

# Learning finite state machine models of evolving systems:

*From evolution over time to variability in space*

Carlos Diego Nascimento Damasceno

Advisors: Adenilso Simão (University of Sao Paulo) & Mohammad Mousavi (University of Leicester)



# Agenda

1. Introduction
2. Research Problem
3. Research Objectives
  - Learning to Reuse
  - Learning from Difference
  - Learning by Sampling
4. Final Remarks and Future Work

# Introduction

## Software maintenance [IEEE, 2006]

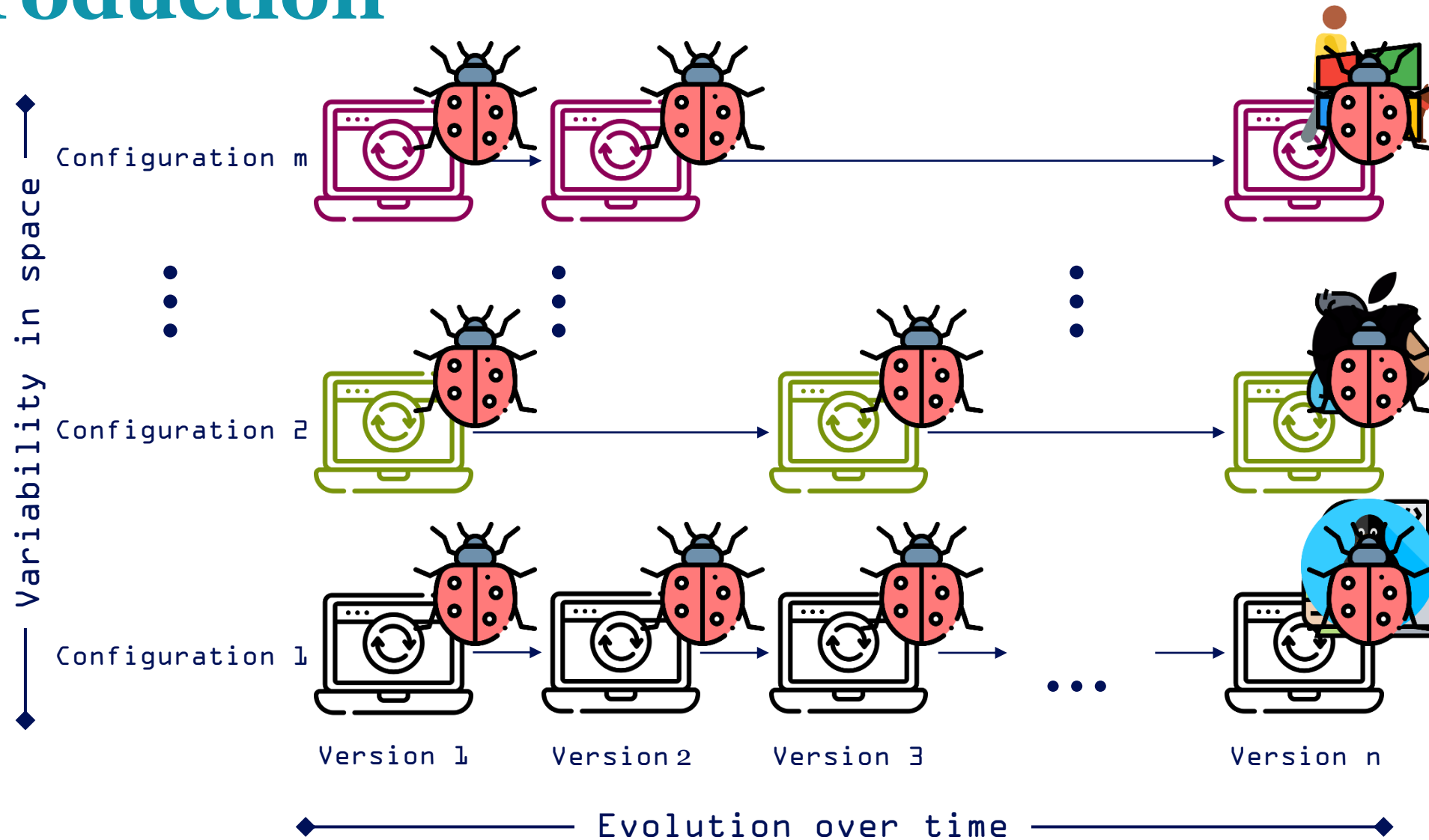
*“... modifications after delivery to correct faults, to improve non-functional attributes ...”*



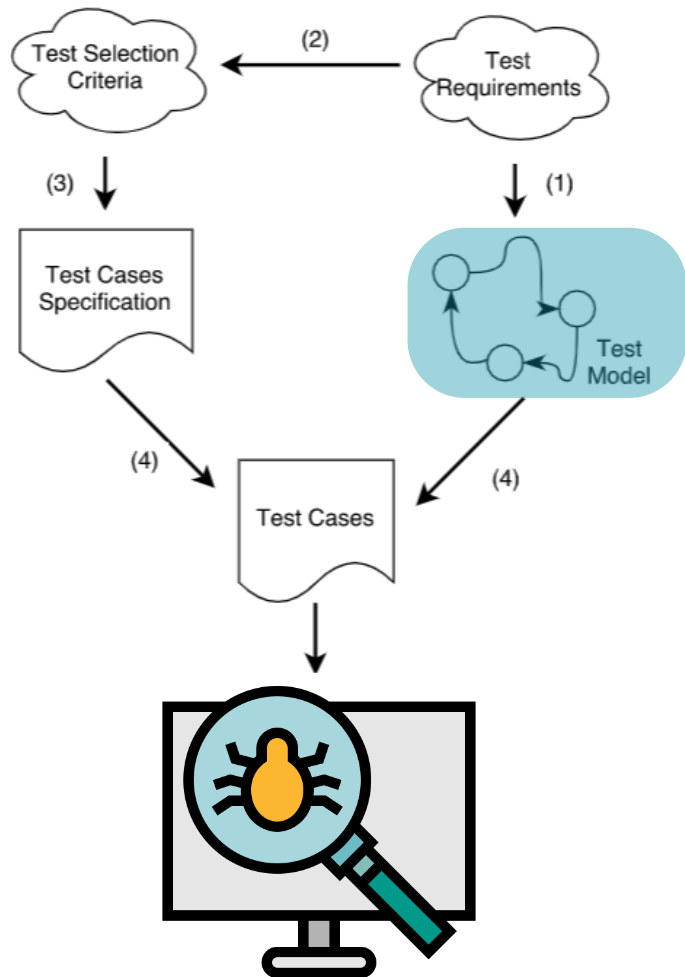
## Software evolution [Lehman, 1979]

*“... programs must be modified because they operate in or address problems in the real world ...”*

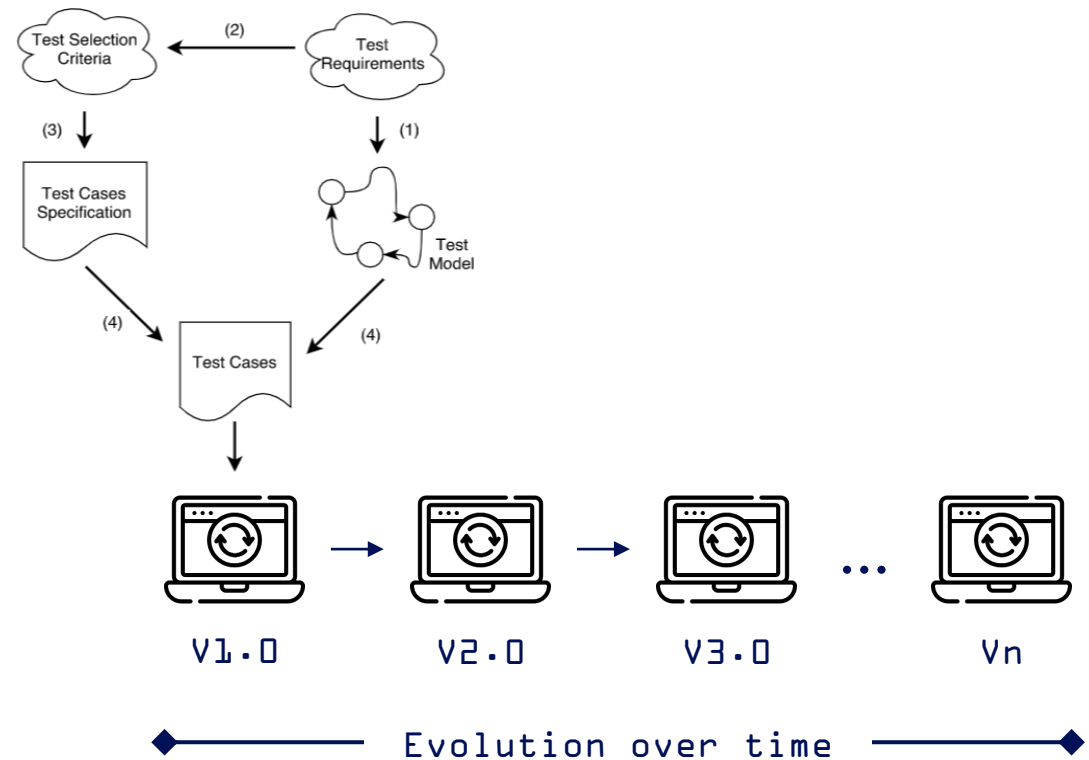
# Introduction



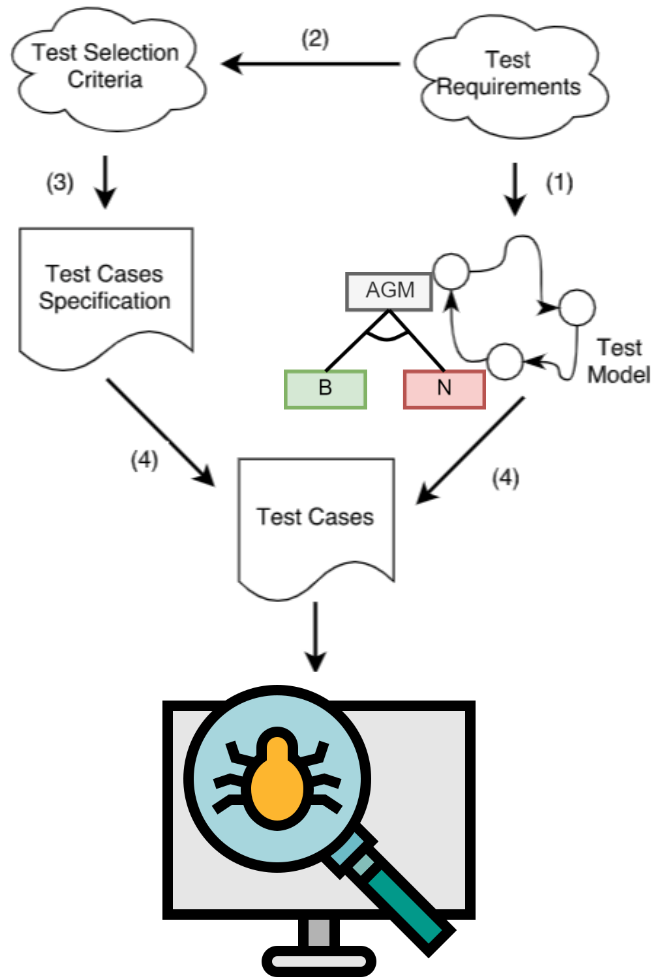
# Introduction



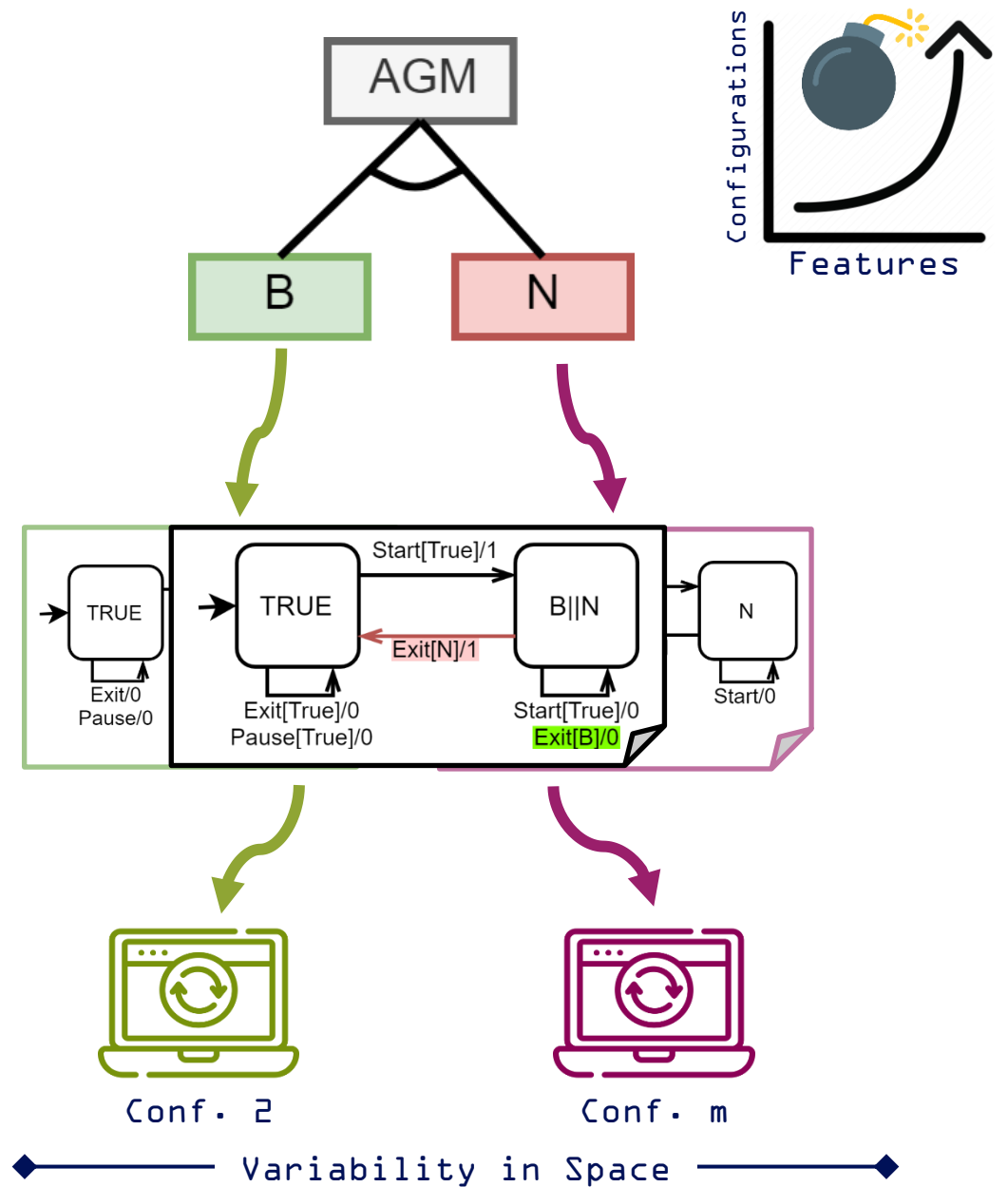
Model-Based Testing (MBT)



# Introduction



Model-Based Testing (MBT)

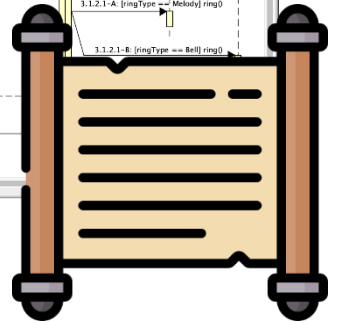
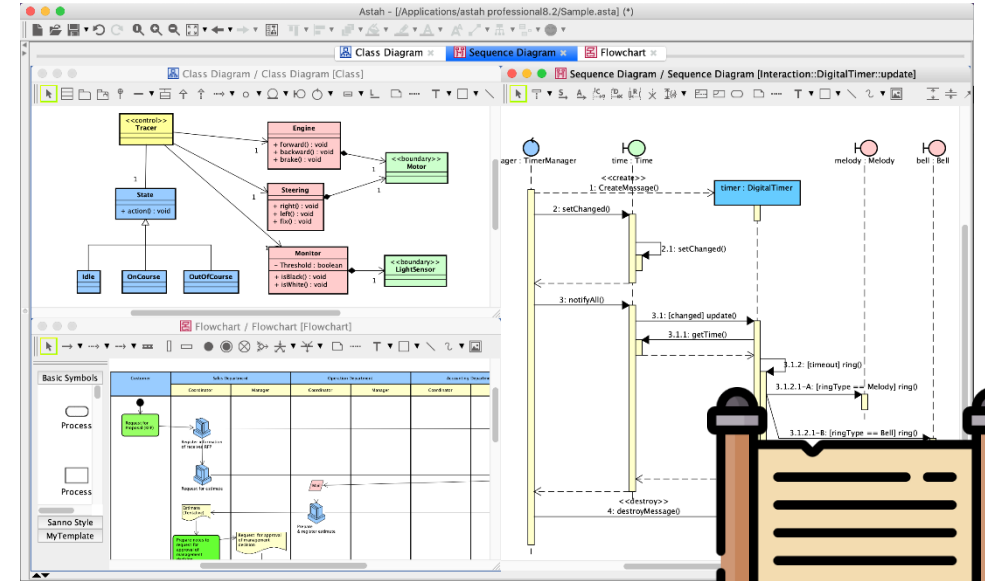
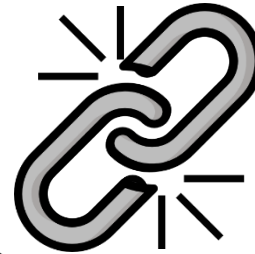


# Introduction

```
com.example.android.wiktionary
├── BuildConfig
├── ExtendedWikiHelper
├── LookupActivity
├── Manifest
├── R
├── SimpleWikiHelper
└── WordWidget

/**
 * Before jumping into background thread, start sliding in the
 * {@link ProgressBar}. We'll only show it once the animation finishes.
 */
@Override
protected void onPreExecute() {
    mTitleBar.startAnimation(mSlideIn);
}

/**
 * Perform the background query using {@link ExtendedWikiHelper}, which
 * may return an error message as the result.
 */
@Override
protected String doInBackground(String... args) {
    // ...
}
```



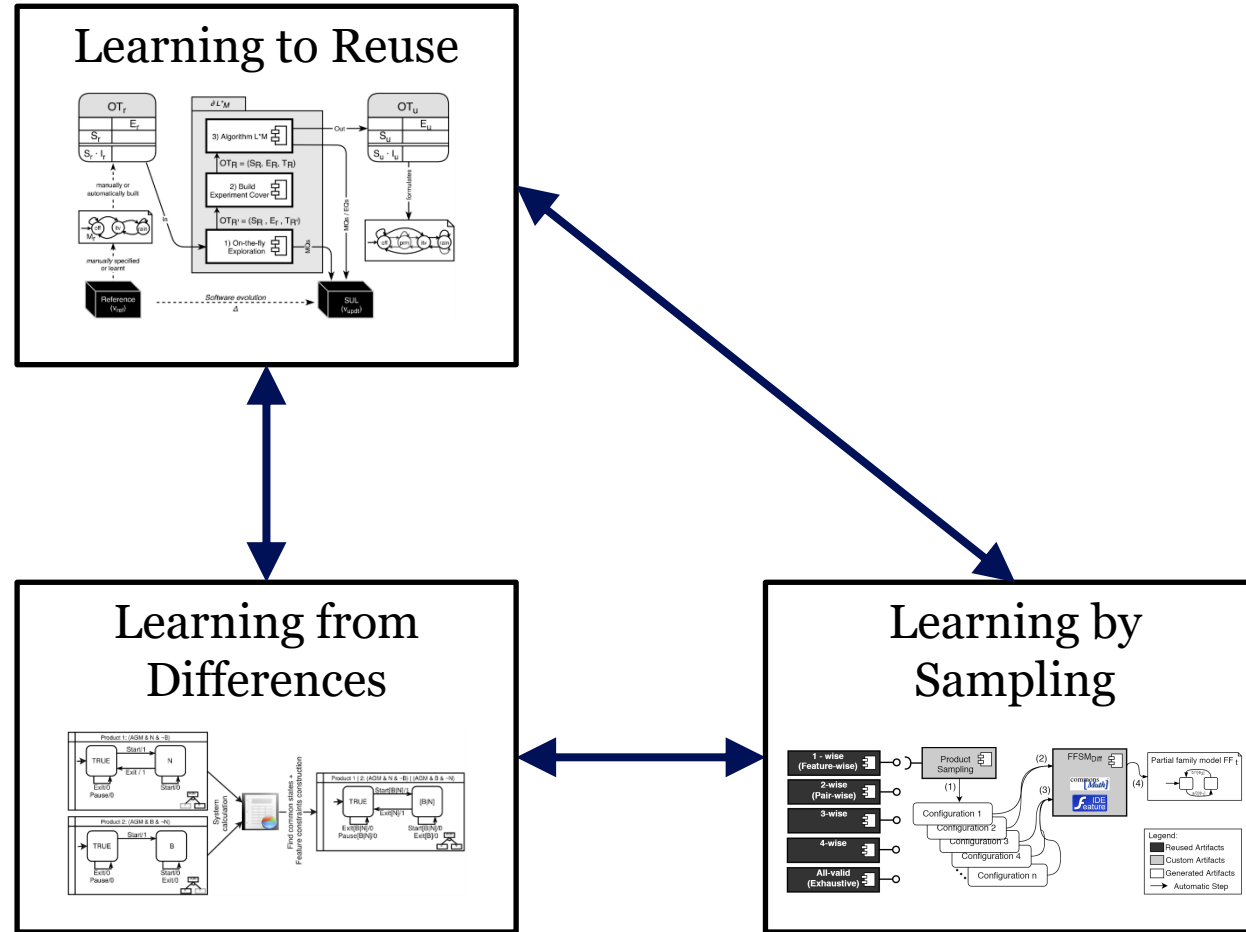
*Source code and models should be maintained and evolve together!*

# Research Problem

*How can we efficiently and effectively learn finite state machines specifying the behavior of an evolving system?*



