

Heuristics for Improving Model Learning Based Testing

Muhammad Naeem Irfan
VASCO-LIG, Computer Science Lab,
Grenoble Universities, 38402 Saint Martin d'Hères
France



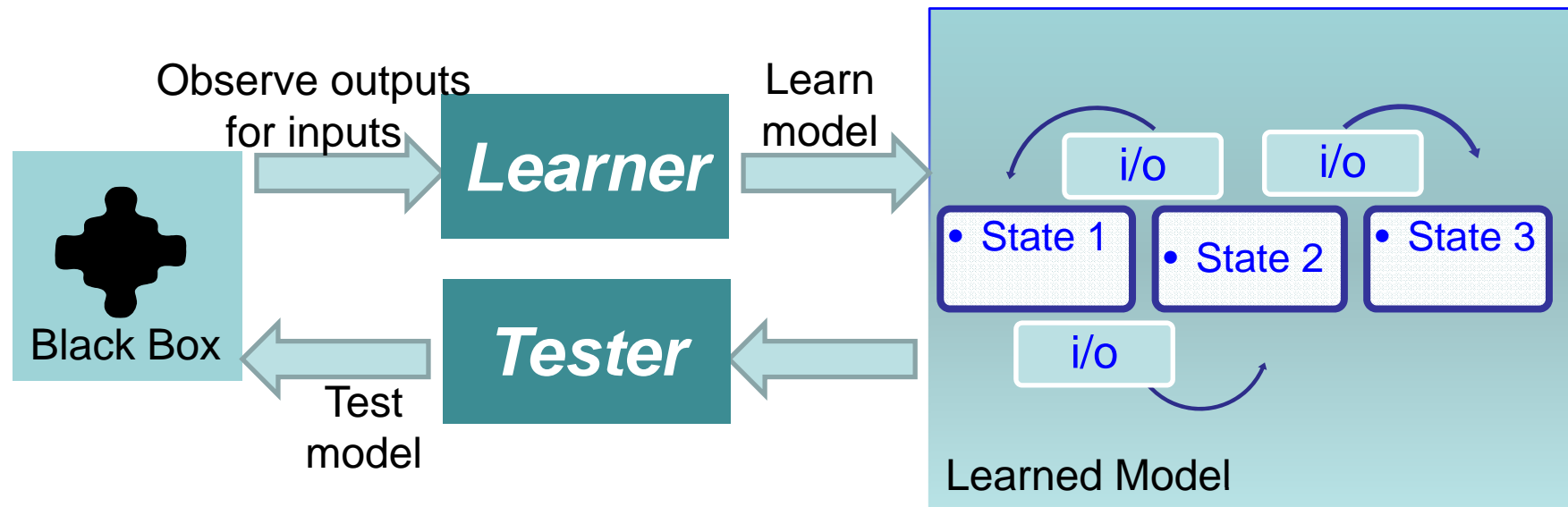
Introduction

Component Based Software Engineering

