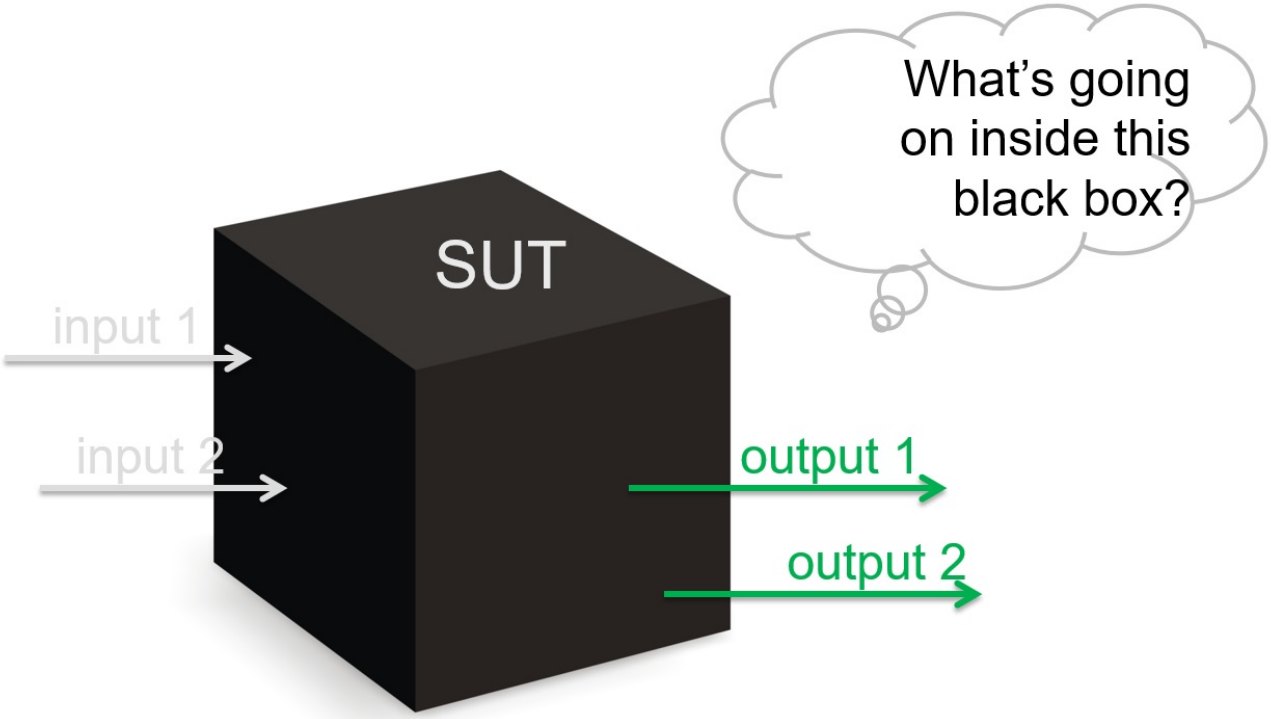
# Finding Security Vulnerabilities in Protocol Implementations Using Active Automata Learning

Frits Vaandrager

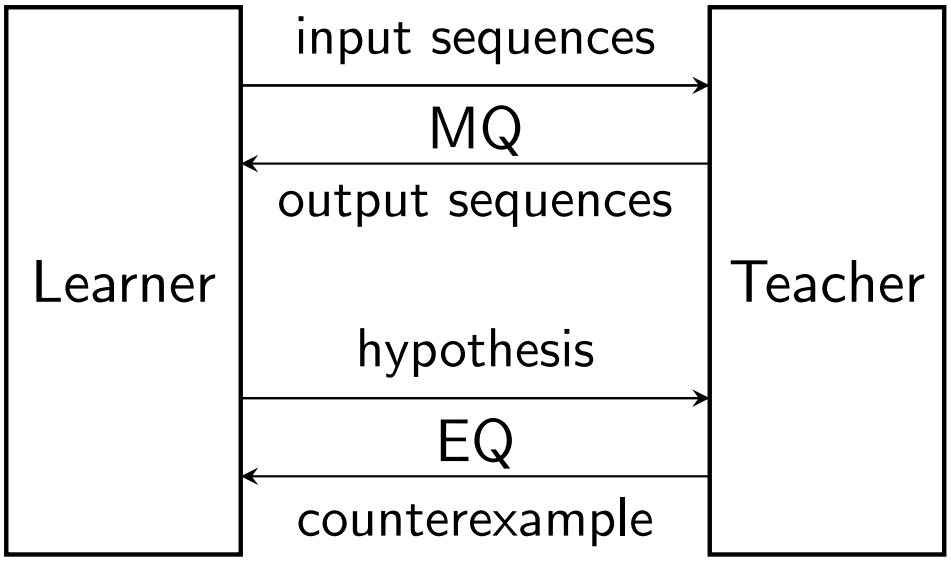
Radboud University Nijmegen

ICGI, Wroclaw, September 2018

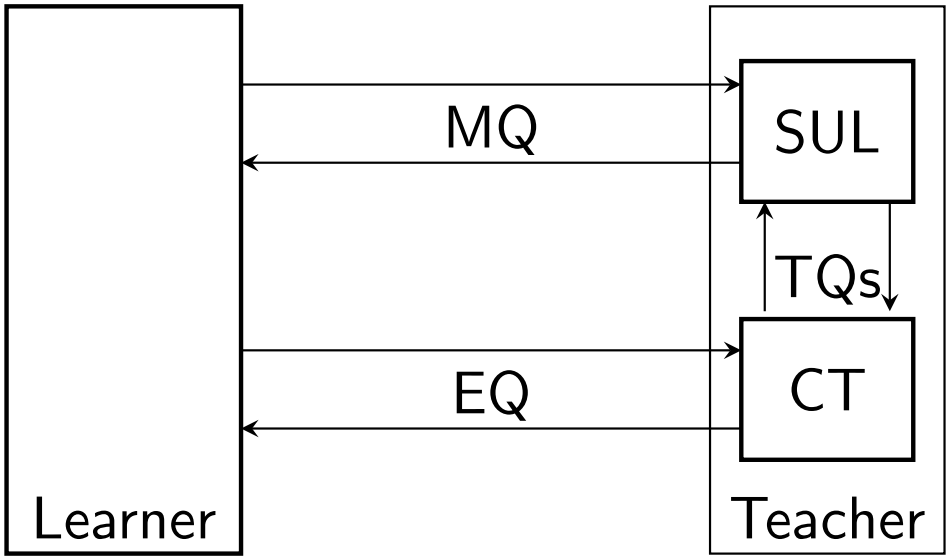
# Research Question



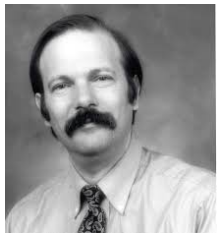
# Minimally adequate teacher (Angluin)