# Research Objectives



# Learning to Reuse

Adaptive Model Learning for Evolving Systems



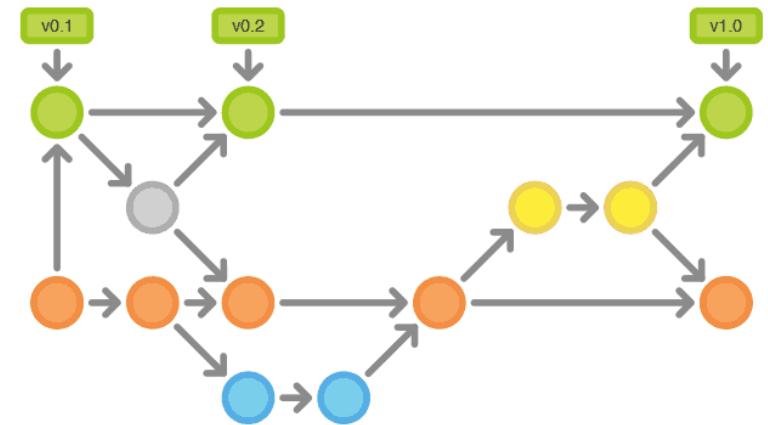
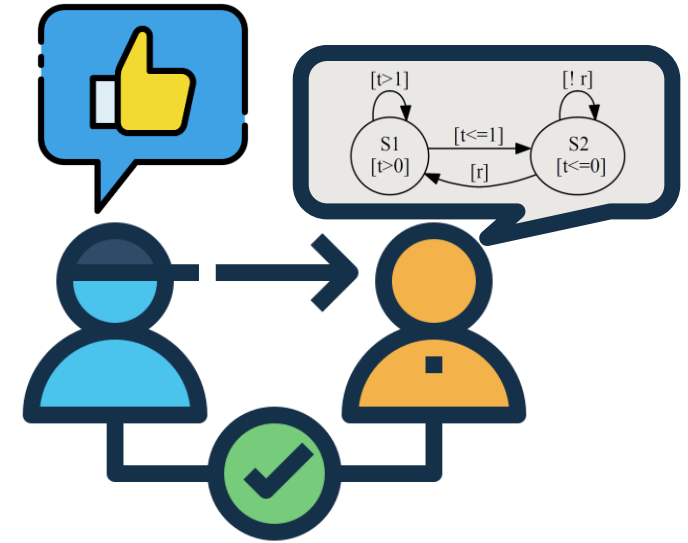
# Context (Learning to Reuse)

## 1. Software analysis is a *model-based* activity

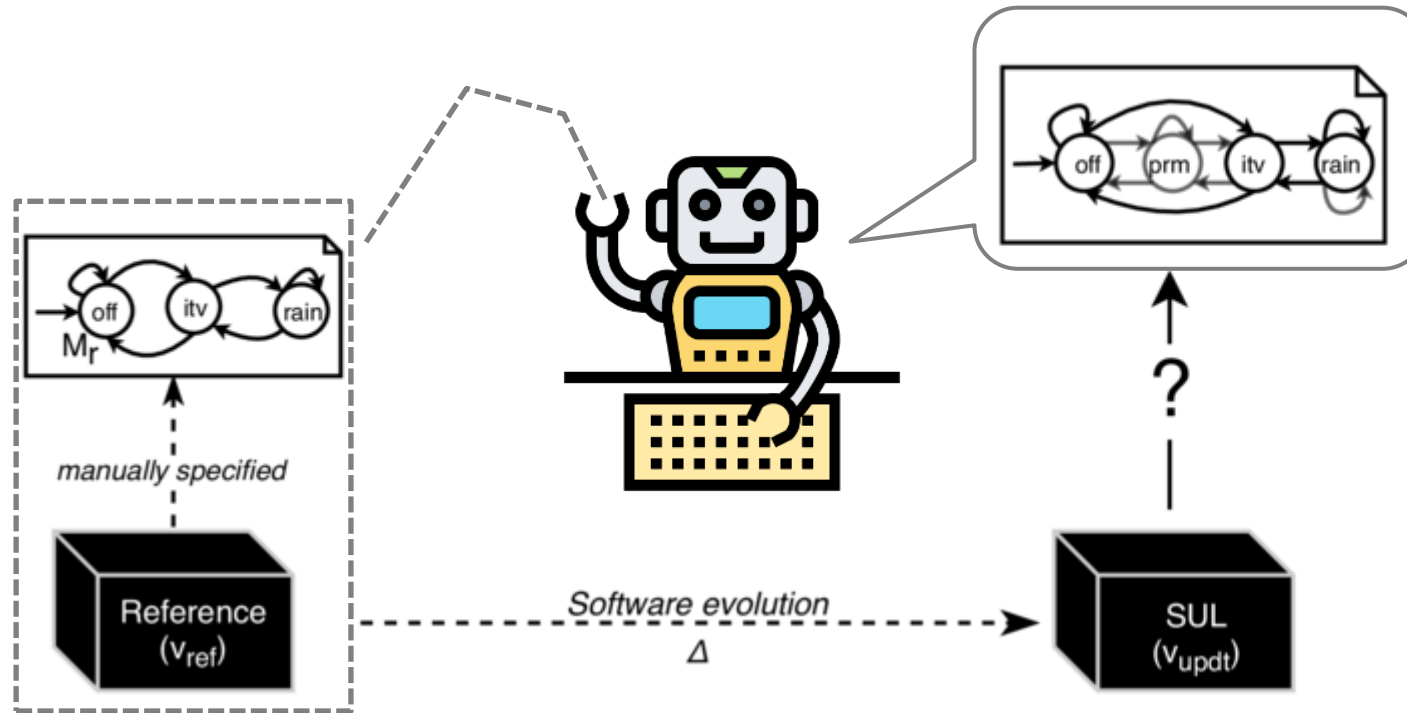
- Models stuck to engineers' minds
- Formally denoted as explicit models

## 2. Software undergoes changes along the life-cycle

- Evolution over-time (e.g., update, upgrade)
- Models may become outdated

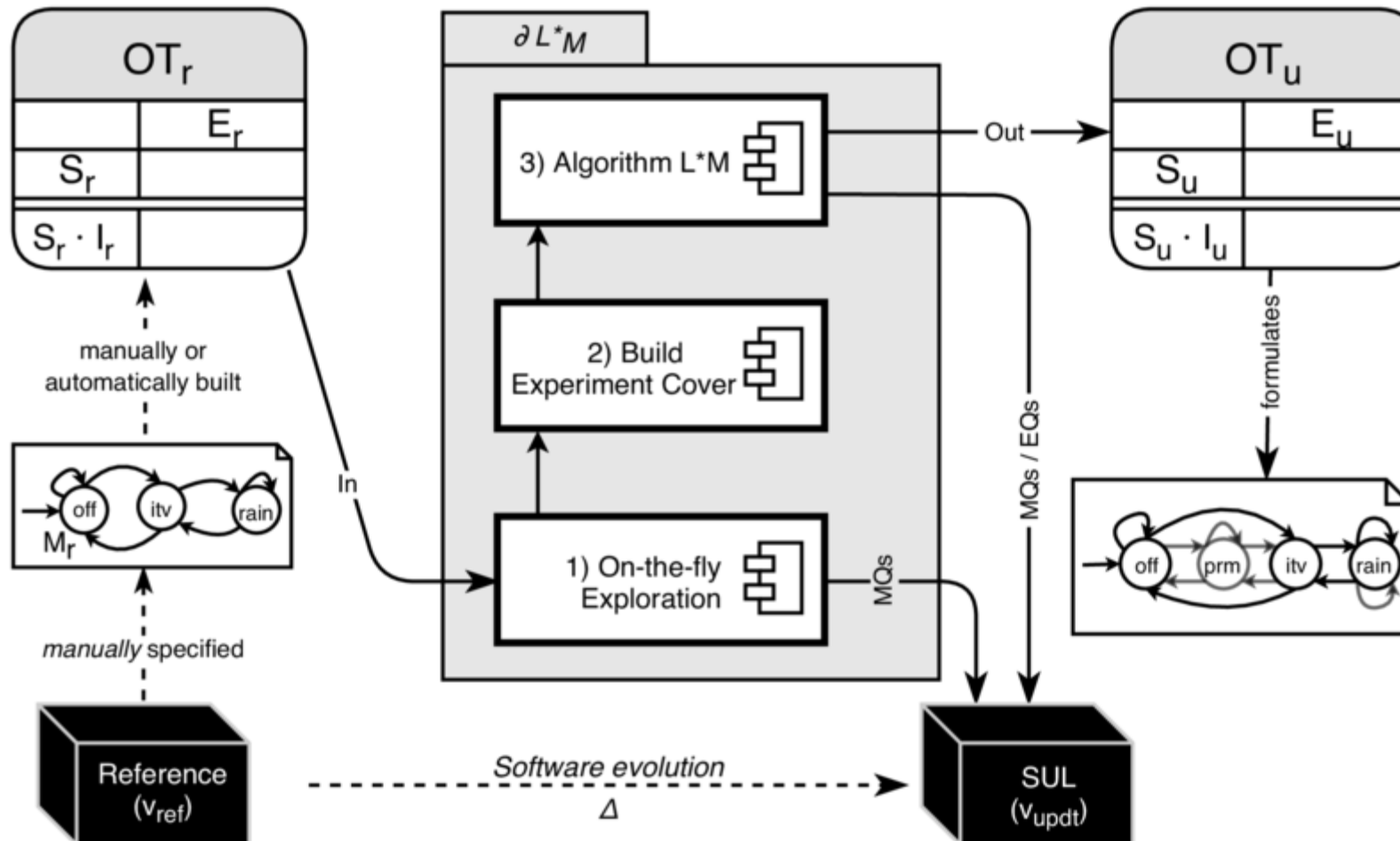


# Research Problem (Learning to Reuse)



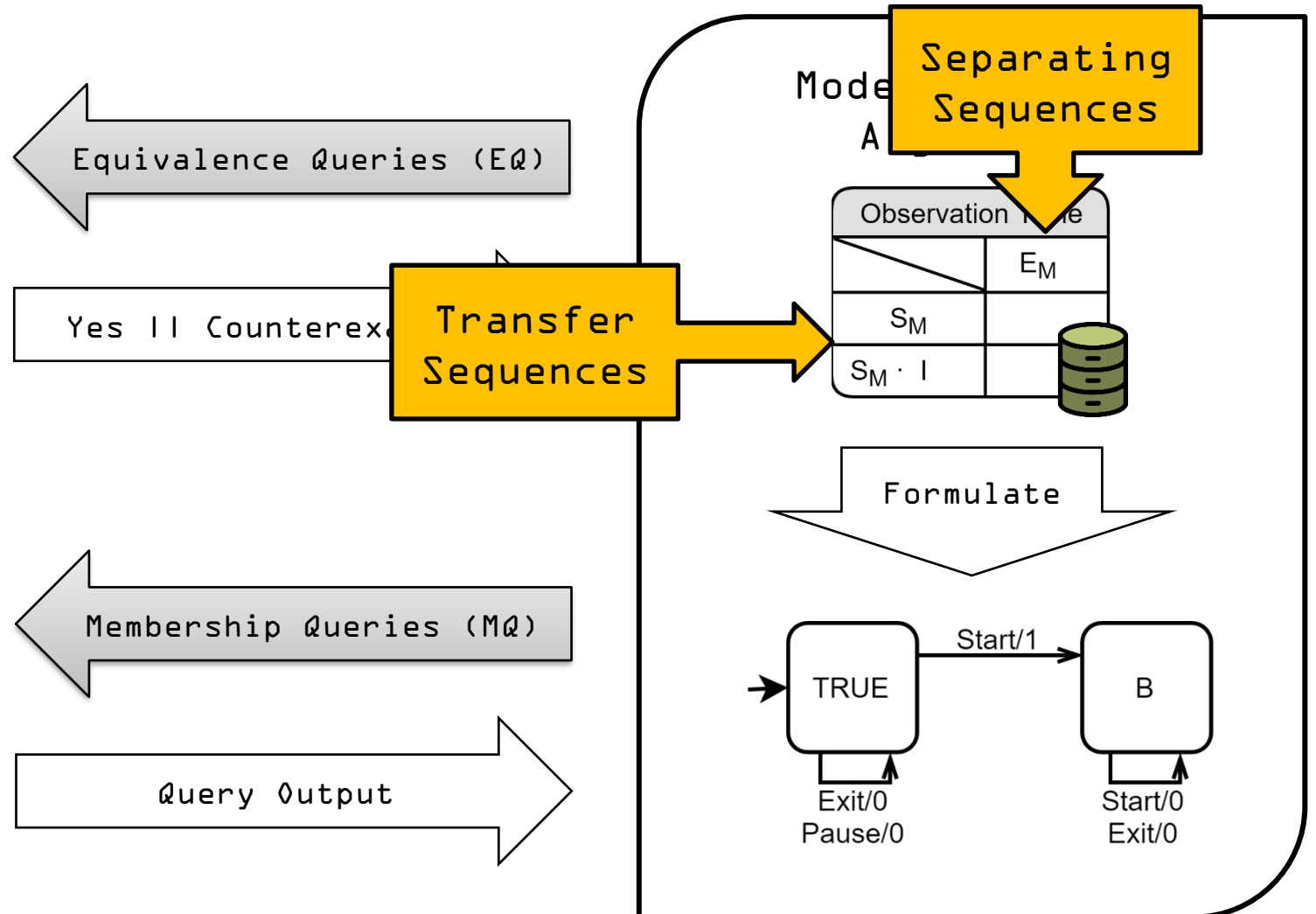
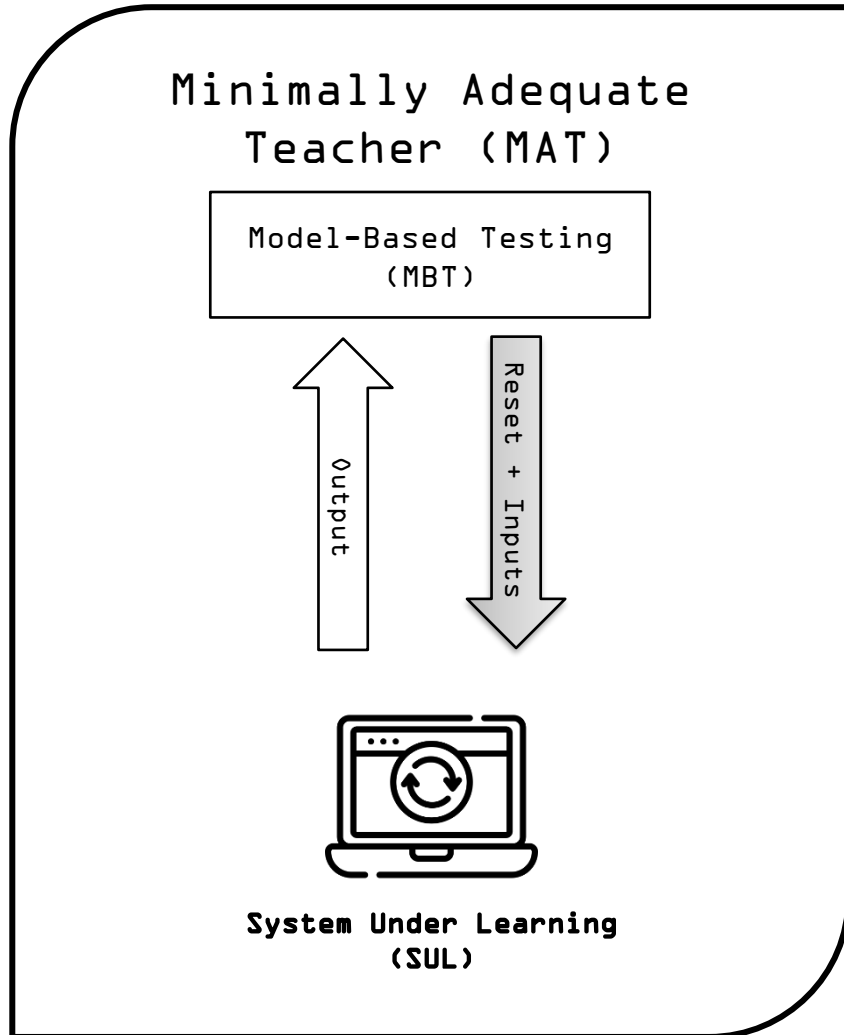
How can we **efficiently** construct **behavioral models** from **evolving systems**?

# Contribution (Learning to Reuse)



An adaptive algorithm that is **more efficient** than the state-of-the-art for **learning behavioral models** from **evolving systems**

# Model Learning



# System Under Learning

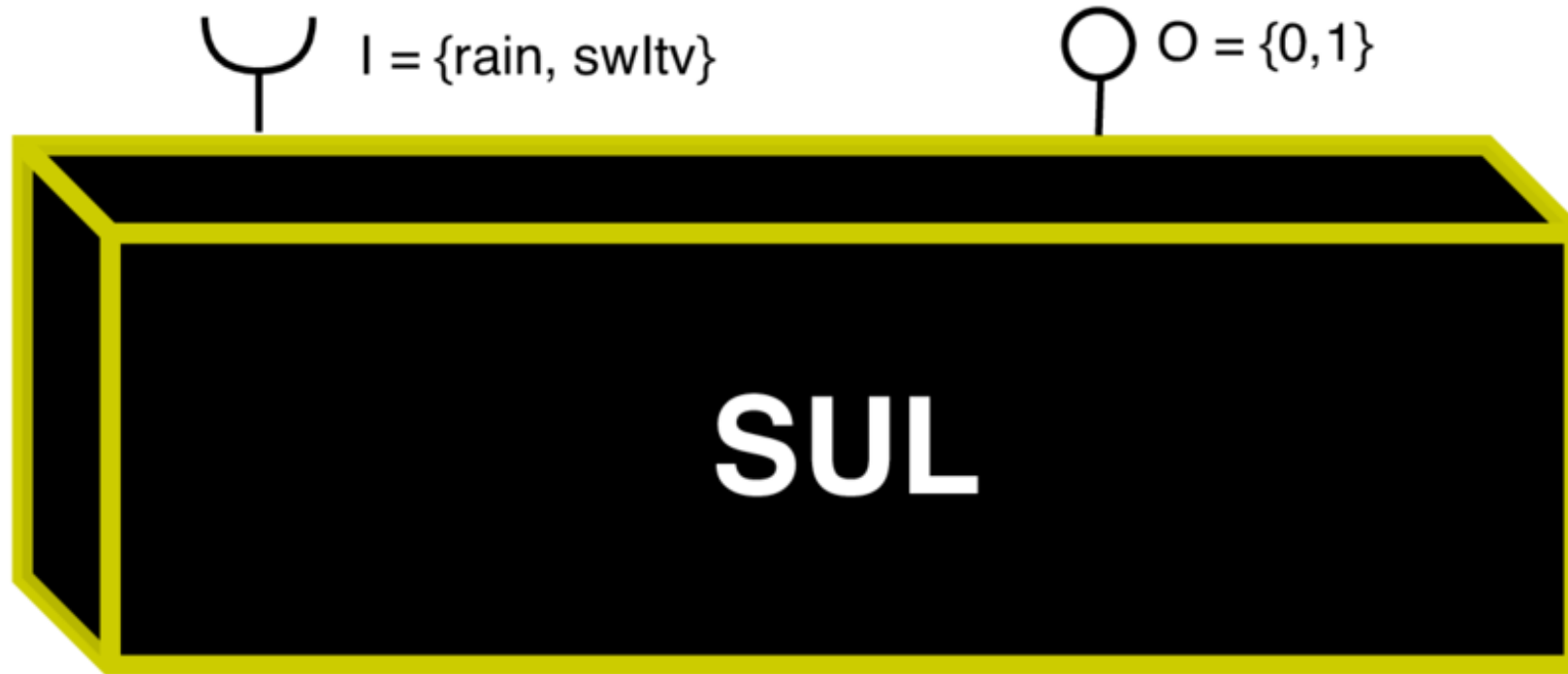


Figure: Windscreen wiper supporting intervaled and fast wiping

# Model Learning

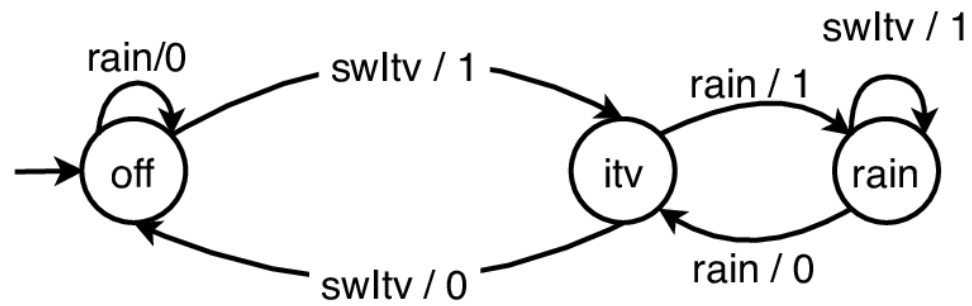


Figure: Final Hypothesis

		rain	swl tv	rain · rain
<i>S</i>	$\epsilon$	0	1	0 · 0
	swl tv	1	0	1 · 0
	swl tv · rain	0	1	0 · 1
<i>S · I</i>	rain	0	1	0 · 0
	swl tv · swl tv	0	1	0 · 0
	swl tv · rain · rain	1	0	1 · 0
	swl tv · rain · swl tv	0	1	0 · 1

Table: Final OT

EQ = Yes



What if the SUL evolves?