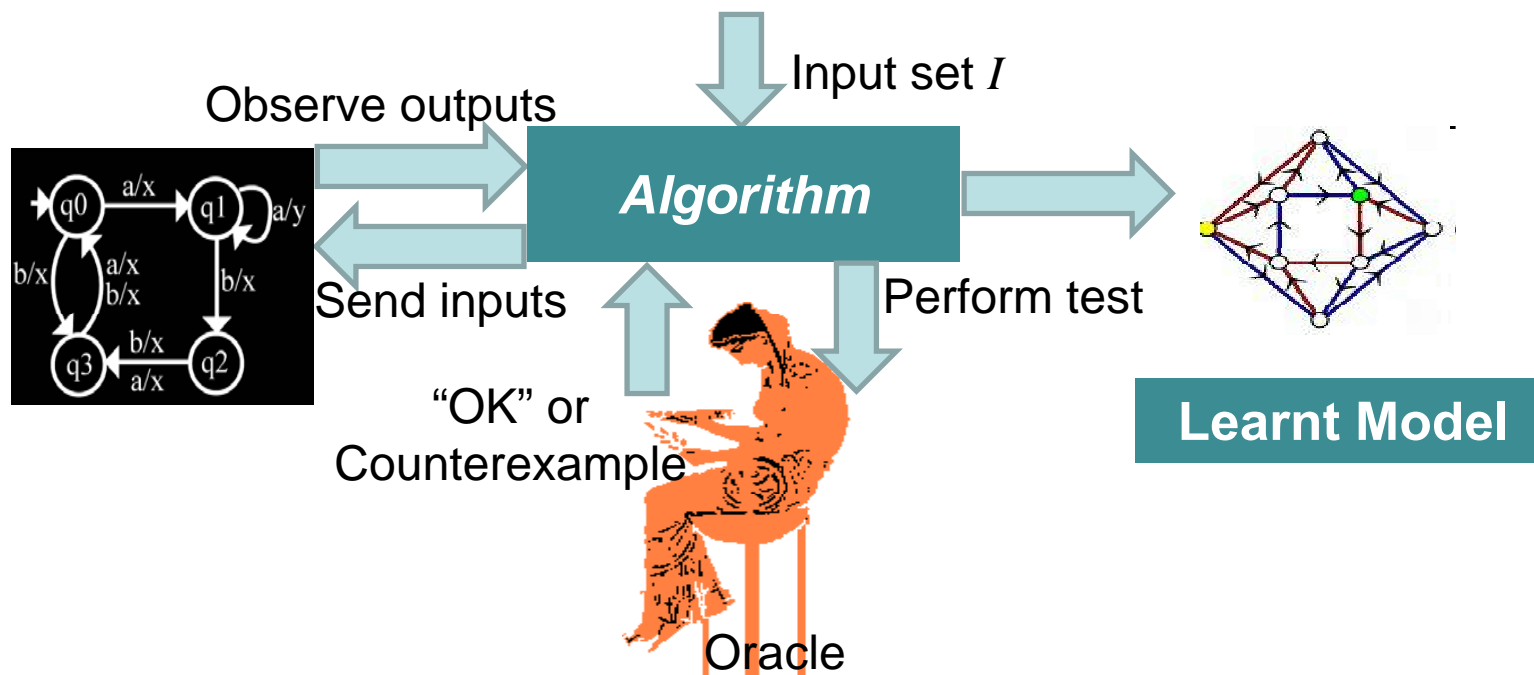
- Formal models of such components are missing
- Methods to construct the model from source code are available
- Source code for components (black boxes) is most often missing
- Construct the model by sending inputs and observing outputs