# Black box checking (Peled, Vardi & Yannakakis)

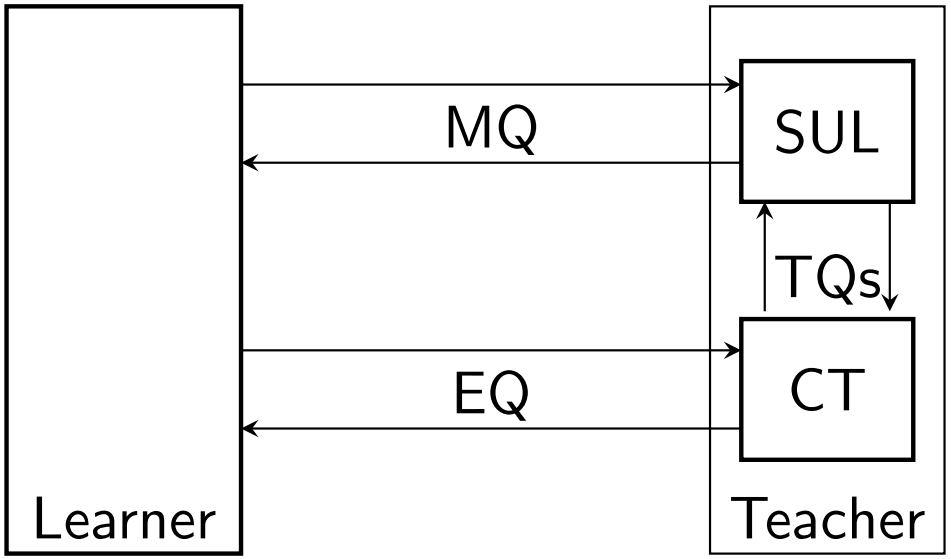




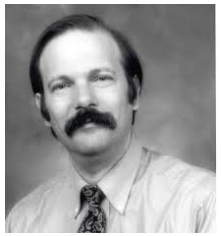



Learner: Formulate hypotheses
Conformance Tester (CT): Test correctness hypotheses

# Black box checking (Peled, Vardi & Yannakakis)








Learner: Formulate hypotheses
Conformance Tester (CT): Test correctness hypotheses

Model learning and conformance testing two sides of same coin!

# LearnLib


HOME NEWS DOWNLOADS FEATURES RESOURCES TEAM HELP

Welcome to the **LearnLib** home page! LearnLib is a *free, open-source* (Apache License 2.0) Java library for *active automata learning*. It is mainly being developed at the Chair for Programming Systems at TU Dortmund University, Germany; a complete list of contributors can be found on the team page.

**Note:** The open-source LearnLib is a from-scratch re-implementation of the former closed-source version. See the features page for a comparison of the feature sets of the two version.