# Model Learning for Evolving Systems

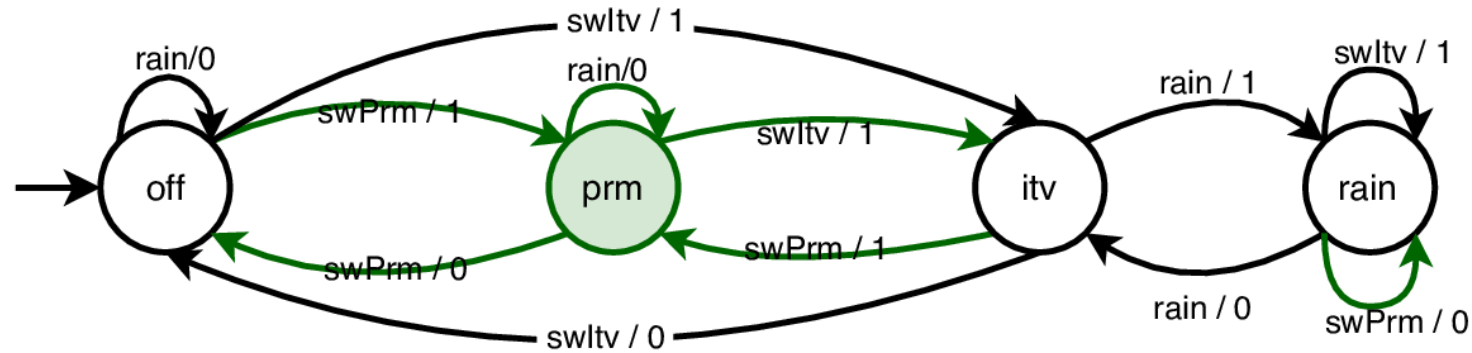
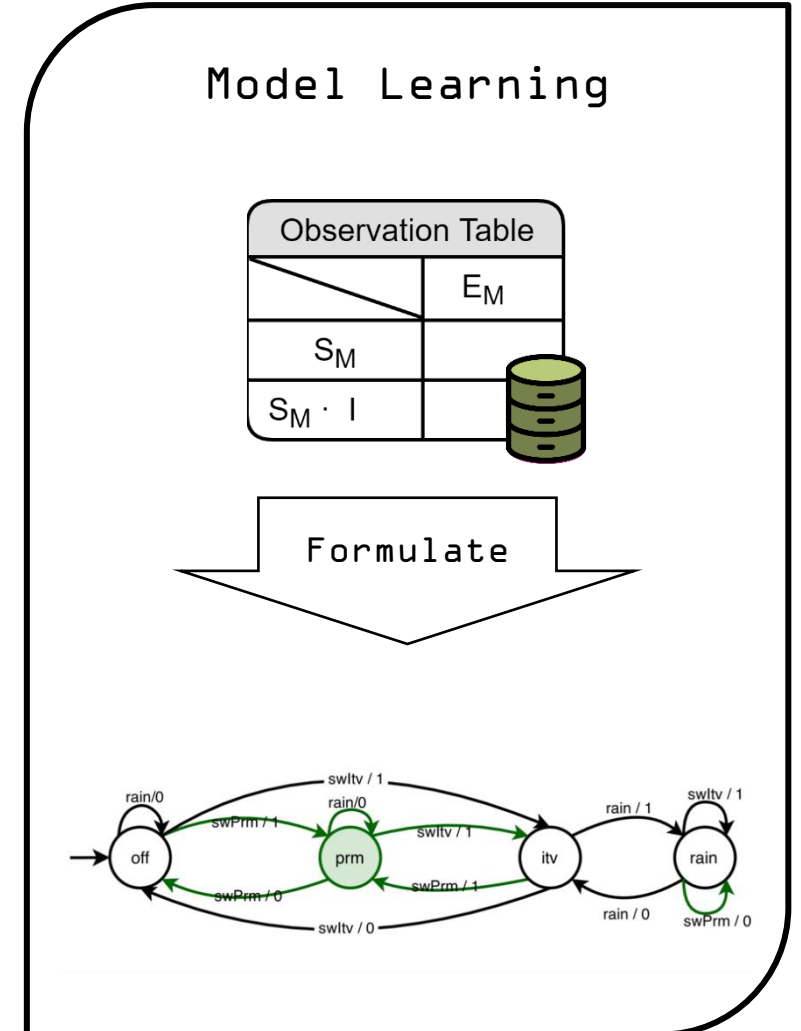
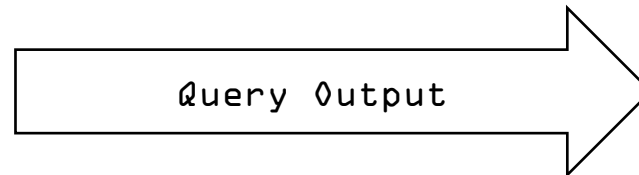
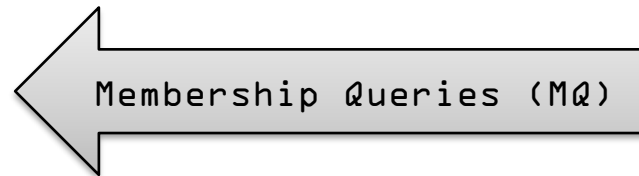
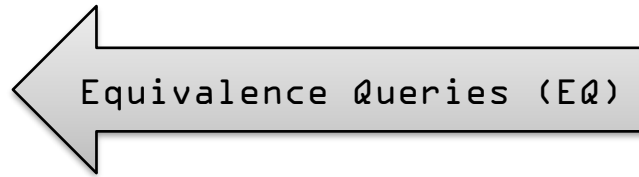
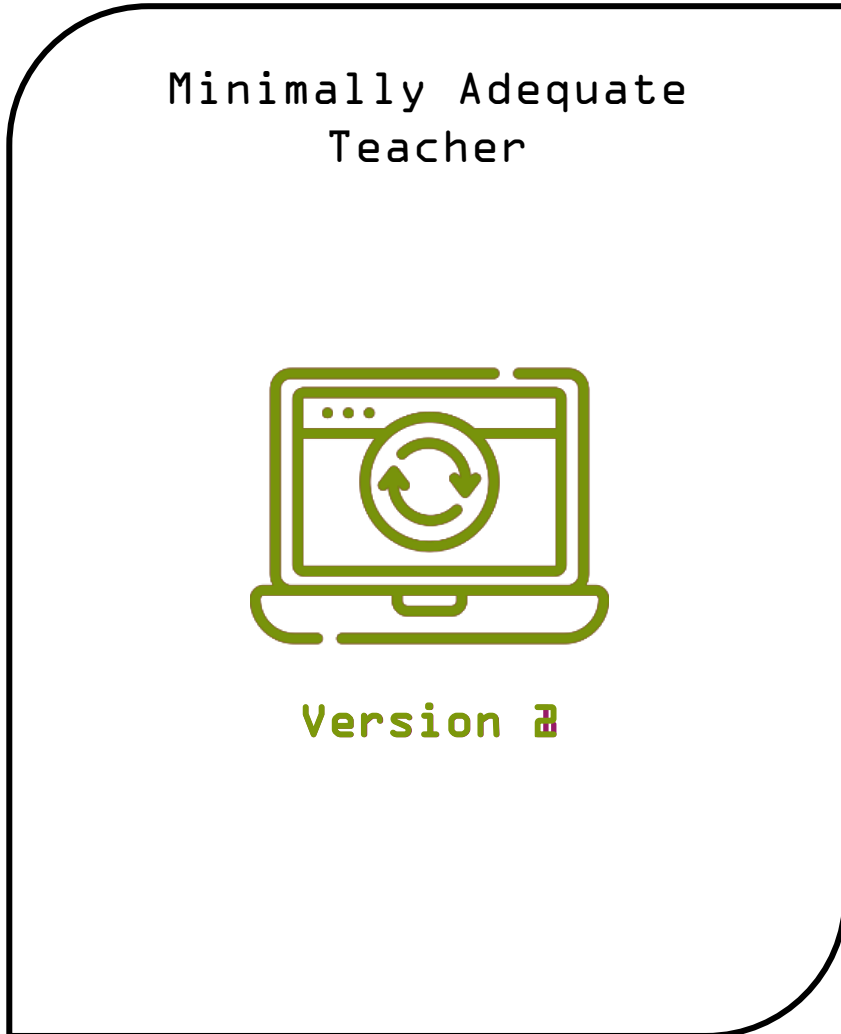


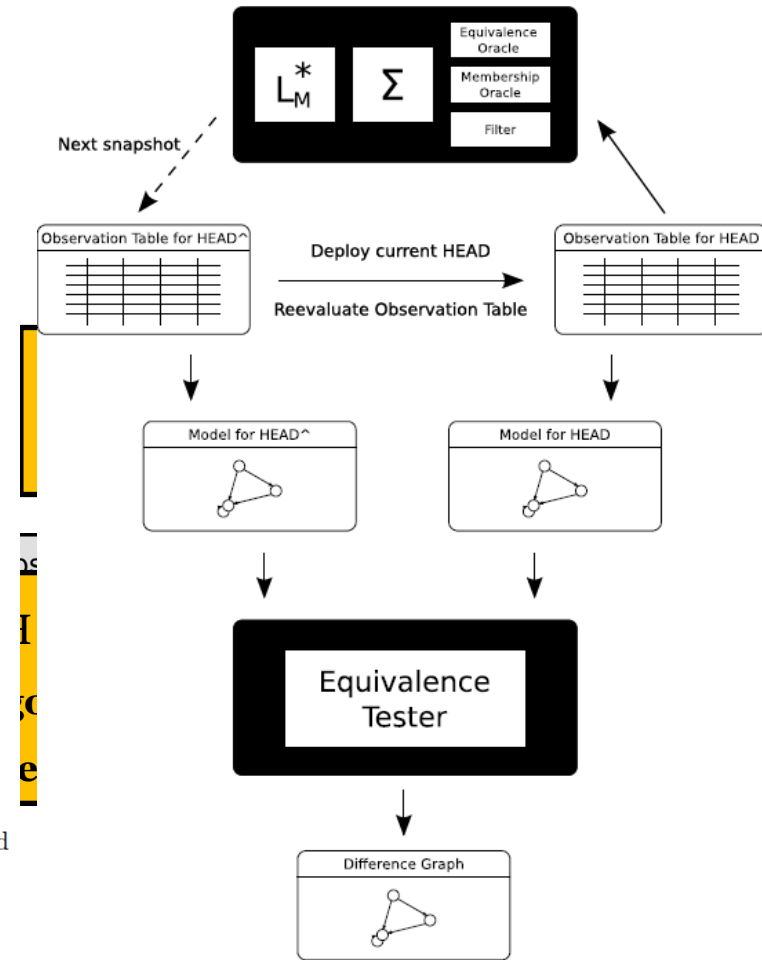
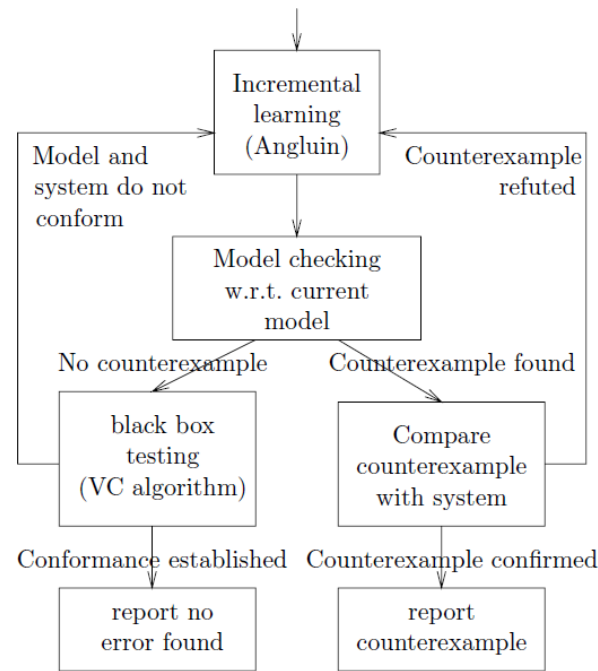
Figure: Windscreen wiper supporting intervaled and fast wiping + **permanent movement**

# Model Learning for Evolving Systems

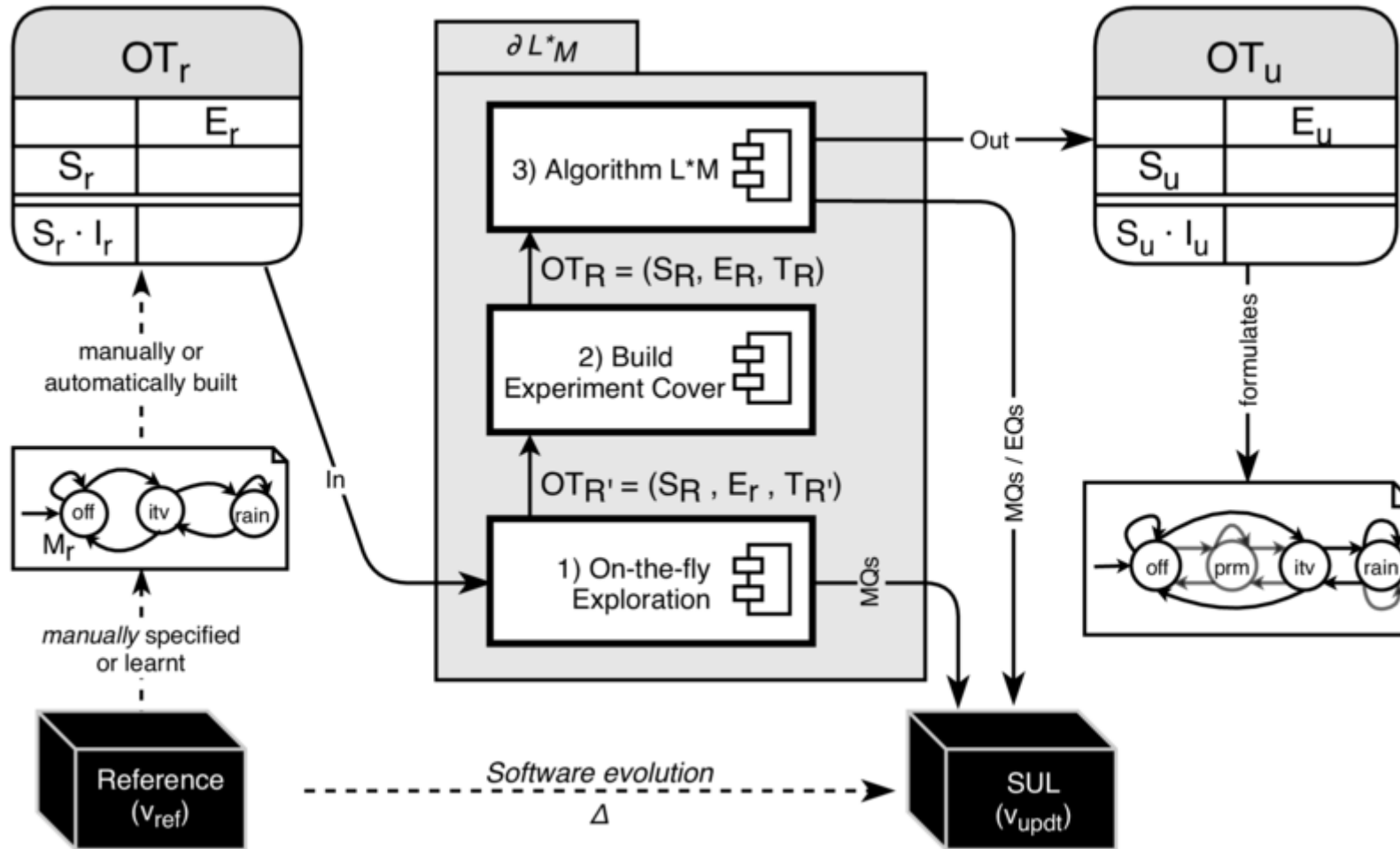


# Adaptive Model Learning

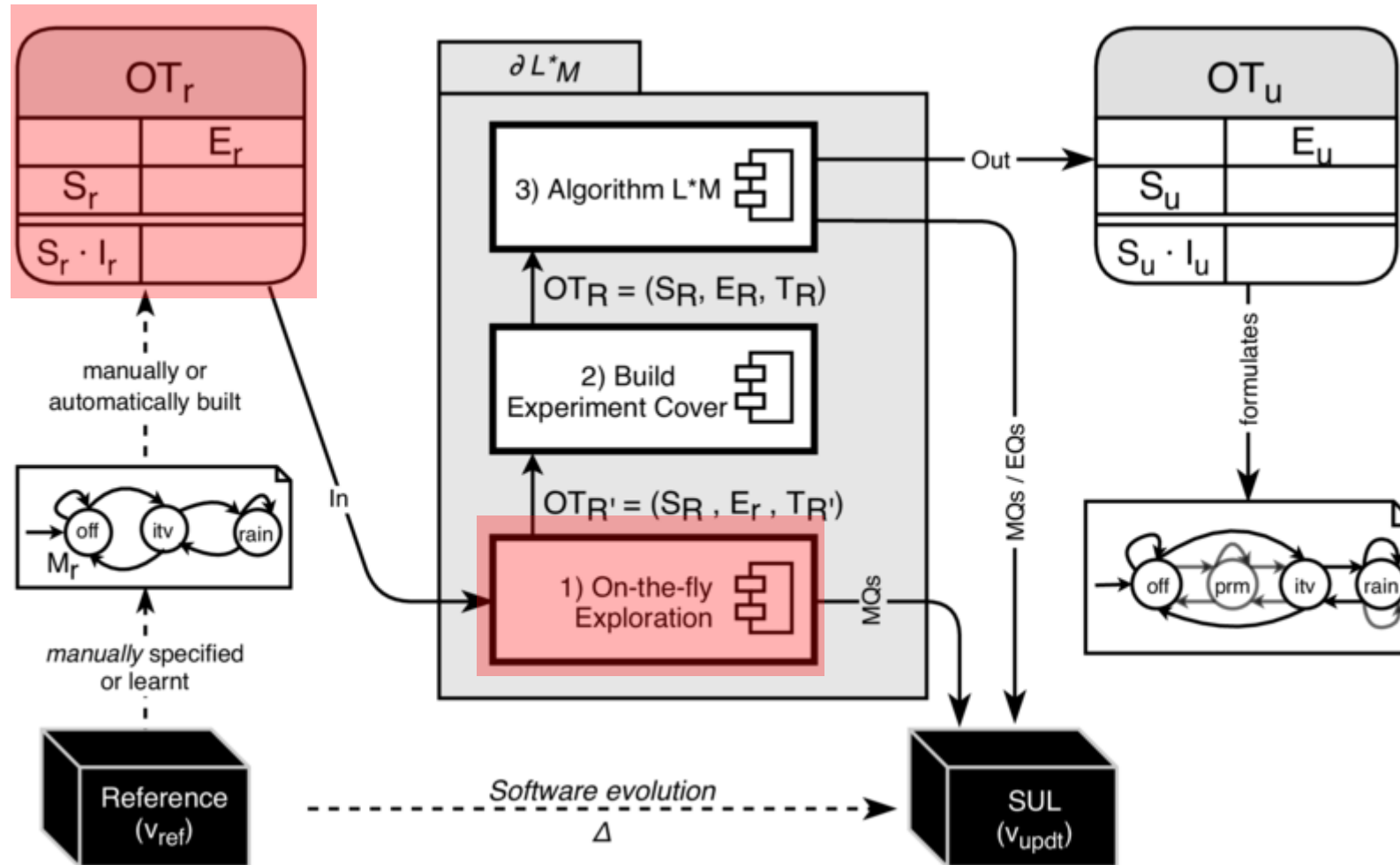
- **What:** Variant of model learning
- **How:** Reuse transfer/separating sequences from existing models
- **Why:** Speed up model learning
  - Find states maintained in newer versions
  - Reduce the time for model checking



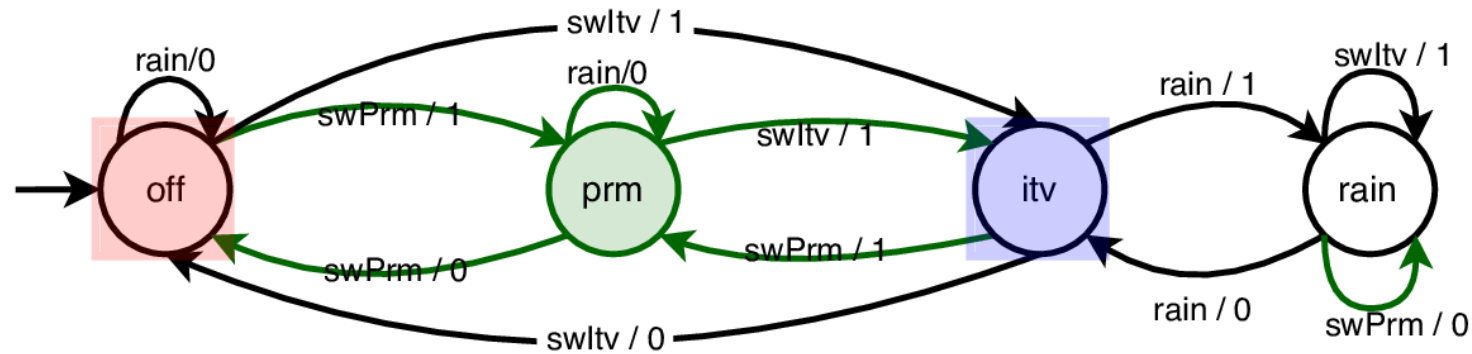
# Partial-Dynamic $L^*_M$ algorithm



# 1) On-the-fly exploration of the reused OT



# 1) On-the-fly exploration of the reused OT



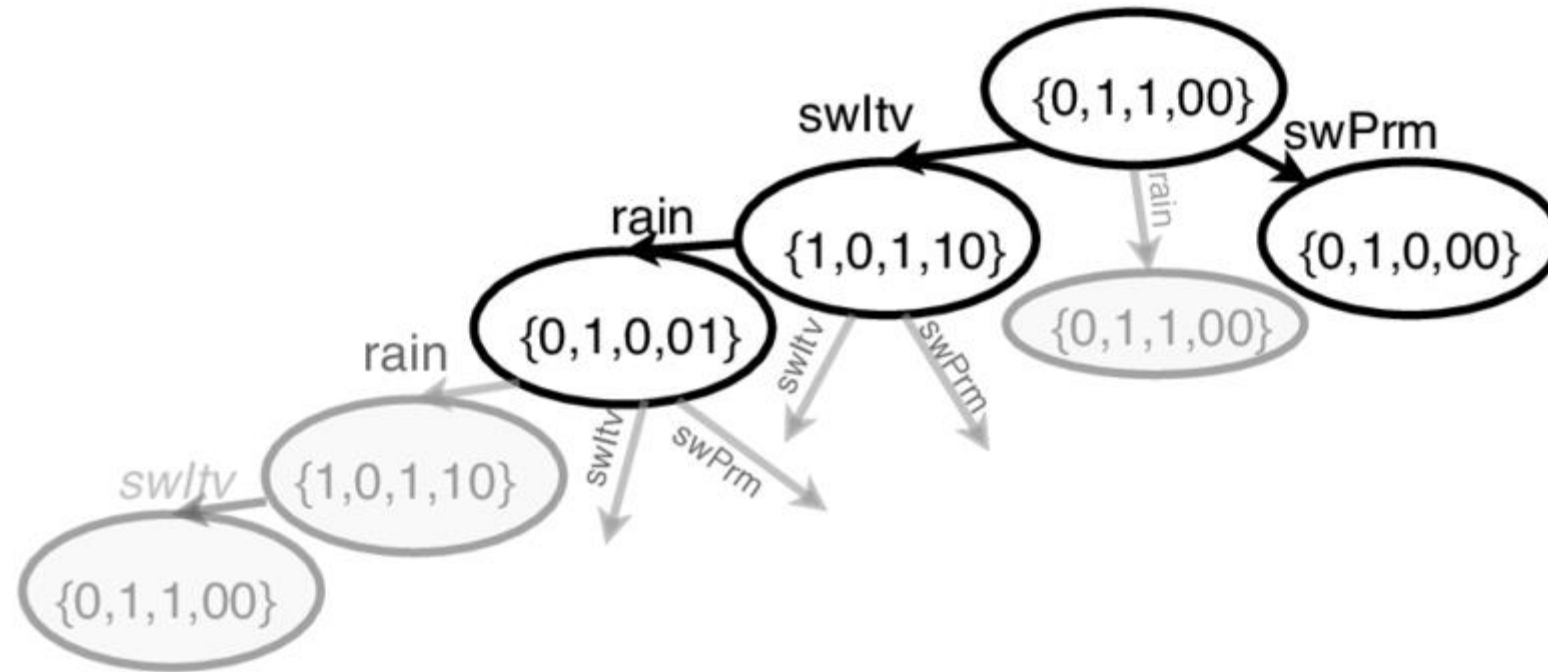
Let the sets of reused prefixes and suffixes be

$$S_r = \{ \epsilon, \text{swltv}, \text{swltv} \cdot \text{rain}, \text{swltv} \cdot \text{rain} \cdot \text{rain}, \text{swltv} \cdot \text{rain} \cdot \text{rain} \cdot \text{swltv}, \text{rain} \}$$

$$E_r = \{ \text{rain}, \text{swltv}, \text{swPrm}, \text{rain} \cdot \text{rain} \}$$

