Learning From Families

Inferring Behavioral Variability From Software Product Lines

Carlos Diego Nascimento Damasceno
PhD candidate @ University of Sao Paulo, BR[1]
Visiting PhD researcher @ University of Leicester, UK[2]
damascenodiego@usp.br

December 3rd 2019

[1] Supervisor: Adenilso Simao
[2] Supervisor: Mohammad Mousavi

Context

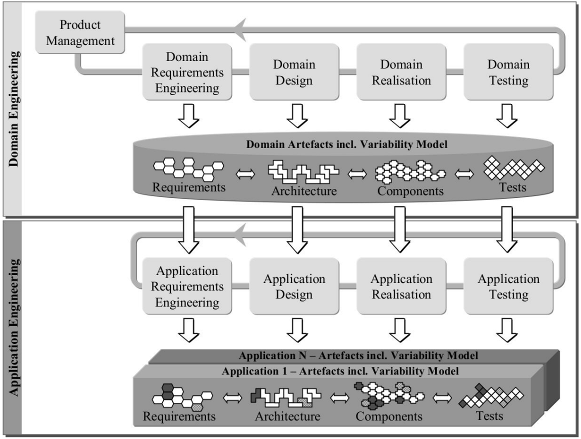


Figure: In software product lines (SPL), software variants are developed simultaneously from a common set of reusable assets

Context - Family models

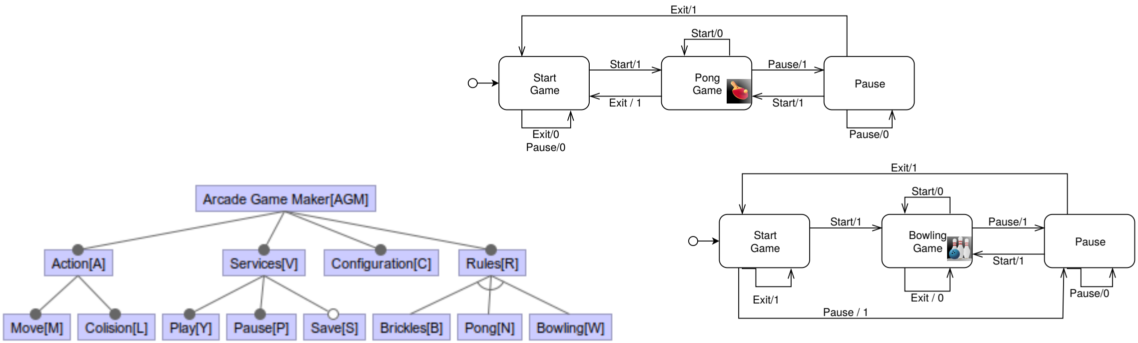


Figure: A family model unifies multiple state machines of a product-line into a single model where states and transitions are annotated with feature constraints [3]

[3] e.g., featured finite state machine - FFSM (Fragal et al., 2017)

Context - Family models

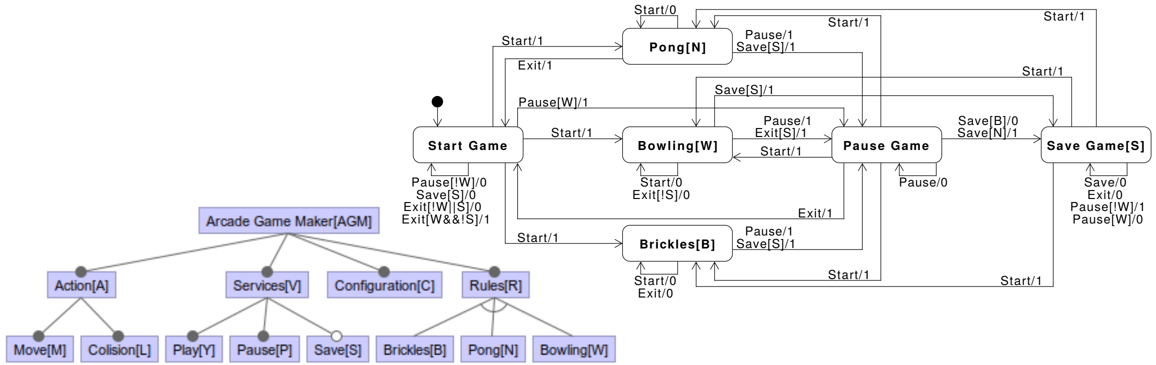


Figure: A family model unifies multiple state machines of a product-line into a single model where states and transitions are annotated with feature constraints [3]