**Background**

- Read some Papers on LearnLib
- Papers citing LearnLib at Google Scholar

Search

EXTERNAL LINKS

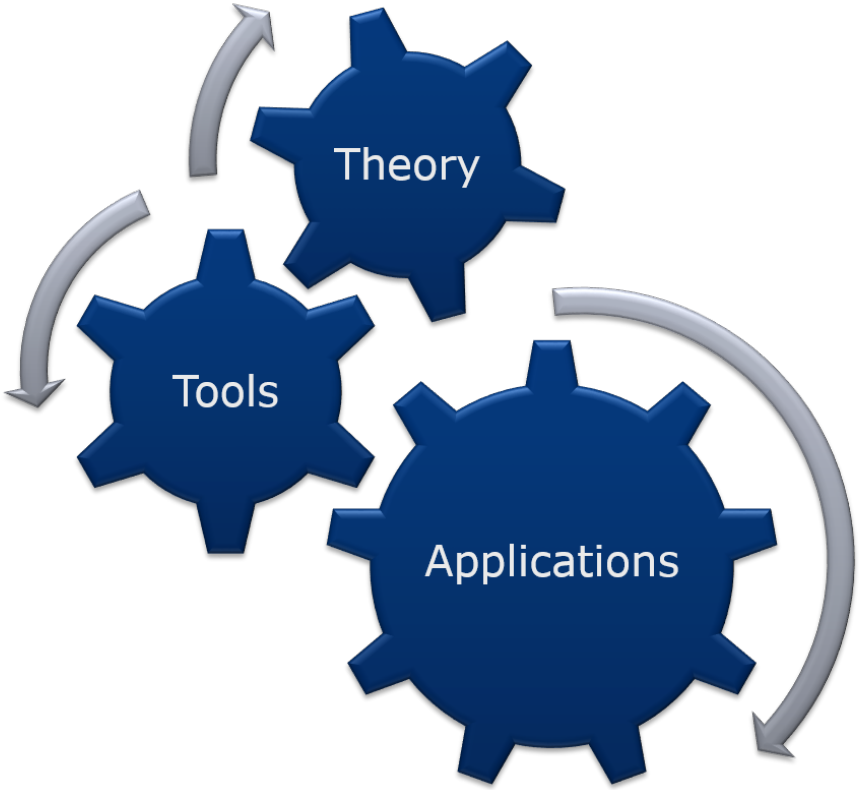
LearnLib @ GitHub
AutomataLib @ GitHub

RECENT POSTS

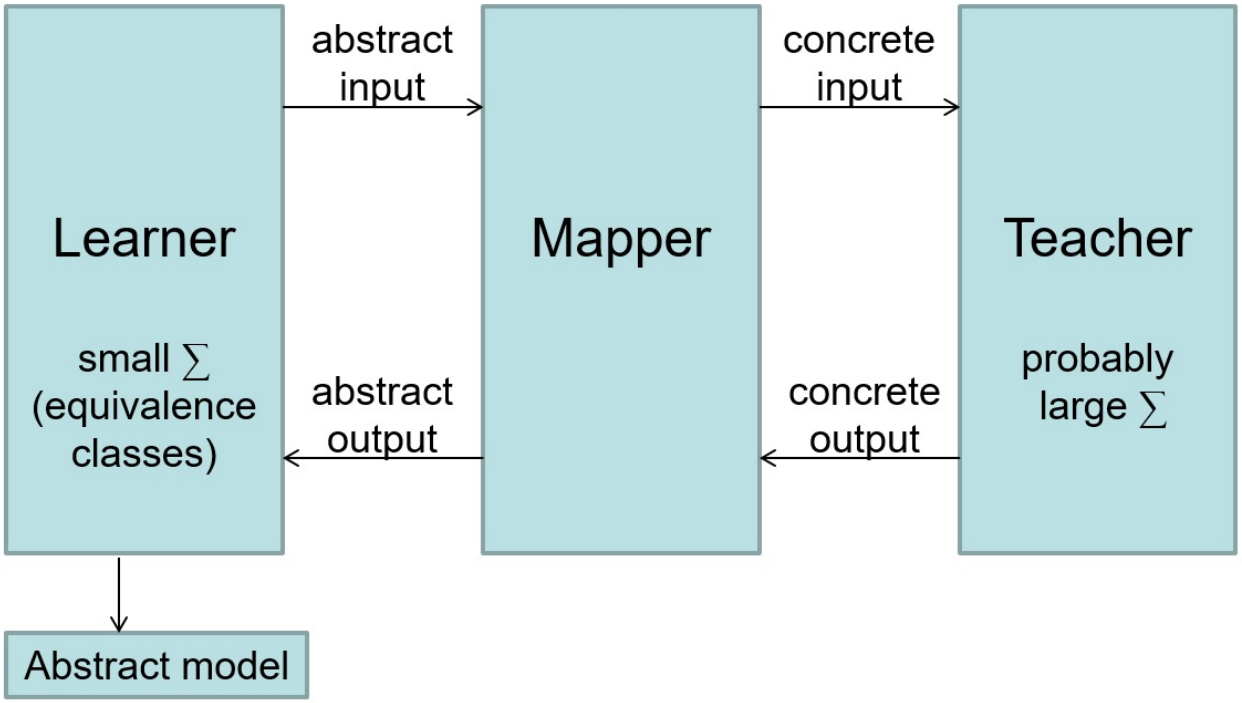
Open Source release of LearnLib


Prof. Dr. Bernhard Steffen

# Research method (FMSD, 2015)



# A theory of mappers



# A theory of mappers (cnt)

Formally, a mapper can be viewed as a transducer (deterministic Mealy machine). A mapper $\mathcal{A}$ induces an abstraction operation $\alpha_{\mathcal{A}}$ and a concretization operator $\gamma_{\mathcal{A}}$.

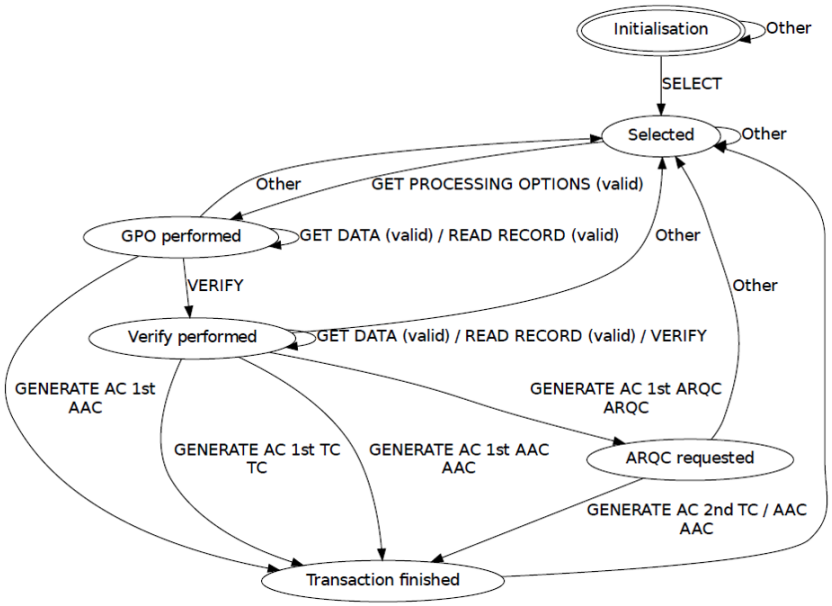
**Theorem**

*For a mapper $\mathcal{A}$ and nondeterministic Mealy machines $\mathcal{M}$ and $\mathcal{H}$, $\alpha_{\mathcal{A}}(\mathcal{M}) \leq \mathcal{H}$ implies $\mathcal{M} \leq \gamma_{\mathcal{A}}(\mathcal{H})$*

**Theorem**

*Suppose $\alpha_{\mathcal{A}}(\mathcal{M})$ has no transitions with output $\perp$. Then $\mathcal{M} \leq \gamma_{\mathcal{A}}(\mathcal{H})$ implies $\alpha_{\mathcal{A}}(\mathcal{M}) \leq \mathcal{H}$.*

# EMV protocol (Aarts et al, 2013)




- EMV = Europay/Mastercard/Visa
- Compatibility between smartcards and terminals
- SEPA requires EMV compliance
- EMV standard has >700 pages
- Learning took at most 1500 membership queries, less than 30 minutes
- Useful for fingerprinting cards

# E.dentifier2 (WOOT'14)

NOS | Zoeken binnen NOS.nl | Vandaag 8° | Morgen 9° | Verkeer 2 km | AEX 379,41 | meer

NOS.nl | Nieuws | Binnenland | Buitenland | Politiek | Economie | Opmerkelijk | Sport | Televisie | Radio | Mobiel

Economie | Overzicht | Nieuwsarchief | Video & audio | Journaal 24 | Politiek 24 | Dossiers | Financieel

## E-bankieren ABN Amro kwetsbaar

donderdag 16 aug 2012, 18:02 (Update: 17-08-12, 08:39)


ABN Amro NOS

Internetbankieren bij ABN Amro is gevoelig voor fraude. Internetcriminelen kunnen sommige betalingstransacties onderscheppen, aanpassen en doorsluizen naar hun eigen rekening.

Video


**Internetbankieren met E.dentifier ABN Amro onveilig**

Internetbankieren met de E.dentifier2 van de ABN AMRO is onveilig als ie wordt gebruikt met een USB-kabel. Dat zegt de Radboud... (meer)

