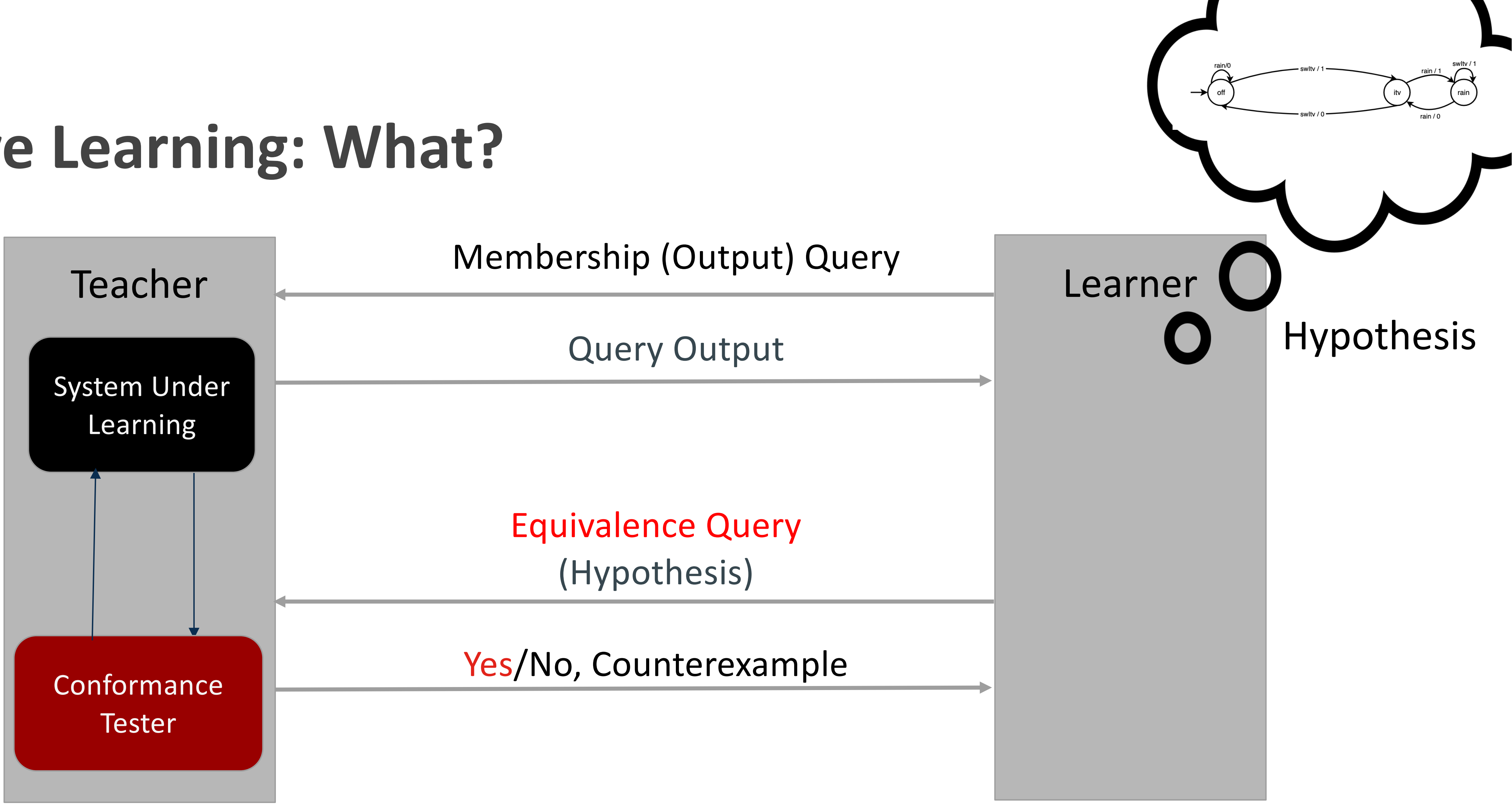


		$rain$	$swItv$	$rain \cdot rain$
S_r	ϵ	0	1	$0 \cdot 0$
	$swItv$	1	0	$1 \cdot 0$
	$swItv \cdot rain$	0	1	$0 \cdot 1$
$S_r \cdot I_r$	$rain$	0	1	$0 \cdot 0$
	$swItv \cdot swItv$	0	1	$0 \cdot 0$
	$swItv \cdot rain \cdot rain$	1	0	$1 \cdot 0$
	$swItv \cdot rain \cdot swItv$	0	1	$0 \cdot 1$

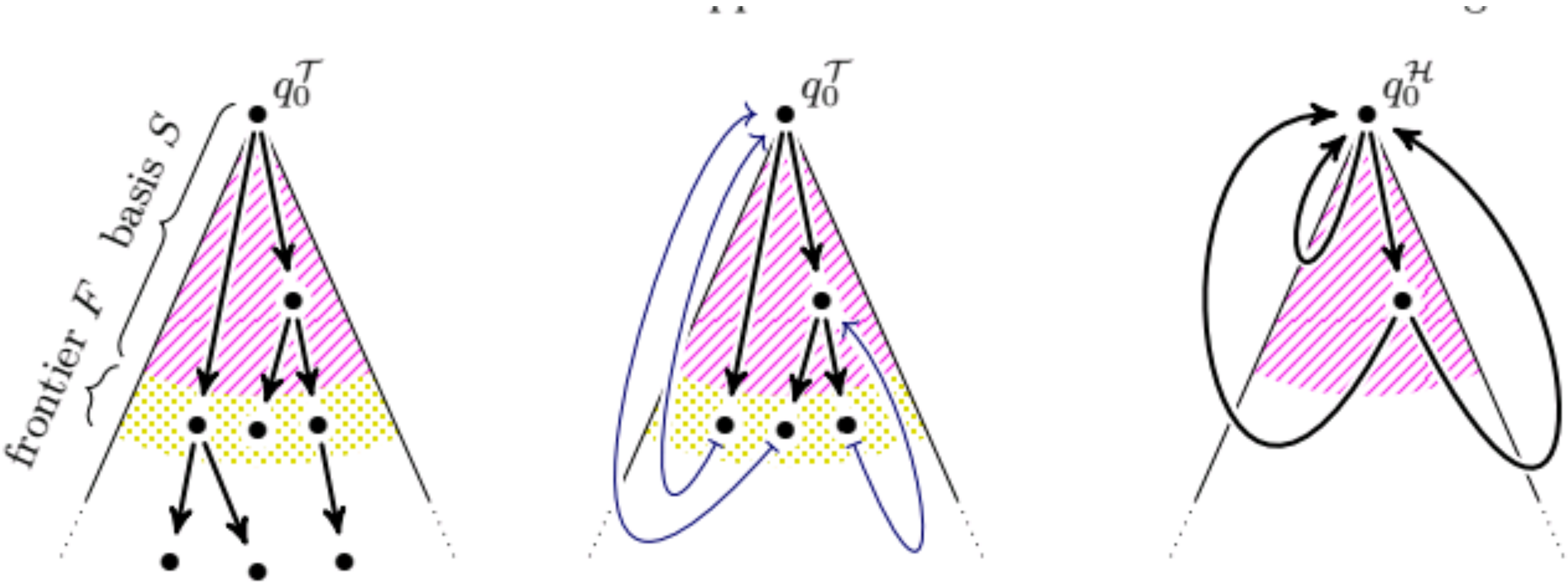
[Rivest and Schapire. Inference of finite automata using homing sequence. I&C. 1993]

[Irfan Oriat, Groz. Angluin-style finite-state machine inference with non-optimal counter-examples.]

Active Learning: What?

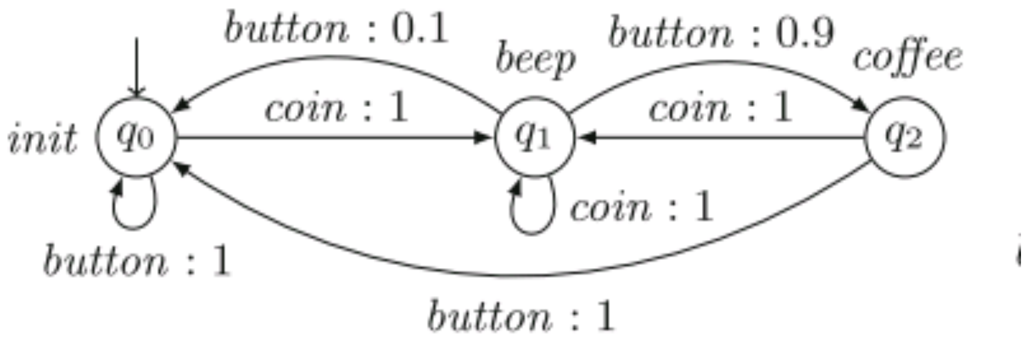


Beyond L*



[M. Isberner. Foundations of Active Automata Learning An Algorithmic Perspective. Ph.D. Thesis. 2015]
[Vaandrager et al. A New Approach for Active Automata Learning Based on Apartneess.]

Beyond Finite Automata



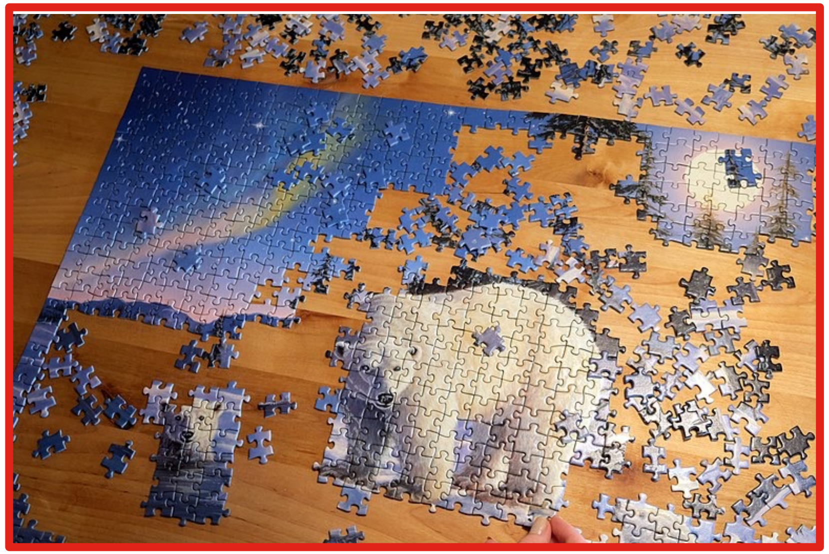
		button	*coin*
S	ϵ	$\{init\colon\ 247\}$	$\{beep\colon\ 414\}$
	$coin \cdot beep$	$\{coffee\colon\ 147, init\colon\ 16\}$	$\{beep\colon\ 134\}$
$Lt(S)$	$button \cdot init$	$\{init\colon\ 69\}$	$\{beep\colon\ 82\}$
	$coin \cdot beep \cdot button \cdot coffee$	$\{init\colon\ 64\}$	$\{beep\colon\ 53\}$
	$coin \cdot beep \cdot button \cdot init$	$\{init\colon\ 9\}$	$\{beep\colon\ 6\}$
	$coin \cdot beep \cdot coin \cdot beep$	$\{coffee\colon\ 65, init\colon\ 7\}$	$\{beep\colon\ 61\}$

[Tappler, Muškardin, Aichernig, Pill. Active Model Learning for Stochastic Automata.]
[Bacci, Ingolfsdottir, Larsen, Reynouard. Active Learning of Markov Decision Processes using Baum Welch Algorithm.]

Active Automata Learning

Adaptive Learning

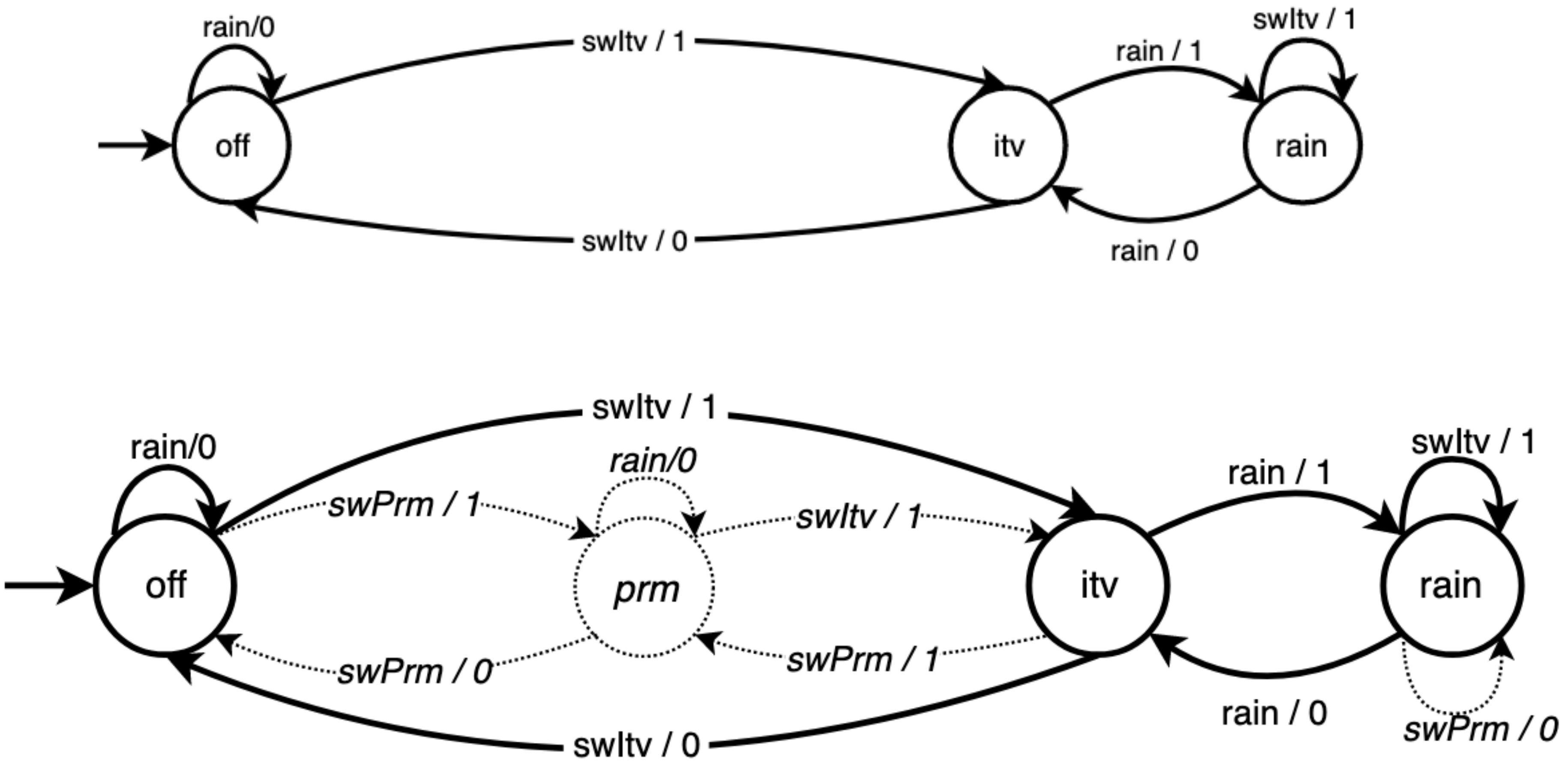
Learning Evolution in Space



Compositional Learning

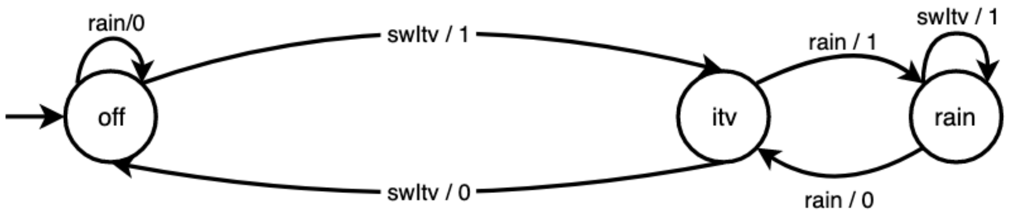
Adaptive Learning

Why?

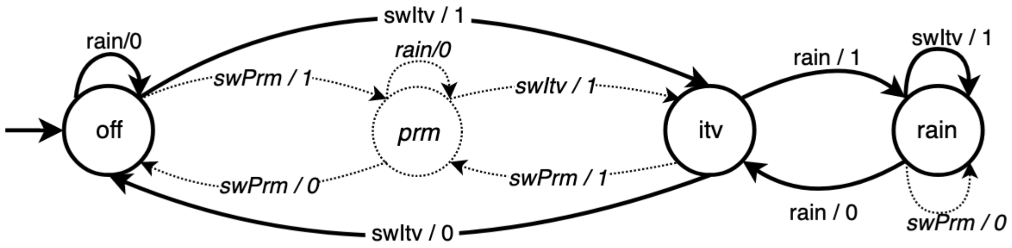

rain/0
swItv / 1
rain / 1
swItv / 1
off
itv
rain
swItv / 0
rain / 0
rain/0
swItv / 1
rain/0
swPrm / 1
swItv / 1
rain / 1
swItv / 1
off
prm
itv
rain
swPrm / 0
swPrm / 1
rain / 0
swPrm / 0
swItv / 0

What?

Given an evolving system that changed over time how can we efficiently learn its evolved behavior?

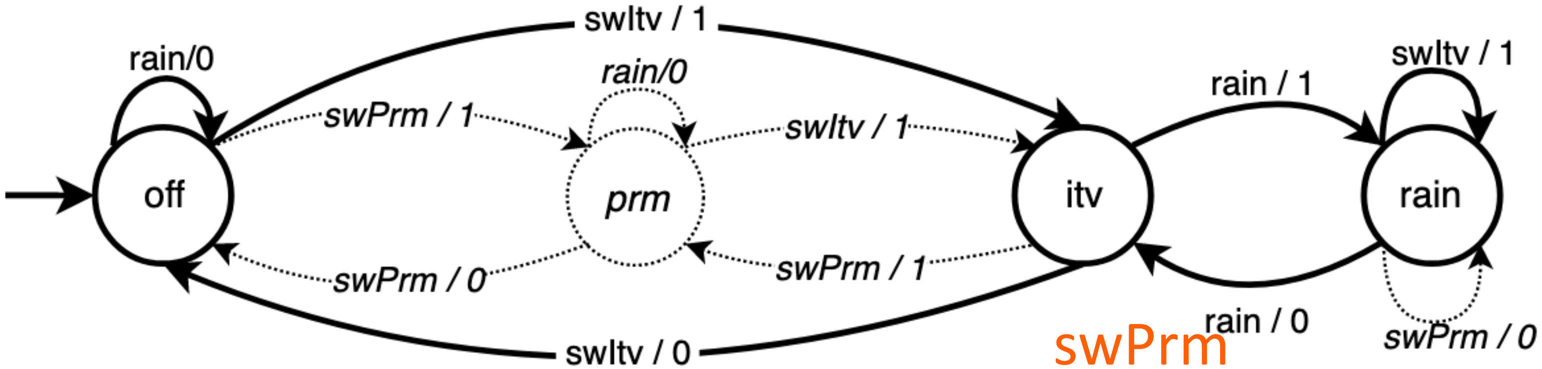


How sensitive is it to the amount of evolution?



