Combining Model-Based Testing and Machine Learning

Roland GROZ
Université Grenoble Alpes (COMUE),
LIG (Laboratoire d'Informatique de Grenoble)
France

TAROT Summer School 2016

In a nutshell:

- *Model-based testing*…
- when writing a model is *not an option* !

Testing a system is somehow LEARNING the behaviour of a system

<u>Problem:</u> test orderly to learn correct & "complete" behaviour

Outline

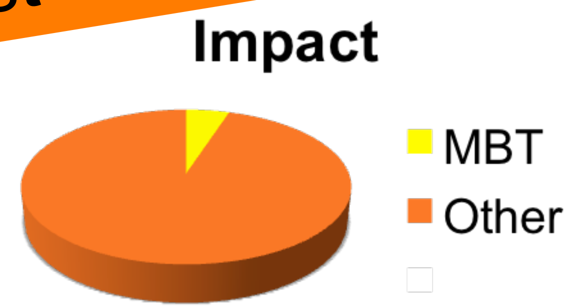
- Motivation: why learning ?
- ML & Soft. Engineering
- Seminal algorithm: L* (Angluin 87)
- Enhancements for various issues
  - Counter-example processing
  - Tree-based (quotient algo)
  - No Reset
  - Integration
  - EFSM
- Related work

Soft. Engineering trends

- MDE & MBT
  - Growing trend in some industries (e.g. embedded)
  - Derive design, code and tests (MBT)
  - Models = 1st class citizens
- Non formal (e.g. Agile)
  - Dominant & growing trend
  - Absence of (formal) models
  - Or pb maintaining spec <-> model
  - Often Test Driven Dvt (TDD)

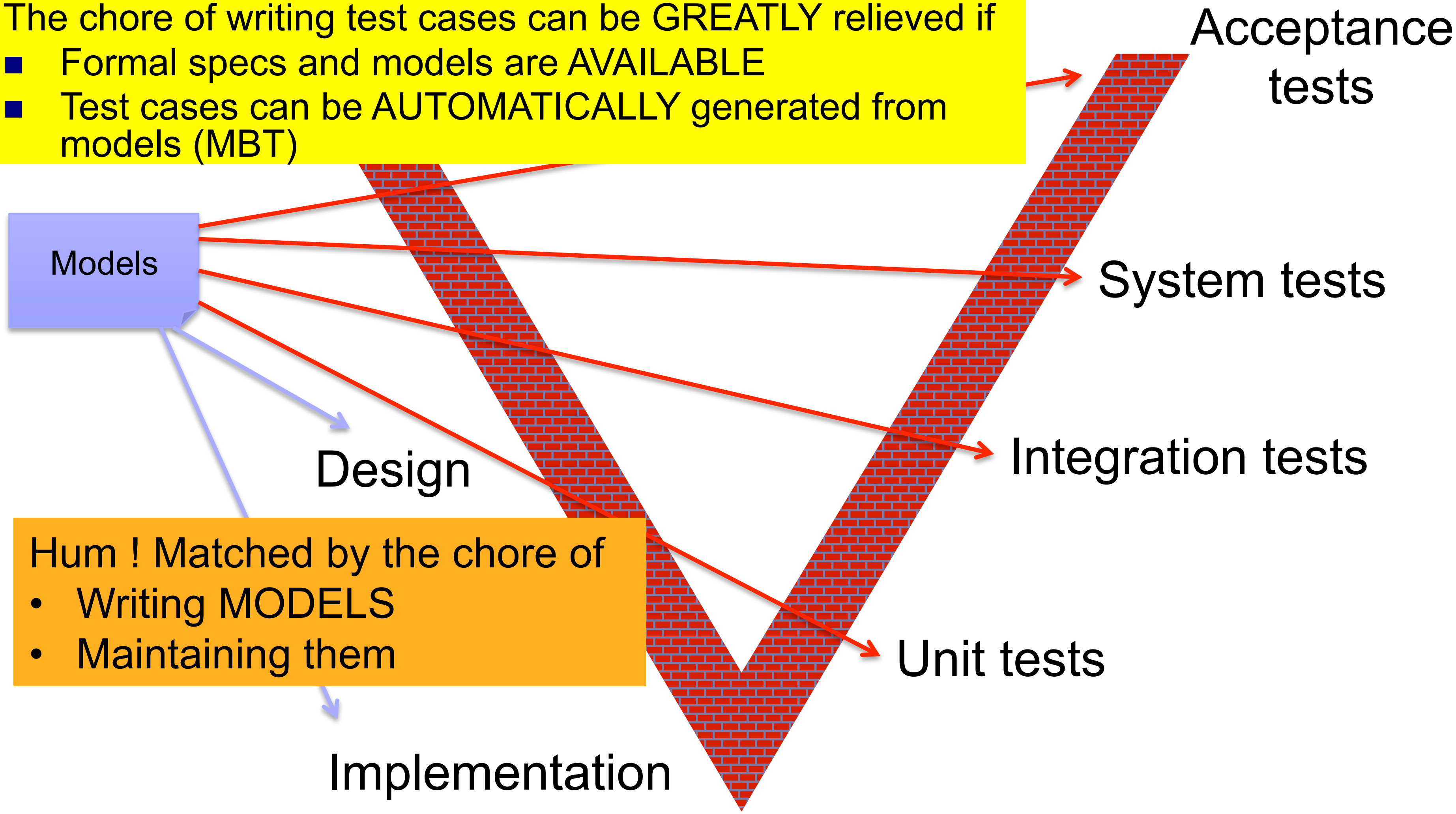
TAROT ☺

Rest of the world ☹



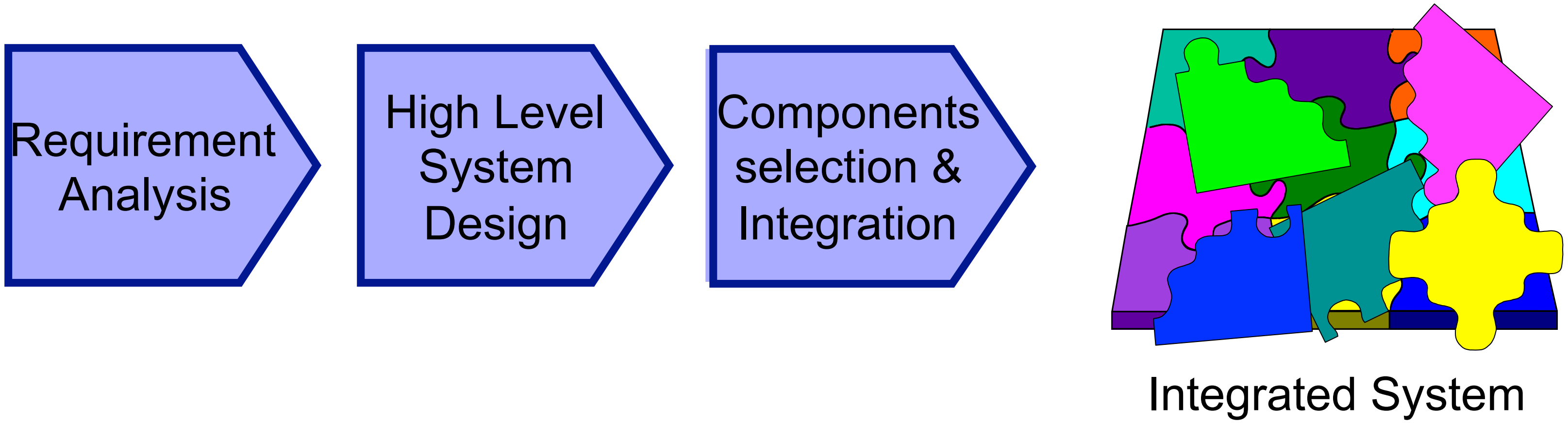
MBT in software development

The chore of writing test cases can be GREATLY relieved if
- Formal specs and models are AVAILABLE
- Test cases can be AUTOMATICALLY generated from models (MBT)



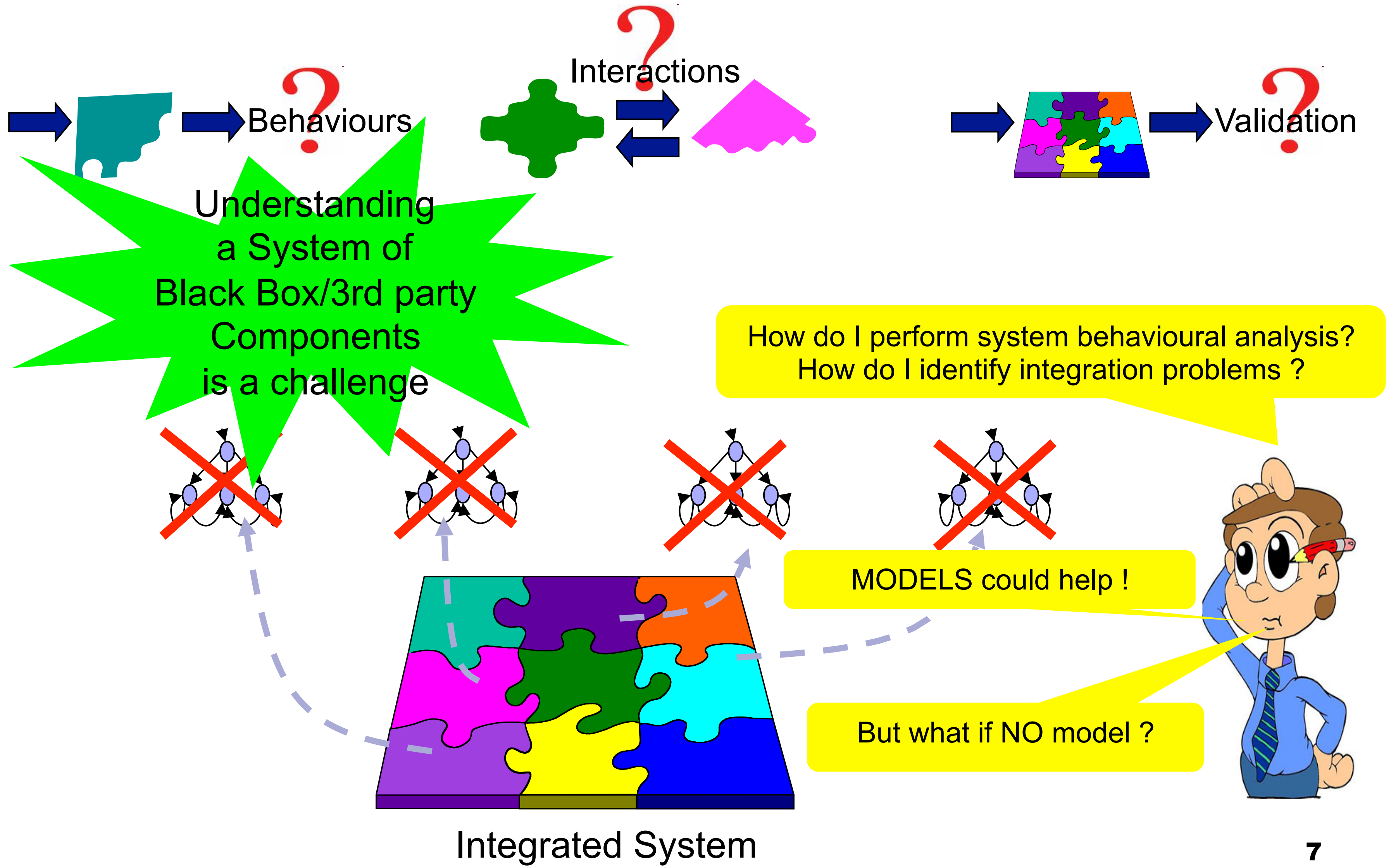
Hum ! Matched by the chore of
- Writing MODELS
- Maintaining them

Component Based Software Engineering



- Rapid Development
- Reuse Components
- Reduce cost
- Flexibility
- Ease of integration