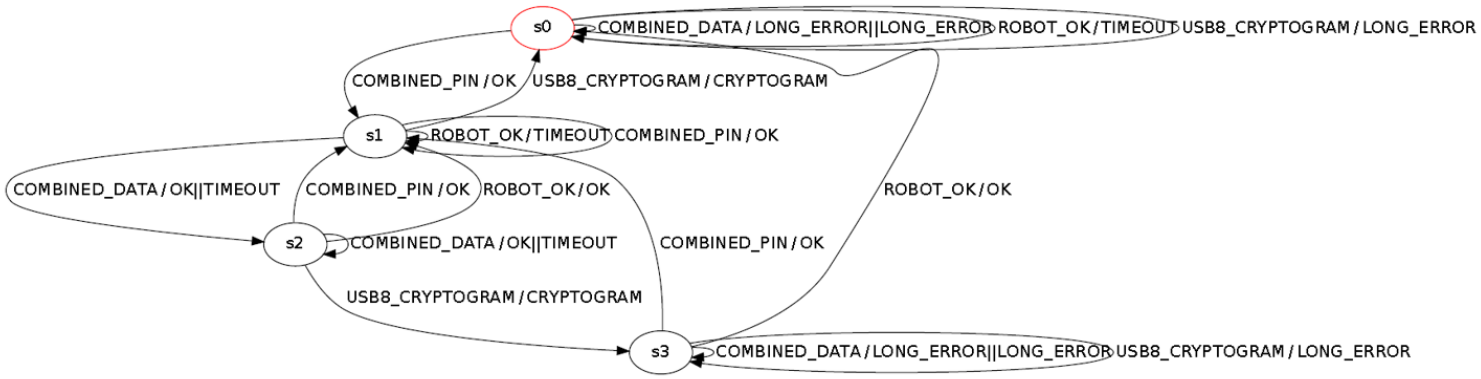
Audio

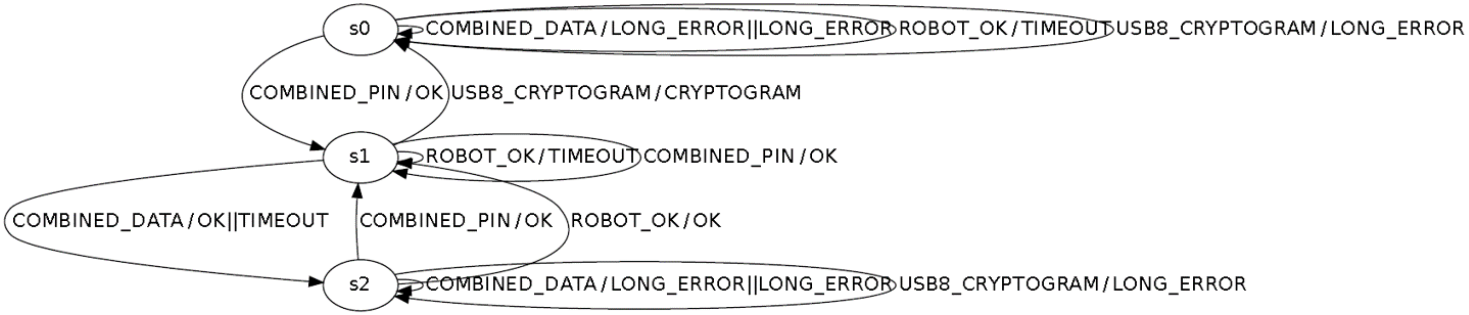
**Softwarefout in identifier ABN Amro**

Onderzoekers van de Radboud Universiteit hebben aangetoond dat het apparaatje waarmee je via ABN Amro kunt internet-bankieren... (meer)

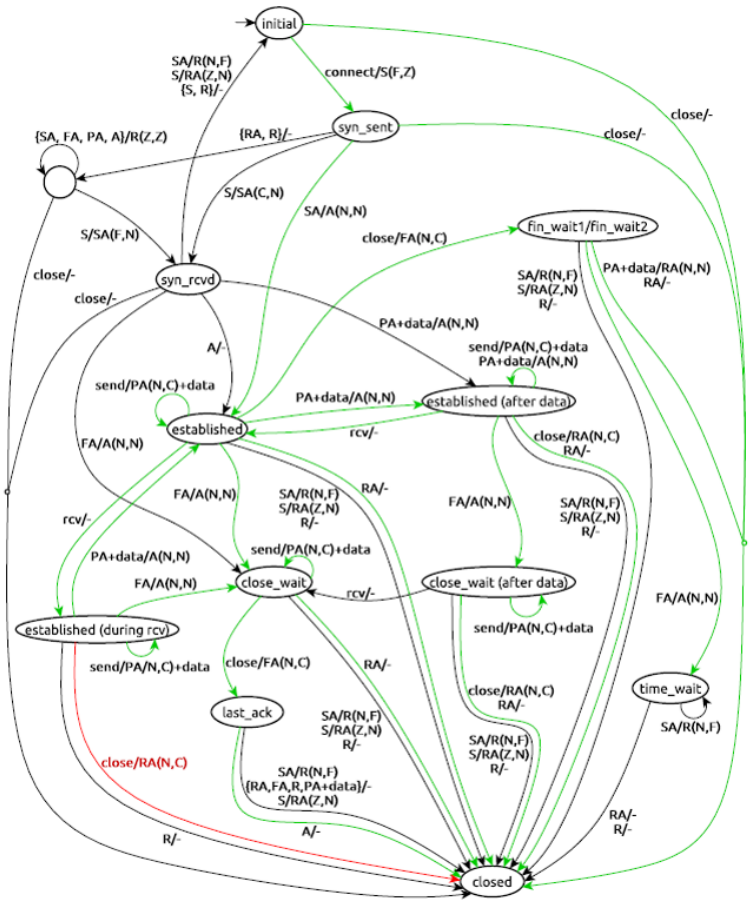
00:00 00:00

# State machines for old and new E.dentifier2



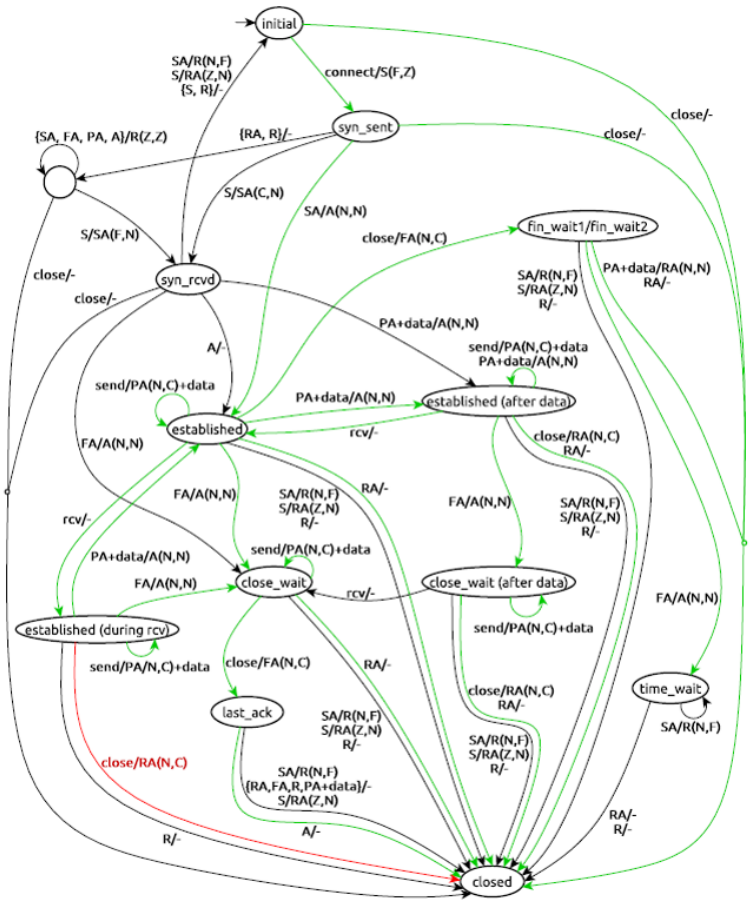
# Bugs in protocol implementations




Standard violations found in implementations of major protocols:

- TLS (Usenix Security'15)
- TCP (CAV'16)
- SSH (Spin'17).

