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)

ICMC USP SÃO CARLOS

Universidade de São Paulo

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]

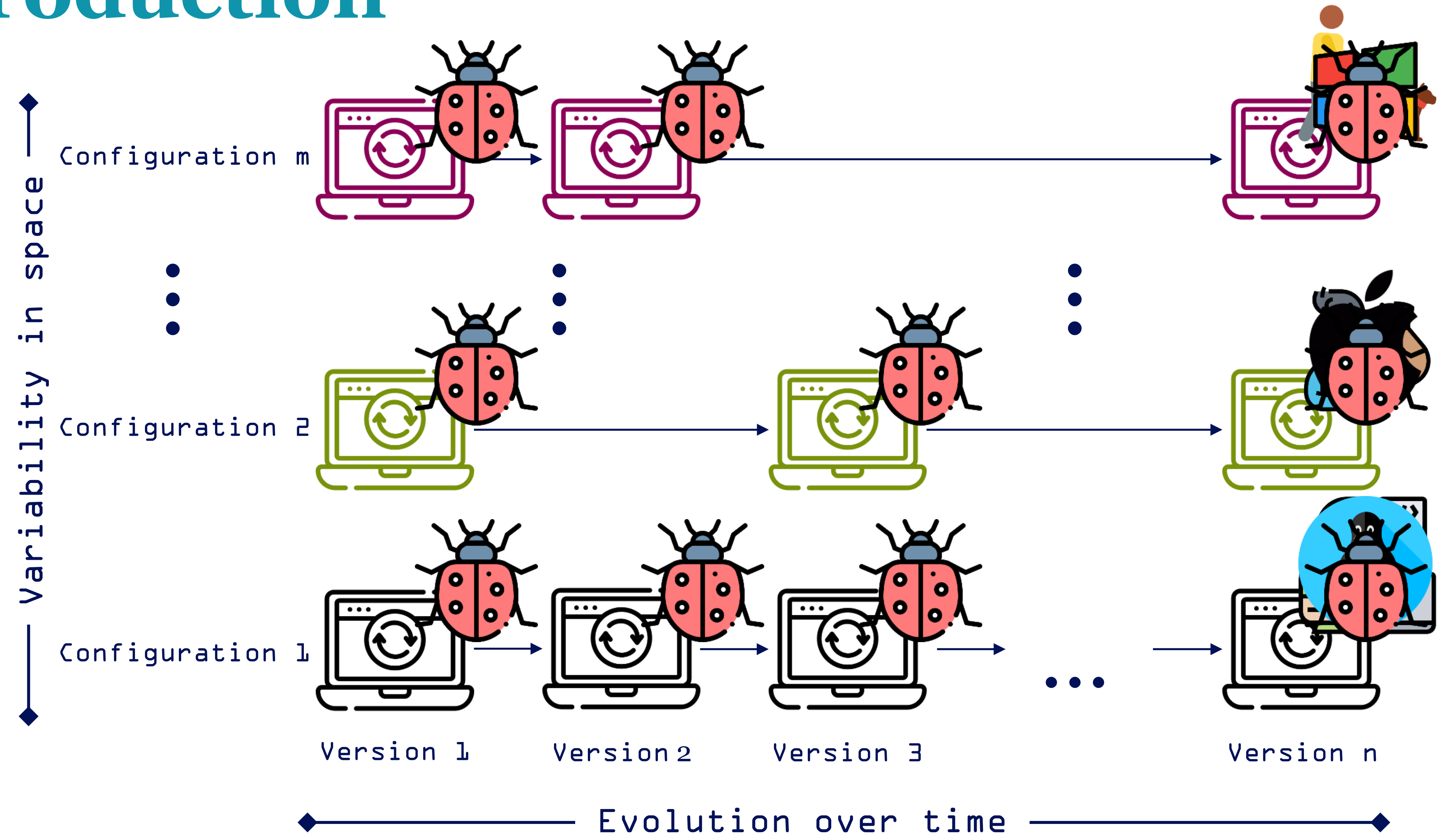
"... modifications after delivery to correct faults, to improve non-functional attributes ..."

Software evolution [Lehman, 1979]

"... programs must be modified because they operate in or address problems in the real world ..."

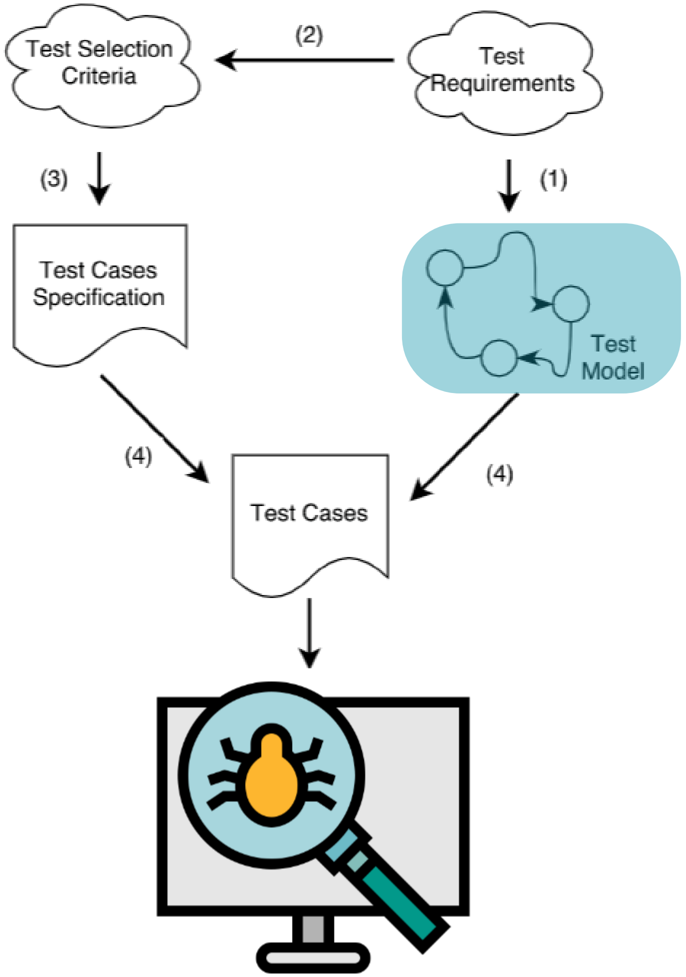
IEEE, 'ISO/IEC/IEEE International Standard for Software Engineering - Software Life Cycle Processes - Maintenance', ISO/IEC 14764:2006, pp. 1–58, Sep. 2006
M. M. Lehman, 'On understanding laws, evolution, and conservation in the large-program life cycle', Journal of Systems and Software, vol. 1, pp. 213–221, 1979

Introduction

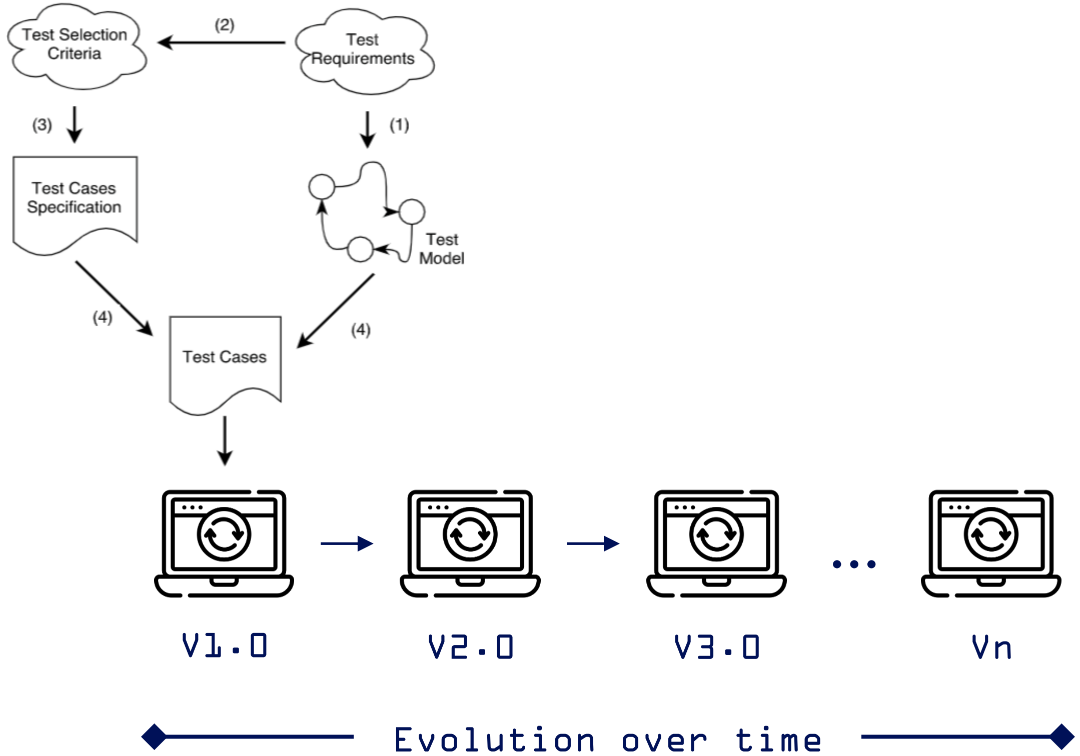


K. Pohl, G. Böckle, and F. J. van der Linden, Software Product Line Engineering: Foundations, Principles and Techniques. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2005.
H. Chockler, D. Kroening, L. Mariani, and N. Sharygina, Eds., Validation of Evolving Software. Cham: Springer International Publishing, 2015.
M. S. Deutsch, 'Tutorial Series 7 Software Project Verification and Validation', Computer, vol. 14, no. 4, pp. 54–70, Apr. 1981

Introduction

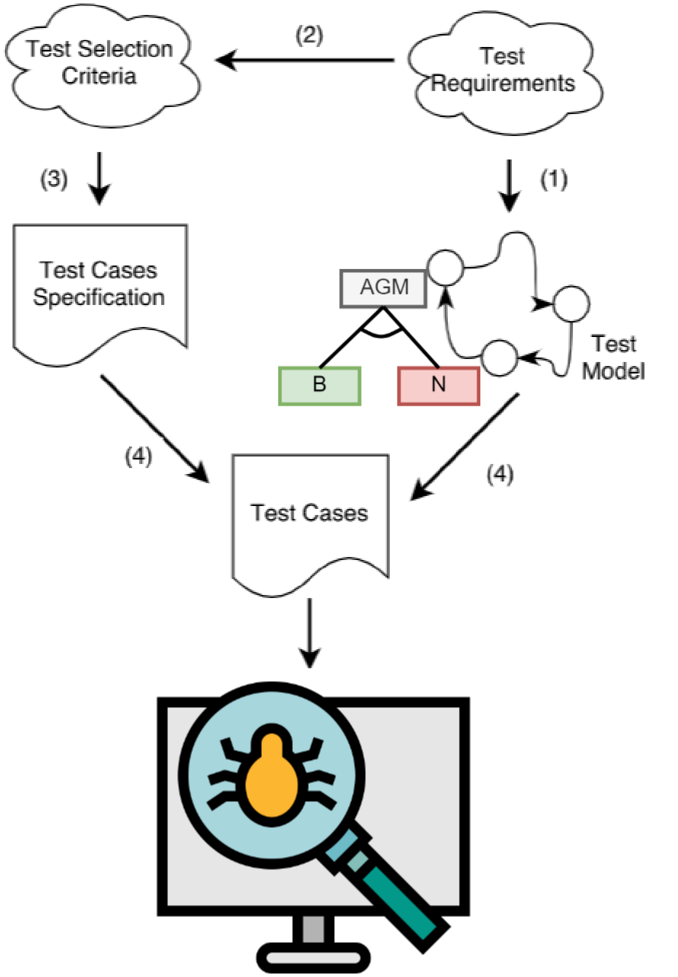


Model-Based Testing (MBT)

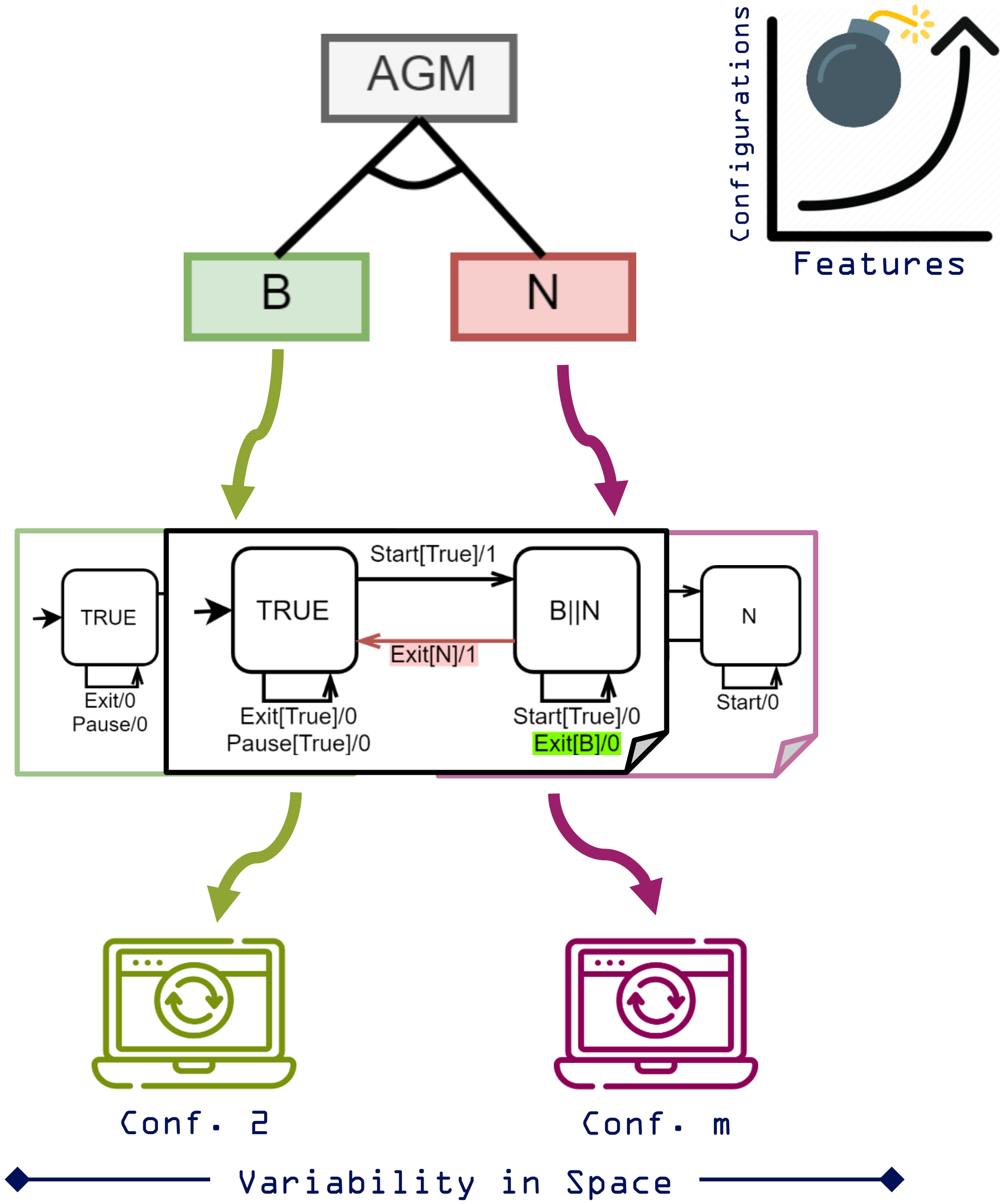


M. Utting, A. Pretschner, and B. Legeard, 'A taxonomy of model-based testing approaches', Software Testing, Verification and Reliability, vol. 22, no. 5, pp. 297–312, 2012.

Introduction

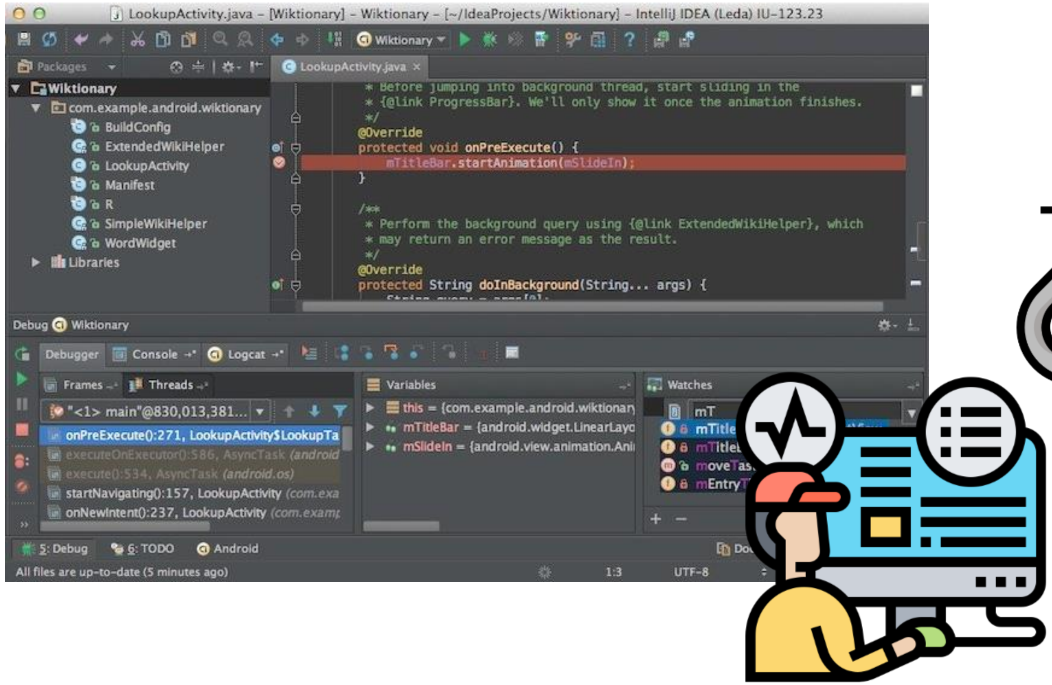


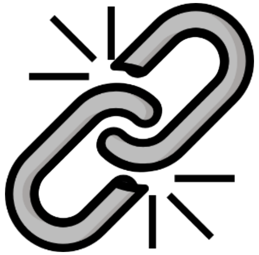
Model-Based Testing (MBT) / Family-based testing

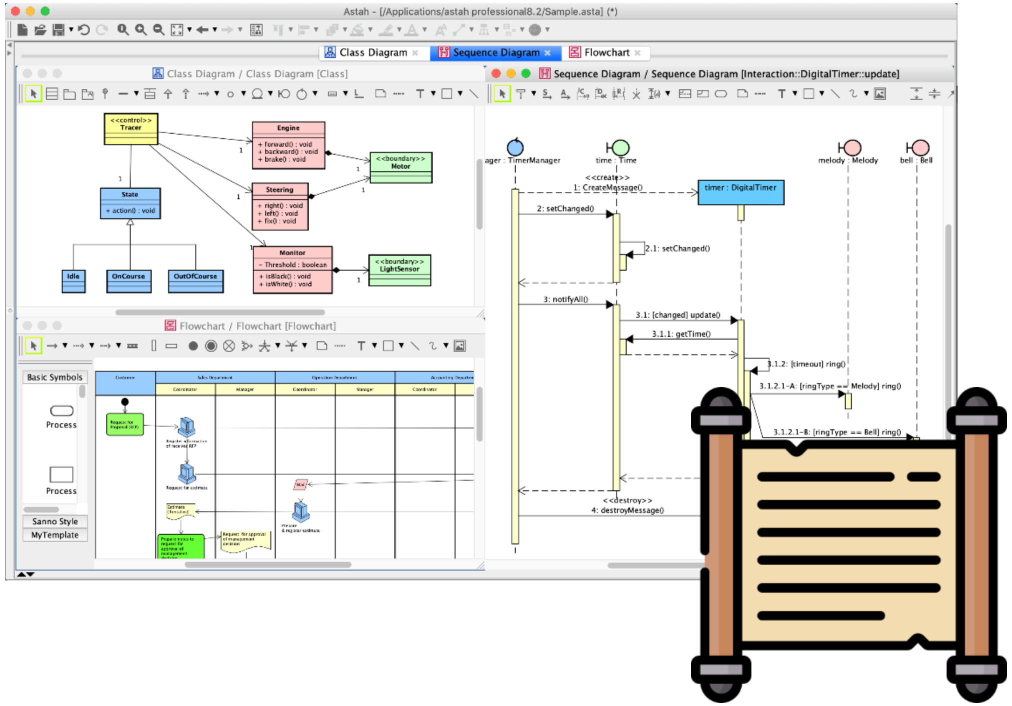


T. Thüm, S. Apel, C. Kästner, I. Schaefer, and G. Saake, ‘A Classification and Survey of Analysis Strategies for Software Product Lines’, ACM Comput. Surv., vol. 47, no. 1, p. 6:1–6:45, Jun. 2014
V. Hafemann Fragal, A. Simao, and M. R. Mousavi, ‘Validated Test Models for Software Product Lines: Featured Finite State Machines’, in `FACS 2016.
S. Oster, ‘Feature Model-based Software Product Line Testing’, PhD Thesis, Technische Universität, Darmstadt, 2012.

Introduction







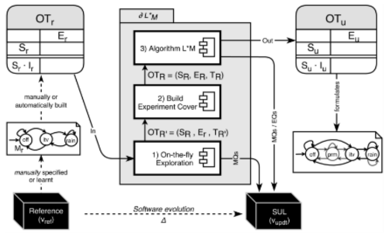
Source code and models should be maintained and evolve together!

Research Problem

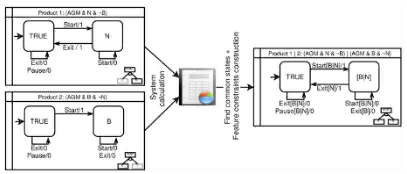
How can we efficiently and effectively learn finite state machines specifying the behavior of an evolving system?

