Back In Black:

Towards Formal, Black Box Analysis Of Sanitizers and Filters

George Argyros*, Ioannis Stais**, Angelos Keromytis* and Aggelos Kiayias***

* Columbia University

** National and Kapodistrian University of Athens

*** The University of Edinburgh

Motivation

- Sanitizers and filters are important components of securing applications.
    - Think code injection attacks.
- Black-Box analysis is often a necessity.
    - Penetration testing, hardware testing.
- Filters need to be fast.
    - Possibility of representing with automata models.
- This talk: focus on regular expression filters.
    - Check the paper for results on sanitizers.

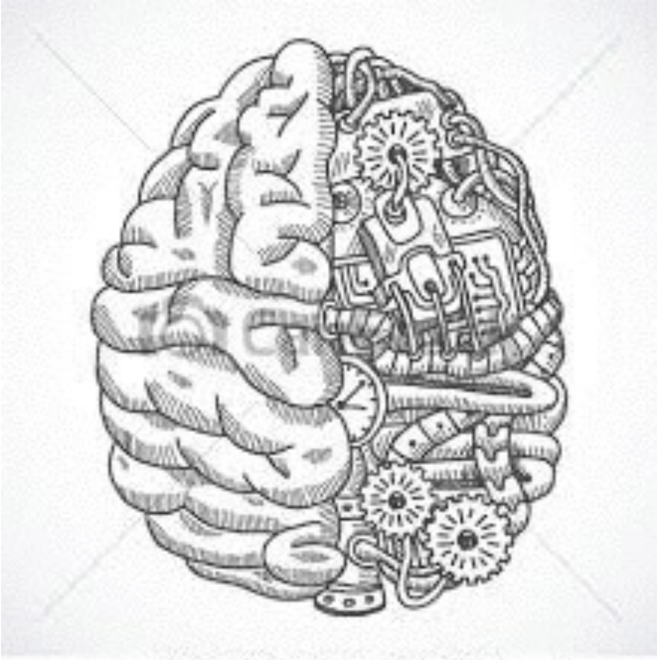
Regular Expression Filters

- Pass untrusted input through Regular Expressions.
  - Reject if match found.
- Widely employed for protecting against code injection attacks.
  - Not very robust.
- Significant components of large scale software.
  - Web Application Firewalls, IDS, DPI and others.
- Represented by Deterministic Finite State Automata (DFA).

Can we efficiently infer
Regular Expression Filters?

Exact Learning From Queries

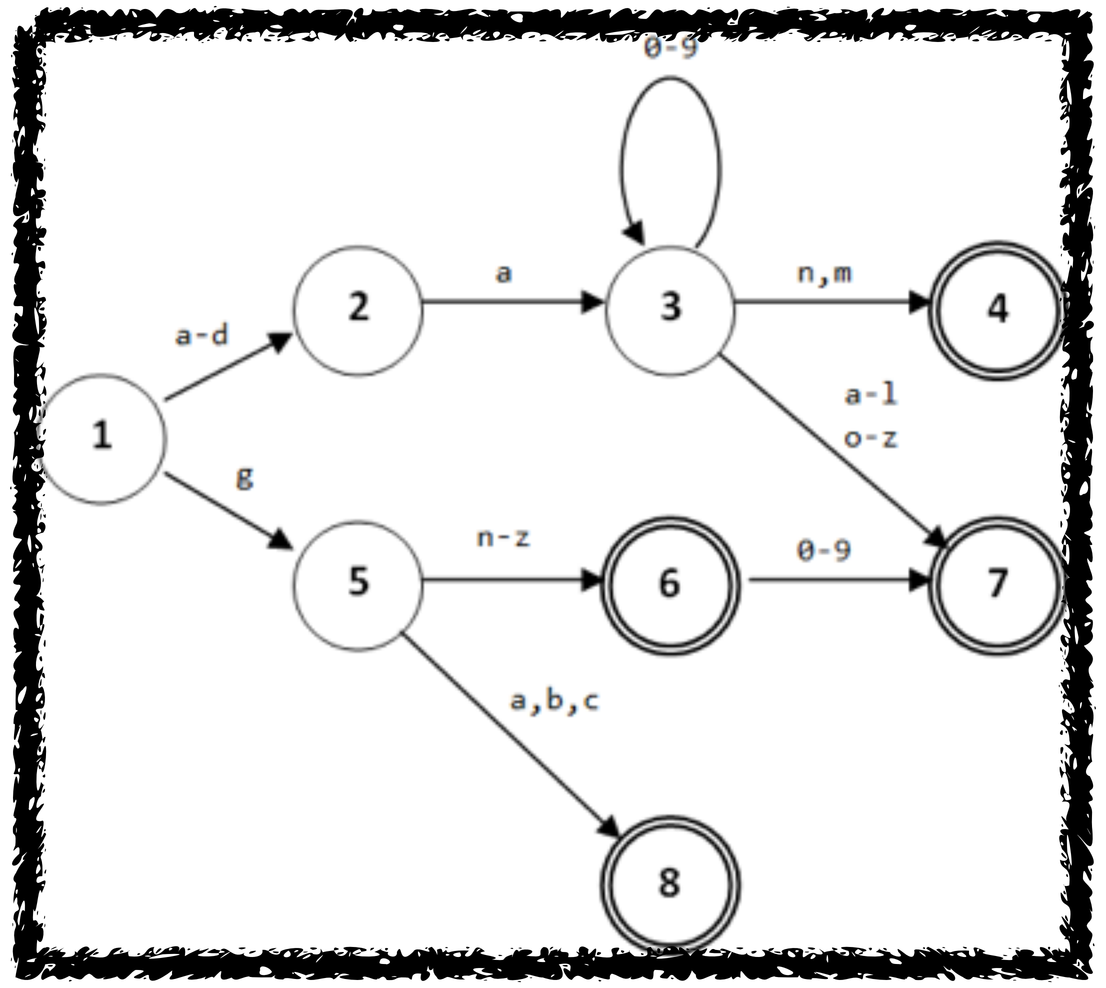
Learning Algorithm



Form of Active Learning.

Two types of Queries.