# Bugs in protocol implementations

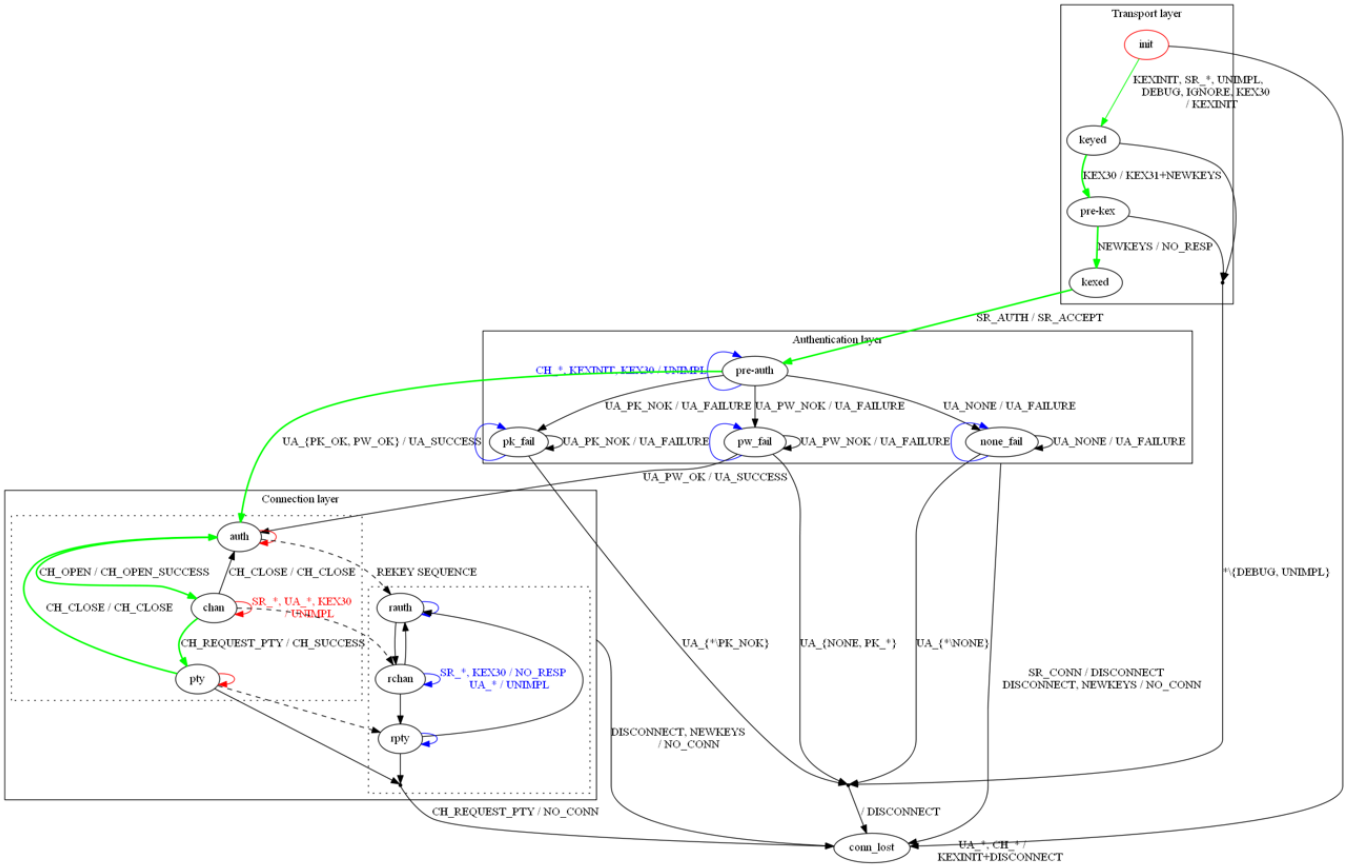



Standard violations found in implementations of major protocols:

- TLS (Usenix Security'15)
- TCP (CAV'16)
- SSH (Spin'17).

These findings led to bug fixes in implementations.

# Learned model for SSH implementation



# SSH model checking results

| | Property | Key word | OpenSSH | Bitvise | DropBear |
|---|---|---|---|---|---|
| Security | Trans. | | ✓ | ✓ | ✓ |
| | Auth. | | ✓ | ✓ | ✓ |
| Rekey | Pre-auth. | | X | ✓ | ✓ |
| | Auth. | | ✓ | X | ✓ |
| Funct. | Prop. 6 | MUST | ✓ | ✓ | ✓ |
| | Prop. 7 | MUST | ✓ | ✓ | ✓ |
| | Prop. 8 | MUST | X* | X | ✓ |
| | Prop. 9 | MUST | ✓ | ✓ | ✓ |
| | Prop. 10 | MUST | ✓ | ✓ | ✓ |
| | Prop. 11 | SHOULD | X* | X* | ✓ |
| | Prop. 12 | MUST | ✓ | ✓ | X |

# Other case studies

- Session Initiation Protocol (SIP)
- Message Queuing Telemetry Transport (MQTT) protocol
- Quick UDP Internet Connections (QUIC) protocol
- WiFi
- IEC 60870-5-104 protocol
- ...

# Lorentz Workshop


Participants from automata learning, model-based testing, cryptography, and security protocol implementation.

Working groups on e.g.,

- WiFi
- side channels in TLS
- LTE

# Automata wiki

Automata Wiki | ICIS eGroupWar | The New York T | Moshe Vardi (@ | 'Een lagere BSA | NOS Nieuws | ICGI 2018

automata.cs.ru.nl

Most Visited | ICIS eGroupWare - Lo...

Radboud University Nijmegen

**Automata Wiki**

select Benchmark | select tag | Search

Home

print · source · login

- Home
- Please contribute!

**Benchmarks**

- Overview
- Benchmark structure
- Modelling frameworks
- Evaluation criteria
- Downloads

**Software**

- Scripts

**People**

- Contributors

**Further reading and links**

- Learning/testing tools
- Other repositories

## *Welcome to the Automata Wiki!*

Active automata learning is emerging as a highly effective technique for obtaining models of protocol implementations and other reactive systems. Many algorithms have been proposed in the literature. Often variations of these algorithms exist for different classes of models, e.g. DFAs, Moore machines, Mealy machines, interface automata, and various forms of register automata. Algorithms for generation of conformance test suites often play a crucial role within active automata learning, as an oracle to determine whether a learned model is correct or not, and also here we see a wide variety of algorithms that have been proposed for different model classes.

Although there has been some excellent experimental work on evaluating algorithms for learning and conformance testing, the number of realistic models used for benchmarking is rather limited, and different papers use different industrial cases. Often the benchmarks used are either small/academic, which do not properly evaluate efficiency, or randomly generated, and it is clear from the experiments that performance of algorithms on randomly generated models is often radically different from performance on models of real systems that occur in practice.

A mature field is characterized by the presence of a rich set of shared benchmarks that can be used to compare different approaches. We have therefore set up this wiki with a publicly available set of benchmarks of state machines that model real protocols and embedded systems. These benchmarks will allow researchers to compare the performance of learning and testing algorithms.

We invite all our colleagues to contribute and send us (links to) other benchmarks they know of, for inclusion in the wiki.

# Automata wiki

Main/Overview | ICIS eGroupWar | The New York T | Moshe Vardi (@ | 'Een lagere BSA | NOS Nieuws | ICGI 2018

automata.cs.ru.nl/Overview

Most Visited | ICIS eGroupWare - Lo...

Please contribute!