[3] e.g., featured finite state machine - FFSM (Fragal et al., 2017)

Problem Statement

- 👍 Family-based analysis (e.g., model-based testing [4] and model checking [5])
- 👍 **Cost** as a function of the **number of features** and **amount of feature sharing**
- 👍 **Redundant analysis** are **avoided/minimised**
- 👎 **Creation and maintenance** of family and product models are challenging
- 👎 **Outdated models** may arise as **products evolve**

[4] Fragal et al. (2017)
[5] ter Beek et al. (2017)

Research objective

Investigate approaches to support the automated construction of family models from SPLs

RQ1) How can we effectively infer product models from evolving system?

Model Learning

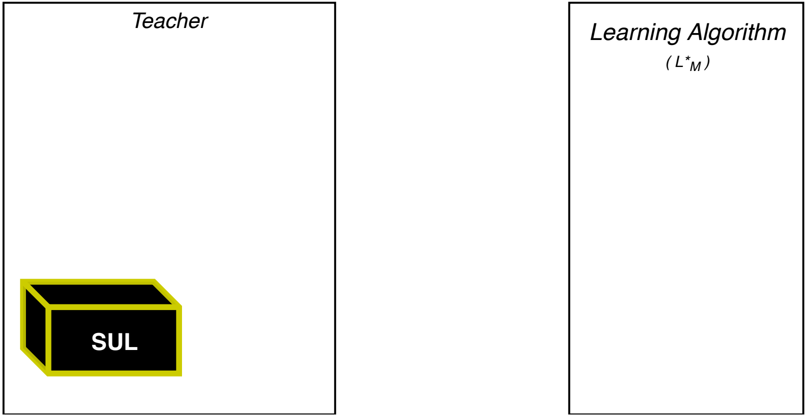


Figure: The Minimally Adequate Teacher (MAT) framework (Angluin, 1987)

Model Learning

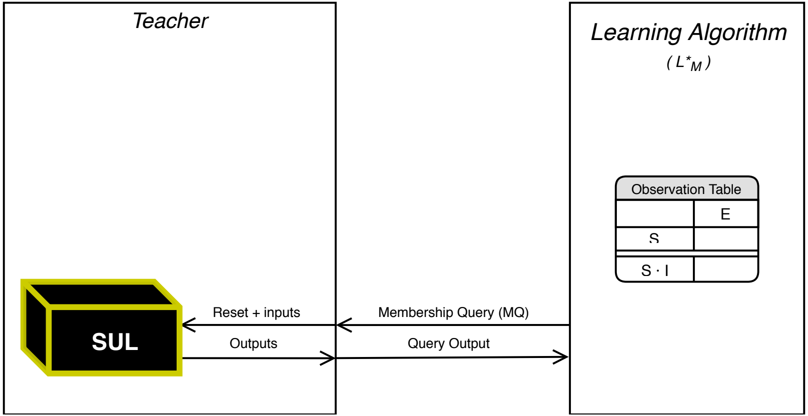


Figure: The Minimally Adequate Teacher (MAT) framework (Angluin, 1987)

Model Learning

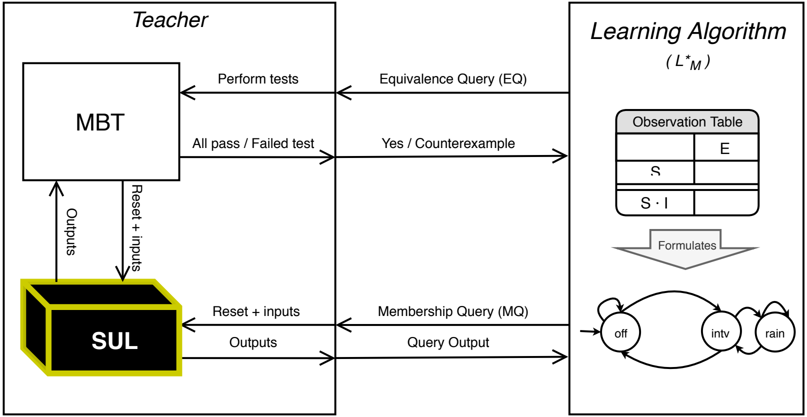


Figure: The Minimally Adequate Teacher (MAT) framework (Angluin, 1987)