Typical Issues in System Dvt

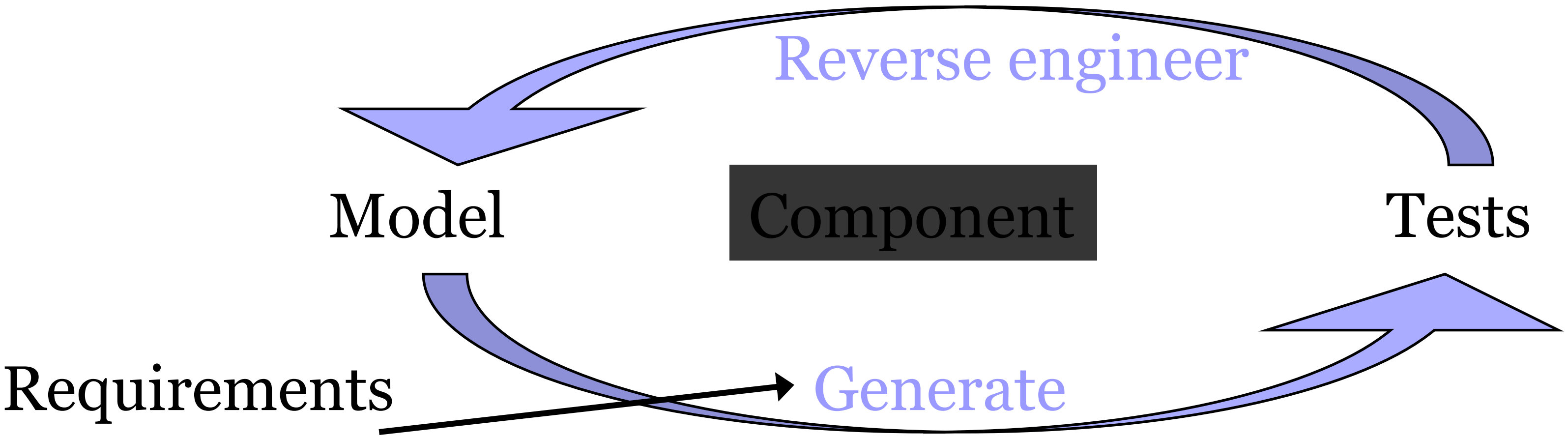


MDE & MBT in the reverse

- MDE assumption
  - Start from model, formal spec
  - Models = 1st class citizens ☺
- Test Driven Development (XP, Agile…)
  - Tests are spec: 1st class citizens
  - Formal models ? No way ! ☹ No time…
- Proposed approach
  - Derive models from tests, & combine with MBT
    - = LEARN models from tests
  - *CHALLENGE: Reconcile*

Test-Driven (or code-driven) dvt  *with Models*

Principle



Partial, incremental and approximate models

Main Technical Goals

- Reverse Engineering
  - Understanding the behaviours of the black box components
    - by deriving the *formal models* of the components/system
    - Can also serve documentation purposes (tests for doc)
- System Validation
  - Being able to derive new systematic tests
  - Analyzing the system for anomalies
    - by model checking (wrt properties)
    - by developing a *framework for integration testing* of the system of black box components

Objections Answers

Objections

- Model is derived from bugged components
  - *Derived tests will consider bug=feature*
- Incremental: stopping criterion ?

Answers

- Unit vs system
  - Combining model-checking & learning
  - Integration testing will reveal errors
- Tunable approximated model of system
- Key notion: counter-examples

Outline

- Motivation: why learning ?
- ML & Soft. Engineering
- Seminal algorithm: L* (Angluin 87)
- Enhancements for various issues
  - Counter-example processing
  - Tree-based (quotient algo)
  - No Reset
  - Integration
  - EFSM
- Related work

Various types of Machine Learning

- Artificial Intelligence (& datamining)
  - Ability to infer rules, recognize patterns
  - Learning from samples
  - E.g. neural networks
- Two major techniques (among others)
  - Statistical inference from collection of data -> e.g. Weka tool in (data) testing

  - *Grammatical inference of language from theoretical computer science*

Pioneering inference in SoftEng

- [Peled 1999] Black Box Checking
  - Using L* + Vasilievski's W-method for Model Checking BB components
- [Steffen, Hagerer 2002] Model generation by Regular extrapolation
  - Applied to testing of telecom switch
- Picked up from 2003 by Dortmund, NASA, Uppsala, Grenoble, Nijmegen, KTH…

Learning languages from samples

"Learning from given positive/negative samples"

- Finding a minimum DFA (Deterministic Finite Automaton) is NP-HARD
  - –*Complexity of automaton identification from given data. [E. Gold 78]*
- Even a DFA with no. of states polynomially larger than the no. of states of the minimum is NP-Complete
  - –*The minimum consistent DFA problem cannot be approximated within any polynomial. [Pitt & Warmuth 93]*
- Probably Approximately Correct (PAC)
  - –*A theory of the learnable. [L.G. Valiant 84]*

Passive Learning

Active learning (Query learning)

- Active Learning
  - "Learning from Queries": inference algorithm can query an oracle of the language
  - **Angluin's Algorithm *L** *[Angluin 87]***
    - Reference algorithm
    - Two types of queries: membership, equivalence
    - Learns Deterministic Finite Automaton (DFA) in <u>polynomial time</u>
  - Applied in formal Software Engineering
    - Black Box Checking [Peled 99]
    - Learning and Testing Telecom Systems [Steffen 02-03]
    - Protocol Testing [Shu & Lee 08]
    - …

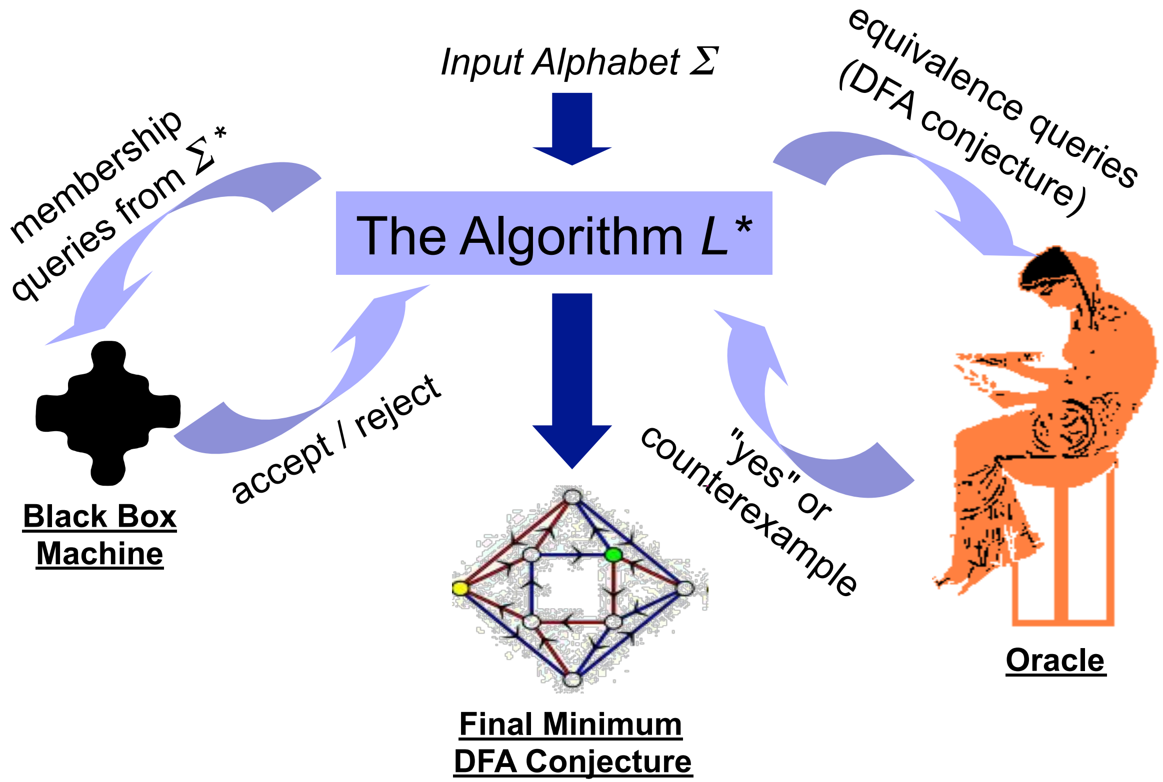
Dana Angluin
Yale University

Active Learning

Outline

- Motivation: why learning ?
- ML & Soft. Engineering
- Seminal algorithm: L* (Angluin 87)
- Enhancements for various issues
  - Counter-example processing
  - Tree-based (quotient algo)
  - No Reset
  - Integration
  - EFSM
- Related work

Concept of the Regular Inference
(Angluin's Algorithm *L**)



Assumptions:
- The input alphabet Σ is known
- Machine can be reset

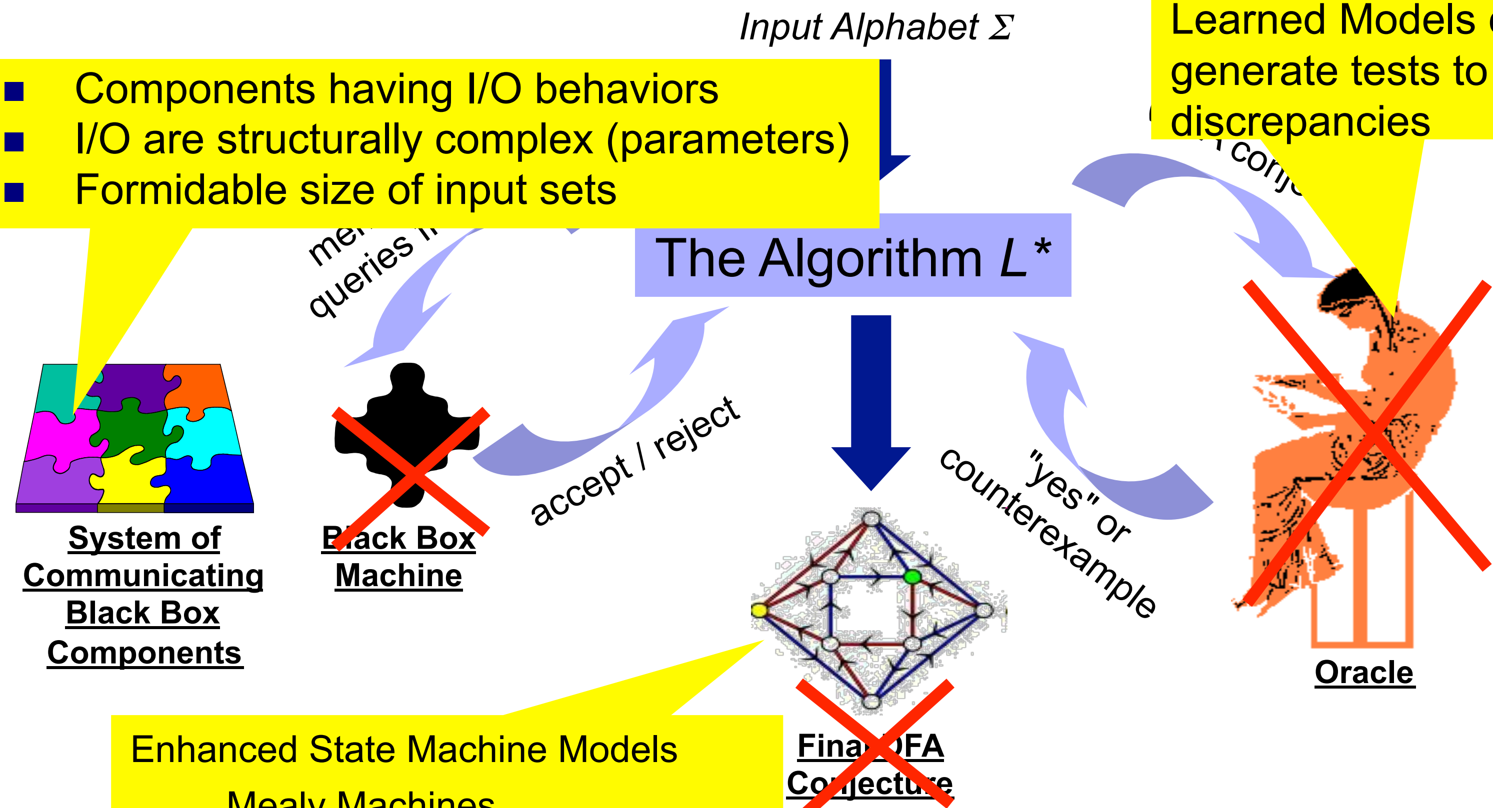
Complexity : $O(|\Sigma|\, m\, n^2)$
- $|\Sigma|$: the size of the input alphabet
- n : the number of states in the actual machine
- m : the length of the longest counterexample