**Benchmarks**

- Overview
- Benchmark structure
- Modelling frameworks
- Evaluation criteria
- Downloads

**Software**

- Scripts

**People**

- Contributors

**Further reading and links**

- Learning/testing tools
- Other repositories
- Bibliography

## Overview benchmarks

The benchmark models are stored per benchmark and grouped within a benchmark per model type. Below we first list the benchmarks per model type and finally we list all benchmarks combined. For more details about how the benchmark models are stored within this wiki we refer to the GlobalDataStructure page in the Help section.

The wiki also support tagging of benchmarks, eg. the page Tag/Protocol lists all benchmarks tagged as protocol. See the page Tag for all defined tags.

In the Table page we list the statistics, such as number of states, for the register and mealy benchmarks.

### Per model type

#### DFA benchmarks

- BenchmarkRandomDFAs
- BenchmarkToyDFA

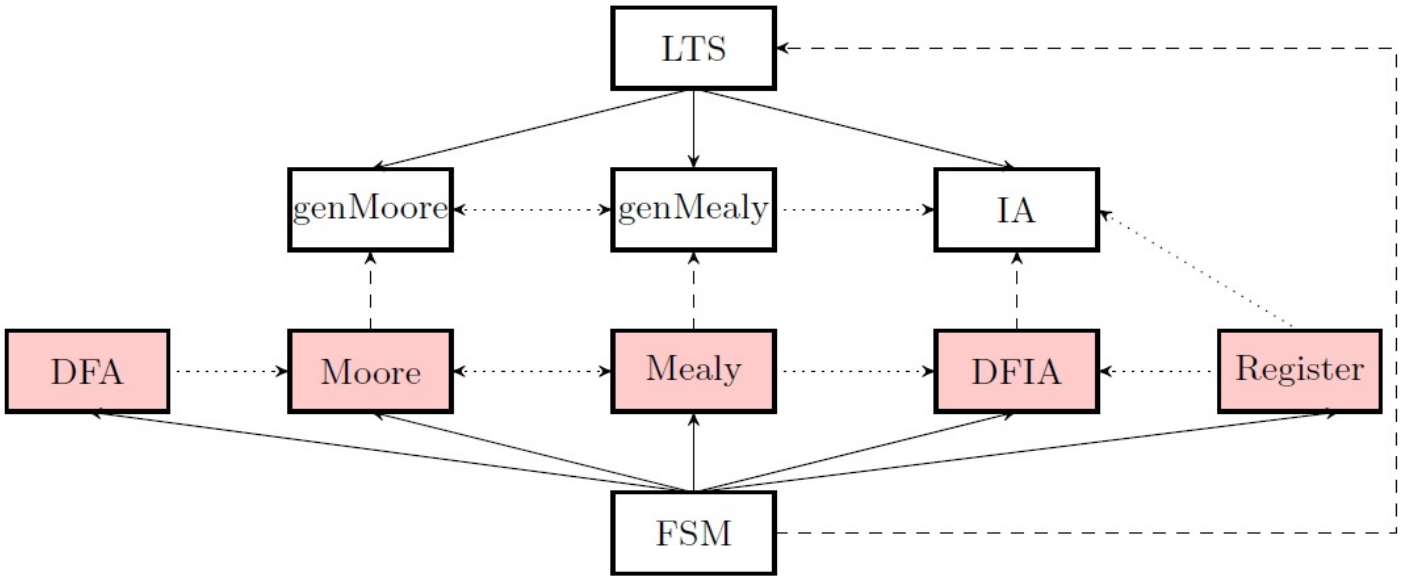
#### Moore machine benchmarks

- BenchmarkRandomMoore
- BenchmarkToyMoore

#### Mealy machine benchmarks

- BenchmarkBankcard
- BenchmarkCircuits
- BenchmarkCoffeeMachine
- BenchmarkEdentifier2
- BenchmarkESMcontroller
- BenchmarkFromRhapsodyToDezyne
- BenchmarkIEC60870-5-104