Model Learning (Example)

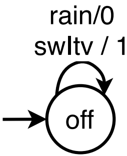


Figure: Initial Hypothesis

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

Table: Initial observation table (OT)

Model Learning (Example)

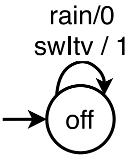


Figure: First Hypothesis

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

Table: First observation table

Model Learning (Example)

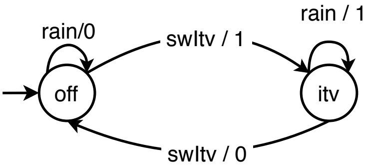


Figure: Second Hypothesis

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

Table: Second observation table

Model Learning (Example)

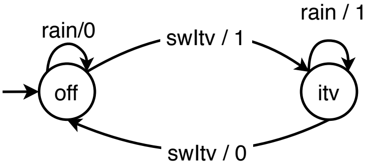


Figure: Second Hypothesis

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

Table: Second observation table ($\mathcal{H} \neq$ SUL)

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

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

Model Learning (Example)

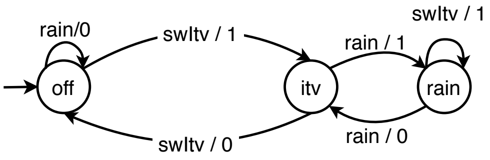


Figure: Mealy machine of a windscreen wiper supporting intervaled and fast wiping

		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

What if our SUL evolves?

Adaptive model learning for evolving systems

- Reuse transfer and/or separating sequences from pre-existing models
  - Reduce the time for model checking (Groce et al., 2002; Chaki et al., 2008)
  - Find states maintained in newer versions (Windmüller et al., 2013)

Adaptive model learning for evolving systems

- Reuse transfer and/or separating sequences from pre-existing models
  - Reduce the time for model checking (Groce et al., 2002; Chaki et al., 2008)
  - Find states maintained in newer versions (Windmüller et al., 2013)

Adaptive model learning for evolving systems

- Reuse transfer and/or separating sequences from pre-existing models
  - Reduce the time for model checking (Groce et al., 2002; Chaki et al., 2008)
  - Find states maintained in newer versions (Windmüller et al., 2013)

Adaptive model learning for evolving systems

- Reuse transfer and/or separating sequences from pre-existing models
  - Reduce the time for model checking (Groce et al., 2002; Chaki et al., 2008)
  - Find states maintained in newer versions (Windmüller et al., 2013)
- Research Gaps:
  - Reuse low quality sequences $\rightarrow$ Irrelevant `MQs` (Huistra et al., 2018)
  - How can we calculate good-quality subsets of sequences?

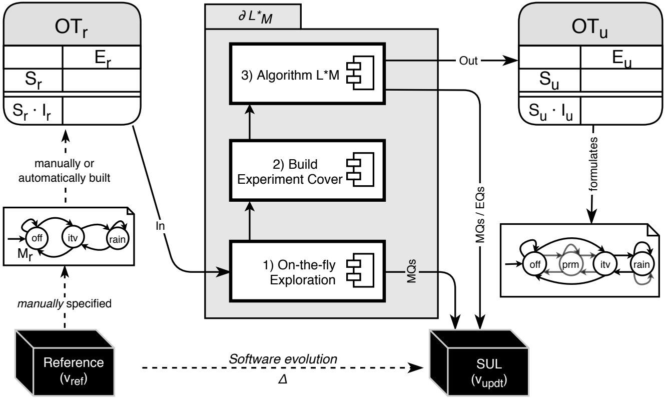
Adaptive model learning for evolving systems

- Reuse transfer and/or separating sequences from pre-existing models
  - Reduce the time for model checking (Groce et al., 2002; Chaki et al., 2008)
  - Find states maintained in newer versions (Windmüller et al., 2013)
- Research Gaps:
  - Reuse low quality sequences → Irrelevant `MQs` (Huistra et al., 2018)
  - How can we calculate good-quality subsets of sequences?