Our Context of Inference (testing s/w)

Test Strategies and heuristics

Learned Models can be used to generate tests to find discrepancies

- Components having I/O behaviors
- I/O are structurally complex (parameters)
- Formidable size of input sets



Enhanced State Machine Models

Mealy Machines

Parameterized Machines

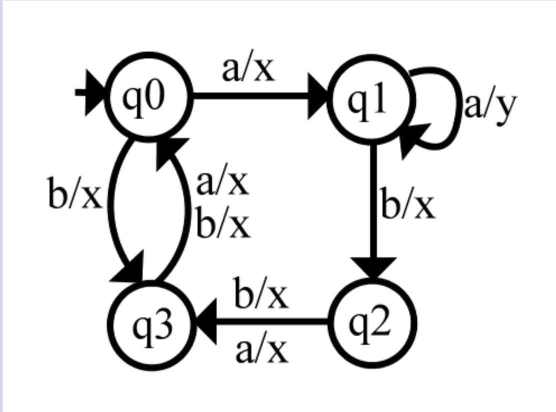
More adequate for complex systems

DFAs may result in transition blow up

Preliminaries

- **Mealy Machine**: $M = (Q, I, O, \delta, \lambda, q_0)$
  - Q : *set of states*
  - I : *set of input symbols*
  - O : *set of output symbols*
  - δ : *transition function*
  - λ : *output function*
  - q_0 : *initial state*

- Input Enabled
  - $dom(\delta) = dom(\lambda) = Q \times I$

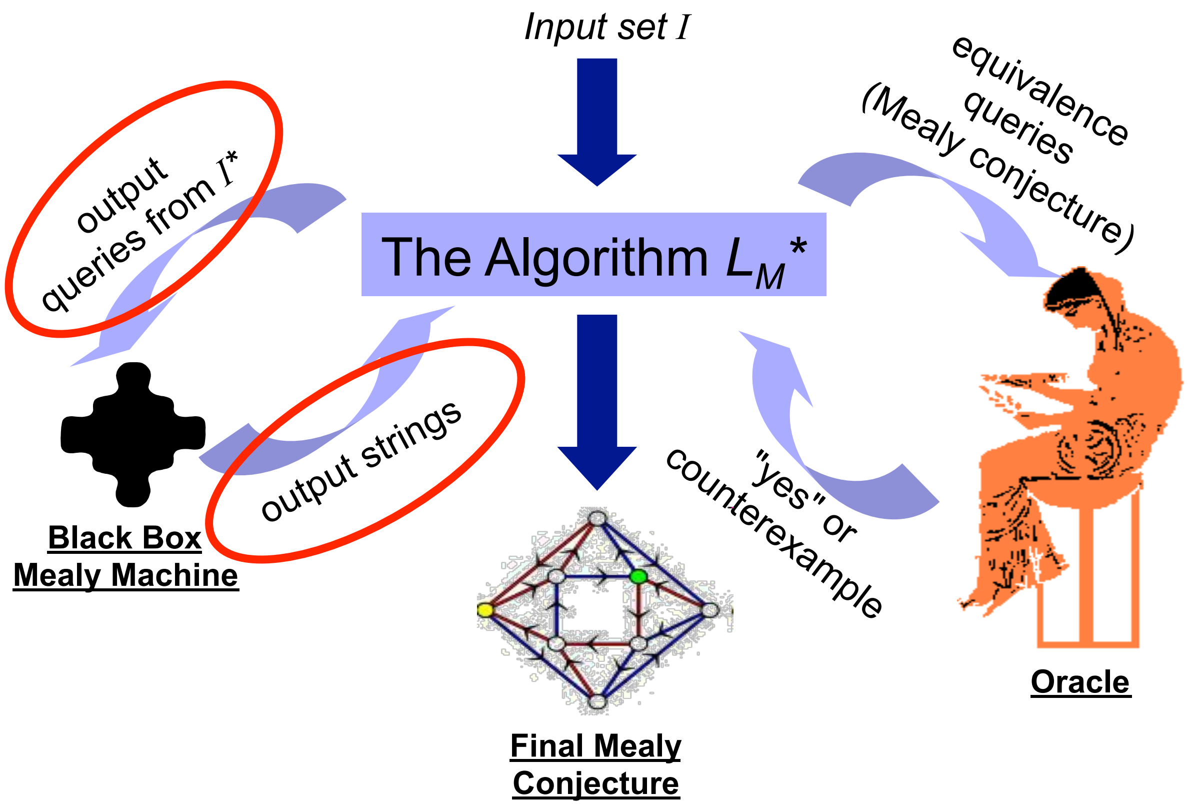


- $Q = \{q_0, q_1, q_2, q_3\}$
- $I = \{a,b\}$
- $O = \{x,y\}$

Running example

Mealy Machine Inference Algorithm

The Algorithm L_M*



Assumptions:

- The input set *I* is known
- Machine can be reset
- For each input, the corresponding output is observable

Basic principles of L_M* algorithm

Discriminating sequences

	a	b
ε	x	x
a	y	x
b	x	x
aa	y	x
ab	x	x

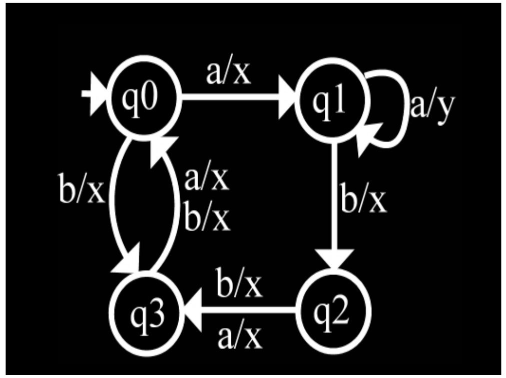
S (span seq for)
States

S · I
lookahead
tail state id

Build queries
row.col
submit row.col ->

record output <-
for col

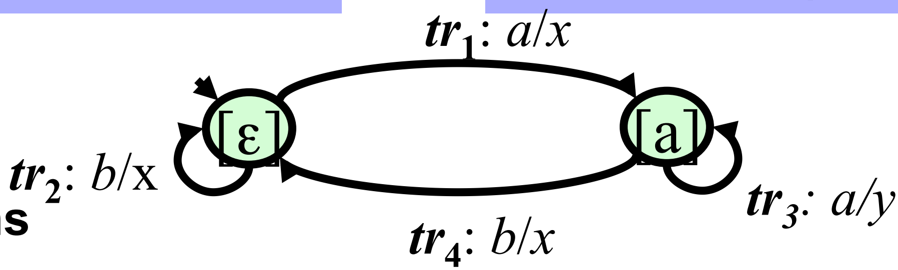
I={a, b}



Observation Table

Black Box Mealy Machine Component

Conjecture:
minimal FSM
consistent with observations



•ε is an empty string

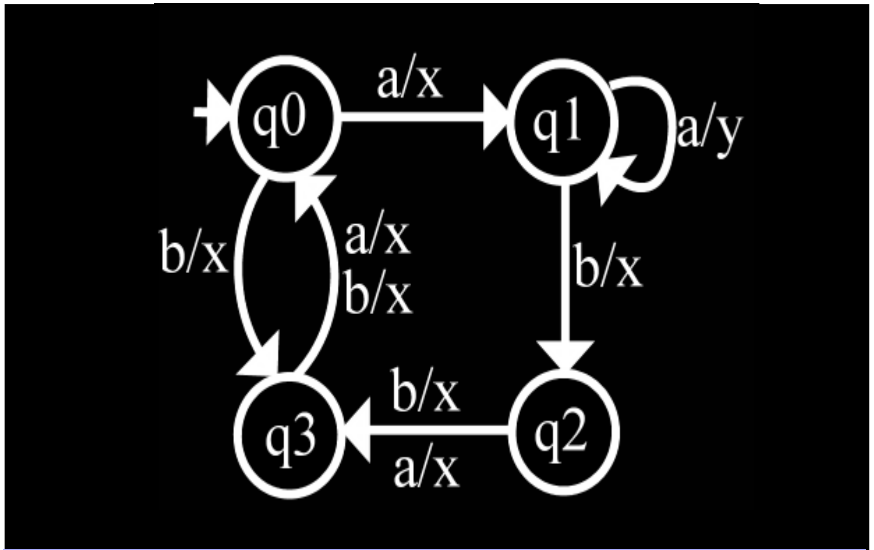
Mealy Machine Inference Algorithm L_M* (1/6)

Initialization

		E_M: a	E_M: b
S_M	ε	x	x
$S_M \cdot I$	a	y	x
	b	x	x

Observation Table (S_M, E_M, T_M)

I={a, b}



Black Box Mealy Machine Component

Initialization

- $S_M = \varepsilon$
- $E_M = I$

Output Queries:

$s \cdot e$, $s \in (S_M \cup S_M \cdot I)$, $e \in E_M$

- = a / x

•ε is an empty string

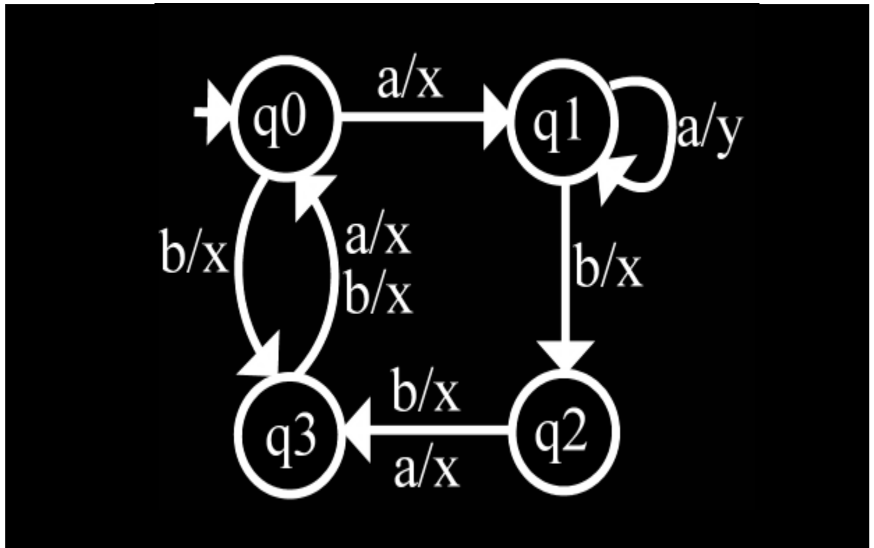
Mealy Machine Inference Algorithm L_M* (2/6)

Concept: Closed

		E_M	
		a	b
S_M	ε	x	x
$S_M \cdot I$	a	y	x
	b	x	x
	aa	y	x
	ab	x	x

Observation Table (S_M,E_M,T_M)

I={a, b}



Black Box Mealy Machine Component

<u>Concepts</u>:

- Closed : All the rows in S_M • I must be equivalent to the rows in S_M
  - *Same behaviour = known state*
- Consistency