Research Objectives

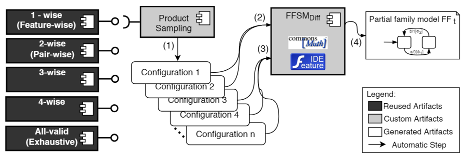
Learning to Reuse



Learning from Differences



Learning by Sampling



Learning to Reuse

Adaptive Model Learning for Evolving Systems

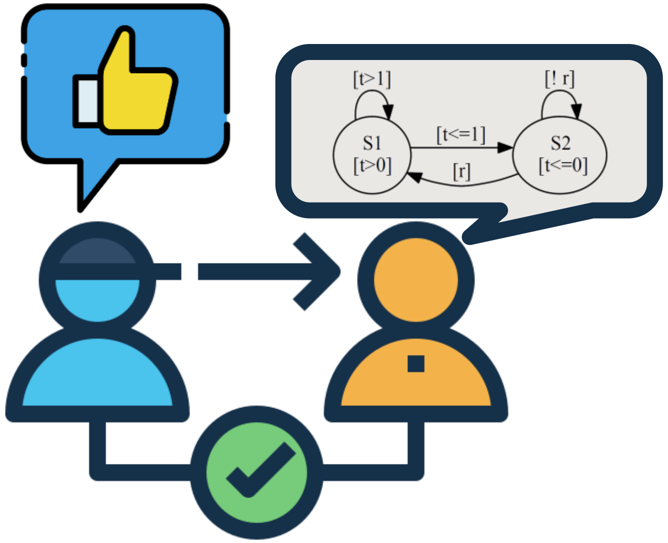
ICMC SÃO CARLOS USP

Universidade de São Paulo

UNIVERSITY OF LEICESTER

Context (Learning to Reuse)

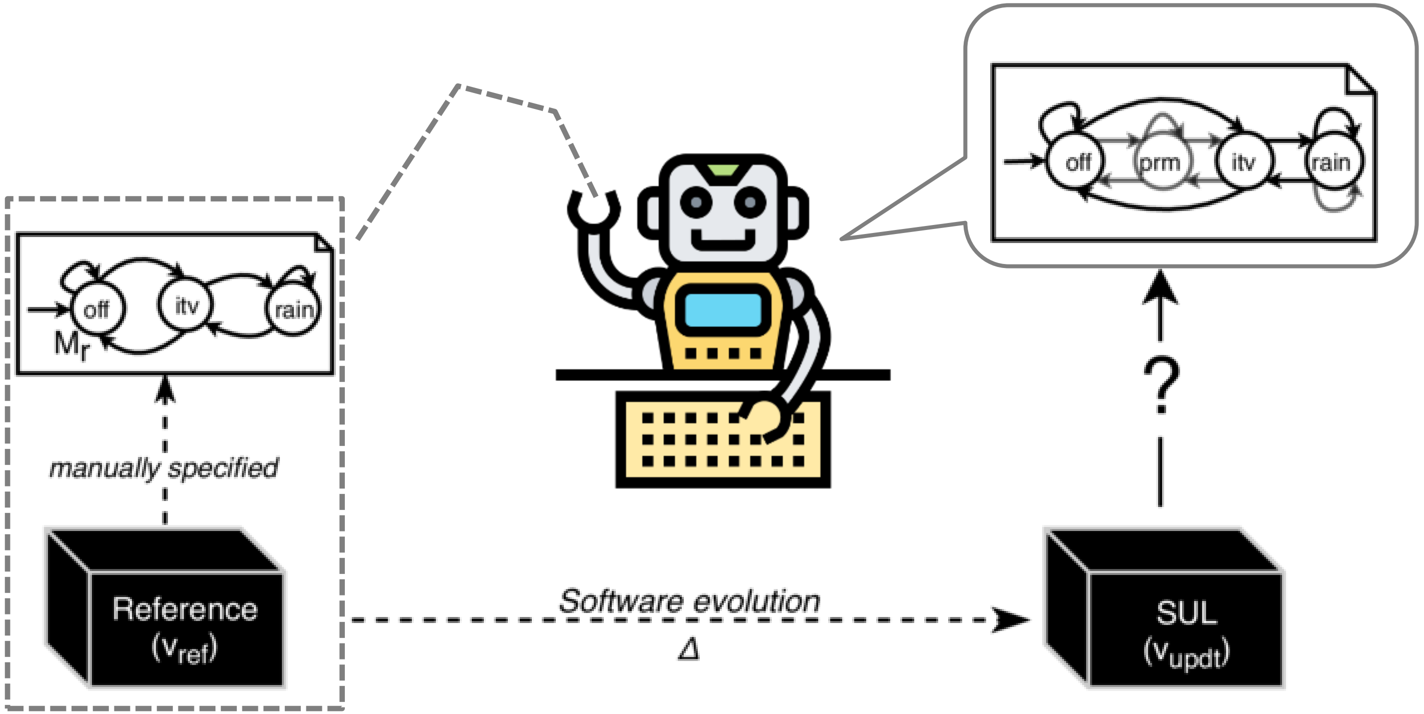
1. Software analysis is a *model-based* activity
   - Models stuck to engineers' minds
   - Formally denoted as explicit models



2. Software undergoes changes along the life-cycle
   - Evolution over-time (e.g., update, upgrade)
   - Models may become outdated

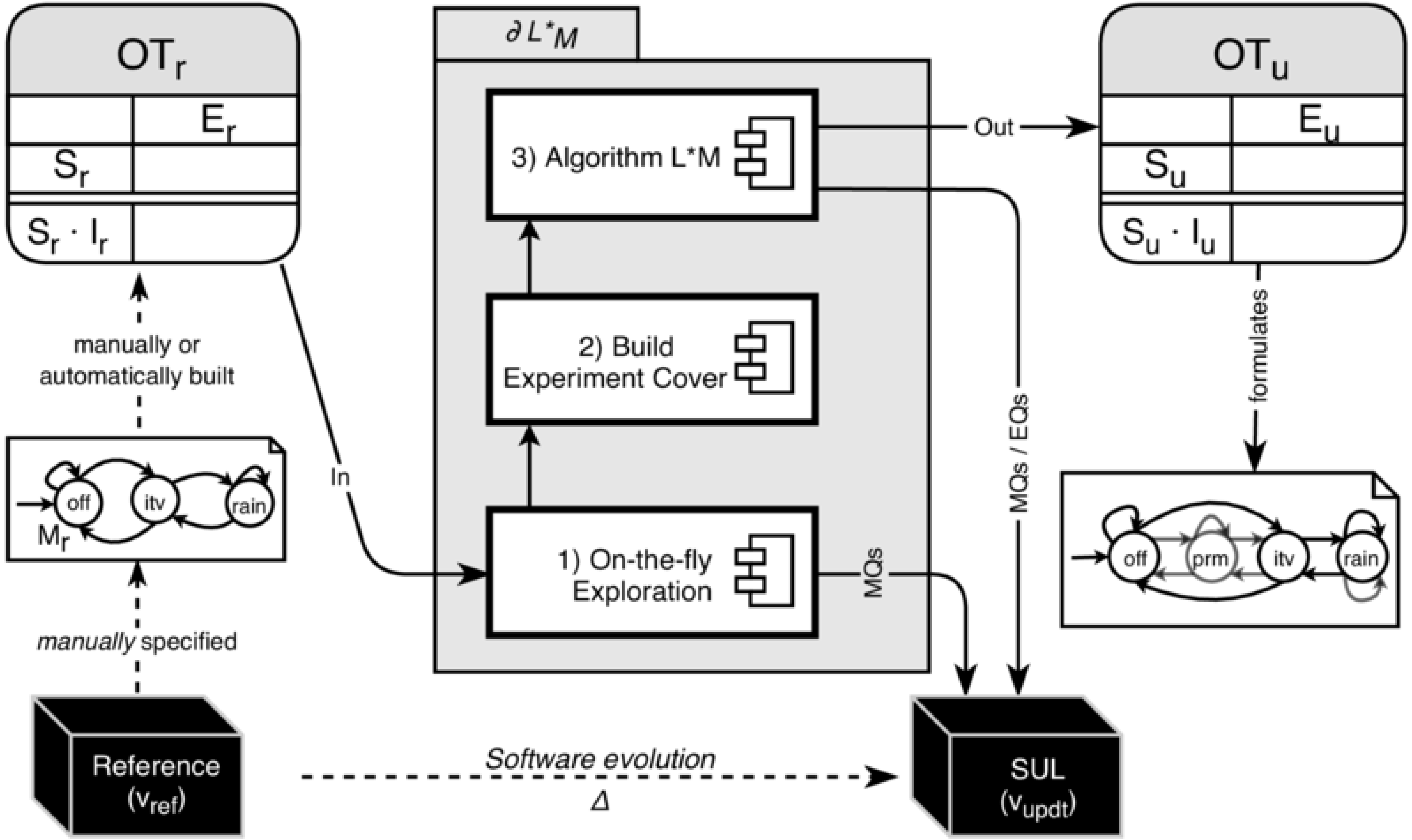
R. V. Binder, Testing Object-oriented Systems: Models, Patterns, and Tools. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1999.
N. Walkinshaw, 'Chapter 1 - Reverse-Engineering Software Behavior', in Advances in Computers, vol. 91, A. Memon, Ed. Elsevier, 2013, pp. 1–58.

Research Problem (Learning to Reuse)



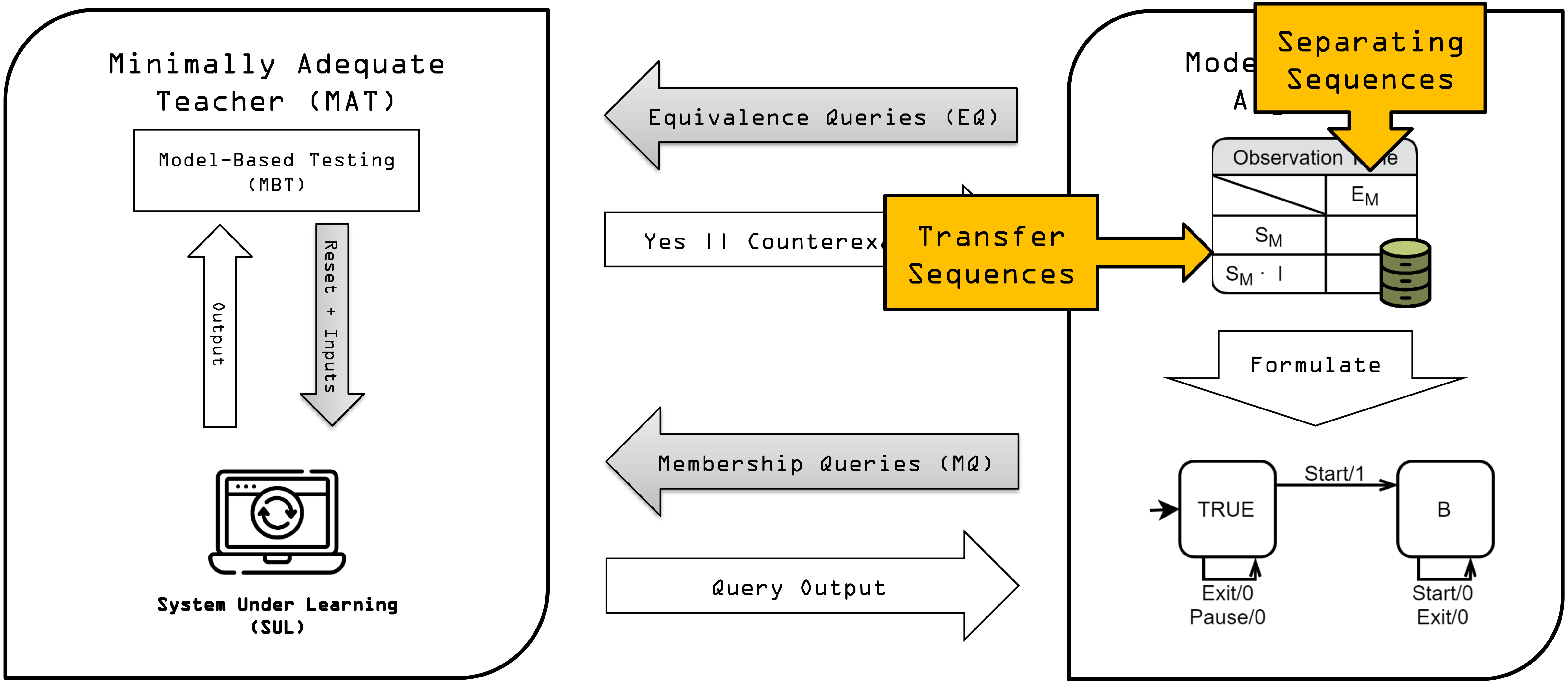
How can we ***efficiently*** *construct* ***behavioral models*** *from* ***evolving systems****?*

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



D. Angluin, 'Learning regular sets from queries and counterexamples', Information and Computation, vol. 75, no. 2, pp. 87–106, 1987
F. Vaandrager, 'Model Learning', Commun. ACM, vol. 60, no. 2, pp. 86–95, Jan. 2017

System Under Learning

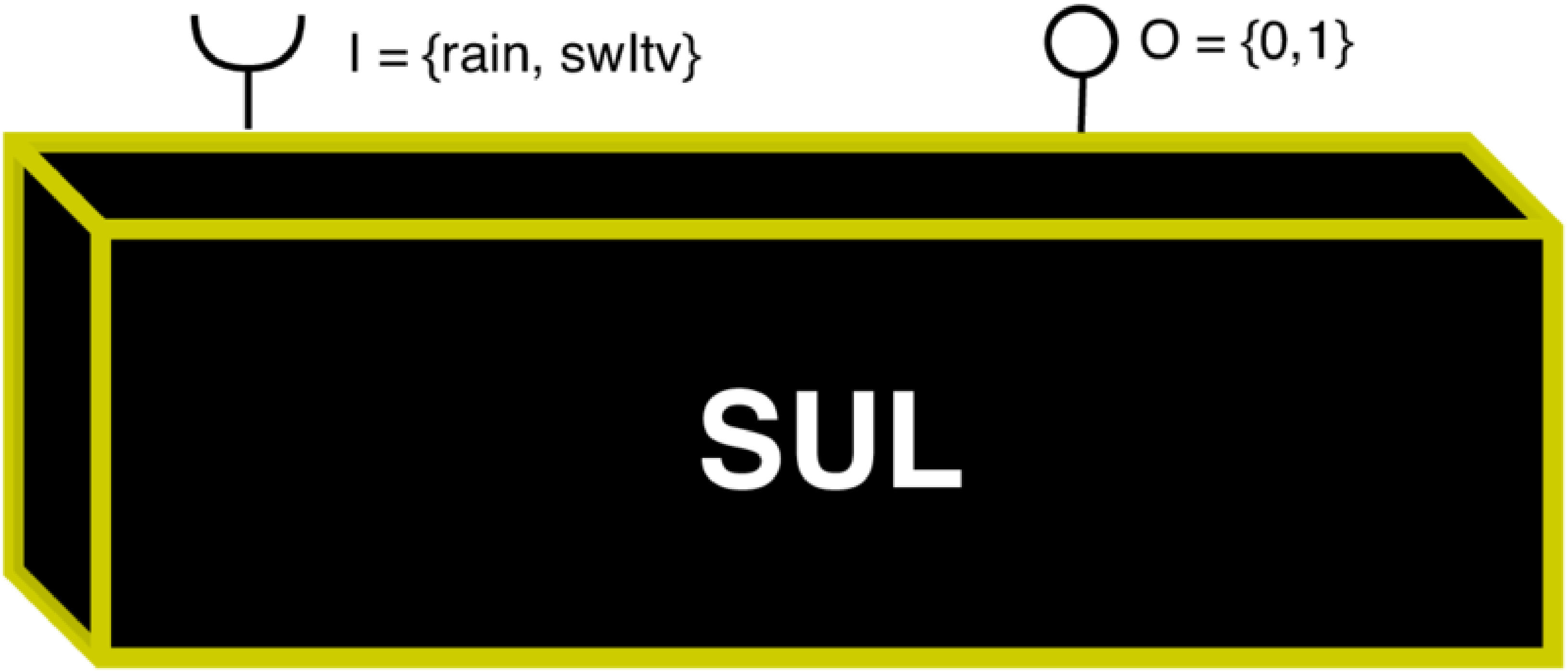


Figure: Windscreen wiper supporting intervaled and fast wiping

Model Learning

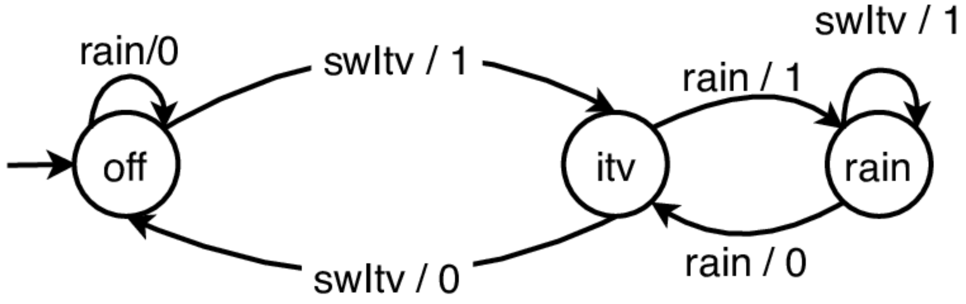


Figure: Final Hypothesis

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

Table: Final OT

$$\mathrm{EQ} = \mathrm{Yes}$$

What if the SUL evolves?

Model Learning for Evolving Systems

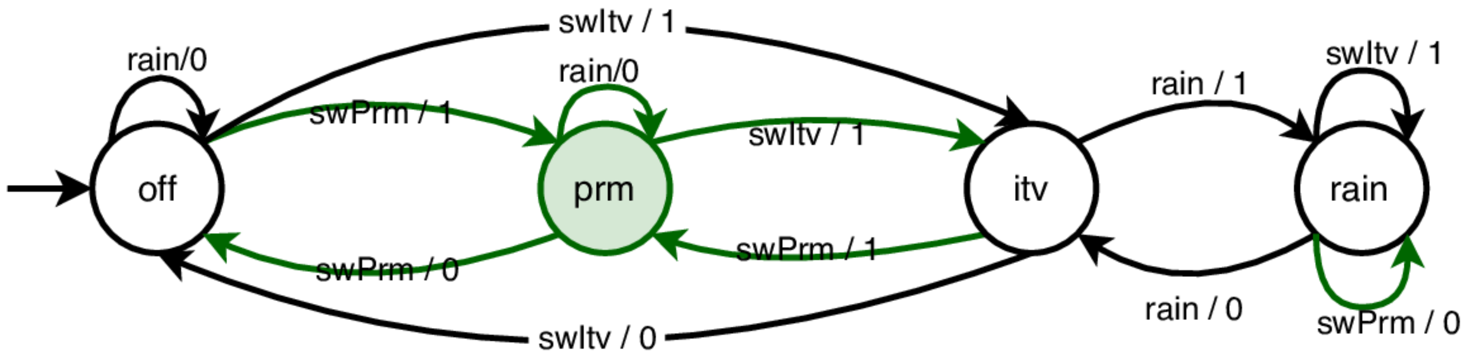
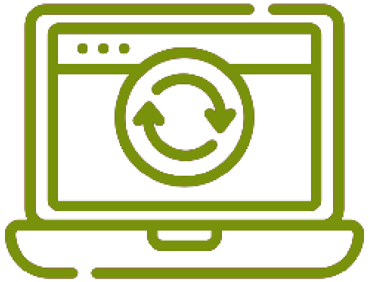


Figure: Windscreen wiper supporting intervaled and fast wiping + **permanent movement**

Model Learning for Evolving Systems

Minimally Adequate Teacher



Version 2

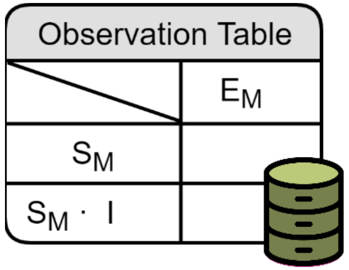
Equivalence Queries (EQ)

Yes || Counterexample

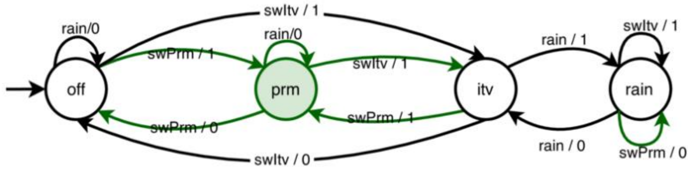
Membership Queries (MQ)