**Goal:** Find a  $S_R \subseteq S_r$  with the same state coverage capability but less prefixes

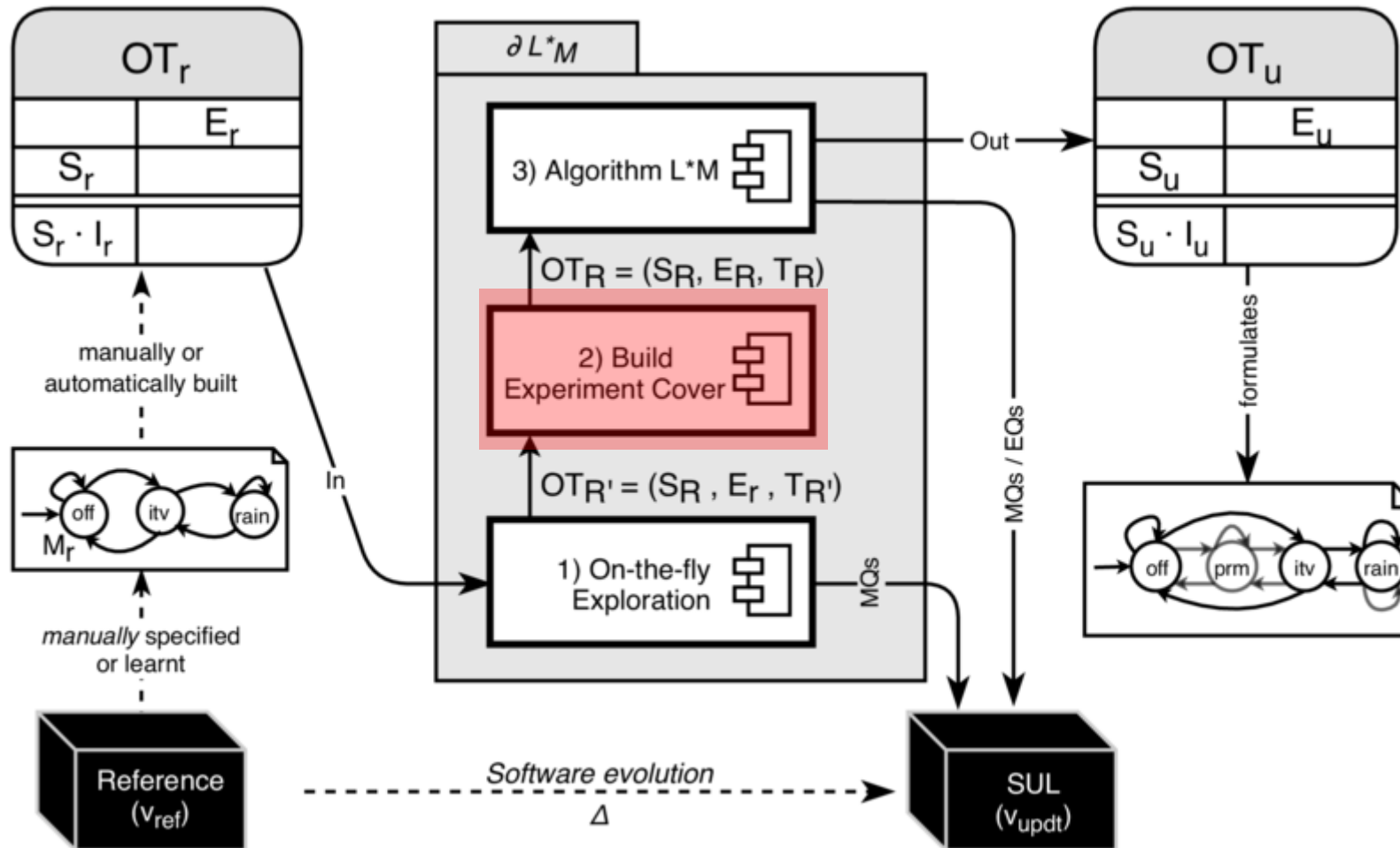
# 1) On-the-fly exploration of the reused OT



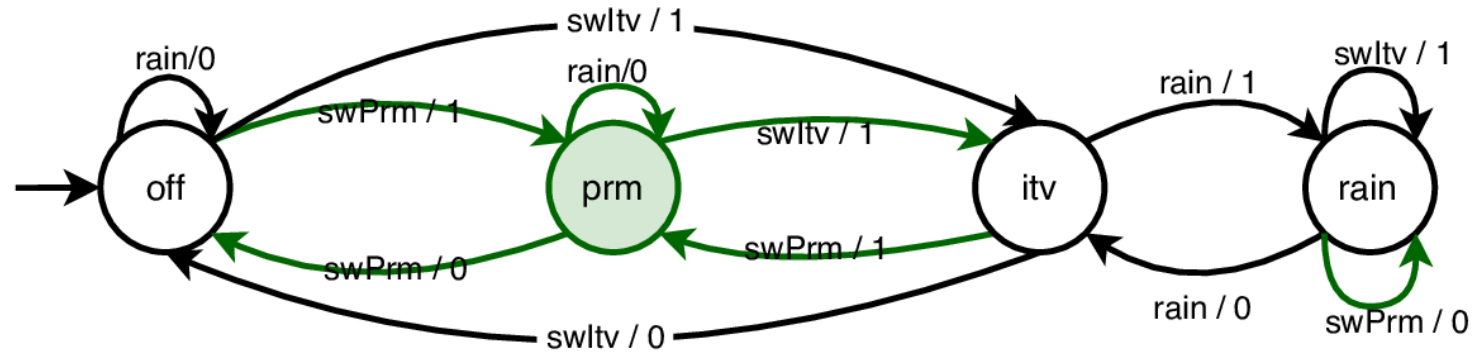
**On-the-fly exploration of the tree representation of the set of transfer sequences**



## 2) Build the experiment cover tree



## 2) Build the experiment cover tree



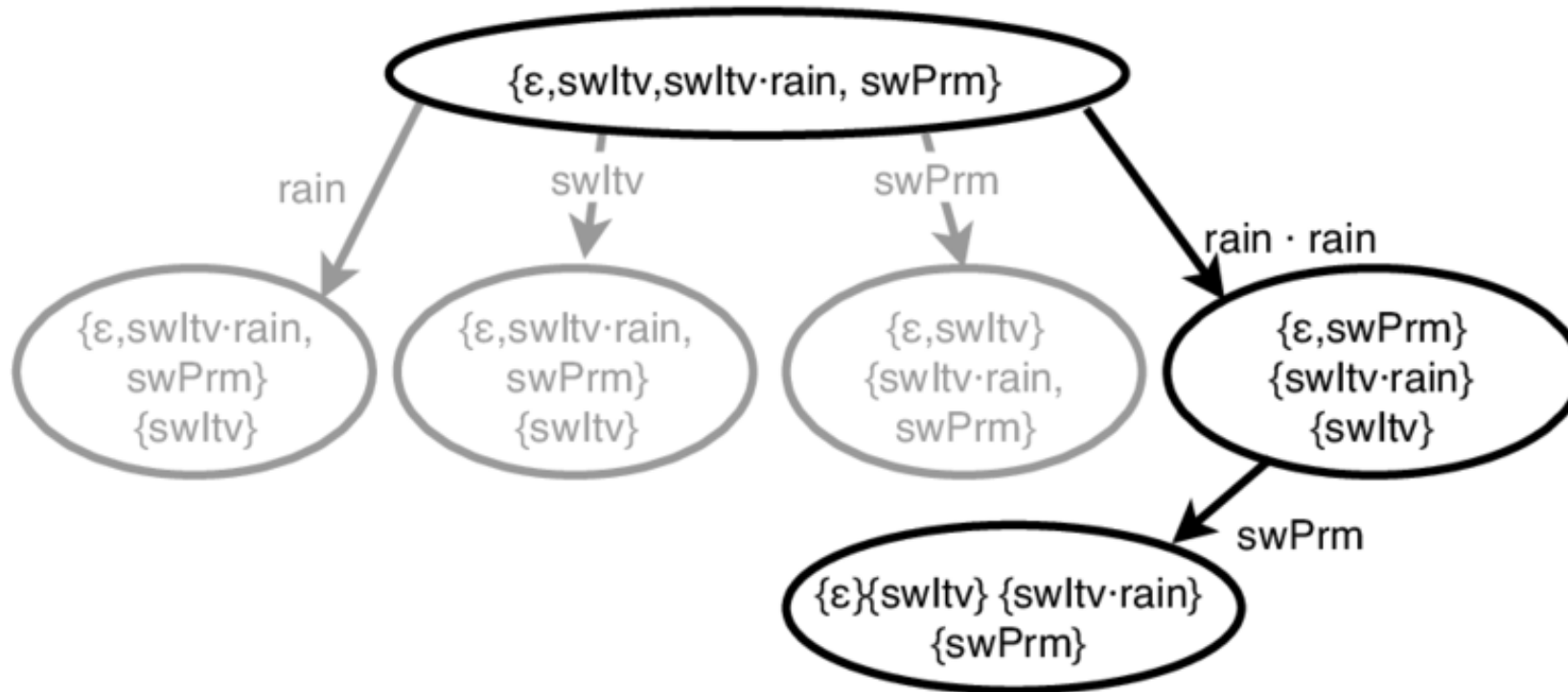
Let the sets of prefixes and suffixes be

$$S_R = \{ \epsilon, swltv, swltv \cdot rain, swPrm \}$$

$$E_r = \{ rain, swltv, swPrm, rain \cdot rain \}$$

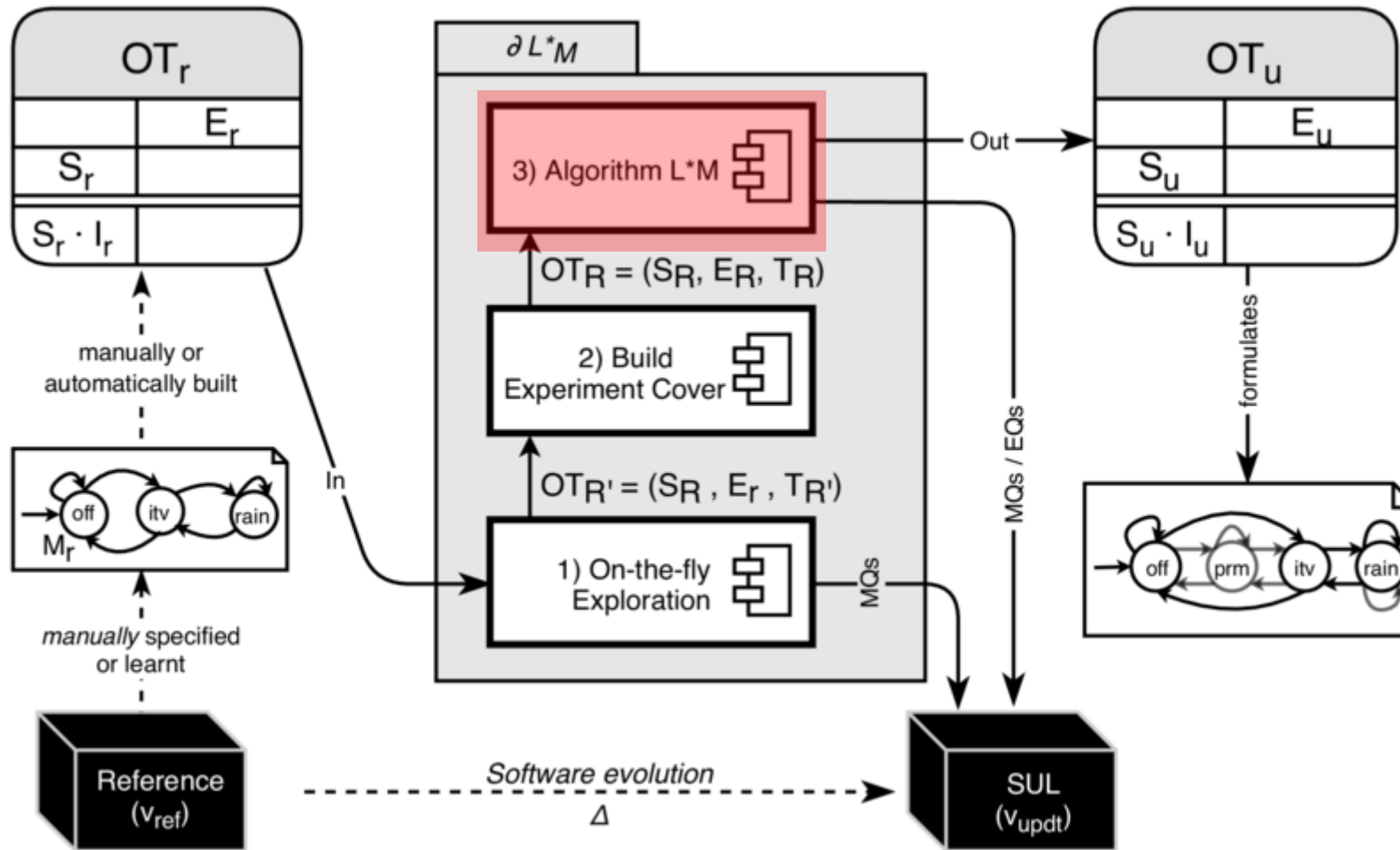
**Goal:** Find a **smaller subset**  $E_R \subseteq E_r$  of **representative** separating sequences.

## 2) Build the experiment cover tree



**Group transfer sequences into equivalence classes to find a smaller subset of separating sequences**

# 3) Starting $L^*_M$ using the outcomes of $\partial L^*_M$



# Empirical Evaluation

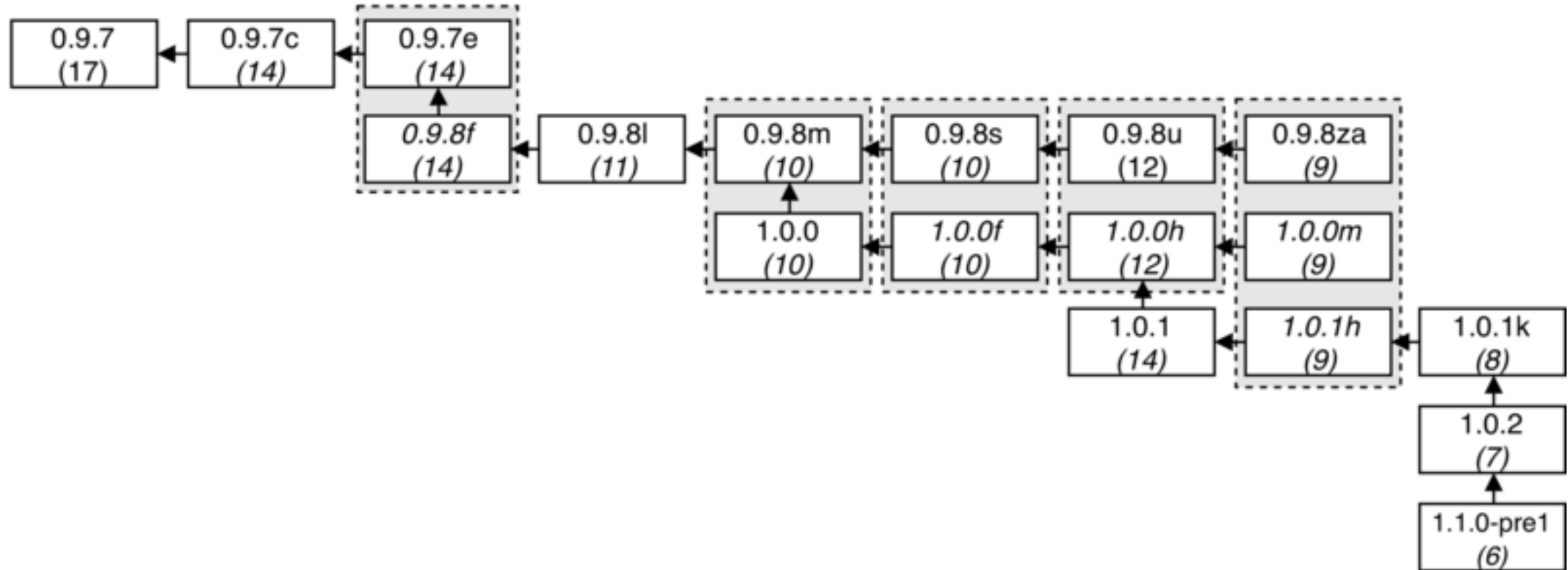
# Empirical Evaluation (Research Questions)

**RQ1)** Is our technique more efficient than the state-of-the-art of adaptive learning?

**RQ2)** Is the effectiveness of adaptive learning strongly affected by the temporal distance between versions?

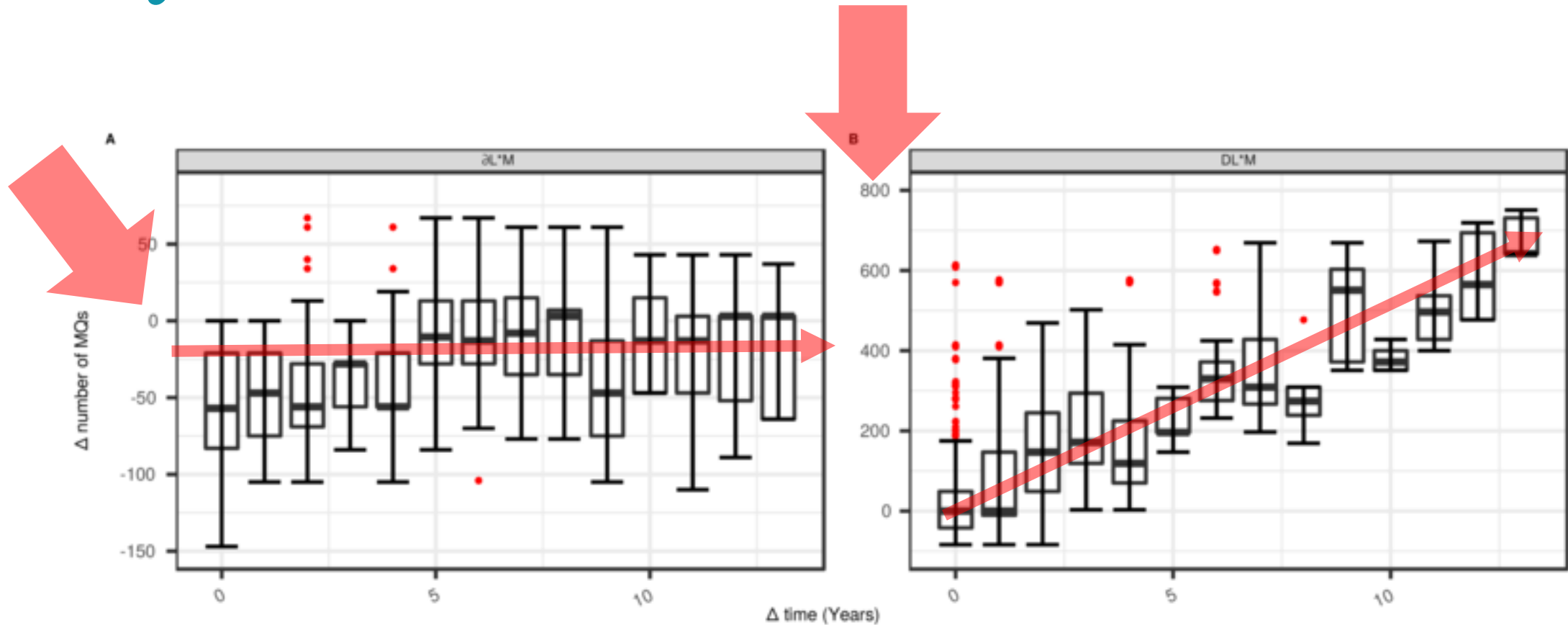


# Subject Systems



Subject systems: 18 state machines describing versions of the OpenSSL toolkit

# Analysis of Results (Average number of MQs)



The temporal distance between fewer MQs did not affect the performance of the algorithm



# Summary (Learning to Reuse)



The state-of-the-art adaptive learning algorithms...

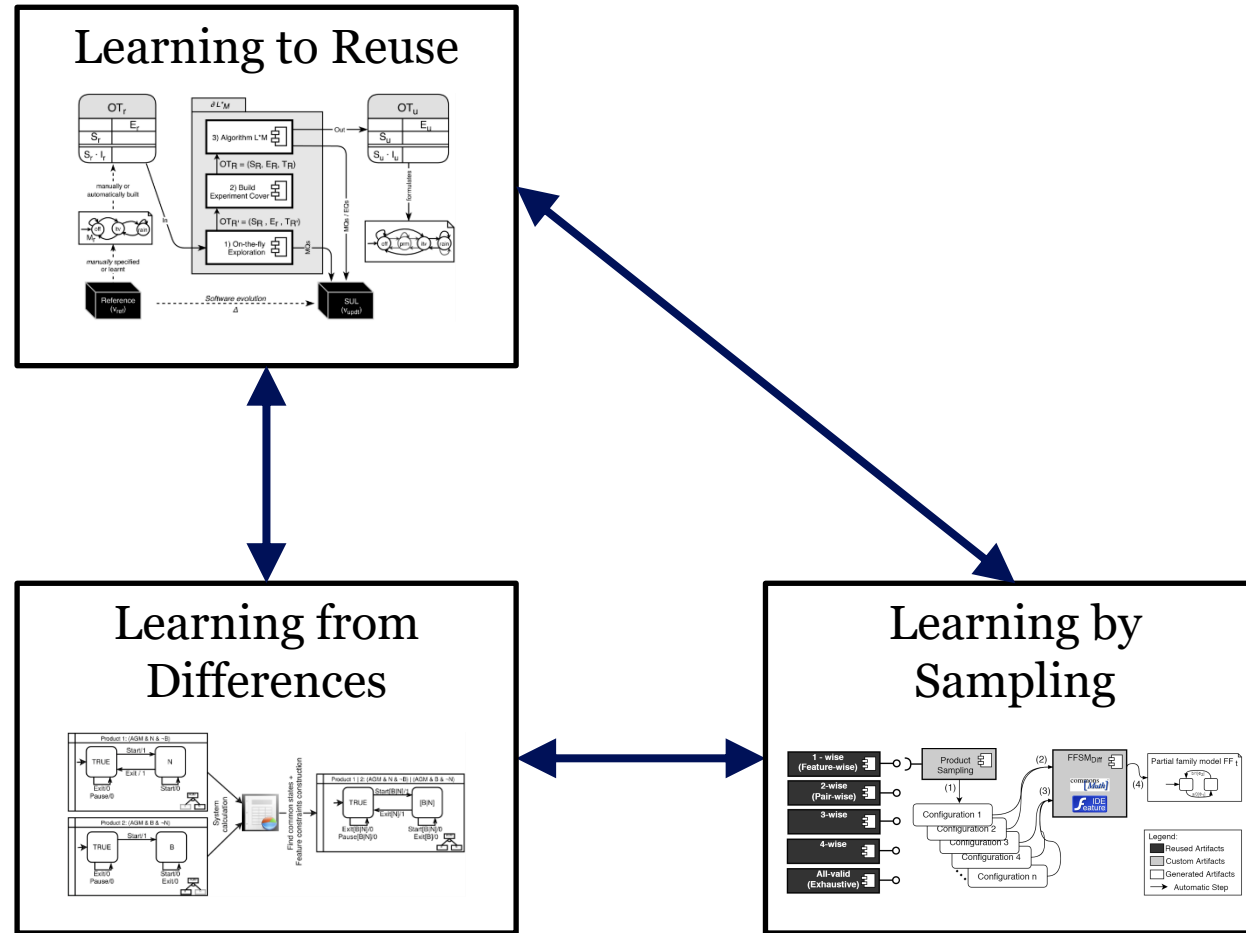
1. More sensitive to software evolution!