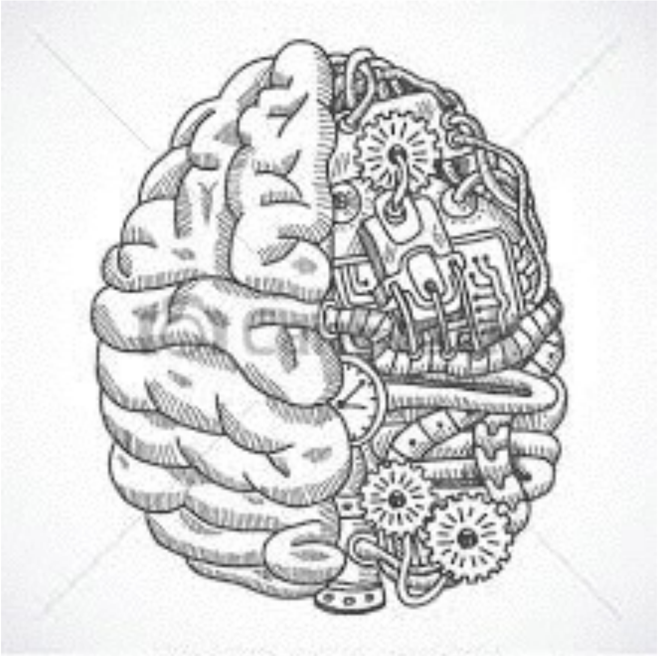
Target ***M***



Exact Learning From Queries

Membership Query

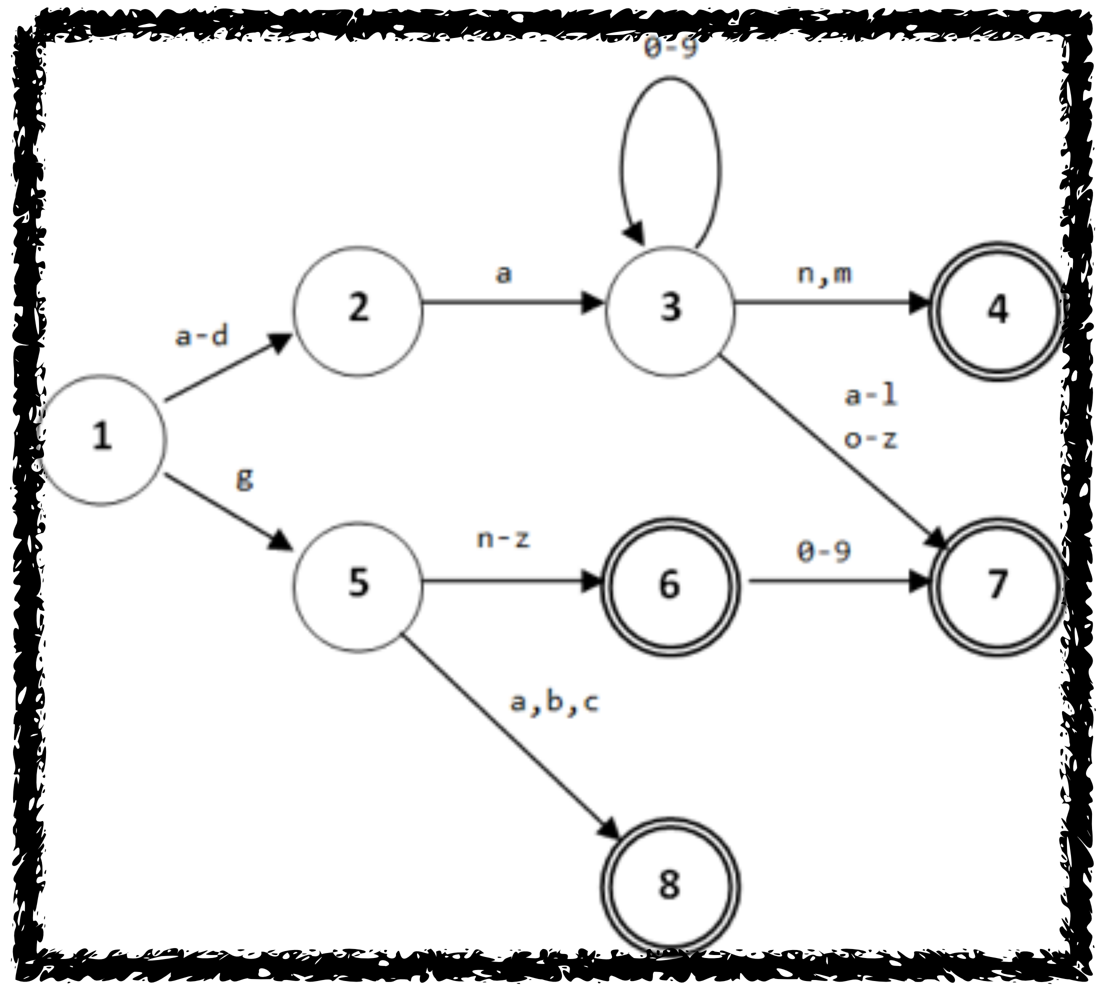
Learning Algorithm



string ***s***

Is **s** accepted by **M**?