Adaptive model learning for evolving systems

- Reuse transfer and/or separating sequences from pre-existing models
  - Reduce the time for model checking (Groce et al., 2002; Chaki et al., 2008)
  - Find states maintained in newer versions (Windmüller et al., 2013)
- Research Gaps:
  - Reuse low quality sequences → Irrelevant `MQs` (Huistra et al., 2018)
  - How can we calculate good-quality subsets of sequences?

The partial-Dynamic L^*_M algorithm [6]



[6] Accepted at the iFM 2019 (Damasceno et al., 2019b) - **Thu @ 10:30 - Aud 1**

Step 1: On-the-fly exploration of the reused OT

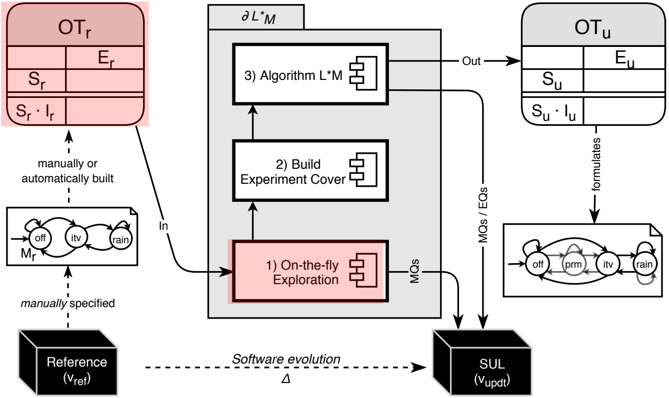


Figure: The `partial-Dynamic` L^*_M algorithm starts by exploring reused `OT`s on-the-fly to discard *redundant* transfer sequences [7]

---

[7] **Improvement #1:** We optimized Chaki et al. (2008)

Step 2: Building an experiment cover tree

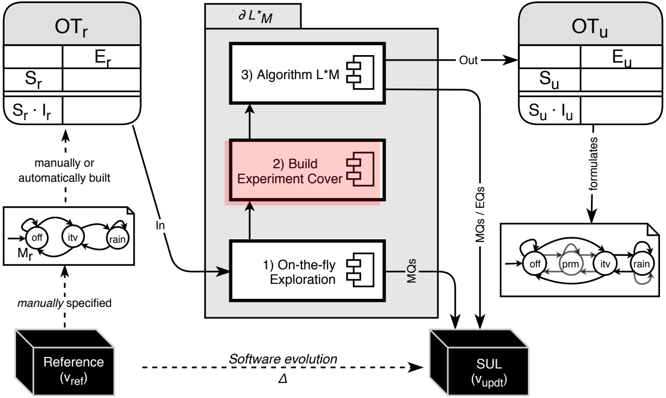


Figure: The `partial-Dynamic` L^*_M algorithm searches for *deprecated* separating sequences [8]

[8] **Improvement #2:** We used breadth-first search to minimize the set of separating sequences

Step 3: Running L_M^* using the outcomes of partial-Dynamic L_M^*

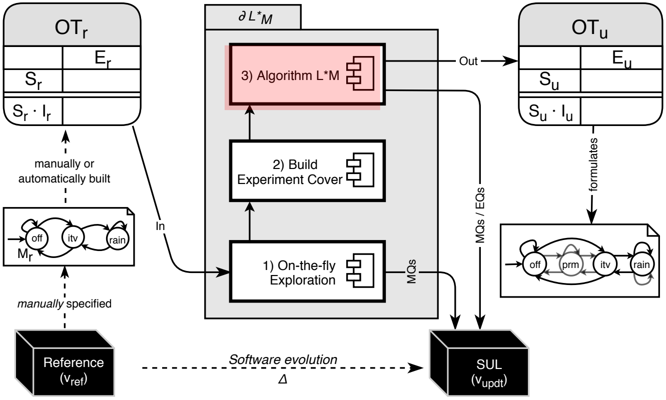


Figure: The L_M^* algorithm **starts by reusing transfer and separating sequences** to reach and distinguish more states than **in the traditional setup** (i.e., initial state only) [9]

