UNIVERSITY OF LEICESTER

MODELING, MODEL LEARNING, AND MODEL BASED TESTING FOR VARIABILITY-INTENSIVE SYSTEMS

MOHAMMAD REZA MOUSAVI

JOINT WORK WITH MANY PEOPLE, INCLUDING:

Harsh Beohar

The University Of Sheffield.

Diego Damasceno

Radboud University

Adenilso Simao

Universidade de São Paulo

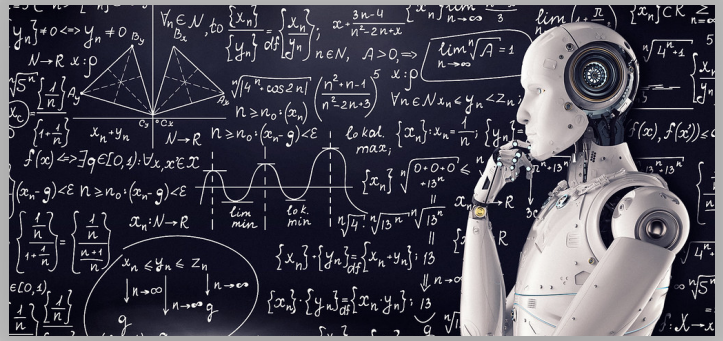
Mahsa Varshosaz

IT University of Copenhagen

Variability as a **first-class** citizen!

Modeling variability explicitly

Efficient testing using variability modeling

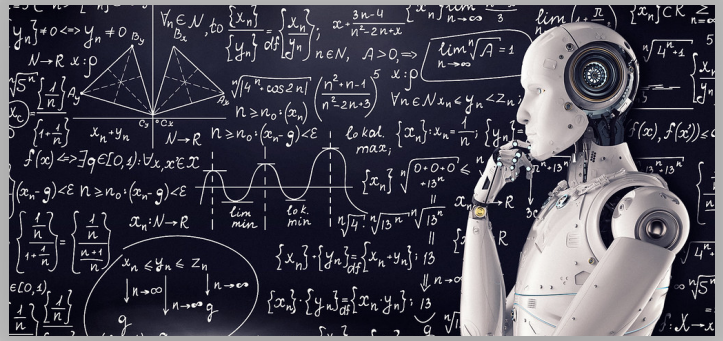


Learning about variability

Variability as a **first-class** citizen!

Modeling variability explicitly

Efficient testing using variability modeling



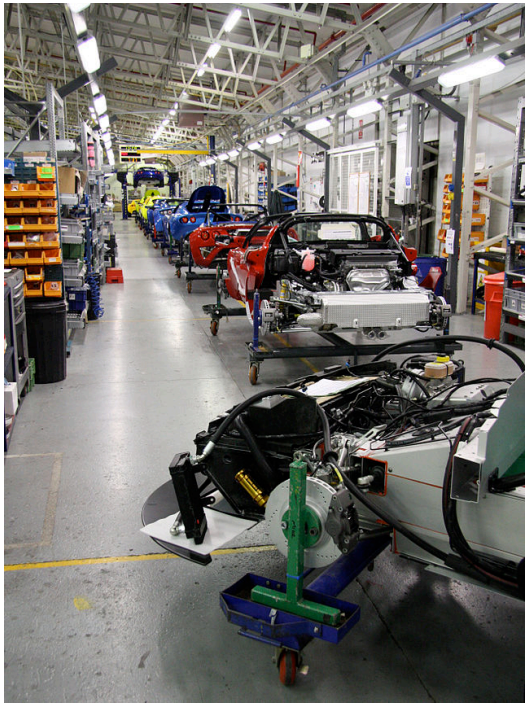
Learning about variability

UNIVERSITY OF LEICESTER

MODELING VARIABILITY

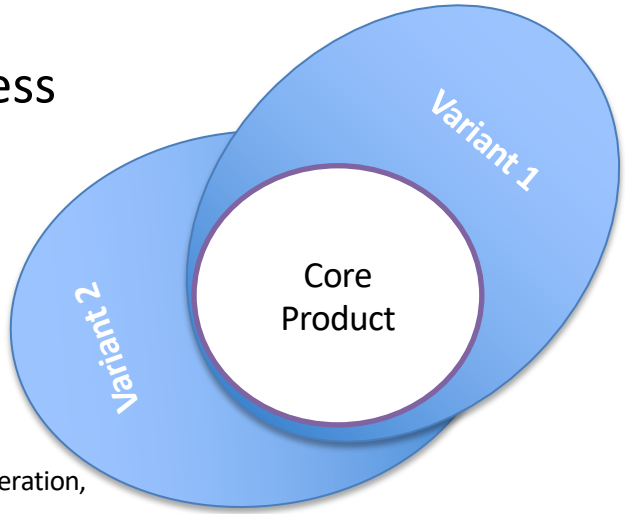
Variability Modeling

- Common core
- Variability points
  - Various models of variability
  - Huge challenges in analysis



Modeling Variability: Incremental

- Neat structure
- Limited expressiveness
- Reasonable analysis possibilities



[Simao and Petrenko,
Fault Coverage-Driven Incremental Test Generation,
The Computer J., 2010]

[Kästner, Apel, Kuhlemann.
Granularity in Software Product Lines, ICSE 2008]

Modeling Variability: Annotational

- Less clear structure
- More expressive: (space and time variability)
- Difficulties with analyzing deletion and modifications

```
delta dStore {
    adds {
    }
}
```

```
delta dSingleStore {
    modifies store.Store {
    }
}
```

```
delta dMultiStore {
    modifies store.Store {
    }
}
```

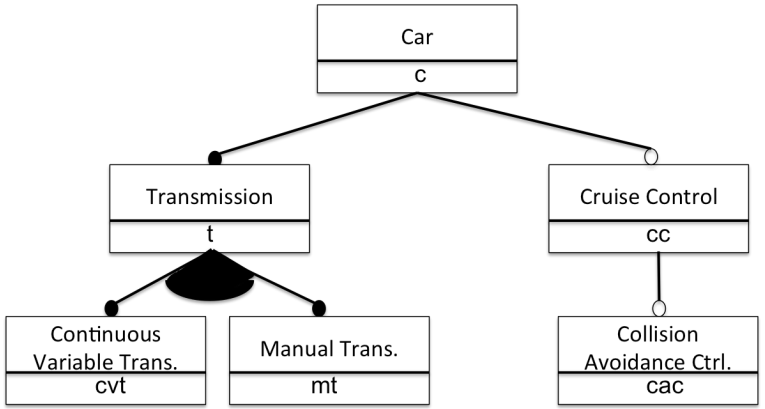
```
delta dAccessControl {
    modifies store.Store {
        adds private boolean sealed = false;
        modifies read()
            if(!sealed) {
                return original();
            }
        modifies readAll() {
            if(!sealed)
                return original();
            }
        modifies set(Object value) {
            if(!sealed)
                original();
            }
    }
}
```

[Varshosaz, Beohar, MRM,
Delta-Oriented FSM-Based Testing, ICFEM 2015]

[Schaefer et al. Delta-Oriented Prog., SPLS 2010]
[Apel et al. Strategies for product-line verification:
Case studies and experiments]

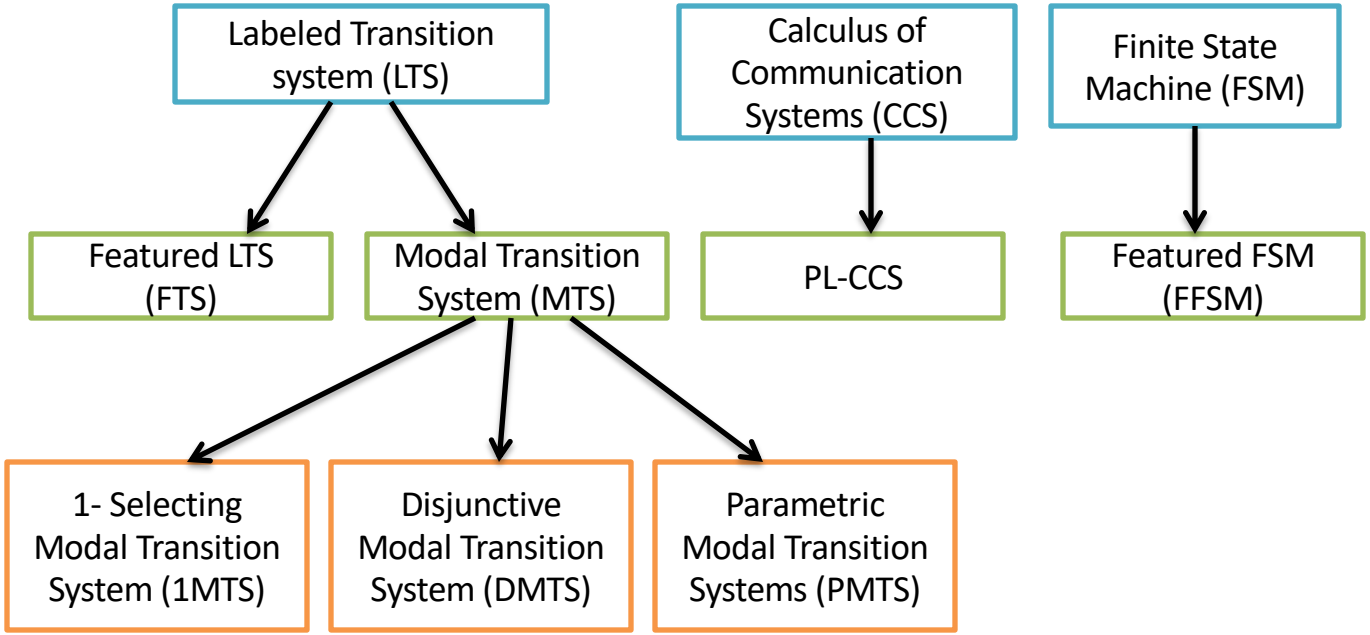
Modeling Variability: Compositional

- Clear structure
- Very expressive
- Difficult to implement and analyze

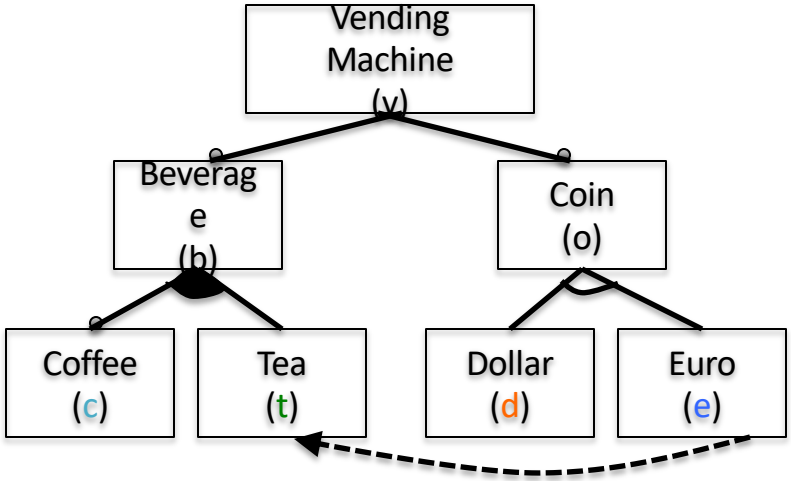


[Beohar and MRM, Spinal Test-Suited for Software Product Lines, MBT 2014]

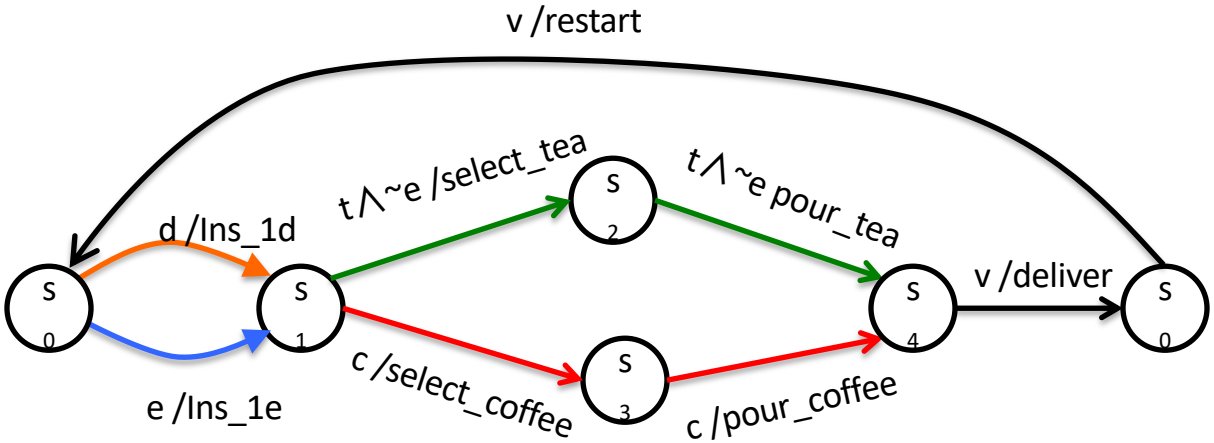
Formalisms for (Behavioral) Variability Modeling