The  $\partial L^*_M$  algorithm ...

2. Required fewer MQs than the other techniques
3. Temporal distance did not affect its performance



# Research Objectives



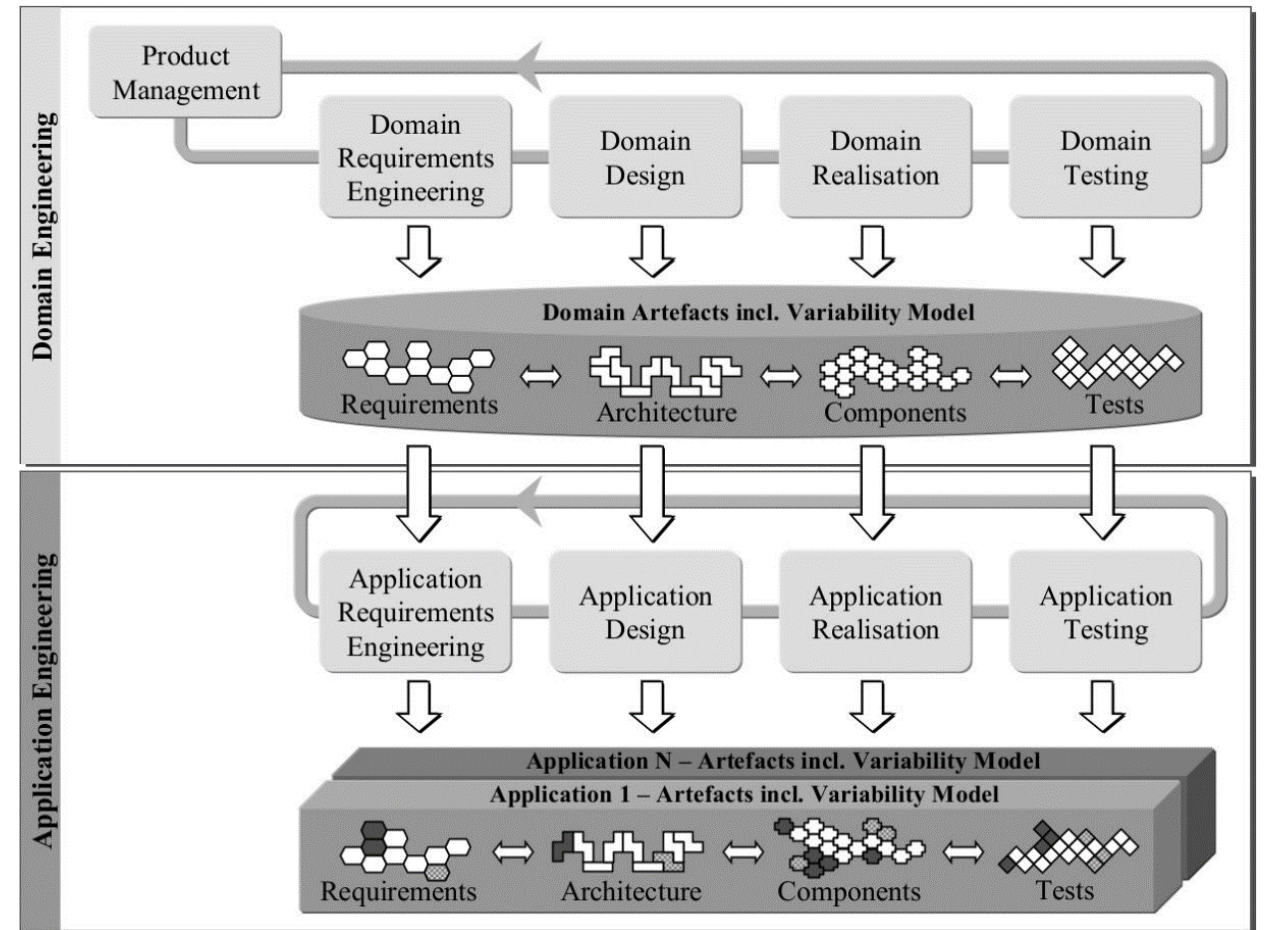
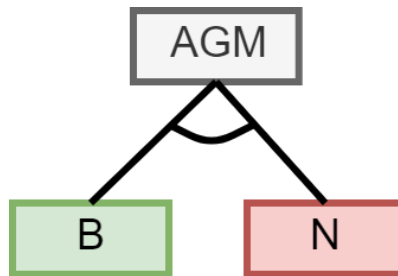
# Learning from Difference

An Automated Approach for Learning Family Models from Software Product Lines



# Context (Learning from Difference)

- Software product lines (SPL)
  - Variability *in space* (e.g., feature model)
  - Common set of reusable assets
  - Product configurations

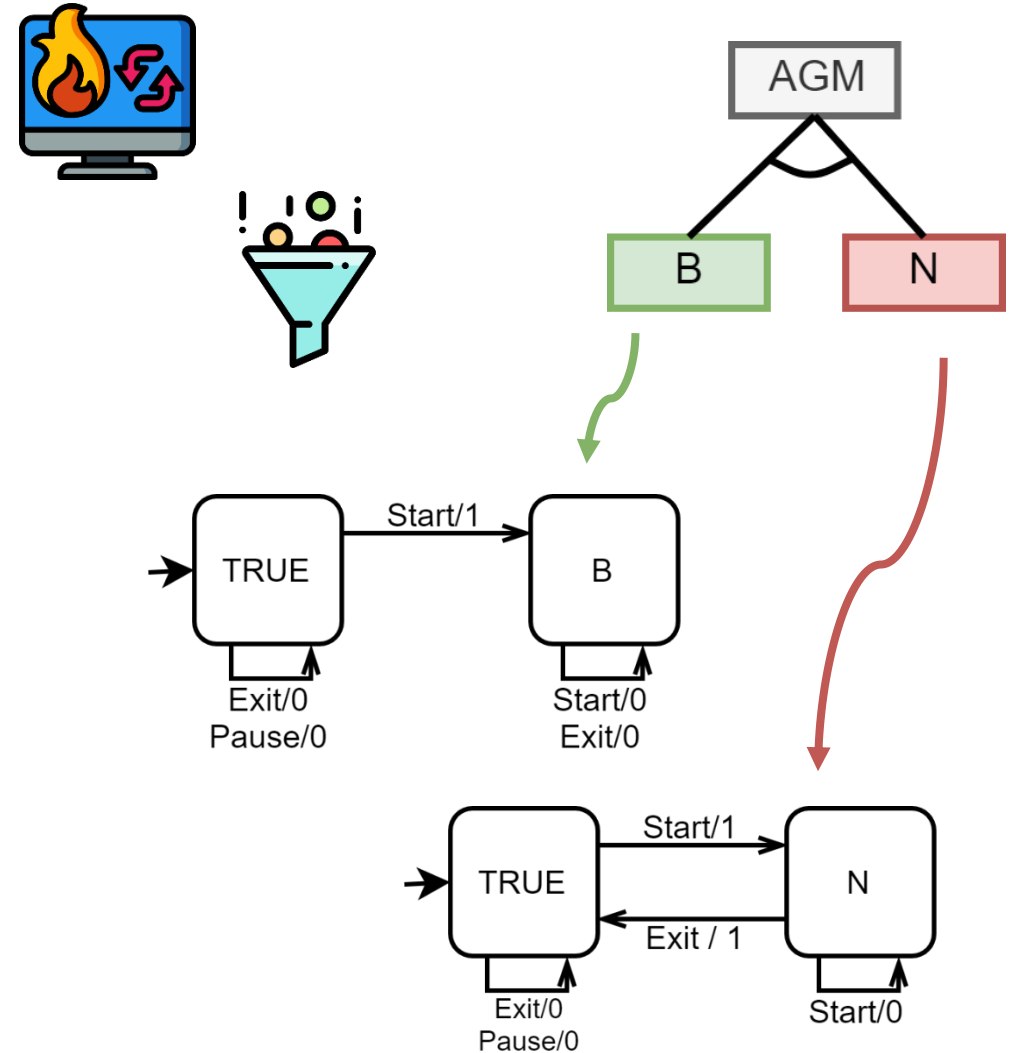


# Context (Learning from Difference)

- Analysis and modeling of SPLs
  - Product-based strategies
    - Traditional MBT + Individual product specifications
    - E.g., exhaustive analysis, configuration sampling

## ISSUES

- Redundant analysis
- Scalability (e.g., exponential)
- Feature interaction problem (e.g., T-wise)

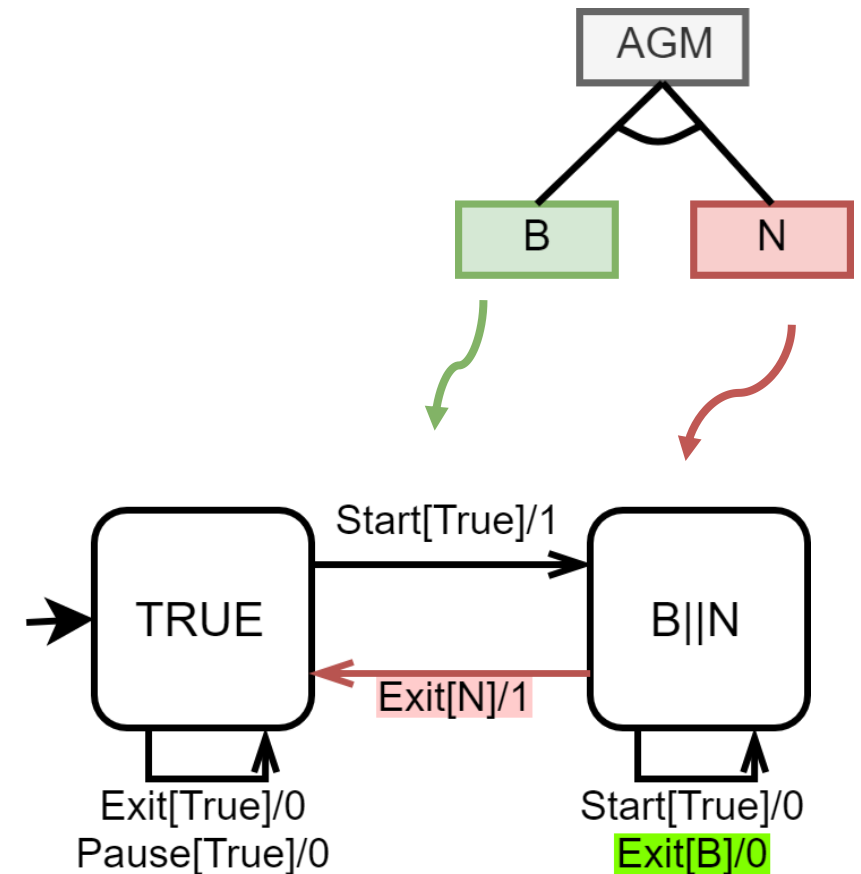


# Context (Learning from Difference)

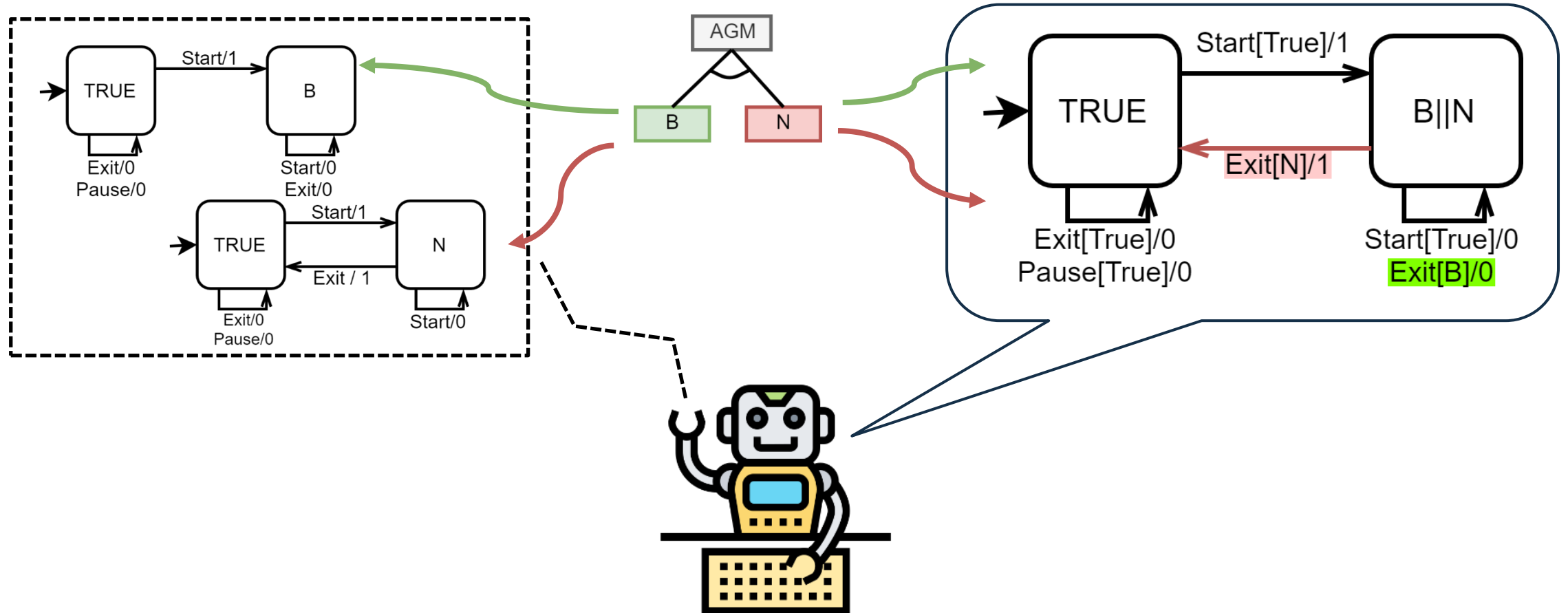
- Analysis and modeling of SPLs
  - Family-based strategies
    - Corner-stone of efficient model-based SPL analysis
    - Family models (e.g., Featured Finite State Machine - FFSM)

## ISSUES

- Model maintenance and evolution
- Traceability vs. Crosscutting features
- Commonalities/variabilities are often unknown

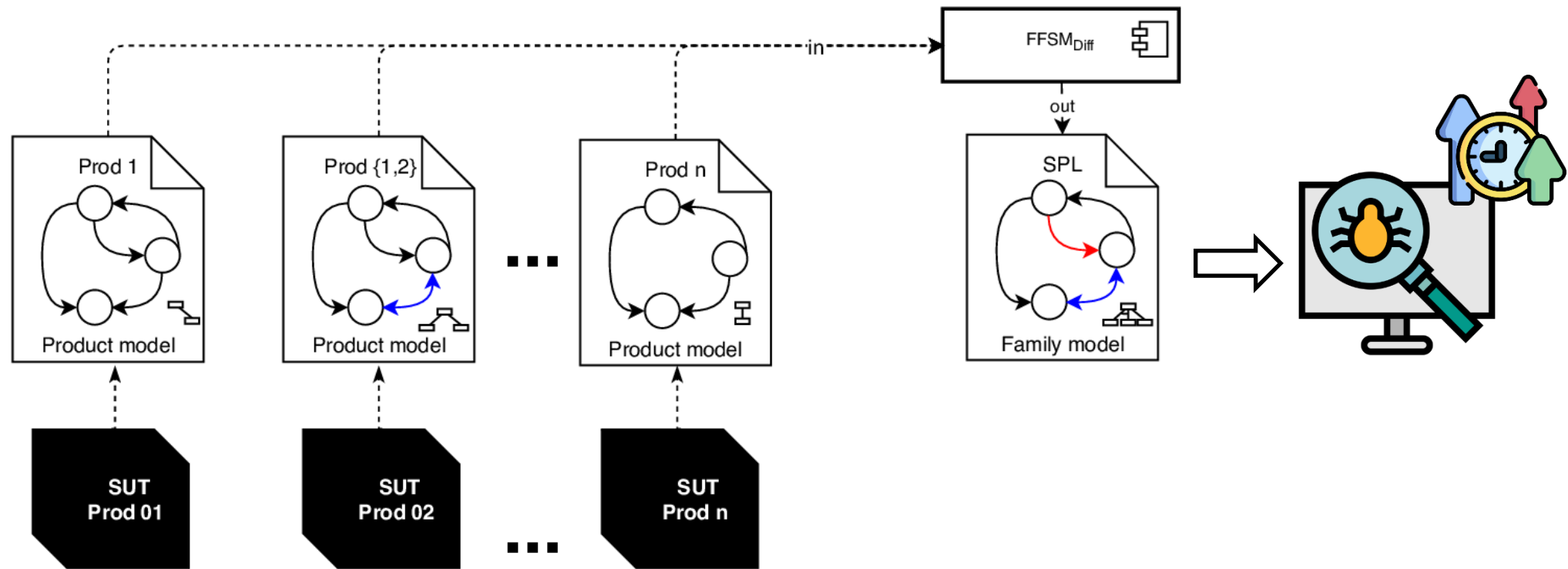


# Research Problem (Learning from Difference)



How can we **leverage** the concept of **model learning** to the task of **behavioral variability modeling**?

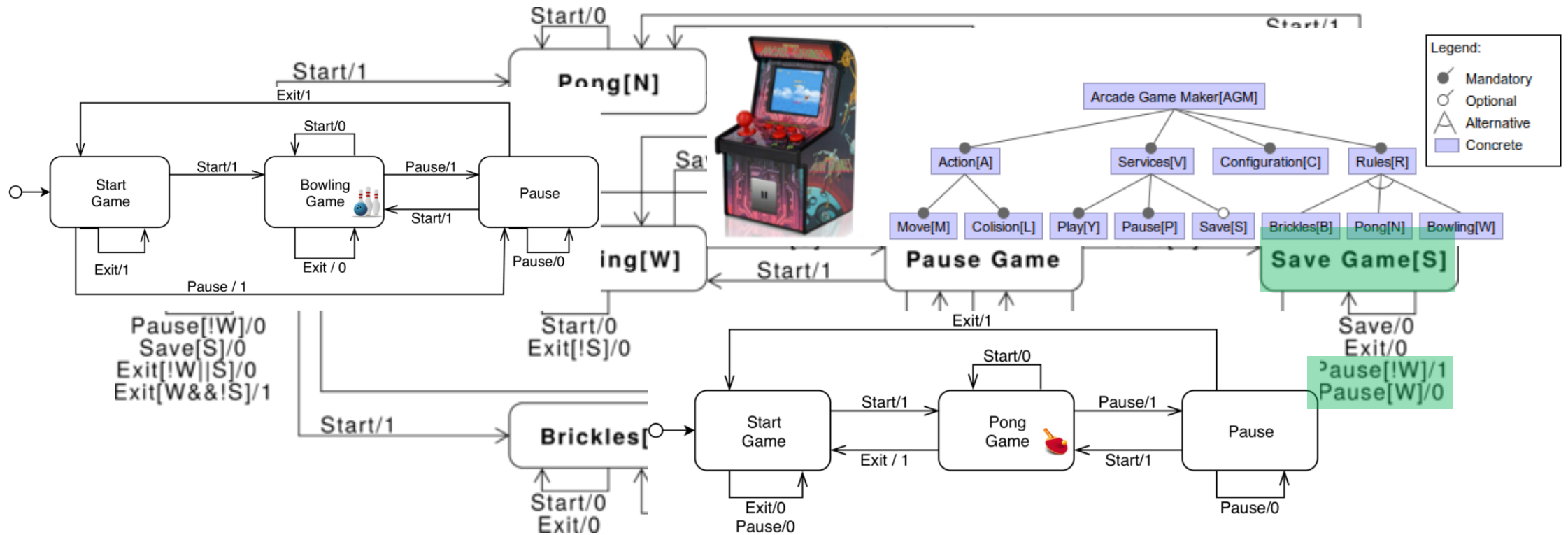
# Contribution (Learning from Difference)



The FFSDiff (FFSM<sub>Diff</sub>) algorithm for **learning succinct family models** from individual product specifications of **software product lines**