[9] **Improvement #3:** We use the subsets of reused sequences as the initial setup for model learning

partial-Dynamic L^*_M - Empirical evaluation

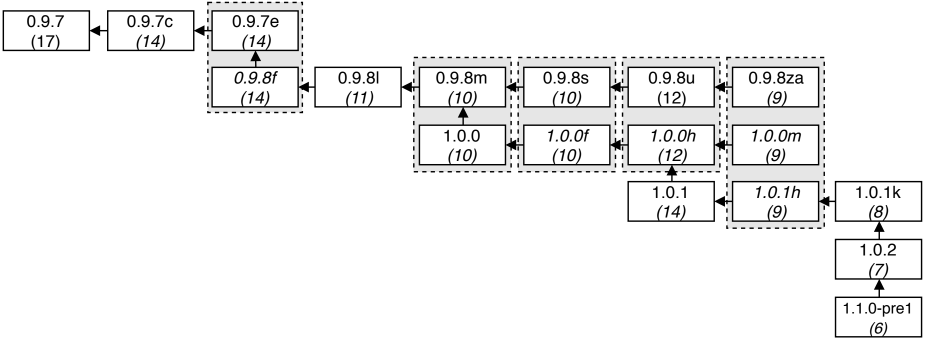


Figure: OpenSSL toolkit: 18 FSMs versions used as SUL (de Ruiter, 2016)

partial-Dynamic L_M^* - Main findings

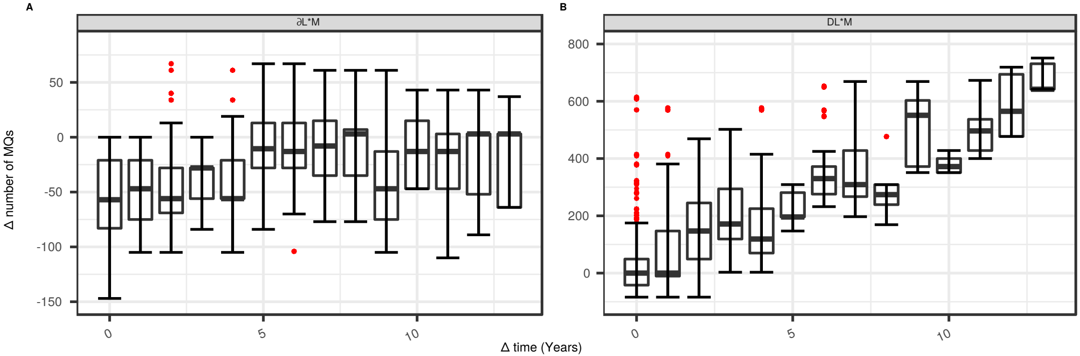


Figure: Our technique required less MQs

partial-Dynamic L^*_M - Main findings

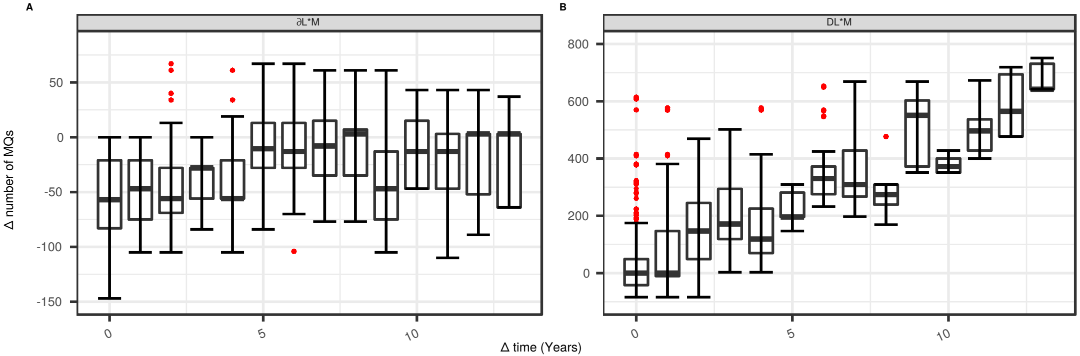


Figure: Our technique was not influenced by the temporal distance between versions

RQ2) How can we merge state machines into family models?