Introduction

Reverse engineering models

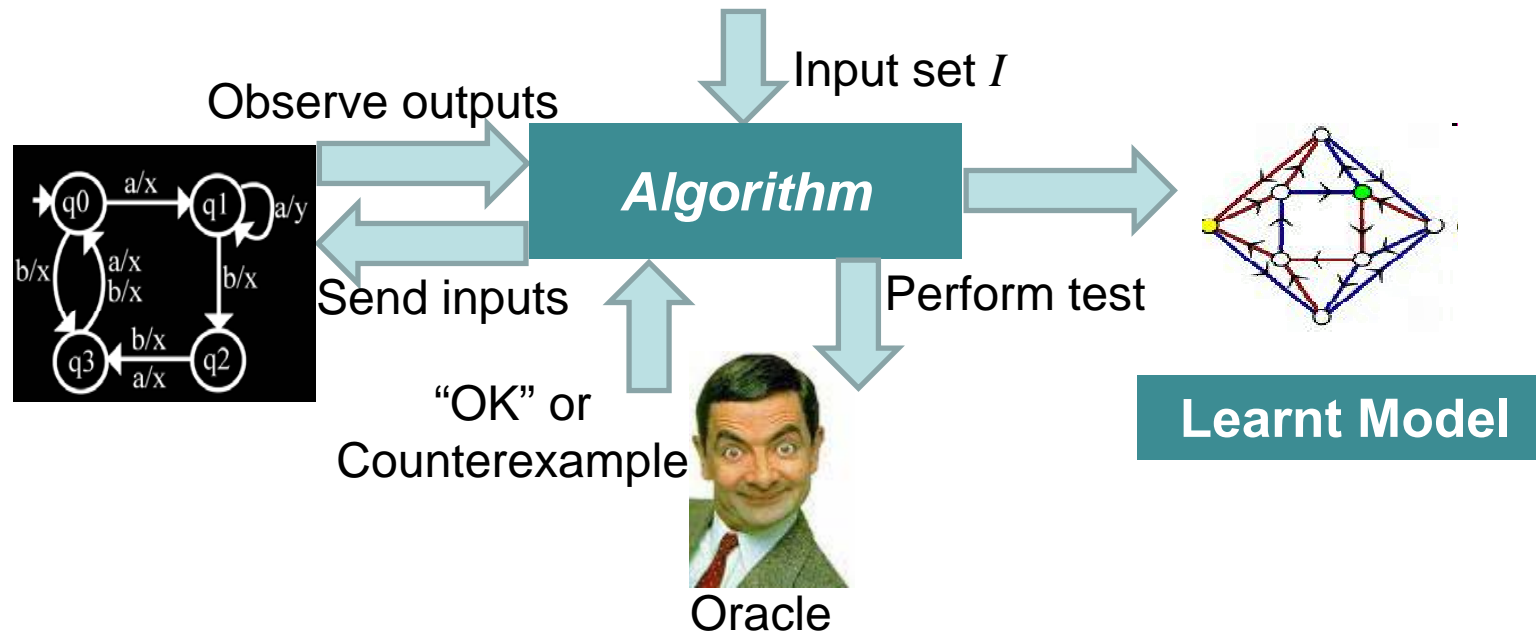


Angluin Style Learning



- Complexity $O(|I|^2 nm + |I| m n^2)$
 - I : the size of the input set
 - n : the number of states in the actual machine
 - m : the length of the longest counterexample
- Method by Shahbaz to process the counter example $O(|I|^2 n + |I| m n^2)$

Problem Statement



- Oracle is of low quality
- The length of counterexample m is important parameter for the complexity
- How we can reduce relying on the quality of counterexample
- Process the counterexample more efficiently

Learning Black Box

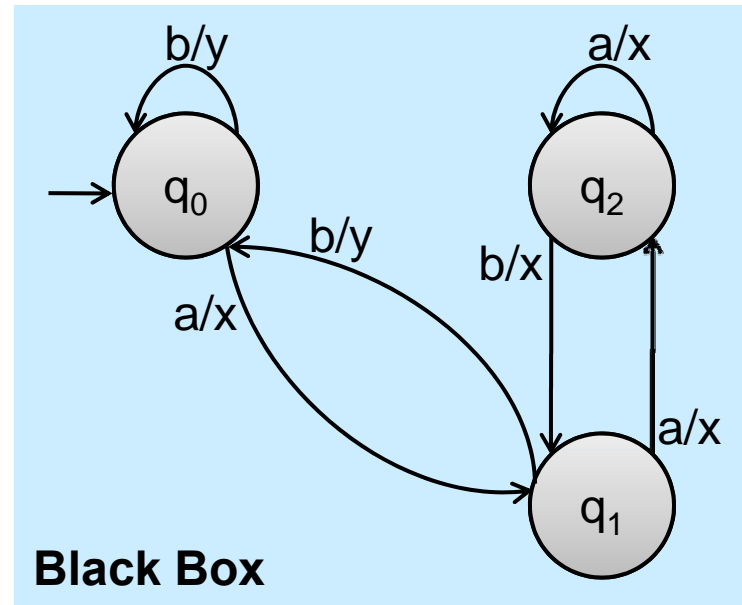
		Discriminating Sequences		
		a	b	
States S_M	ϵ	x	y	
Equivalent States S_{M-1}	a	x	y	
	b	x	y	

Observation Table

Properties:

- Closed
- Consistent

ϵ : an empty string



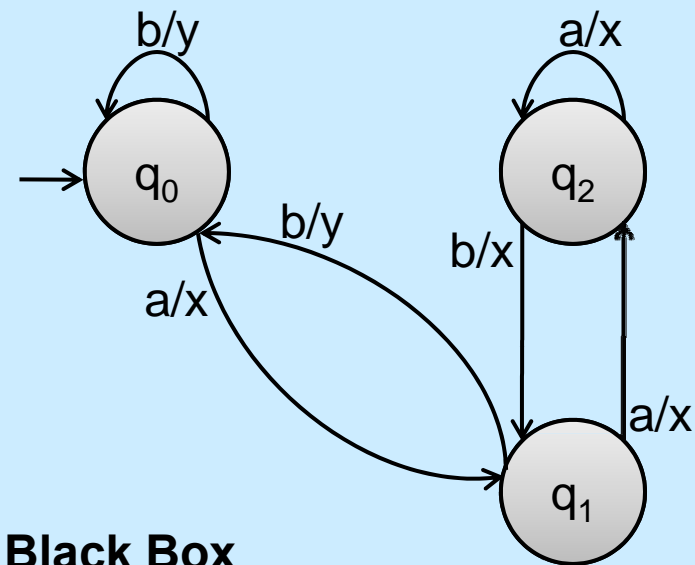
Output Queries

s.d, $s \in (St \cup EqSt)$, $d \in (DescSeq)$

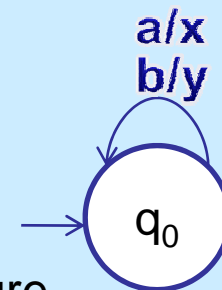
- = a/x

Constructing the Conjecture

- $S_M = \text{states}$, $S_{M \cdot I} = \text{eq. states}$
 $E_M = \text{disc.seq}$ & $T_M = \text{output fun}$
- Rows Equivalence
 $r_1, r_2 \in (S_M \cup S_{M \cdot I})$ are equivalent,
iff $\forall e \in E_M, T_M(r_1, e) = T_M(r_2, e)$
- Closeness
iff $\forall r_1 \in S_{M \cdot I} \exists r_2 \in S_M (r_1 \cong_{EM} r_2)$
- Consistency
iff $\forall i \in I \text{ if } (r_1 \cong_{EM} r_2) \Rightarrow (r_1 \cdot i \cong_{EM} r_2 \cdot i)$



Black Box



Conjecture

Counterexample: a b a b a a b

Conjecture: x y x y x x y

Black Box: x y x y x x x

Processing the Counterexample

Counterexample: a b a b a a b

Discriminating Sequences

	a	b	
ϵ	x	y	
a	x	y	
b	x	y	

**Observation Table
(Shahbaz Method)**

Discriminating Sequences

	a	b	
ϵ	x	y	
a	x	y	
b	x	y	

**Observation Table
(New Method)**

States

Eq. States

Difference between Algorithms

Processing the Counterexample

a b a b a a b

Add suffixes to **Discriminating Sequences** until new state is identified

49

	a	b	ab	aab	baab	abaab	babaab
ϵ	x	y	xy	xxx	yxxx	xyxxx	yxyxxx
a	x	y	xx	xxx	yxxx	xxxxx	yxyxxx
aa	x	x	xx	xxx	xxxx	xxxxx	xxxxxxx
b	x	y	xy	xxx	yxxx	xyxxx	yxyxxx
ab	x	y	xy	xxx	yxxx	xyxxx	yxyxxx
aaa	x	x	xx	xxx	xxxx	xxxxx	xxxxxxx
aab	x	y	xx	xxx	yxxx	xxxxx	yxyxxx