Query Output

Model Learning

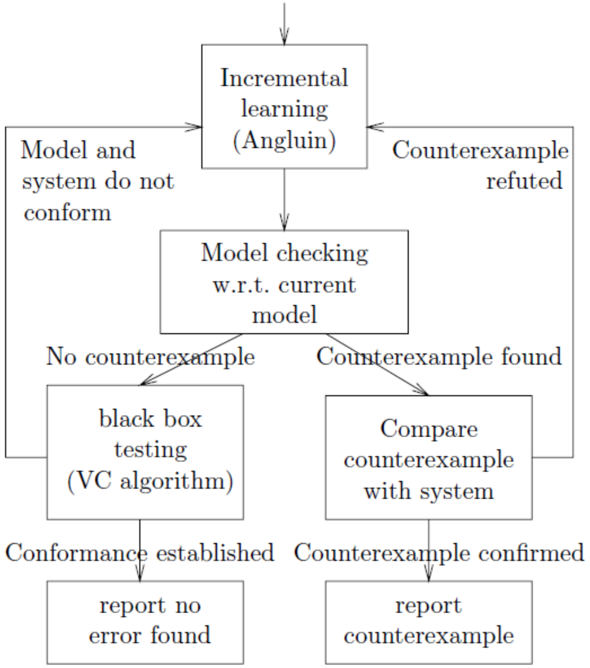


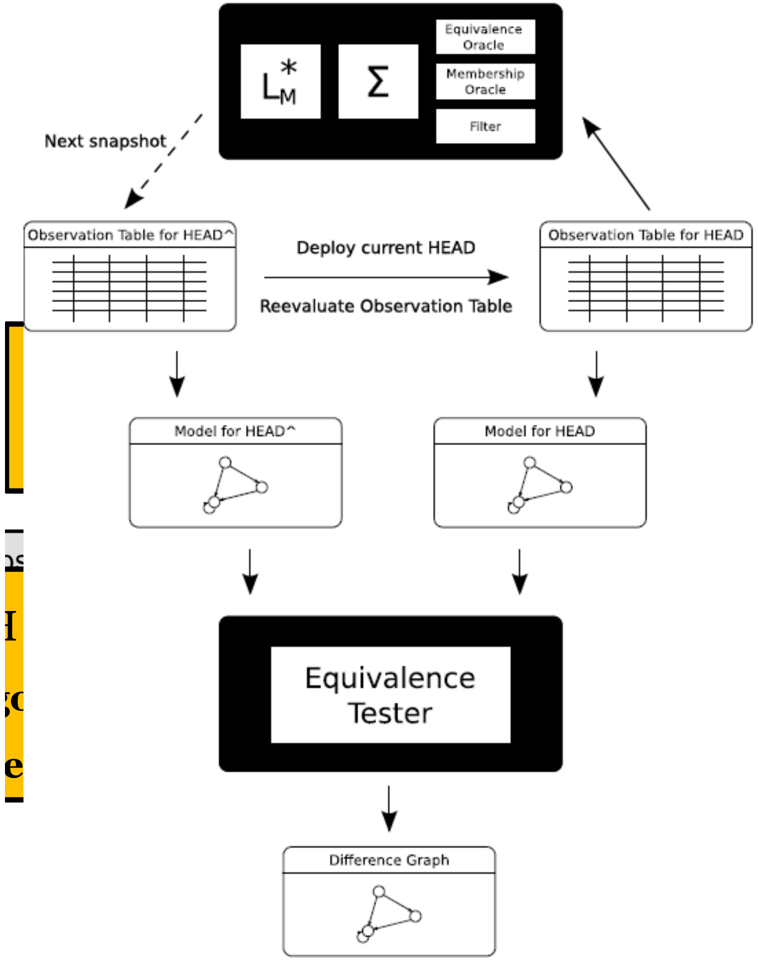
Formulate



Adaptive Model Learning

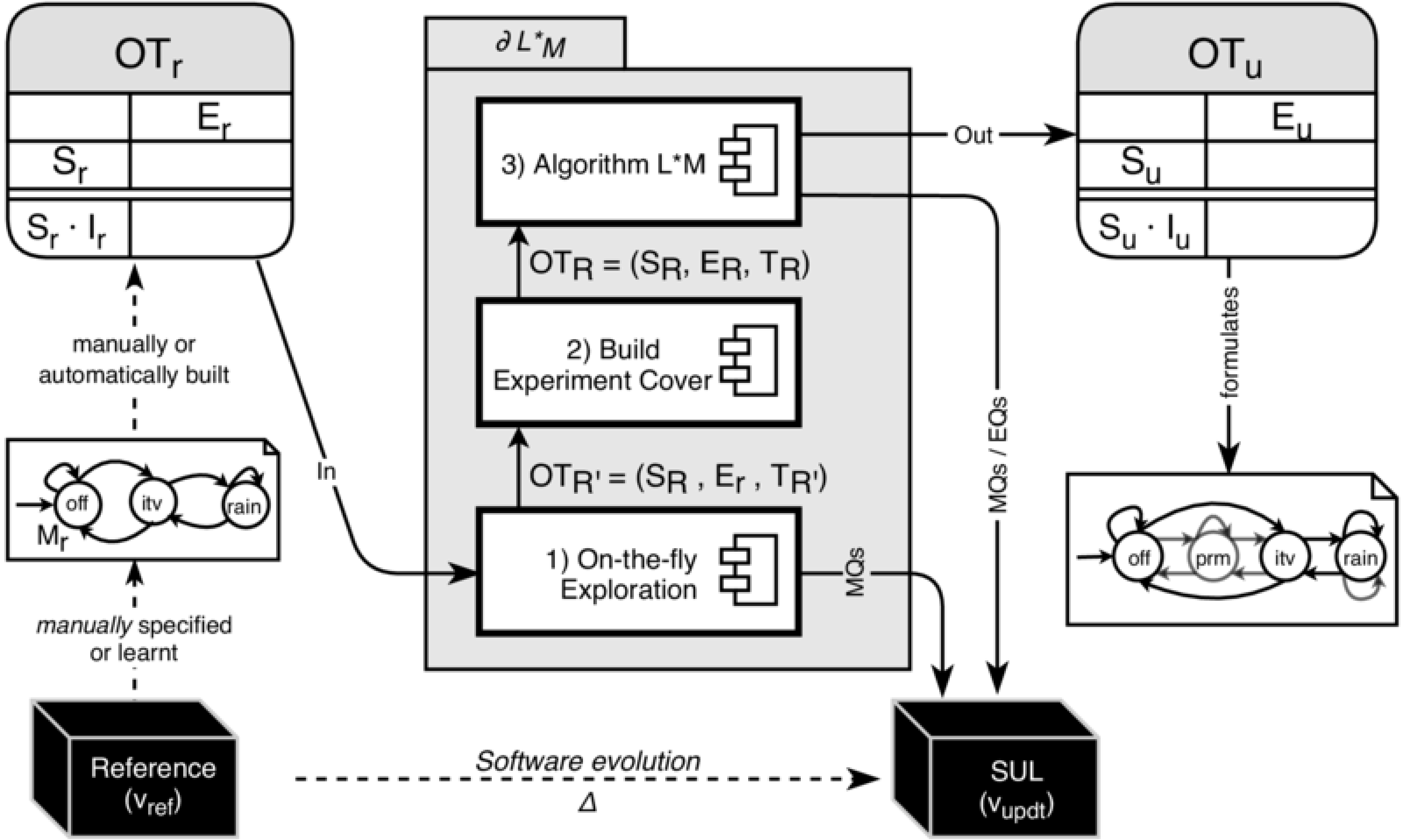
- **What:** Variant of model learning
- **How:** Reuse transfer/separating sequences from existing models
- **Why:** Speed up model learning
  - Find states maintained in newer versions
  - Reduce the time for model checking



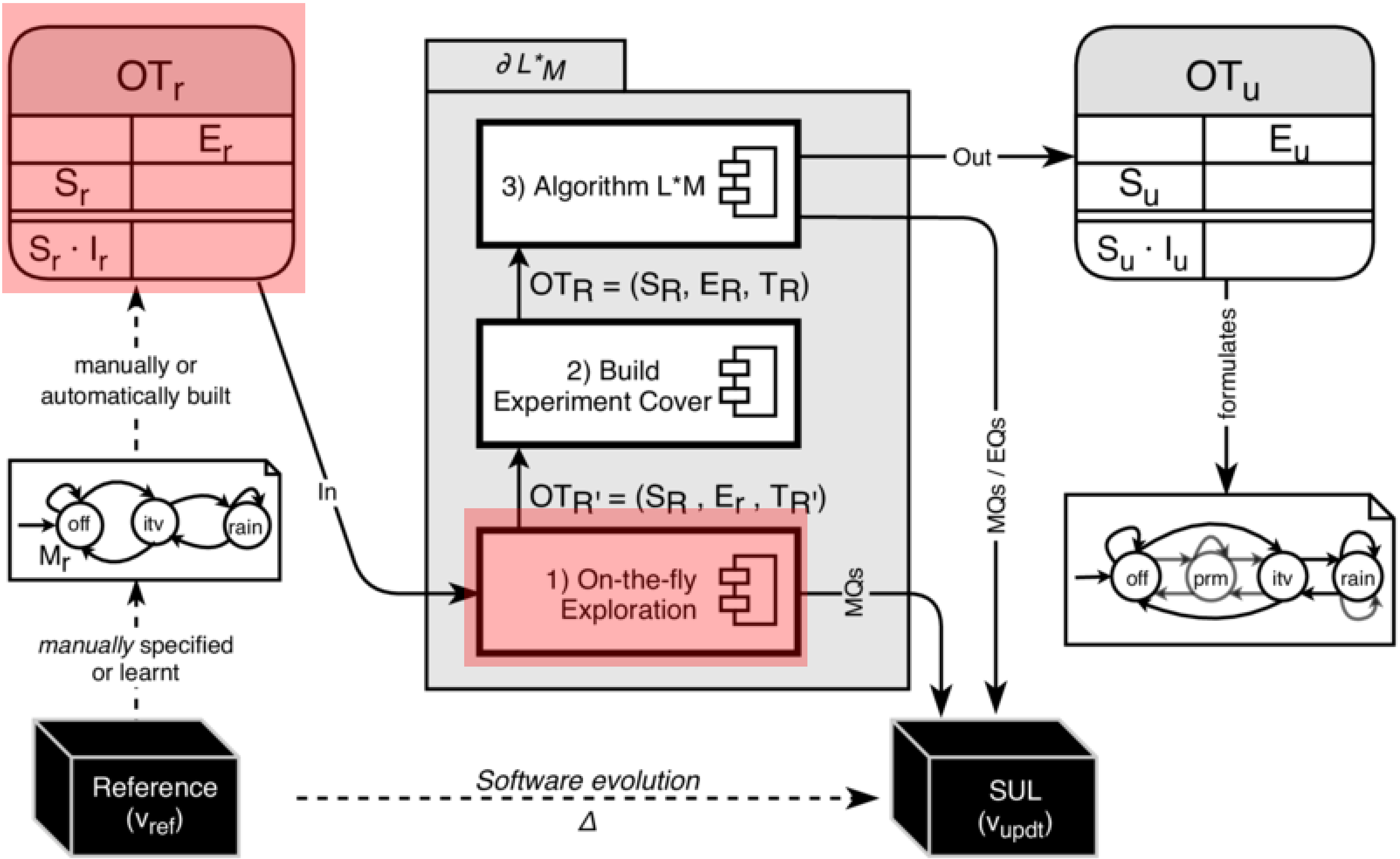


D. Huistra, J. Meijer, and J. van de Pol, ‘Adaptive Learning for Learn-Based Regression Testing’, in Formal Methods for Industrial Critical Systems, 2018
S. Windmüller, J. Neubauer, B. Steffen, F. Howar, and O. Bauer, ‘Active Continuous Quality Control’, in Proceedings of the CBSE 2013
A. Groce, D. Peled, and M. Yannakakis, ‘Adaptive Model Checking’, in Proceedings of the TACAS 2002

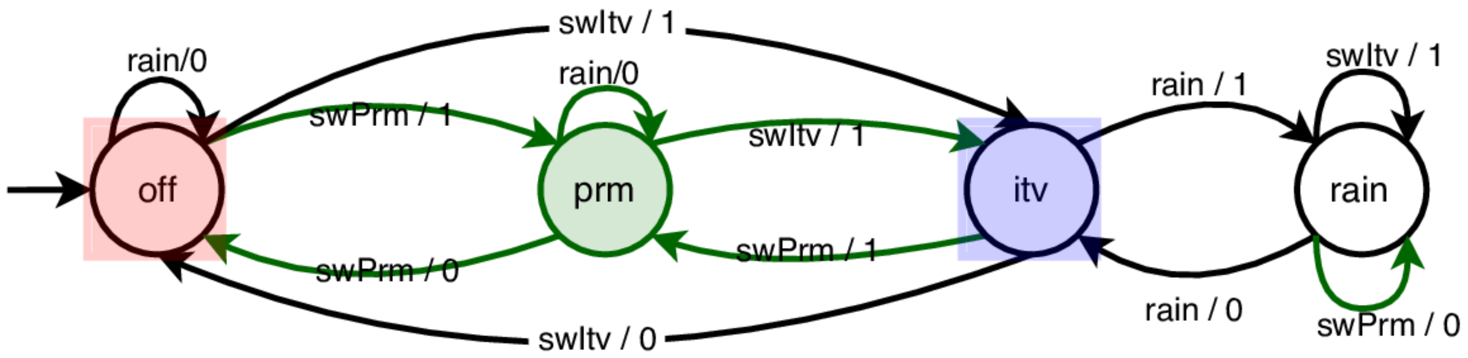
Partial-Dynamic L^*_M algorithm



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



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



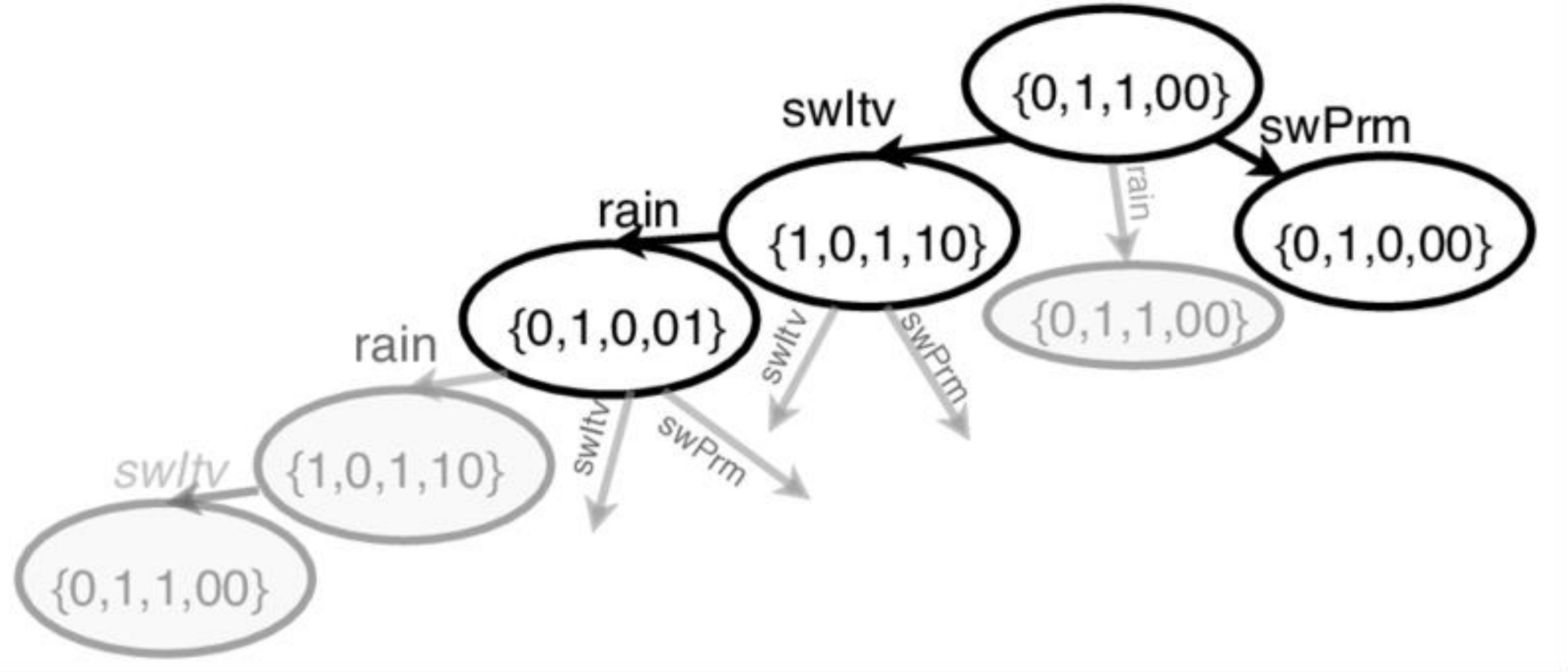
Let the sets of reused prefixes and suffixes be

$$S_r = \{\ \epsilon,\ swItv,\ swItv \cdot rain,\ swItv \cdot rain \cdot rain,\ swItv \cdot rain \cdot rain \cdot swItv,\ rain\ \}$$

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

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

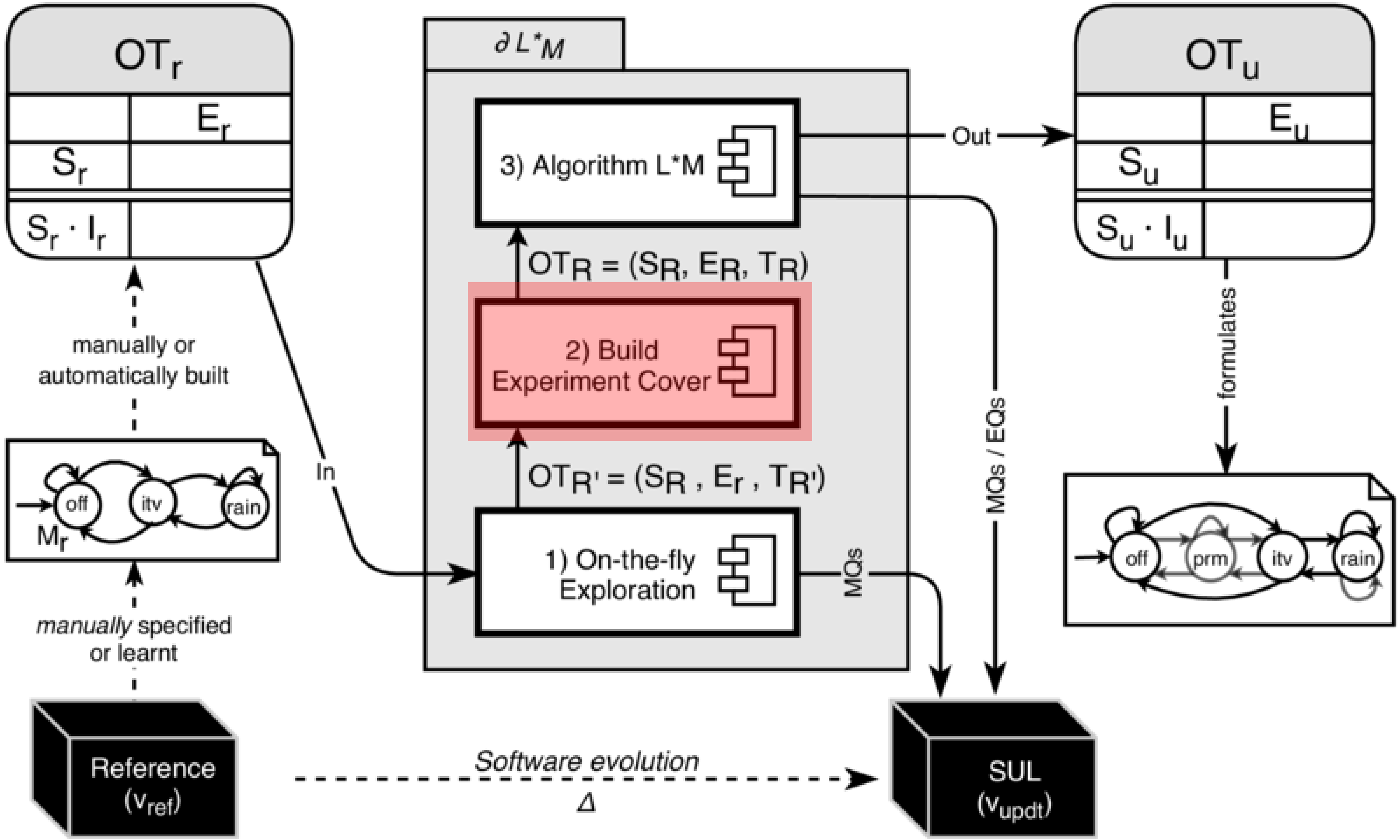
1) On-the-fly exploration of the reused OT



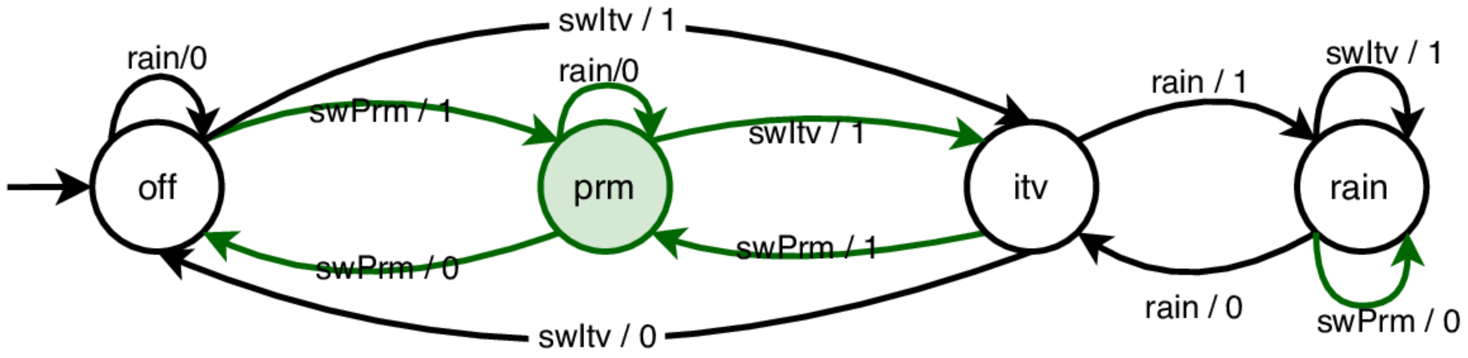
On-the-fly exploration of the **tree representation** of the set of **transfer sequences**