The $FFSM_{Diff}$ algorithm[10]

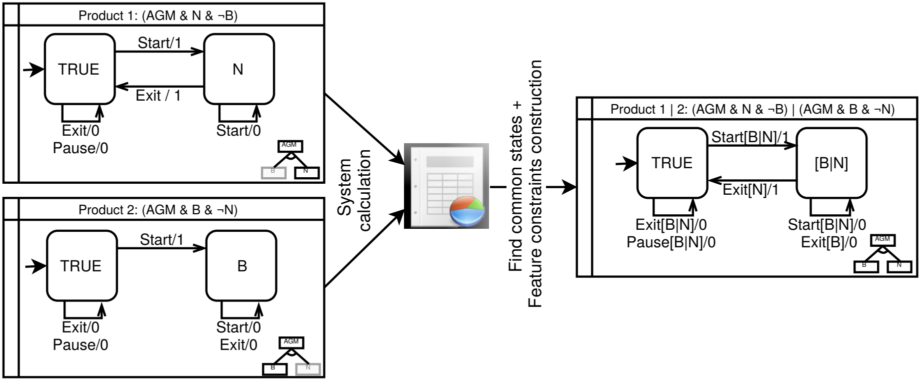


Figure: An automated technique to learn **fresh FFSM** and **include new FSMs into existing FFSMs** by **comparing products models** and **incorporating variability** to express product-specific behaviors with feature constraints

[10] This paper has been published at the SPLC 2019 (Damasceno et al., 2019a)

The $FFSM_{Diff}$ algorithm - Main findings

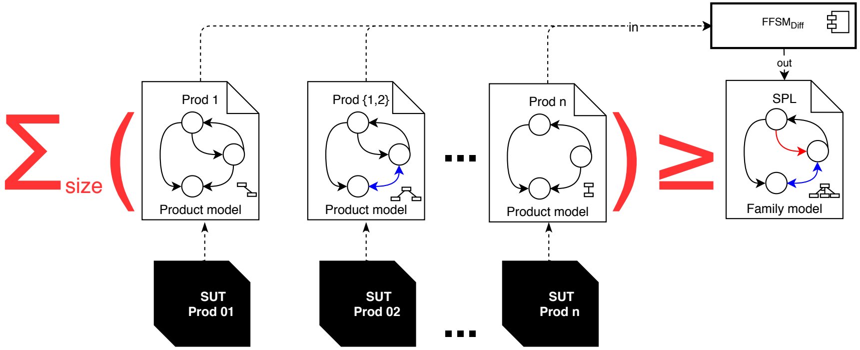


Figure: Product models can be effectively merged into succinct FFSMs, especially if there is high feature sharing