•ε is an empty string

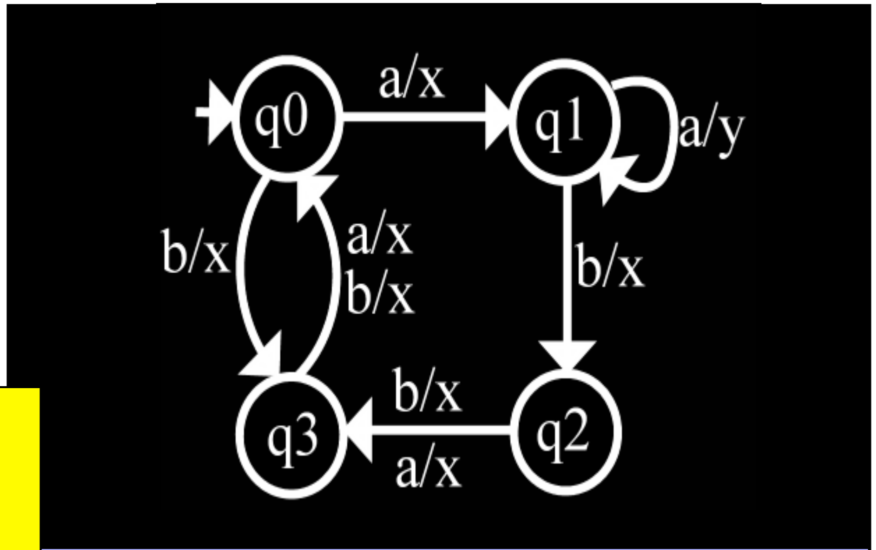
Mealy Machine Inference Algorithm L_M^* (3/6)

Making Conjecture

tr_1 tr_3 tr_2 tr_4

		E_M: a	E_M: b
S_M	ε	x	x
	a	y	x
$S_M \cdot I$	b	x	x
	aa	y	x

I={a, b}



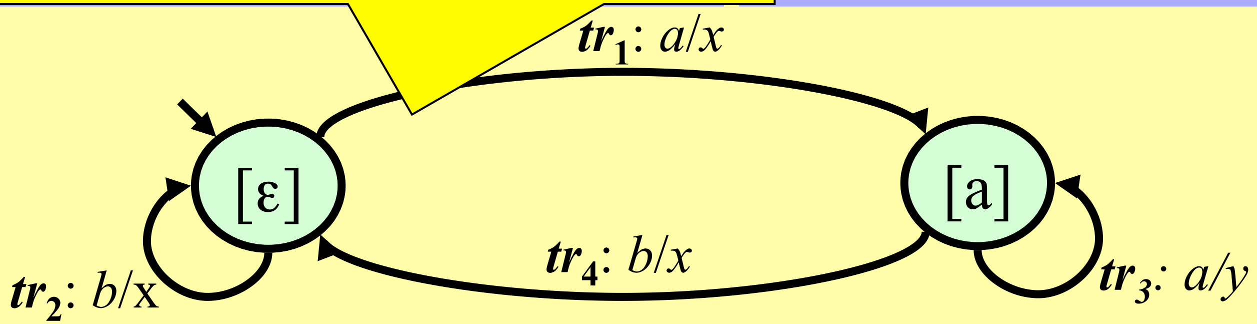
Black Box Mealy Machine Component

<u>*Counterexample:*</u>

a b a b b a a

component's response: x x x x x x y

conjecture's response: x x x x x x x



Mealy Machine Inference Algorithm L_M^* (4/6)

Processing Counterexamples

		E_M: a	E_M: b
S_M	ε	x	x
S_M	a	y	x
$S_M \cdot I$	b	x	x
$S_M \cdot I$	aa	y	x
$S_M \cdot I$	ab	x	x

Observation Table (S_M,E_M,T_M)

Counterexample: **a b a b b a a**

<u>Method:</u>

Add all the prefixes of the counterexample to S_M

→

	a	b
ε	x	x
a	y	x
ab	x	x
aba	x	x
abab	x	x
ababb	x	x
ababba	x	x
ababbaa	y	x
aa	y	x
b	x	x
abb	x	x
abaa	x	x
ababa	y	x
ababbb	x	x

Mealy Machine Inference Algorithm L_M^* (5/6)

Concept: Consistency

	a	b	
ε	x	x	xy
a	y	x	yy
ab	x	x	xx
aba	x	x	xx
abab	x	x	xy
ababb	x	x	xx
ababba	x	x	xy
ababbaa	y	x	yy
aa	y	x	yy
b	x	x	xx
abb	x	x	xx
abaa	x	x	xy
ababa	y	x	yy
ababbb	x	x	xy

Observation Table (S_M,E_M,T_M)

Concepts:

- Closed
- Consistency : All the successor rows of the equivalent rows must also be equivalent

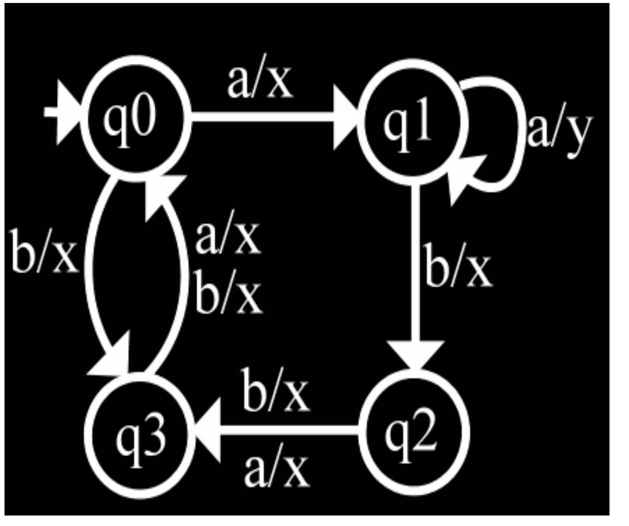
- First inconsistency
  - **ε** and **ab** look similar… but not **ε.a** and **ab.a**
- Later inconsistency:
  - **ab** and **aba**, but not **aba** and **abaa**
- …

Mealy Machine Inference Algorithm L_M* (6/6)

Termination: Conjecture = Black Box

	a	b	aa	aaa	baa
ε	x	x	xy	xxx	xxxy
a	y	x	yy	xxx	xxxx
ab	x	x	xx	xxy	xxxx
aba	x	x	xx	xxx	xxxy
abab	y	x	yy	xxx	xxxx
ababb	x	x	xy	xxx	xxxy
ababba	x	x	xy	xxx	xxxy
ababbaa	x	x	xx	xxy	xxxx
b	x	x	xx	xxy	xxxx
aa	x	x	xx	xxy	xxxx
abb	x	x	xx	xxx	xxxy
abaa	y	x	yy	xxx	xxxx
ababa	x	x	xy	xxx	xxxy
ababbb	x	x	xy	xxx	xxxy
ababbab	x	x	xx	xxy	xxxx
ababbaaa	x	x	xx	xxy	xxxx
ababbaab	x	x	xx	xxx	xxxx

Final Observation Table (S_M,E_M,T_M) after processing counterexample according to L_M*



<u>Complexity :</u> **O(|Σ| m n²)**

- |Σ| : the size of the input alphabet
- n : the number of states in the actual machine
- m : the length of the longest counterexample

Outline

- Motivation: why learning ?
- ML & Soft. Engineering
- Seminal algorithm: L* (Angluin 87)
- Enhancements for various issues
  - Counter-example processing
  - Tree-based (quotient algo)
  - No Reset
  - Integration
  - EFSM
- Related work

Other algorithms derived from L*

- Counter-example processing
  - Rivest & Schapire (1993)
    - Do not add prefixes (avoid compatibility check)
    - Dichotomic search for discriminating suffix
      - Complexity falls to $O(|\Sigma|n^2 + n \log m)$
      - But flawed (Balcazar 97)
    - Corrected by Shahbaz, Irfan and Groz (2009):
      - Suffix1by1
- Only membership queries
  - Howar: Zulu competition at ICGI 2010

Processing Counterexamples avoiding consistency checks

Counterexample

a b a b b a a

Add all the suffixes to E_M

	a	b
ε	x	x
a	y	x
b	x	x
aa	y	x
ab	x	x

Observation Table (S_M,E_M,T_M) before processing counterexample

All rows remain inequivalent (inconsistency never occurs)

	a	b	aa	baa	bbaa	abbaa
ε	x	x	xy	xxx	xxxy	xxxxx
a	y	x	yy	xxx	xxxx	yxxxx
b	x	x	xx	xxy	xxxx	xxxxy
aa	y	x	xx	xxx	xxxy	xxxxx
ab	x	x	yy	xxx	xxxx	yxxxx

Observation Table (S_M,E_M,T_M) after processing counterexample

Comparison of the two Methods

Total Output Queries in L_M+ : **64**

		b	aa	baa	bbaa	abbaa
ε	x	x	xy	xxx	xxxy	xxxxx
a	y	x	yy	xxx	xxxx	yxxxx
b	x	x	xx	xxy	xxxx	xxxxy
ab	x	x	xx	xxx	xxxy	xxxxx
aa	y	x	yy	xxx	xxxx	yxxxx
ba	x	x	xy	xxx	xxxy	xxxxx
bb	x	x	xy	xxx	xxxy	xxxxx
aba	x	x	xx	xxy	xxxx	xxxxy
abb	x	x	xx	xxy	xxxx	xxxxy

Final Observation Table (S_M,E_M,T_M) after processing counterexample according to L_M^+

Total Output Queries in L_M* : **86**

a	y	x	yy	xxx	xxxx
ab	x	x	xx	xxy	xxxx
ab	x	x	xx	xxx	xxxy
aba	y	x	yy	xxx	xxxx
abab	x	x	xy	xxx	xxxy
ababba	x	x	xy	xxx	xxxy
ababbaa	x	x	xx	xxy	xxxx
b	x	x	xx	xxy	xxxx
aa	x	x	xx	xxy	xxxx
abb	x	x	xx	xxx	xxxy
abaa	y	x	yy	xxx	xxxx
ababa	x	x	xy	xxx	xxxy
ababbb	x	x	xy	xxx	xxxy
ababbab	x	x	xx	xxy	xxxx
ababbaaa	x	x	xx	xxy	xxxx
ababbaab	x	x	xx	xxx	xxxx

Final Observation Table (S_M,E_M,T_M) after processing counterexample according to L_M^*

Comparison of the two Methods

Complexity of L_M*:

$O(|I|^2 n m + |I| m n^2)$

Complexity of L_M+:

$O(|I|^2 n + |I| m n^2)$

- I : the size of the input set
- n : the number of states in the actual machine
- m : the length of the longest counterexample

Outline

- Motivation: why learning ?
- ML & Soft. Engineering
- Seminal algorithm: L* (Angluin 87)
- Enhancements for various issues
  - Counter-example processing
  - Tree-based (quotient algo)
  - No Reset
  - Integration
  - EFSM
- Related work

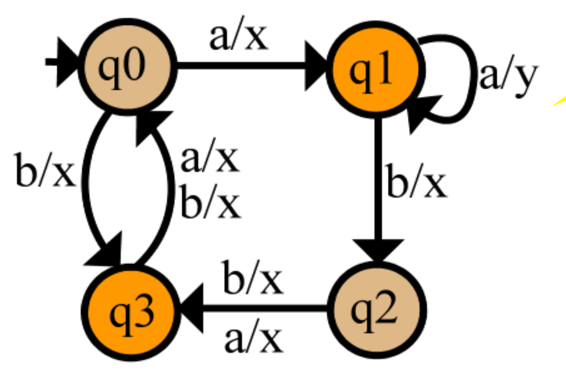
Other active learning algorithms

- Other data structures: trees vs tables
  - Kearns & Vazirani (1994): binary tree
    - $O(|\Sigma|n^3 + nm)$
  - Z-quotient: tree & quotient automata
    - Petrenko, Li, Groz (HASE 2014)
  - TTT
    - Isberner, Howar, Steffen (RV 2014)