* On-the-fly: We alternate between the tree traversal steps and MQs

2) Build the experiment cover tree



2) Build the experiment cover tree



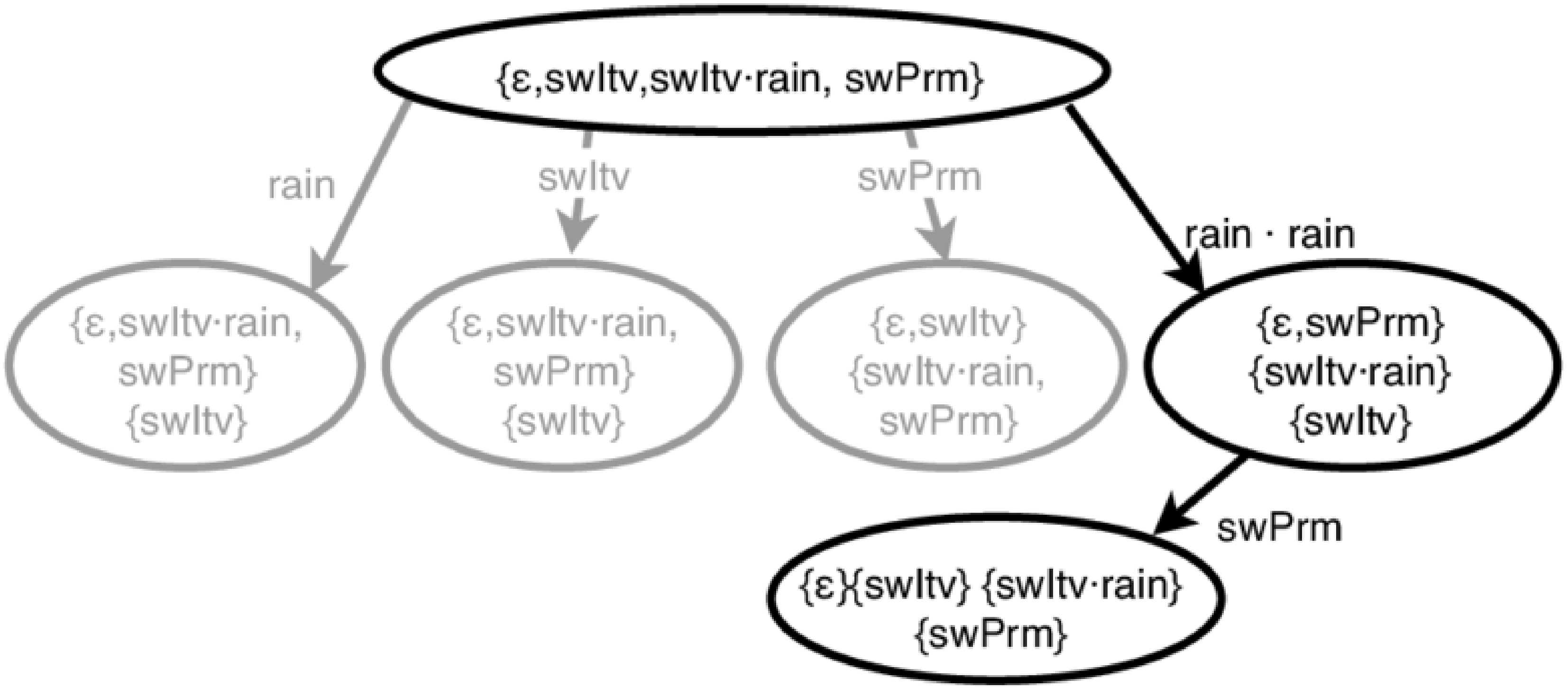
Let the sets of prefixes and suffixes be

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

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

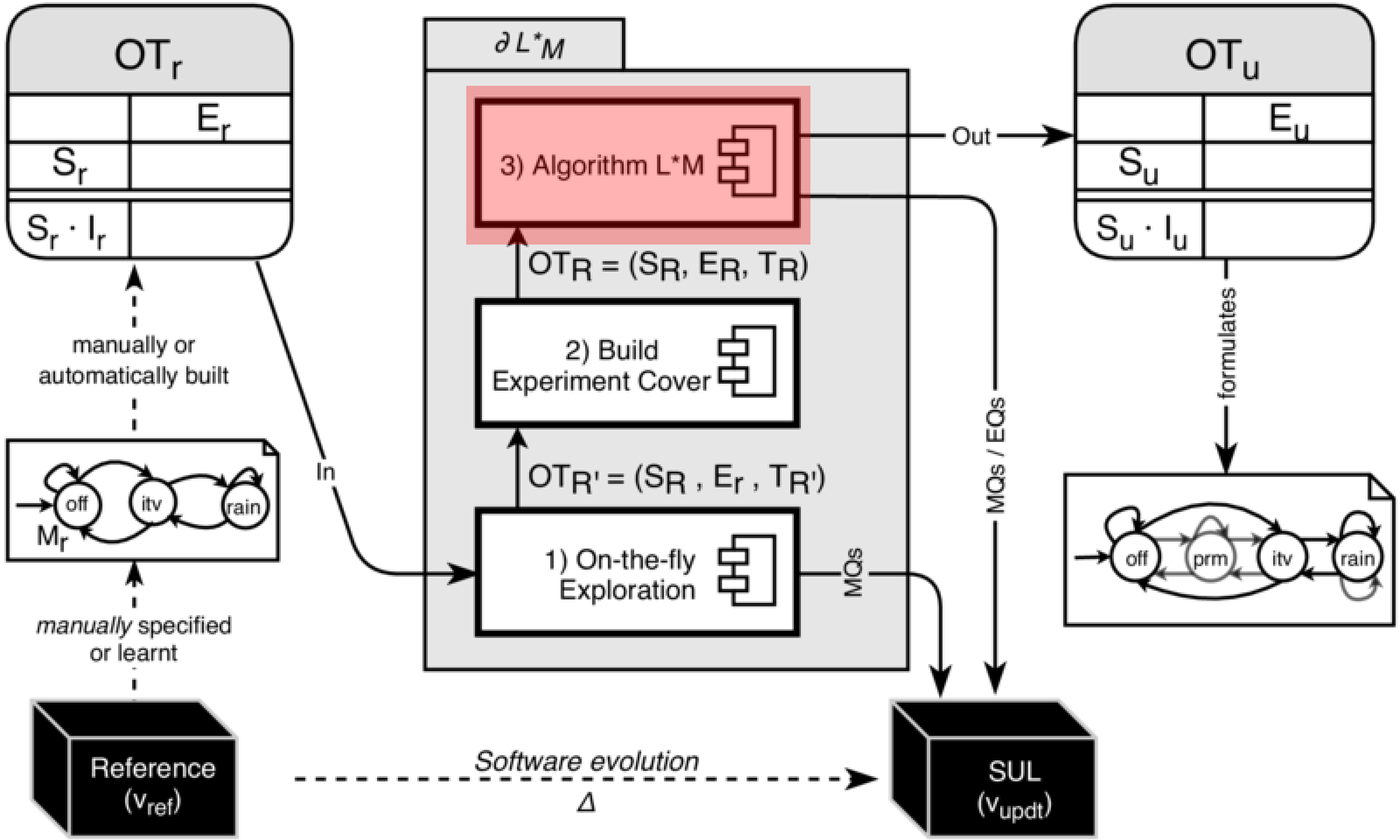
Goal: Find a smaller subset $E_R \subseteq E_r$ of representative separating sequences.

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 ∂L^*_M



Empirical Evaluation

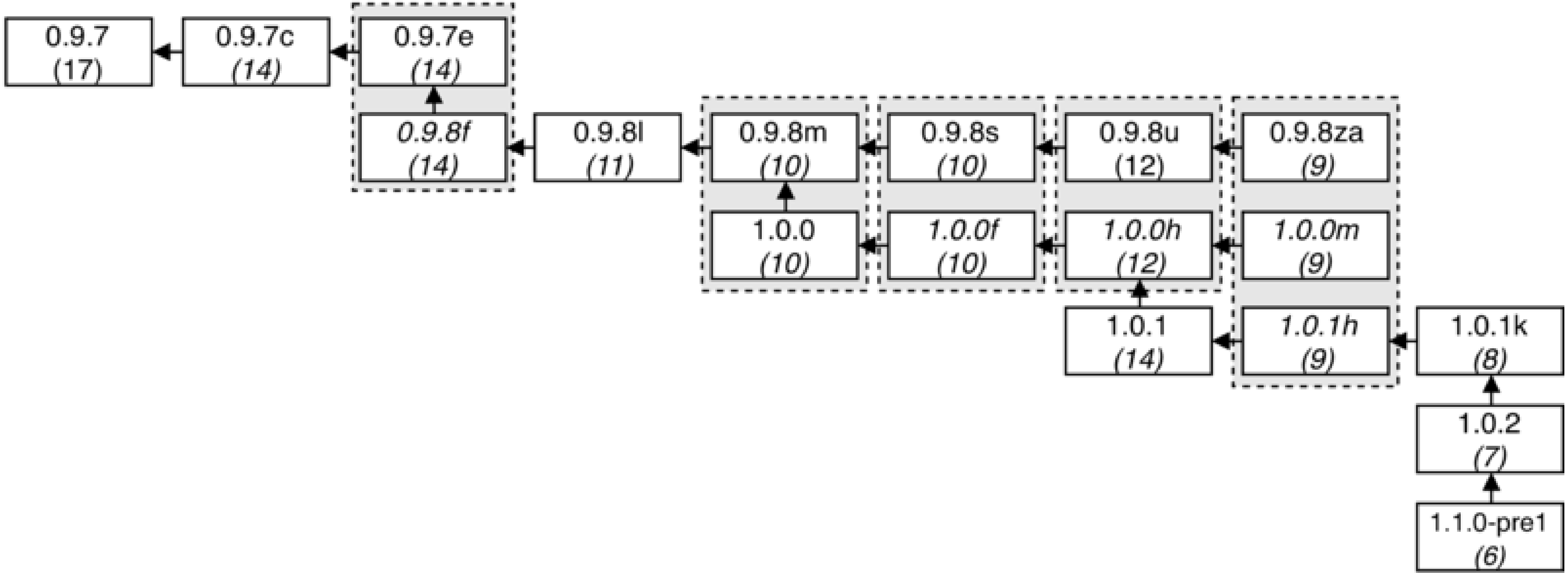
Empirical Evaluation (Research Questions)

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?

D. Huistra, J. Meijer, and J. van de Pol, 'Adaptive Learning for Learn-Based Regression Testing', in Formal Methods for Industrial Critical Systems, 2018
S. Windmüller, J. Neubauer, B. Steffen, F. Howar, and O. Bauer, 'Active Continuous Quality Control', in Proceedings of the CBSE 2013
A. Groce, D. Peled, and M. Yannakakis, 'Adaptive Model Checking', in Proceedings of the TACAS 2002
J. de Ruiter, 'A Tale of the OpenSSL State Machine: A Large-Scale Black-Box Analysis', in Secure IT Systems, vol. 10014, B. B. Brumley and J. Röning, Eds. Cham: Springer, 2016,
M. Isberner, F. Howar, and B. Steffen, 'The Open-Source LearnLib', in CAV 2015

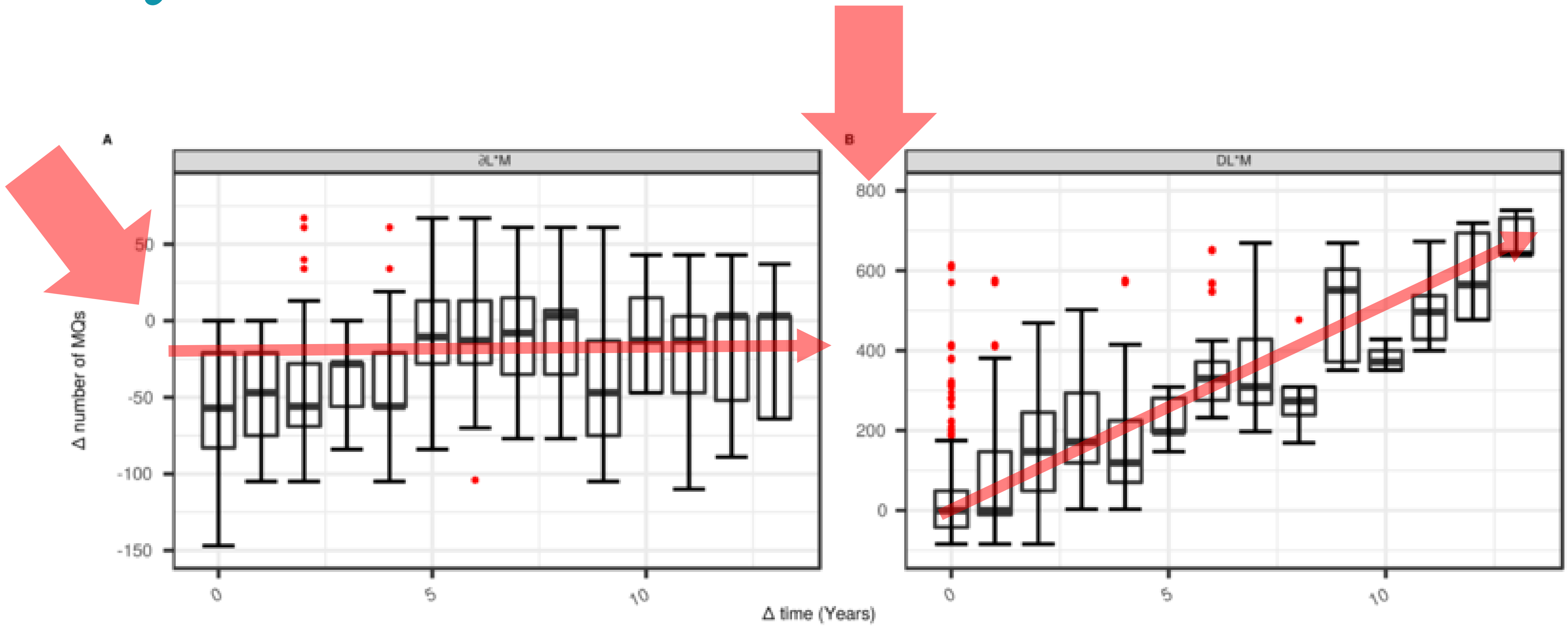
Subject Systems



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

OpenSSL Foundation, Inc., 'OpenSSL Releases at Github', 2018. https://github.com/openssl/openssl/releases.
J. de Ruiter, 'A Tale of the OpenSSL State Machine: A Large-Scale Black-Box Analysis', in Secure IT Systems, vol. 10014, B. B. Brumley and J. Röning, Eds. Cham: Springer, 2016,

Analysis of Results (Average number of MQs)



The temporal distance between versions did not affect the performance of the ∂L^*_M algorithm

The ∂L^*_M algorithm required **fewer MQs** than the DL^*_M algorithm

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

Summary (Learning to Reuse)

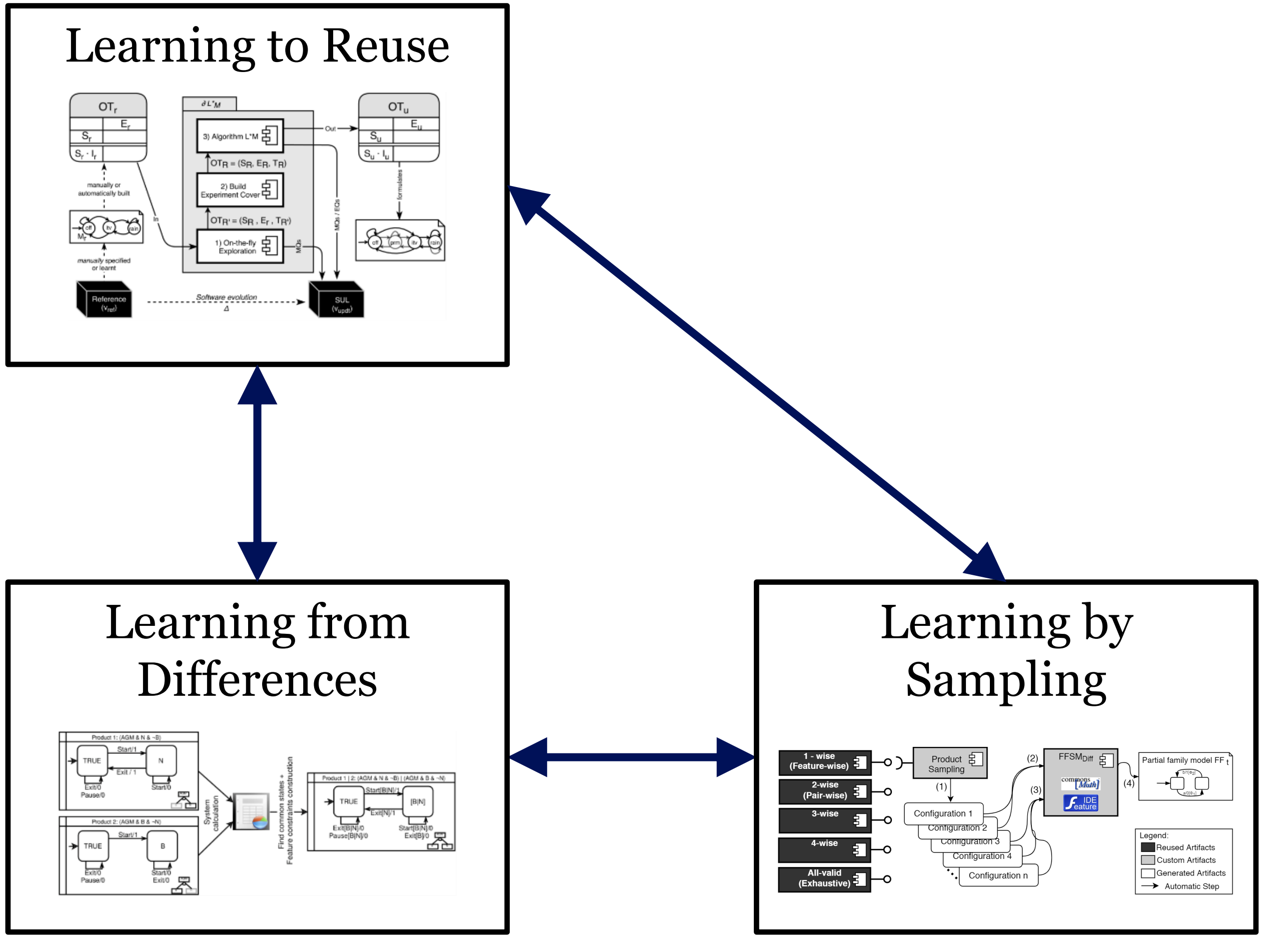
The state-of-the-art adaptive learning algorithms…

1. More sensitive to software evolution!

The ∂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

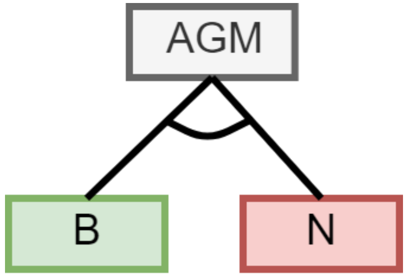
ICMC USP SÃO CARLOS

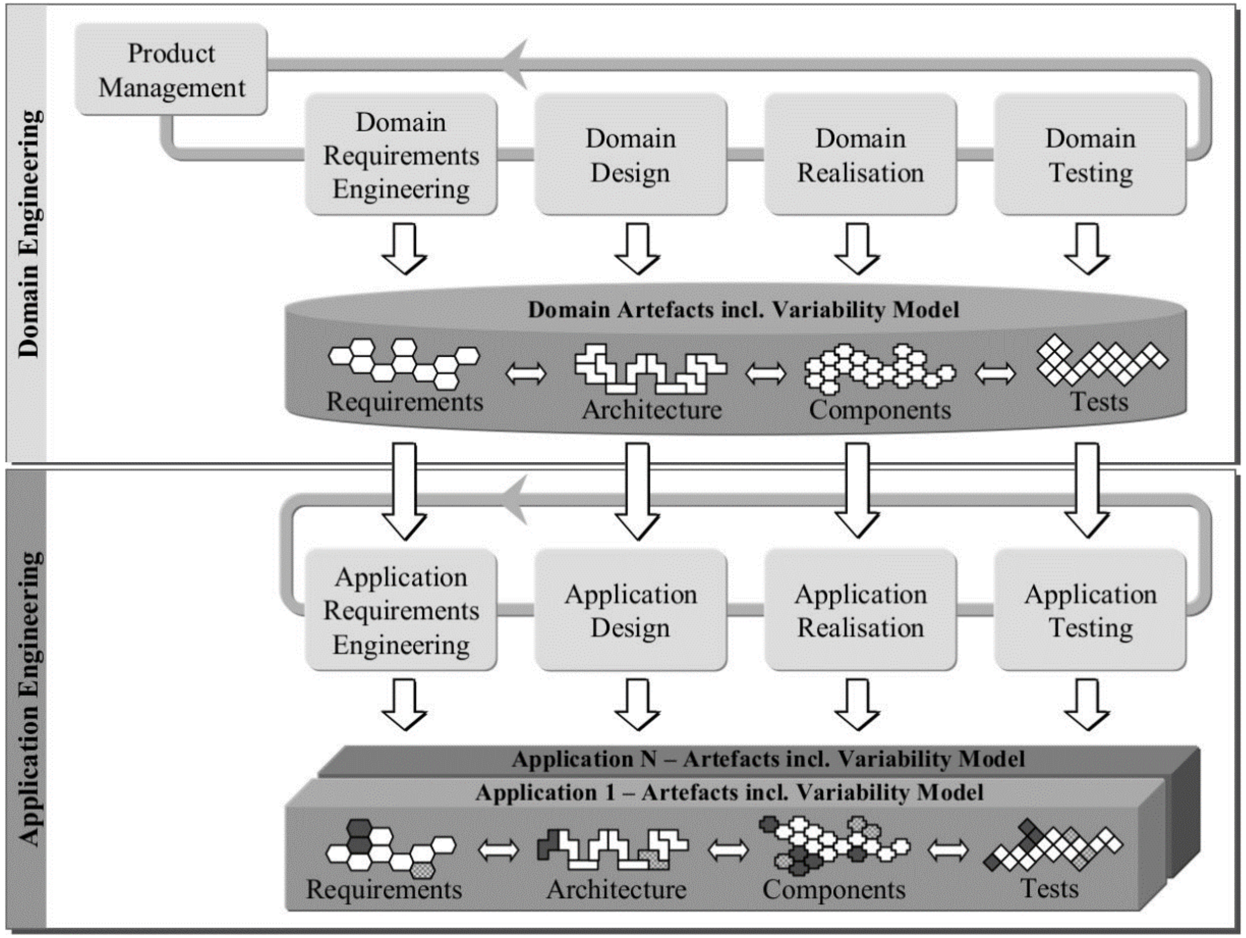
Universidade de São Paulo

UNIVERSITY OF LEICESTER

Context (Learning from Difference)