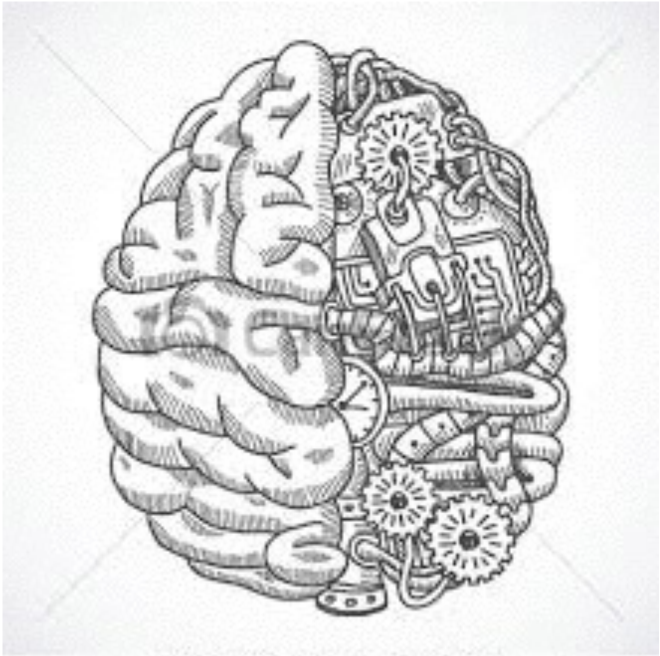
Target ***M***



Exact Learning From Queries

Equivalence Query

Learning Algorithm

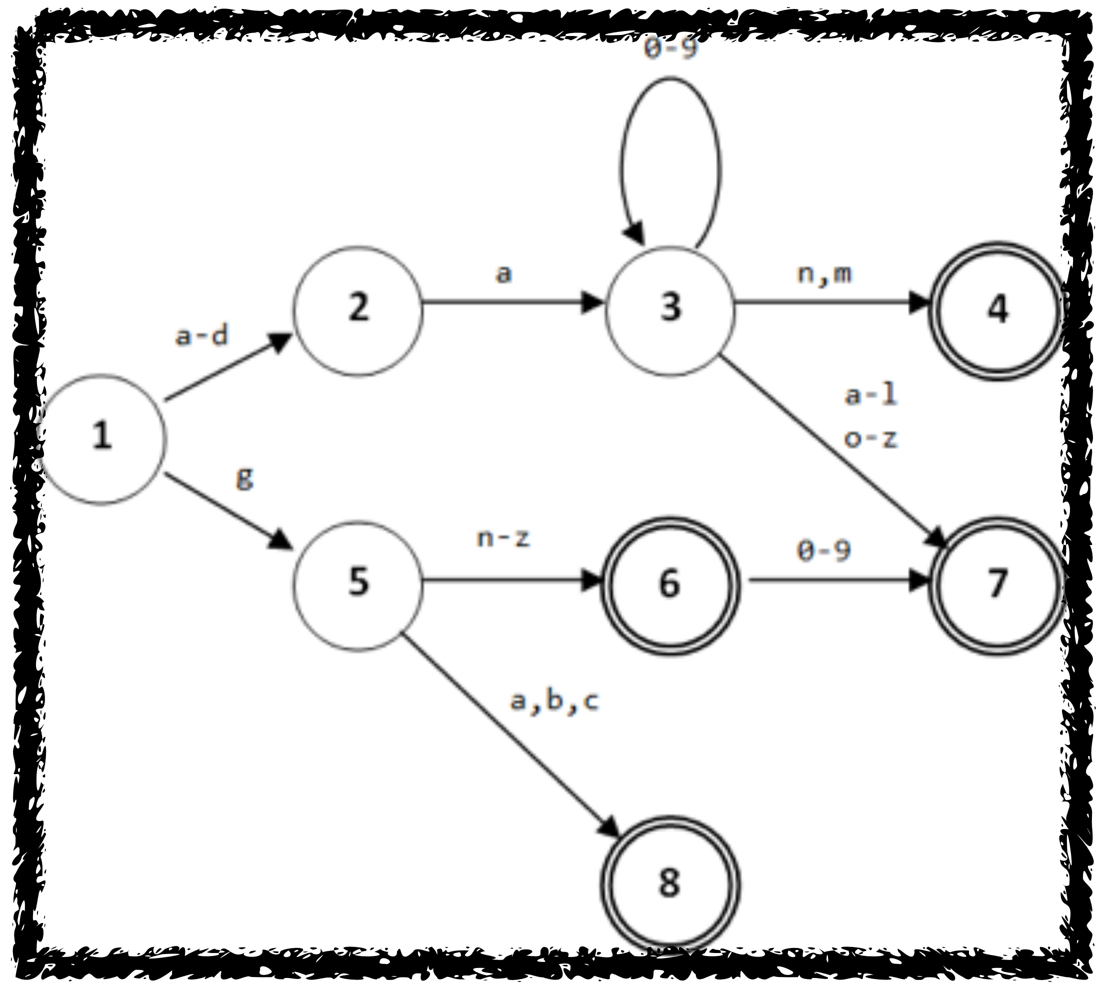


© Can Stock Photo - csp22045049

Model ***H***

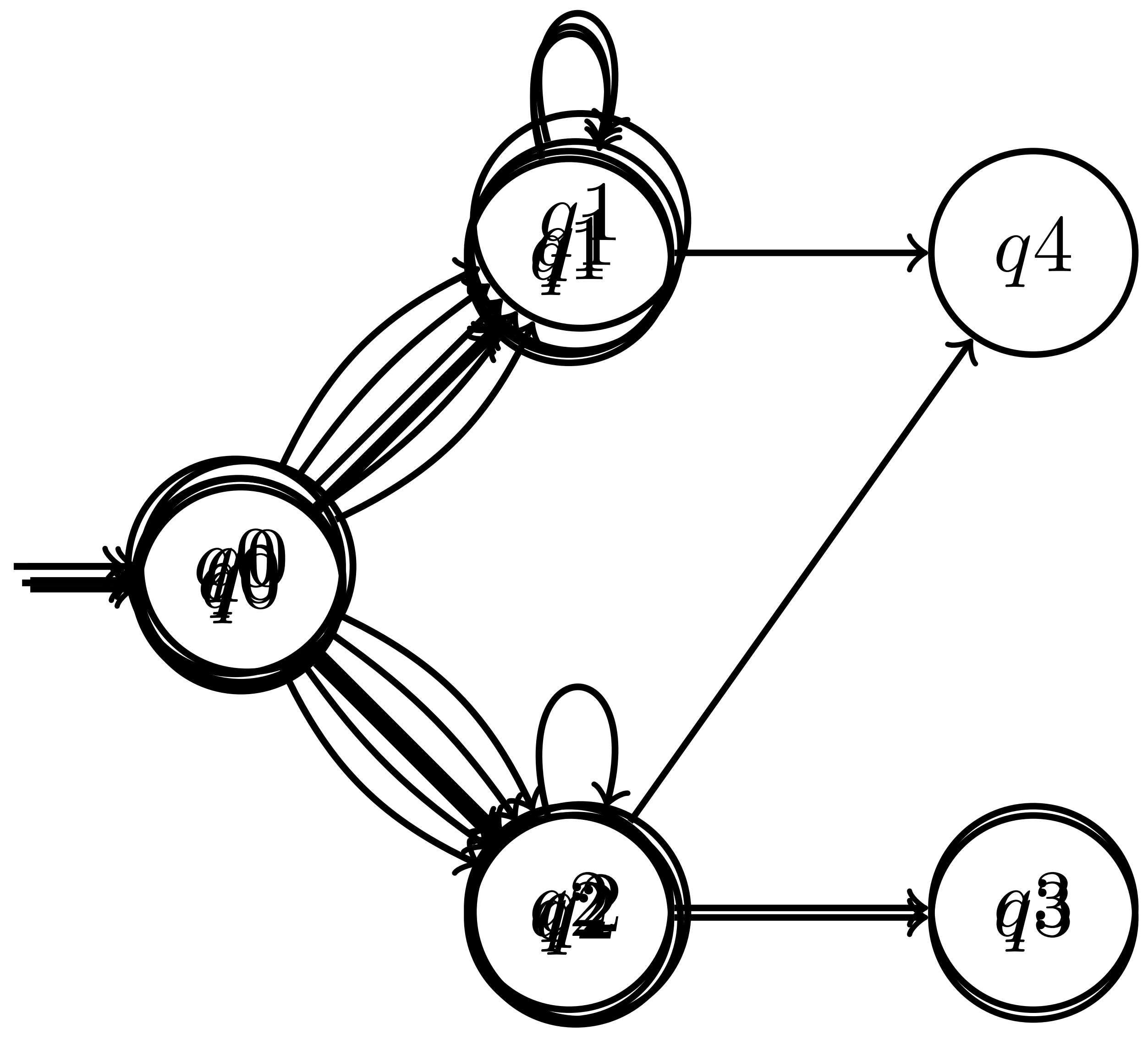
Is ***M*** = ***H*** ? Yes, or provide counterexample.