Example: Vending Machine

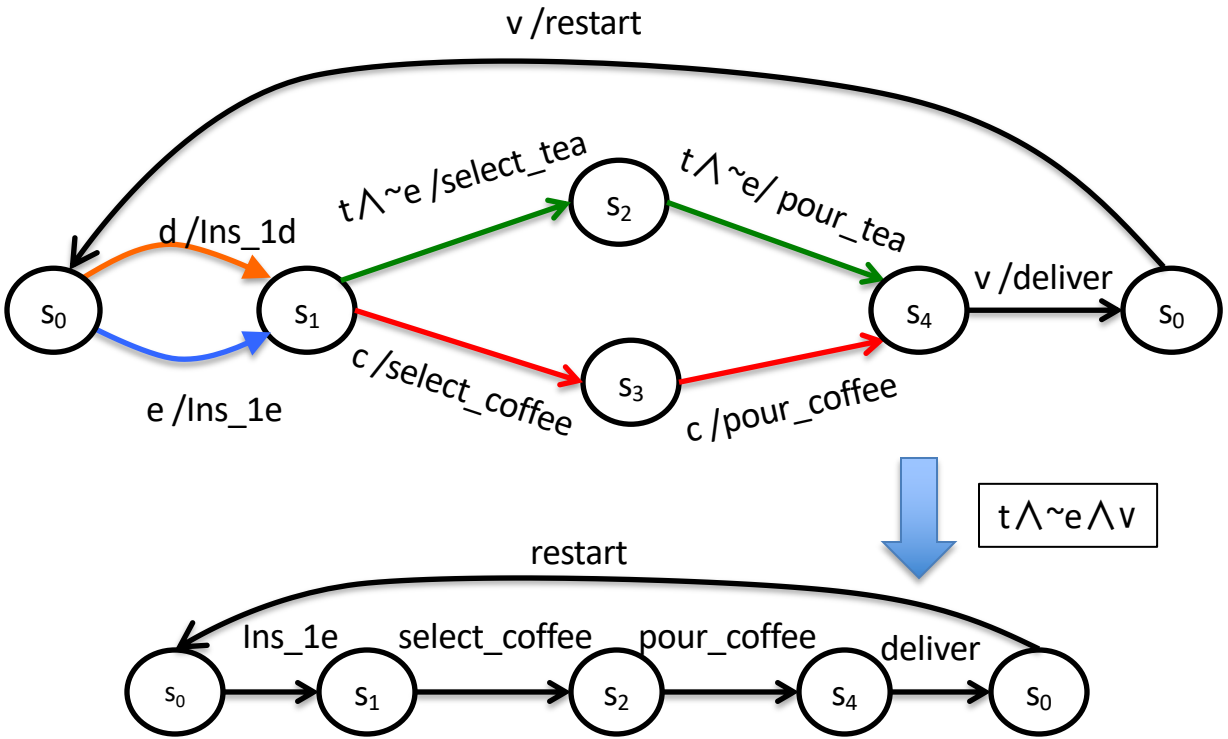


Featured Transition Systems



[A. Classen, M. Cordy, P.-Y. Schobbens, P. Heymans, A. Legay, J.-F. Raskin, Featured Transition Systems: Foundations for Verifying Variability-Intensive Systems and Their Application to LTL Model Checking]

Featured Transition Systems



PL-CCS

- An extension of CCS with binary variant operation

- Syntax

  Nil |α.e |e + e |e ⊕ e | e || e | e[f] | e\L

[A. Gruler, M. Leucker, K. Scheidemann,
Modeling and model checking software product lines]

[F. Ghassemi and MRM,
Product line process theory]

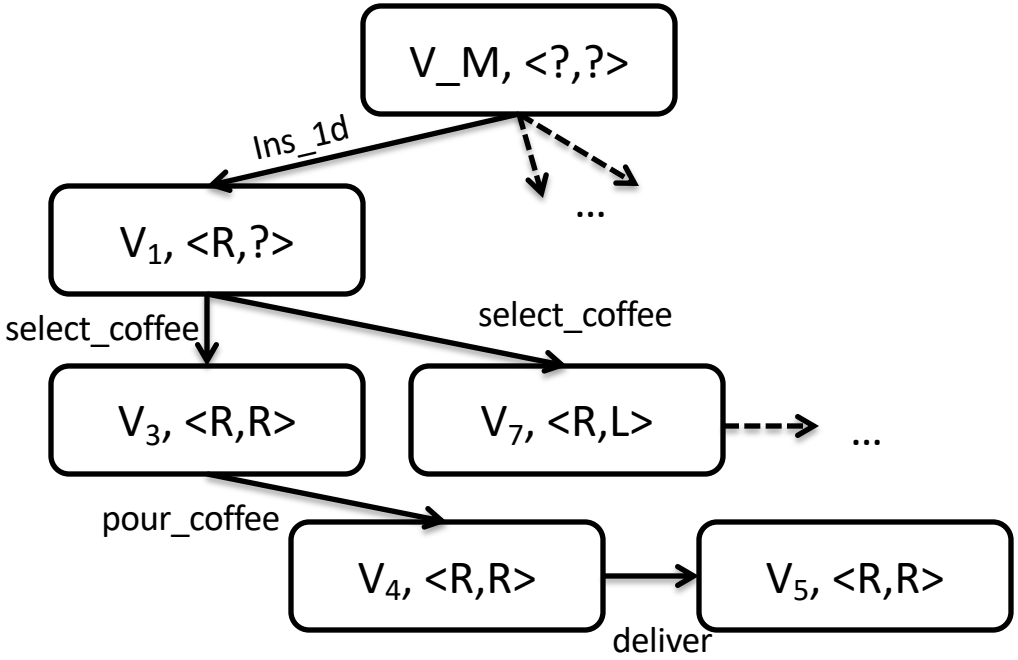
PL-CCS

Vending_Machine= Ins_1e.$V_1 \oplus_1$ Ins_1d.V_2

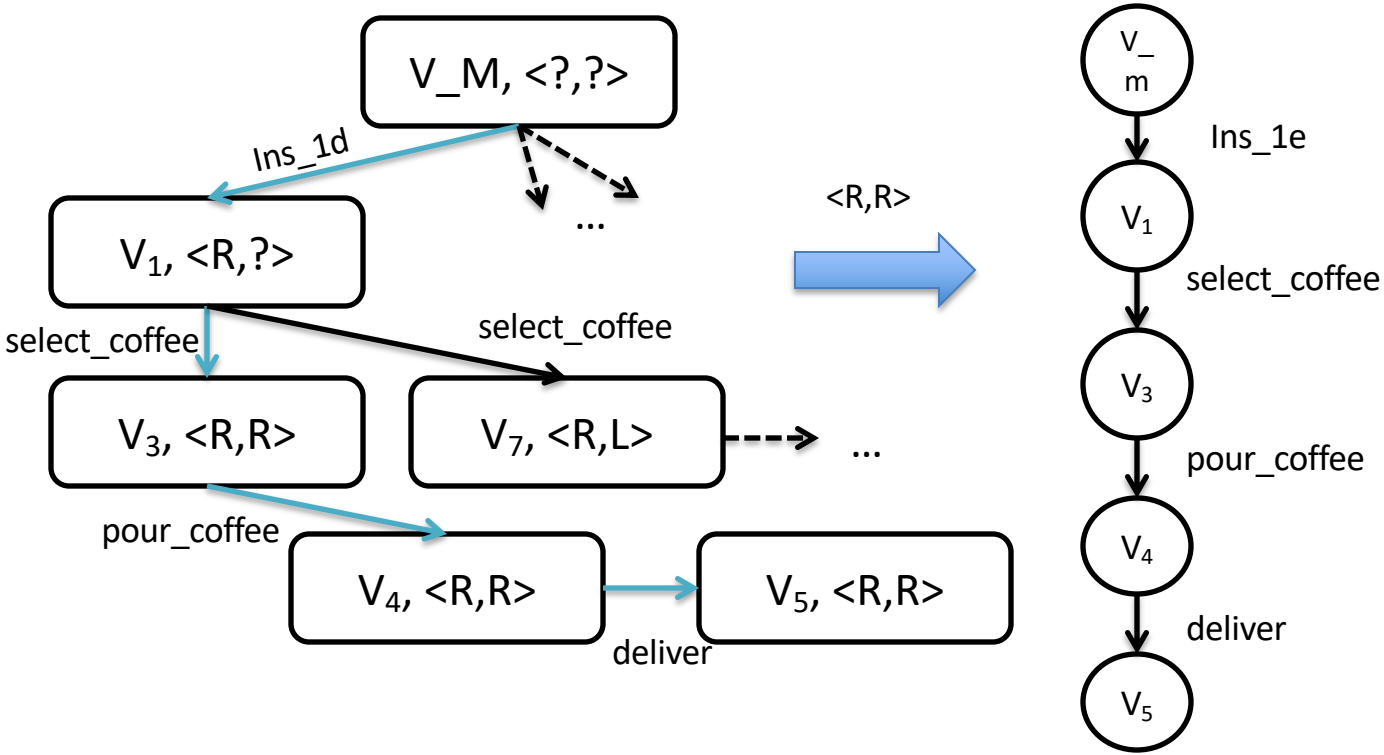
V_1= select_coffee.pour_coffee.deliver.restart.
Vending_Machine

V_2= (V_1+ select_tea.pour_tea.deliver.restart.
Vending_Machine) $\oplus_2$ V1

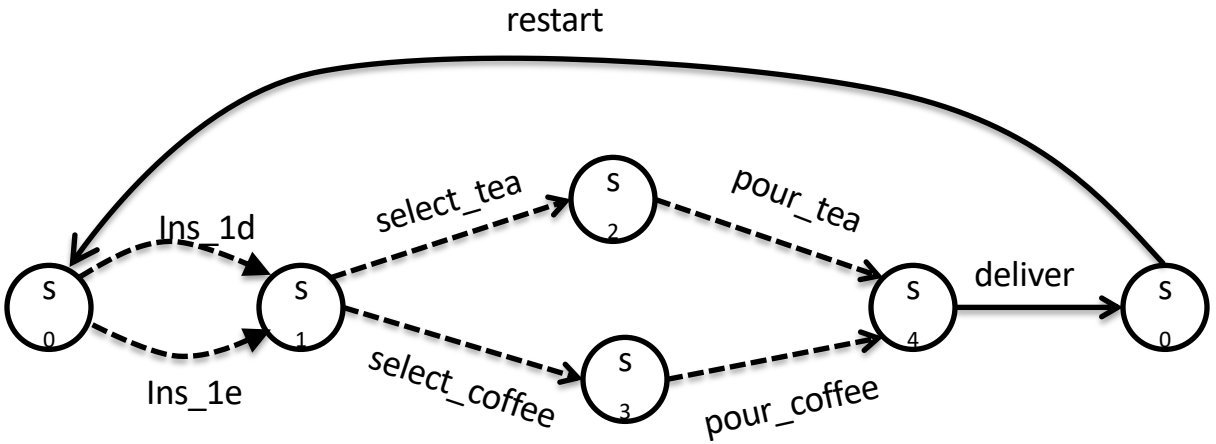
PL-LTSs: Semantic Domain for PL-CCSs



PL-LTSs: Semantic Domain for PL-CCS

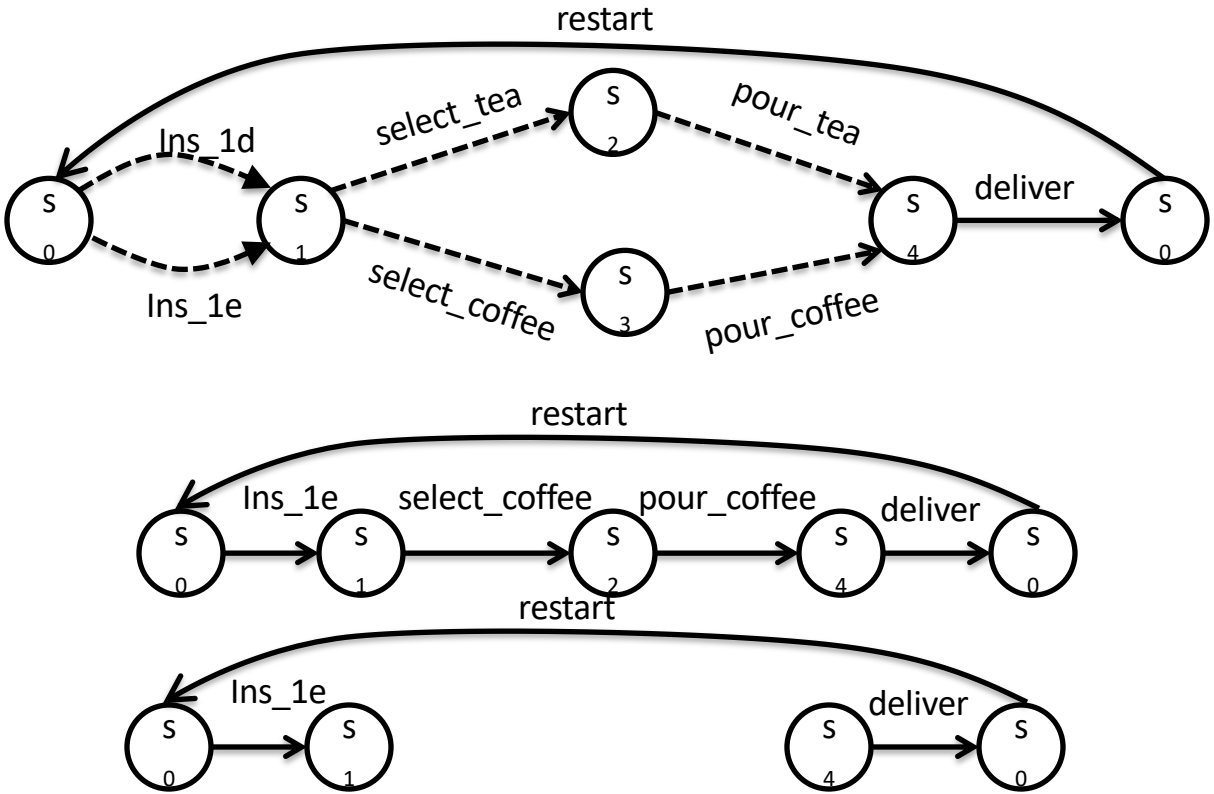


Modal Transition Systems



[Larsen and Thomsen, A modal process logic]

Modal Transition Systems



Comparison of the basic models

- Expressiveness: can they specify the same class of product lines?
- Technical methods:
  - Specifying “difficult” patterns
  - Encodings
    - Fully abstraction
    - Compositionality

[Felleisen, On the expressive power of programming languages]

[Nestmann, Fzzati, Mero, Modeling consensus in a process calculus]

[Palamidessi, Comparing the expressive power of the synchronous and the asynchronous pi-calculus]

[Gorla and Nestmann, Full abstractions for expressiveness: history, myths and facts]

Comparative Expressiveness

- Product line structure P =(M, [ ]), where
  - M: the class of product line models or specifications: MTSs, 1-MTSs, FTSs, and PL-CCSs, and
  - [ ]: M -> 2^{LTS} the semantic function from product line models to sets LTSs.