Mealy Machine Quotients

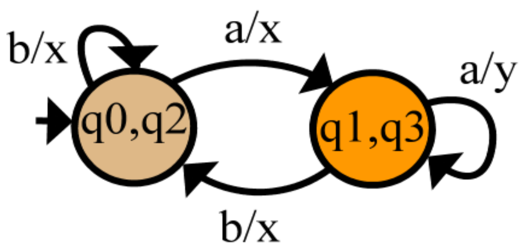
- Let Φ be a set of strings from I then
  - the states *s1* and *s2* are *Φ-equivalent* if they produce same outputs for all the strings in Φ
  - A quotient based upon Φ-equivalence is called *Φ-quotient*

Φ = {a, b, ab, ba, bb, bba}



Mealy Machine *M*

q_0 and q_2 are Φ-Equivalent
q_1 and q_3 are Φ-Equivalent



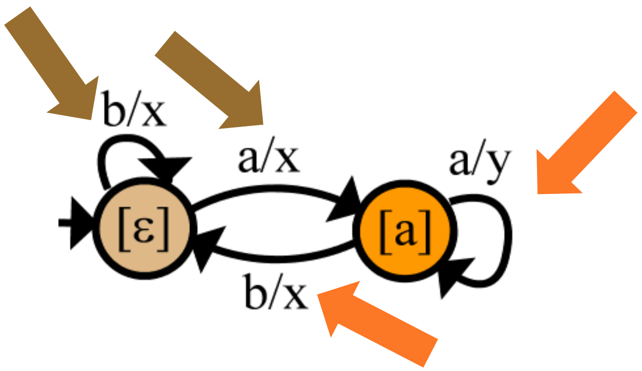
Φ-quotient of *M*

Relation between the Conjecture and the Black Box Machine

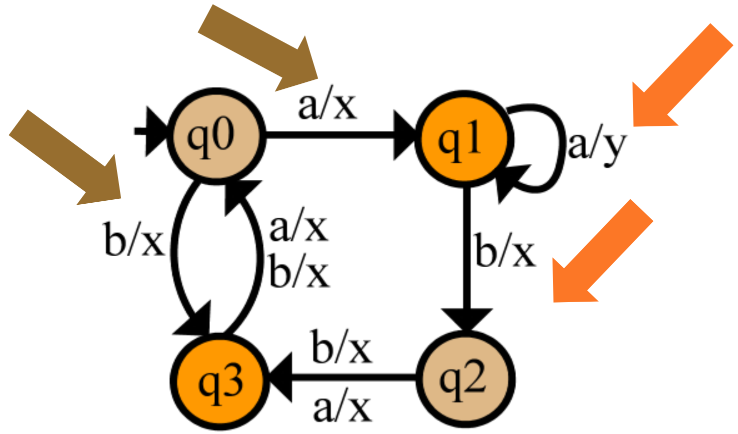
E_M-Quotient

		E_M: a	E_M: b
S_M	ε	x	x
S_M	a	y	x
$S_M \cdot I$	b	x	x
$S_M \cdot I$	aa	y	x
$S_M \cdot I$	ab	x	x

Closed (and Consistent) Observation Table (S_M,E_M,T_M)

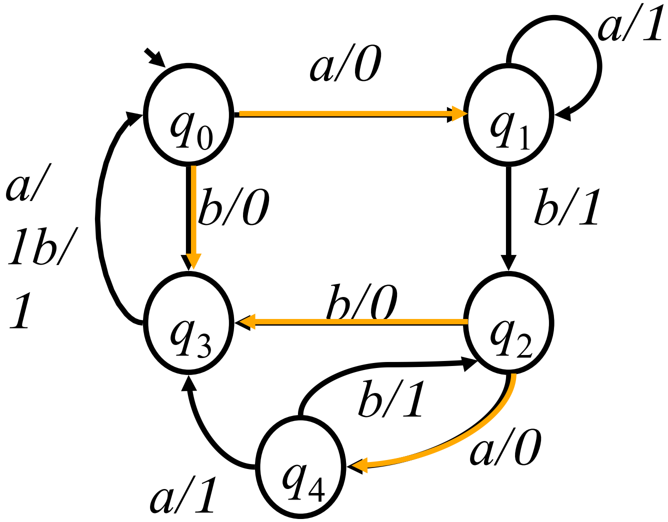


Conjecture from the Observation Table (S_M,E_M,T_M)



Black Box Mealy Machine

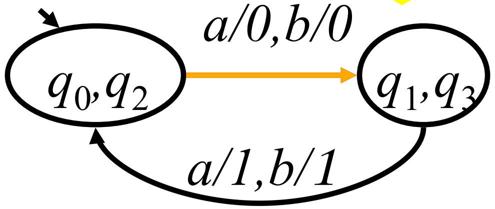
Initial k-Quotient



Machine ***M***

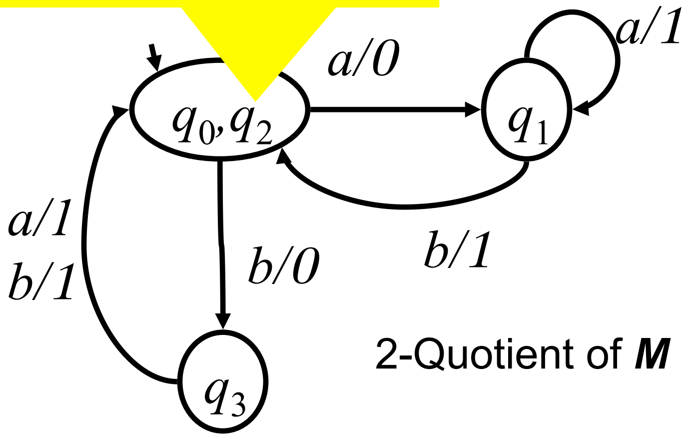
3-Quotient(***M***) ≡ ***M***

q_0 and q_2 are 1-Equivalent: a/0,b/0
q_1 and q_3 are 1-Equivalent: a/1,b/1



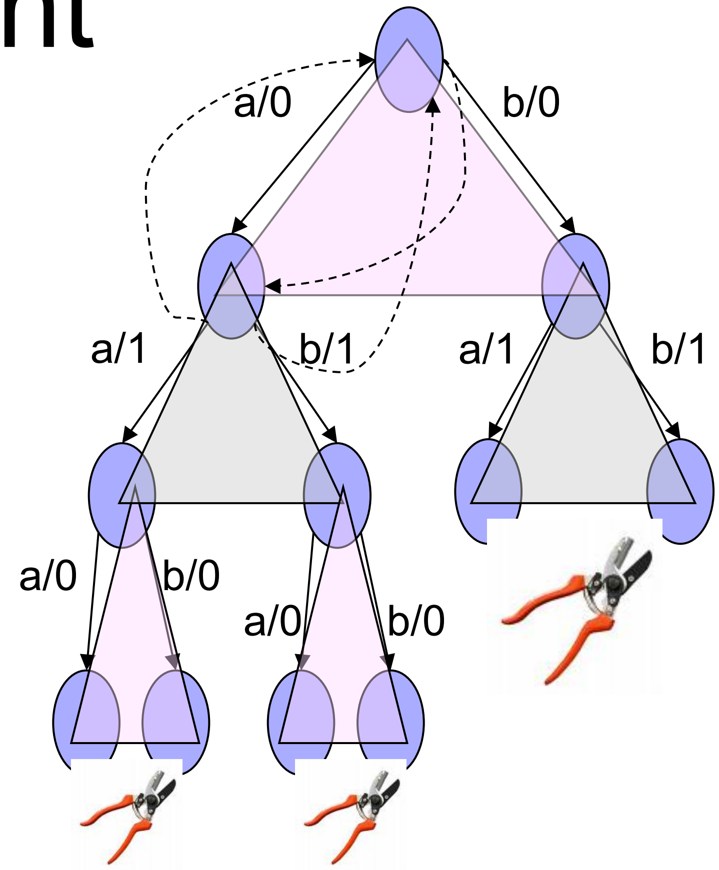
1-Quotient of ***M***

q_0 and q_2 are still 2-Equivalent
q_1 and q_3 are 2-Disting. a/1 a/?

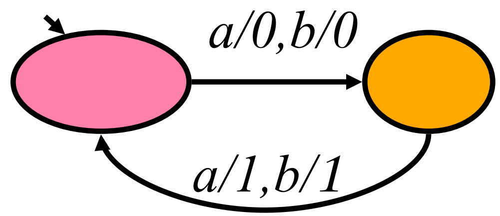


2-Quotient of ***M***

Inferring a k-quotient
(example with k=1)



- BFS exploration of traces of increasing length
- Pruning under node k-equiv to another one
- Final step: merging node when trace included, and redirecting transitions



Groz,Li,Petrenko,Shahbaz TestCom 2008

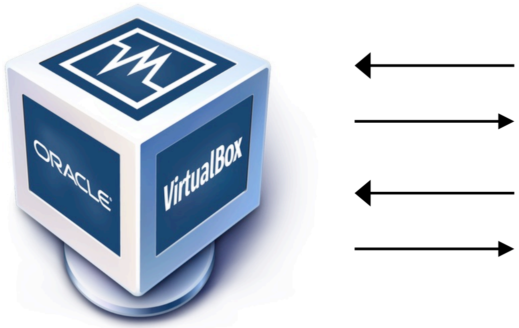
*Extended to arbitrary Σ-quotients Σ ⊆ I**

Outline

- Motivation: why learning ?
- ML & Soft. Engineering
- Seminal algorithm: L* (Angluin 87)
- Enhancements for various issues
  - Counter-example processing
  - Tree-based (quotient algo)
  - No Reset
  - Integration
  - EFSM
- Related work

Motivational example

- Reverse-engineer models of Web applications to detect <u>security vulnerabilities</u>
- E-Health app provided by Siemens as a Virtual Machine



Learner

- single I/O RTT over LAN: < 1 ms
- reset=reboot VM: ~1 minute

- Timewise: <u>reset is $O(10^5)$</u> RTT in example
- Many systems CANNOT be reset AT ALL.

Key difficulties when no reset

- How can we know in which state seq is applied ?
- No backtrack possible to check other sequence
- Losing track: we no longer know from where we apply an input
  - ➔ localizer procedure

Can we infer a Black-Box machine without reset?