Target ***M***



Learning Deterministic Finite Automata

[Angluin '87], [Rivest-Schapire '93]

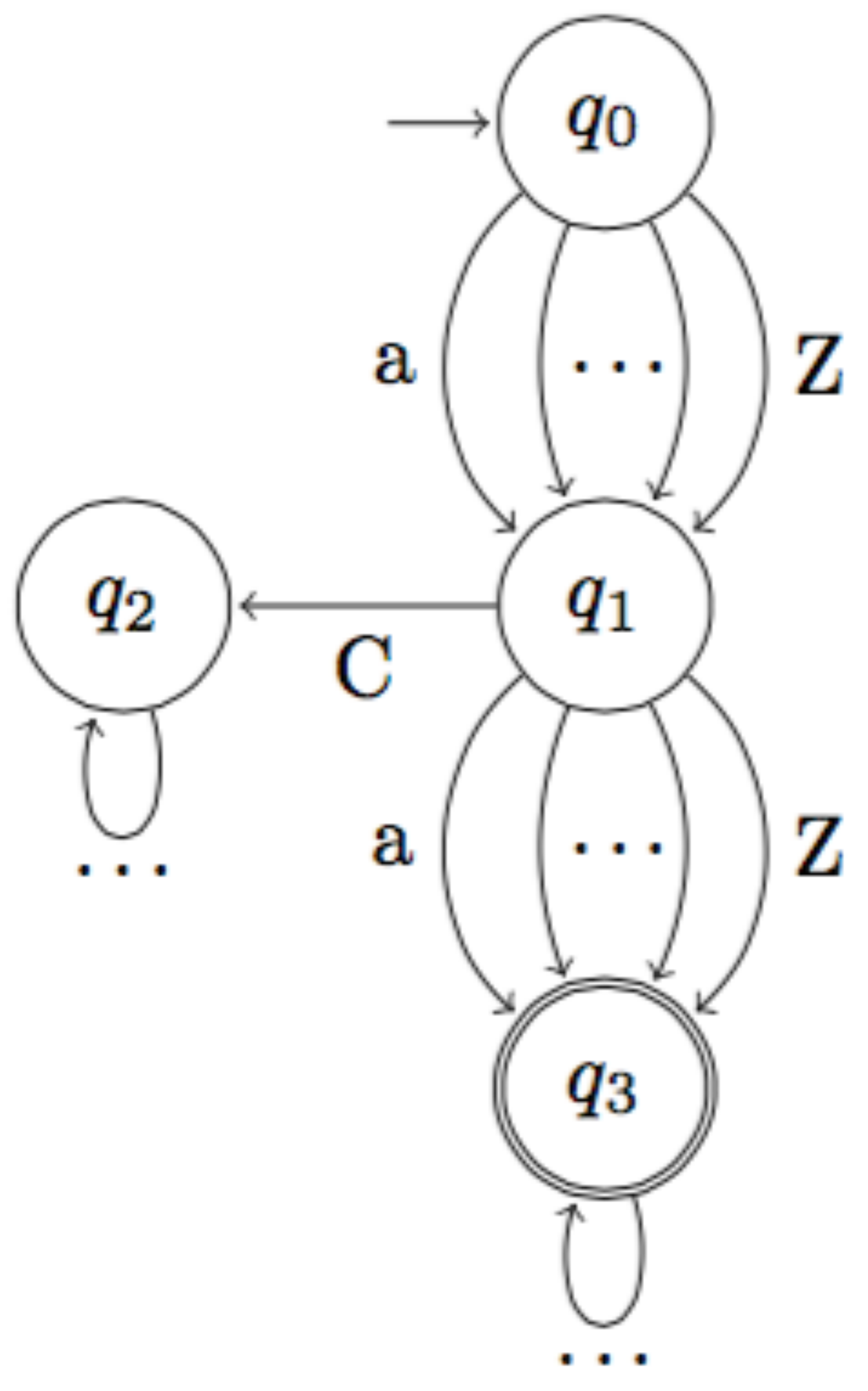


- Start with an initial state.
- Test **all** transitions from that state.
- When *valid* DFA is formed test for Equivalence.
- Counterexamples provide access to previously undiscovered states.

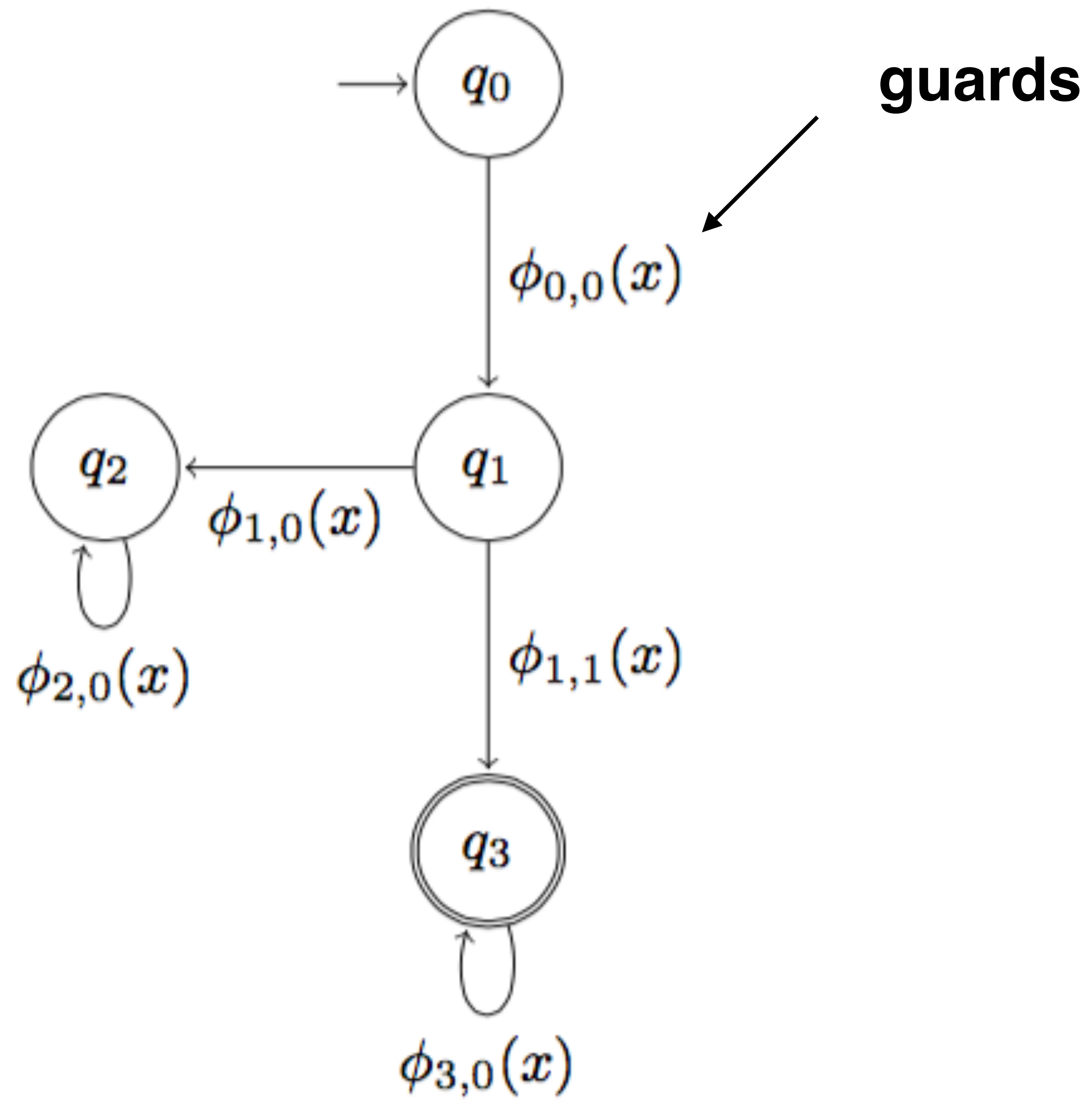
Testing all transitions is inefficient for large Alphabets!