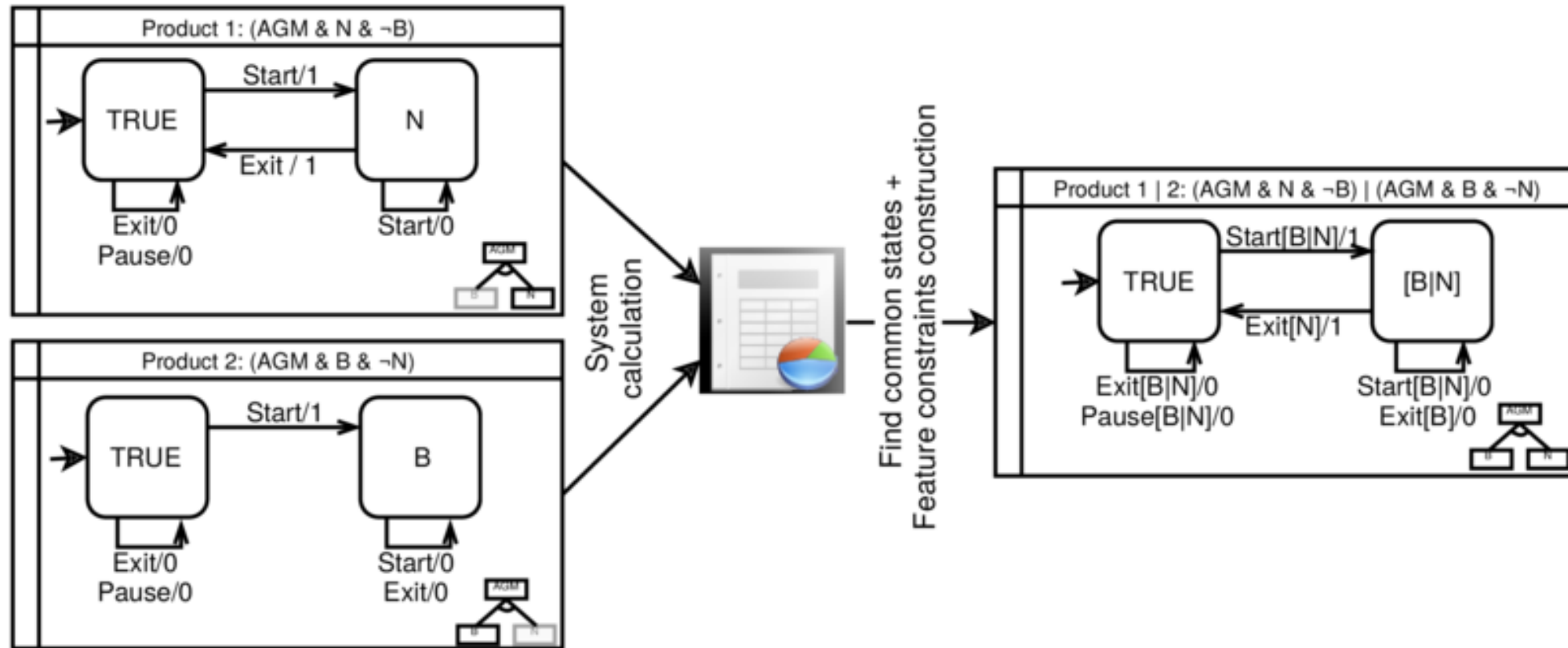


# Featured Finite State Machines (FFSM)



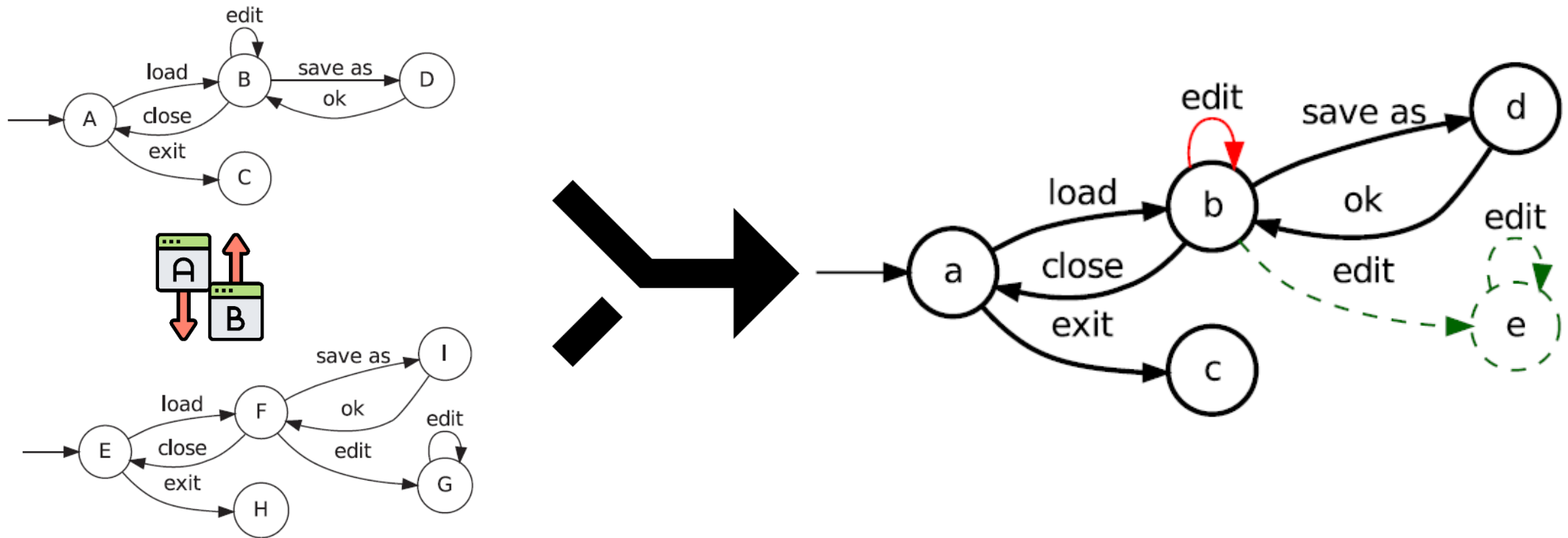
An FFSM is a family-based representation of a product-line that unifies product-specific Mealy machines and captures the functionality of features and their interactions in terms of conditional states/transitions

# FFSM Difference (FFSM<sub>Diff</sub>)



The FFSM<sub>Diff</sub> can learn FFSMs from a product models by employing state-based model comparison and express product-specific behaviors with feature constraints using feature model analysis

# State-based model comparison (LTS<sub>Diff</sub> algorithm)



Comparing the Structures of Two State Machines of a Text Editor

# State-based model comparison (LTS<sub>Diff</sub> algorithm)

$$S_{Succ}^G(a, b) = \frac{1}{2} \frac{\sum_{(c,d,i,o) \in Succ_{a,b}} (1 + k \times S_{Succ}^G(c, d))}{|\sum_r^{out}(a) - \sum_u^{out}(b)| + |\sum_r^{out}(b) - \sum_u^{out}(a)| + |Succ_{a,b}|}$$

Figure: Global similarity score <sup>4</sup>

## 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**.

# State-based model comparison (LTS<sub>Diff</sub> algorithm)

Pair	(St,St)	(St,Po)	(St,Pa)	(Bo,St)	(Bo,Po)	(Bo,Pa)	(Pa,St)	(Pa,Po)	(Pa,Pa)	#Match
(St,St)	10.0	0.0	0.0	0.0	-0.5	0.0	0.0	0.0	0.0	1
(St,Po)	-0.5	8.0	0.0	0.0	0.0	0.0	0.0	0.0	-0.5	2
(St,Pa)	-0.5	0.0	8.0	0.0	-0.5	0.0	0.0	0.0	0.0	2
(Bo,St)	0.0	0.0	0.0	9.5	0.0	0.0	0.0	0.0	0.0	1
(Bo,Po)	0.0	0.0	0.0	0.0	7.5	0.0	0.0	0.0	-0.5	2
(Bo,Pa)	0.0	0.0	0.0	0.0	0.0	12.0	0.0	0.0	0.0	0
(Pa,St)	0.0	0.0	0.0	0.0	-0.5	0.0	7.5	0.0	0.0	2
(Pa,Po)	-0.5	0.0	0.0	0.0	0.0	0.0	0.0	10.0	0.0	1
(Pa,Pa)	-0.5	0.0	0.0	0.0	-0.5	0.0	0.0	0.0	5.5	3

Table 1: Illustration of a system of linear equations

# State-based model comparison (LTS<sub>Diff</sub> algorithm)

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

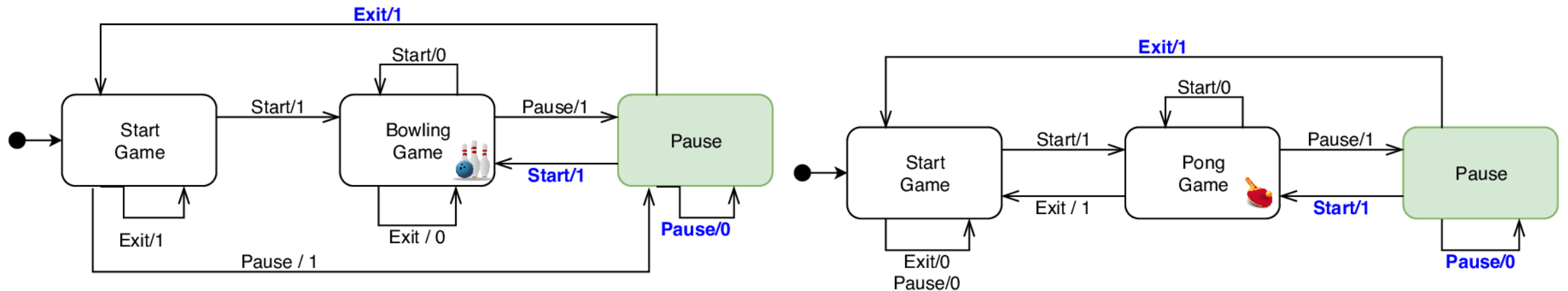


Figure: Two examples of product FSMs and their similarity scores

# The FFSM<sub>Diff</sub> algorithm

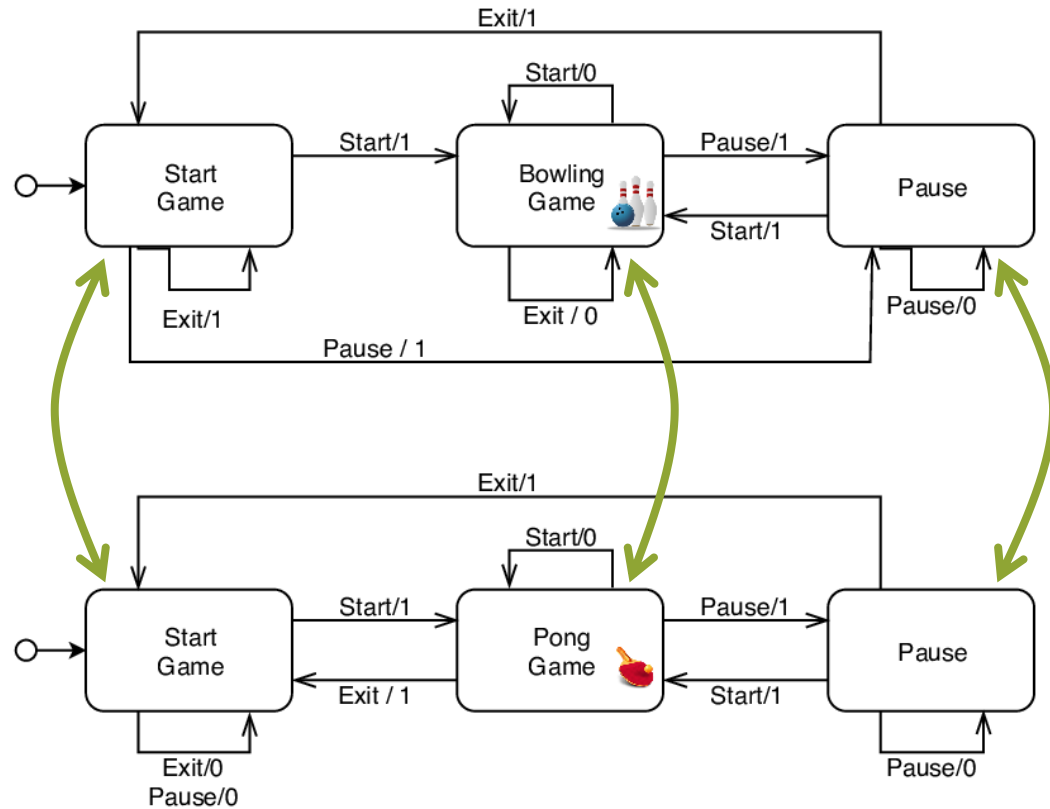


Figure: Two examples of product FSMs

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

$$\text{pair}(St, Po) = 0.29$$

$$\text{pair}(St, Pa) = 0.28$$

$$\text{pair}(Bo, St) = 0.11$$

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

$$\text{pair}(Bo, Pa) = 0$$

$$\text{pair}(Pa, St) = 0.29$$

$$\text{pair}(Pa, Po) = 0.11$$

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

Figure: Pairwise state similarity

Our modification #2: The state mapping is used to annotate conditional states/transition

# The FFSM<sub>Diff</sub> algorithm

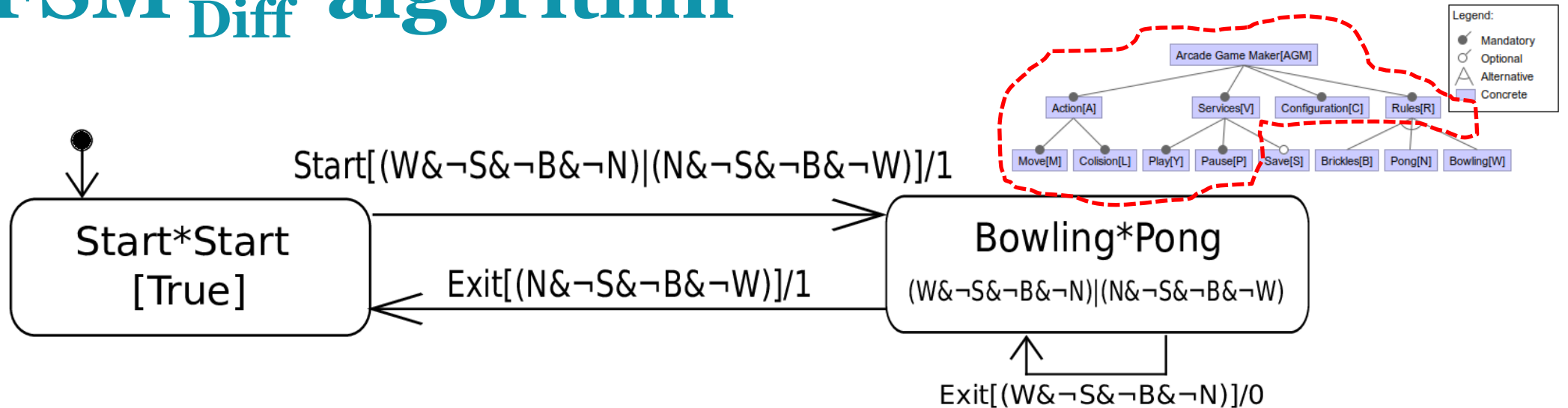


Figure: Fragment of the FFSM learnt from two products of the AGM SPL.

## Simplified configuration – Example

$$\rho_{Bowling} = (W \wedge \neg S \wedge \neg B \wedge \neg N)$$

$$\rho_{Pong} = (N \wedge \neg S \wedge \neg B \wedge \neg W)$$

**Our modification #3:** We use feature model analysis to identify core features of the SPL and simplify feature constrains



# Empirical Evaluation

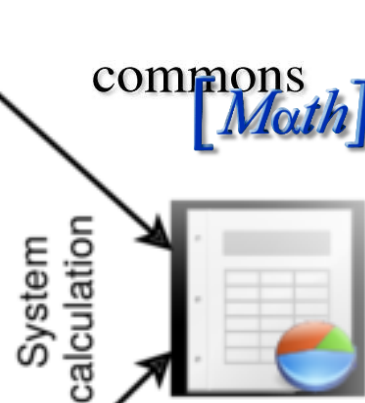
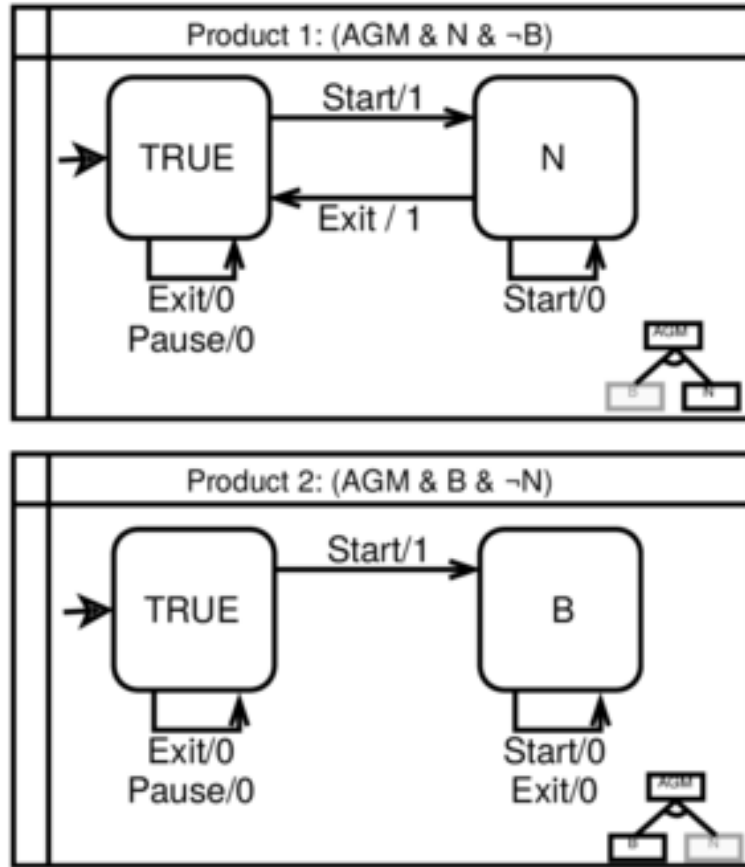
# Empirical Evaluation

**RQ1)** Is our approach effective in learning succinct family models compared to the total size of the product pairs under learning?

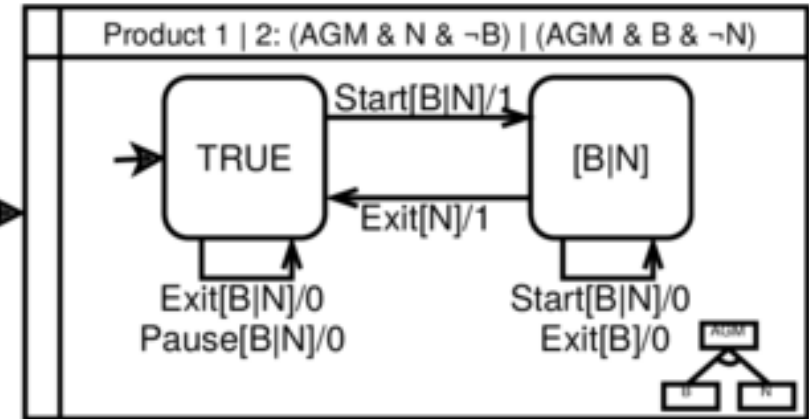
**RQ2)** Is the size of learned family models influenced by the configuration similarity degree of the products under learning?

**RQ3)** Is our approach effective in learning succinct family models compared to the total size of the hand-crafted models?

# Empirical Evaluation



Find common states +  
Feature constraints construction



# Subject Systems

SPL		Feature model		Family model	
ID	Name	Features	Valid conf.	States	Transitions
AGM	Arcade Game Maker	13	6	6	35
VM	Vending Machine	9	20	14	197
WS	Wiper System	8	8	13	112
AEROUC5	Aero UC5	7	9	25	450
CPTERMINAL	Card Payment	13	30	11	176
MINEPUMP	Minepump	9	32	25	575

Table 10 – Description of the SPLs under learning - Feature and family models

# Analysis of Results (RQ1 – Size of Product Pairs)

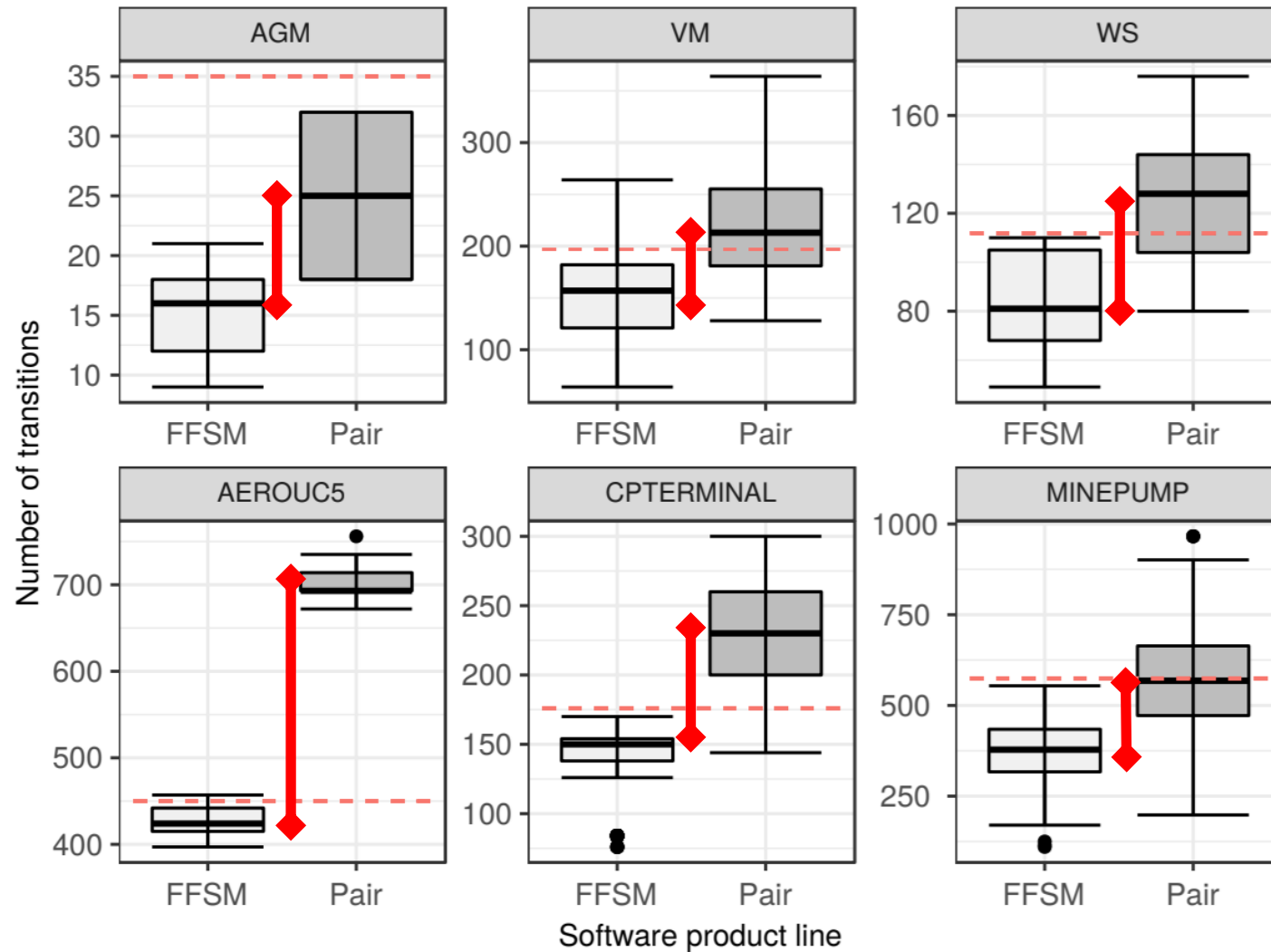


Figure 26 – Number of transitions in the learned FFSSMs and pairs of products

# Analysis of Results (RQ2 – Configuration similarity)

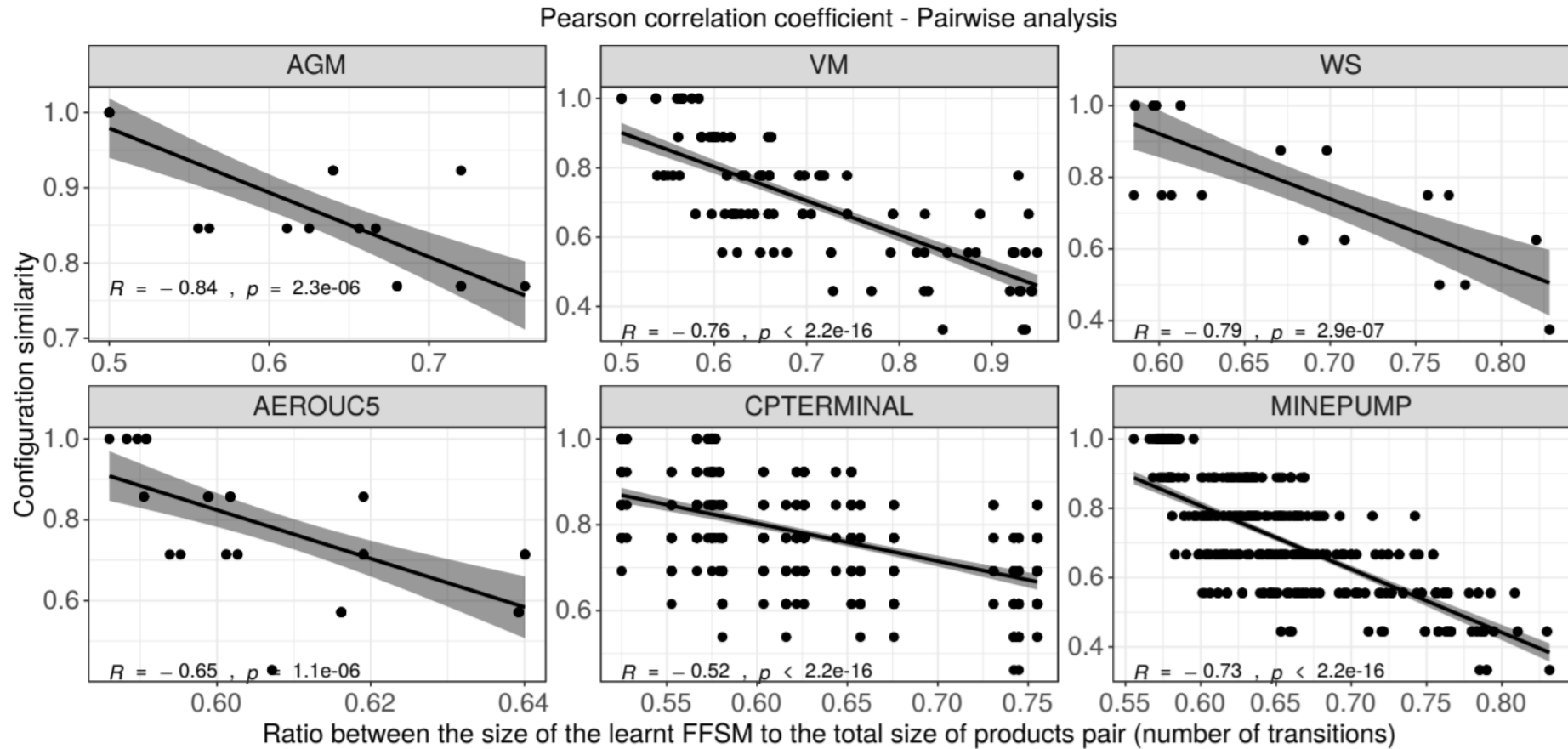


Figure 28 – Scatter plots for the relationship between the normalized size of the learned FFSM and configuration similarity