Problem, assumptions, *result*

Stronger assumptions

Groz, Simao et al 2015

- Known bound N on nb of states: $n \leq N$
- Known W-set for BB
  - Card(W) = p

Algo: polynomial in N

<<O(f N^{p+2}) bound

but mean O(f $N^{1.9}$) for p=2

Rivest & Schapire 1993

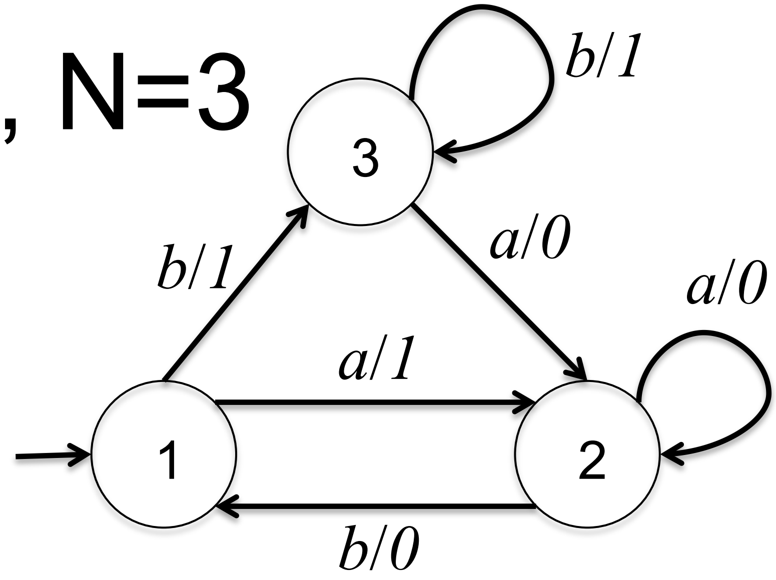
- Oracle knows BB, can answer yes or no
- Oracle can provide CE
  - |Largest CE| = m
- Known Homing Sequence for BB

Algo: polynomial in n

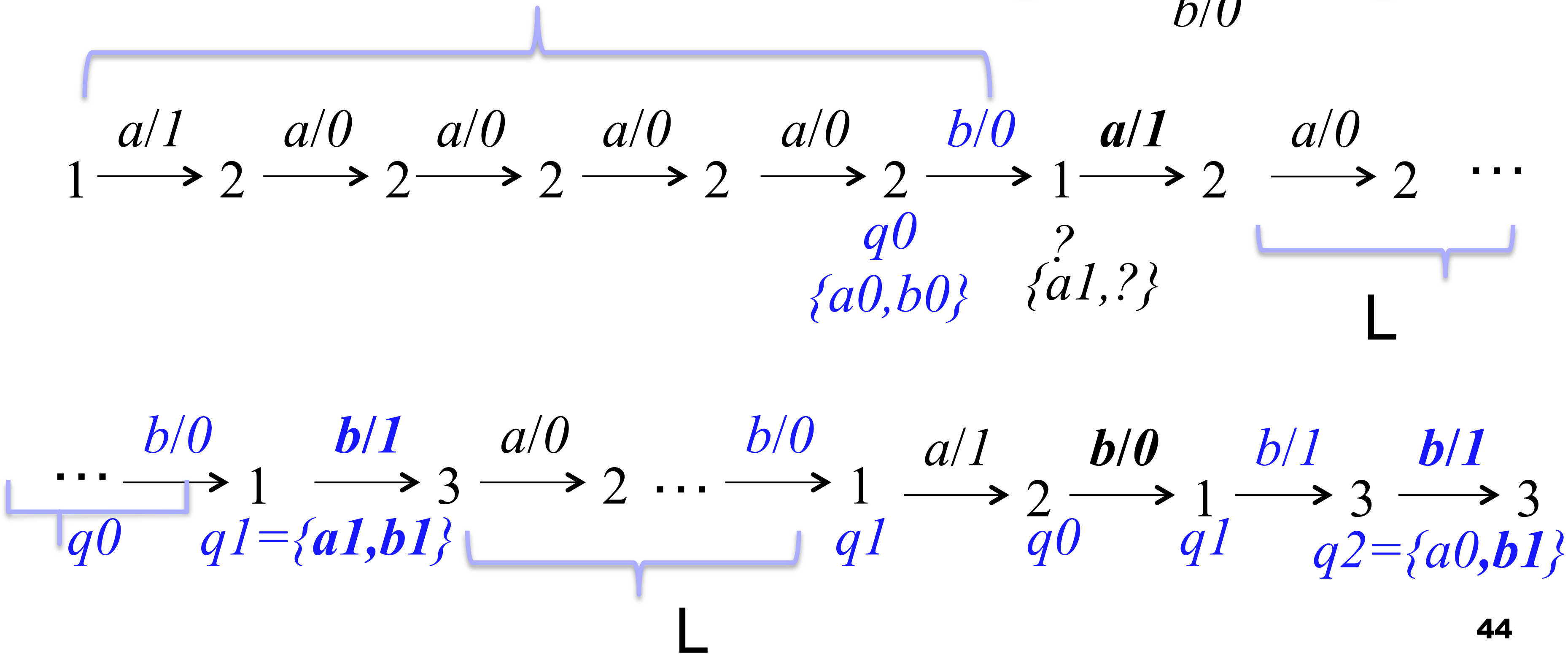
~O(f m n^3)

Lower practical complexity for p<=2

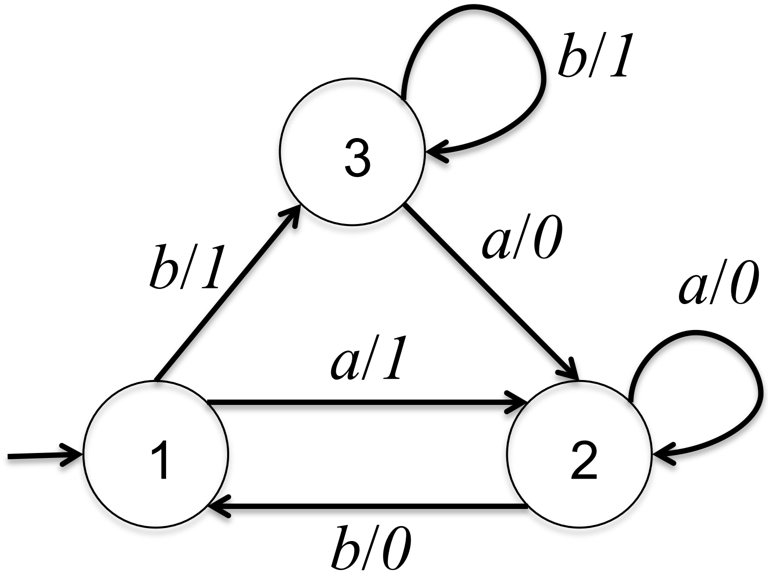
Example: W = {a, b}, N=3



Localizer seq. L = a^5b



Example (end)



$$\overbrace{3 \xrightarrow{a/0} 2 \;\dots\; \underset{q0}{2} \xrightarrow{b/0}}^{L} \underset{q1}{1} \xrightarrow{a/1} 2 \xrightarrow{a/0} 2 \xrightarrow{\mathbf{b/0}} 1 \xrightarrow{b/1} 3$$

$$3 \xrightarrow{a/0} 2 \xrightarrow{b/0} 1 \xrightarrow{b/1} 3 \xrightarrow{b/1} 3$$

It pays off to learn without reset !

Learner

- single I/O RTT over LAN: <1ms
- reset=reboot VM: ~1minute

finite state

Cost of single reset ~sequence of 10^5 inputs

- If we know W of 2 elements, it is FASTER to learn WITHOUT reset !
- If we know W of 3 elements, it may still pay off depending on number and length of queries

Outline

- Motivation: why learning ?
- ML & Soft. Engineering
- Seminal algorithm: L* (Angluin 87)
- Enhancements for various issues
  - Counter-example processing
  - No Reset
  - Integration
  - EFSM
- Related work

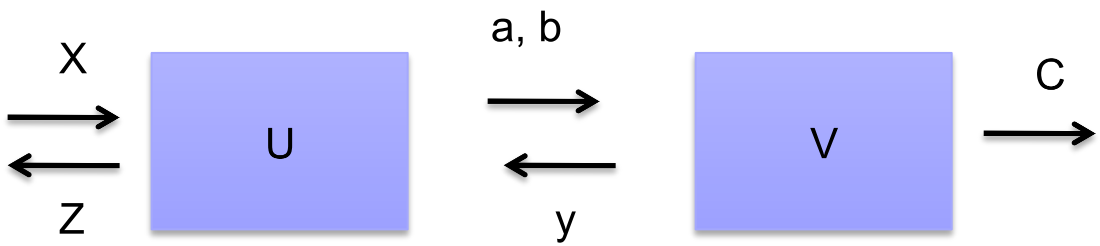
Integration testing

- Popular issues
  - Architecture, testability
  - Integration order, stubbing
  - Interoperability testing
- *Combining integration with Model learning*
  - *Unit learning (1st approach)*
  - *Deriving integration tests from combined learned models*

No formal models ☹

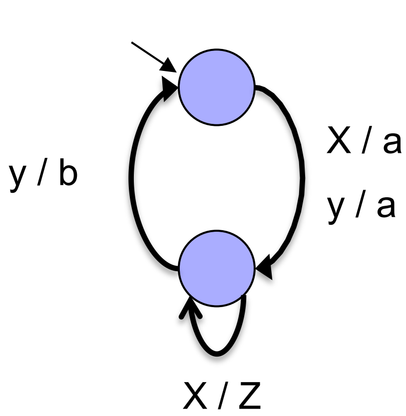
☺

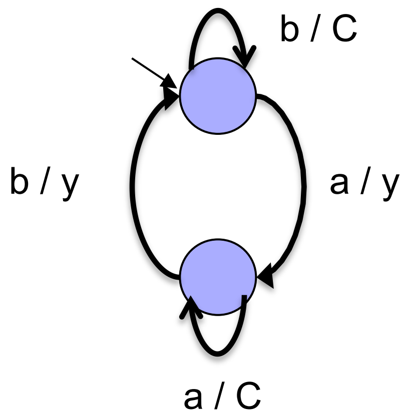
Integration exposes models



Component U: I_U={X,y}

Component V: I_V={a,b}



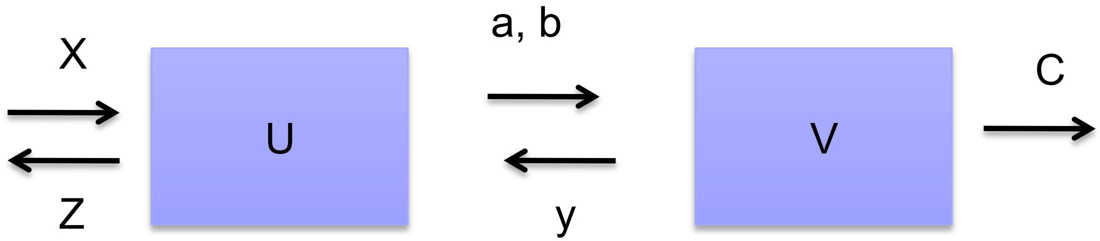


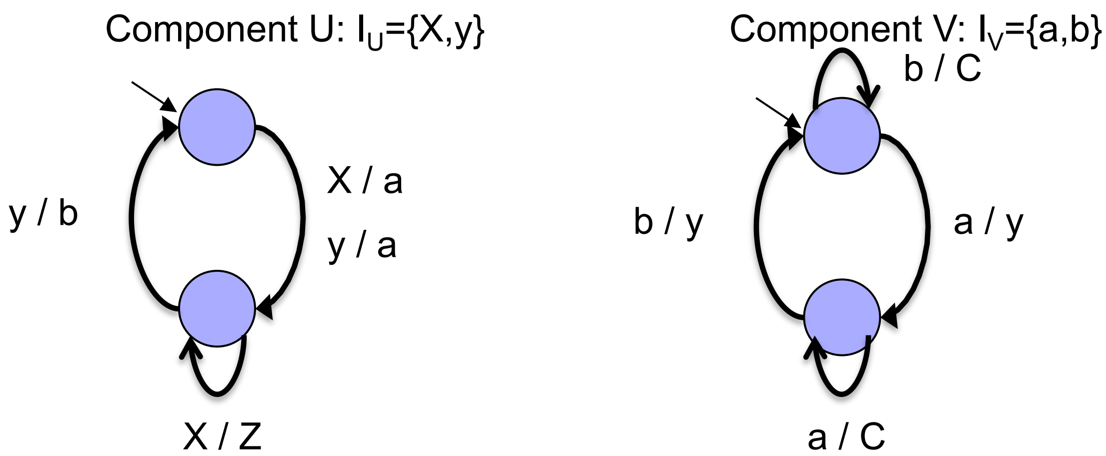
Composed Model: X a y b y a y b y a y … Livelock !