[Groce, Peled, and Yannakakis. Adaptive model checking. 2002]
[Chaki, Clarke, Sharygina, Sinha.
Verification of evolving software via component substitutability analysis. 2008]

How?

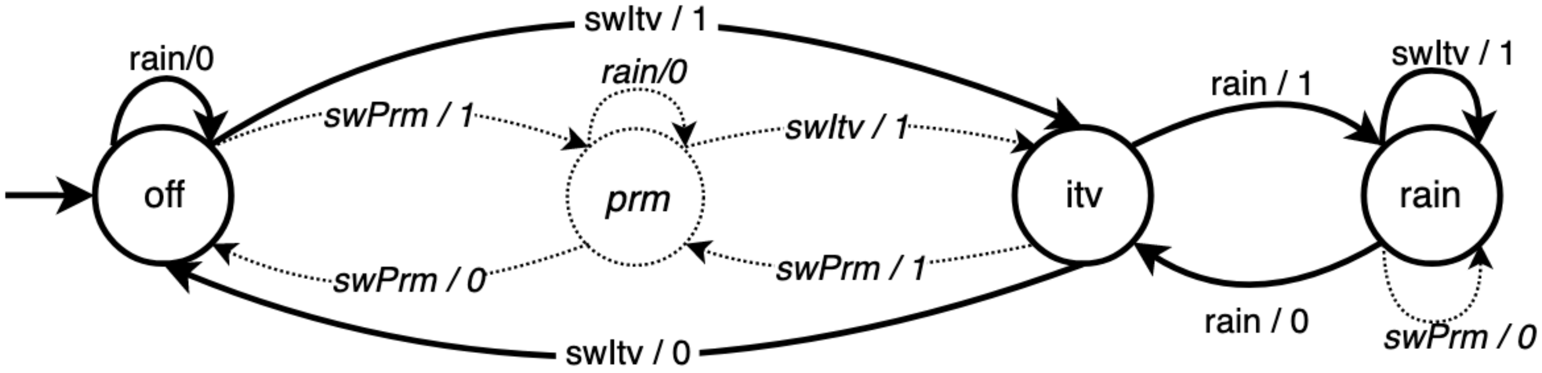


swPrm

Anything redundant?

		$rain$	$swItv$	$rain \cdot rain$
S_r	ϵ	0	1	$0 \cdot 0$
	$swItv$	1	0	$1 \cdot 0$
	$swItv \cdot rain$	0	1	$0 \cdot 1$
$S_r \cdot I_r$	$rain$	0	1	$0 \cdot 0$
	$swItv \cdot swItv$	0	1	$0 \cdot 0$
	$swItv \cdot rain \cdot rain$	1	0	$1 \cdot 0$
	$swItv \cdot rain \cdot swItv$	0	1	$0 \cdot 1$

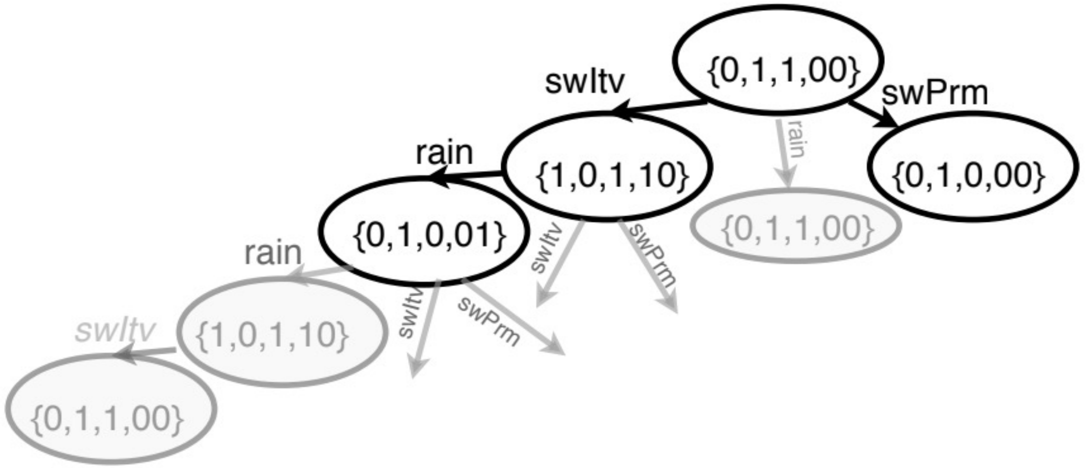
How?



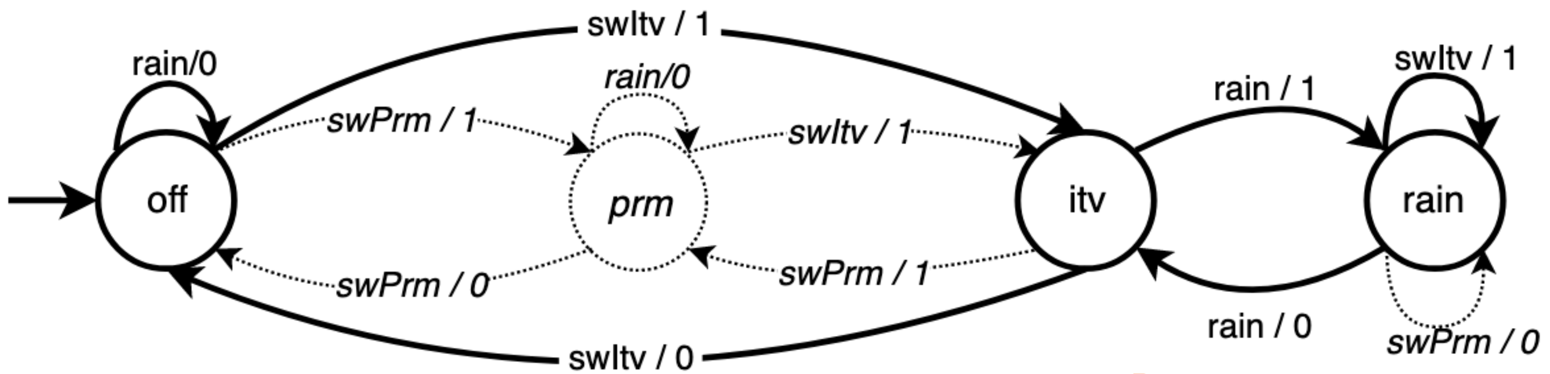
Anything redundant?

swPrm

		$rain$	$swItv$	$rain \cdot rain$
S_r	ϵ	0	1	$0 \cdot 0$
	$swItv$	1	0	$1 \cdot 0$
	$swItv \cdot rain$	0	1	$0 \cdot 1$
$S_r \cdot I_r$	$rain$	0	1	$0 \cdot 0$
	$swItv \cdot swItv$	0	1	$0 \cdot 0$
	$swItv \cdot rain \cdot rain$	1	0	$1 \cdot 0$
	$swItv \cdot rain \cdot swItv$	0	1	$0 \cdot 1$



How?

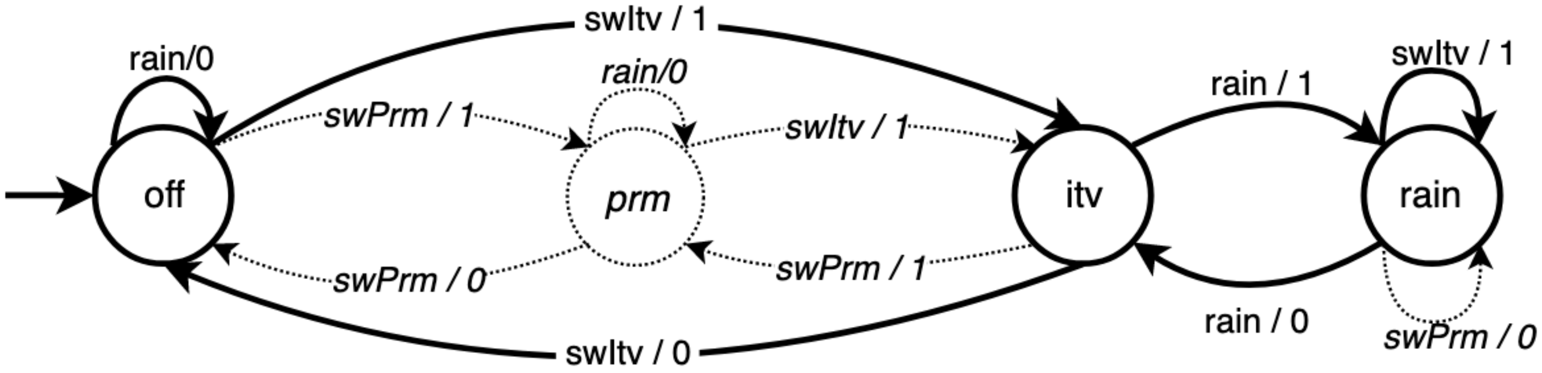


swPrm

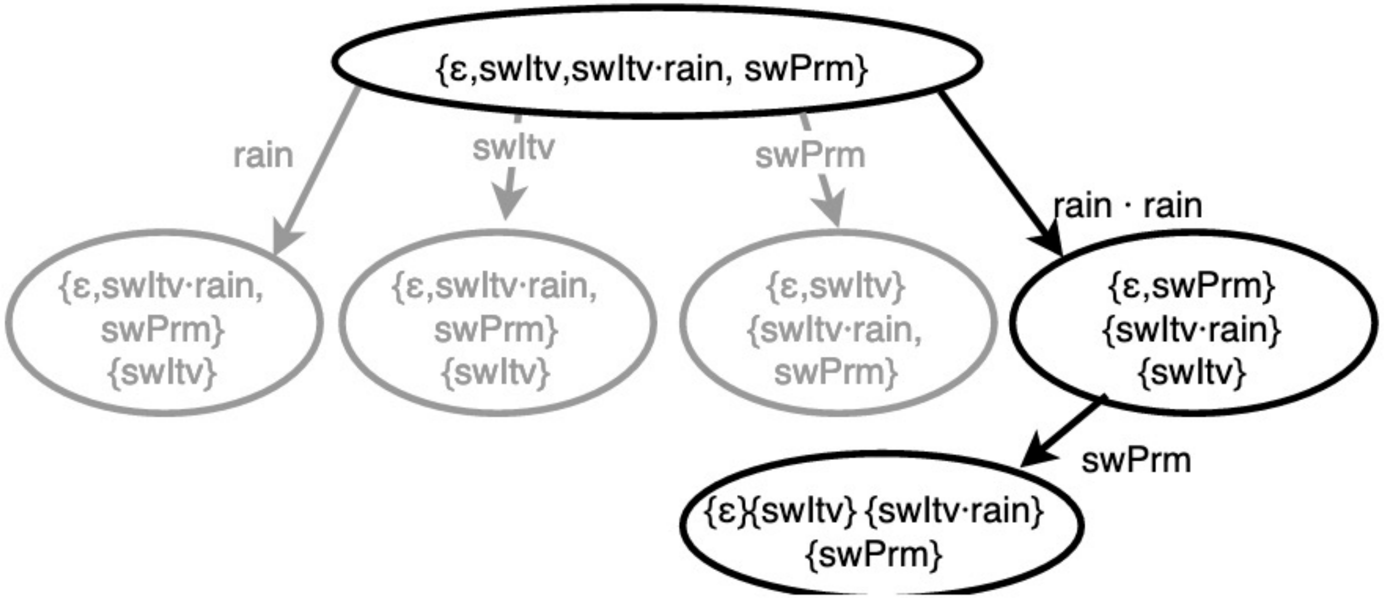
New experiments?

		$rain$	$swItv$	$rain \cdot rain$
S_r	ϵ	0	1	$0 \cdot 0$
	$swItv$	1	0	$1 \cdot 0$
	$swItv \cdot rain$	0	1	$0 \cdot 1$
$S_r \cdot I_r$	$rain$	0	1	$0 \cdot 0$
	$swItv \cdot swItv$	0	1	$0 \cdot 0$
	$swItv \cdot rain \cdot rain$	1	0	$1 \cdot 0$
	$swItv \cdot rain \cdot swItv$	0	1	$0 \cdot 1$

How?



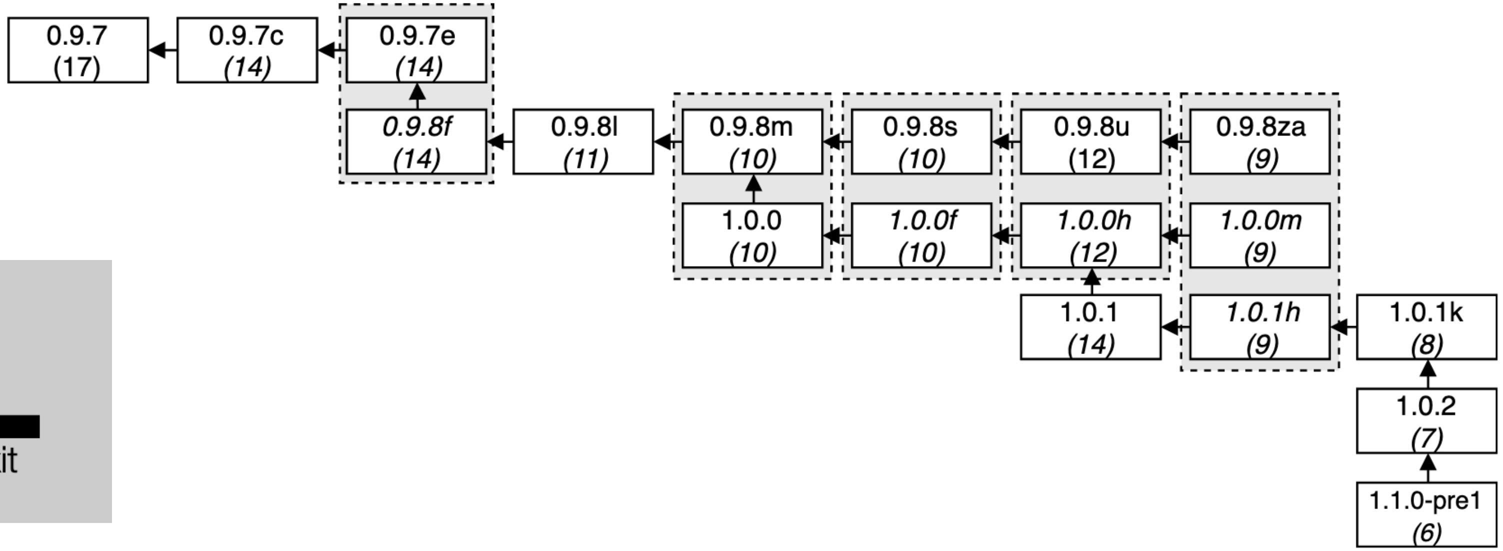
swItv / 1
rain/0
rain/0
swPrm / 1
swItv / 1
rain / 1
swItv / 1
off
prm
itv
rain
swPrm / 0
swPrm / 1
rain / 0
swPrm / 0
swItv / 0



{ε,swItv,swItv·rain, swPrm}
rain
swItv
swPrm
rain · rain
{ε,swItv·rain, swPrm} {swItv}
{ε,swItv·rain, swPrm} {swItv}
{ε,swItv} {swItv·rain, swPrm}
{ε,swPrm} {swItv·rain} {swItv}
swPrm
{ε}{swItv} {swItv·rain} {swPrm}

Does it Really work?

OpenSSL
Cryptography and SSL/TLS Toolkit



[https://www.openssl.org/]
[De Ruiter. A tale of the openssl state machine. 2016]

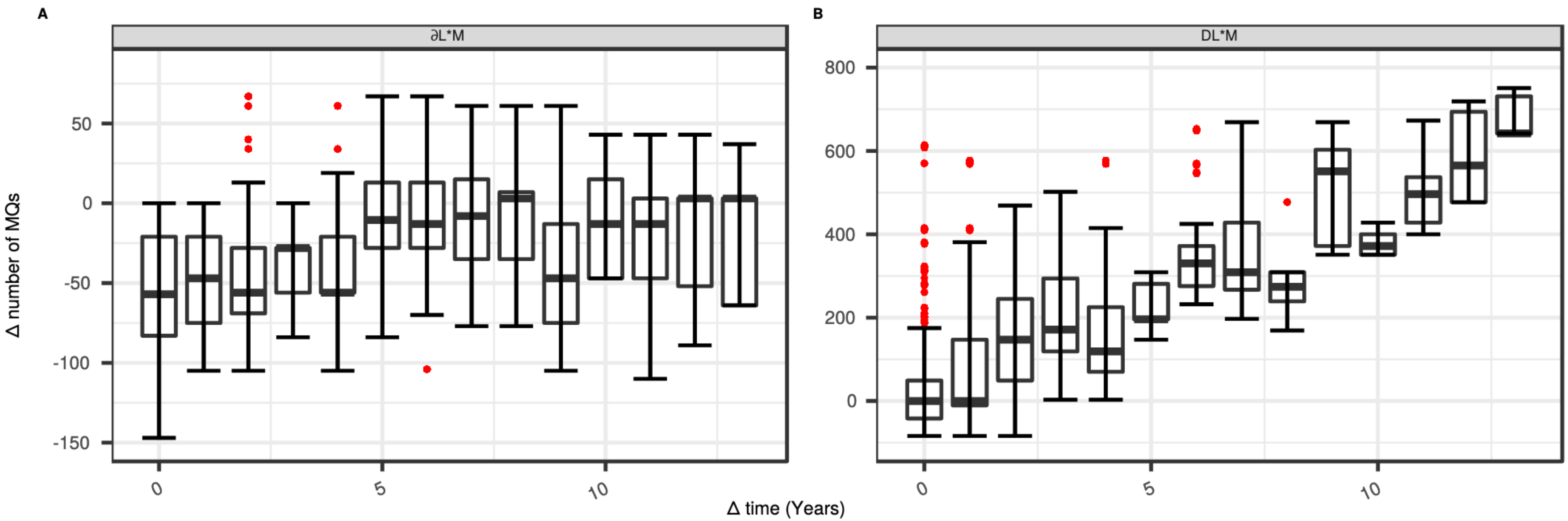
DOES IT REALLY WORK?

Given an evolving system that changed over time how can we efficiently learn its evolved behavior?

How sensitive is it to the amount of evolution?

OpenSSL
Cryptography and SSL/TLS Toolkit

Does it Really work?