# Analysis of Results (RQ3 – Size of Handcrafted models)

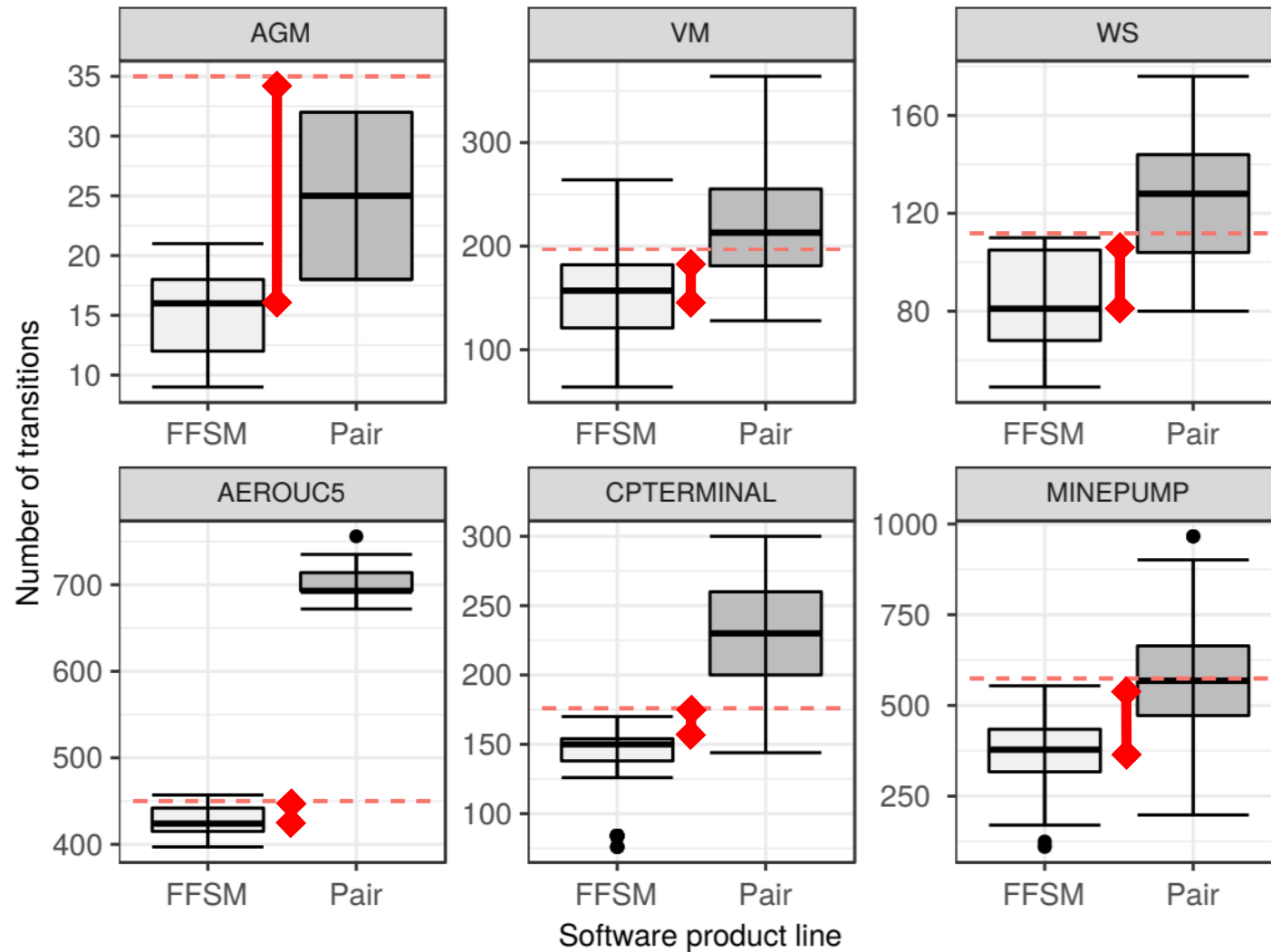


Figure 26 – Number of transitions in the learned FFSSMs and pairs of products

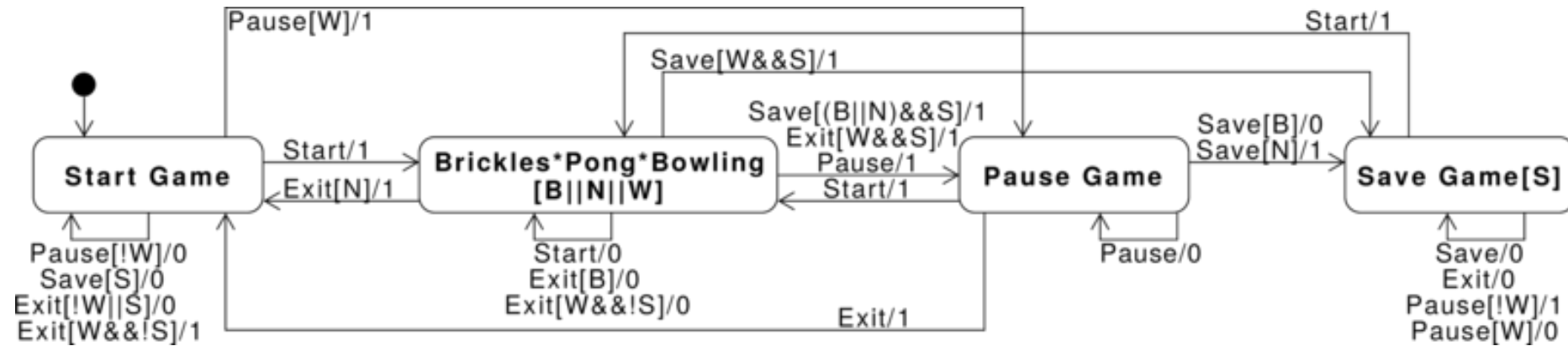
# Analysis of Results (RQ3 – Size of Handcrafted models)



Figure 29 – Size of the recovered FFSMs



# Analysis of Results (RQ3 – Size of Handcrafted models)



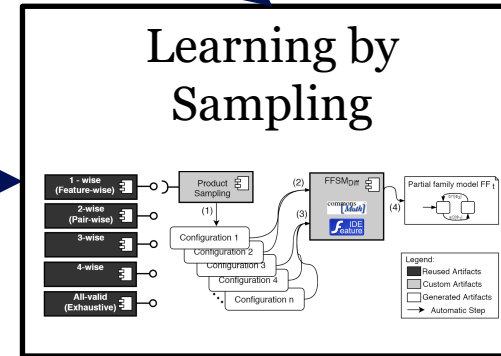
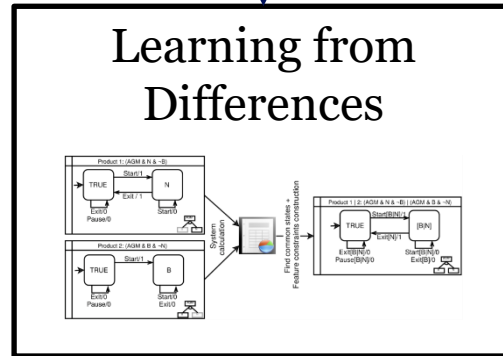
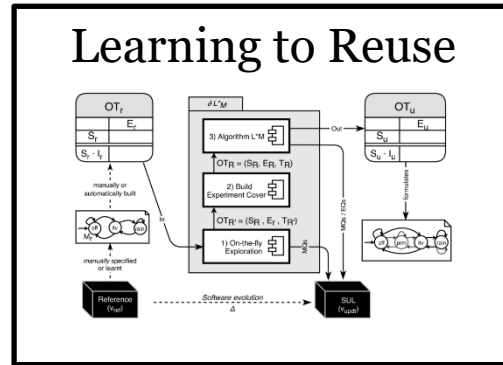
# Summary (Learning from Difference)

The FFSM<sub>Diff</sub> algorithm is able to...



1. Learn **fresh FFSMs** from products pairs
  - Especially if there is **high feature reuse** (i.e., configuration similarity)
2. Incorporate **new product behavior** into an existing FFSM
  - Family model recovery (e.g., reverse engineering, re-engineering)

# Research Objectives



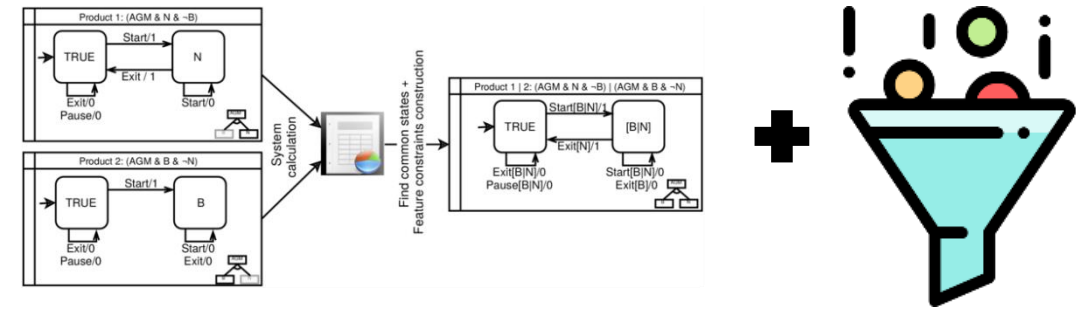
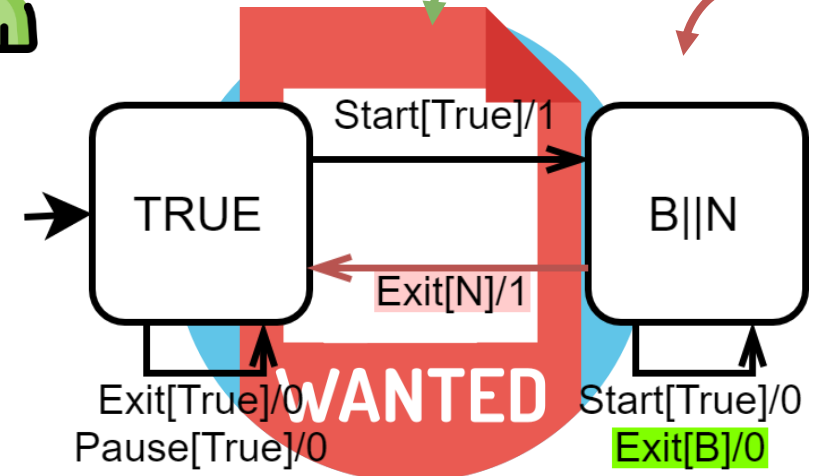
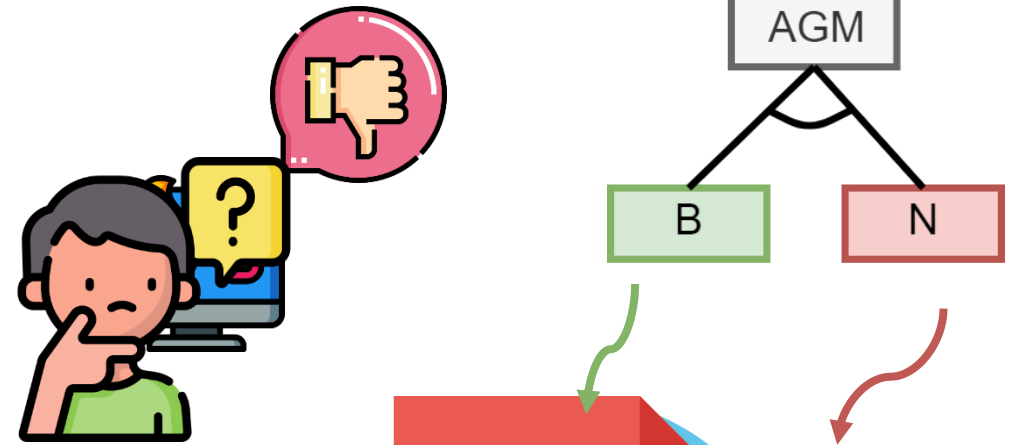
# Learning by Sampling

Learning Behavioral Family Models from Software Product Lines

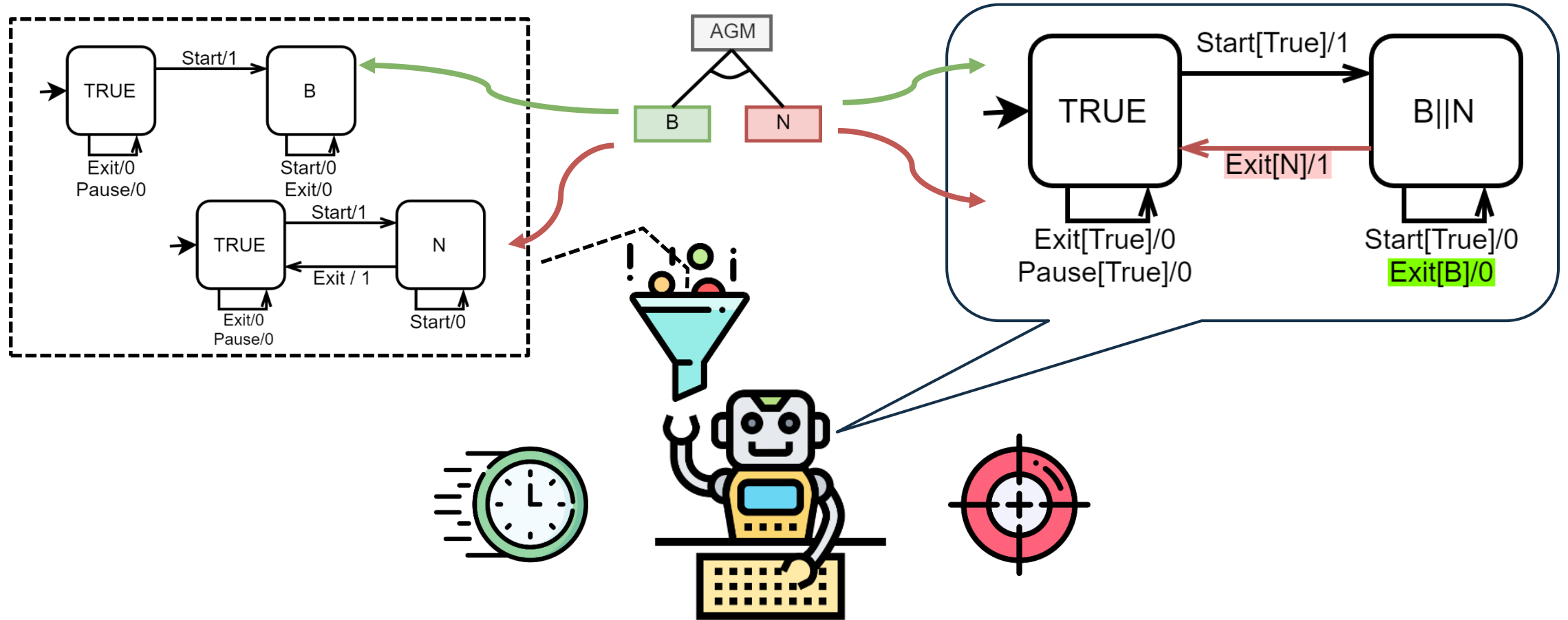


# Context (Learning by Sampling)

- Software product lines (SPL)
  - Product-based strategies: *Impractical?*
  - Family-based strategies: *Models?*
- Family model learning  $\rightarrow$  FFSM<sub>Diff</sub>
  - Exhaustive learning
  - Learning by Sampling



# Research Problem (Learning by Sampling)



How can we **optimize family model learning** to make it more **effective**?

# Contribution (Learning by Sampling)

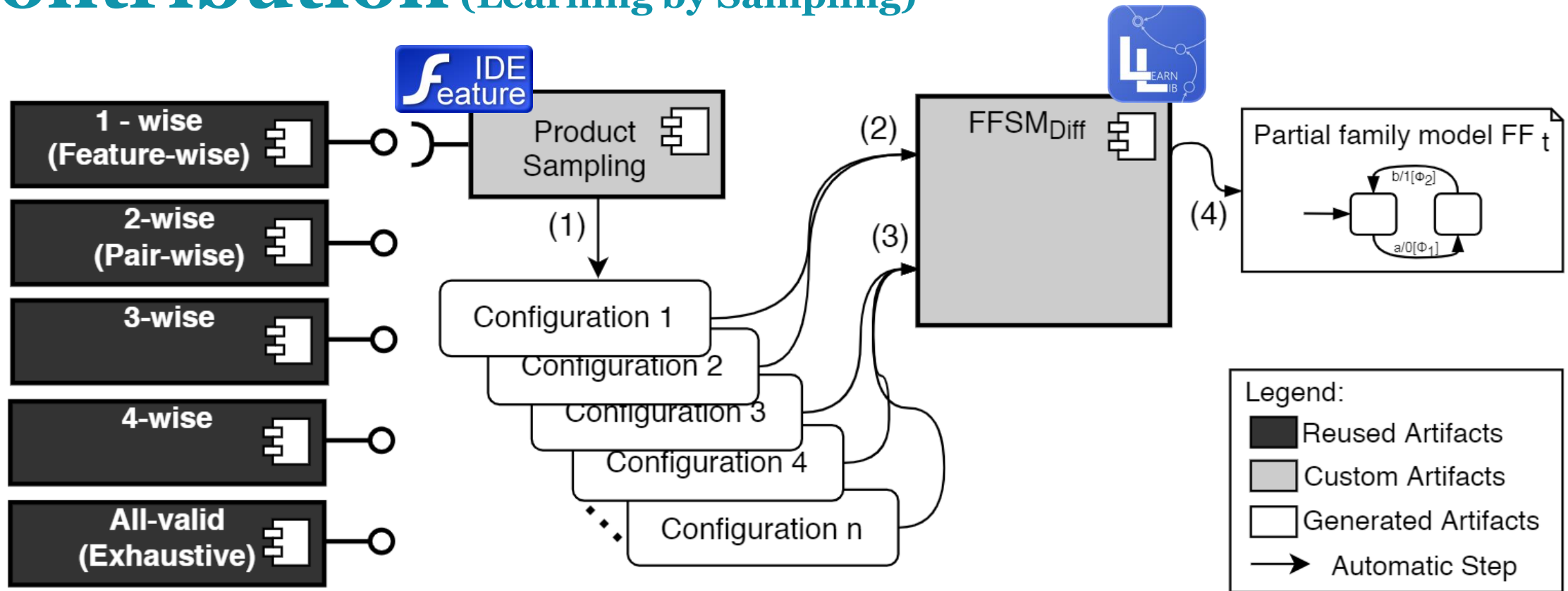


Fig. 8: Experiment design - Learning FFSMs by product sampling

# Empirical Evaluation



# Empirical Evaluation

**RQ4)** Is our approach effective in learning precise family models by sampling compared to exhaustive learning?

# Subject Systems

SPL	Size of the sampled subset generated by T-wise				
	Feature-wise	Pair-wise	3-wise	4-wise	All-valid
AGM	3	6	6	6	6
VM	2	6	13	19	20
WS	2	5	8	8	8
AEROUC5	3	6	9	9	9
CPTERMINAL	3	8	16	24	30
MINEPUMP	3	7	13	24	32