Symbolic Finite Automata (SFA)

Classical Automata



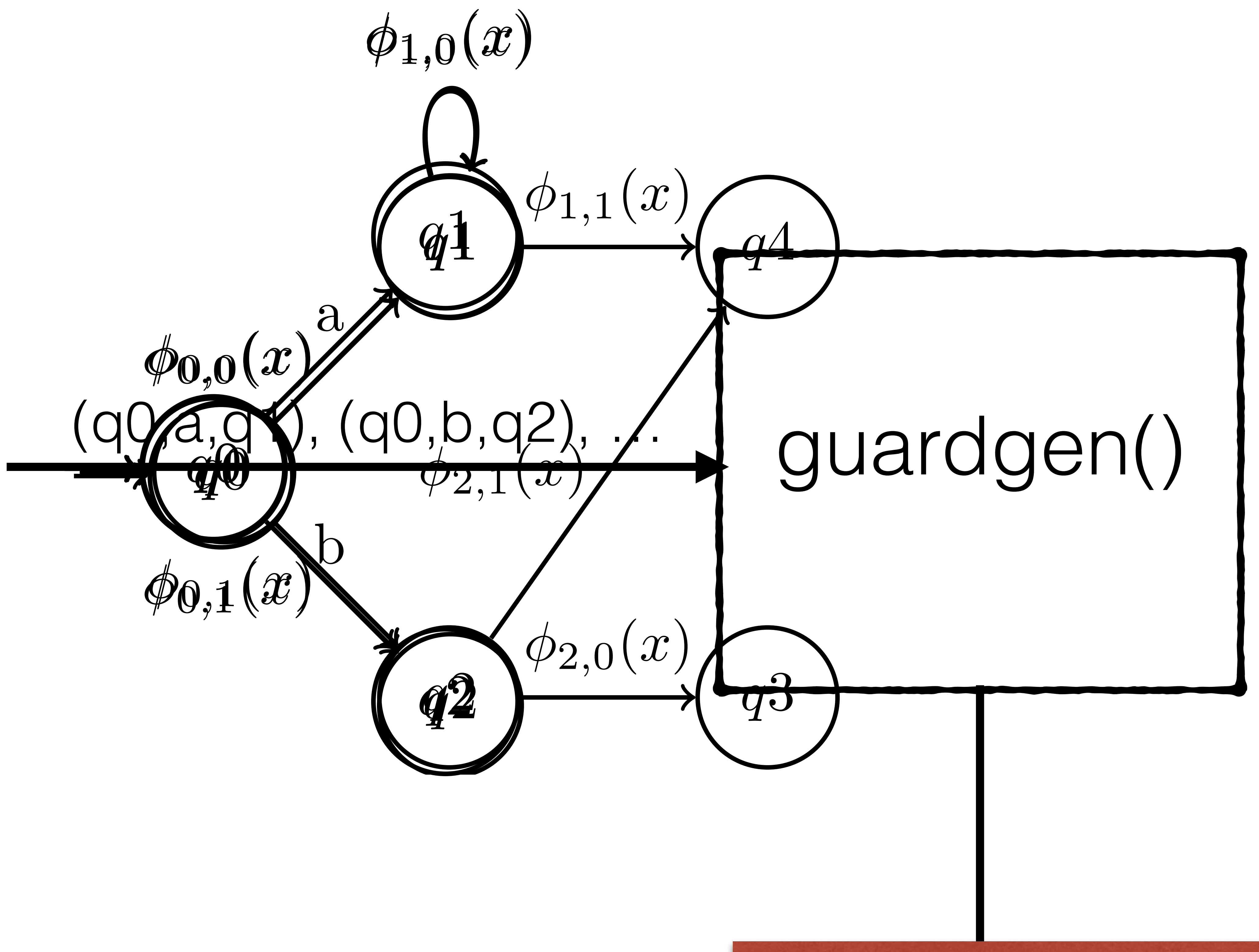
Symbolic Automata



Learning SFA: Challenges

- Alphabet may be infinite!
- How to distinguish causes for counterexamples in the models?
  - Counterexamples due to undiscovered states in the target.
  - Counterexamples due to inaccurate transition guards.

Learning Symbolic Finite Automata



- Start with an initial state.
- Test **sample** transitions from that state.
- Use sample transitions as training set to generate guards.
- Novel counterexample processing method to handle incorrect guards.

Convergence under natural assumptions on guardgen()

Is Exact Learning From Queries a realistic model?

Is Exact Learning from Queries a realistic model?

- Membership Queries? *Test whether input is rejected by the filter.*
- Equivalence Queries?

Grammar Oriented Filter Auditing

or

How to Implement an Equivalence Oracle

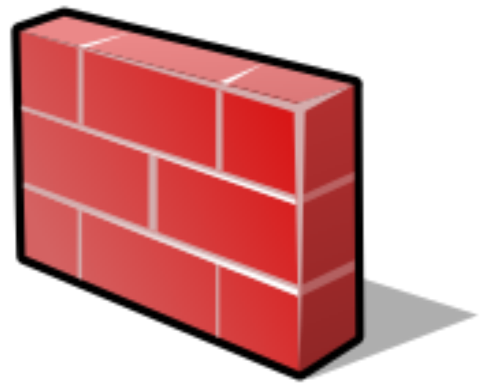
Grammar Oriented Filter Auditing (GOFA)

Grammar Oriented Filter Auditing (GOFA)