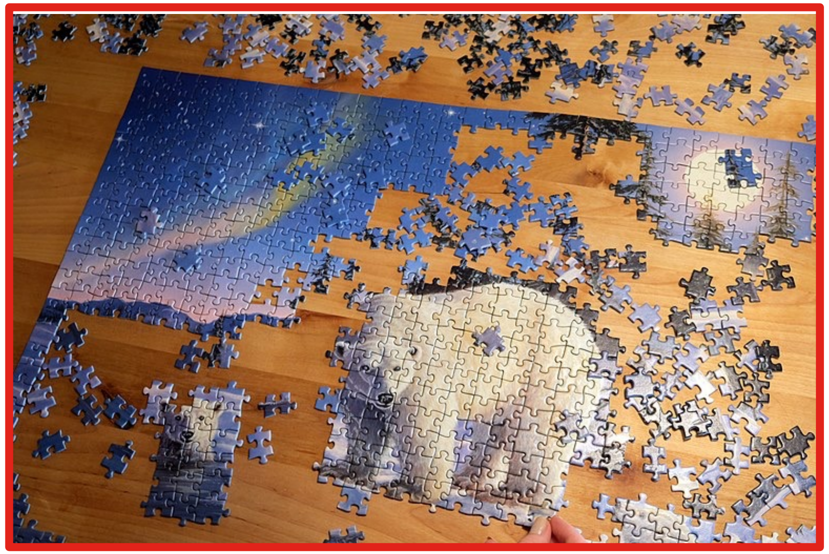


[Damasceno, M.R. Mousavi and A. Simao.
Learning to Reuse: Adaptive Model Learning for Evolving Systems. iFM'19]

Active Automata Learning

Adaptive Learning

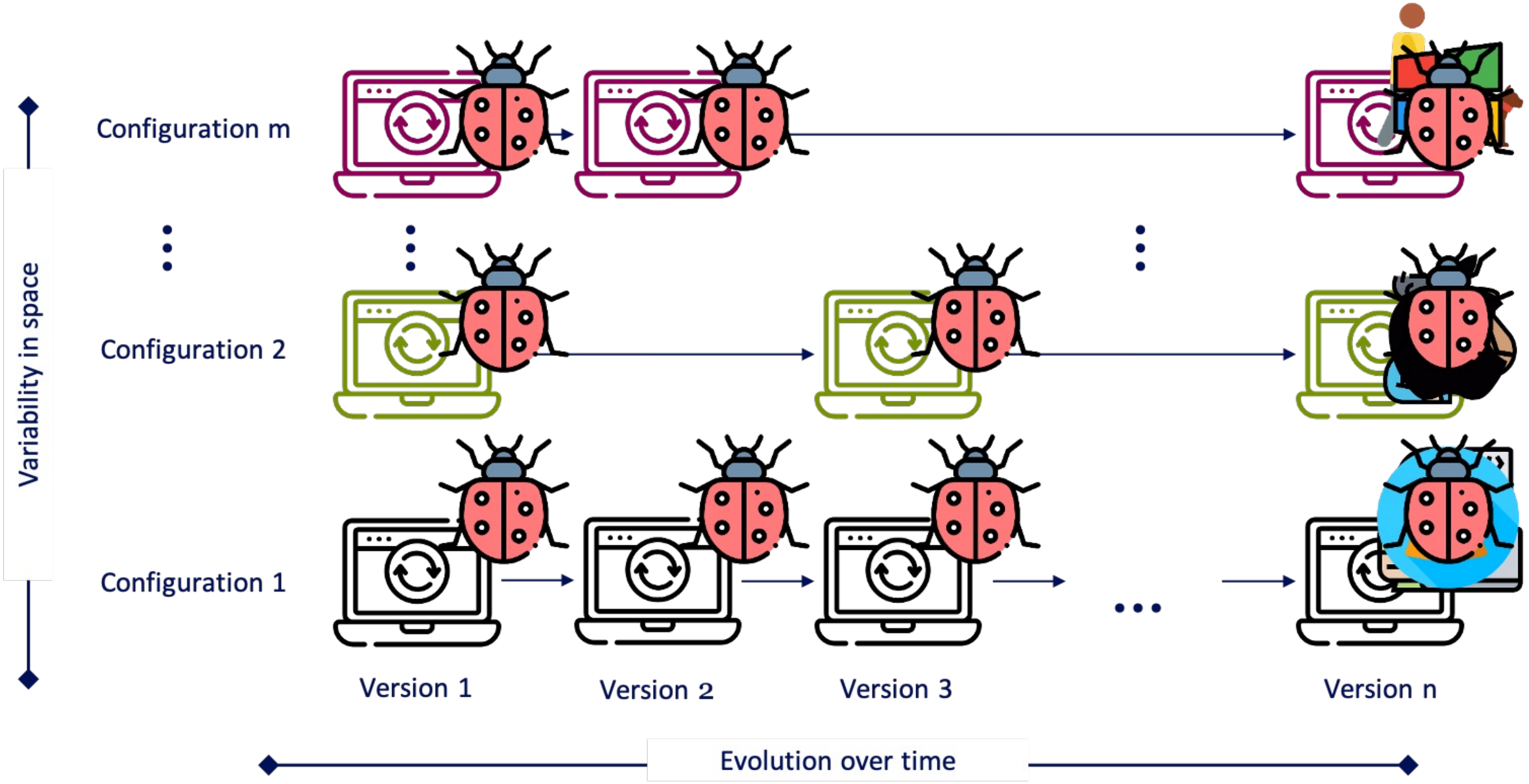
Product-Line Learning



Compositional Learning

Product-Line Learning

Why?

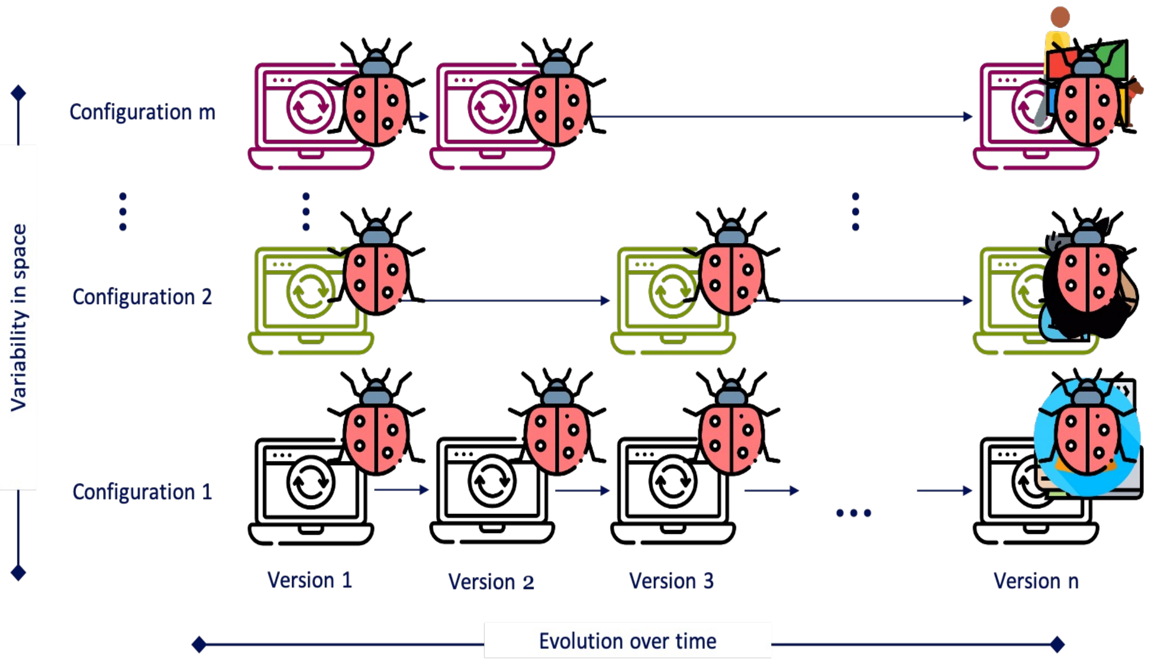


Configuration m
Configuration 2
Configuration 1
Variability in space
Version 1
Version 2
Version 3
Version n
Evolution over time

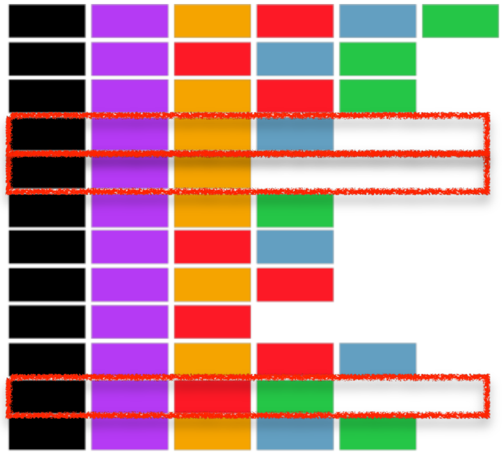
Why?

Given an evolving system that changed in space
how can we succintly summarise the variability?

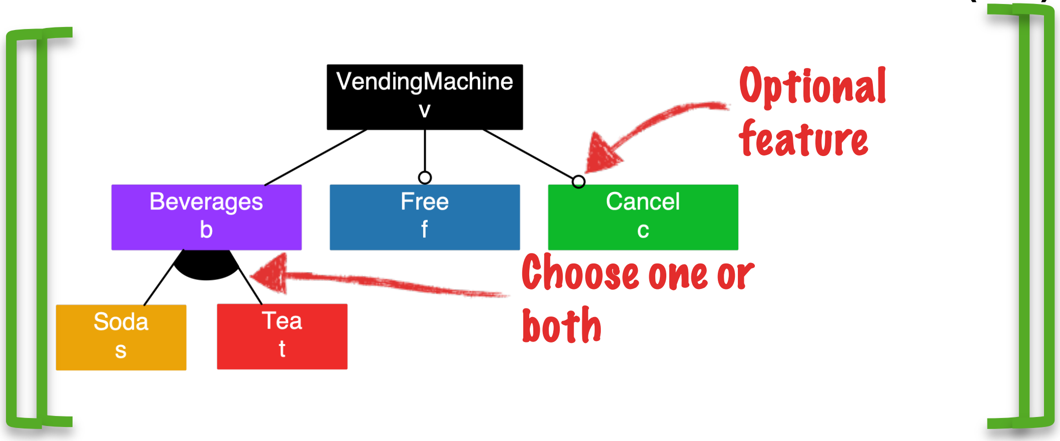
How sensitive is it to the number of configuration samples?



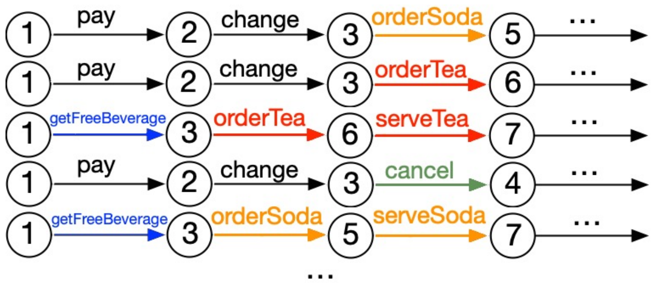
What?

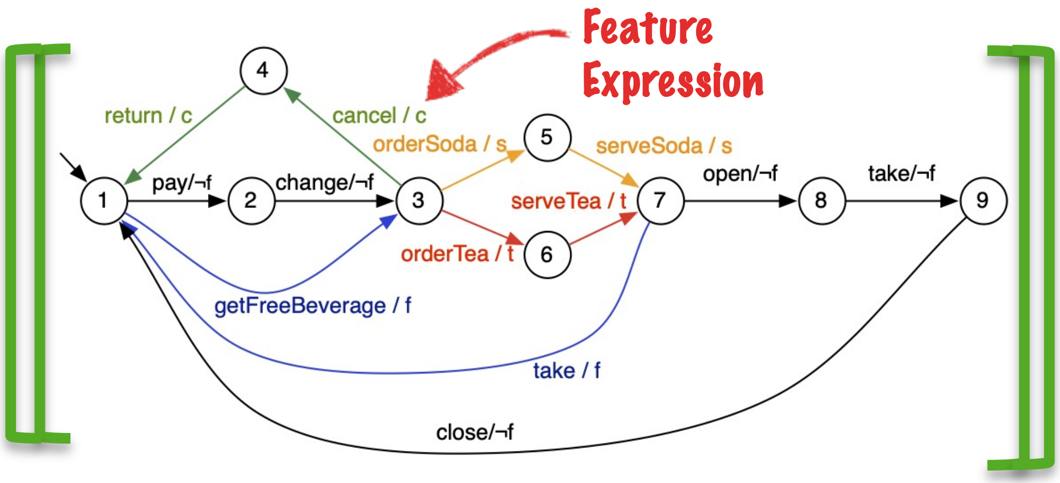


Static Variability:
Feature Model (FM)



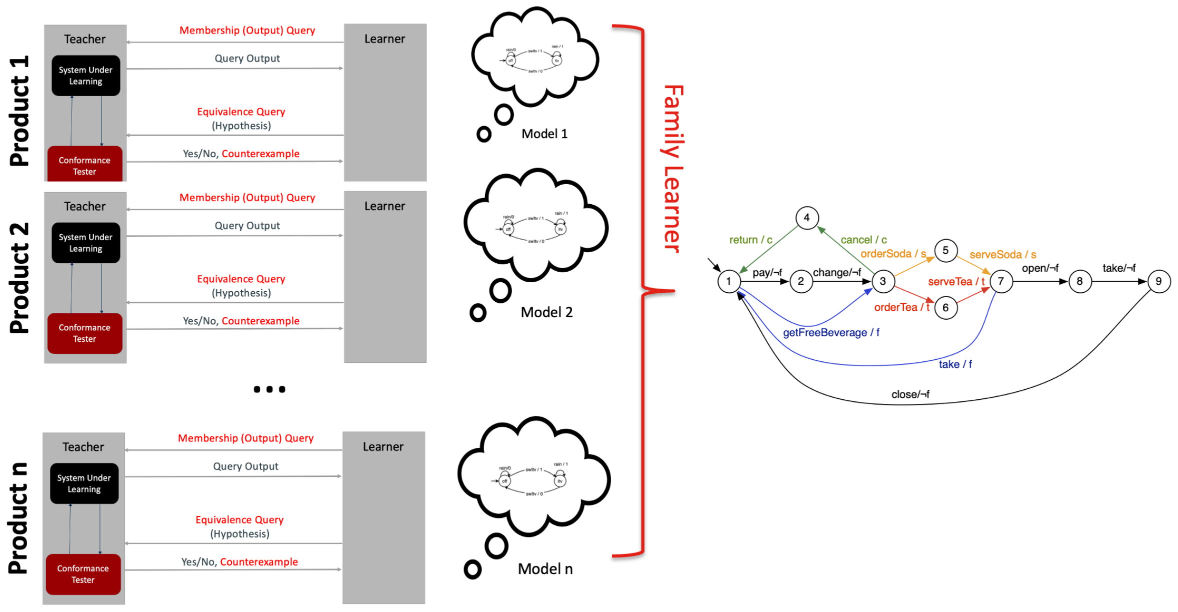
Variability-Aware Behaviour:
Featured Transition System (FTS)





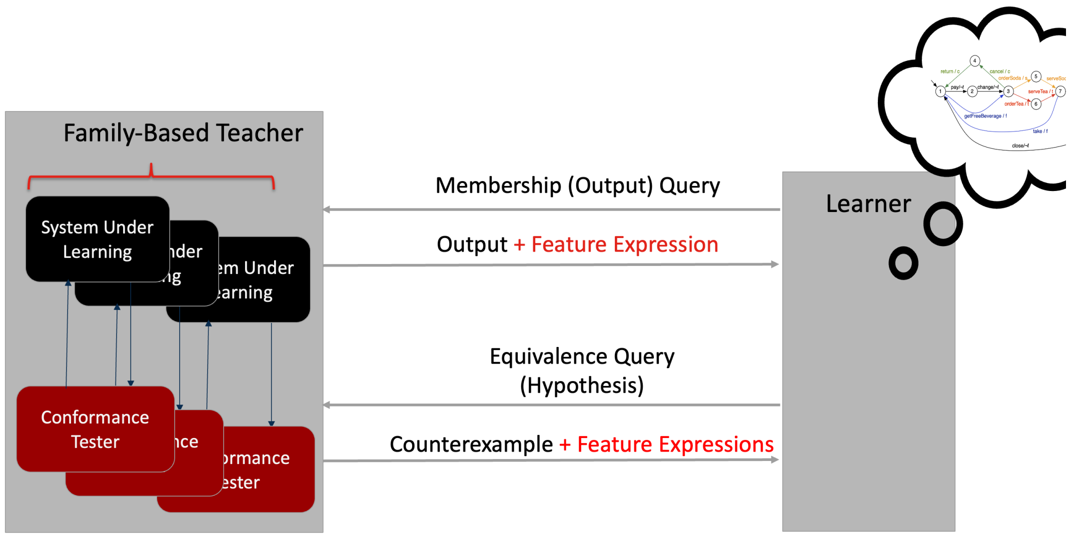
What?