The $FFSM_{Diff}$ algorithm - Main findings

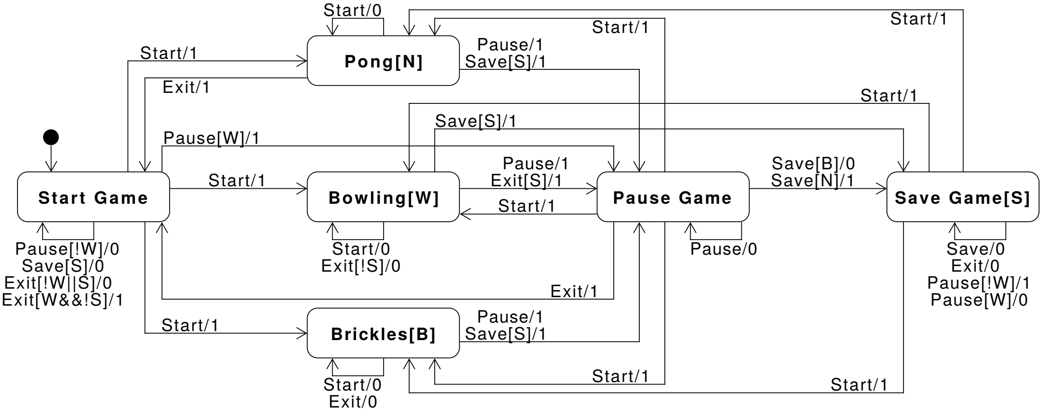


Figure: Alternative FFSM for AGM with fewer states (Fragal, 2017)

The $FFSM_{Diff}$ algorithm - Main findings

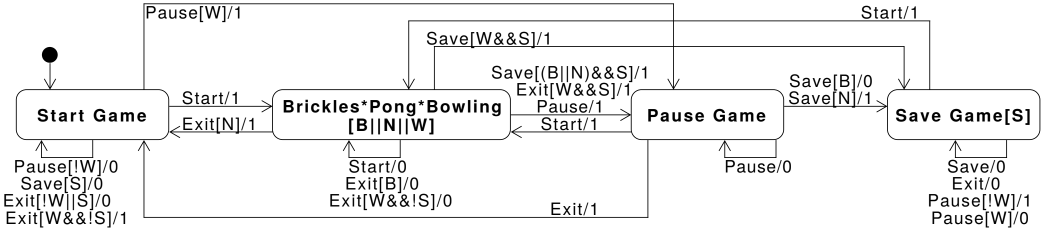


Figure: Alternative FFSM for AGM with fewer states (Fragal, 2017)

RQ3) Family model learning

(Optimization)

Family model learning *(Optimization)*

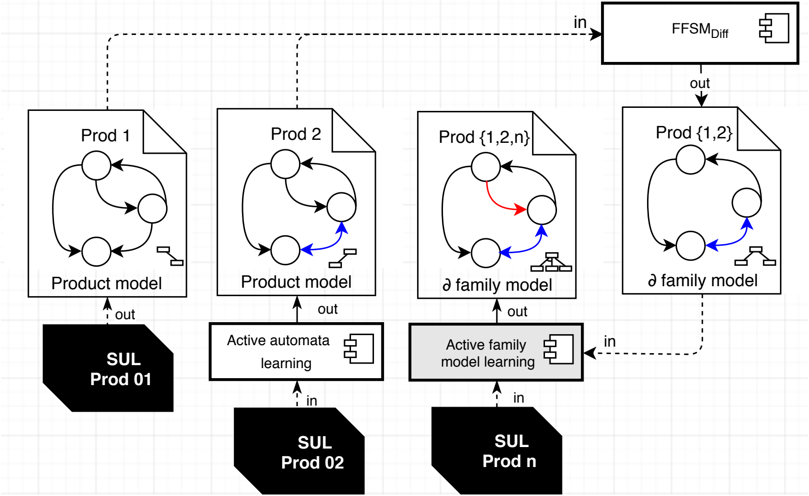


Figure: Family Model Learning

Family model learning *(Optimization)*

- Replace **traditional FSMs** by **partial family models**
  - **Expected result:** reduction on the number of queries
- **Current issue:**
  - Lack of FSMs large and complex enough for family model learning