Analysing the problem

- Artefact ?
  - Possibly: models are approximate
- Check sequence on real system
  1. If Livelock confirmed: report error
  2. If Real sequence differ: counter example

Integration provides counter-examples



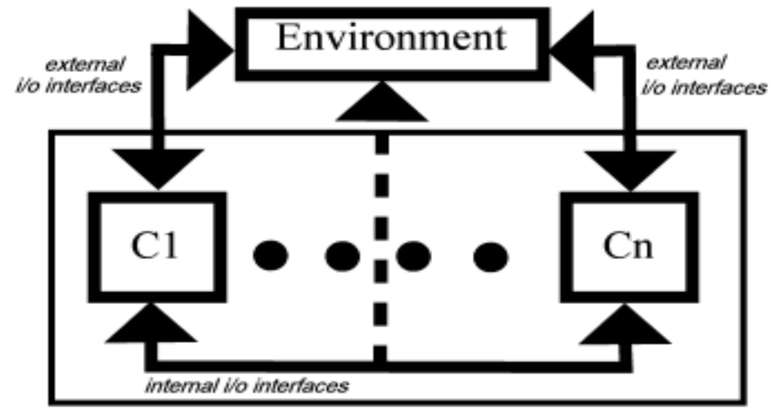


Composed Model: X a y b y a y b y a y … Livelock !

Real : X a y b y a y b y b C

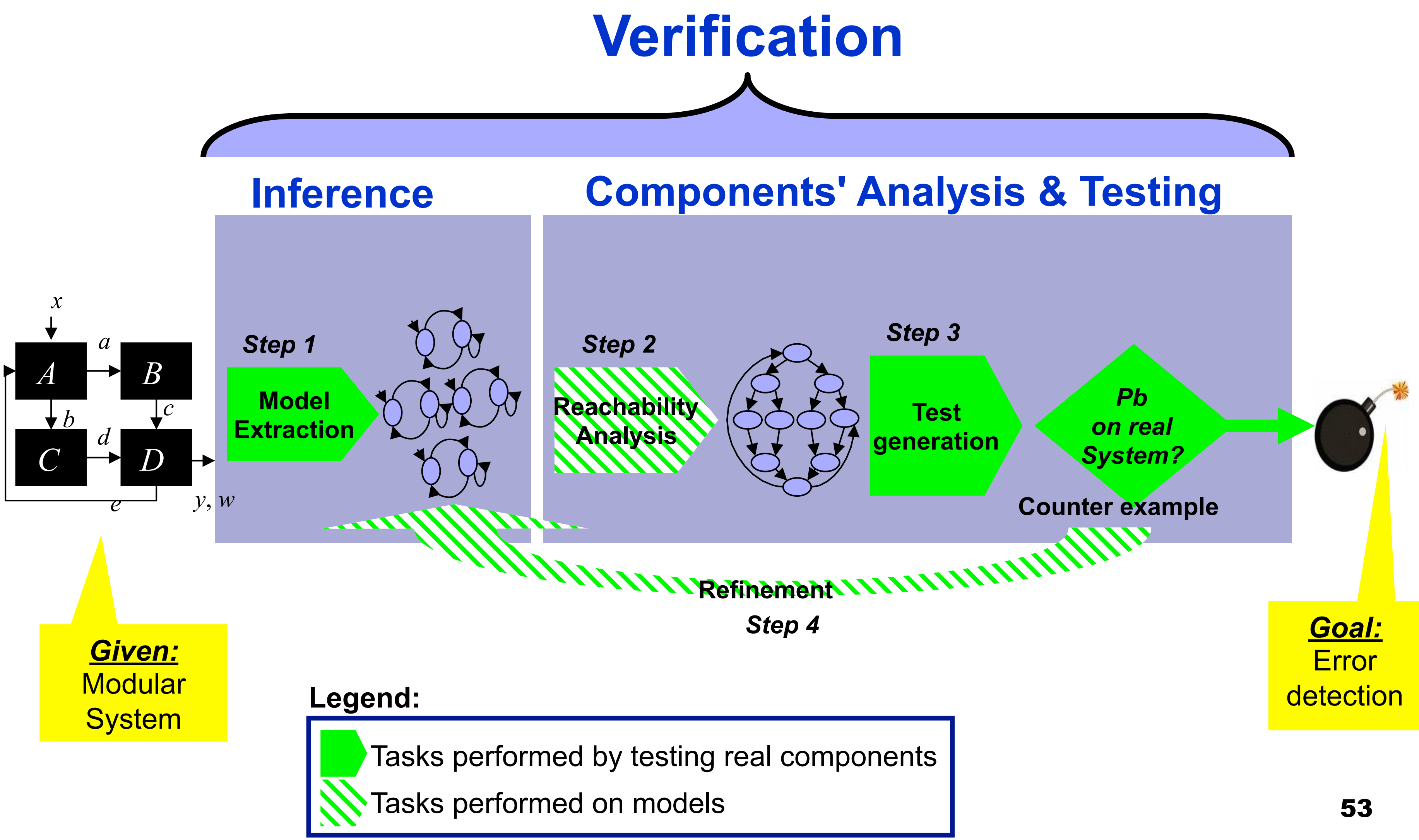
-> Refine U model with (projected) counter-example

System architecture & assumptions

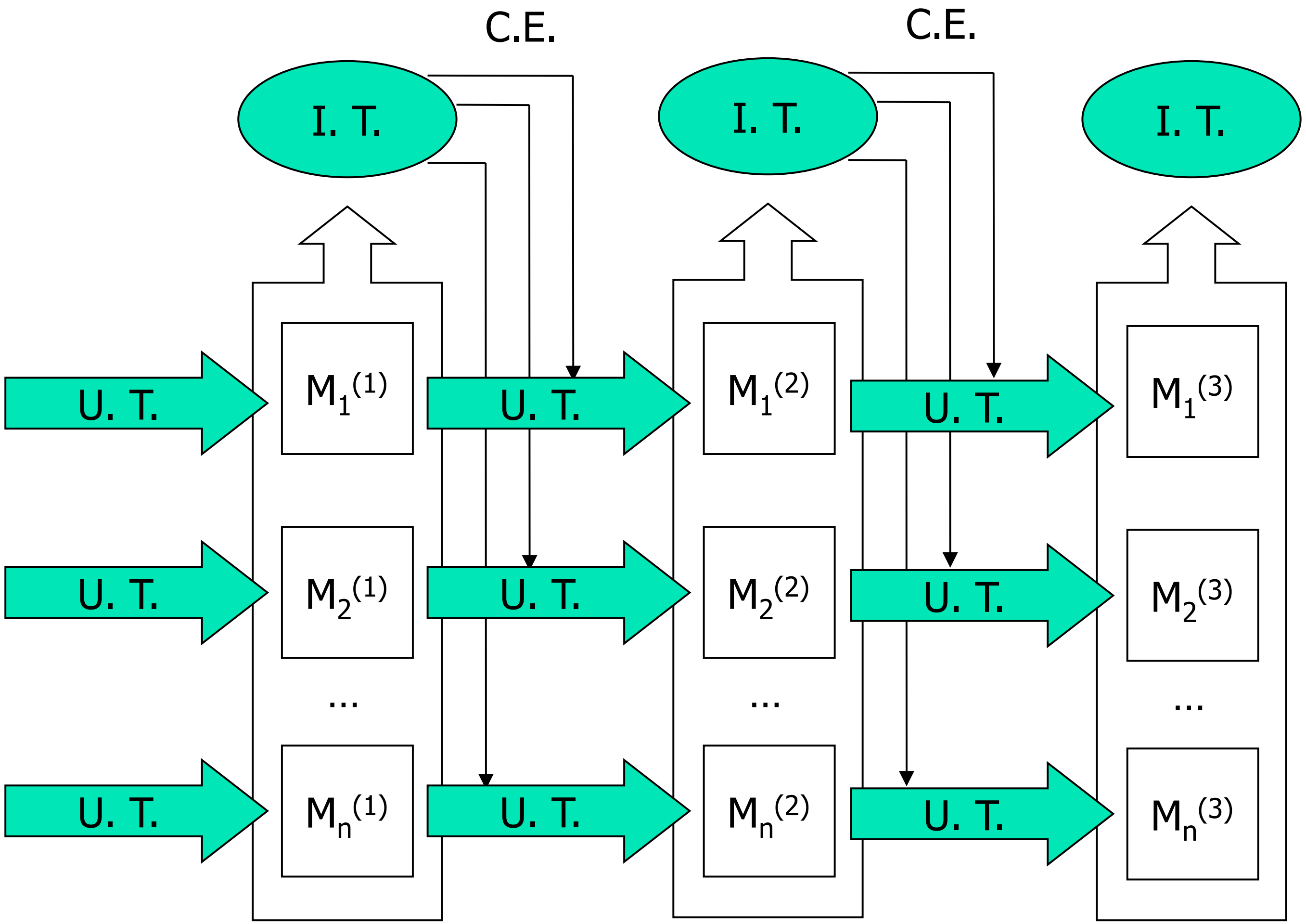


- System of communicating Mealy Machine Components
- Components are deterministic and input-enabled
- System has *External* and *Internal* i/o interfaces
  - External interface is controllable
  - External and Internal interfaces are observable
- Single Message in Transit and Slow Environment

Overview (simplified)



Iterations

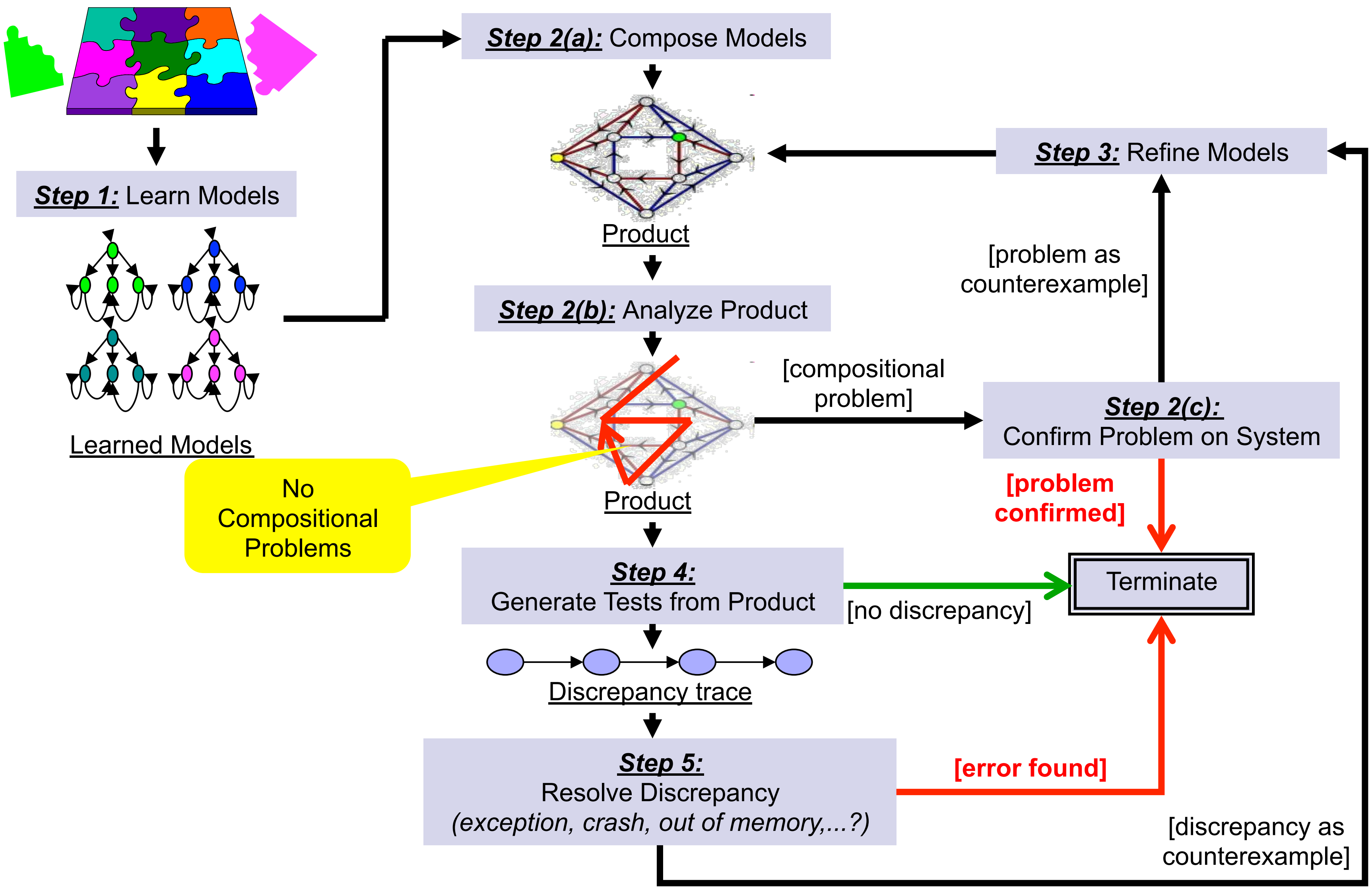


U. T.:
Unit testing

I. T.:
Integration testing

C.E.:
Counter-example

Learning & Testing Framework



Outline

- Motivation: why learning ?
- ML & Soft. Engineering
- Seminal algorithm: L* (Angluin 87)
- Enhancements for various issues
  - Counter-example processing
  - No Reset
  - Integration
  - EFSM
- Related work