Observation Table after processing counterexample with Shahbaz method

21

	a	b	ab
ϵ	x	y	xy
a	x	y	xx
aa	x	x	xx
b	x	y	xy
ab	x	y	xy
aaa	x	x	xx
aab	x	y	xx

Observation Table after processing counterexample

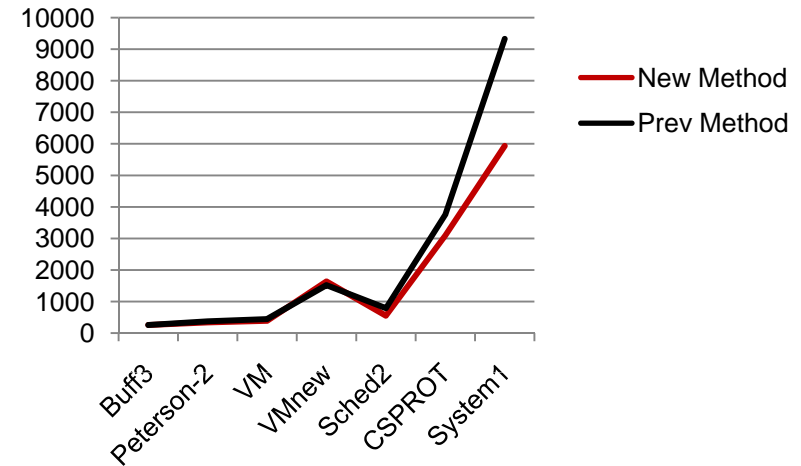
Edinburg Concurrency Workbench Results

CWB Examples

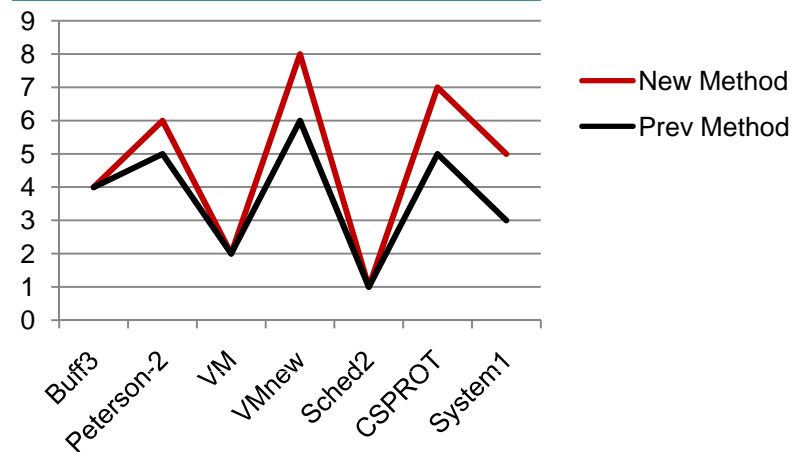
Examples	New Method	Shahbaz Method	CE New Method	CE Shahbaz Method
Buff3	259	259	4	4
Peterson-2	340	374	6	5
VM	392	448	2	2
VMnew	1638	1521	8	6
Sched2	553	790	1	1
CSPROT	3094	3757	7	5
System1	5936	9328	5	3

- Gain for number of tests conducted
- Lose for the number of counter examples processed

Tests



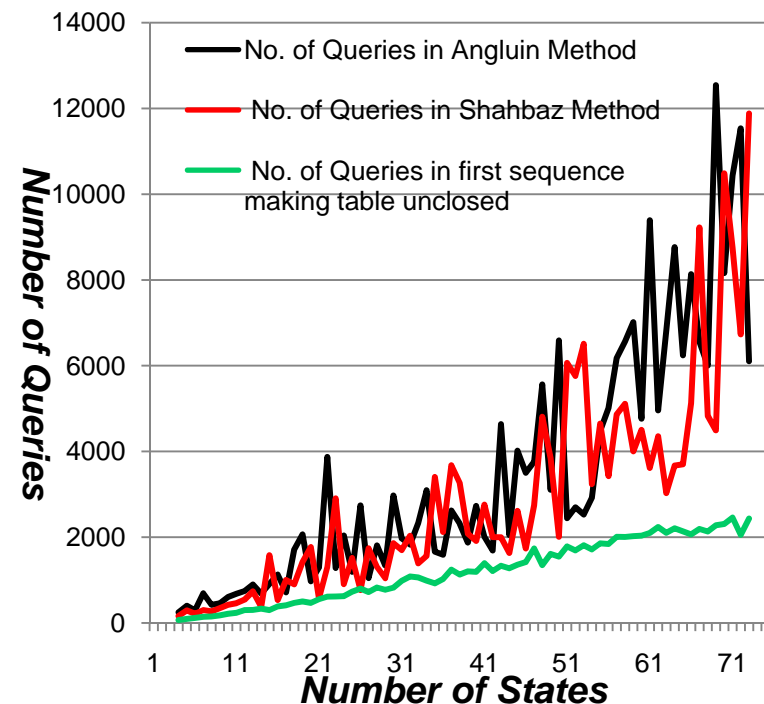
Counterexamples



Randomly Generated Machines Results

Results for learning randomly generated machines with alphabet size 3 and output size 2 for All the Variants of Learning Algorithms

State Size	Angluin Method	Shahbaz Method	New Method
3	185	115	40
4	202	148	55
5	248,6667	162,6667	70,66667
6	403,4	294,5	97,6
7	294,4	215,6	120,8
8	693,6	301	146,4
9	424,7	269,6	152
10	457,875	344,875	179,125
11	606,7857	425	217,0714
12	680,5	459,4167	238,6667
13	744,8333	543,3333	303,3333
14	898,5	738,1667	306,25
15	683,6667	375,6667	334,8333
16	907,375	1582,292	303,1667
17	1130,107	534,1429	386,4643
18	713,8	1001	412,5



Comparison of Methods

- New method less number of discriminating sequences

- Shahbaz and Angluin method consumes all the discriminating sequences

- Worst case complexity for Shahbaz and New method is same

- Gain for processing the Counterexample from Random Walk method

Perspectives

- Find efficient heuristics to bring down the average complexity of algorithm
 - Process the counterexamples
- How to deal with the abstractions to be made on actual interactions with black box components
- How to perform random walks more efficiently to obtain the counterexamples
- Enable learning algorithms to learn implementations as parameterized machines and NFSMs

Questions