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



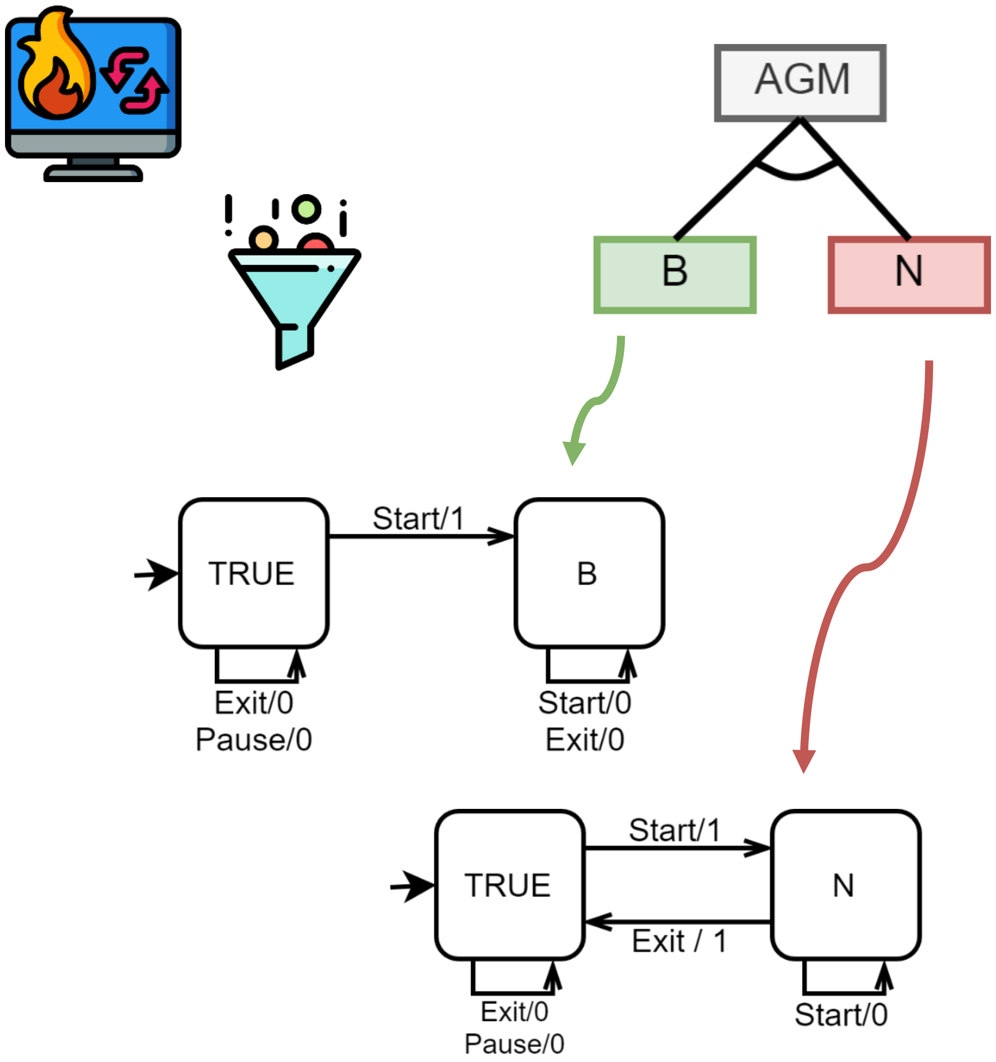


Context (Learning from Difference)

- 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)



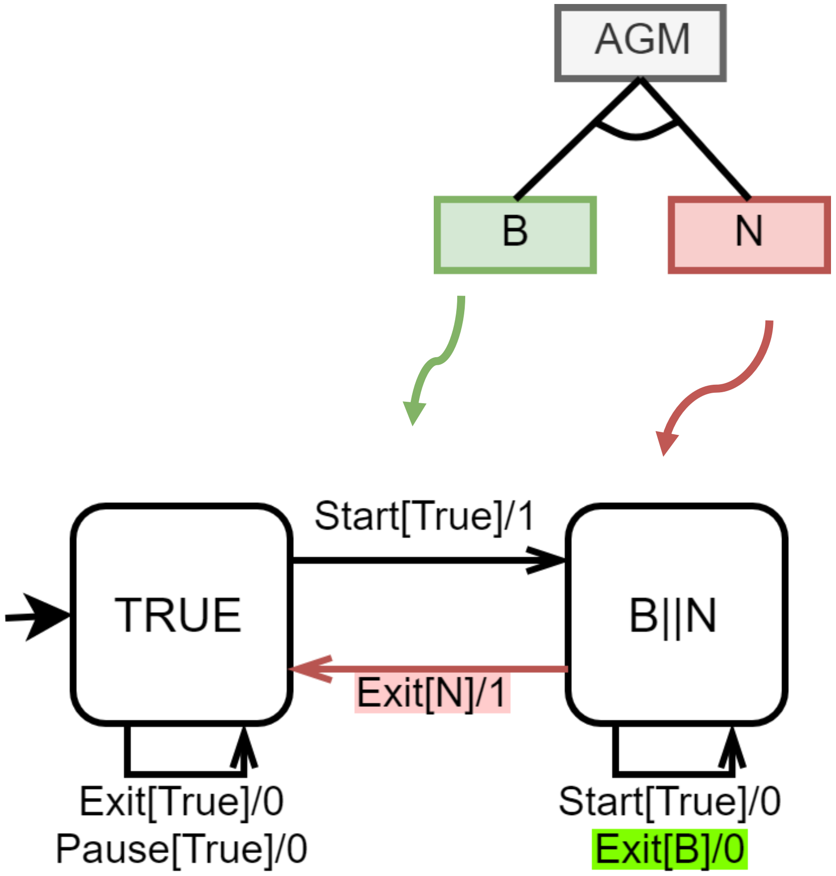
T. Thüm, S. Apel, C. Kästner, I. Schaefer, and G. Saake, ‘A Classification and Survey of Analysis Strategies for Software Product Lines’, ACM Comput. Surv., vol. 47, no. 1, p. 6:1–6:45, Jun. 2014
S. Oster, ‘Feature Model-based Software Product Line Testing’, PhD Thesis, Technische Universität, Darmstadt, 2012.

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)

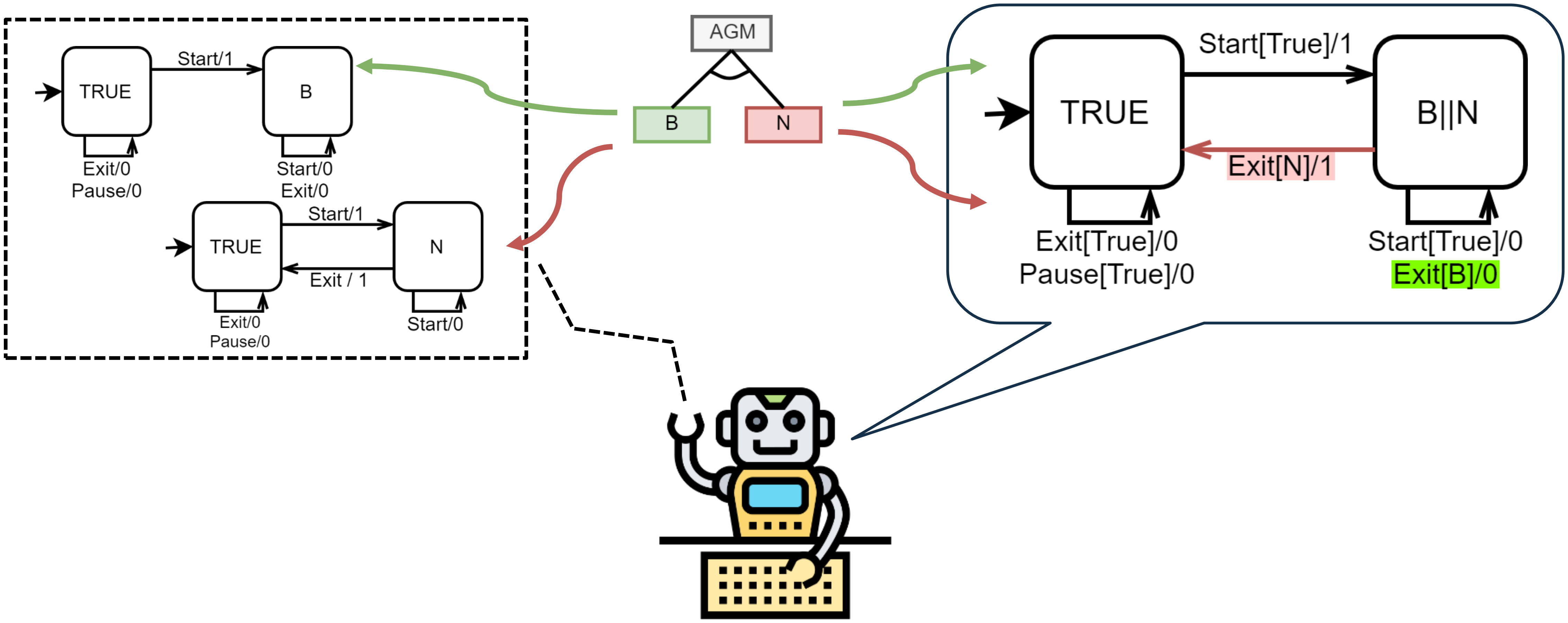
ISSUES

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



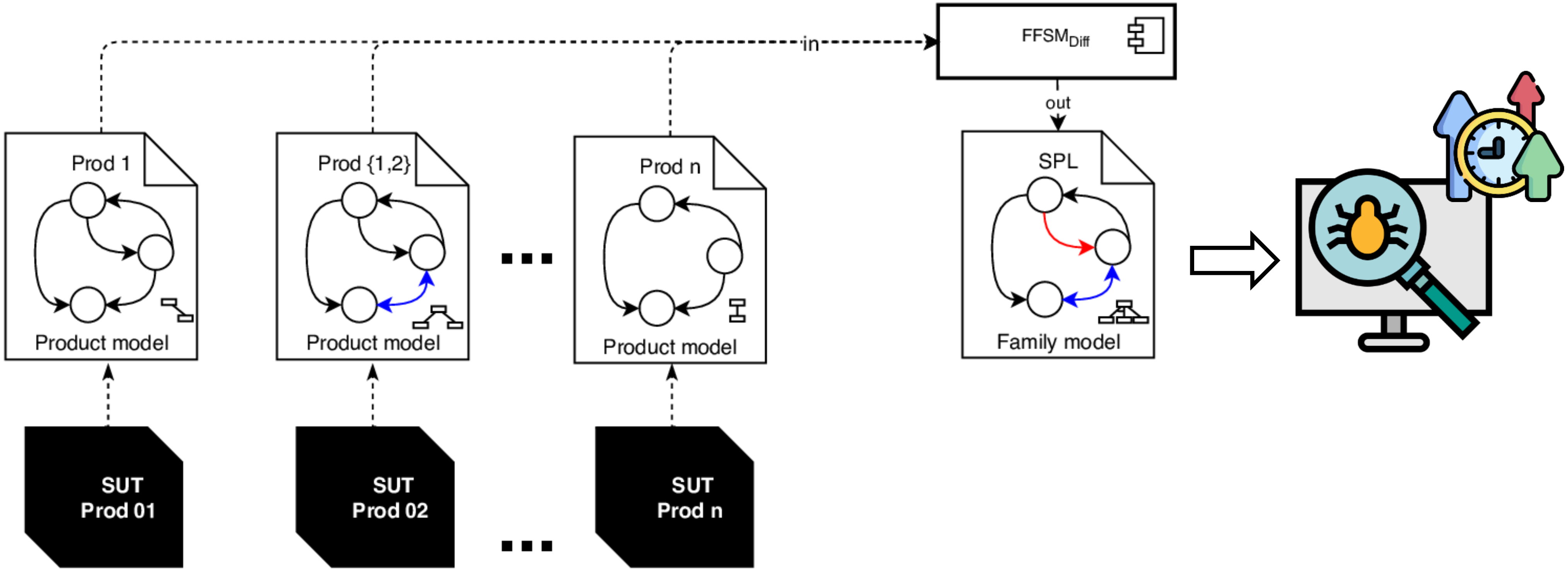
T. Thüm, S. Apel, C. Kästner, I. Schaefer, and G. Saake, 'A Classification and Survey of Analysis Strategies for Software Product Lines', ACM Comput. Surv., vol. 47, no. 1, p. 6:1–6:45, Jun. 2014
S. Oster, 'Feature Model-based Software Product Line Testing', PhD Thesis, Technische Universität, Darmstadt, 2012.
V. Hafemann Fragal, A. Simao, and M. R. Mousavi, 'Validated Test Models for Software Product Lines: Featured Finite State Machines', in `FACS 2016.

Research Problem (Learning from Difference)



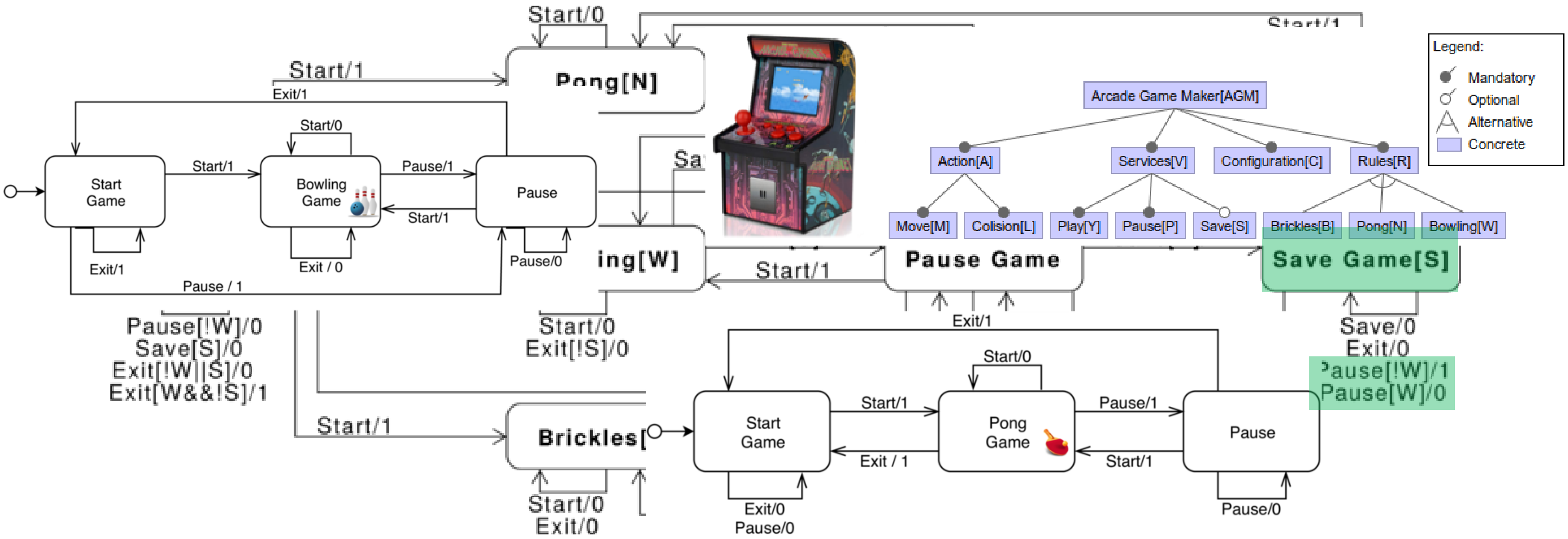
How can we ***leverage*** *the concept of* ***model learning*** *to the task of* ***behavioral variability modeling****?*

Contribution (Learning from Difference)



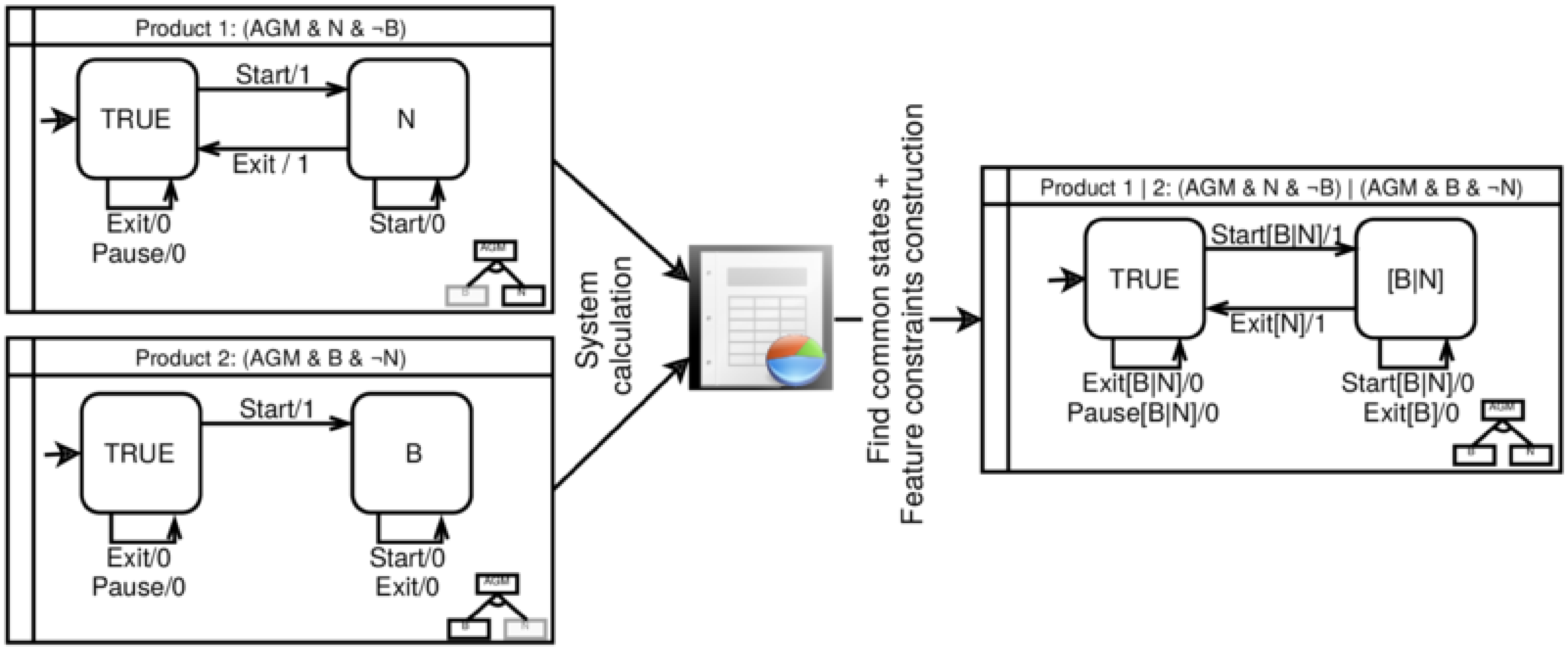
The FFSM Difference (FFSM $_{Diff}$) algorithm for **learning succinct family models** from individual product specifications of **software product lines**

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

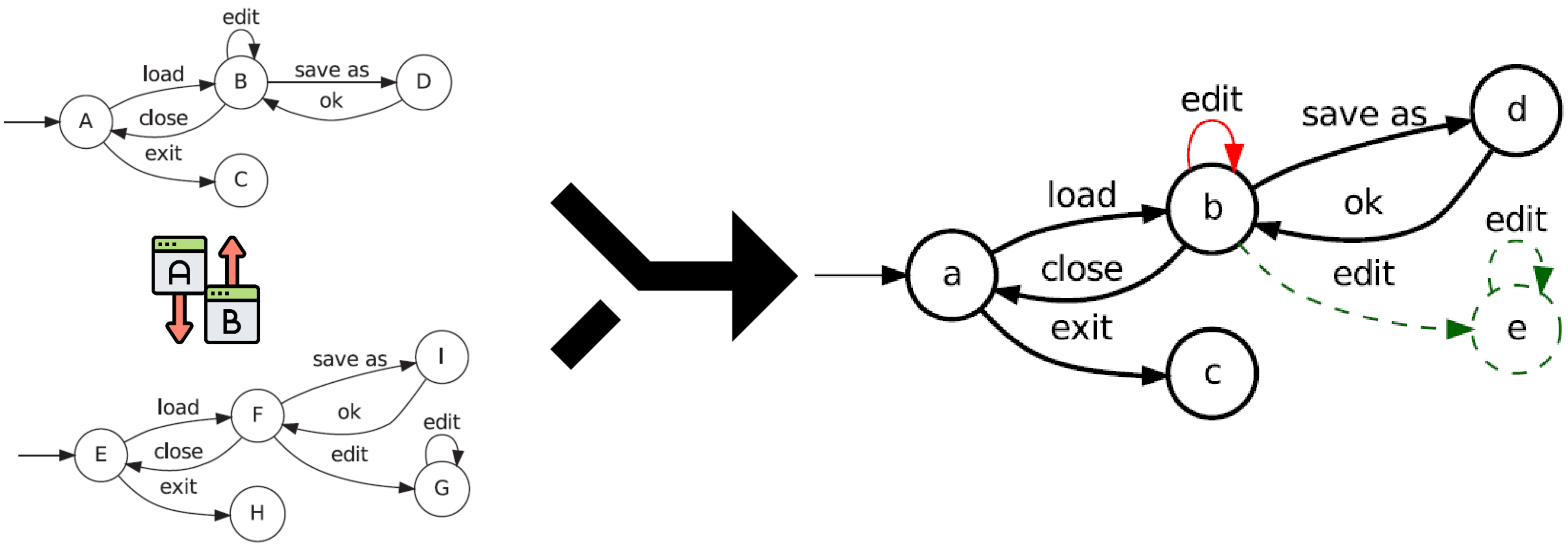
FFSM Difference (FFSM $_{Diff}$)



The FFSM $_{Diff}$ 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

V. Hafemann Fragal, A. Simao, and M. R. Mousavi, 'Validated Test Models for Software Product Lines: Featured Finite State Machines', in `FACS 2016.
N. Walkinshaw and K. Bogdanov, 'Automated Comparison of State-Based Software Models in Terms of Their Language and Structure', ACM TOSEM, vol. 22, 2013
D. Benavides, S. Segura, and A. Ruiz-Cortés, 'Automated analysis of feature models 20 years later: A literature review', Information Systems, vol. 35, no. 6, pp. 615–636, 2010

State-based model comparison (LTS $_{Diff}$ algorithm)



Comparing the Structures of Two State Machines of a Text Editor

State-based model comparison (LTS $_{Diff}$ algorithm)