Table 13 – Number of configurations in the subsets generated by each criteria

# Analysis of Results (RQ4 – Learning by Sampling)

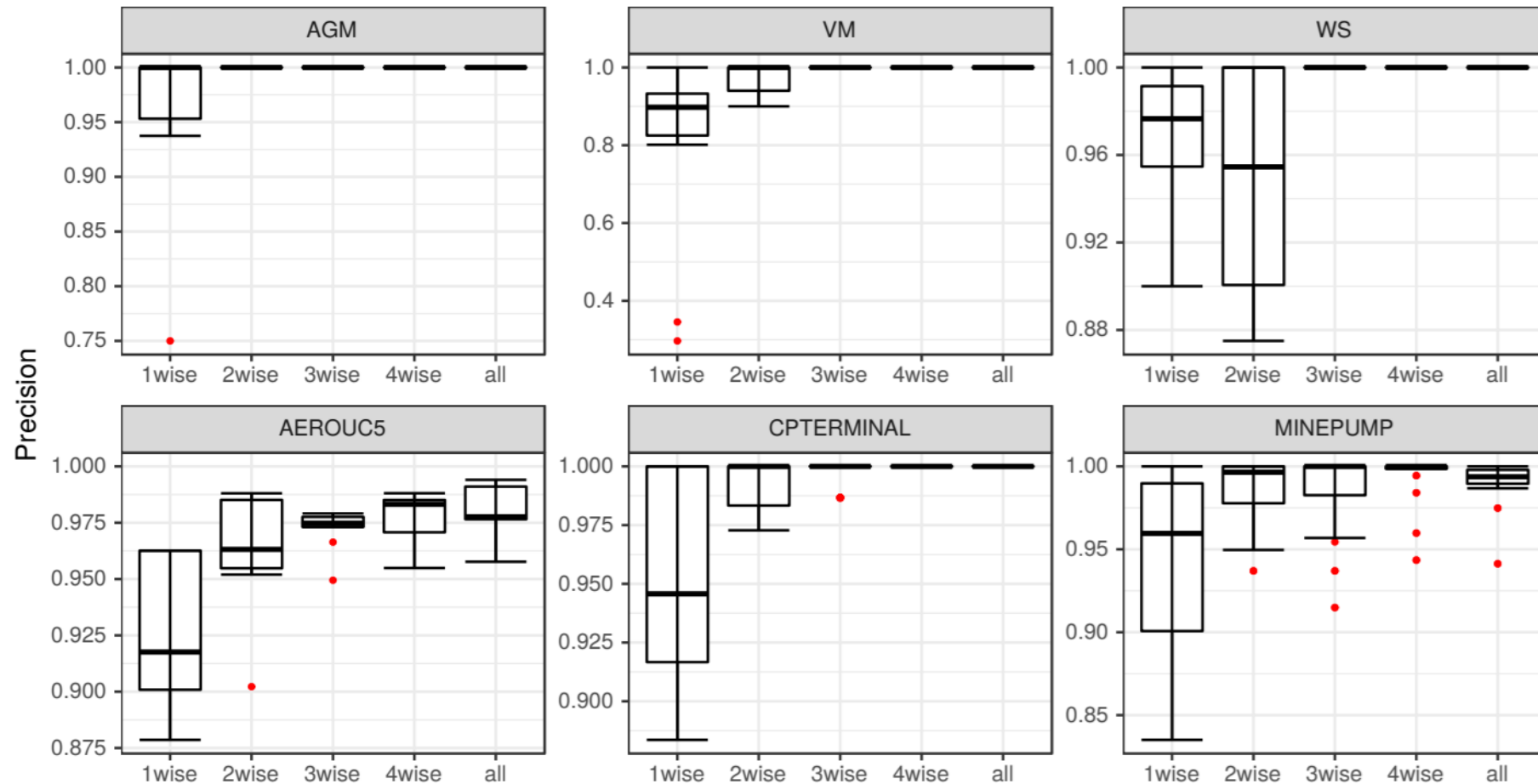


Figure 31 – Model precision by sampling criteria

Higher values of T

More precise family models

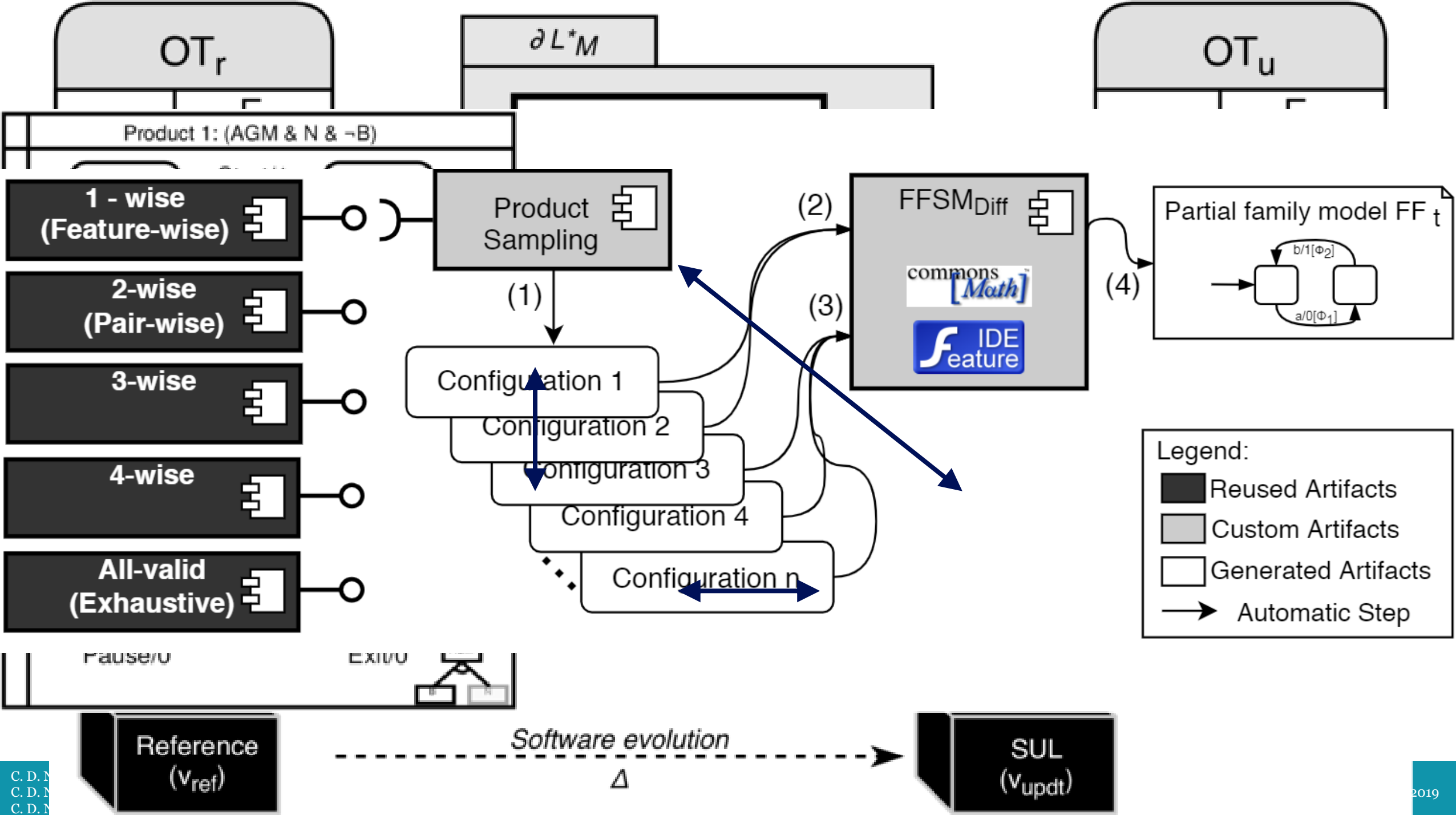
# Summary (Learning by Sampling)

- 1. Learning by sampling** can lead to family models **as precise as** those obtained **by exhaustive analysis**
- 2. Higher interaction strengths** lead to higher coverage
- We show evidences that **product sampling** can be **helpful** to **family model learning and recovery**





# Final Remarks and Future Work






# The main publications of this PhD Thesis



**Learning to Reuse: Adaptive Model Learning for Evolving Systems**

Carlos Diego N. Damasceno<sup>1,2</sup>, Mohammad Reza Mousavi<sup>2</sup>, and Adenilso da Silva Simao<sup>1</sup>

<sup>1</sup> University of Sao Paulo (ICMC-USP), São Carlos, SP, Brazil  
damascenodiego@usp.br, adenilso@icmc.usp.br  
<sup>2</sup> University of Leicester, Leicester, UK  
mm789@leicester.ac.uk





**Learning From Families: Inferring Behavioral Variability From Software Product Lines**

Carlos Diego Nascimento Damasceno

University of Sao Paulo (BR) and University of Leicester (UK)  
damascenodiego@usp.br

**Abstract**  
Family models are behavioral specifications extended with variability constraints that enable efficient model-based analysis of software product lines (SPL). Albeit reasonably



**Learning from Difference: An Automated Approach for Learning Family Models from Software Product Lines**


Carlos Diego N. Damasceno  
damascenodiego@usp.br  
University of Sao Paulo, BR and  
University of Leicester, UK

Mohammad Reza Mousavi  
mm789@leicester.ac.uk  
University of Leicester  
Leicester, UK

Adenilso Simao  
adenilso@icmc.usp.br  
University of Sao Paulo (ICMC-USP)  
São Carlos, SP, BR

1 INTRODUCTION

ABSTRACT



Empirical Software Engineering manuscript No.  
(will be inserted by the editor)

**Learning by Sampling: Learning Behavioral Family Models from Software Product Lines**

Carlos Diego Nascimento Damasceno  
Mohammad Reza Mousavi  
Adenilso da Silva Simao

# Other contributions

## RESEARCH Open Access

### Similarity testing for role-based access control systems

Carlos Diego N. Damasceno<sup>1,2\*</sup>, Paulo C. Masiero<sup>1,2</sup> and Adenildo Simão<sup>1,2</sup>

**Abstract:** Access control systems demand rigorous verification and validation approaches; otherwise, they can end up with security breaches. Finite state machines based testing has been successfully applied to RBAC systems, and enabled to obtain effective test cases, but very expensive. To deal with the cost of three test suites, test prioritization techniques can be applied to improve fault detection along test execution. Recent studies have shown that similarity functions can be very efficient at prioritizing test cases. This technique is named similarity testing and assumes the hypothesis that resembling test cases tend to have similar fault-detection capabilities. Thus, there is a gain from similar test cases, and fault detection rates can be improved if test diversity increases.

**Objective:** In this paper, we propose a similarity testing approach for RBAC systems named RBAC similarity and compare to simple diversity and random prioritization. RBAC similarity combines the disparity degree of pairs of test cases with their relation to the RBAC policy under test to maximize test diversity and the coverage of its constraints.

**Method:** Five RBAC policies and fifteen test suites were prioritized using each of the three test prioritization techniques and compared using the Average Percentage Faults Detected metric.

**Results:** Our results showed that the combination of the disparity degree to the relevance of a test case to RBAC policies in the RBAC similarity can be more effective than random prioritization and simple diversity, by itself, or most of the priorizations.

**Conclusion:** The RBAC similarity criterion is suitable as a test prioritization criteria for test suites generated from finite state machines models specifying RBAC systems.

**Keywords:** Finite state machines, Role-Based Access Control (RBAC), Test prioritization, Similarity testing.

**1 Introduction**  
Access control is one of the major pillars of software security. It is responsible for ensuring that only intended users can access data and only the required permissions to accomplish a task is guaranteed (Friedrich et al., 2007). In this context, the Role-Based Access Control (RBAC) model has been established as one of the most significant access control paradigms. In RBAC, users receive privileges through role assignments and activate these during sessions (ANSI, 2004). Despite its simplicity, mistakes can occur during development and lead to faults, or other security breaches. Therefore, software verification and validation becomes necessary.

The authors. 2018. Open Access. This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

**ICMC USP** Um Algoritmo Paralelo para Priorização de Testes Baseada em Simulação usando OpenMPI  
Carlos Diego N. Damasceno<sup>1,2\*</sup>, Paulo C. Masiero<sup>1,2</sup> and Adenildo Simão<sup>1,2</sup>

**Resumo:** Priorização de testes baseada em simulação usa funções de similaridade para aprimoramento de testes, maximizando sua diversidade [1, 2]. Testes parelhos tendem a detectar falhas semelhantes em um sistema, logo sua execução simultânea não traz ganhos [3]. Cálculo de matriz de similaridade (SM) com base no grau de similaridade entre todos os pares de testes (costo de O(N<sup>2</sup>)). Otimização de testes: algoritmo de priorização baseado em simulação usando OpenMPI (PASP).

**Palavras-chave:** Máquinas de estados finitos, teste de software, teste de priorização, teste de diversidade.

**Abstract:** Access control systems demand rigorous verification and validation approaches; otherwise, they can end up with security breaches. Finite state machines based testing has been successfully applied to RBAC systems, and enabled to obtain effective test cases, but very expensive. To deal with the cost of three test suites, test prioritization techniques can be applied to improve fault detection along test execution. Recent studies have shown that similarity functions can be very efficient at prioritizing test cases. This technique is named similarity testing and assumes the hypothesis that resembling test cases tend to have similar fault-detection capabilities. Thus, there is a gain from similar test cases, and fault detection rates can be improved if test diversity increases.

**Objective:** In this paper, we propose a similarity testing approach for RBAC systems named RBAC similarity and compare to simple diversity and random prioritization. RBAC similarity combines the disparity degree of pairs of test cases with their relation to the RBAC policy under test to maximize test diversity and the coverage of its constraints.

**Method:** Five RBAC policies and fifteen test suites were prioritized using each of the three test prioritization techniques and compared using the Average Percentage Faults Detected metric.

**Results:** Our results showed that the combination of the disparity degree to the relevance of a test case to RBAC policies in the RBAC similarity can be more effective than random prioritization and simple diversity, by itself, or most of the priorizations.

**Conclusion:** The RBAC similarity criterion is suitable as a test prioritization criteria for test suites generated from finite state machines models specifying RBAC systems.

**Keywords:** Finite state machines, Role-Based Access Control (RBAC), Test prioritization, Similarity testing.

**1 Introduction**  
Access control is one of the major pillars of software security. It is responsible for ensuring that only intended users can access data and only the required permissions to accomplish a task is guaranteed (Friedrich et al., 2007). In this context, the Role-Based Access Control (RBAC) model has been established as one of the most significant access control paradigms. In RBAC, users receive privileges through role assignments and activate these during sessions (ANSI, 2004). Despite its simplicity, mistakes can occur during development and lead to faults, or other security breaches. Therefore, software verification and validation becomes necessary.

### Evaluating test characteristics and effectiveness of FSM-based testing methods on RBAC systems

Carlos Diego N. Damasceno<sup>1,2\*</sup>, Paulo Cesar Masiero<sup>1,2</sup> and Adenildo Simão<sup>1,2</sup>

**Abstract:** Access control mechanisms demand rigorous software testing approaches; otherwise they can end up with security breaches. Finite state machines (FSM) have been used for testing Role-Based Access Control (RBAC) mechanisms and complete, but significantly larger, test suites can be obtained. Experimental studies have shown that recent FSM testing methods can reduce the overall test suite length for random FSMs. However, since the similarity between random FSMs and those specifying RBAC mechanisms is effective, these outcomes cannot be immediately generalized to RBAC. In this paper, we compare the characteristics and effectiveness of test suites generated by traditional and recent FSM testing methods for RBAC policies specified as FSM models. The methods W, ISI and SPY were applied on RBAC policies specified as FSMs and the test suites obtained were evaluated considering test characteristics (number of faults, average test case length, and test suite length) and effectiveness on the RBAC fault domain. Our results corroborate some outcomes of previous investigations in which test suites presented different characteristics. On average, the SPY method generated test suites with 25% less tests, average test case length 75% greater than W and ISI, and overall length 40% lower. There was no difference among FSM testing methods for RBAC regarding effectiveness. However, the SPY method significantly retained the overall test suite length and the number of errors.

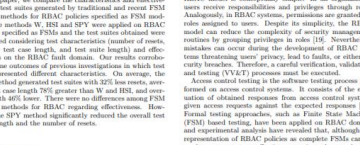


Figure 1: Tempo médio de execução do PLMPSP. Figure 2: Tamanho do PLMPSP. Figure 3: Eficiência do PLMPSP.

**CCS Concepts**  
Security and privacy → Access control; Software and its engineering → Formal software verification; Empirical software validation.

**Keywords**  
Finite state machines, Role-Based Access Control (RBAC), Experiments, Confidence Testing.

**Introduction**  
Access control is one of the major pillars of software security. It is responsible for ensuring that only intended users can access data and only the required permissions to accomplish a task is guaranteed (Friedrich et al., 2007). In this context, the Role-Based Access Control (RBAC) model has been established as one of the most significant access control paradigms. In RBAC, users receive privileges through role assignments and activate these during sessions (ANSI, 2004). Despite its simplicity, mistakes can occur during development and lead to faults, or other security breaches. Therefore, software verification and validation becomes necessary.

### Trusted Autonomous Vehicles: an Interactive Exhibit

Hugo L. S. Azeiteiro, Carlos Diego N. Damasceno, Rayssa Diniziana, Genesio Keflikian, Mihai Mohantaruah, Muhammad Reza Mousavi, Jemima Ogunniye, Jan Oliver Engwert, Nervo Xavier Verdonck, Seyed Wali.

**Abstract:** Recent surveys about autonomous vehicles show that the public is concerned about the safety consequences of system or equipment failures and the vehicles' reactions to unexpected situations. We believe that informing about the technology and quality, i.e., safety and reliability, of autonomous vehicles is paramount to improving public expectations, perception and acceptance. In this paper, we report on the design of an interactive exhibit to illustrate (1) basic technologies employed in autonomous vehicles, i.e., sensors and object classification; and (2) basic principles for ensuring their quality, i.e., employing software testing and simulation. We subsequently report on the delivery of this exhibit titled "Trusted Autonomous Vehicles" at the Brazil Society Summer Institute Exhibition 2019. We describe the process of designing and developing the artifacts used in our exhibit, the theoretical background associated to them, the design of the exhibit, and the lessons learned. The activities and findings of this study can be used by other educators and researchers interested in promoting trust in autonomous vehicles among the general public.

**1. INTRODUCTION**  
The automotive industry is facing a drastic change of paradigm: computer systems (and in particular AI-enabled software) are taking the control step in an industry which was traditionally control oriented mechanics and later electronics. It is predicted that by 2025 road possibly much more than half of the value of a modern car [1] will be with their decrease in the cost of hardware, software will be the dominant cost factor [2]. Moreover, the absolute majority of the innovation in modern cars is embodied by software [2][4].

This new paradigm is a great marker for more automated and autonomous functions. The Society of Automotive Engineers (SAE) classifies the levels of autonomy in 6 levels [5]: from an automation level 0 to full automation level 5. This work is based under the Creative Commons Attribution International License (CC BY 4.0).

### Data Analysis of Multiplex Sequencing at SOLiD Platform: A Probabilistic Approach to Characterization and Reliability Increase

Fábio Manoel França Lobato<sup>1</sup>, Carlos Diego Damasceno<sup>2\*</sup>, Daniela Soares Leite<sup>1</sup>, Andréa Kelly Ribeiro dos Santos<sup>3</sup>, Sylvain Darnet<sup>4</sup>, Carlos Renato Franco<sup>5</sup>, Nandamudi Lankalapalli Vijaykumar<sup>6</sup>, Adamo Lima de Santana<sup>7</sup>

**Abstract:** New sequencing technologies such as Illumina/Sequen, SOLiD/ABI, and GS/Beckte, revolutionized the biological researches. In this context, the SOLiD platform has a particular sequencing type, known as multiplex run, which enables the sequencing of several samples in a single run. It implies in cost reduction and simplifies the analysis of related samples. Meanwhile, this sequencing type requires an additional filtering step to ensure the reliability of the results. Thus, we propose in this paper a probabilistic model which considers the intrinsic characteristics of each sequencing to characterize multiplex runs and filter low-quality data, increasing the data analysis reliability of multiple sequencing performed on SOLiD. The results show that the proposed model proves to be satisfactory due to: 1) identification of faults in the sequencing process; 2) adaptation and development of new protocols for sample preparation; 3) the assignment of a degree of confidence to the data generated; and 4) guiding a filtering process, without discarding useful data in an arbitrary manner.

**Keywords:** Probabilistic Modeling, Health Informatics, SOLiD Sequencing System, Statistical Analysis, Multiplex Sequencing.



1



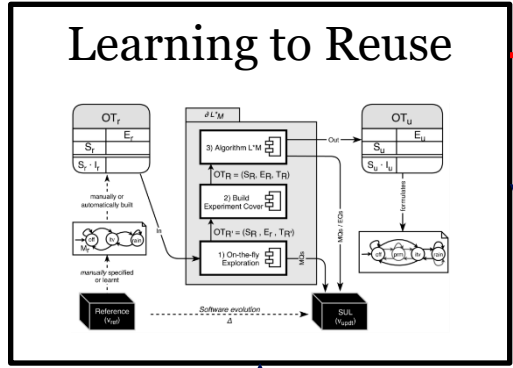
2



# Future Work

Active family model learning

Fingerprinting evolving systems



Adaptive Discriminative Model

Incremental Configurable Queries

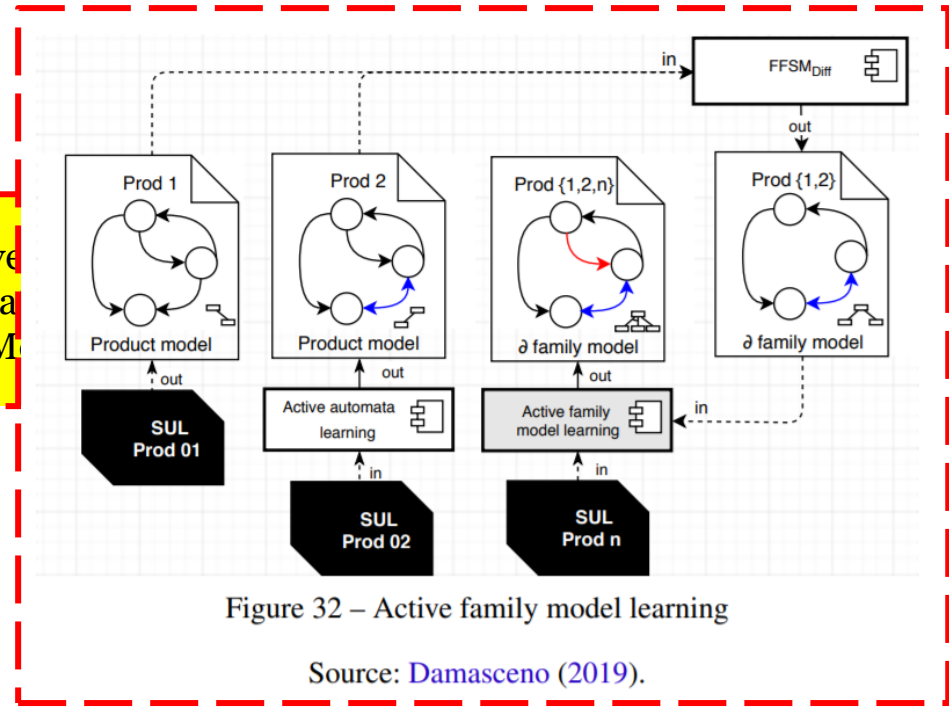
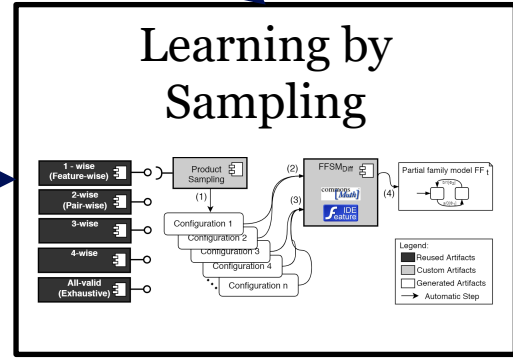
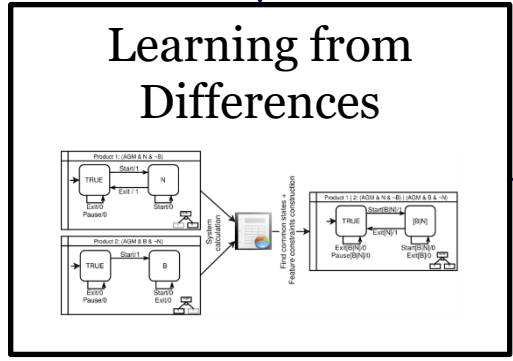


Figure 32 – Active family model learning

Source: Damasceno (2019).

# Thank you



Questions?