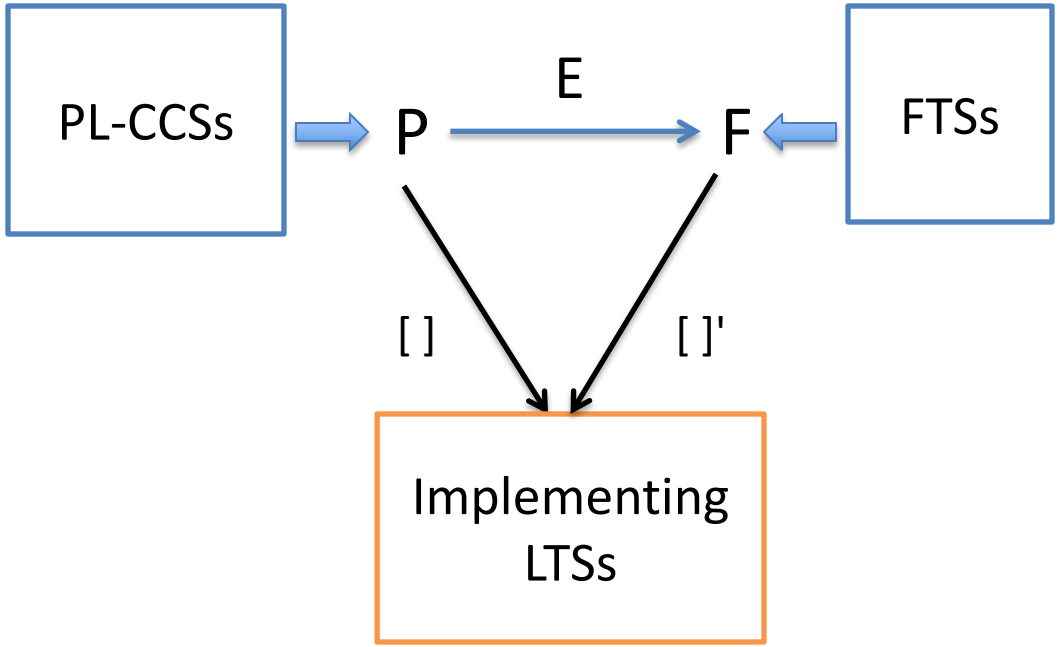
Comparative Expressiveness (cont…)

- For **M**=(M,[ ]) and **M**’=(M’,[ ]’),

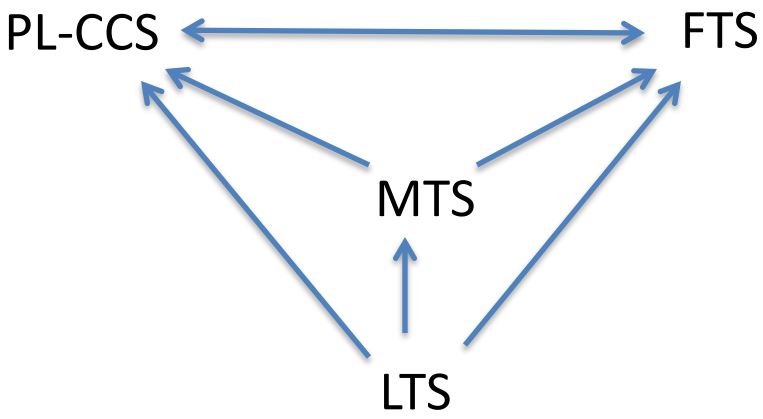
  E: M →M’ is an encoding iff [ ]= [ ]’ o E

- **M** is at-least as expressive as **M**’ iff
  there exists an encoding E: M →M'

Comparative Expressiveness (cont…)

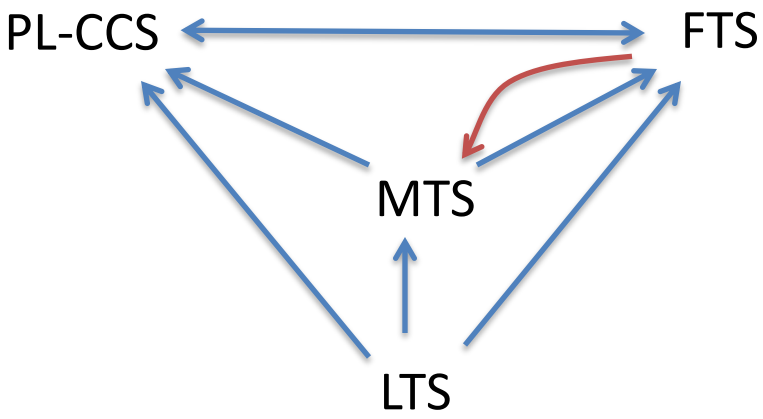


Comparative Expressiveness (cont…)



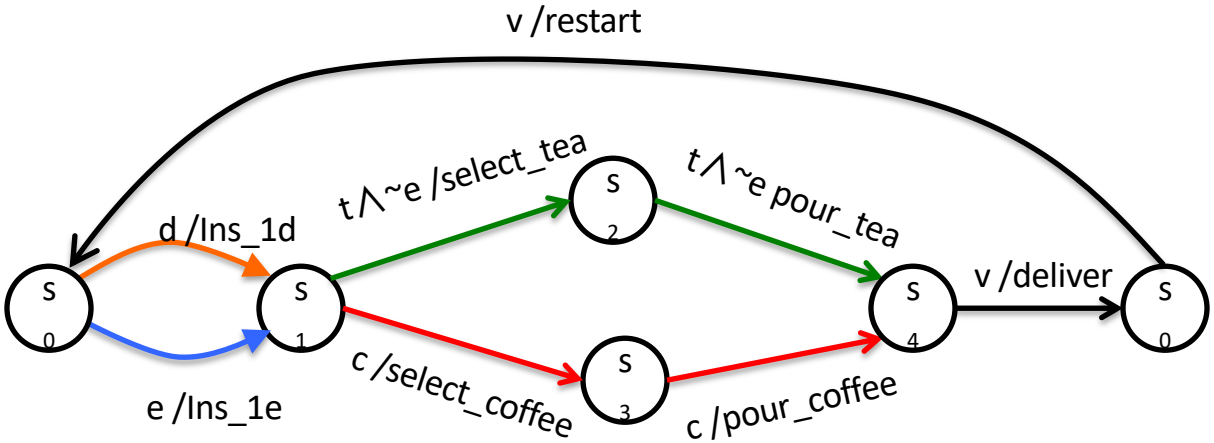
H. Beohar, M. Varshosaz. M. R. Mousavi, Basic Behavioral Models for Software Product Lines: Expressiveness and Testing Pre-Orders, SCP 2016

Comparative Expressiveness (cont…)



[M. Varshosaz, L. Luthmann, P. Mohr, M. Lochau, M.R. Mousavi.
Modal transition system encoding of featured transition systems. JLAMP 2019.]

Featured Transition Systems: Case for Expressiveness



[A. Classen, M. Cordy, P.-Y. Schobbens, P. Heymans, A. Legay, J.-F. Raskin, Featured Transition Systems: Foundations for Verifying Variability-Intensive Systems and Their Application to LTL Model Checking]

Succinctness

- Comparing the size of the models in terms of the number of states

- Comparison makes sense for formalisms with the same expressive power

Succinctness (cont…)

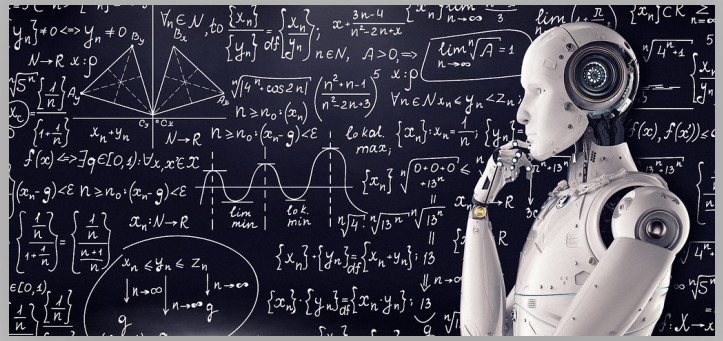
- Considering PL-CCS and FTS
  - Proved there exist a class of FTSs F such that |E(F)| (independent from the choice of E) is exponential in terms of |F|
  - Proved for any PL-LTS P the size of |E'(P)| is linear in terms of |P| (we can always define an encoding E')

H. Beohar, M. Varshosaz. M. R. Mousavi, Basic behavioral models for software product lines: Revisited. Sci. Comput. Program. 2018

Variability as a **first-class** citizen!

Modeling variability explicitly

Efficient testing using variability modeling

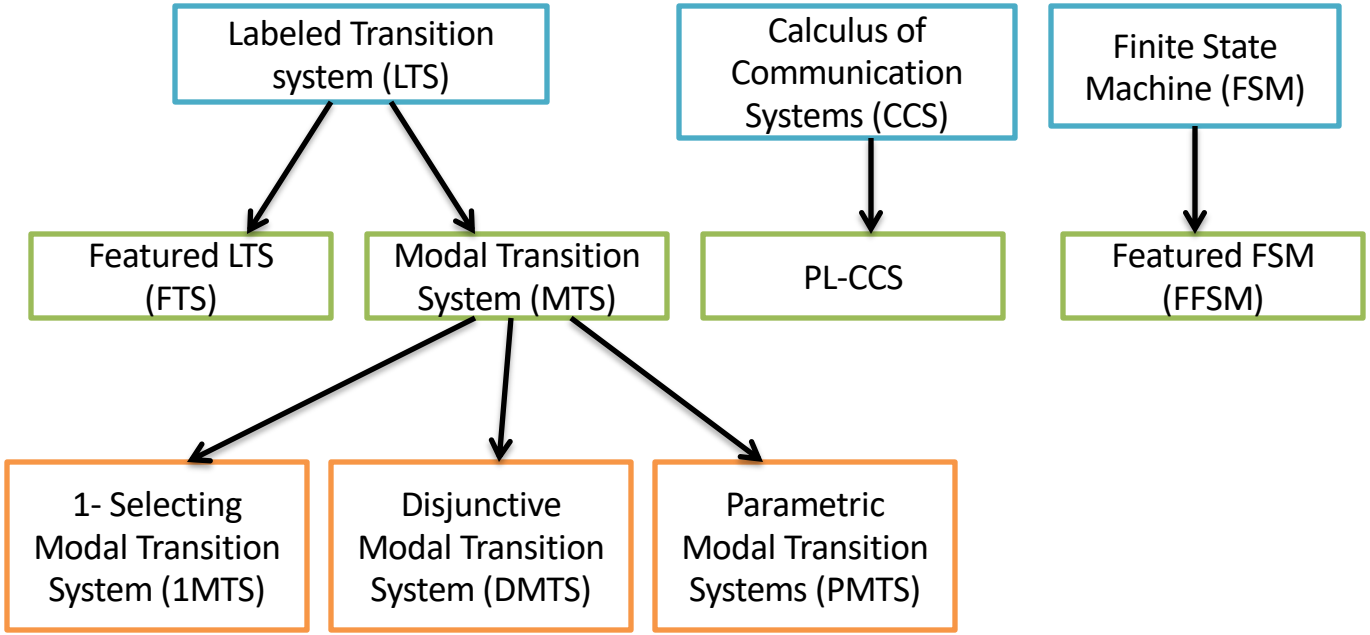


Learning about variability

UNIVERSITY OF LEICESTER

MODEL-BASED TESTING FOR VARIABILITY-INTENSIVE SYSTEMS

Formalisms for Variability

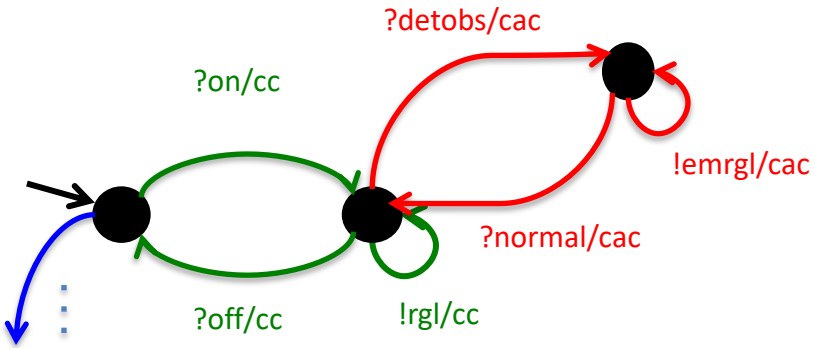


MBT for Featured Transition Systems

Features ={c,cc,cac,…}

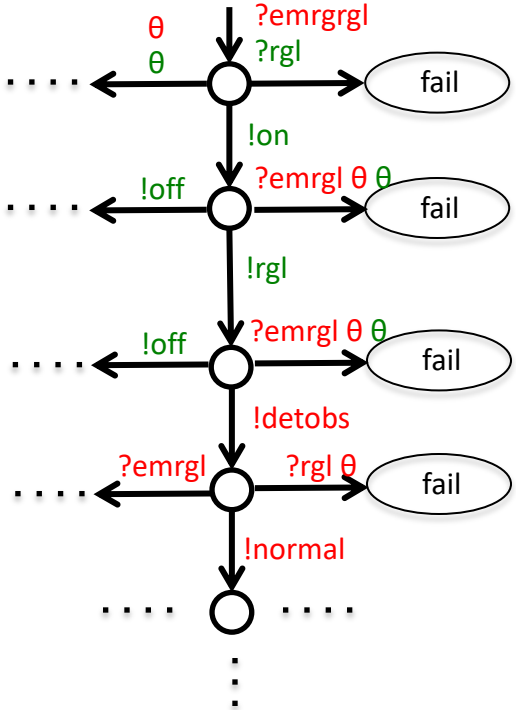
Valid product {c,cc,…}

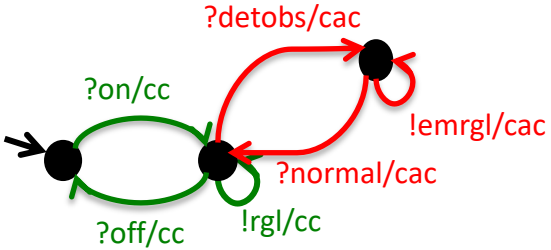
Valid product {c,cc,cac,…}



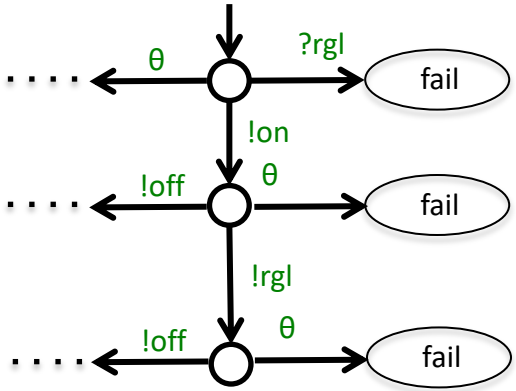
For a more serious model of cruise control, see:
M.A. de Langen, Vehicle Function Correctness, Masters Thesis, TU/e, 2012.