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

Figure: Global similarity score [4]

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', ACM TOSEM, vol. 22, 2013

State-based model comparison (LTS $_{Diff}$ 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 Diff algorithm)

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

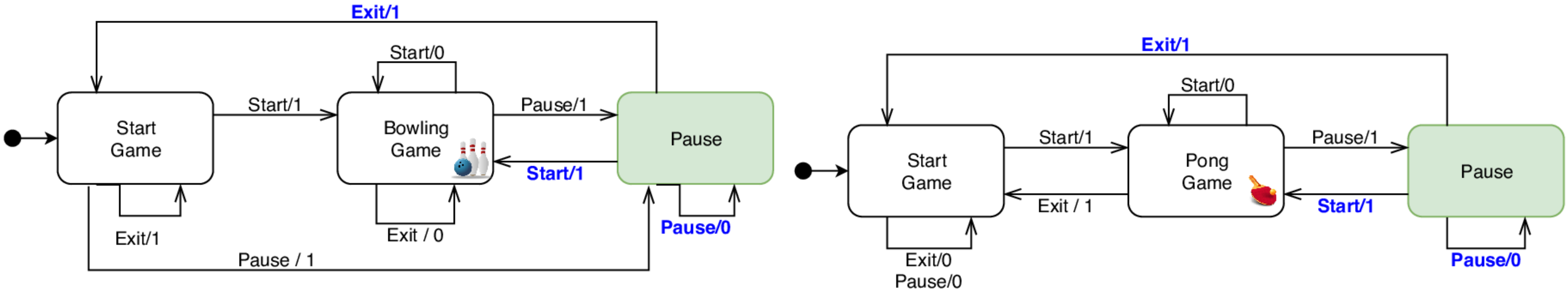


Figure: Two examples of product FSMs and their similarity scores

The FFSM $_{Diff}$ algorithm

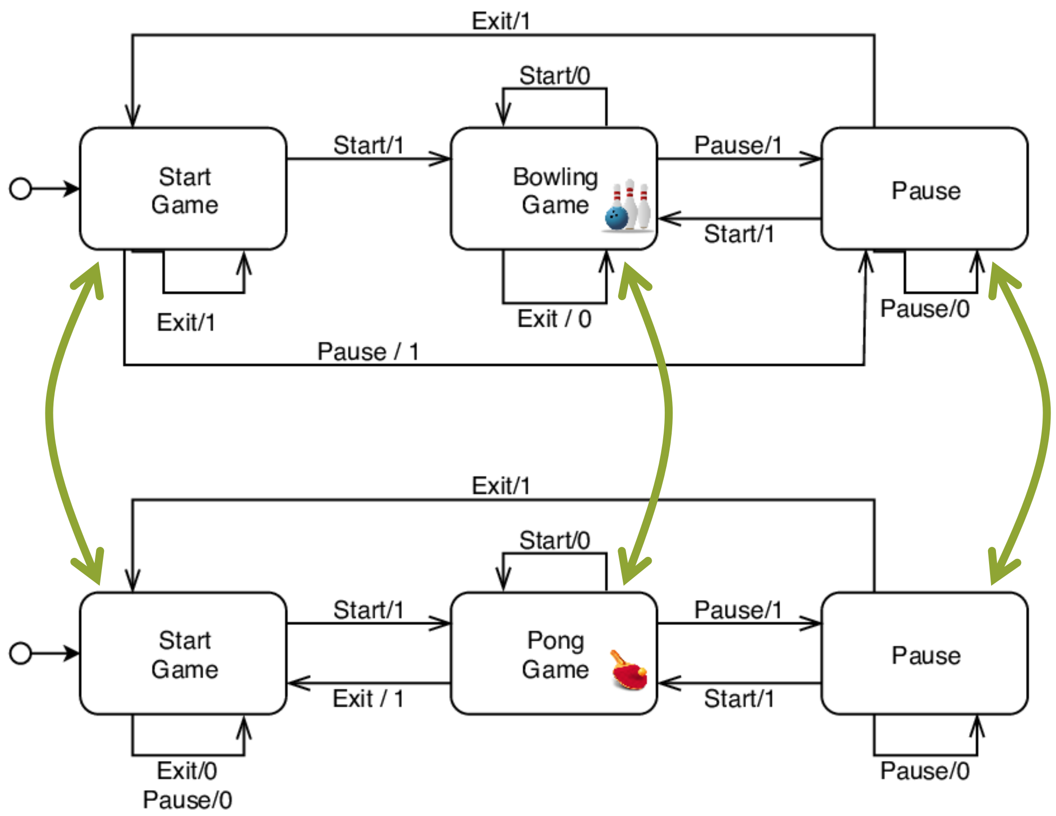


Figure: Two examples of product FSMs

pair(St, St) = 0.12

pair(St, Po) = 0.29

pair(St, Pa) = 0.28

pair(Bo, St) = 0.11

pair(Bo, Po) = 0.31

pair(Bo, Pa) = 0

pair(Pa, St) = 0.29

pair(Pa, Po) = 0.11

pair(Pa, Pa) = 0.58

Figure: Pairwise state similarity

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

The FFSM $_{Diff}$ algorithm

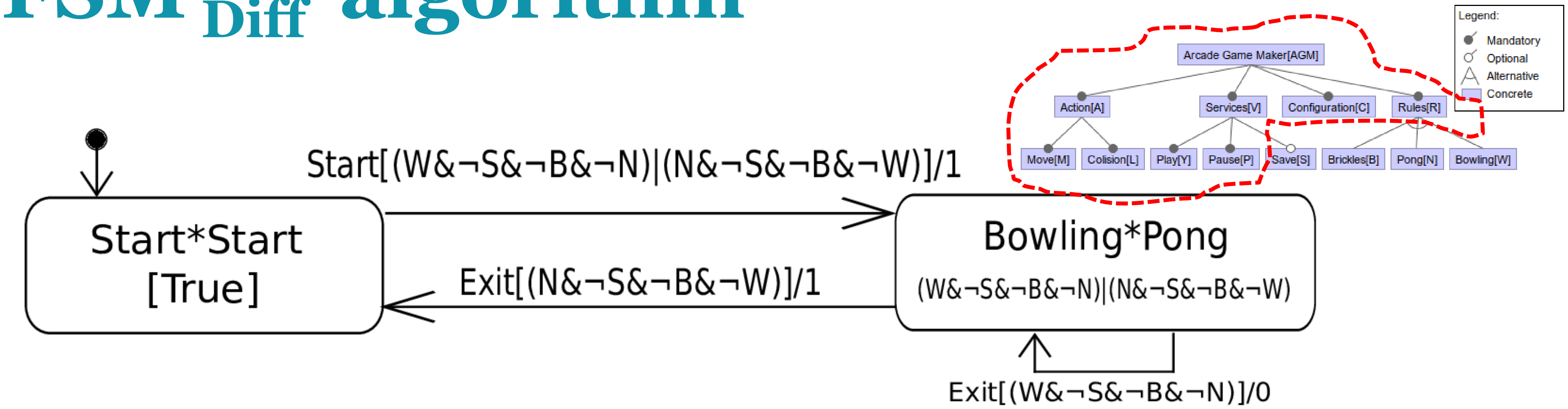


Figure: Fragment of the FFSM learnt from two products of the AGM SPL.

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**

V. Hafemann Fragal, A. Simao, and M. R. Mousavi, 'Validated Test Models for Software Product Lines: Featured Finite State Machines', in `FACS 2016.
D. Benavides, S. Segura, and A. Ruiz-Cortés, 'Automated analysis of feature models 20 years later: A literature review', Information Systems, vol. 35, no. 6, pp. 615–636, 2010

Empirical Evaluation

Empirical Evaluation

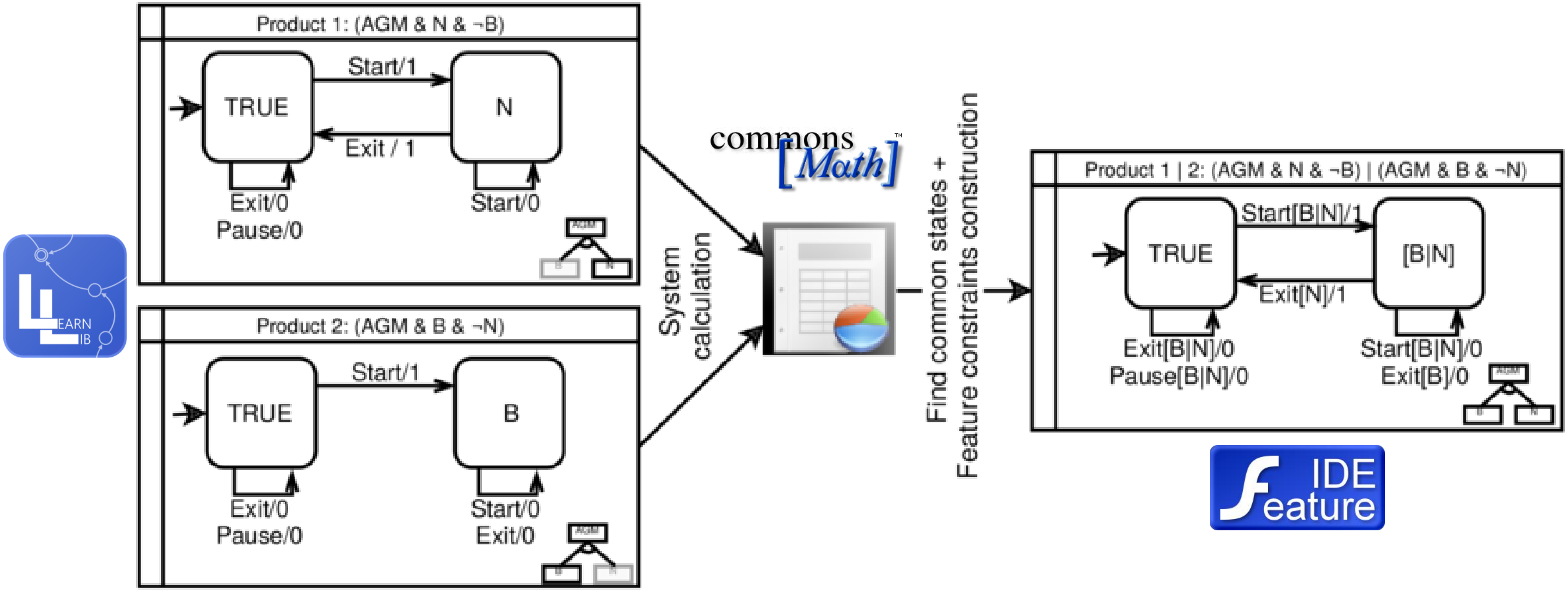
RQ1) Is our approach effective in learning succinct family models compared to the total size of the product pairs under learning?

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?

C. Henard, et al., 'Bypassing the Combinatorial Explosion: Using Similarity to Generate and Prioritize T-Wise Test Configurations for Software Product Lines', IIEEE Trans. Software Eng., vol. 40, 2014

Empirical Evaluation



M. Isberner, F. Howar, and B. Steffen, ‘The Open-Source LearnLib’, in CAV 2015
Apache, Commons Math: The Apache Commons Mathematics Library. 2016.
T. Thüm, et al., ‘FeatureIDE: An extensible framework for feature-oriented software development’, Science of Computer Programming, vol. 79, 2014

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

A. Classen, 'Modelling with FTS: a Collection of Illustrative Examples'. 2010, [Online]. Available: https://researchportal.unamur.be/en/publications/modelling-with-fts-a-collection-of-illustrative-examples
H. Samih, H. L. Guen, R. Bogusch, M. Acher, and B. Baudry, 'Deriving Usage Model Variants for Model-Based Testing: An Industrial Case Study', in Proceedings of the ICECCS 2014
X. Devroey, G. Perrouin, A. Legay, P.-Y. Schobbens, and P. Heymans, 'Search-based Similarity-driven Behavioural SPL Testing', in Proceedings of the VaMoS 2016
V. Hafemann Fragal, A. Simao, and M. R. Mousavi, 'Validated Test Models for Software Product Lines: Featured Finite State Machines', in `FACS 2016.

Analysis of Results (RQ1 – Size of Product Pairs)

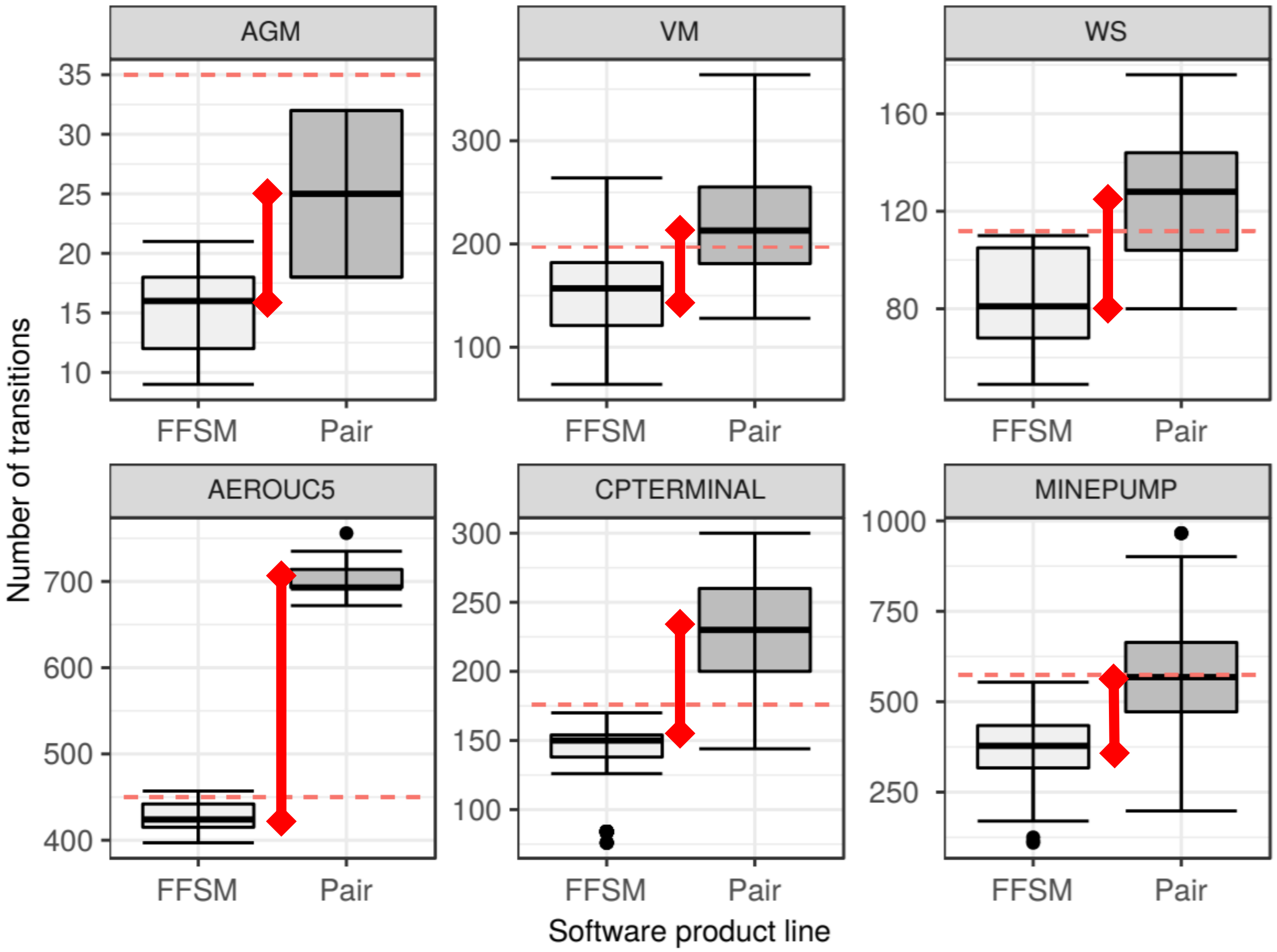


Figure 26 – Number of transitions in the learned FFSMs and pairs of products

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

Analysis of Results (RQ2 – Configuration similarity)

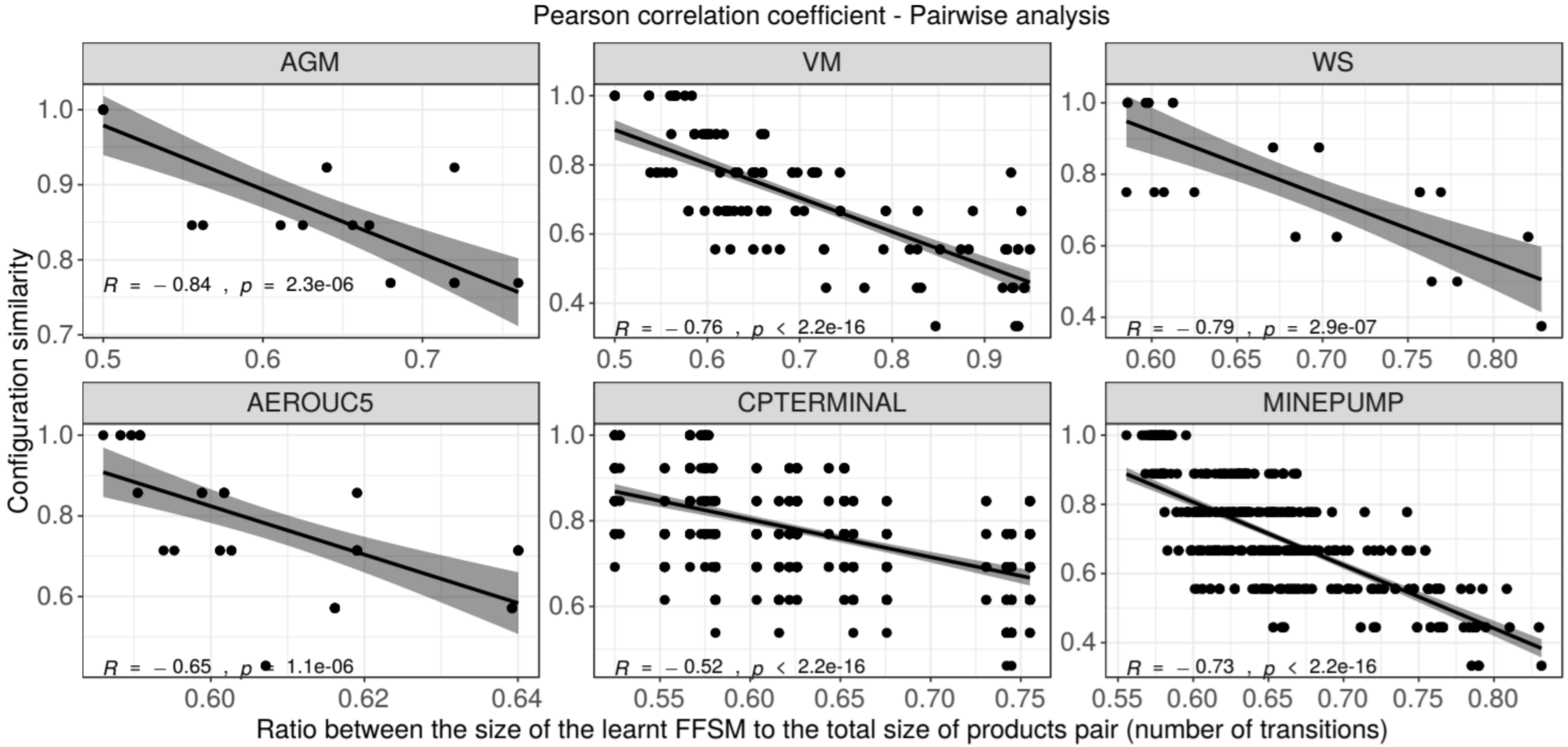


Figure 28 – Scatter plots for the relationship between the normalized size of the learned FFSM and configuration similarity

Analysis of Results (RQ3 – Size of Handcrafted models)

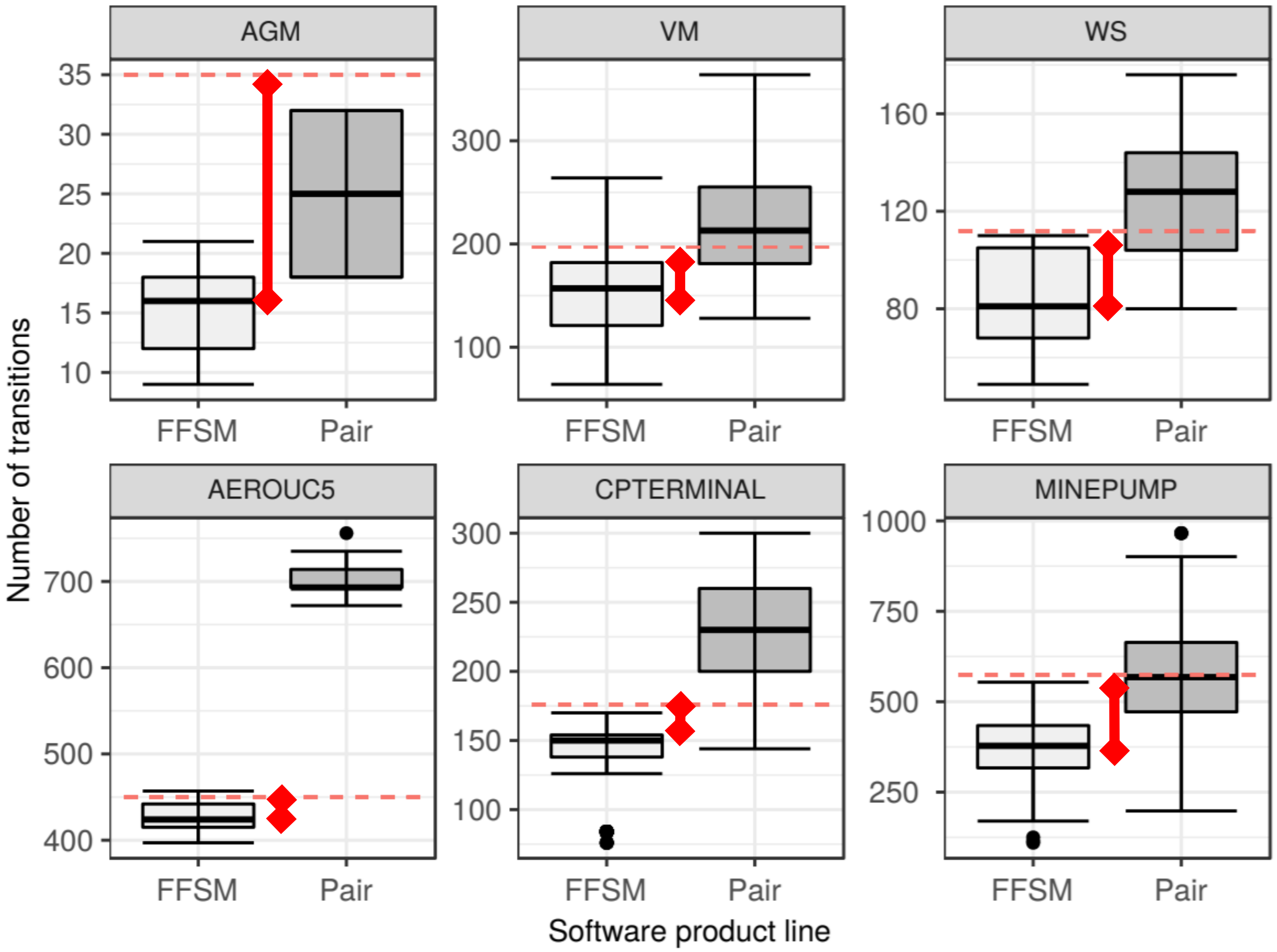


Figure 26 – Number of transitions in the learned FFSMs and pairs of products

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

Analysis of Results (RQ3 – Size of Handcrafted models)