Variability-Aware Learner



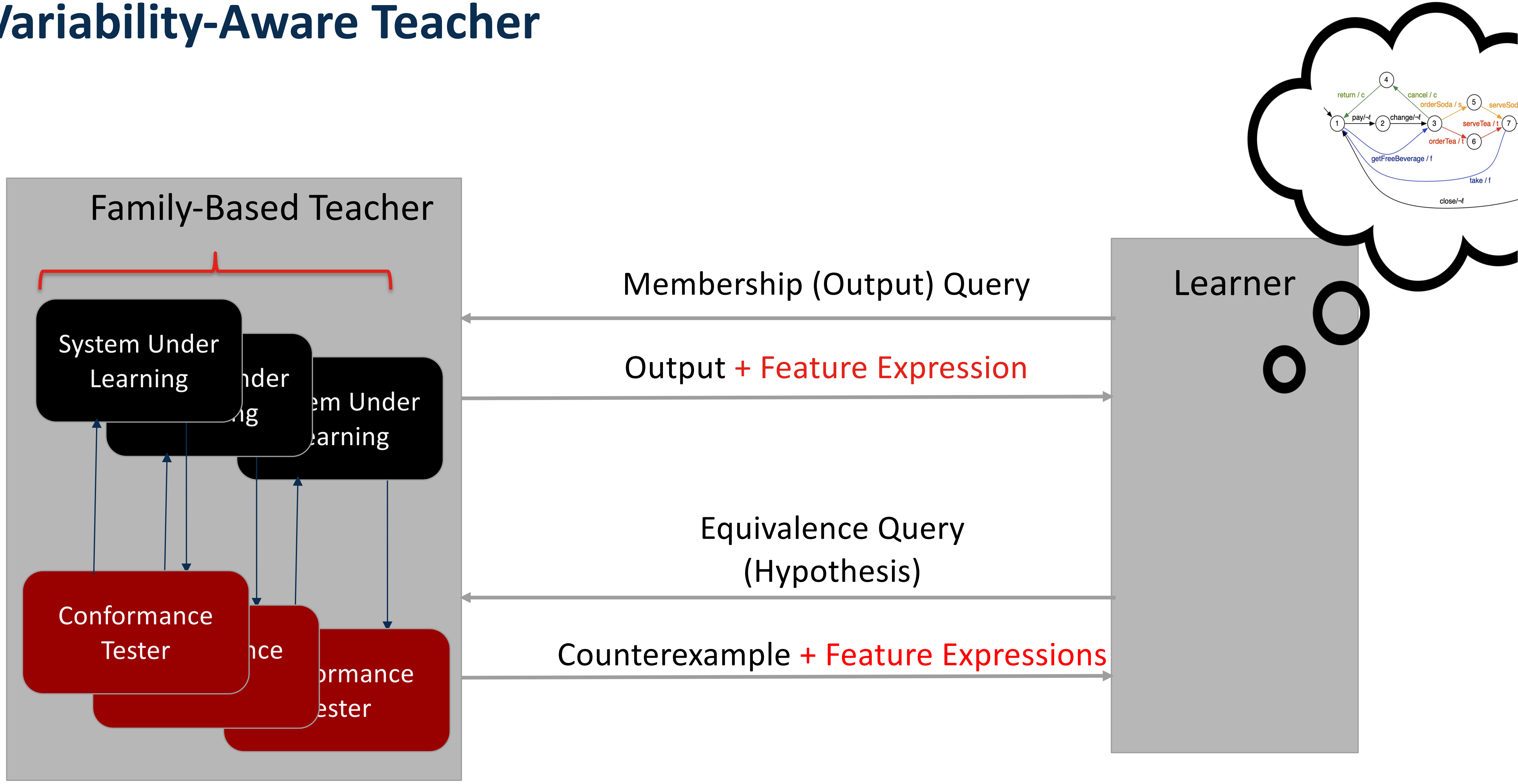
[Damasceno, Mousavi, Simao.
Learning by Sampling: Learning Behavioral Family Models from Software Product Lines. EMSE 21]
[Tavassoli, Damasceno, Khosravi, Mousavi,
Adaptive Behavioral Model Learning
for Software Product Lines. SPLC 2022]

Variability-Aware Teacher

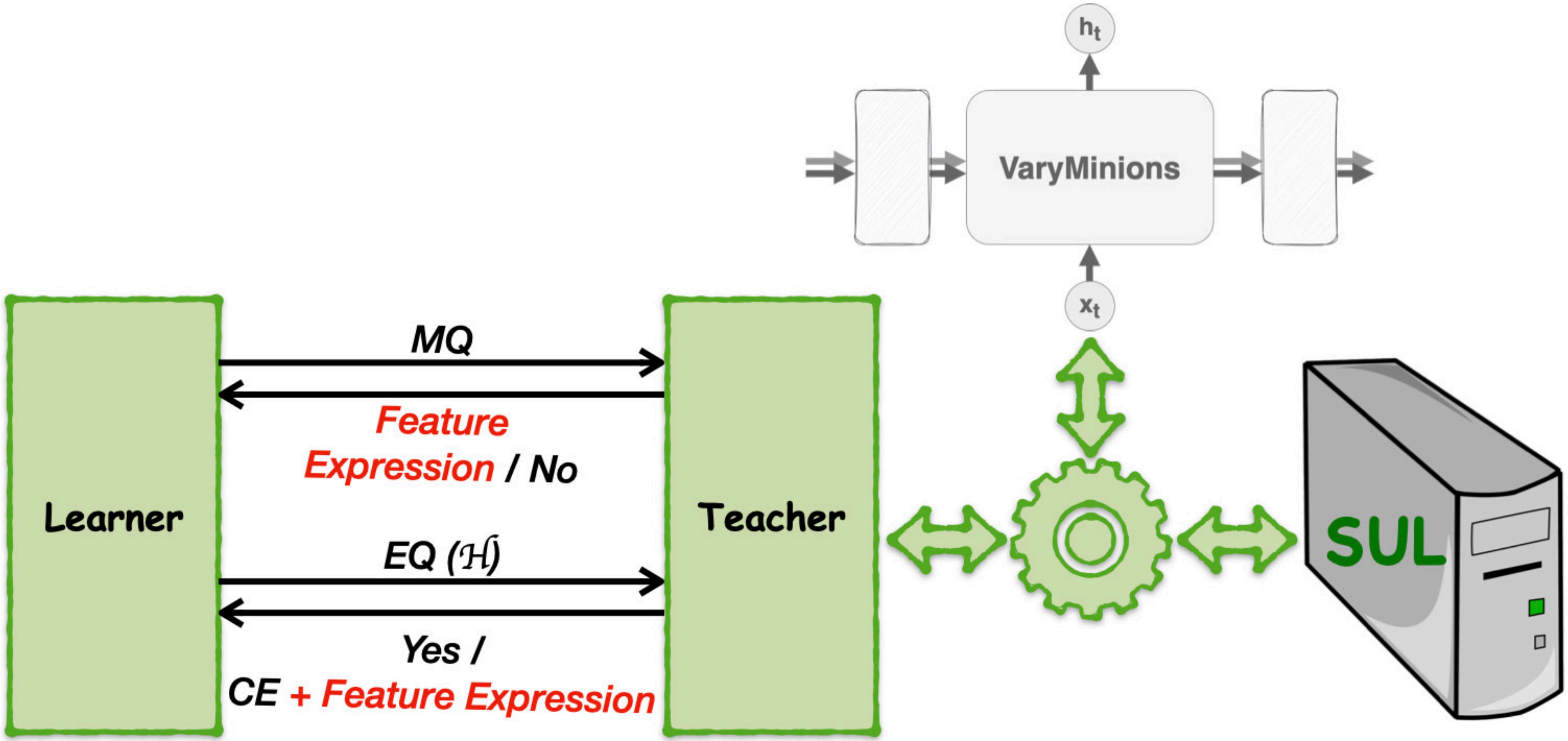


[Fortz, Temple, Devroey, Heymans, Perrouin.
VaryMinions: Leveraging RNNs to Identify Variants in Event Logs. MalTeSQuE'21]
[Fortz. *Variability-aware Behavioural Learning.*
SPLC Doctoral Sumposium'23]
[Fortz. *LIFTS: learning featured transition systems.*
SPLC Doctoral Sumposium'21]

Variability-Aware Teacher



Featured-L* (FL*)

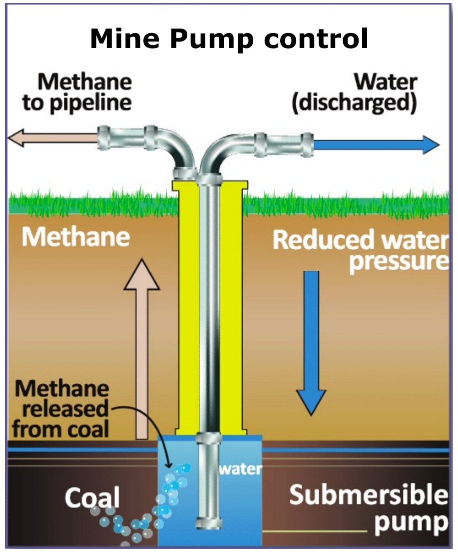


Membership query:
$\langle p, fe \rangle \cdot s \in \mathcal{L}_{SUL}$?

Equivalence query:
$\mathcal{L}(\mathcal{H}) = \mathcal{L}_{SUL}$?

Case Studies

Model	Features	Products	States	Transitions	Actions
Forum	5	6	5	5	5
SVM	9	24	9	13	13
Minepump	9	32	25	41	24
CP Terminal	21	4.774	11	17	16
Sferion™	25	64	525	46	12



Evaluation Metrics

RQ1 How to automatically learn Featured Transition Systems?

- **RQ1.1** Time
- **RQ1.2** Number of membership queries
- **RQ1.3** Number of equivalence queries and learning rounds
- **RQ1.4** Number of resets

Results

Model	Time	MQ	EQ	Rounds	Resets
Forum	1 s.	546	7	3	15.596
SVM	9 s.	19.836	23	6	219.430
Minepump	9 m.	186.984	54	11	2,934,811
CP Terminal	17 m.	39.780	33	9	436.942
Sferion™	117 m.	72.803	363	6	57,057,295

Contributions

- FE as first-class citizen
- Direct mapping on transitions
- Fully family-based approach

- Observation table of 8,904 prefixes X 21 suffixes (Minepump)
- Up to 129 states and 356 transitions (Sferion™)

- After simplification:

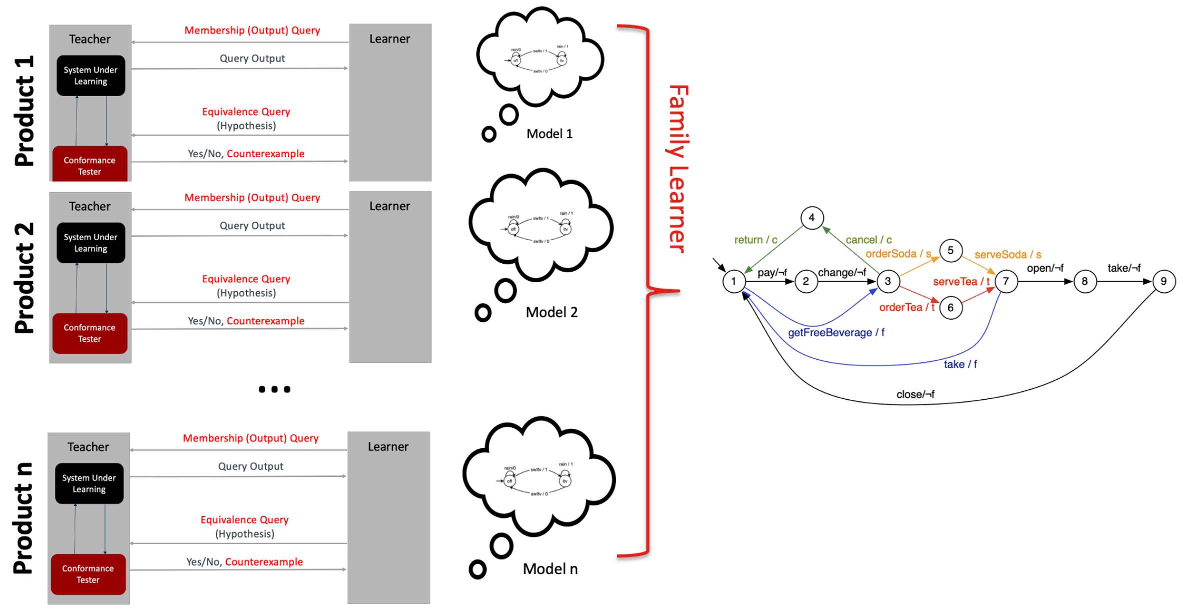
  77 % to 98 % of reduction for the observation table

  up to 80 % of reduction for the automaton (Minepump)

- Less than 2 hours of execution for each study (2 Cores, 2 Sockets, 16Go RAM)

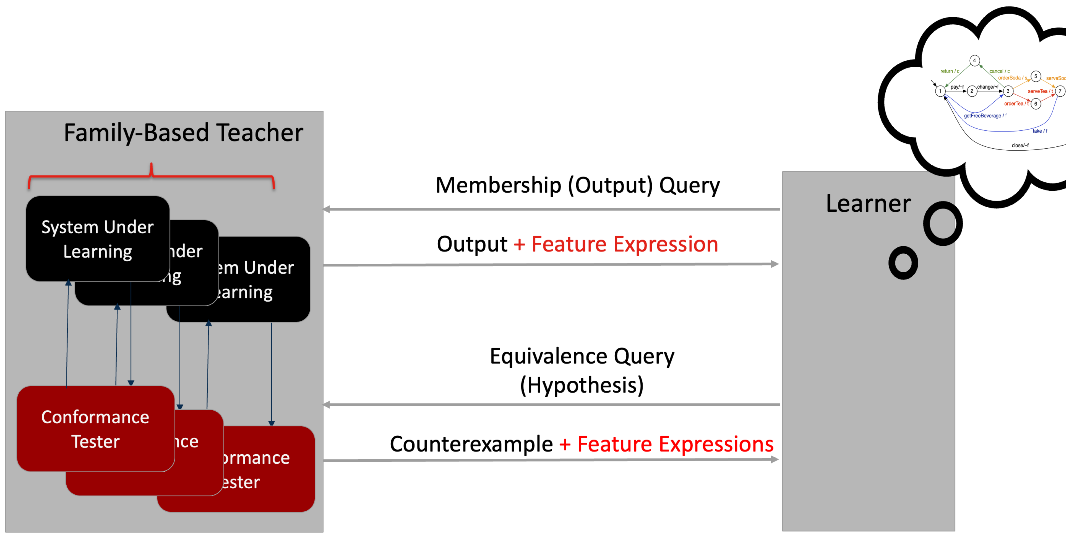
What?

Variability-Aware Learner



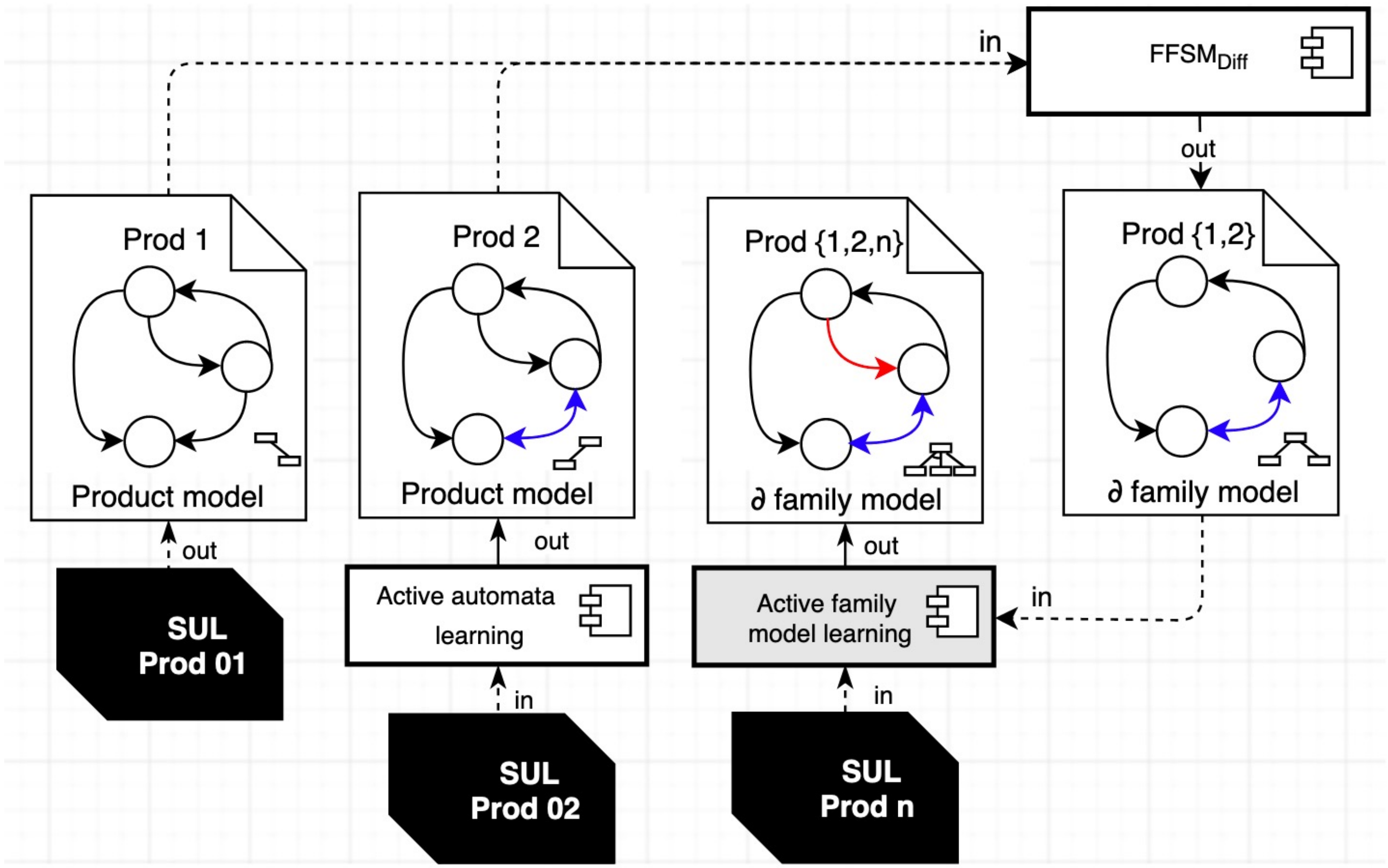
[Damasceno, Mousavi, Simao.
Learning by Sampling: Learning Behavioral Family Models from Software Product Lines. EMSE 21]
[Tavassoli, Damasceno, Khosravi, Mousavi,
Adaptive Behavioral Model Learning
for Software Product Lines. SPLC 2022]

Variability-Aware Teacher



[Fortz, Temple, Devroey, Heymans, Perrouin.
VaryMinions: Leveraging RNNs to Identify Variants in Event Logs. MalTeSQuE'21]
[Fortz.
SPLC Doctoral Sumposium'21 and '23]

What?



[Damasceno, Mousavi, Simao.
Learning by Sampling: Learning Behavioral Family Models from Software Product Lines. EMSE 21]

How?

$$S^G_{Succ}(a,b) = \frac{1}{2}\frac{\sum_{(c,d,i,o)\ \in\ Succ_{a,b}}(1 + k \times S^G_{Succ}(c,d))}{|\sum^{out}_r(a) - \sum^{out}_u(b)| + |\sum^{out}_r(b) - \sum^{out}_u(a)| + |Succ_{a,b}|}$$

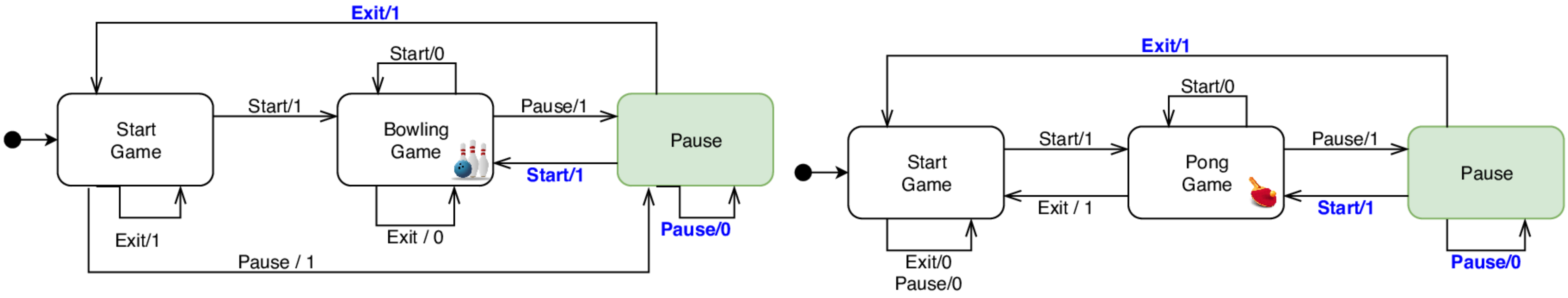
Global similarity score (Outgoing and incoming transitions)

- Pairwise similarity based on surrounding matching transitions and connected state pairs.
- Attenuation ratio k gives precedence to the closest state pairs.
- Matching transitions and distinct transitions.