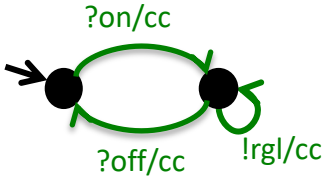
Test Suite


θ
θ
?emrgrgl
?rgl
fail
!on
!off
?emrgl θ θ
fail
!rgl
!off
?emrgl θ θ
fail
!detobs
?emrgl
?rgl θ
fail
!normal

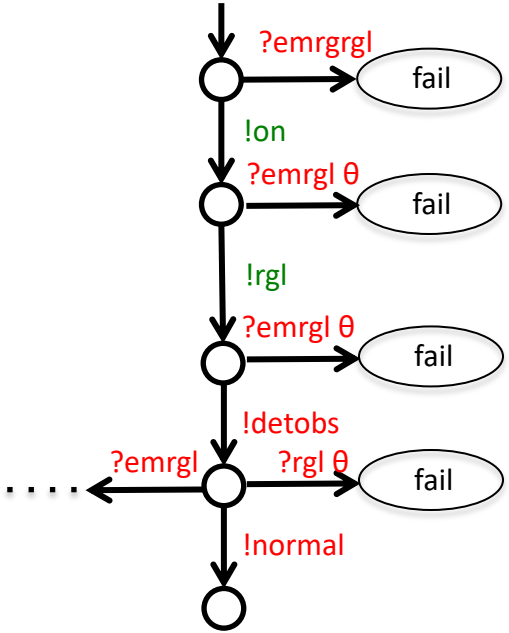

?detobs/cac
?on/cc
!emrgl/cac
?normal/cac
?off/cc
!rgl/cc

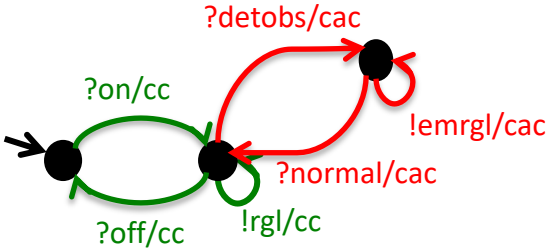
Test Suites: Projection





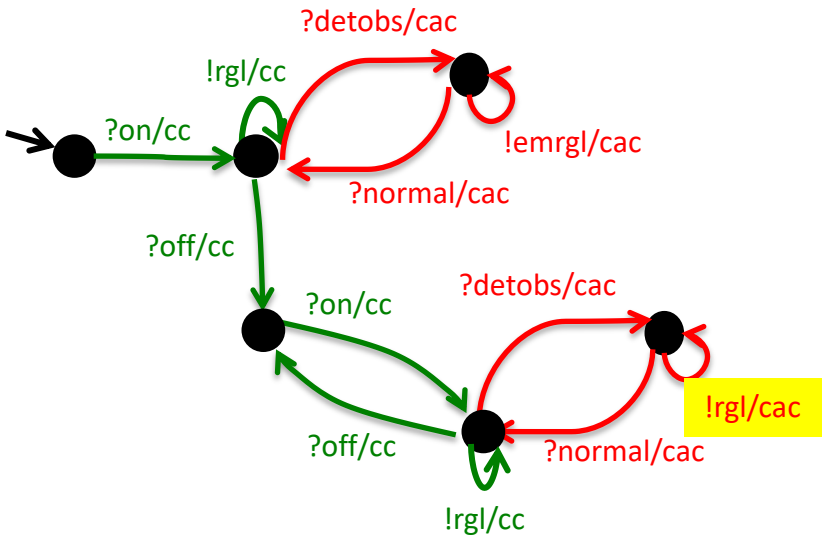
Test Suites: Residual Testing





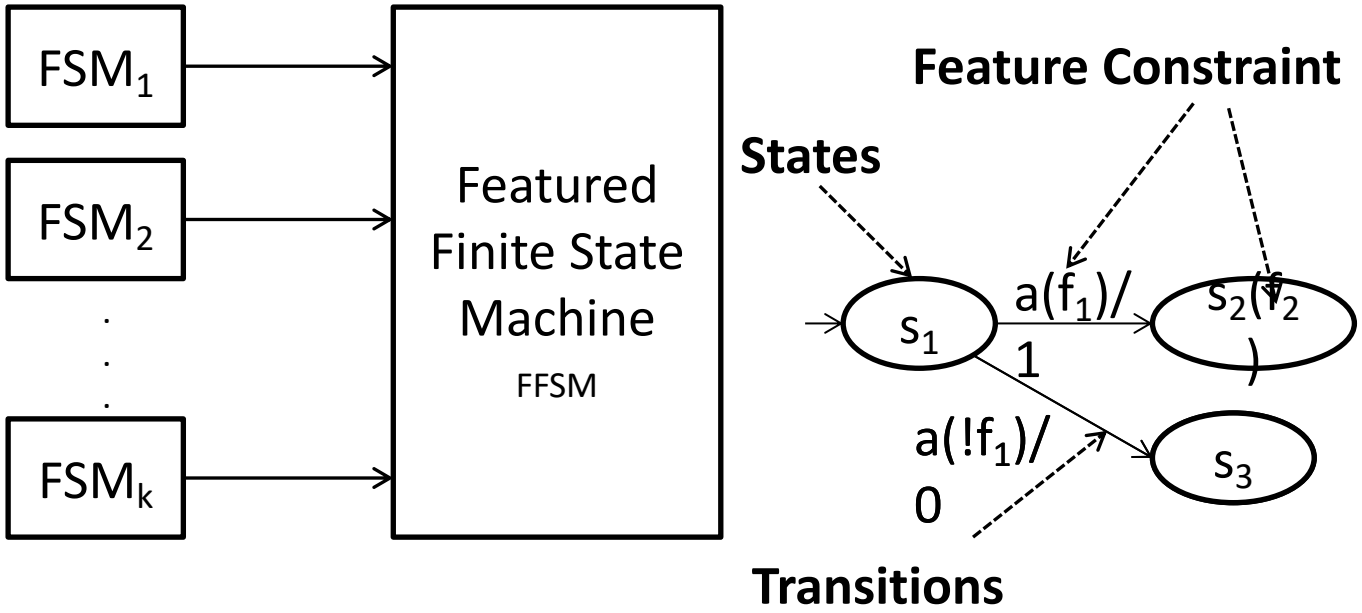
[H. Beohar, M.R. Mousavi.
Input-output conformance testing for software product lines. JLAMP. 2016]

Feature Interaction



[H. Beohar, M.R. Mousavi.
Input-output conformance testing for software product lines. JLAMP. 2016]

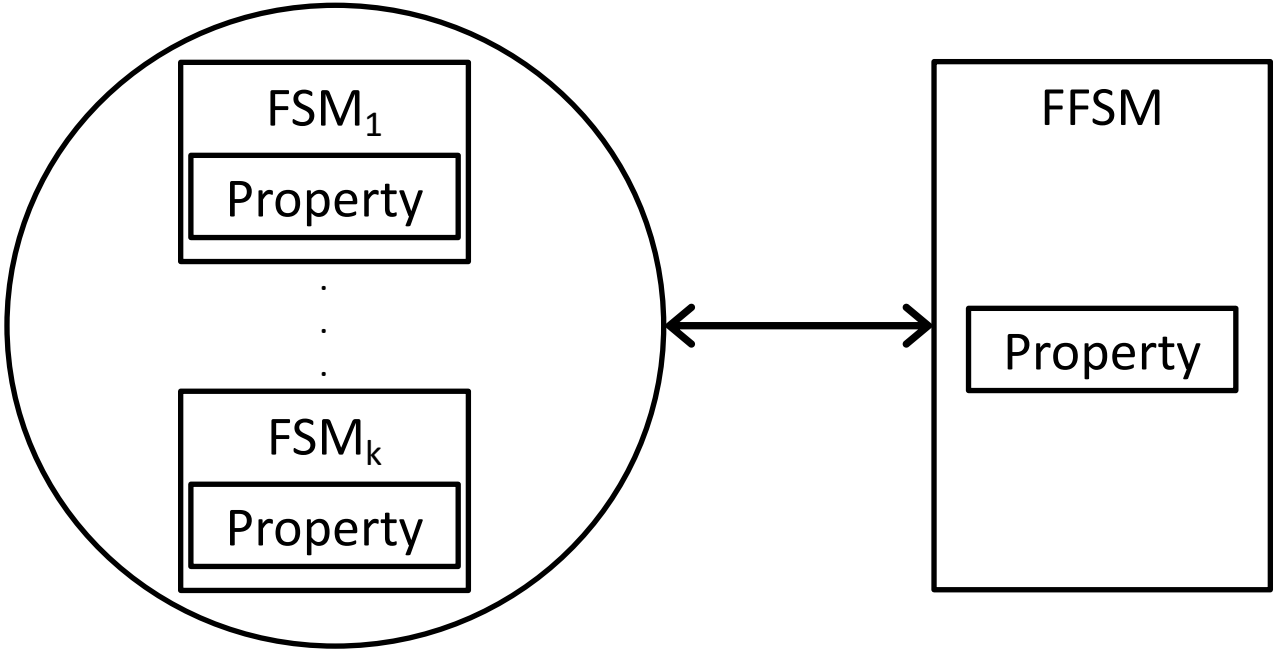
Featured FSM



[Hafemann, MRM, Simao, Featured Finite State Machines, Proc. FACS'17]
[Hafemann, MRM, Simao, Hieratchical Featured Finite State Machines, SCP'18]
[Hafemann, MRM, Simao and Turker. Extending HSI for Featured Finite State Machines, Computer J'19]

Model Validation

- Extend FSM properties to FFSM (hypothesis)
- Theorems - formal proofs



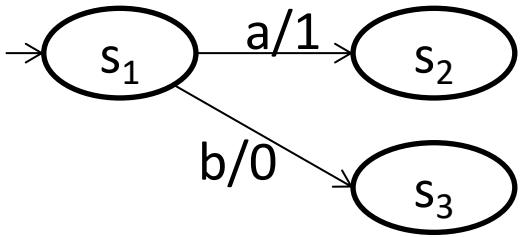
Specification Properties

- Test gen. methods (Full Fault Coverage Criteria)
- FSM Properties
  - *Deterministic*
  - *Complete (for some methods)*
  - *Initially Connected*
  - *Minimal (reduced)*