Summary

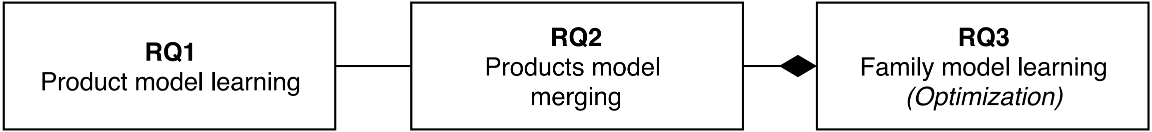


Figure: Summary

Future work

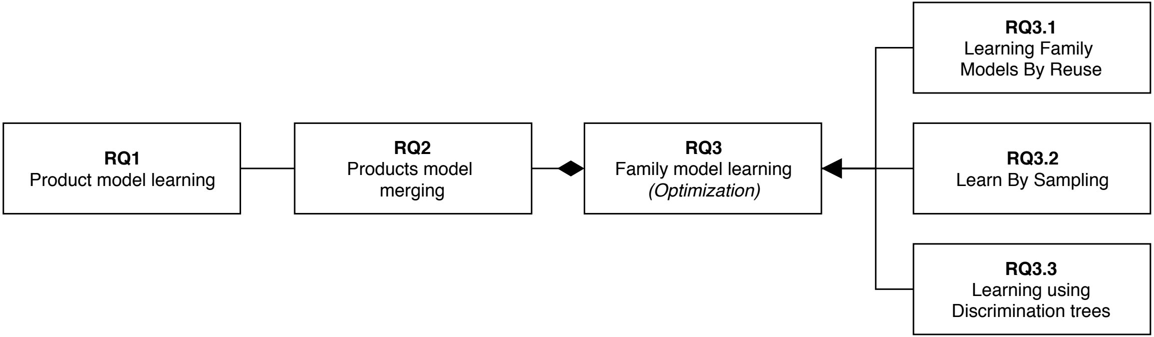


Figure: Future work

Questions?

https://damascenodiego.github.io/projects/

UNIVERSITY OF LEICESTER

Universidade de São Paulo

CAPES

CNPq
Conselho Nacional de Desenvolvimento Científico e Tecnológico

FAPESP
SÃO PAULO RESEARCH FOUNDATION

References I

Angluin, D. (1987). Learning regular sets from queries and counterexamples. *Information and Computation*, 75(2):87–106.

Chaki, S., Clarke, E., Sharygina, N., and Sinha, N. (2008). Verification of evolving software via component substitutability analysis. *Formal Methods in System Design*, 32(3):235–266.

Damasceno, C. D. N., Mousavi, M. R., and da Silva Simao, A. (2019a). Learning from difference: An automated approach for learning family models from software product lines. In *Proceeedings of the 23rd International Systems and Software Product Line Conference - Volume 1, SPLC 2019*, Paris, France. ACM Press.

Damasceno, C. D. N., Mousavi, M. R., and Simao, A. (2019b). Learning to reuse: Adaptive model learning for evolving systems. In *Integrated Formal Methods - 15th International Conference, IFM 2019, Bergen, Norway, December 2-6, 2019, Proceedings*. Springer.

de Ruiter, J. (2016). A tale of the openssl state machine: A large-scale black-box analysis. In *Secure IT Systems - 21st Nordic Conference, NordSec 2016, Oulu, Finland, November 2-4, 2016, Proceedings*, pages 169–184.

References II

Fragal, V. H. (2017). *Automatic generation of configurable test-suites for software product lines*. PhD thesis, Universidade de São Paulo. [Online] `http://www.teses.usp.br/teses/disponiveis/55/55134/tde-10012019-085746/`.

Fragal, V. H., Simao, A., and Mousavi, M. R. (2017). *Validated Test Models for Software Product Lines: Featured Finite State Machines*, pages 210–227. Springer International Publishing, Cham.

Groce, A., Peled, D., and Yannakakis, M. (2002). Adaptive model checking. In *Proceedings of the 8th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, TACAS '02, pages 357–370, London, UK, UK. Springer-Verlag.

Huistra, D., Meijer, J., and Pol, J. (2018). Adaptive learning for learn-based regression testing. In Howar, F. and Barnat, J., editors, *Formal Methods for Industrial Critical Systems*, Lecture Notes in Computer Science, pages 162–177, Switzerland. Springer Publishers.

ter Beek, M. H., de Vink, E. P., and Willemse, T. A. C. (2017). *Family-Based Model Checking with mCRL2*, pages 387–405. Springer, Berlin, Heidelberg.

Windmüller, S., Neubauer, J., Steffen, B., Howar, F., and Bauer, O. (2013). Active continuous quality control. In *Proceedings of the 16th International ACM Sigsoft Symposium on Component-based Software Engineering*, CBSE '13, pages 111–120, New York, NY, USA. ACM.