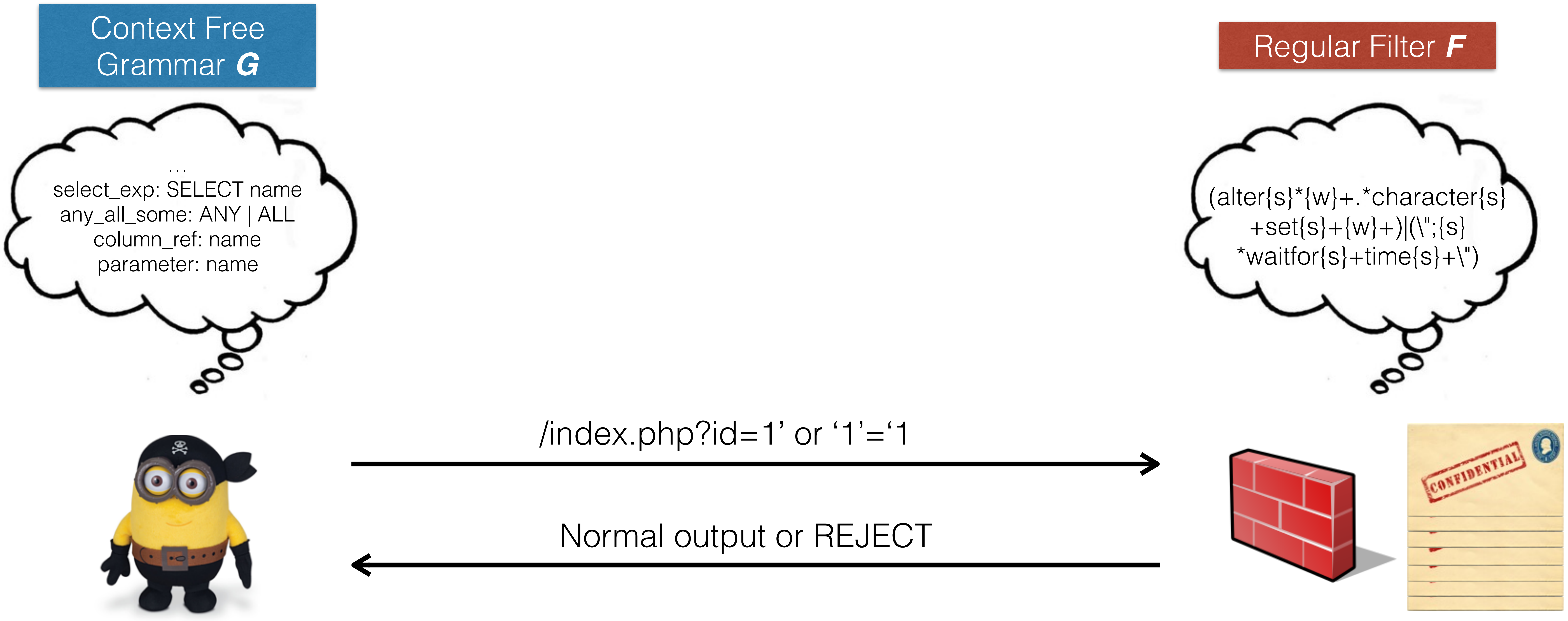
Context Free Grammar ***G***

Grammar Oriented Filter Auditing (GOFA)

Context Free Grammar ***G***



Grammar Oriented Filter Auditing (GOFA)



Grammar Oriented Filter Auditing (GOFA)

Context Free Grammar ***G***

...
select_exp: SELECT name
any_all_some: ANY | ALL
column_ref: name
parameter: name

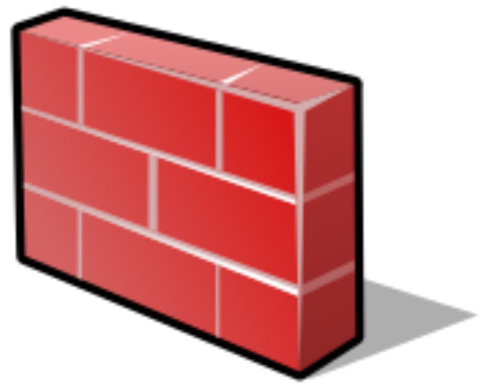
May Require Exponential Number of Queries!

Regular Filter ***F***

(alter{s}*{w}+.*character{s}
+set{s}+{w}+)|(\";{s}
*waitfor{s}+time{s}+\")

/index.php?id=1’ or ‘1’=‘1

Normal output or REJECT



Solving GOFA

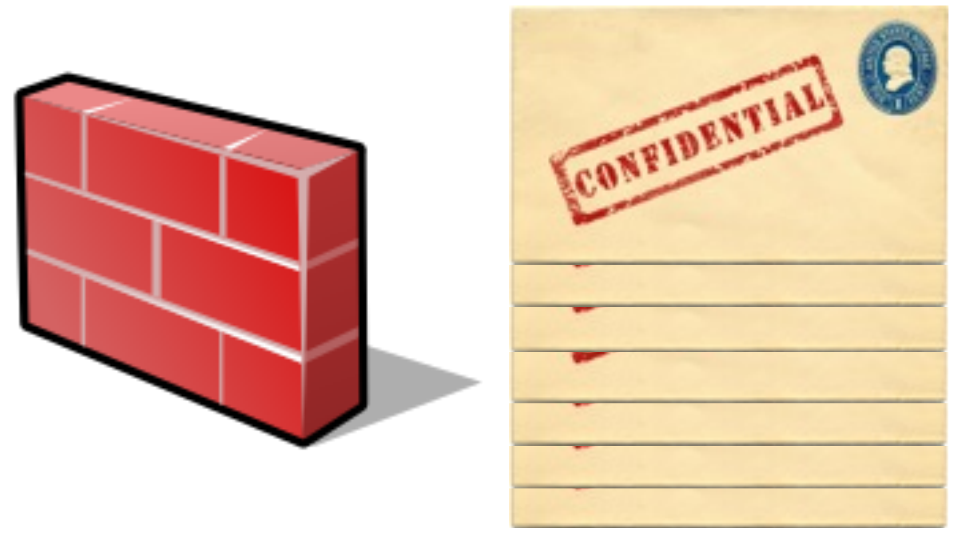
- In an ideal (White-Box) world both ***G*** and ***F*** are available:
  1. Compute $\bar{F}$, the set of strings not rejected by ***F.***
  2. Check $\mathcal{L}(G \cap \bar{F})$ for emptiness.

- In practice ***F*** is unavailable.
  - Learn a model for ***F***!