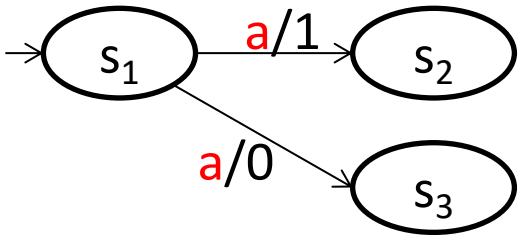
Specification Properties

- Extend FSM property to FFSM
  - *Deterministic FSM*

Deterministic

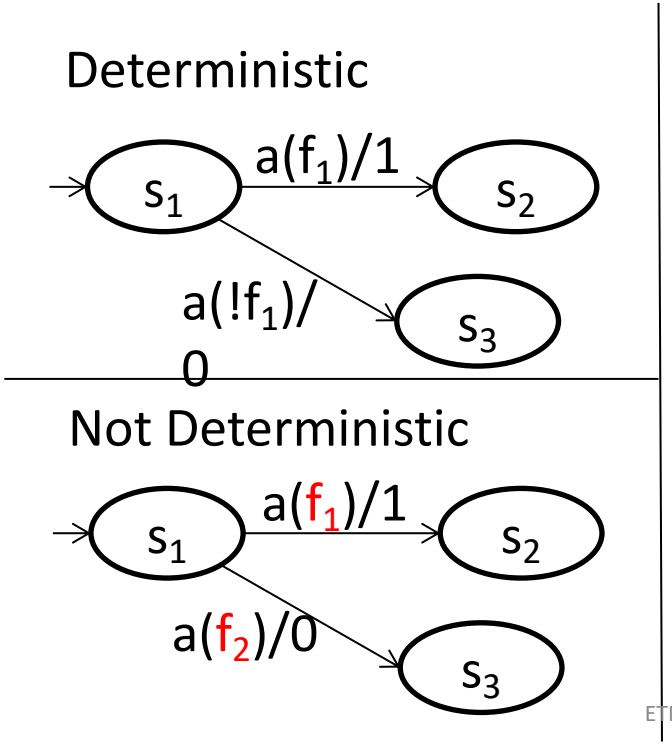


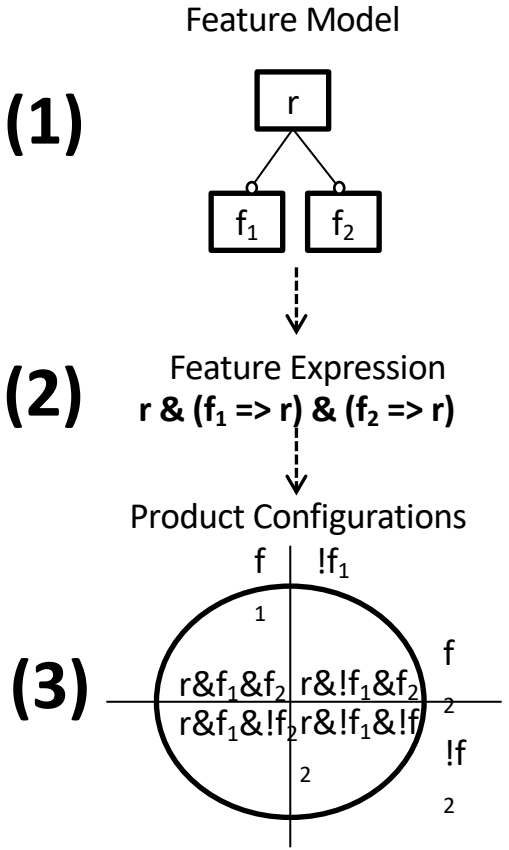
Not Deterministic



Specification Properties

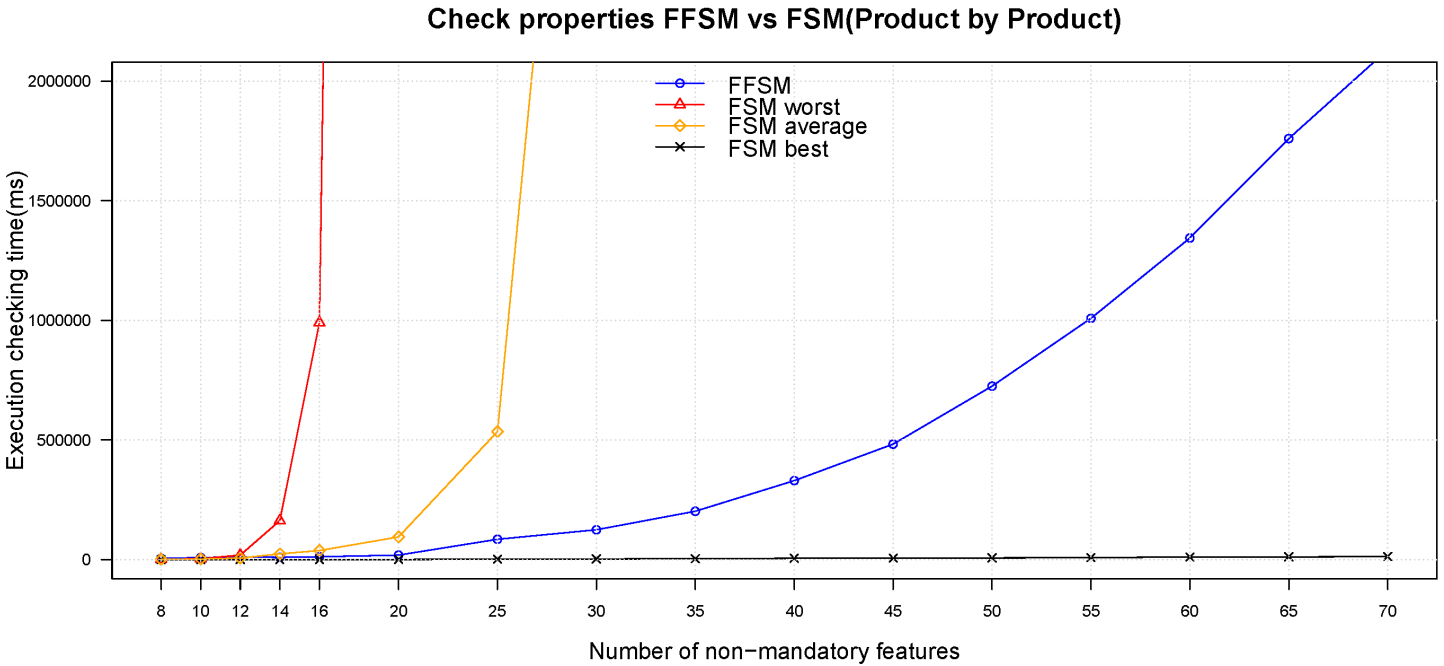
- Extend FSM property to FFSM
  - *Deterministic FFSM*





No product configuration should contain both transitions

Experimental Results



Experimental Results

- Case Study
  - Part of the Body Comfort System – BCS (State/Conf.)
  - 13 non mandatory features (8 are independent)
- Random feature models and FFSMs
  - 8 to 70 non-mandatory features
  - 8 to 70 conditional states/inputs
  - Map one conditional state per non-mandatory feature
- Execution time limit – 30 minutes
  - FFSM with 50 conditional states (approx. 8 minutes)
  - 384 FSMs (approx. 11 minutes)

Experimental Results

- One FFSM
  - Exceed time limit over 68 conditional states
- FSM worst case (2^k FSMs)
  - Exceed time limit over 16 conditional states
- FSM average case ($2^{k/2}$ FSMs)
  - Exceed time limit over 26 conditional states
- FSM best case (k+1 FSMs)
  - Does not exceed time limit

Modeling and MBT for Variability

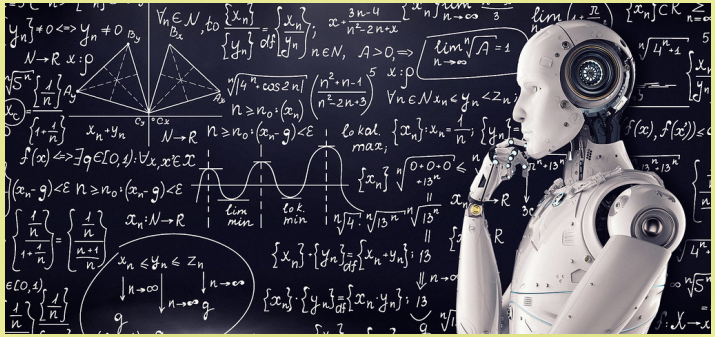
- Modeling product lines
  - Comparative expressiveness for fundamental models : PL-CCS, FTS, MTS, 1MTS
  - Featured FSMS: efficient validation techniques

- Model-based testing of product lines
  - Delta oriented modification of FSM-based testing algorithms, minimizing concretization effort.
  - Input/output conformance testing of IOFTSs
    - Reducing test effort using residual test suits

Variability as a **first-class** citizen!

Modeling variability explicitly

Efficient testing using variability modeling



Learning about variability

UNIVERSITY OF LEICESTER

LEARNING VARIATION IN SPACE

JOINT WORK WITH:

DIEGO DAMASCENO

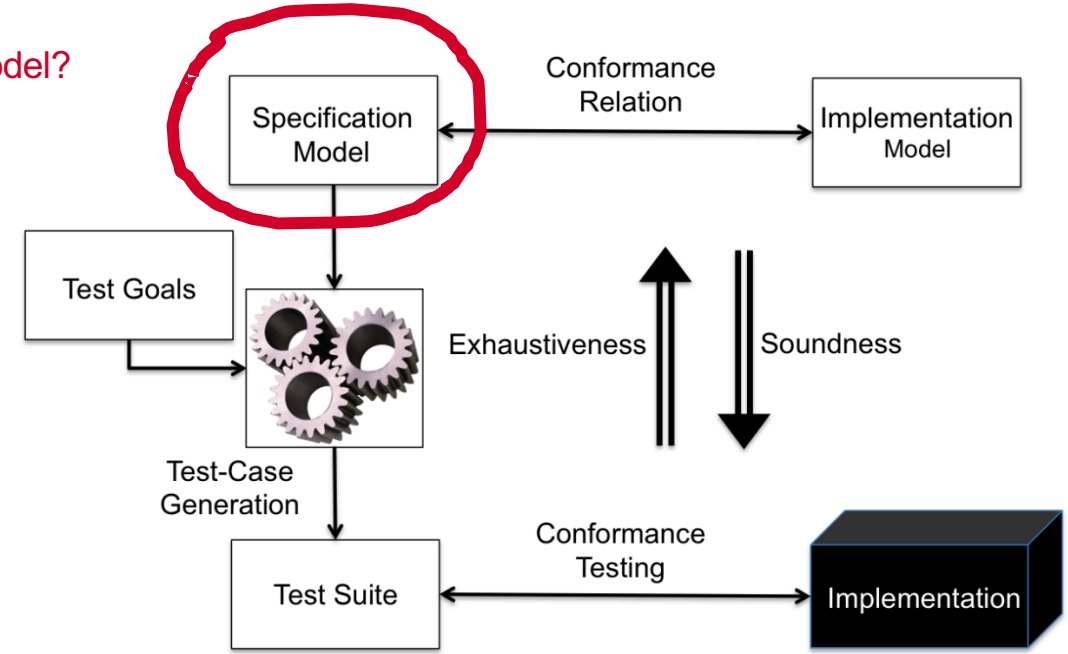
Radboud University

ADENILSO SIMAO

USP Universidade de São Paulo

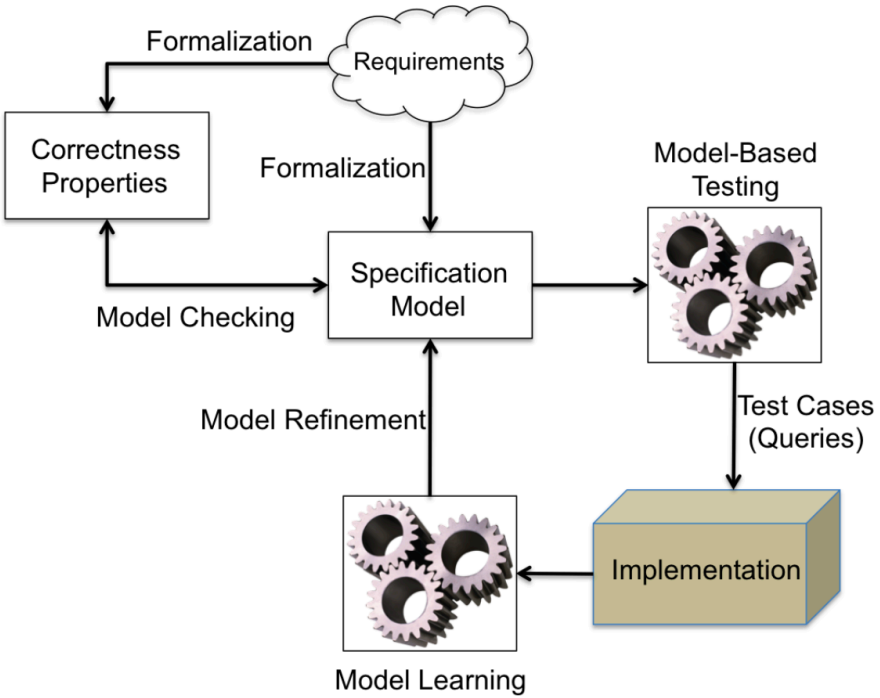
Active Learning: Why?

Model? What model?



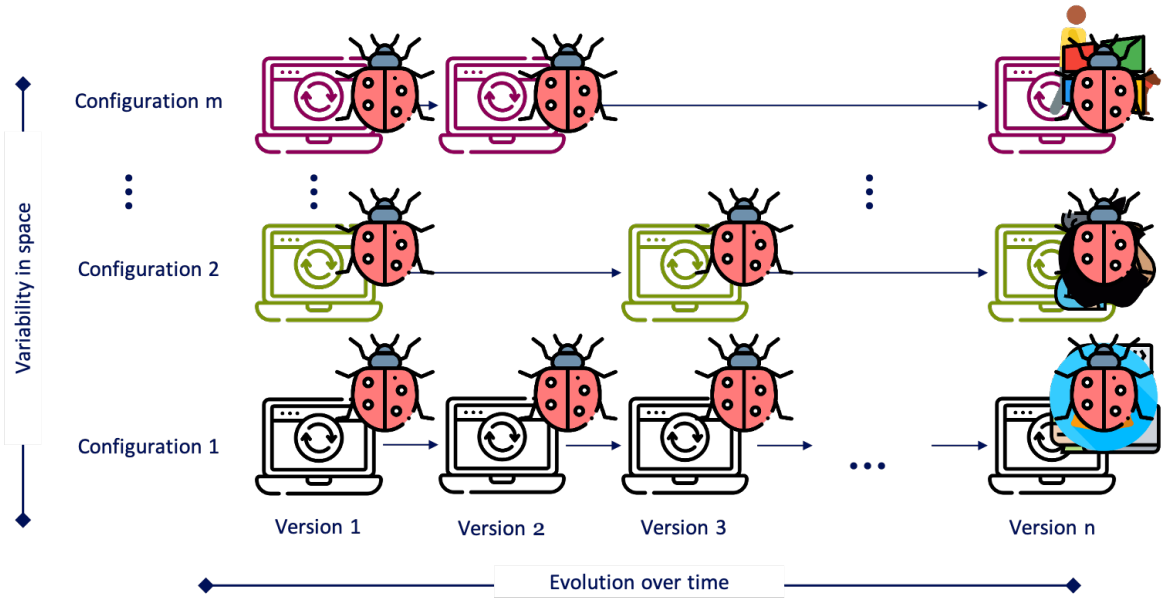
[Aichernig, Mostowski, Mousavi, Tappler and Taromiras, Model Learning and Model-Baed Testing]

Model Learning: Why?



[Aichernig, Mostowski, Mousavi, Tappler and Taromirad. Model Learning and Model-Baed Testing]

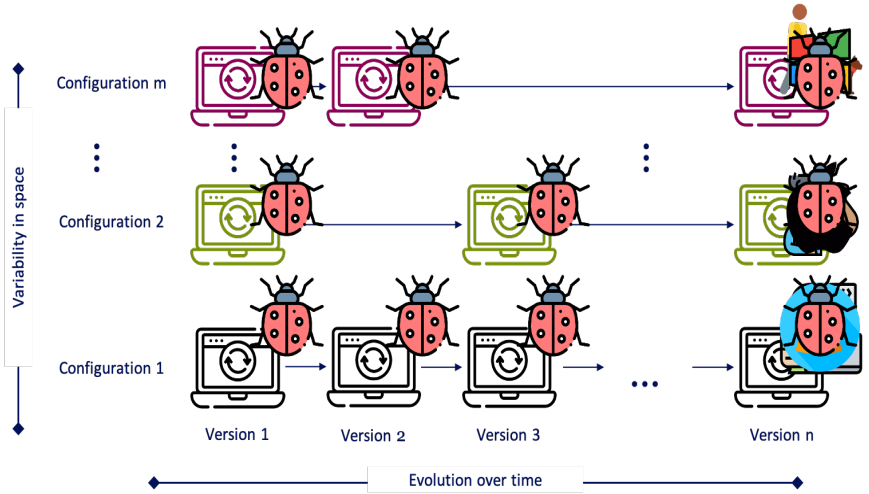
Why?


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

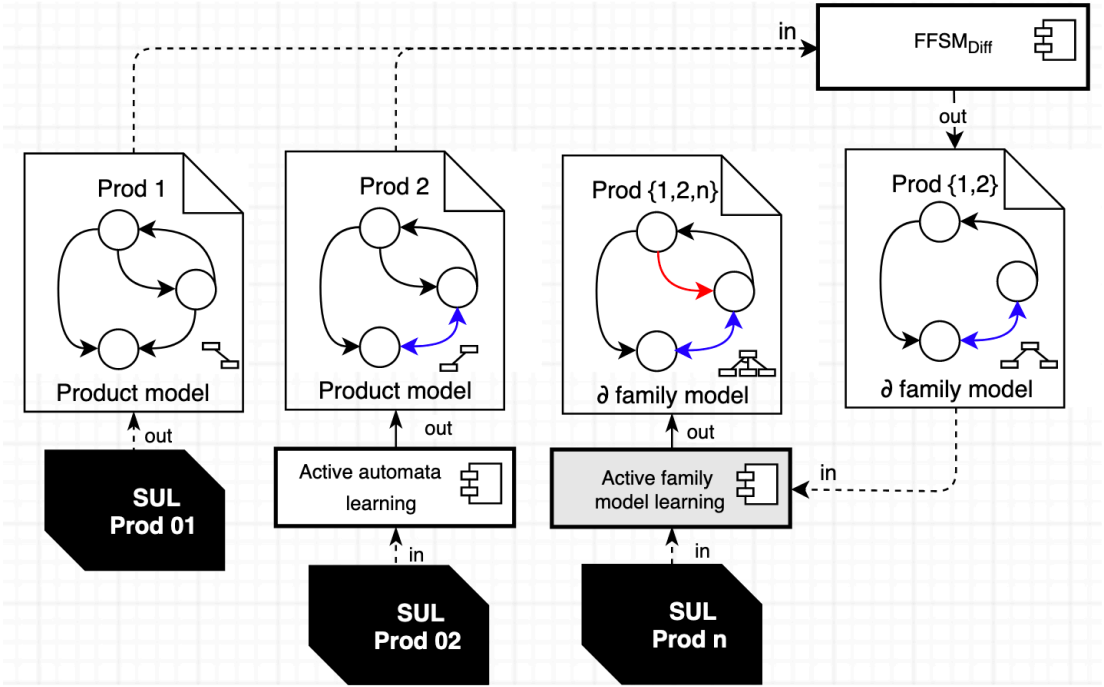
Why?