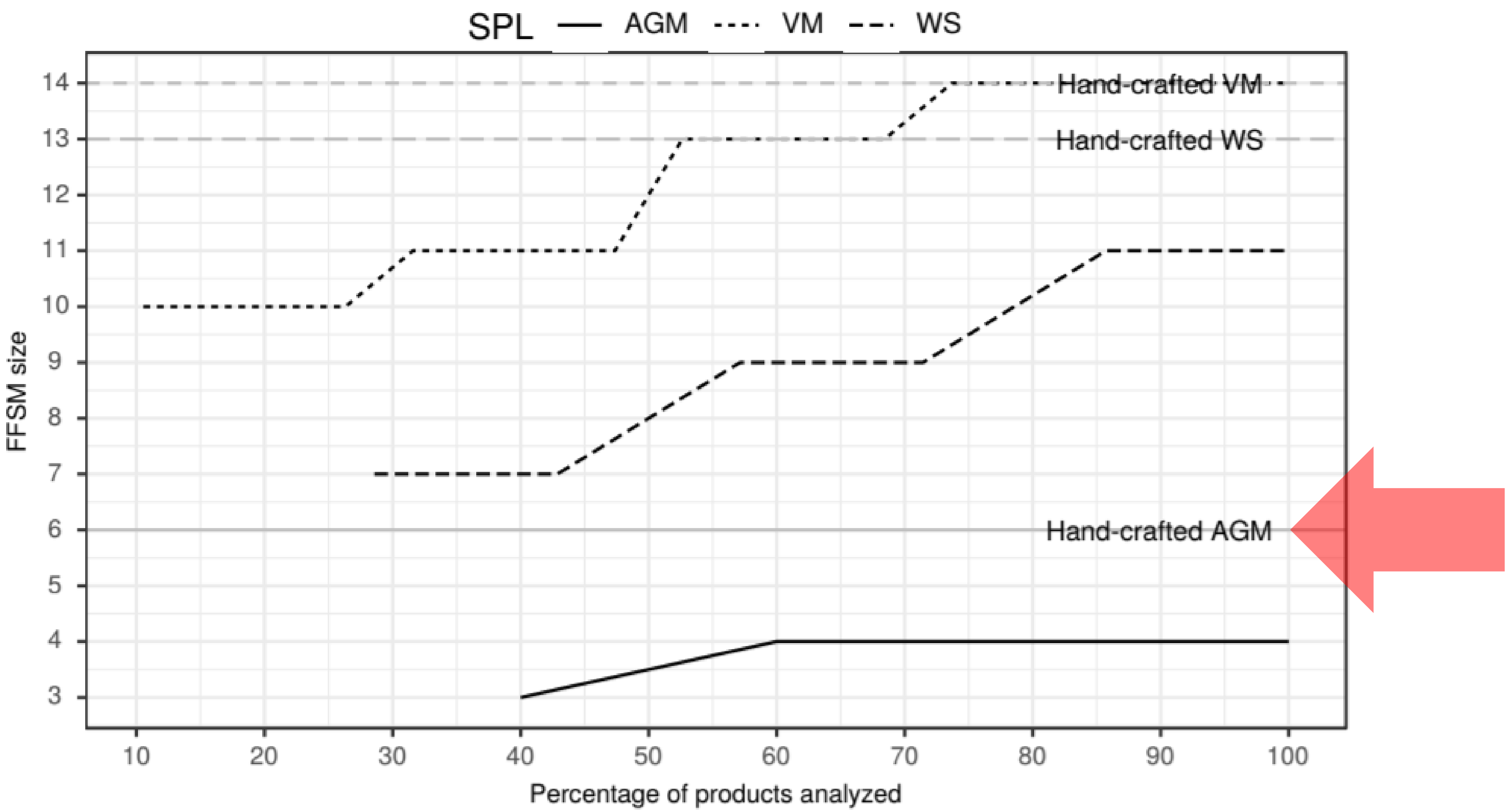
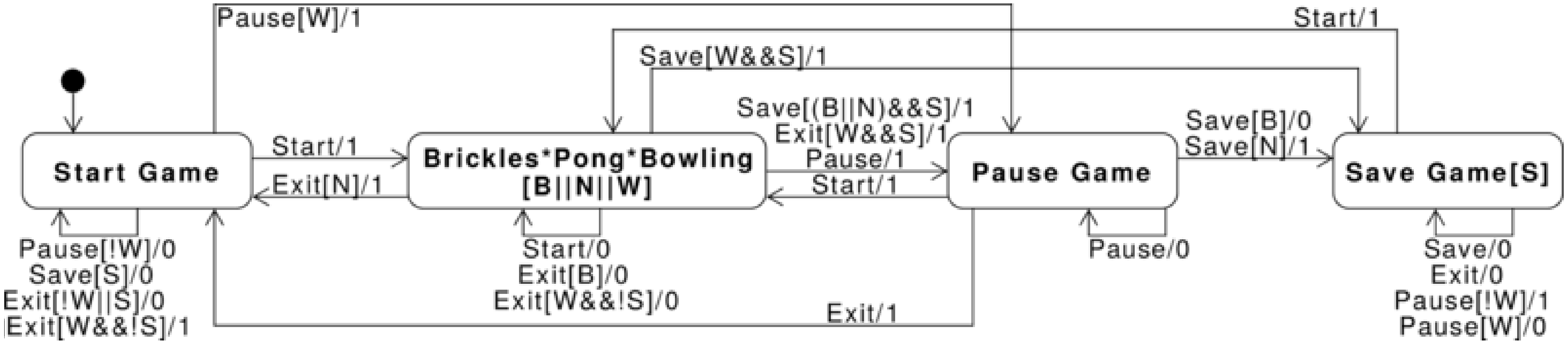


Figure 29 – Size of the recovered FFSMs

Analysis of Results (RQ3 – Size of Handcrafted models)



V. H. Fragal, 'Automatic generation of configurable test-suites for software product lines', PhD Thesis, Universidade de São Paulo, 2017.

Summary (Learning from Difference)

The $FFSM_{Diff}$ algorithm is able to…

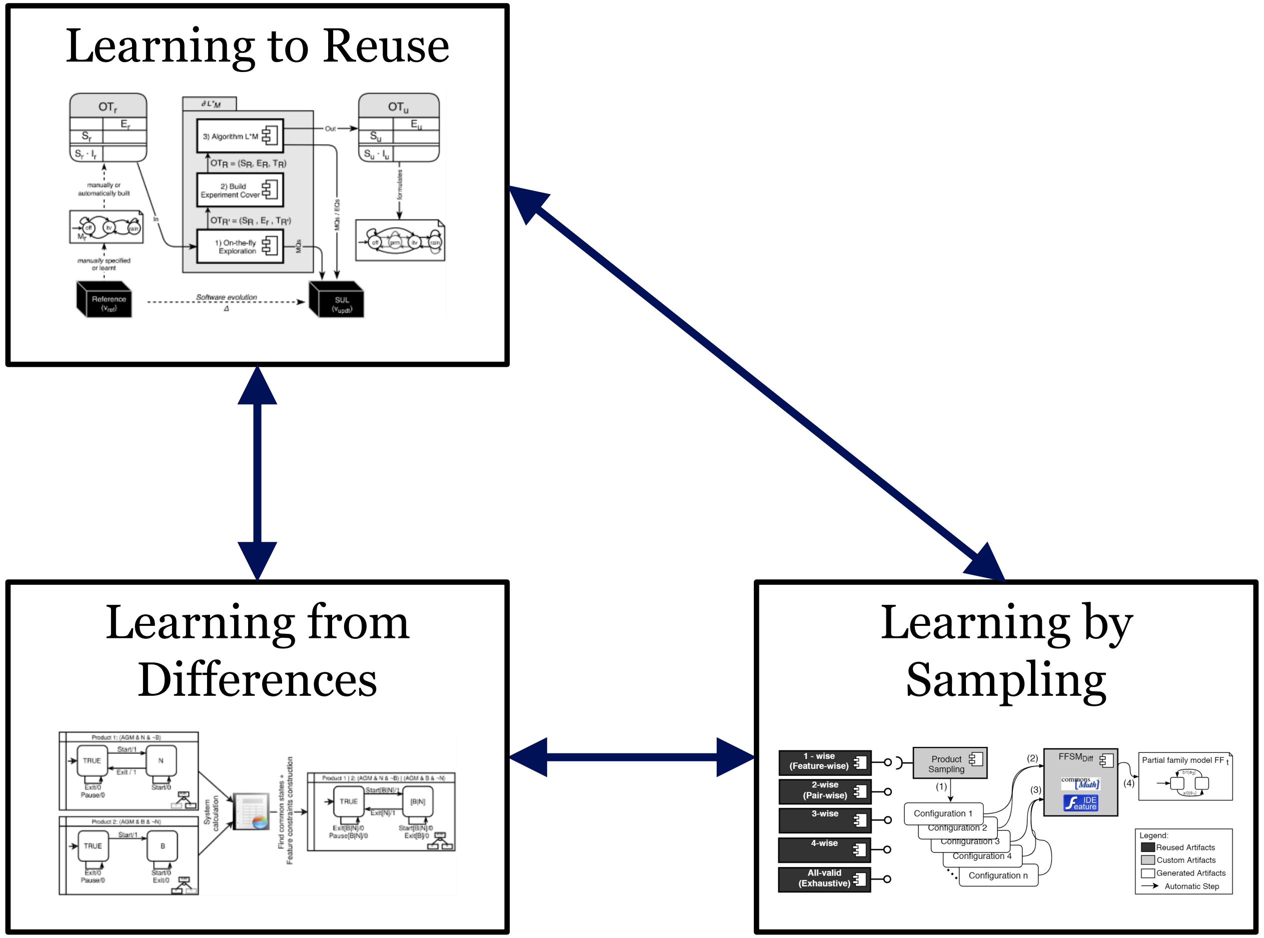
1. Learn **fresh FFSMs** from products pairs

   - 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

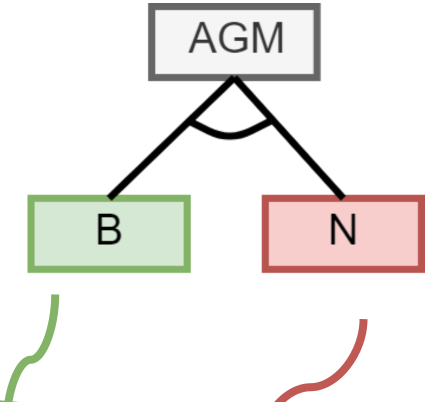
ICMC SÃO CARLOS USP

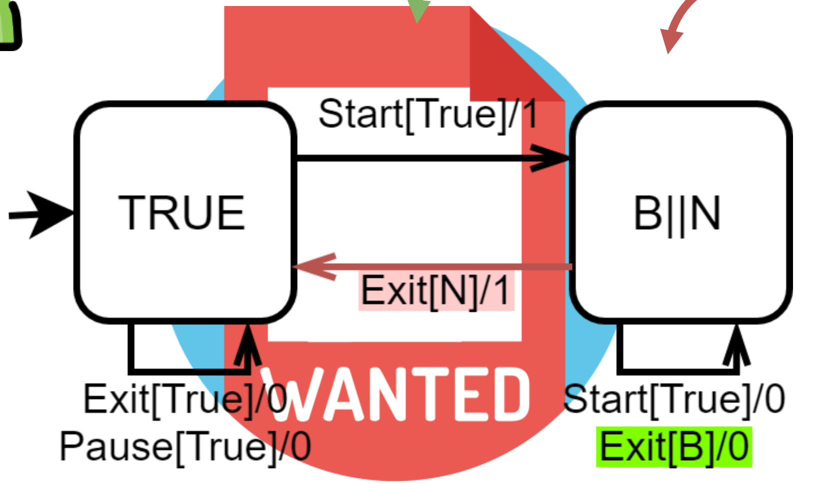
Universidade de São Paulo

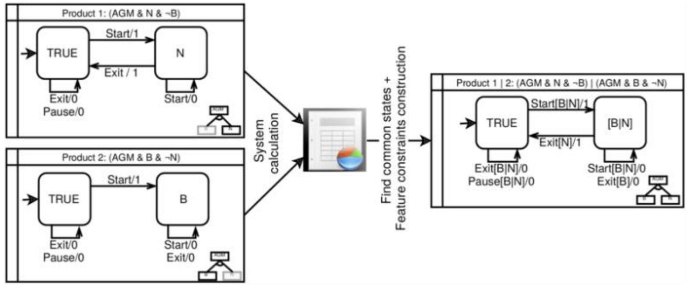
UNIVERSITY OF LEICESTER

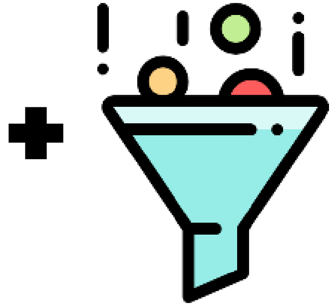
Context (Learning by Sampling)

- Software product lines (SPL)
  - Product-based strategies: *Impractical?*
  - Family-based strategies: *Models?*
- Family model learning → FFSM $_{Diff}$
  - Exhaustive learning
  - Learning by Sampling

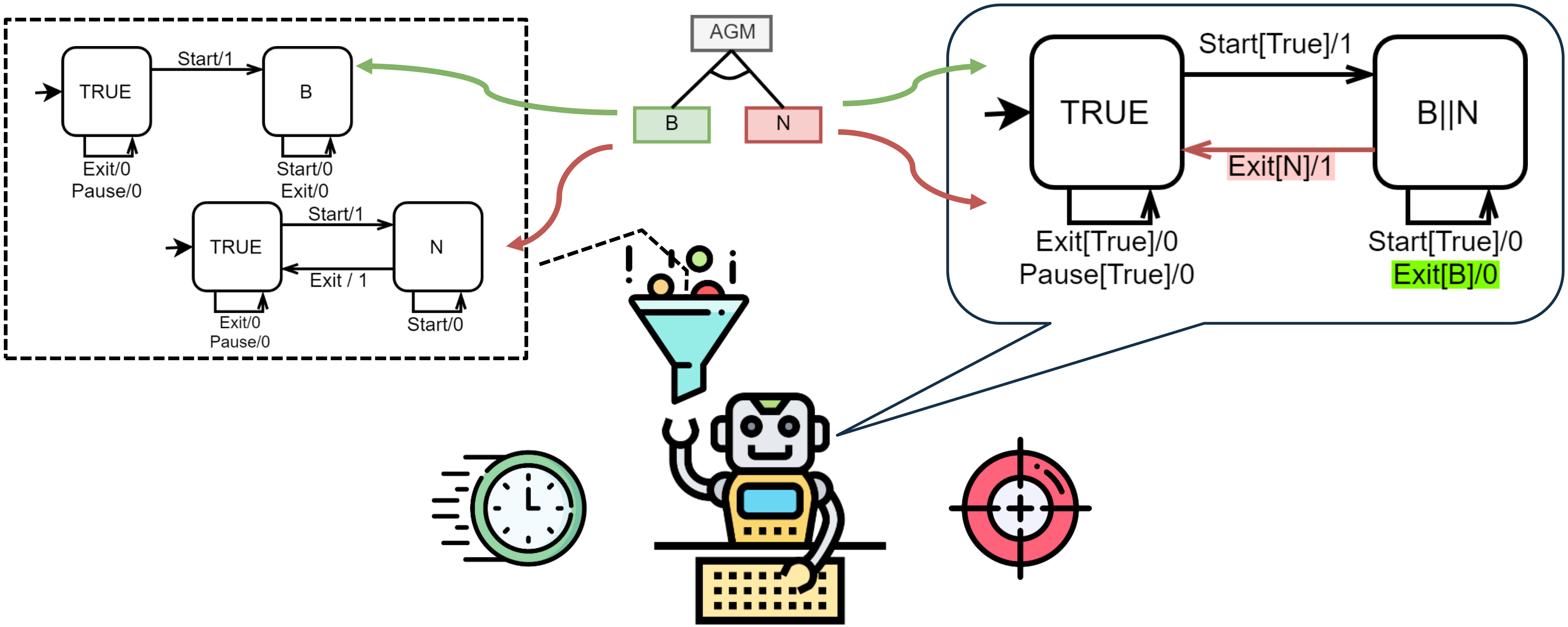








Research Problem (Learning by Sampling)



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

Contribution (Learning by Sampling)

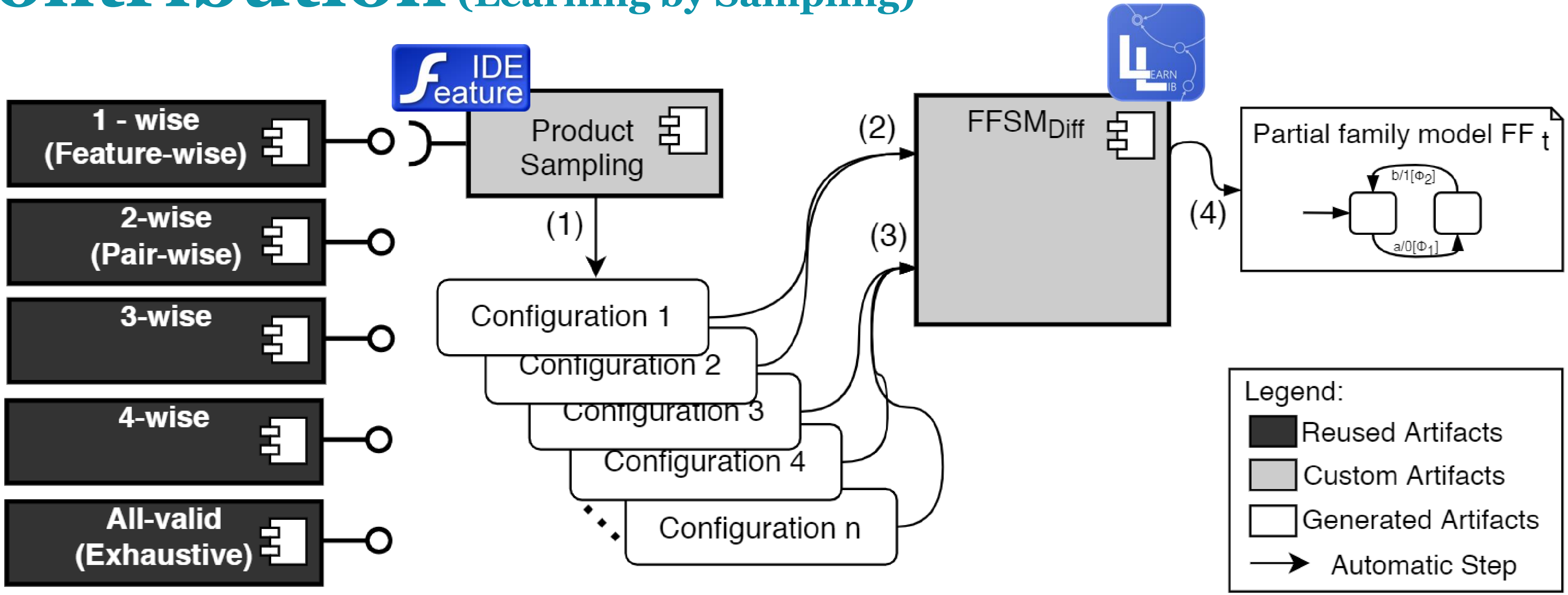


Fig. 8: Experiment design - Learning FFSMs by product sampling

M. Isberner, F. Howar, and B. Steffen, 'The Open-Source LearnLib', in CAV 2015
Apache, Commons Math: The Apache Commons Mathematics Library. 2016.
T. Thüm, et al., 'FeatureIDE: An extensible framework for feature-oriented software development', Science of Computer Programming, vol. 79, 2014

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

A. Classen, ‘Modelling with FTS: a Collection of Illustrative Examples’. 2010, [Online]. Available: https://researchportal.unamur.be/en/publications/modelling-with-fts-a-collection-of-illustrative-examples
H. Samih, H. L. Guen, R. Bogusch, M. Acher, and B. Baudry, ‘Deriving Usage Model Variants for Model-Based Testing: An Industrial Case Study’, in Proceedings of the ICECCS 2014
X. Devroey, G. Perrouin, A. Legay, P.-Y. Schobbens, and P. Heymans, ‘Search-based Similarity-driven Behavioural SPL Testing’, in Proceedings of the VaMoS 2016
V. Hafemann Fragal, A. Simao, and M. R. Mousavi, ‘Validated Test Models for Software Product Lines: Featured Finite State Machines’, in `FACS 2016.