# Automata wiki: supported automata frameworks

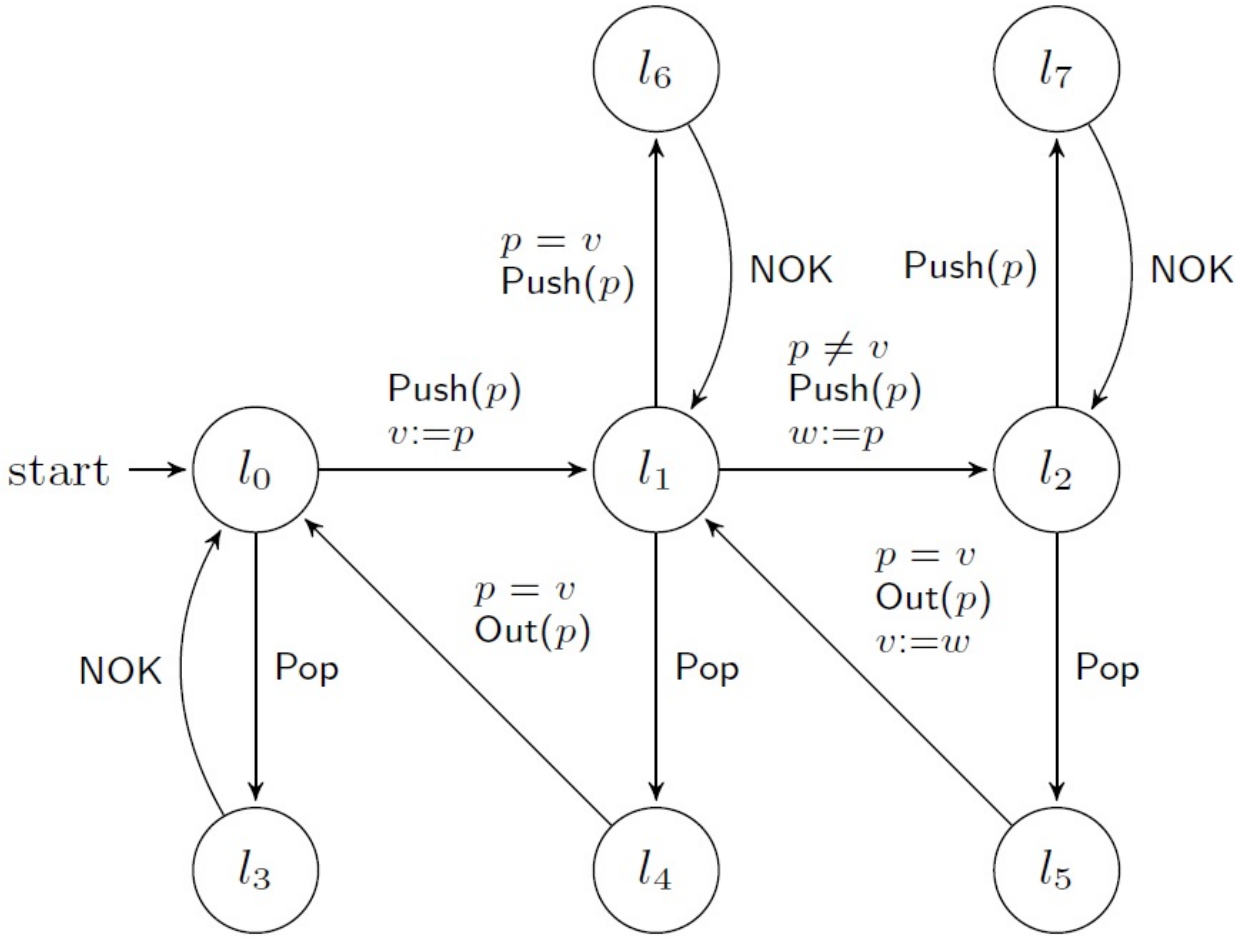


# Automata wiki: most benchmarks are not that big

| | | | |
|---|---|---|---|
| BenchmarkTLS/GnuTLS_3.3.12_client_regular.dot | 7 | 8/10 | 56 |
| BenchmarkTLS/GnuTLS_3.3.8_server_regular.dot | 12 | 8/10 | 96 |
| BenchmarkTLS/GnuTLS_3.3.12_server_full.dot | 9 | 12/12 | 108 |
| BenchmarkTLS/RSA_BSAFE_C_4.0.4_server_regular.dot | 9 | 8/11 | 72 |
| BenchmarkTCP/TCP_Windows8_Server.dot | 38 | 13/10 | 494 |
| BenchmarkTCP/TCP_FreeBSD_Server.dot | 55 | 13/11 | 715 |
| BenchmarkTCP/TCP_Linux_Server.dot | 57 | 12/9 | 684 |
| BenchmarkTCP/TCP_Linux_Client.dot | 15 | 10/11 | 150 |
| BenchmarkTCP/TCP_Windows8_Client.dot | 13 | 10/12 | 130 |
| BenchmarkTCP/TCP_FreeBSD_Client.dot | 12 | 10/12 | 120 |
| BenchmarkX-ray-system-PCS/learnresult5.dot | 9 | 12/9 | 66 |
| BenchmarkX-ray-system-PCS/learnresult4.dot | 7 | 9/7 | 37 |
| BenchmarkX-ray-system-PCS/learnresult6.dot | 9 | 12/9 | 66 |
| BenchmarkX-ray-system-PCS/learnresult3.dot | 7 | 9/7 | 37 |
| BenchmarkX-ray-system-PCS/learnresult2.dot | 3 | 9/4 | 9 |
| BenchmarkX-ray-system-PCS/learnresult1.dot | 8 | 9/7 | 44 |
| BenchmarkESMcontroller/run3_hyp.101.obf.dot | 3410 | 78/151 | 265980 |
| BenchmarkESMcontroller/run1_hyp.135.obf.dot | 3410 | 78/151 | 265980 |

# Register automata

Actions may carry data parameters that may be stored in registers:



# Data types

Register automata may be parametrized by a (relational) structure: a pair $\langle \mathcal{D}, \mathcal{R} \rangle$ where $\mathcal{D}$ is an unbounded domain of data values, and $\mathcal{R}$ is a collection of relations on $\mathcal{D}$.

Examples of simple structures include:

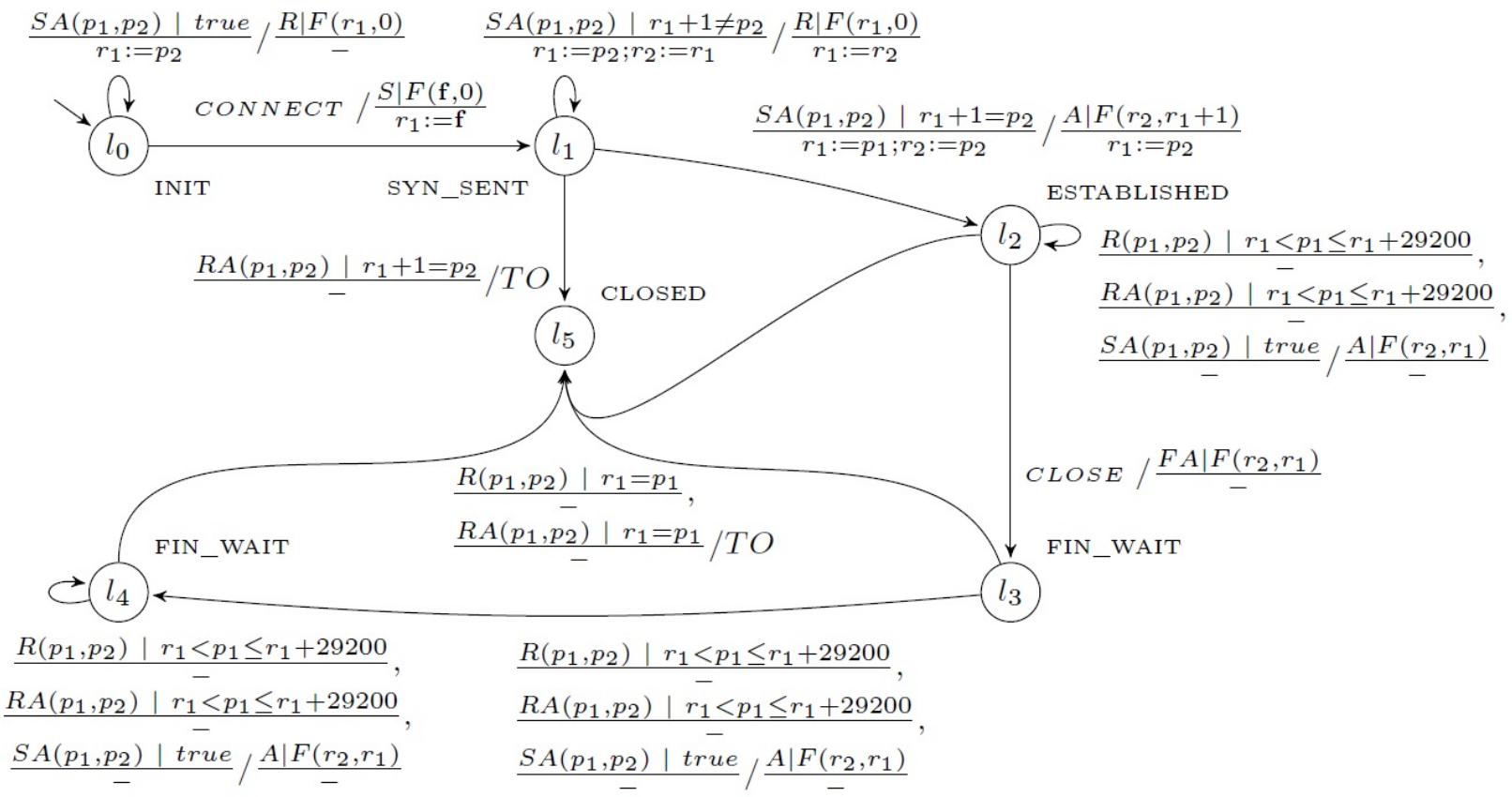
- $\langle \mathbb{N}, \{=\} \rangle$, the natural numbers with equality;
- $\langle \mathbb{R}, \{<\} \rangle$, the real numbers with inequality: this structure also allows one to express equality between elements.