Solving GOFA

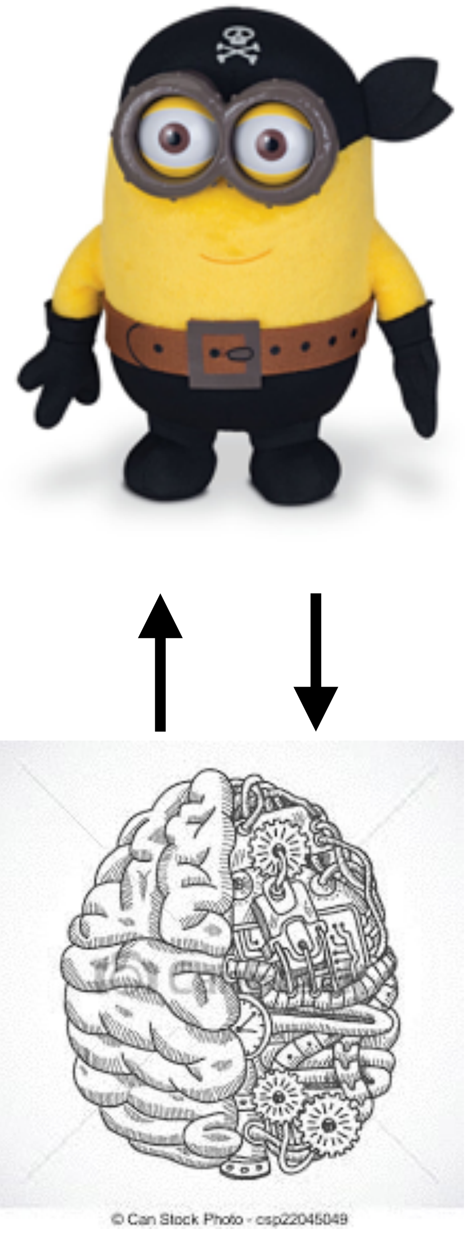
Context Free Grammar ***G***

Regular Filter ***F***

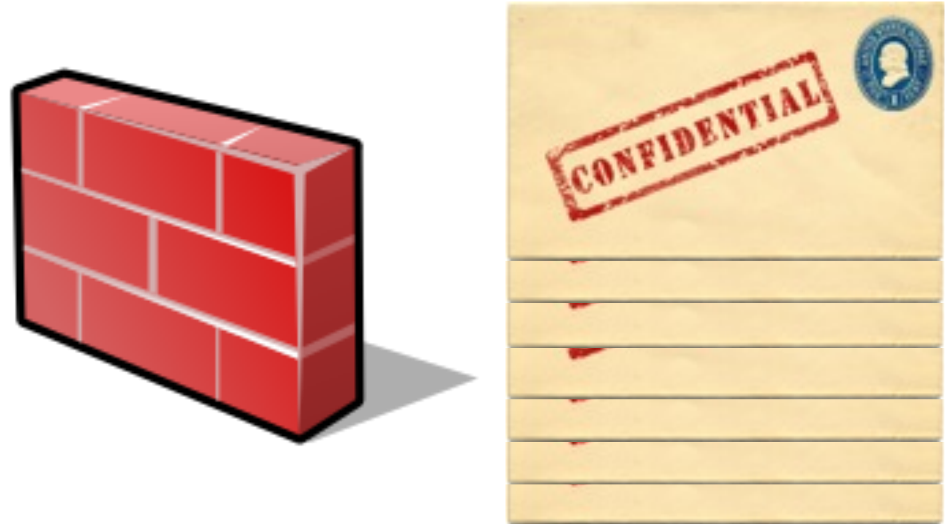


Solving GOFA

Context Free Grammar ***G***



Regular Filter ***F***



Solving GOFA

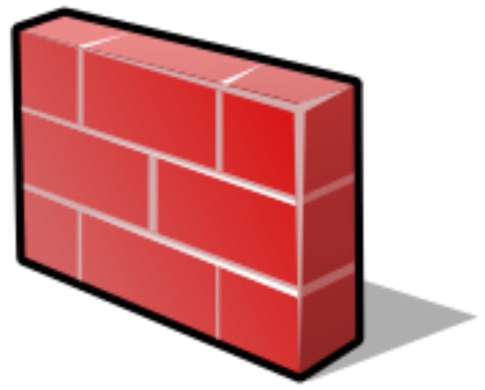
Membership Query

Context Free Grammar ***G***

string ***s***

True if REJECT is returned
False otherwise

Regular Filter ***F***



Solving GOFA

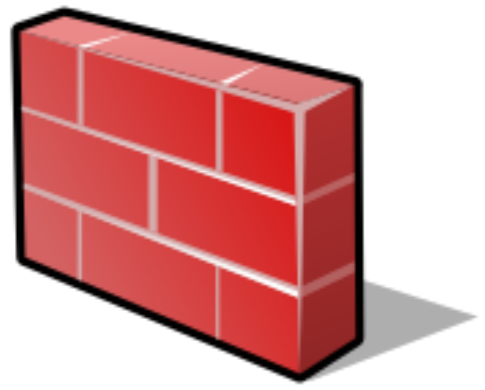
Equivalence Query

One Membership Query per Equivalence Query!

Context Free Grammar ***G***

Regular Filter ***F***

$$s \leftarrow \mathcal{L}(G \cap \bar{H})$$



H

© Can Stock Photo - csp22045049

If REJECT:

s is a counterexample for ***H***.

Otherwise:

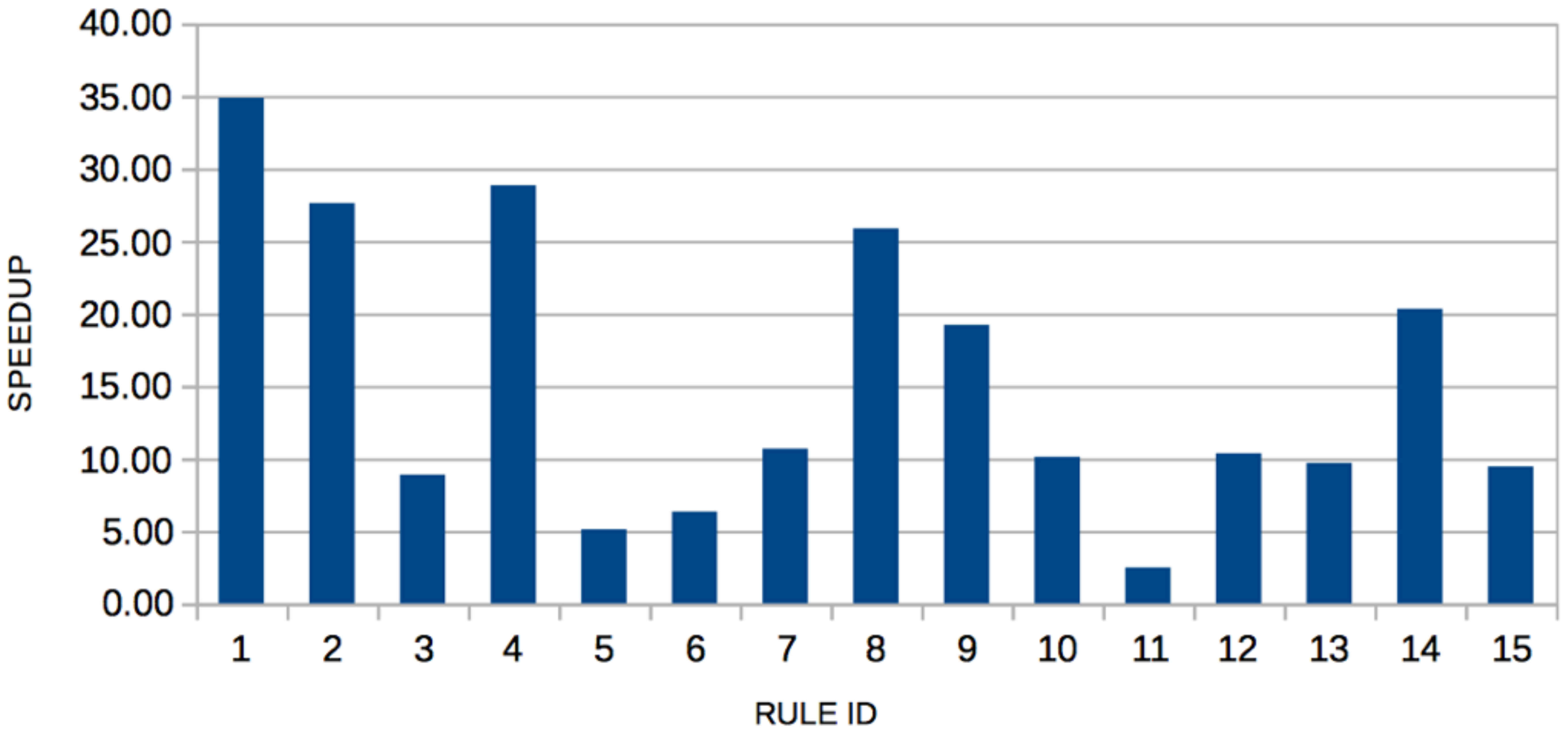
s is a bypass for the filter ***F***.