How can we succintly learn and summarise the variability model?

How sensitive is it to the number of configuration samples?



What?



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

HoW?

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

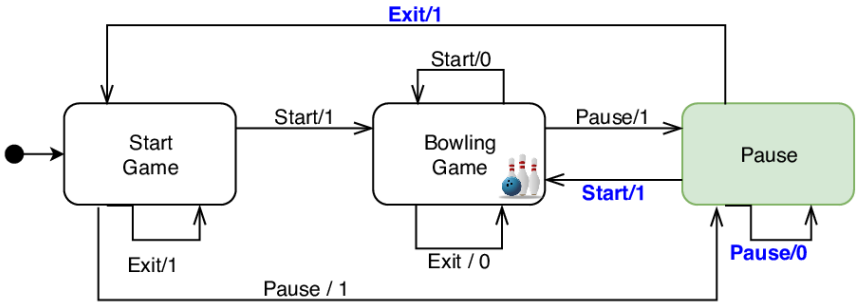
Global similarity score (Outgoing and incoming transitions)

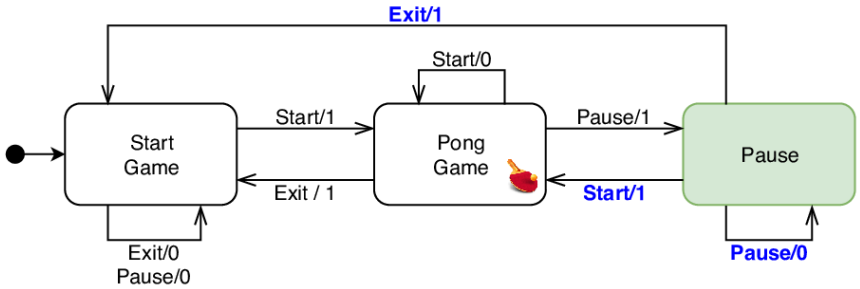
- Pairwise similarity based on surrounding matching transitions and connected state pairs.
- Attenuation ratio k gives precedence to the closest state pairs.
- Matching transitions and distinct transitions.

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

How?

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





The $FFSM_{Diff}$ algorithm

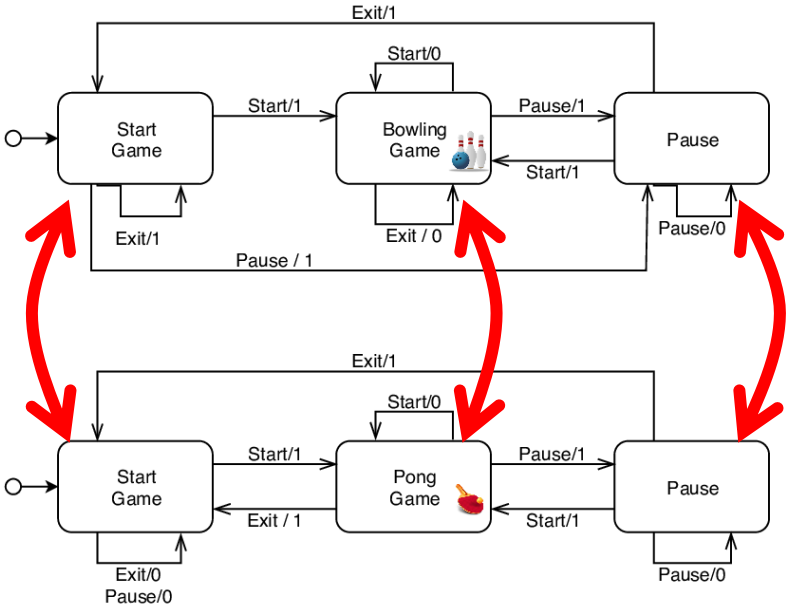


Figure: Two examples of product FSMs

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

$pair(Bo, Pa) = 0$

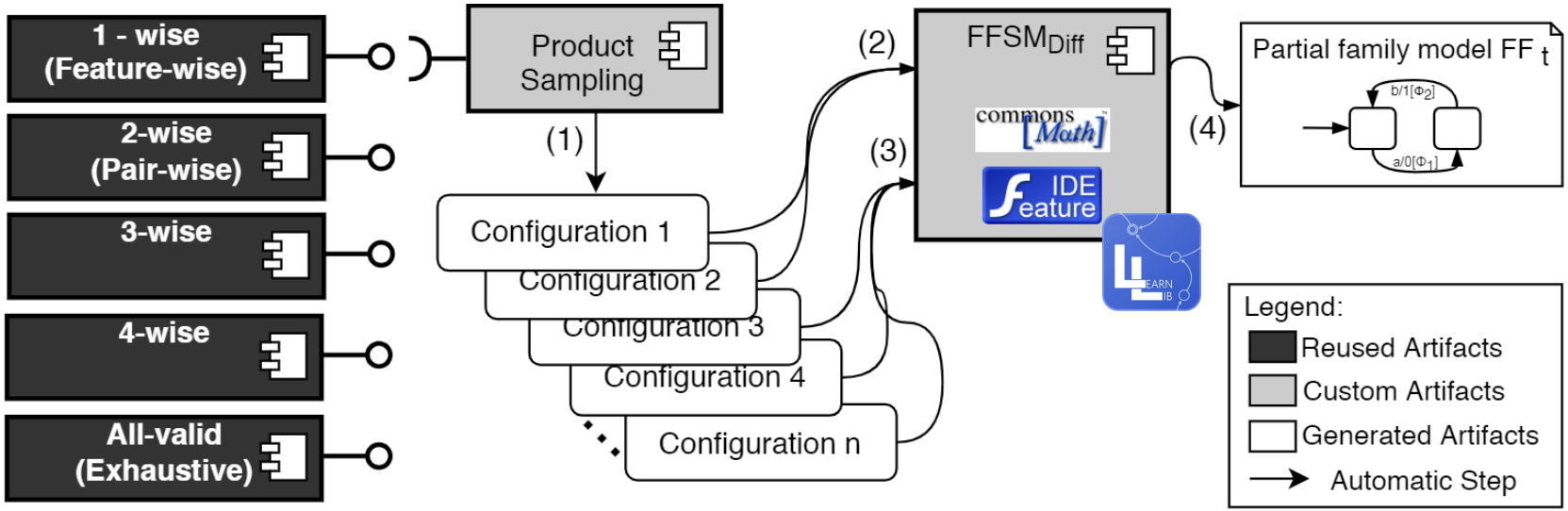
$pair(Pa, St) = 0.29$

$pair(Pa, Po) = 0.11$

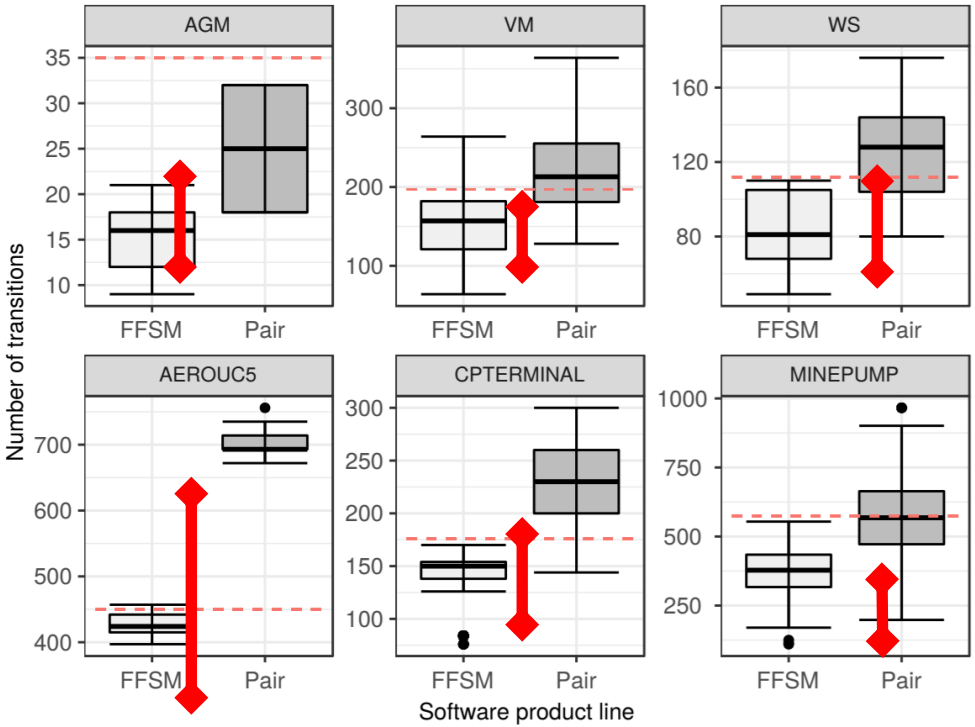
pair(Pa, Pa) = 0.58

Figure: Pairwise state similarity

Experiment Design (cont.)