[N. Walkinshaw and K. Bogdanov,
Automated Comparison of State-Based Software Models
in Terms of Their Language and Structure.]

How?

$$S^G_{Succ}(Pa, Pa) = \frac{1}{2} \times \frac{3 + k \times [S^G_{Succ}(St, St) + S^G_{Succ}(Bo, Po) + S^G_{Succ}(Pa, Pa)]}{0 + 0 + 3} = 0.58$$



The $\text{FFSM}_{\text{Diff}}$ algorithm

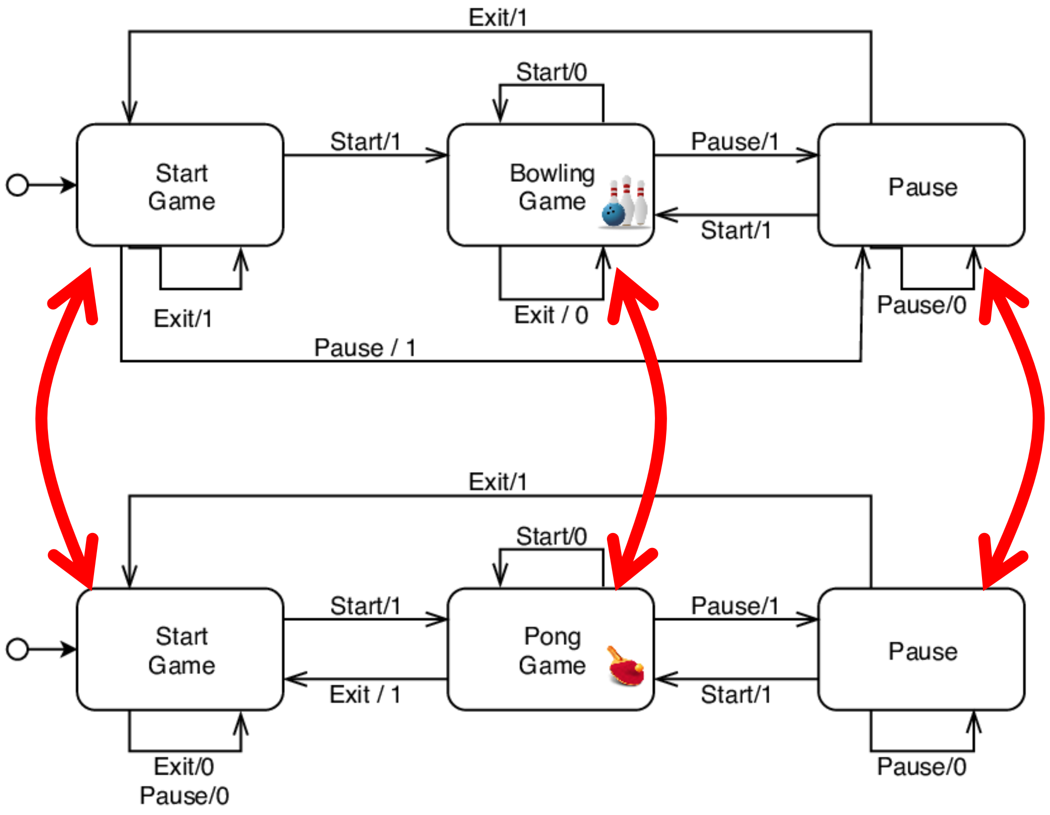


Figure: Two examples of product FSMs

$\text{pair}(\text{St}, \text{St}) = 0.12$

$pair(St, Po) = 0.29$

$pair(St, Pa) = 0.28$

$pair(Bo, St) = 0.11$

$\text{pair}(\text{Bo}, \text{Po}) = 0.31$

$pair(Bo, Pa) = 0$

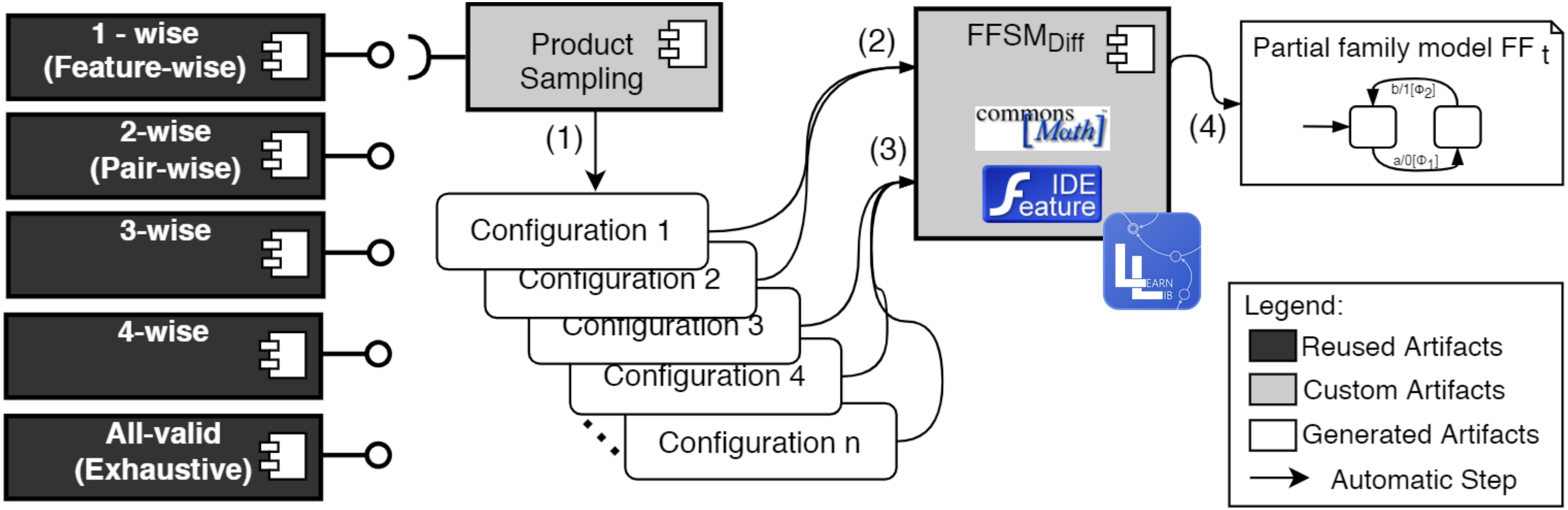
$pair(Pa, St) = 0.29$

$pair(Pa, Po) = 0.11$

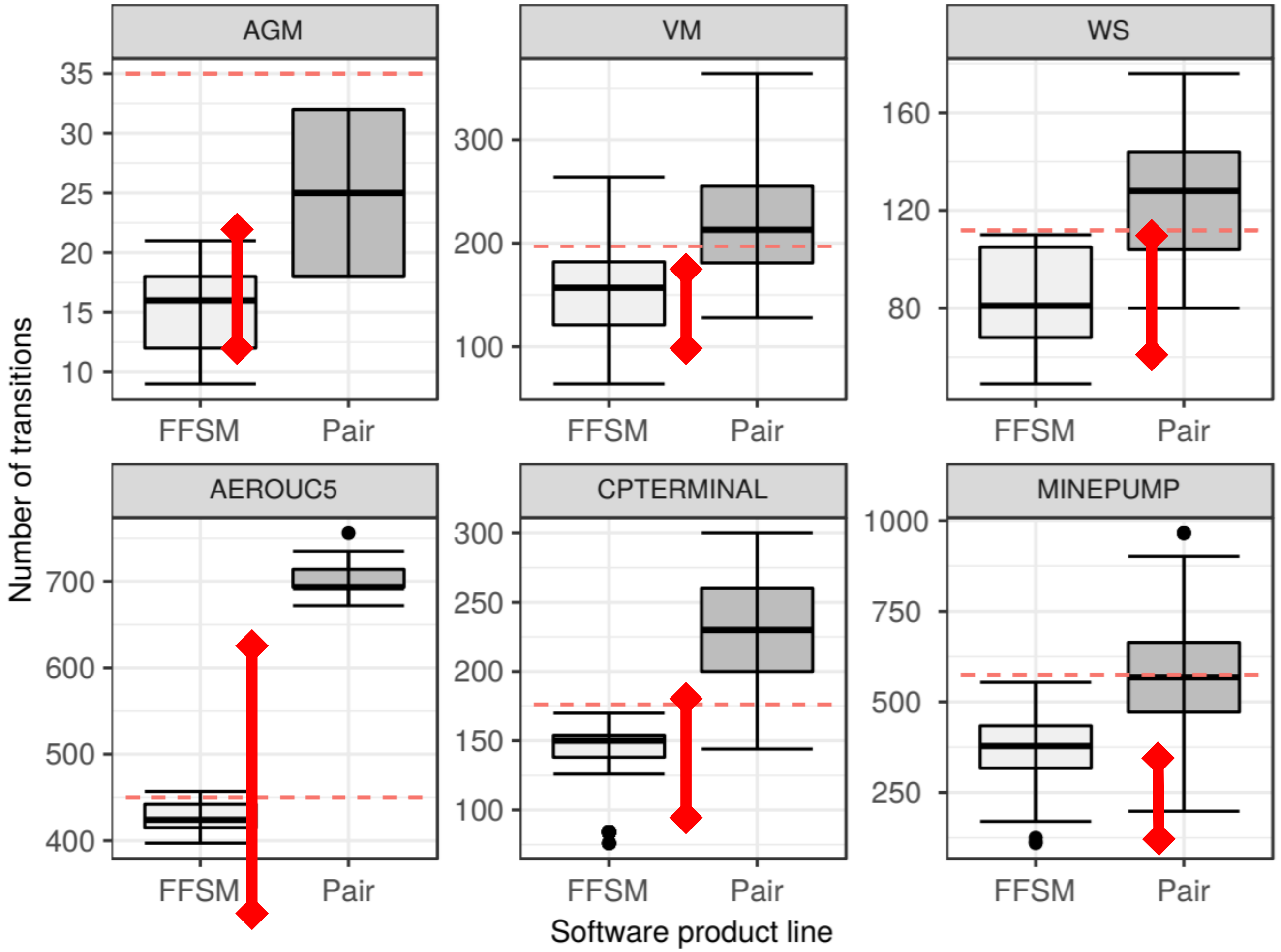
$\text{pair}(\text{Pa}, \text{Pa}) = 0.58$