Evaluation

Experimental Setup

- 15 Regular Expression Filters from popular Web Application Firewalls(WAFs).
    - 7 - 179 states.
    - 13 - 658 transitions.
- Alphabet size of 92 symbols.
    - Includes most printable ASCII characters.

DFA vs SFA Learning



✓On average 15x less queries.

✓Increase in Equivalence queries.

✓Speedup is not a simple function of the automaton size.

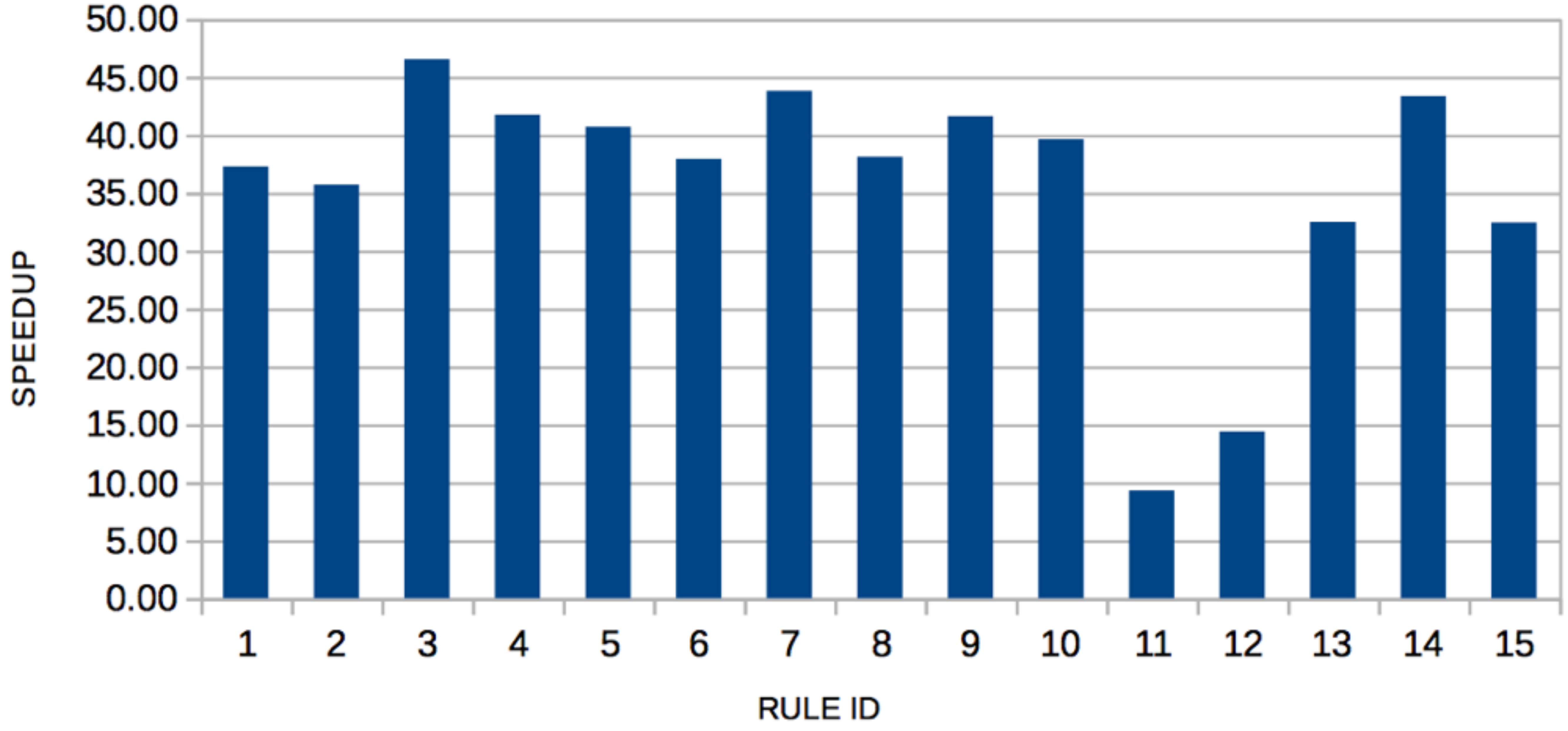
DFA vs SFA Learning

IDS RULES			DFA LEARNING		SFA LEARNING		
ID	STATES	ARCS	MEMBER	EQUIV	MEMBER	EQUIV	SPEEDUP
1	7	13	4389	3	118	8	34.86
2	16	35	21720	3	763	24	27.60
3	25	33	56834	6	6200	208	8.87
4	33	38	102169	7	3499	45	28.83
5	52	155	193109	6	37020	818	5.10
6	60	113	250014	7	38821	732	6.32
7	66	82	378654	14	35057	435	10.67
8	70	99	445949	15	17133	115	25.86
9	86	123	665282	27	34393	249	19.21
10	115	175	1150938	31	113102	819	10.10
11	135	339	1077315	24	433177	4595	2.46
12	139	964	1670331	29	160488	959	10.35
13	146	380	1539764	28	157947	1069	9.68
14	164	191	2417741	29	118611	429	20.31
15	179	658	770237	14	80283	1408	9.43
						AVG=	15.31

GOFA Algorithm Evaluation

- Assume that the grammar G does **not** contain a string that bypasses the filter.
  - How good is the approximation of the filter obtained?
  - How efficient is SFA Learning in the GOFA context?
- What is an appropriate grammar to perform this experiment?
  - Use the filter itself as the input grammar!
  - Intuitively, a **maximal** set that does not include a bypass.

DFA vs SFA Learning in GOFA



✓SFA utilizes x35 less queries.

✓States recovered:

- DFA: 91.95%
- SFA: 89.87%

GOFA: Evading WAF

- Handcrafted grammar with valid suffixes of SQL statements.
  - SELECT * from table WHERE id=**S**
  - Simulates an SQL Injection attack.
- Test GOFA algorithm against live installations of ModSecurity and PHPIDS.
  - Both systems include <u>*non regular*</u> anomaly detection components.

GOFA: Evading WAF

Evasions found for both web application firewalls.

- ✓ **Authentication Bypass:** *1 or isAdmin like 1*
- ✓ **Data Retrieval:** *1 right join users on author.id = users.id*

Evasion attacks aknowledged by ModSecurity team.

Conclusions

- SFAs provide an efficient way to infer regular expressions.
- SFA learning can provide insights for *non regular systems*.
- Similar techniques derived for sanitizers, more in the paper!
- Large space for improvements over presented learning algorithm.
  - Smarter guard generation algorithms.
- We envision ***assisted*** Black-Box testing of sanitizers and filters.
  - Auditor will correct inaccuracies of models.
  - Derive concrete attacks from abstract language constructs.

Back In Black:

Towards Formal, Black Box Analysis Of Sanitizers and Filters

George Argyros*, Ioannis Stais**, Angelos Keromytis* and Aggelos Kiayias***

* Columbia University

** National and Kapodistrian University of Athens

*** The University of Edinburgh