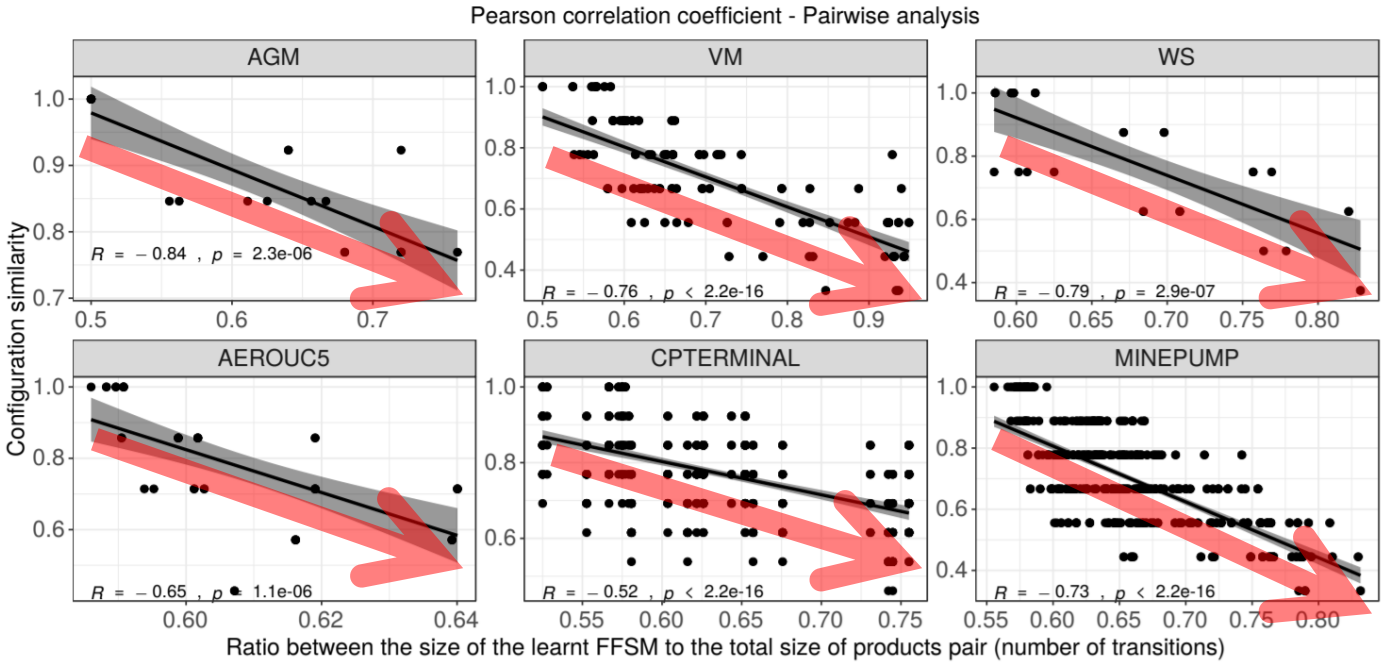


Analysis of Results: Size



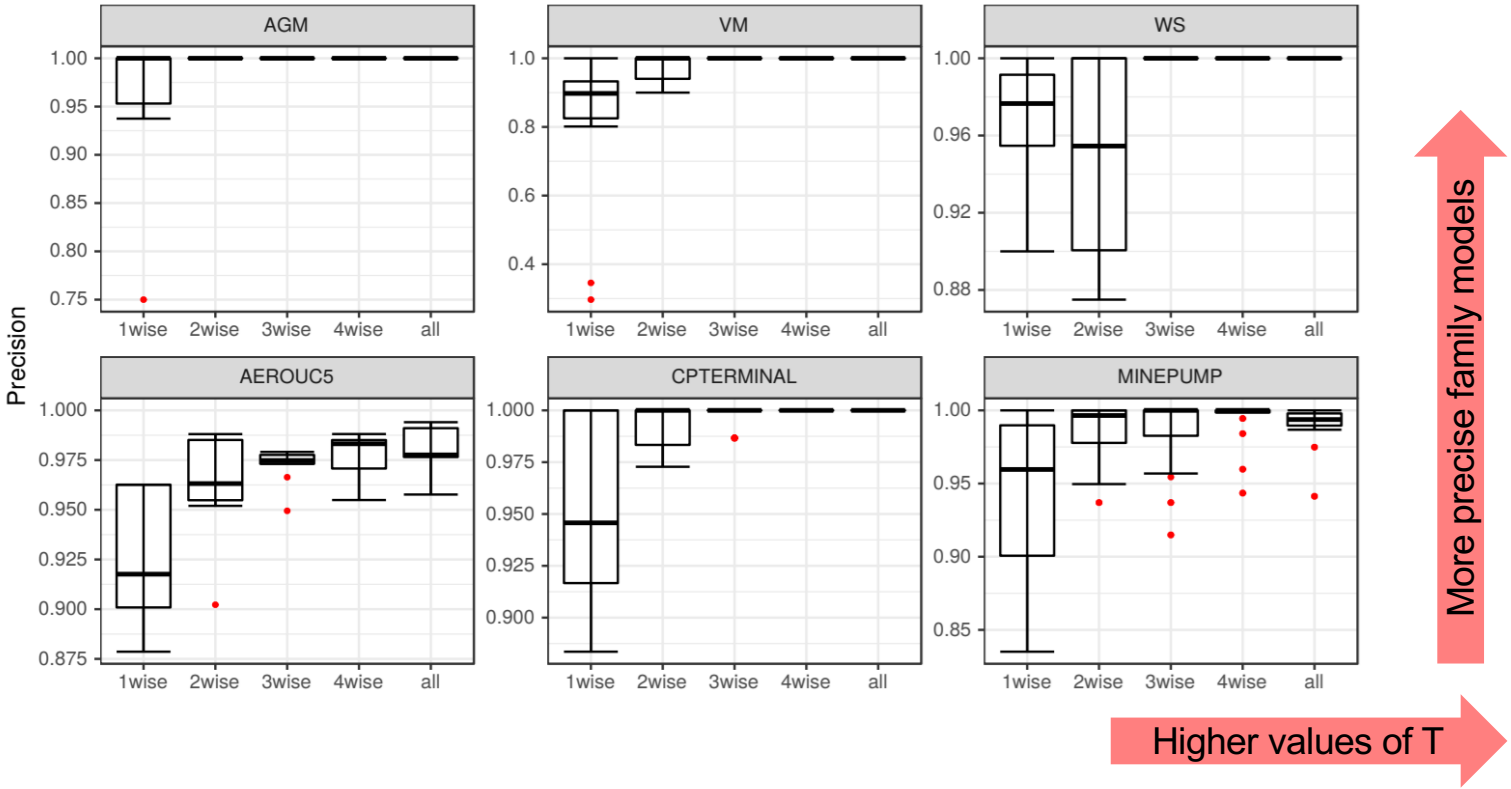
Mann-Whitney-Wilcoxon statistical test
Vargha-Delaney's Â effect size

Analysis of Results



Pearson correlation coefficient

Analysis of Results: Sampling

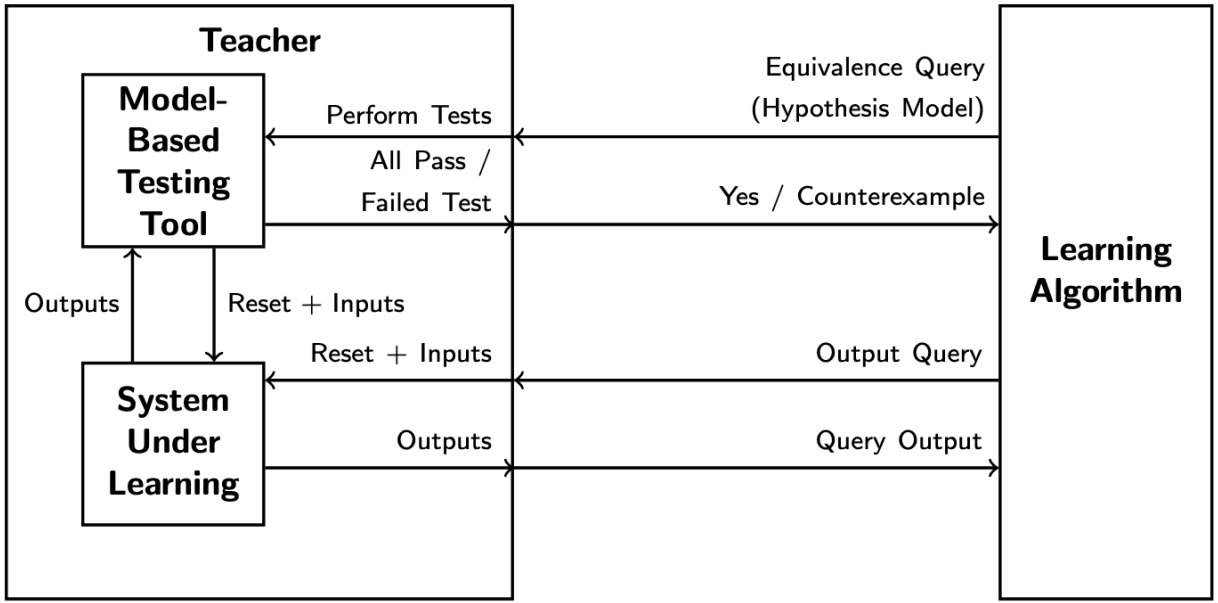


Mann-Whitney-Wilcoxon statistical test
Vargha-Delaney's Â effect size

UNIVERSITY OF LEICESTER

LEARNING VARIABILITY IN TIME

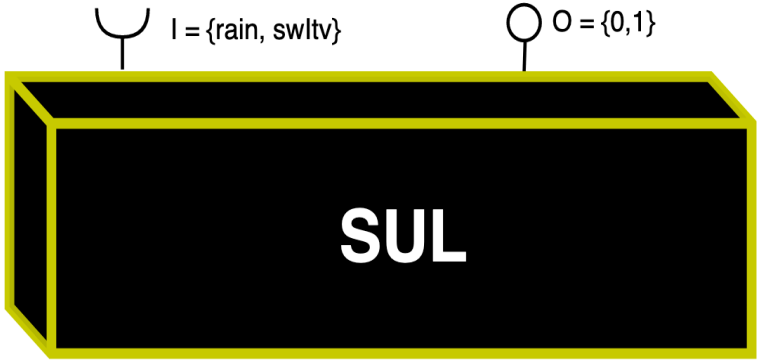
ACTIVE LEARNING



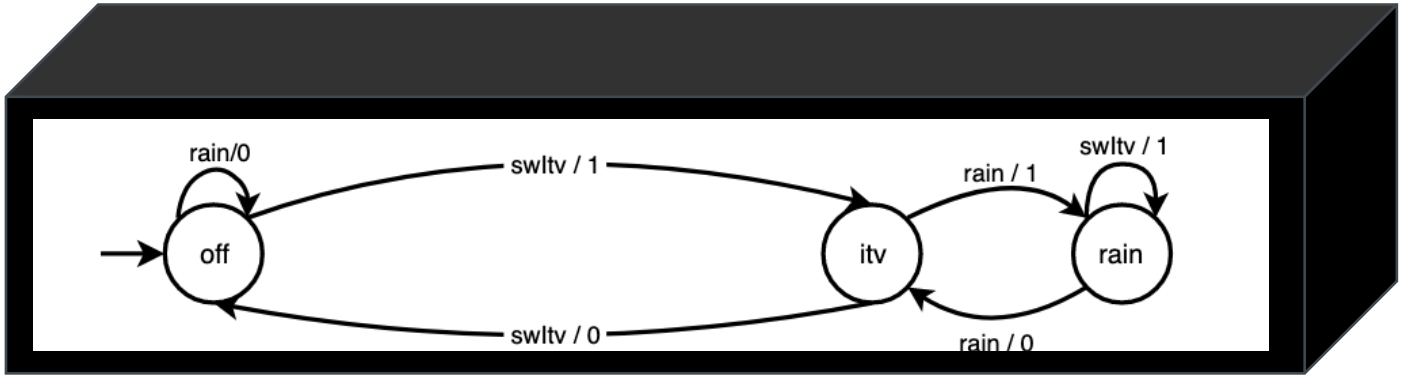
[Dana Angluin. Learning regular sets from queries and counterexamples.]

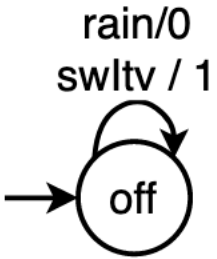
UNIVERSITY OF LEICESTER

ACTIVE LEARNING: HOW?



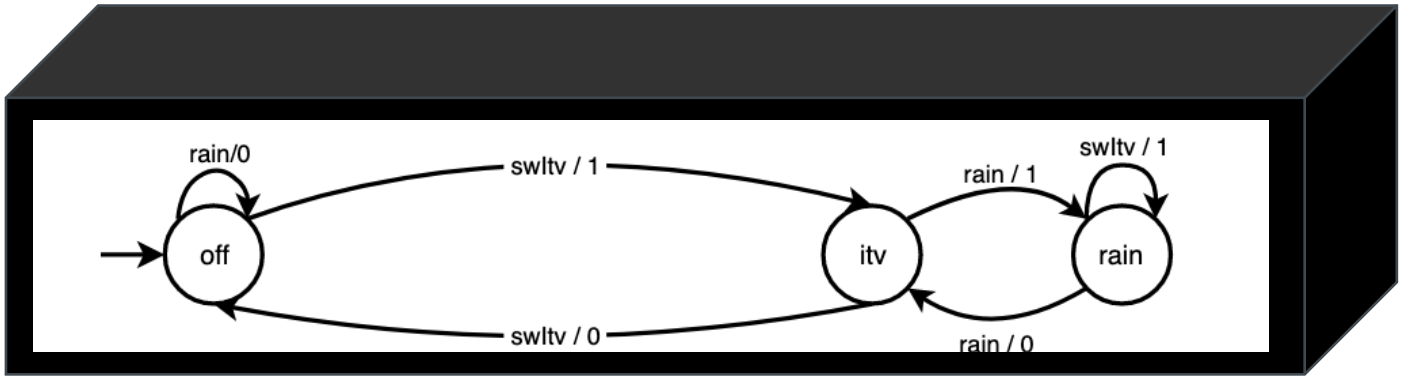
ACTIVE LEARNING: HOW?

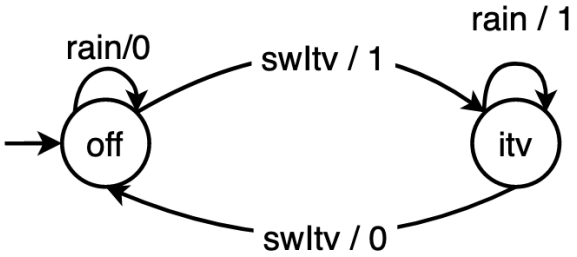




		rain	swItv
S	ϵ	0	1
$S \cdot I$	rain	0	1
	swItv	1	0

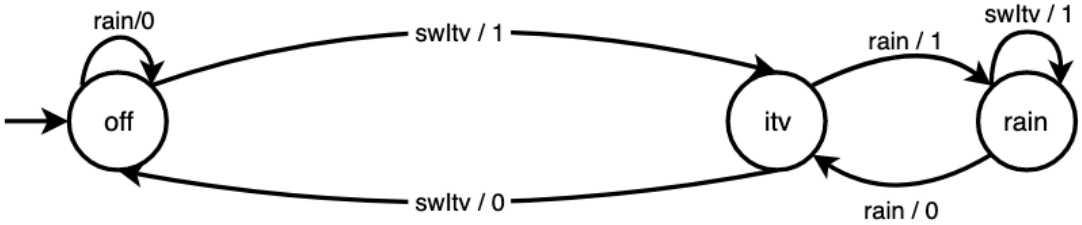
ACTIVE LEARNING: HOW?





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

ACTIVE LEARNING: HOW?



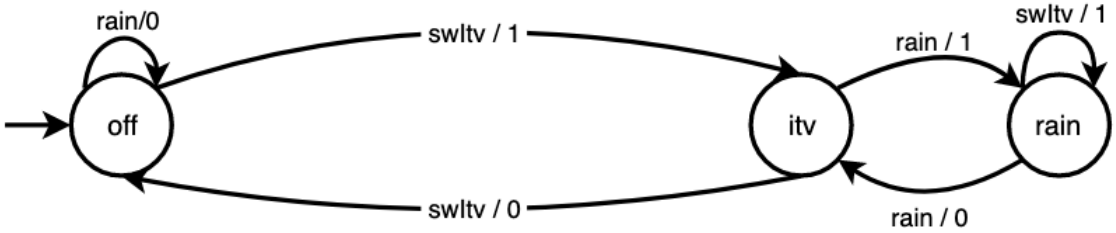
		$rain$	$swItv$	$rain \cdot rain$
S_r	ϵ	0	1	$0 \cdot 0$
	$swItv$	1	0	$1 \cdot 0$
	$swItv \cdot rain$	0	1	$0 \cdot 1$
$S_r \cdot I_r$	$rain$	0	1	$0 \cdot 0$
	$swItv \cdot swItv$	0	1	$0 \cdot 0$
	$swItv \cdot rain \cdot rain$	1	0	$1 \cdot 0$
	$swItv \cdot rain \cdot swItv$	0	1	$0 \cdot 1$

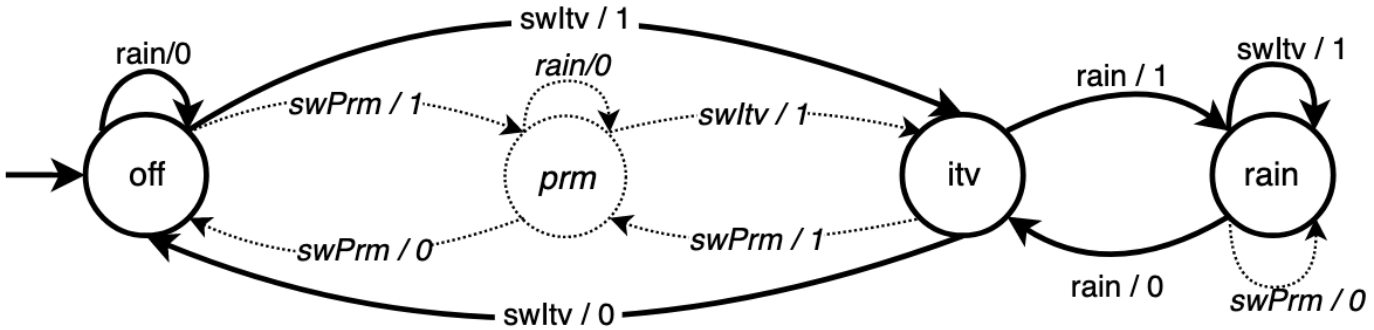
Consistent: $\forall p \in S_r \,.\, I_r \;\exists p' \in S_r \;\cdot\; p \cong p'$

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

UNIVERSITY OF LEICESTER

WHY?