Learning extended FSM

- Dealing with boolean variables
  - Th. Berg, B. Jonsson & H. Raffelt FASE 2006
- Parameterized inputs/outputs
  - no var, arbitrary I/O functions: Shahbaz 2007
  - Var. with equality: Berg, Jonsson… 2008
- With variables
  - Register automata: Howar et al VMCAI 2012
  - With Data Mining inference of guards and output functions: Li, Hossen, Groz

Combining state & data inference

- Connecting to Daikon tool, for dynamic invariant detection
  - Shahbaz ISOLA 2007

Daikon: inductive inference of functions from samples
y=f(x) M. Ernst (U. Washington)

- Weka & FSM inference
  - Dury & Petrenko: security of Web interface
  - Li & Groz: EFSM inference

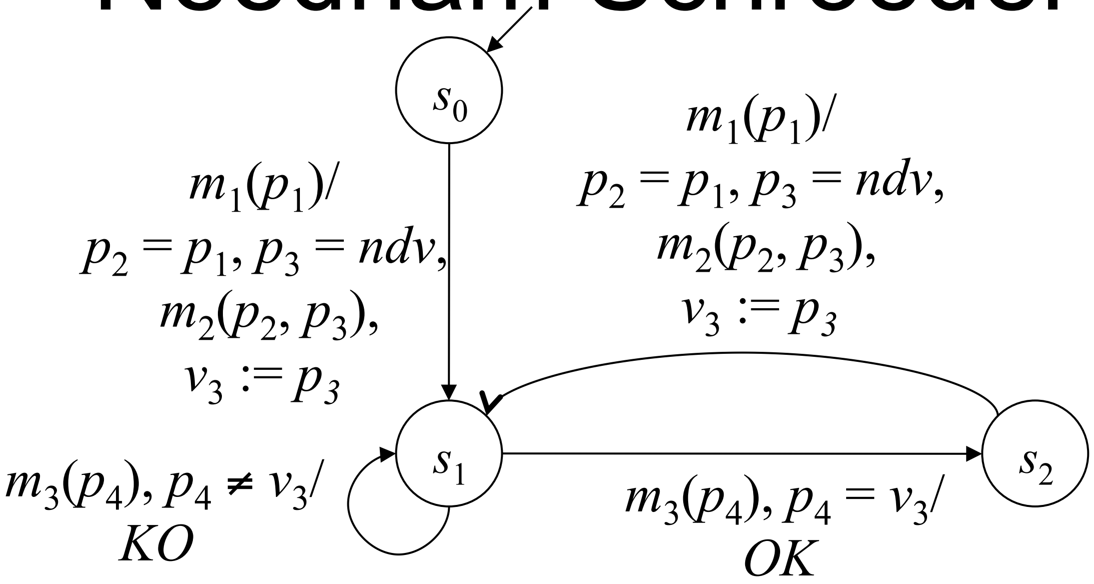
Weka: data mining toolset, clustering (U. Waikato)

Inferring for security

- Input parameters critical (e.g. Cross site scripting...)
- Storing past values: cookies, session IDs
- Non-deterministic values: nonces

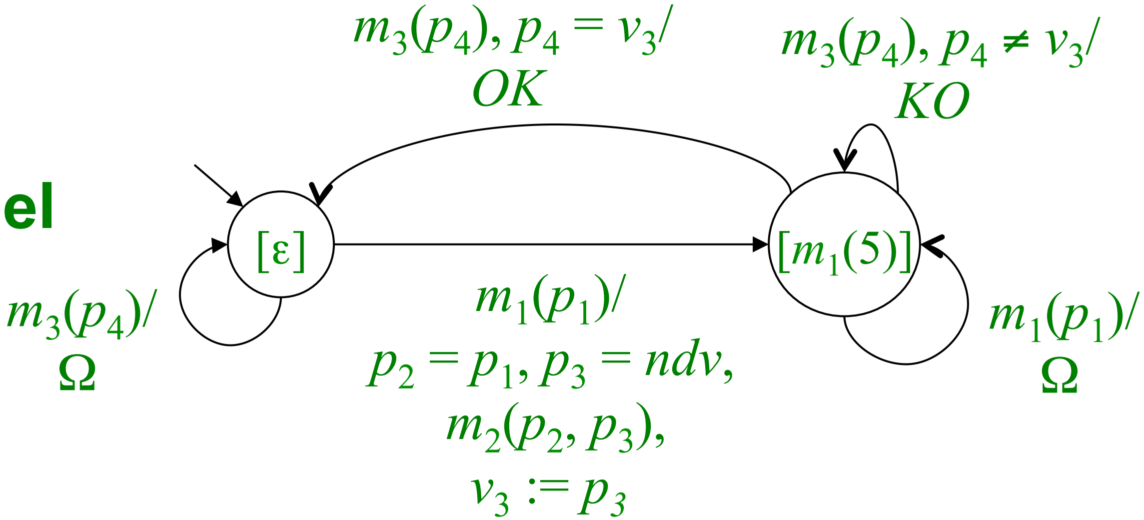
- Model: Extended FSM with ND values, and storage

Needham Schroeder authentication



Extended FSM model of NSPK Responder

Inferred EFSM model



State inference // data inference

	m_1	m_1
ε	(5, m_2) (ndv_3, m_2)	(10, Ω) (ndv_3, Ω)
$m_1(5)$	(5, Ω), (ndv_3, Ω)	(10, KO) (ndv_3, OK)
$m_3(10)$	(5, m_2) (ndv_3, m_2)	(10, Ω) (ndv_3, Ω)
$m_1(5)$ $m_1(5)$	(5, Ω), (ndv_3, Ω)	(10, KO) (ndv_3, OK)
$m_1(5)m_3(10)$	(5, Ω), (ndv_3, Ω)	(10, KO) (ndv_3, OK)

	m_1	m_1
ε	(5, (0, 0, 0, 0) → (5, *ndv*)), (0, (0, 0, 0, 0) → (0, *ndv*))	(10, (0, 0, 0, 0) → ω), (0, (0, 0, 0, 0) → ω)
$m_1(5)$		
$m_3(10)$	(5, (0, 0, 0, 10) → (5, 600)), (0, (0, 0, 0, 10) → (0, 800))	(10, (0, 0, 0, 10) → ω), (0, (0, 0, 0, 10) → ω)
$m_1(5)$ $m_1(5)$	(5, (5, 5, 900, 0) → ω), (0, (5, 5, 110, 0) → ω)	(10, (5, 5, 120, 0) → ω), (130, (5, 5, 130, 0) → ω)
$m_1(5)m_3(10)$	(5, (5, 5, 140, 10) → ω), (150, (5, 5, 150, 10) → ω),	(10, (5, 5, 150, 10) → ω), (160, (5, 5, 160, 10) → ω)

Outline

- Motivation: why learning ?
- ML & Soft. Engineering
- Seminal algorithm: L* (Angluin 87)
- Enhancements for various issues
  - Counter-example processing
  - No Reset
  - Integration
  - EFSM
- Related work

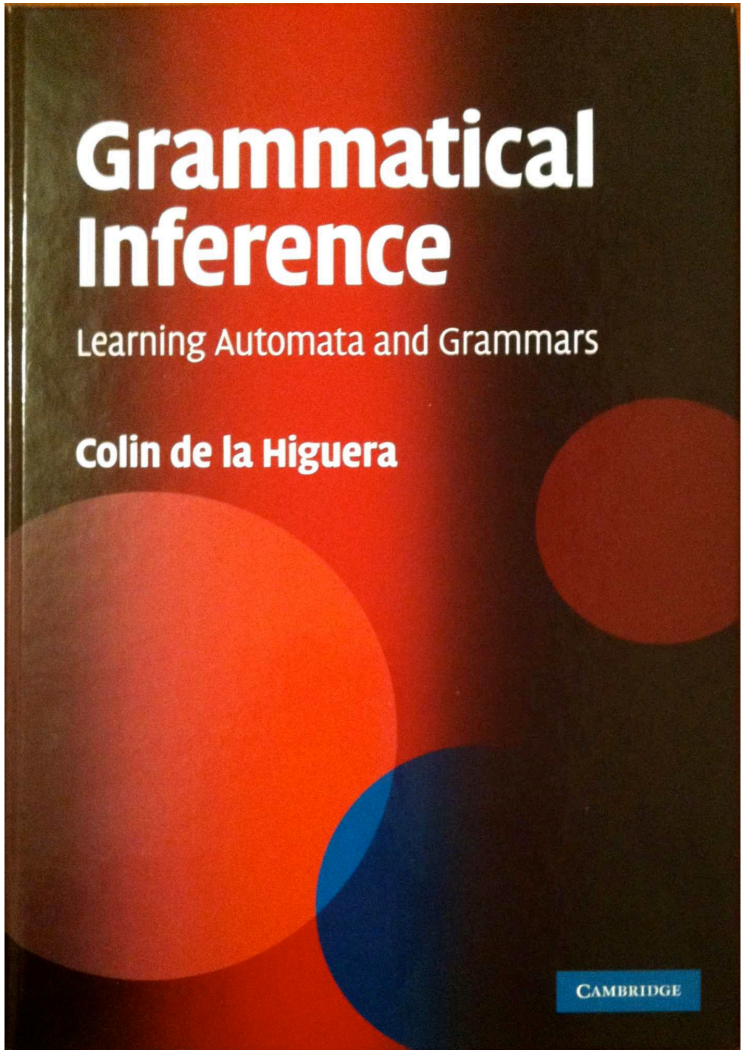
Related work

- Active learning in Soft. Eng/Testing
  - D. Peled (Bar-Ilan): Black Box Checking (1999)
  - C. Pašareanu (NASA): Assume-Guarantee Proof(2008)
  - B. Steffen, H. Raffelt (Dortmund): Dynamic Testing via Automata Learning (2003-2007)
  - D. Lee & G. Shu (Ohio 2007): Security protocol testing
  - B. Jonsson, T. Berg (Uppsala): Register automata
  - K. Meinke (KTH): Learning Based Testing (& model checking), Congruence on Abstract Data Types
  - F. Vaandrager, S. Verwer (Nijmegen): Smartcard

Related work

- Many other approaches
  - Specification mining, becoming popular
    - May assume code available, often passive
    - Typical papers:
      - Ammons (POPL 2002) coined the word
      - Lorenzoli, Mariani, Pezze (ICSE 2008)
      - Bertolino, Inverardi (FSE 2009)
- Use of (statistical) Machine Learning in testing
  - E.g. for test data classification & partition refinement (Briand 2008)

Reference book on learning automata



- For machine learning in general:
  - Many references,
  - e.g. A. Cornéjuols & L. Miclet

- No book as yet for Software Testing & machine learning
  - Planned April 2017 (Springer): outcome of Dagstuhl seminar 2016