Figure: Pairwise state similarity

Experiment Design

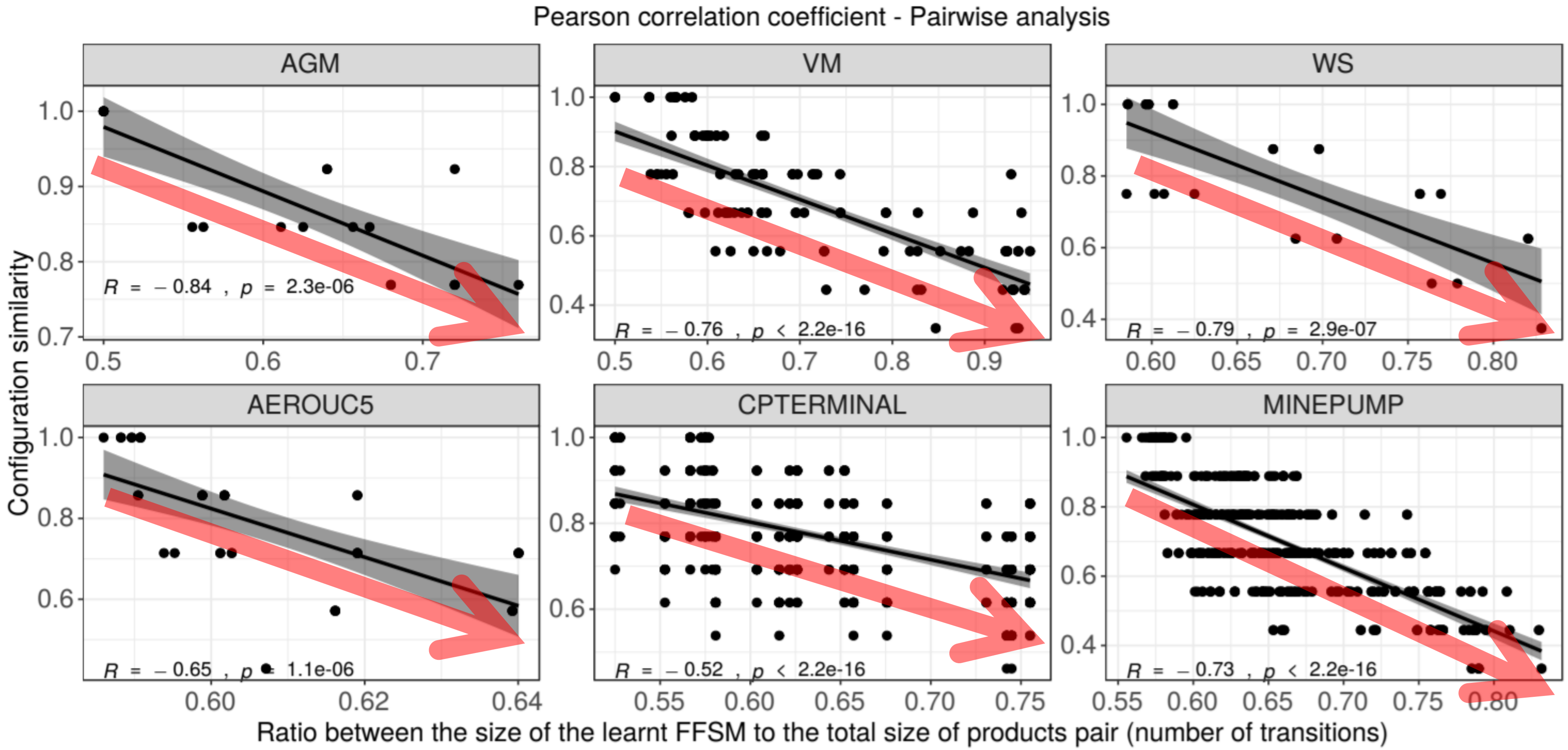


Analysis of Results: Size

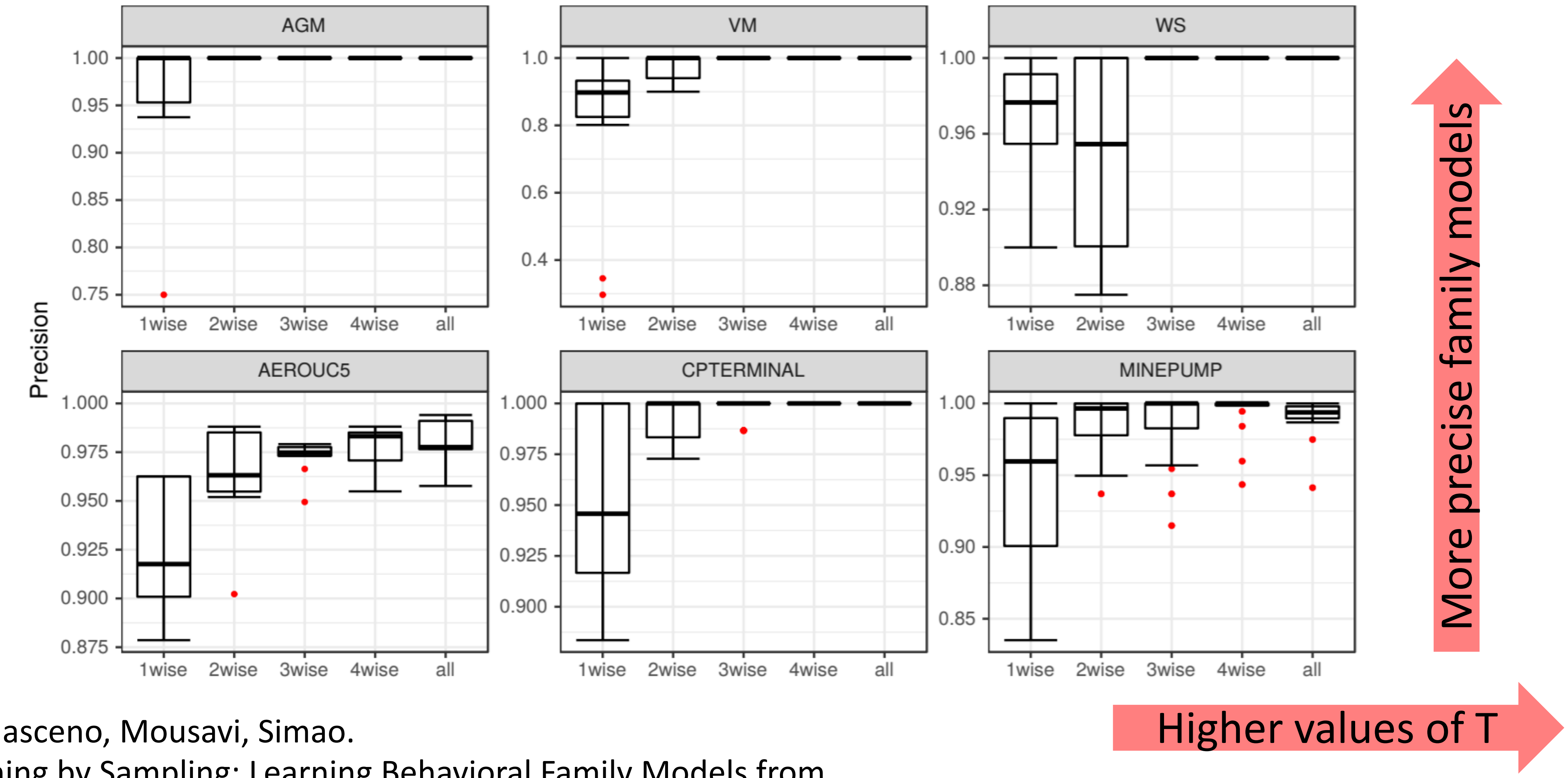


Mann-Whitney-Wilcoxon statistical test
Vargha-Delaney's Â effect size

Analysis of Results



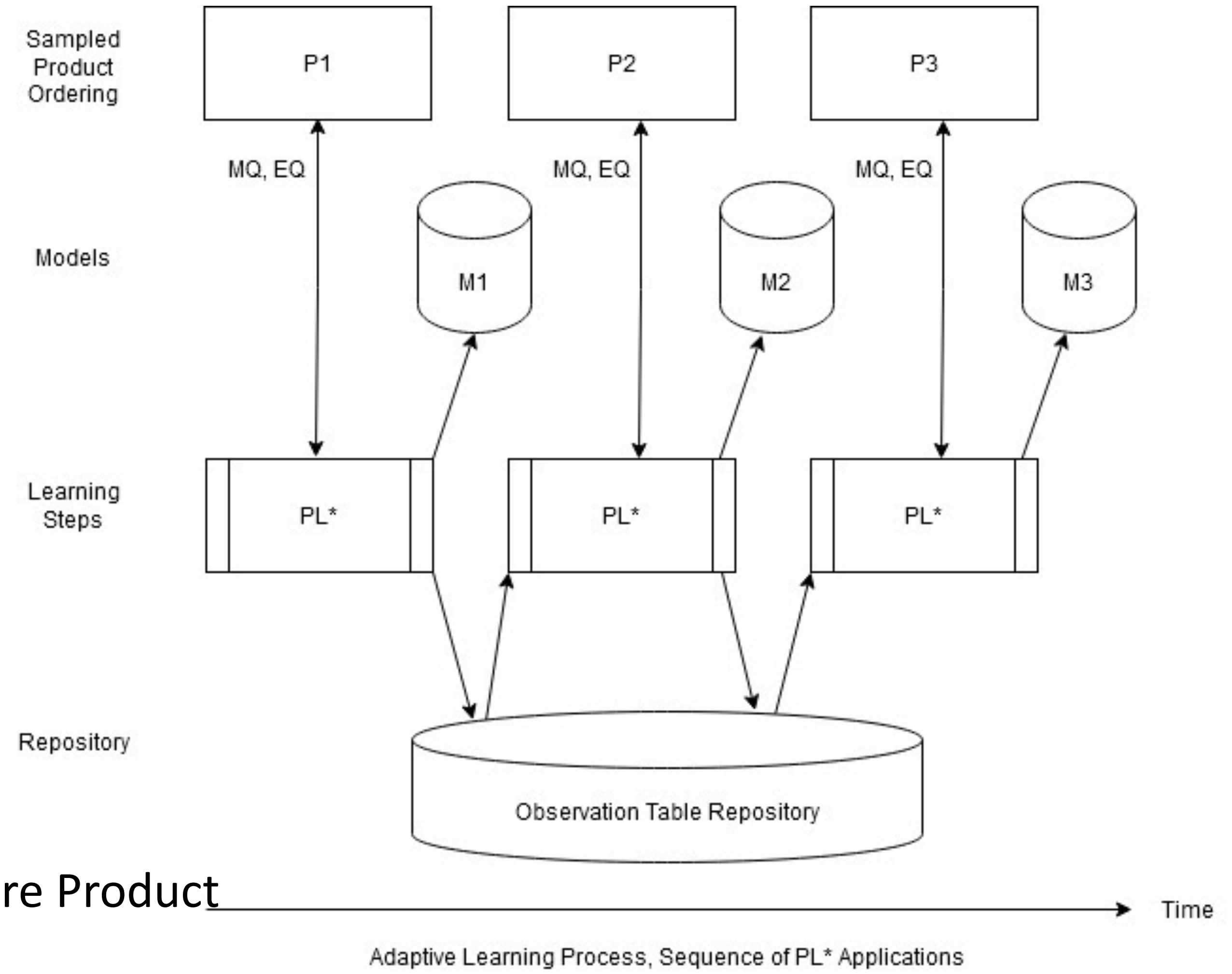
Analysis of Results: Sampling



[Damasceno, Mousavi, Simao.
Learning by Sampling: Learning Behavioral Family Models from Software Product Lines. EMSE 21]

PL*

Building a repository of queries that for changes in space

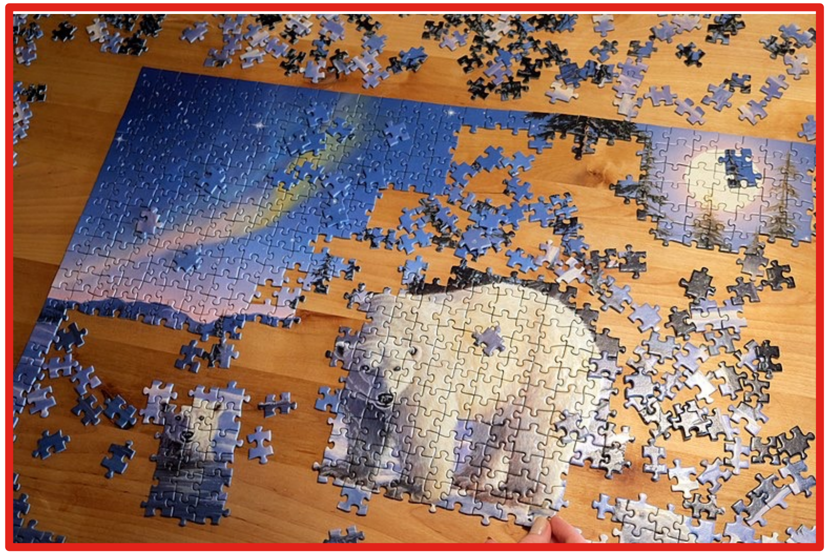


[Tavassoli, Damasceno, Khosravi, Mousavi, Adaptive Behavioral Model Learning for Software Product Lines. SPLC 2022]

Active Automata Learning

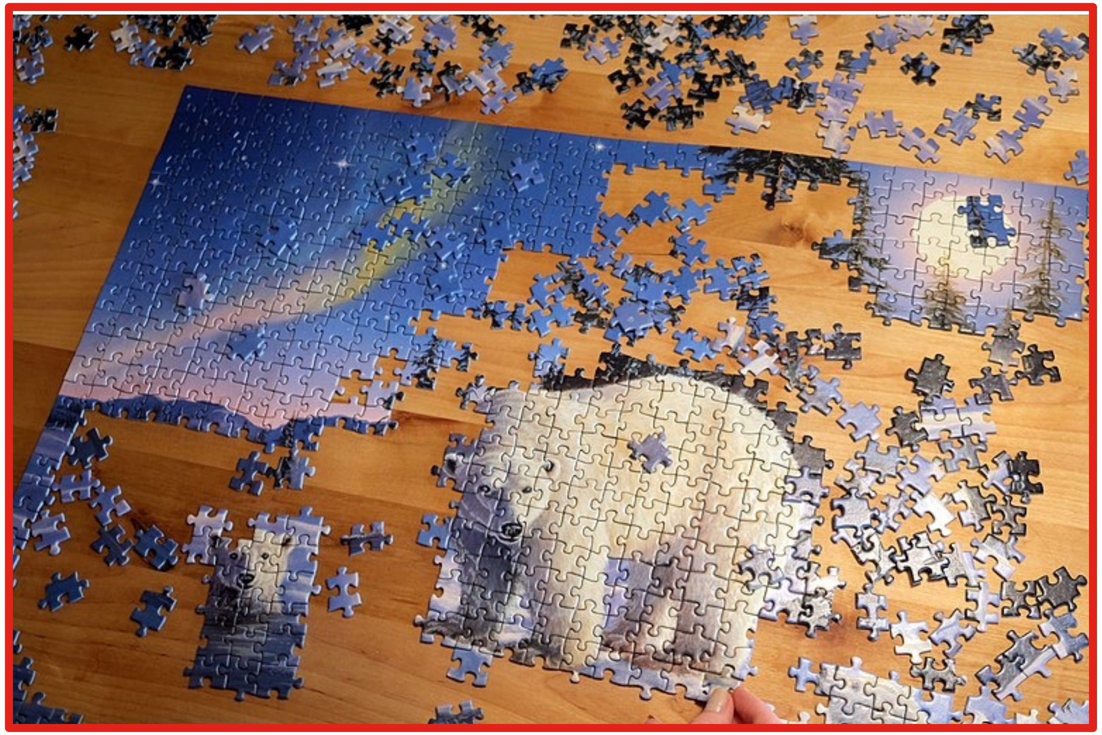
Adaptive Learning

Produc Line Learning

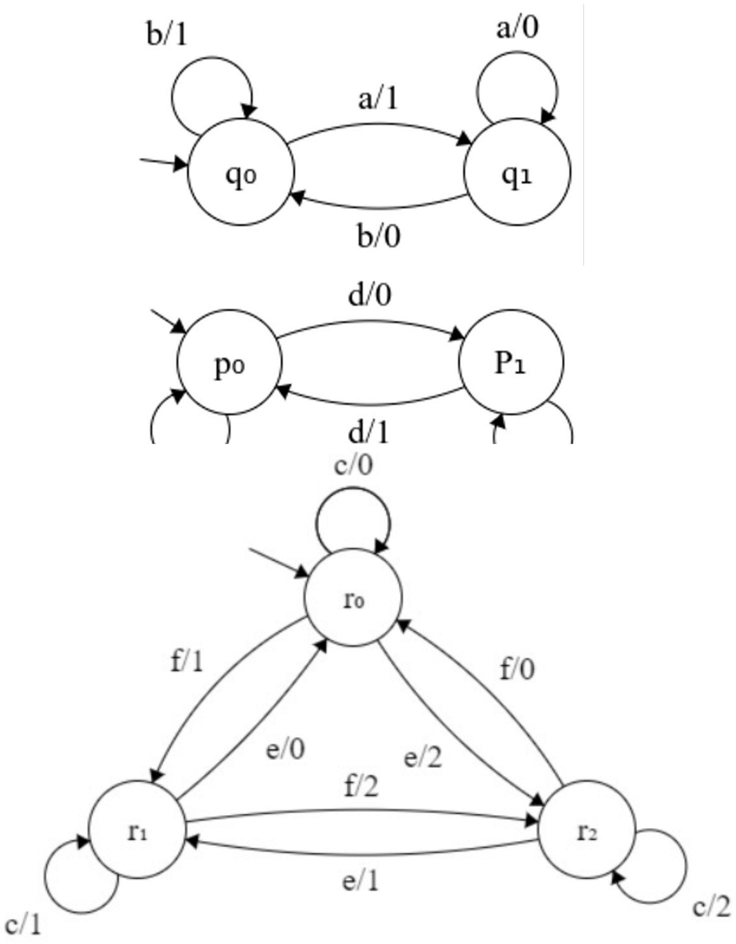


Compositional Learning

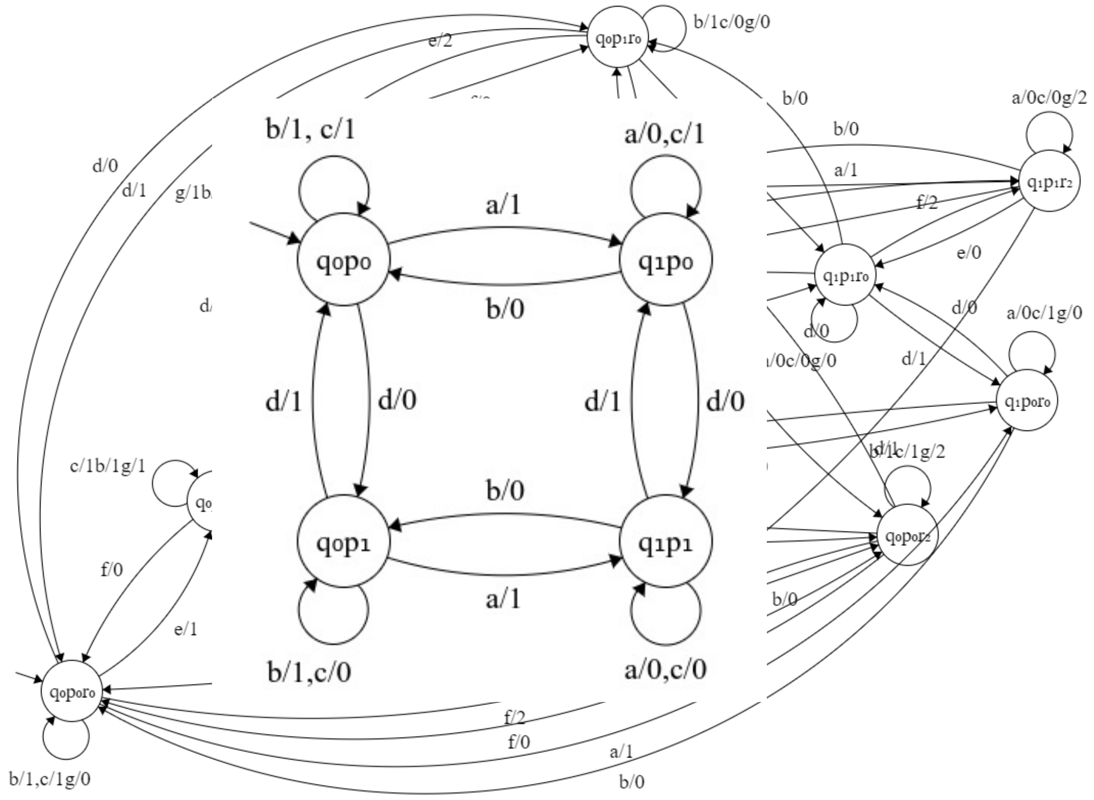
Compositional Learning



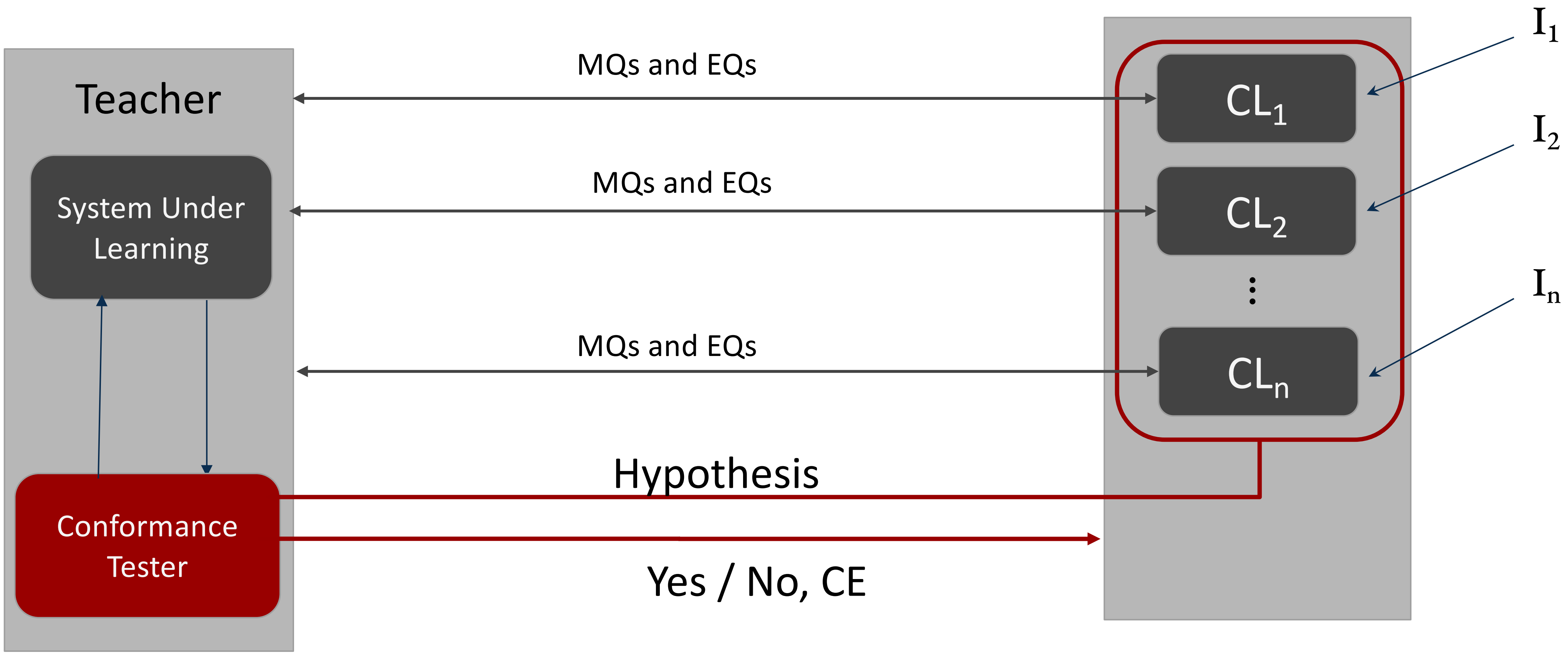
Interleaving Parallel Systems



Interleaving Parallel Composition

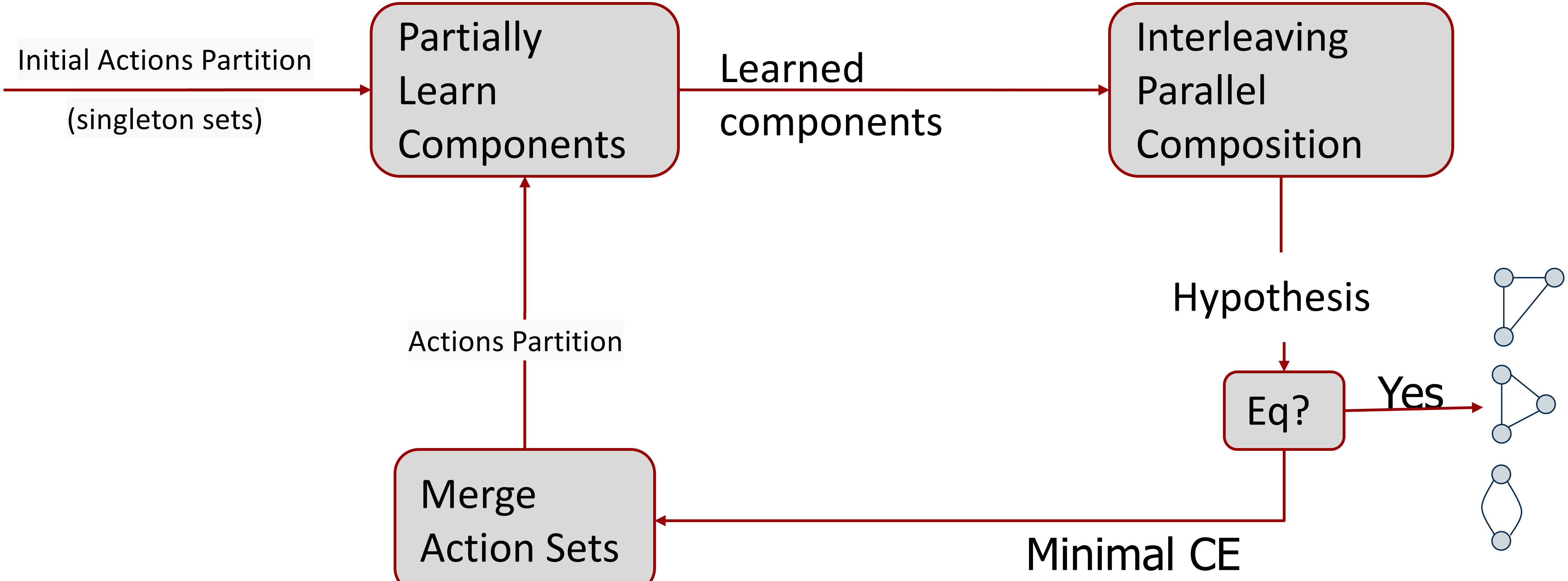


CL*



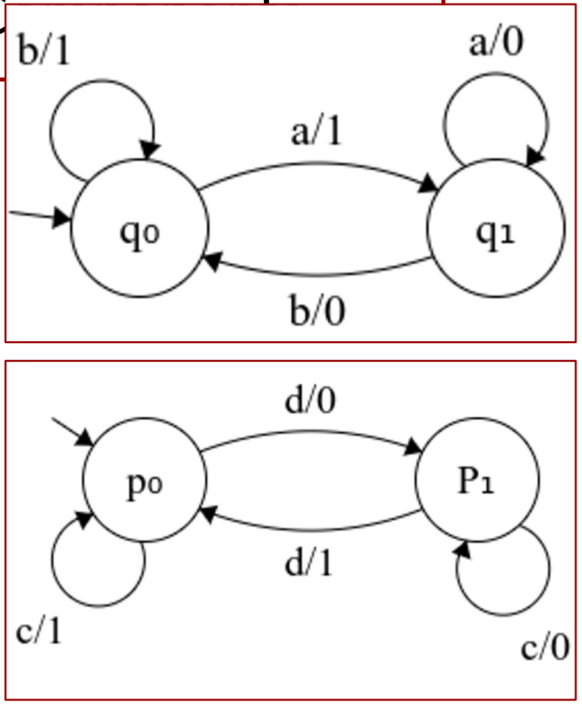
Teacher
System Under Learning
Conformance Tester
MQs and EQs
MQs and EQs
MQs and EQs
Hypothesis
Yes / No, CE
CL_1
CL_2
CL_n
I_1
I_2
I_n