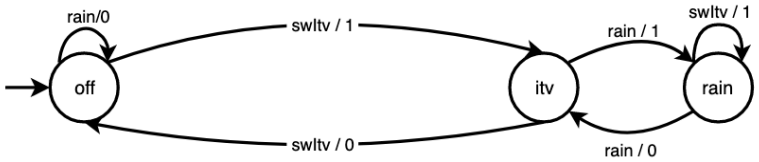

rain/0
swItv / 1
swItv / 1
rain / 1
off
itv
rain
swItv / 0
rain / 0

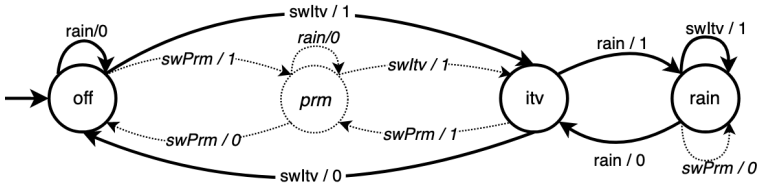

swItv / 1
rain/0
swItv / 1
rain / 1
rain/0
swPrm / 1
swItv / 1
off
prm
itv
rain
swPrm / 0
swPrm / 1
rain / 0
swPrm / 0
swItv / 0

WHAT?

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

How sensitive is it to the amount of evolution?

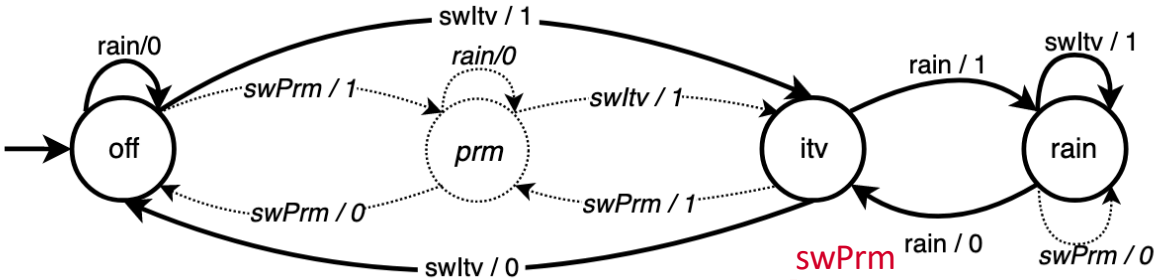




[Groce, Peled, and Yannakakis. Adaptive model checking. 2002]
[Chaki, Clarke, Sharygina, Sinha.
Verification of evolving software via component substitutability analysis. 2008]

UNIVERSITY OF LEICESTER

HOW?



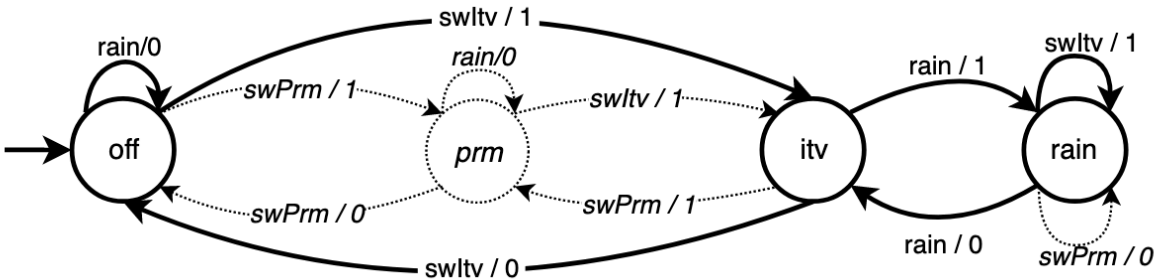
Anything redundant?

swPrm

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

UNIVERSITY OF LEICESTER

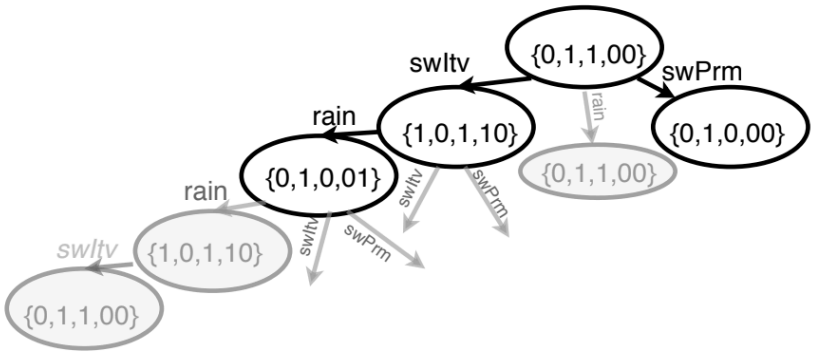
HOW?



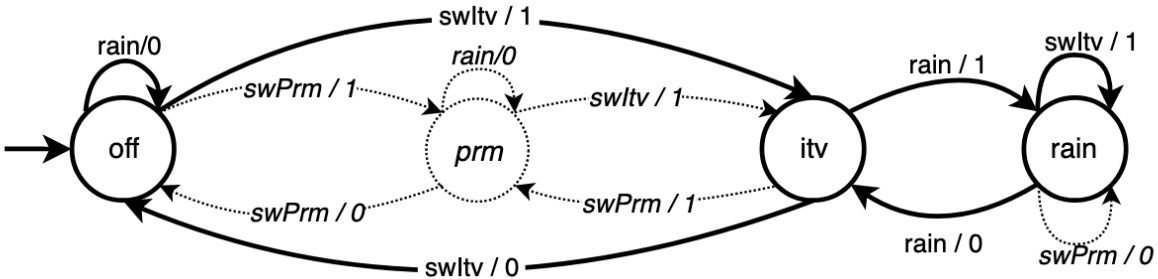
Anything redundant?

swPrm

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



HOW?



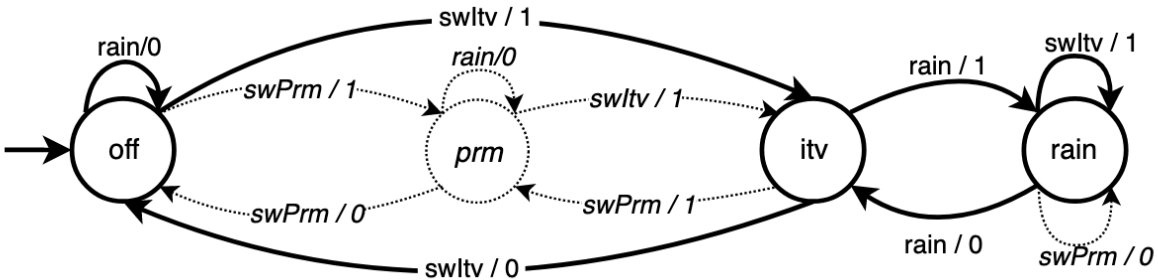
swPrm

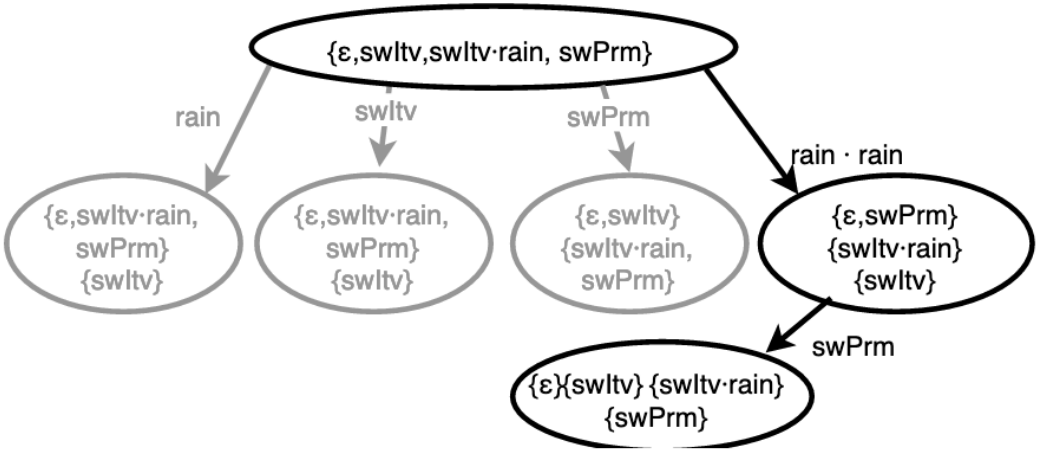
New experiments?

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

UNIVERSITY OF LEICESTER

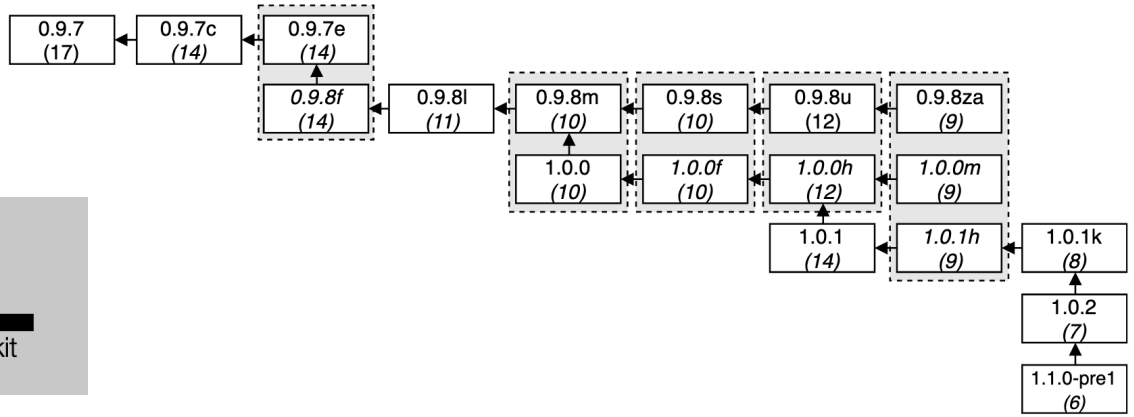
HOW?





UNIVERSITY OF LEICESTER

DOES IT REALLY WORK?



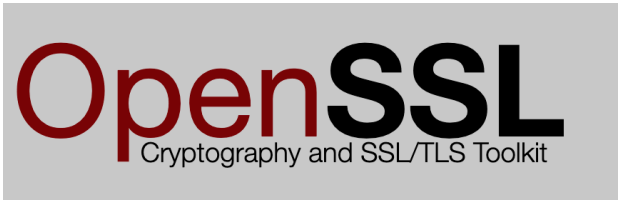
OpenSSL
Cryptography and SSL/TLS Toolkit

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

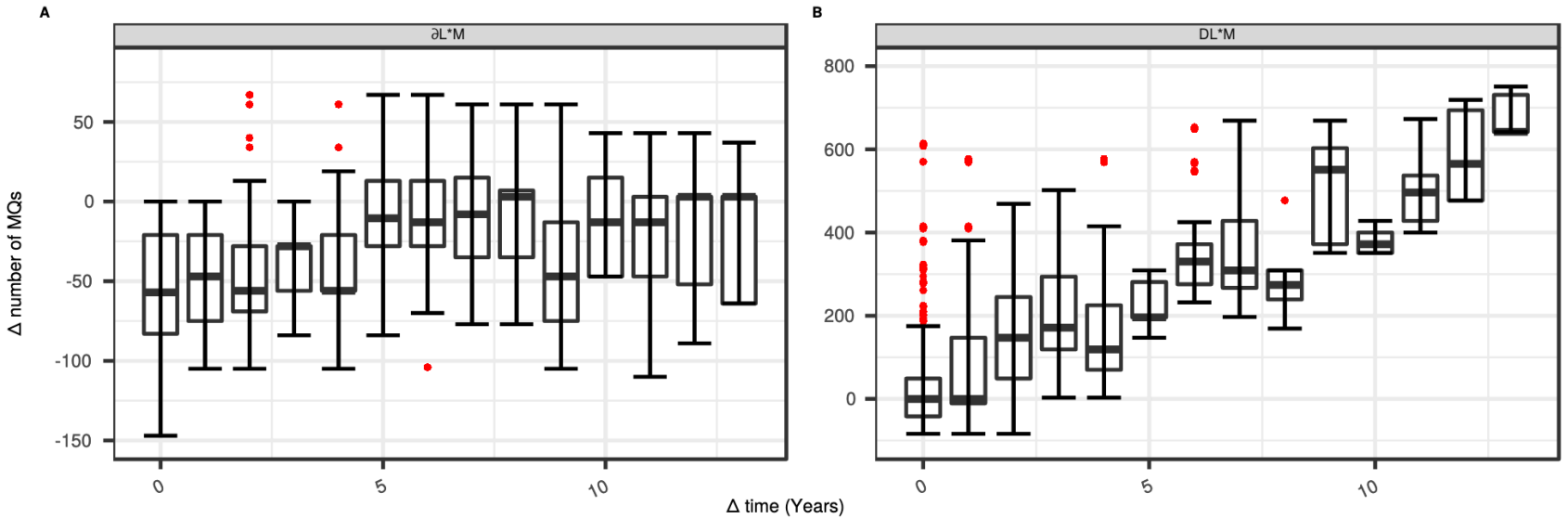
DOES IT REALLY WORK?

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

How sensitive is it to the amount of evolution?



DOES IT REALLY WORK?



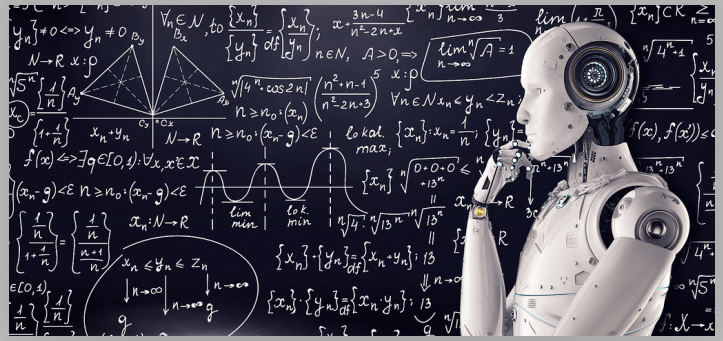
[Damasceno, M.R. Mousavi and A. Simao.
Learning to Reuse: Adaptive Model Learning for Evolving Systems. iFM'19]

UNIVERSITY OF LEICESTER

Variability as a **first-class** citizen!

Modeling variability explicitly

Efficient testing using variability modeling



Learning about variability

UNIVERSITY OF LEICESTER

THANK YOU VERY MUCH!

QUESTIONS?

PLEASE EMAIL: MM789@LE.AC.UK