Analysis of Results (RQ4 – Learning by Sampling)

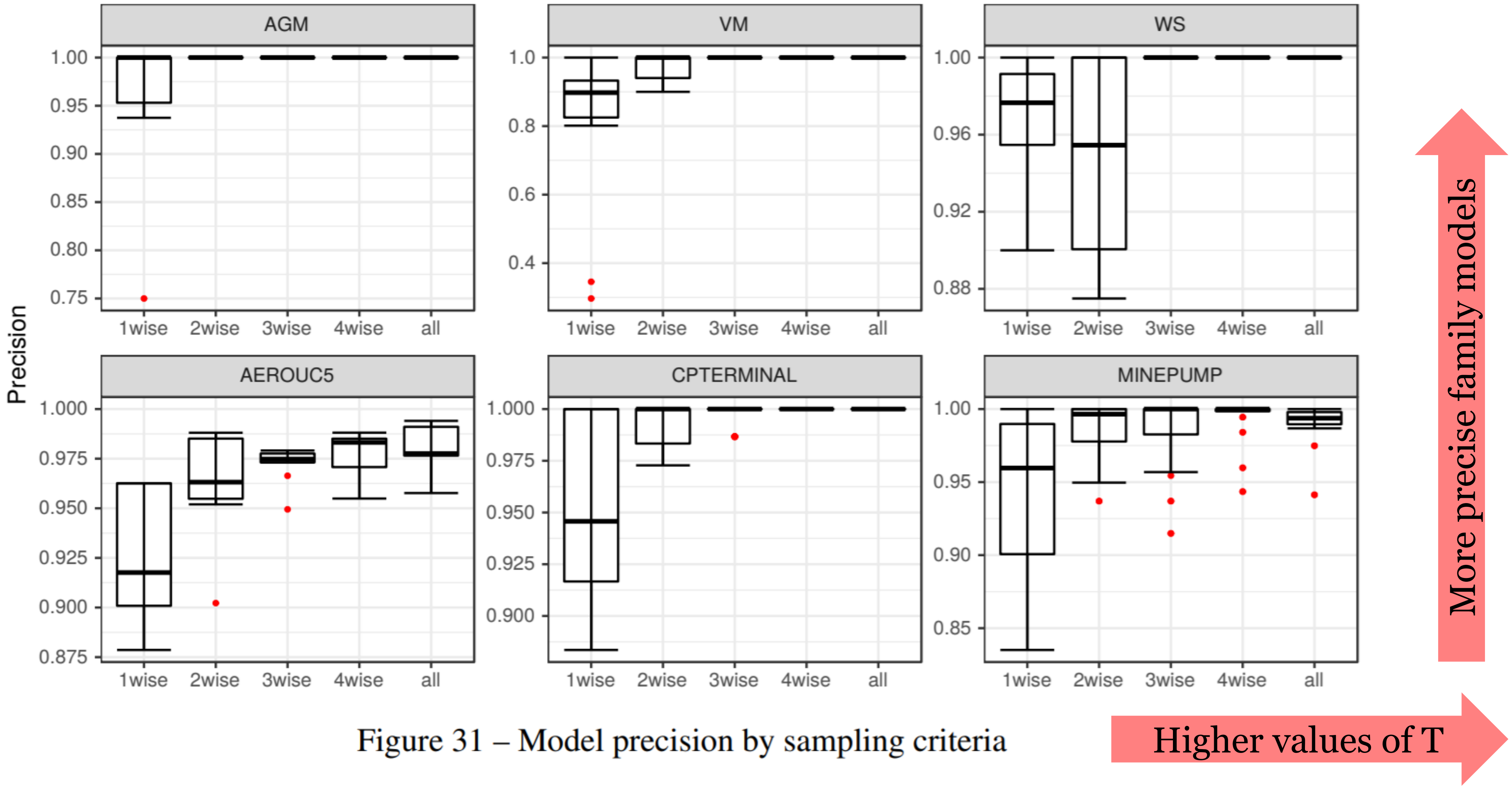


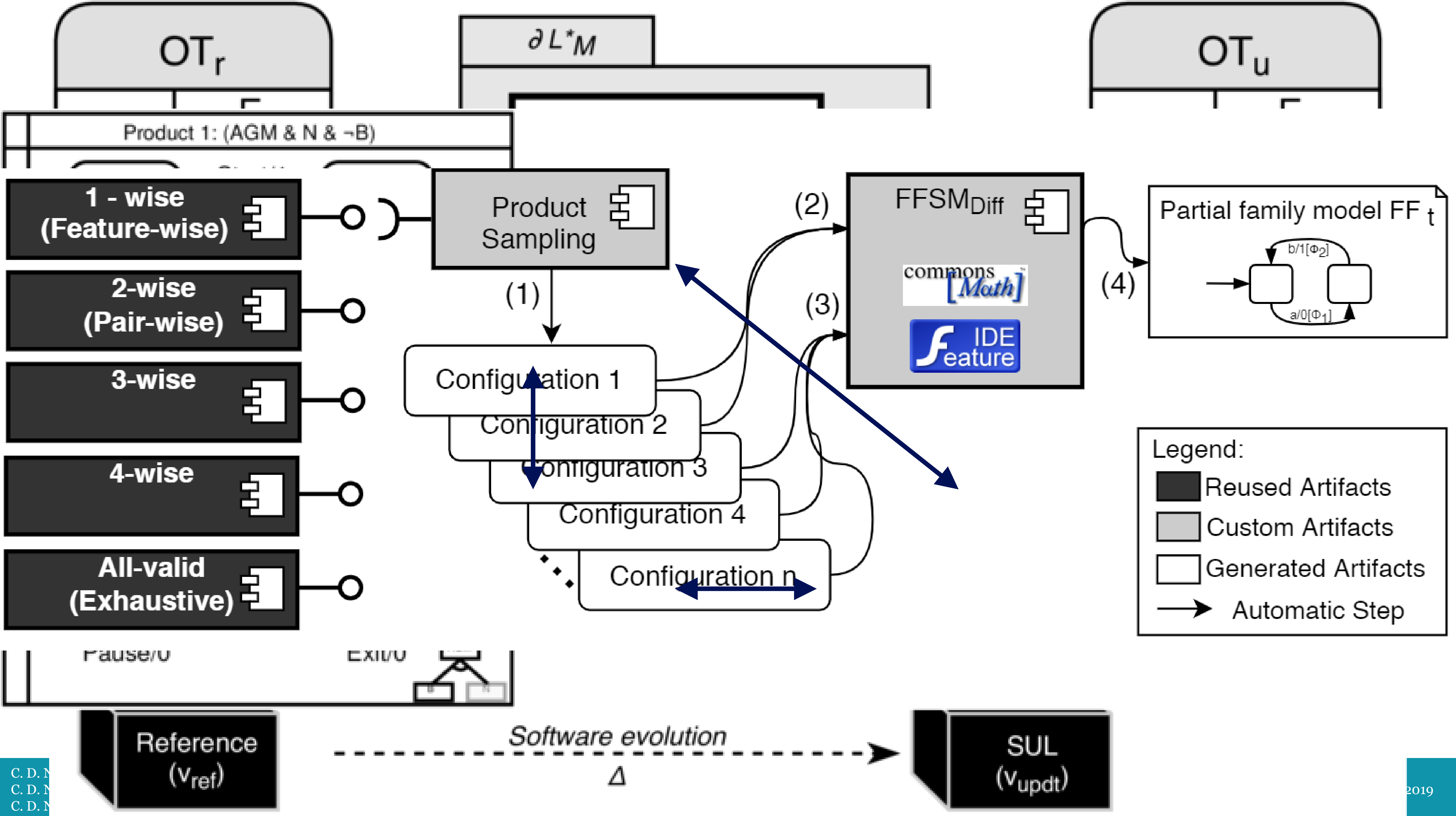
Figure 31 – Model precision by sampling criteria

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

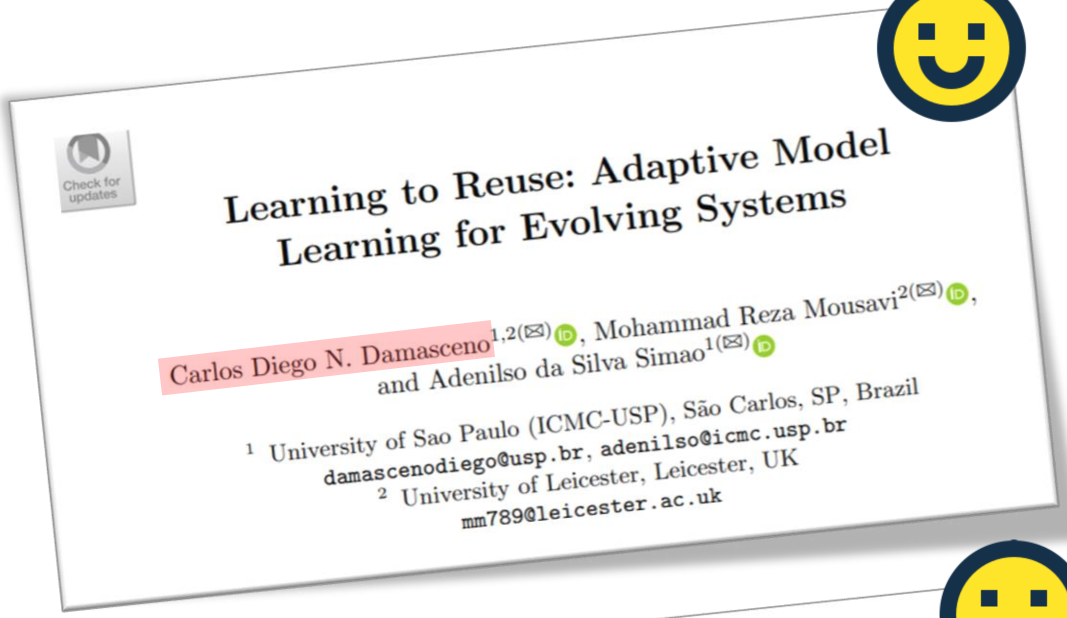
Summary (Learning by Sampling)

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
3. We show evidences that **product sampling** can be **helpful** to **family model learning and recovery**

Final Remarks and Future Work


OT_r
∂L*_M
OT_u
Product 1: (AGM & N & ¬B)
1 - wise (Feature-wise)
2-wise (Pair-wise)
3-wise
4-wise
All-valid (Exhaustive)
Product Sampling
(1)
(2)
(3)
(4)
Configuration 1
Configuration 2
Configuration 3
Configuration 4
Configuration n
FFSM_Diff
commons Math
IDE Feature
Partial family model FF_t
b/1[Φ2]
a/0[Φ1]
Legend:
Reused Artifacts
Custom Artifacts
Generated Artifacts
Automatic Step
Pause/0
Exit/0
Reference (v_ref)
Software evolution
Δ
SUL (v_updt)

The main publications of this PhD Thesis

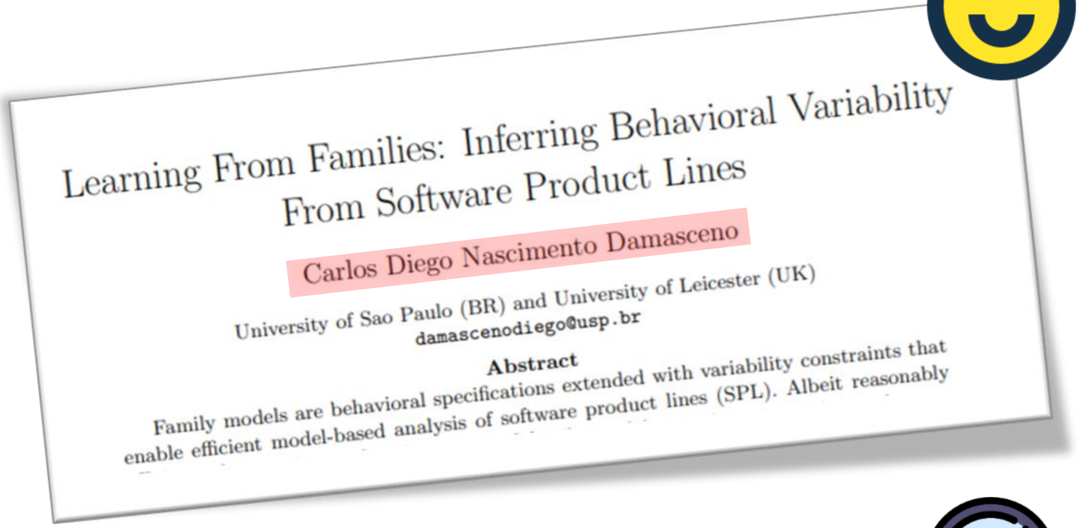


Check for updates

Learning to Reuse: Adaptive Model Learning for Evolving Systems

Carlos Diego N. Damasceno[1,2](✉), Mohammad Reza Mousavi[2](✉), and Adenilso da Silva Simao[1](✉)

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

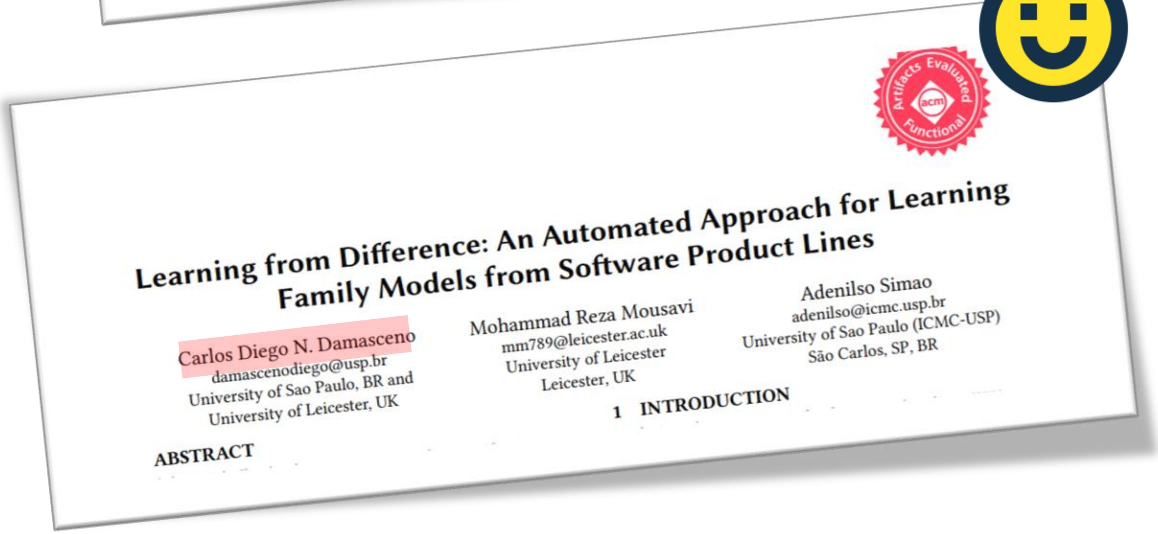


Learning From Families: Inferring Behavioral Variability From Software Product Lines

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



Artifacts Evaluated acm Functional

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

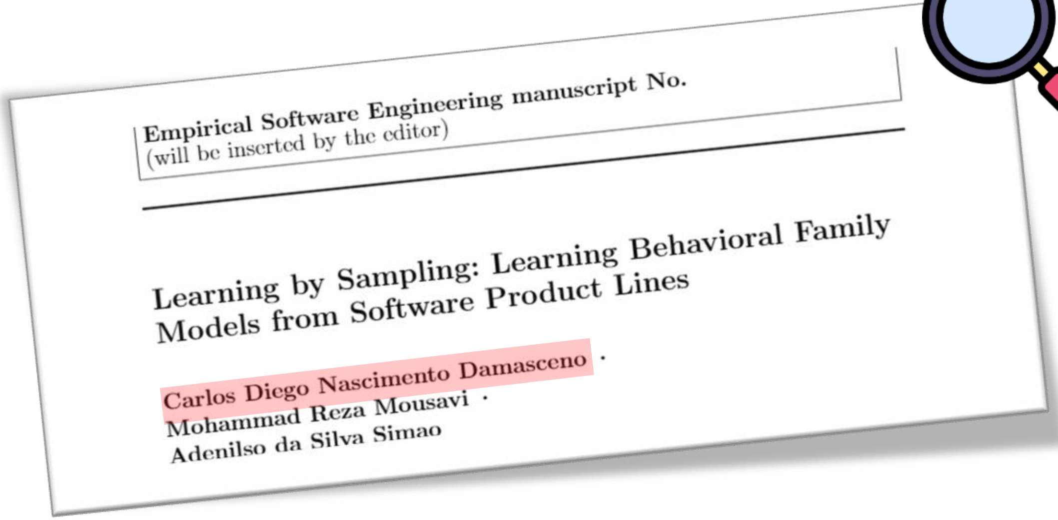
Carlos Diego N. Damasceno
damascenodiego@usp.br
University of Sao Paulo, BR and
University of Leicester, UK

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

ABSTRACT

1 INTRODUCTION

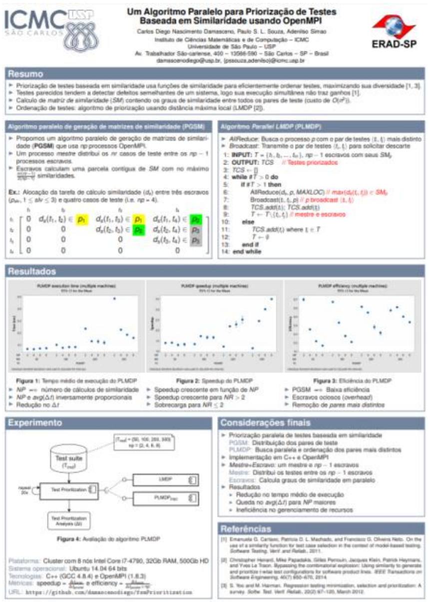


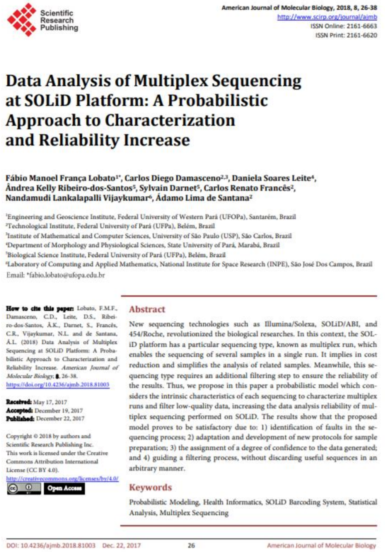
Empirical Software Engineering manuscript No.
(will be inserted by the editor)

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

Carlos Diego Nascimento Damasceno ·
Mohammad Reza Mousavi ·
Adenilso da Silva Simao

Other contributions

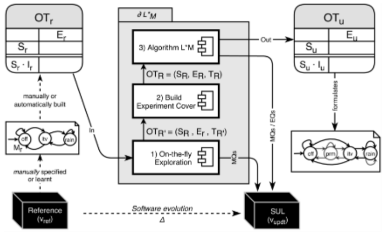




Future Work

Adaptive
Discrimina
M

Learning to Reuse

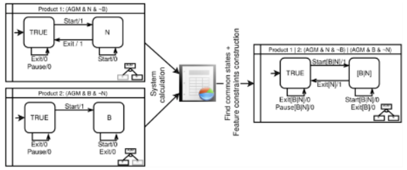


Active family model learning

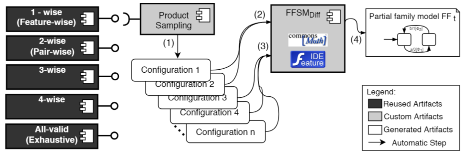
Fingerprinting evolving systems

Incremental Configurable Queries

Learning from Differences



Learning by Sampling



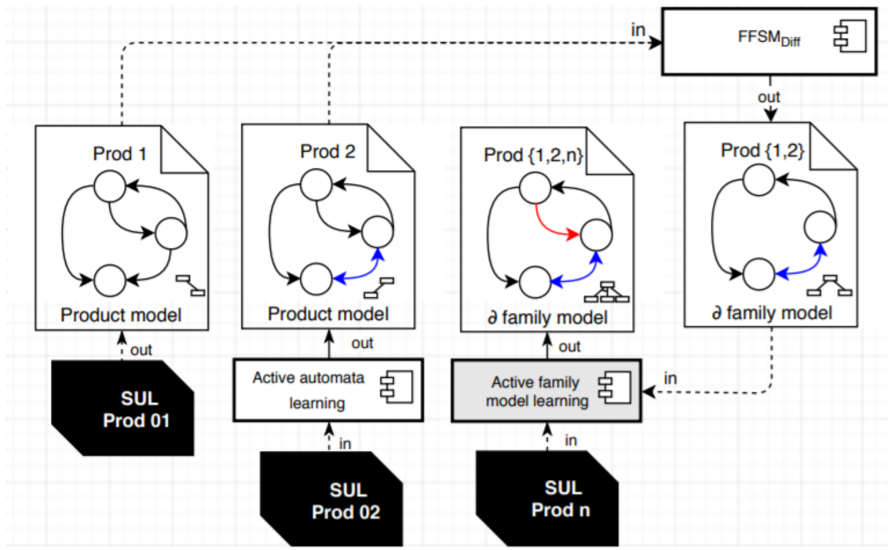


Figure 32 – Active family model learning

Source: Damasceno (2019).

M. Isberner, F. Howar, and B. Steffen, ‘The TTT Algorithm: A Redundancy-Free Approach to Active Automata Learning’, in Runtime Verification, 2014
C. D. N. Damasceno, ‘Learning From Families: Inferring Behavioral Variability From Software Product Lines’, presented at the PhD Symposium at Integrated Formal Methods, Bergen, Norway, 2019.
M. Al-Hajjaji, S. Krieter, T. Thüm, M. Lochau, and G. Saake, ‘IncLing: Efficient Product-Line Testing Using Incremental Pairwise Sampling’, in Proceedings of the GPCE 2016
G. Shu and D. Lee, ‘Network Protocol System Fingerprinting - A Formal Approach’, in Proceedings of the IEEE INFOCOM 2006

Thank you

Questions?