CL* Algorithm

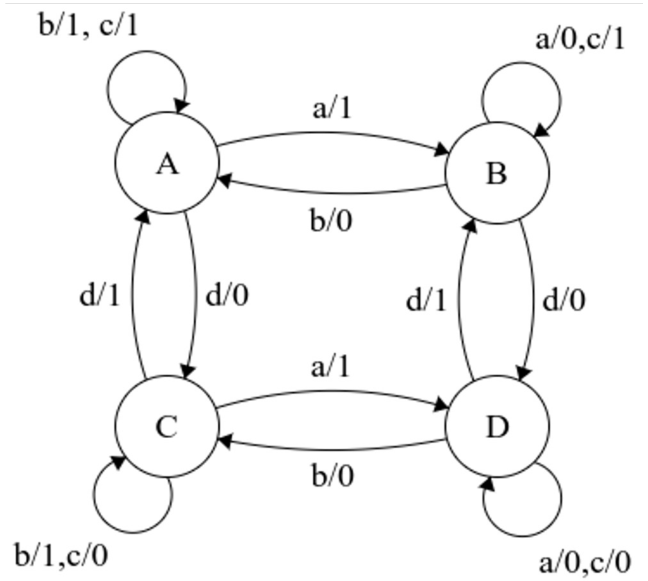


CL* Algorithm - Example

SUL's
C



SUL



$$I = \{a, b, c, d\}$$
$$O = \{0, 1\}$$

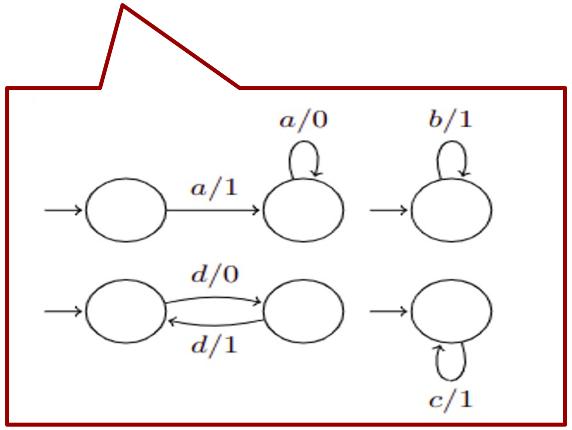
CL* Algorithm - Example

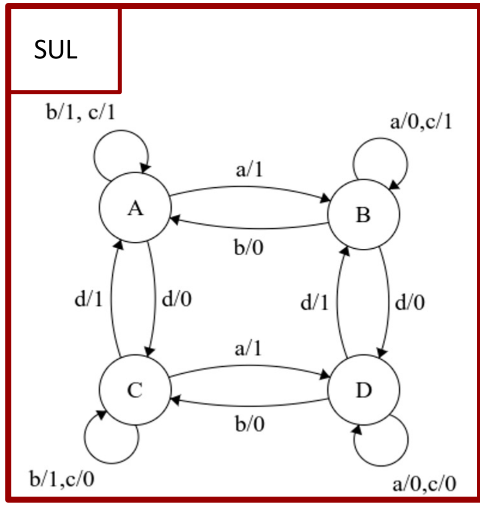
Initial Actions Partition

{a}, {b}, {c}, {d}

Partially Learn Components

Learned components





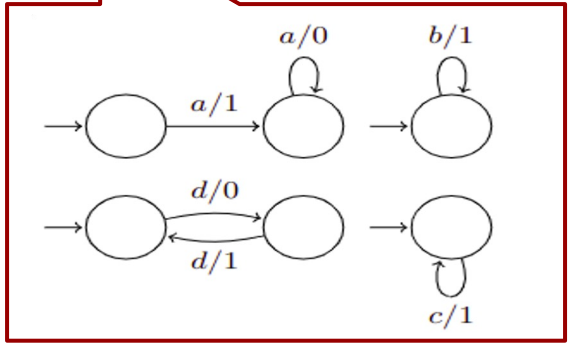
CL* Algorithm - Example

Initial Actions Partition

{a}, {b}, {c}, {d}

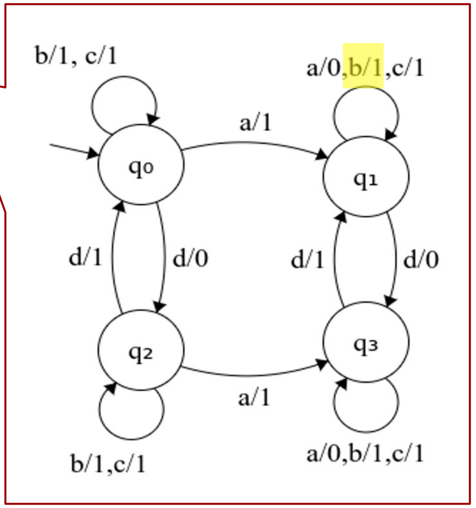
Partially Learn Components

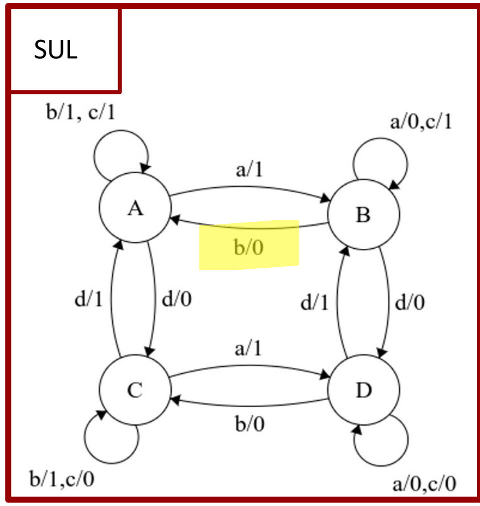
Learned components



Interleaving Parallel Composition

Hypothesis





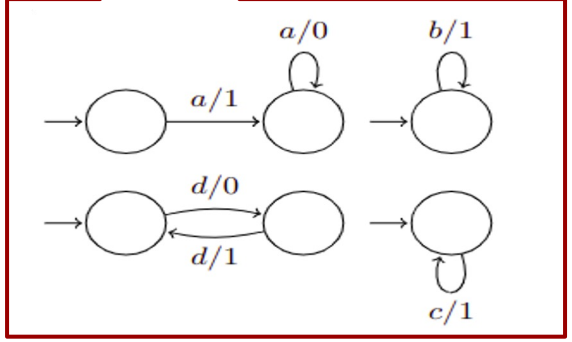
CL* Algorithm - Example

Initial Actions Partition

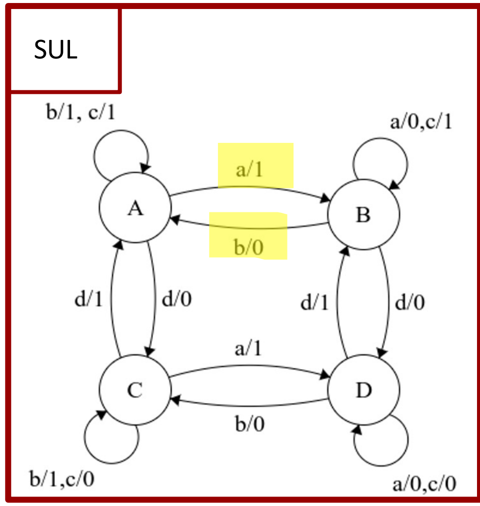
{a}, {b}, {c}, {d}

Partially Learn Components

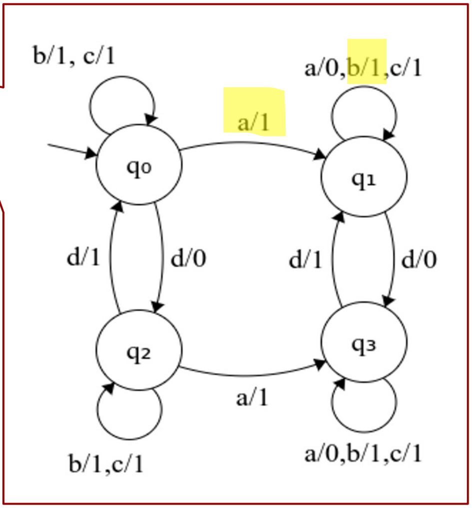
Learned components



Interleaving Parallel Composition



Hypothesis



CE = ab

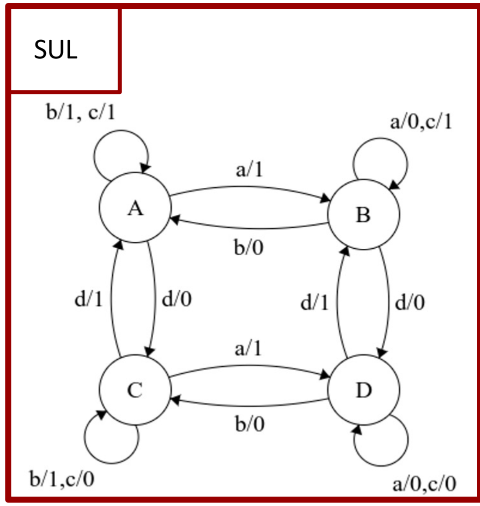
M(ab) = 10

H(ab)=11

Eq?

Minimal CE

CL* Algorithm - Example

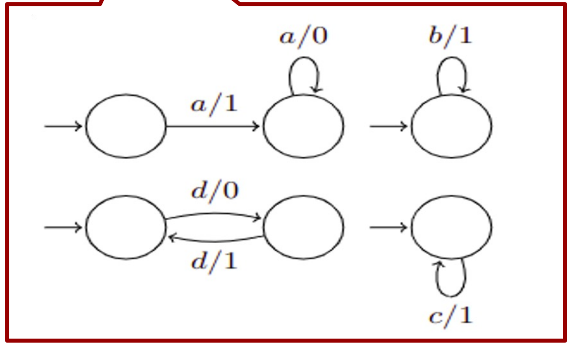


Initial Actions Partition

{a}, {b}, {c}, {d}

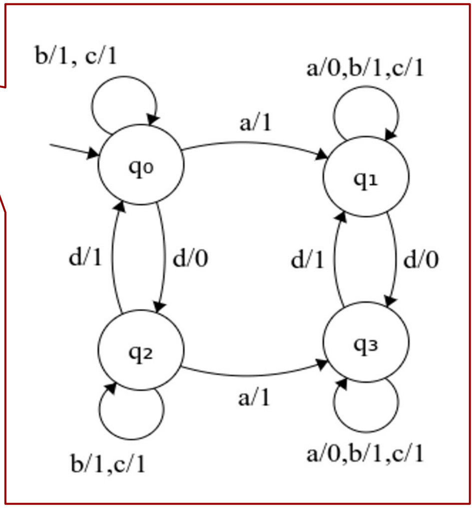
Partially Learn Components

Learned components



Interleaving Parallel Composition

Hypothesis



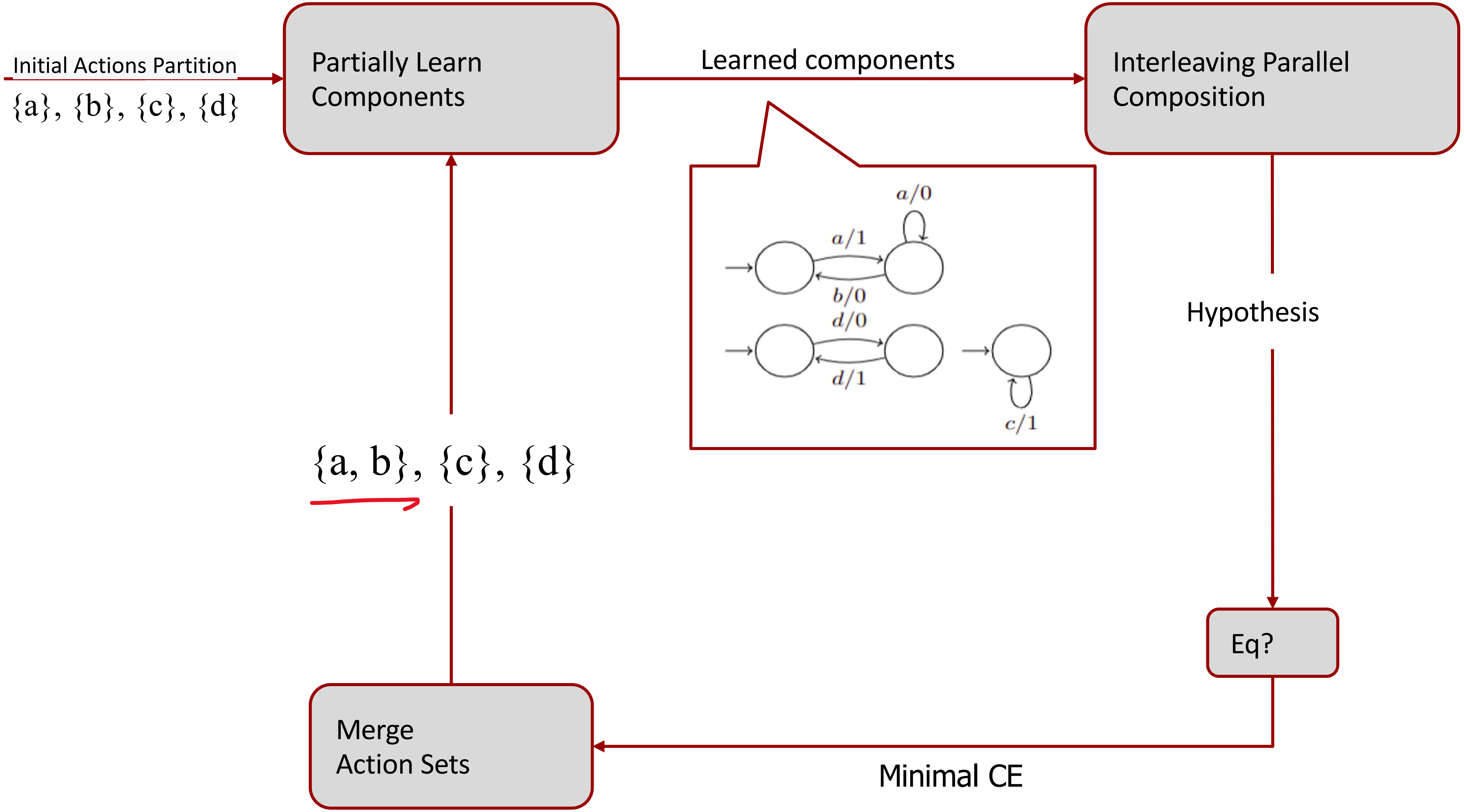
CE = ab
M(ab) = 10
H(ab)=11

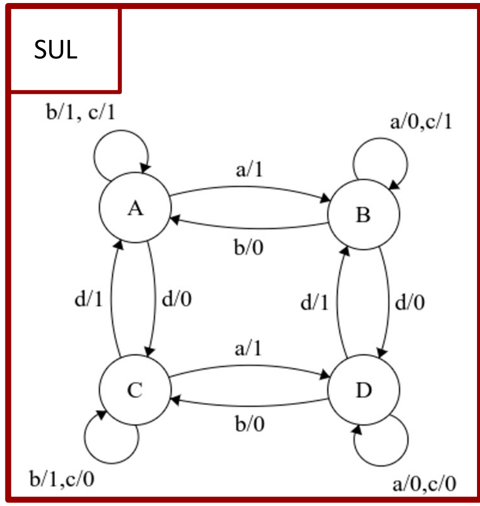
Eq?

Merge Action Sets

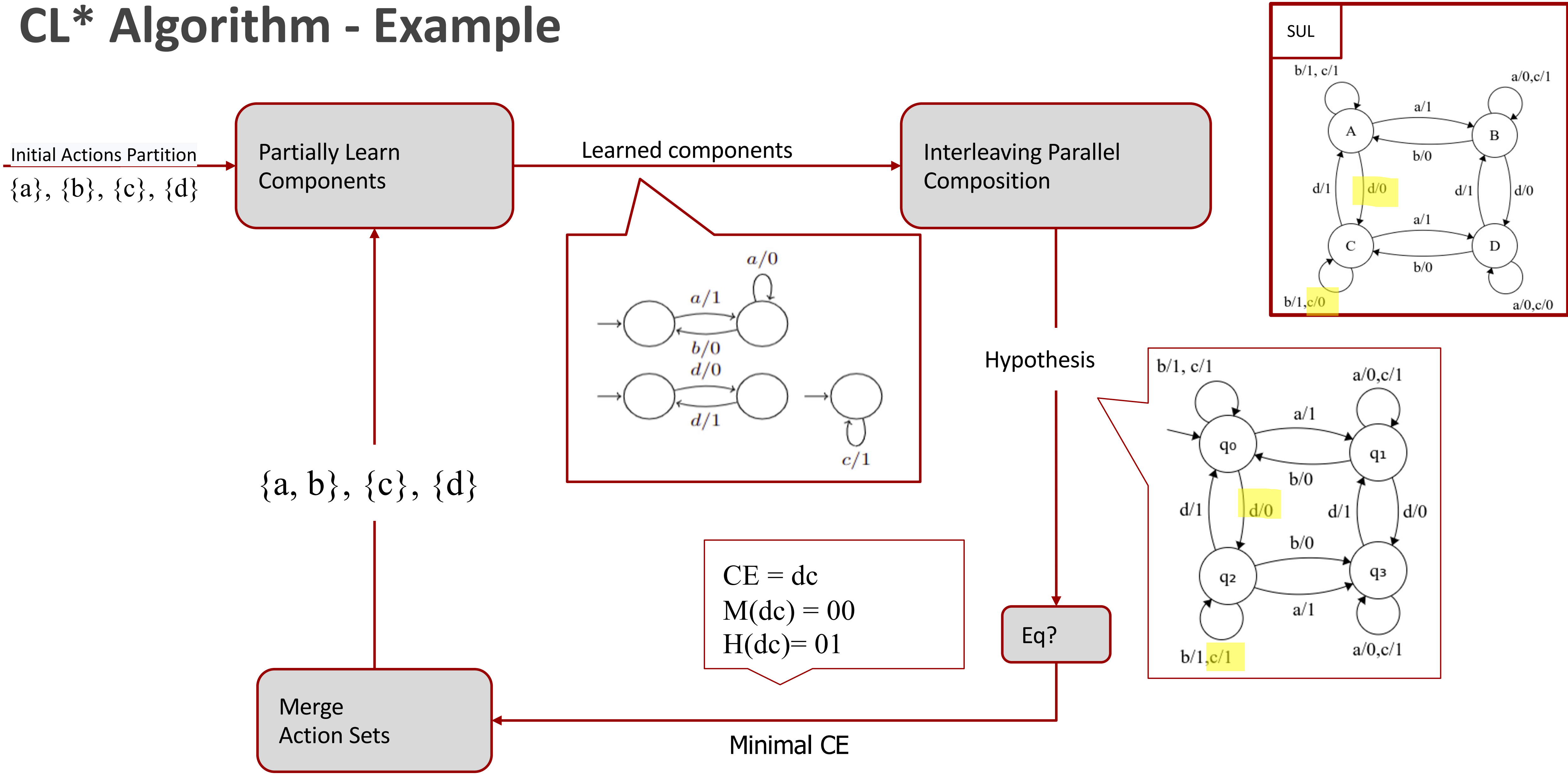
Minimal CE

CL* Algorithm - Example

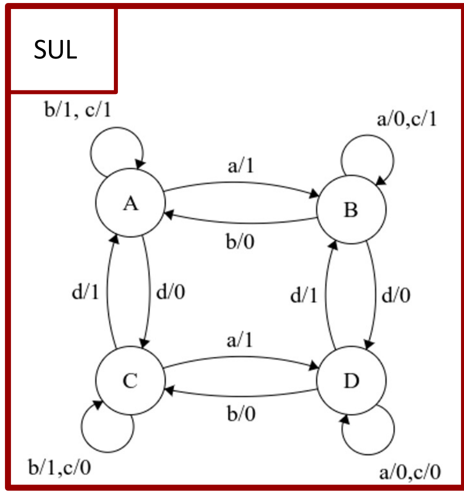




CL* Algorithm - Example



CL* Algorithm - Example

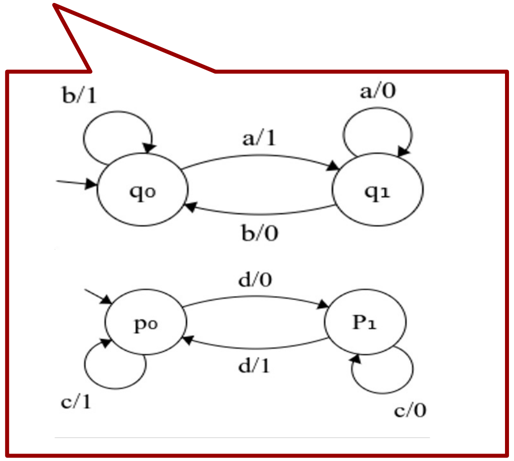


Initial Actions Partition

{a}, {b}, {c}, {d}

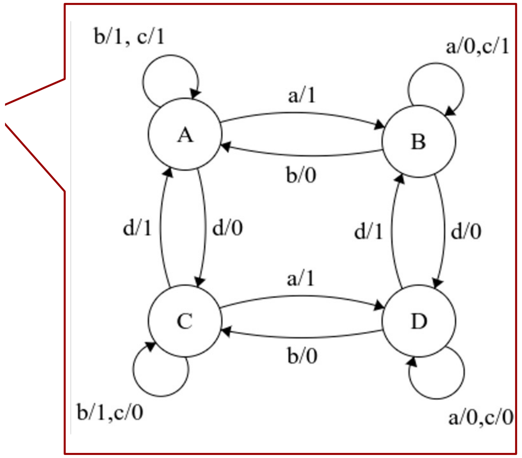
Partially Learn Components

Learned components



Interleaving Parallel Composition

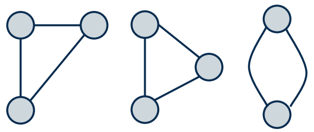
Hypothesis



{a, b}, {c, d}

Eq?

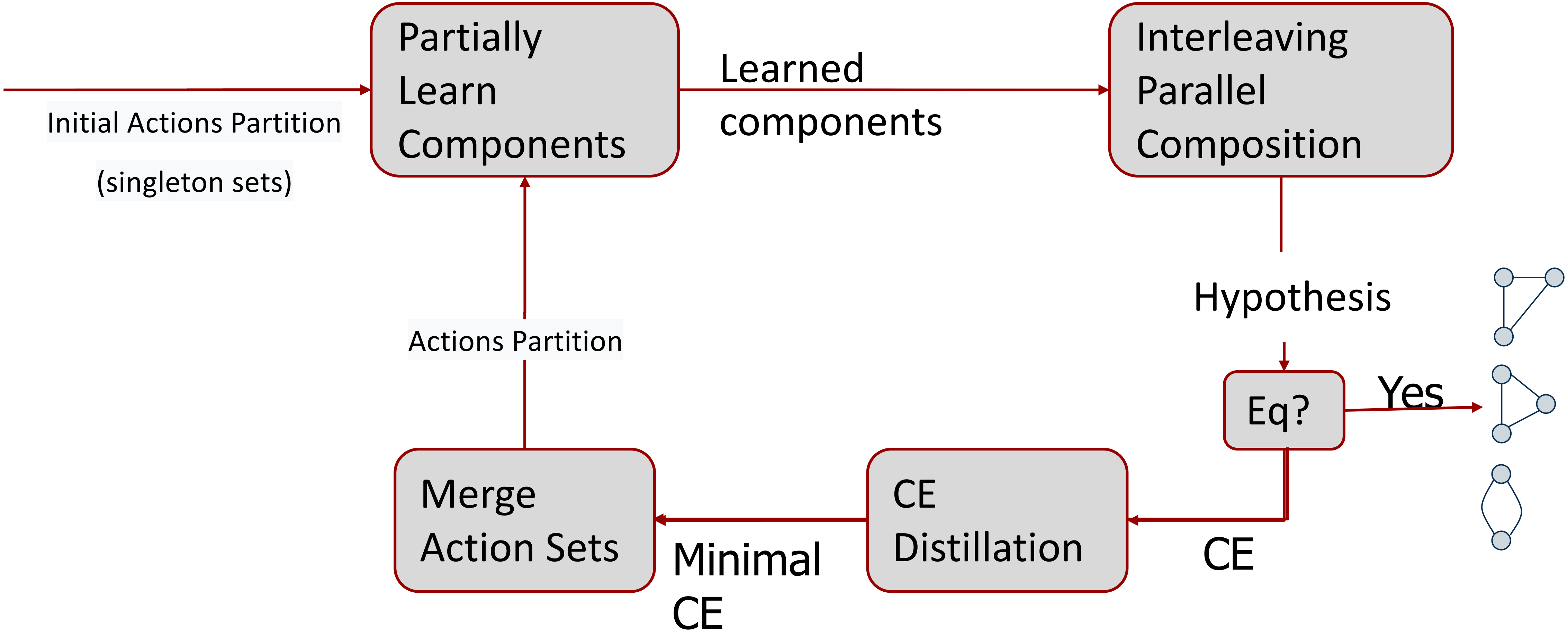
Yes



Merge Action Sets

Minimal CE

CL* Algorithm

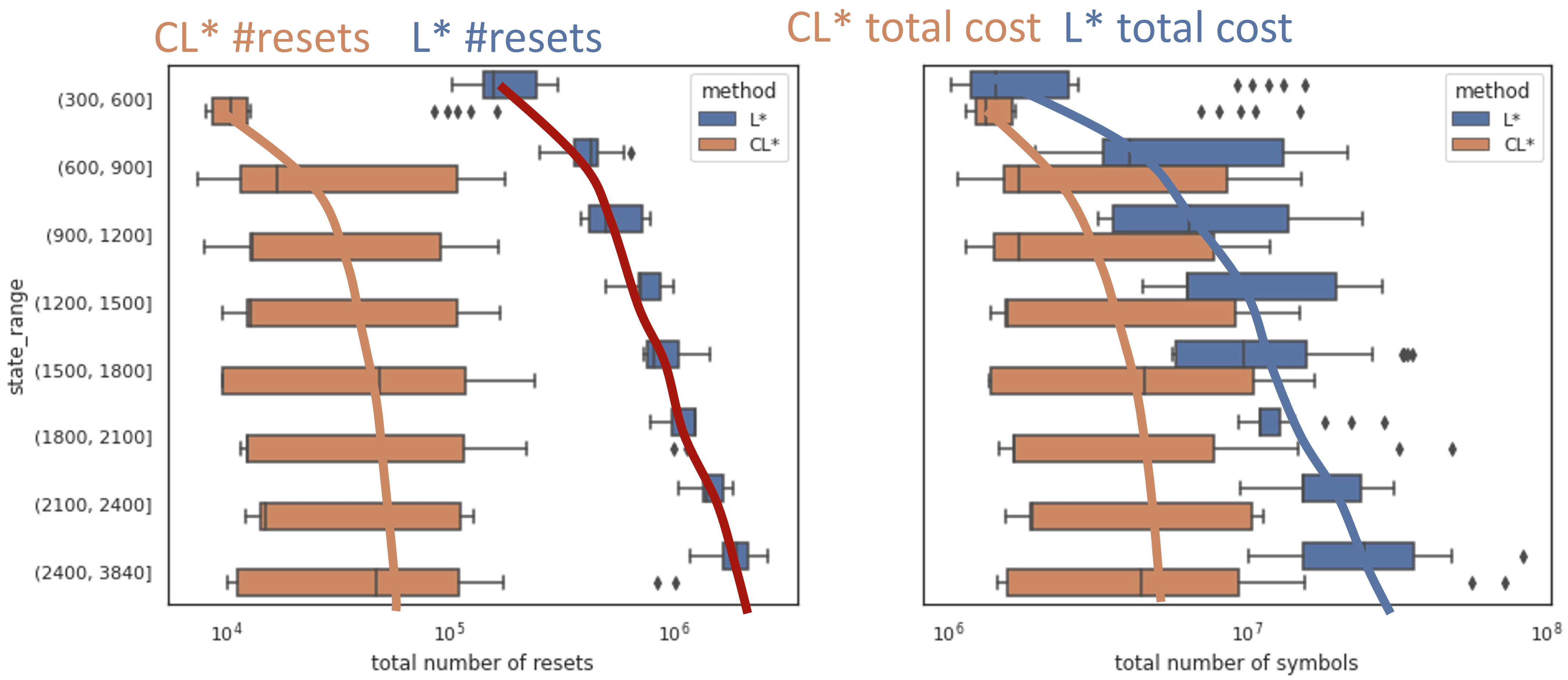


Experiments: Subject Systems

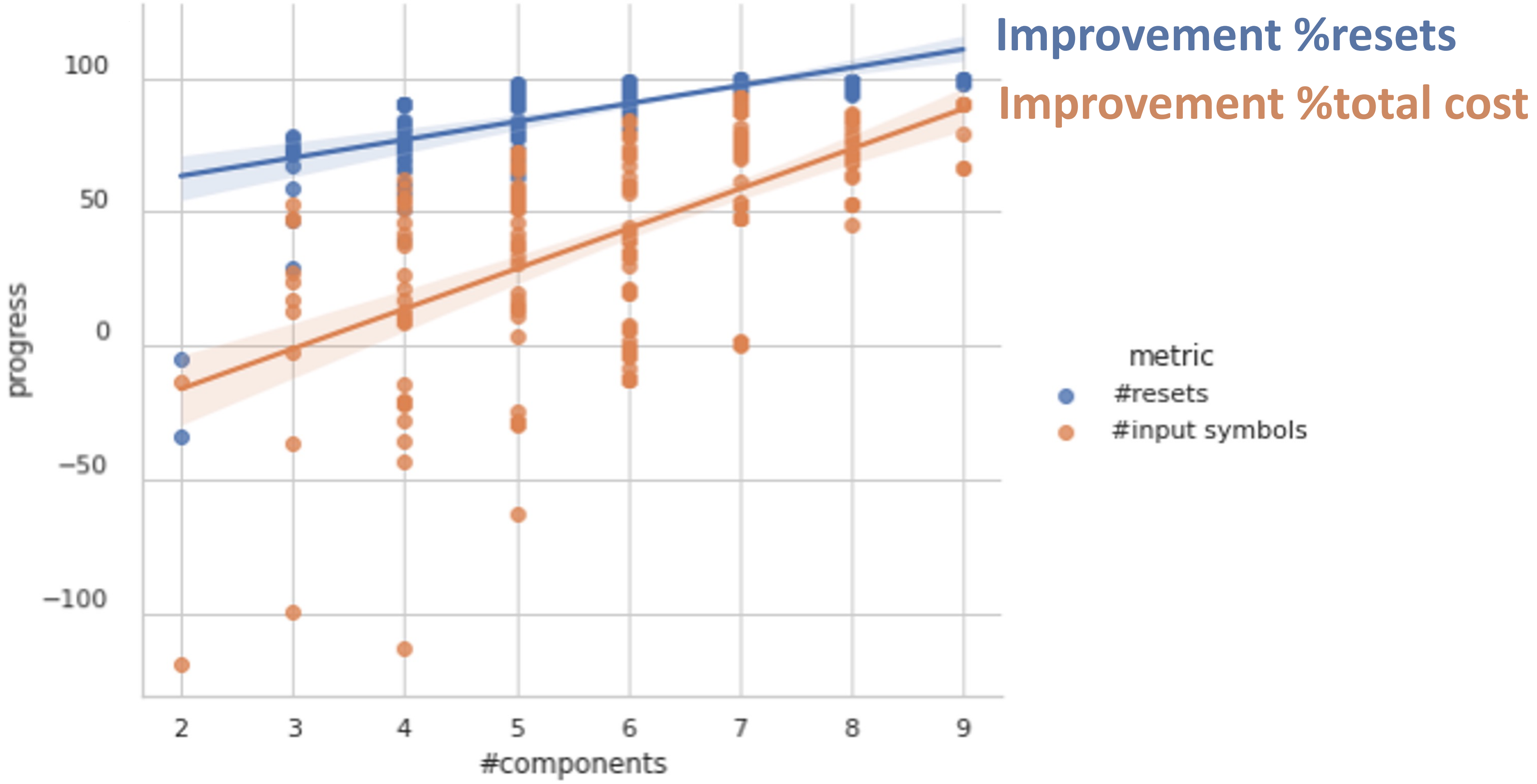
- Body Comfort System
  - An automotive software product line of Volkswagen Golf model.
  - Contains 27 components

- Benchmarks
  - 100 FSMs
  - 2 to 9 components
  - 0 to 3840 states, average: 1278

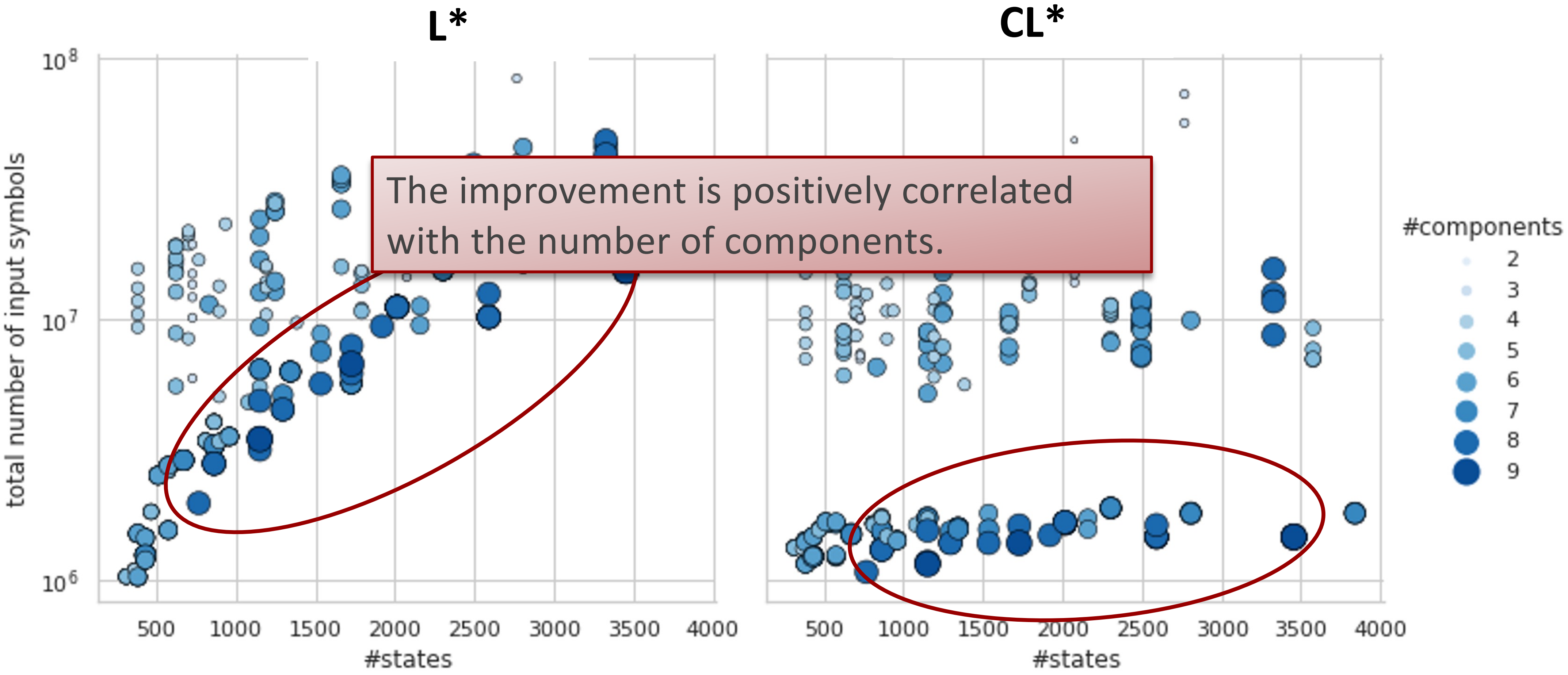
Experiments: Performance



Experiments: Improvement



Experiments: Effect of Parallelism



[Labbaf, Groote, Hojjat, Mousavi, Compositional Learning for Interleaving Parallel Automata. FOSSACS 2023]

Active Automata Learning

Adaptive Learning

Product-Line Learning

Compositional Learning

Thank you very much!

Mohammad Mousavi
mohammad.mousavi@kcl.ac.uk