Transition guards are conjunctions of negated and unnegated relations from $\mathcal{R}$.

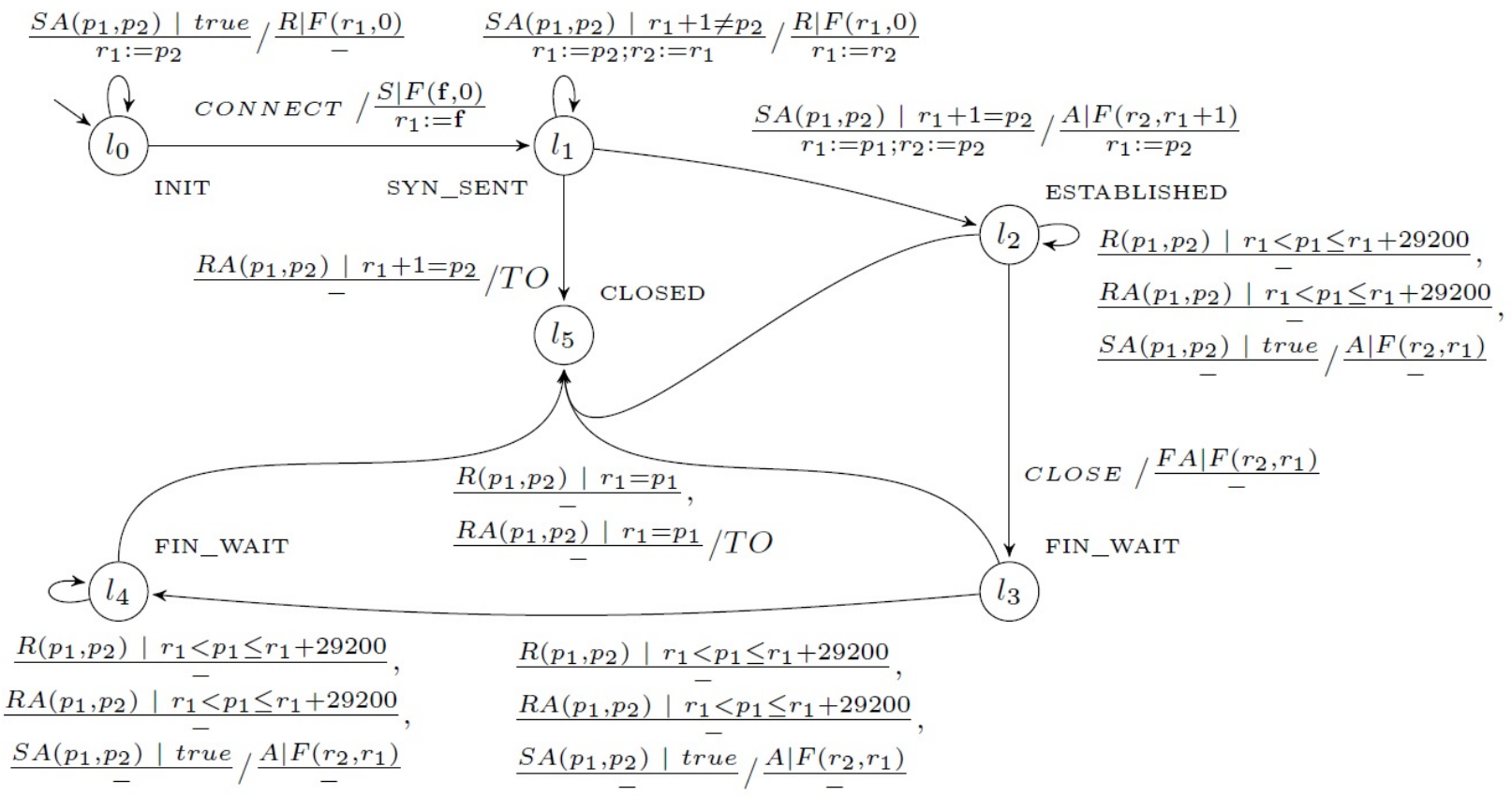
# Learning tools for register automata

- Tomte, Radboud University, can only handle $\langle \mathbb{N}, \{=\} \rangle$
- LearnLib, TU Dortmund, can only handle $\langle \mathbb{N}, \{=\} \rangle$
- RALib, Uppsala/Dortmund, can handle some richer structures

# TCP protocol case study (FMICS-AVoCS'17)



# TCP protocol case study (FMICS-AVoCS'17)




These findings led to bug fix in Linux TCP implementation.

# Limits of black-box learning?

- Model learning is an highly effective bug finding technique

# Limits of black-box learning?

- Model learning is an highly effective bug finding technique
- ... but it has some serious scalability problems

# Limits of black-box learning?

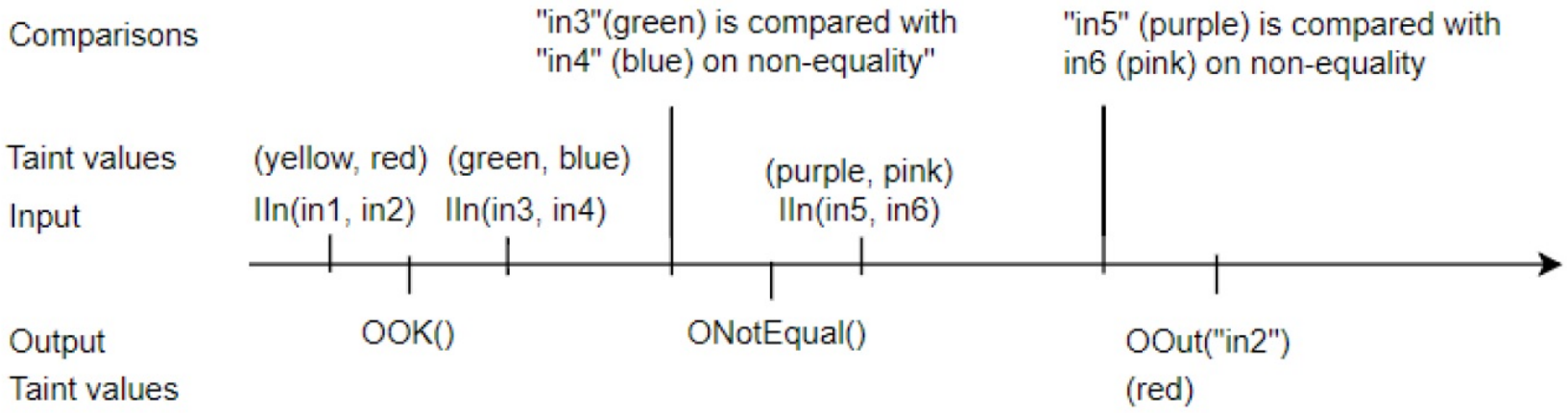
- Model learning is an highly effective bug finding technique
- ... but it has some serious scalability problems
- Can we use white-box information while preserving the extensionality of black-box models?

# Idea: Use Taint Analysis

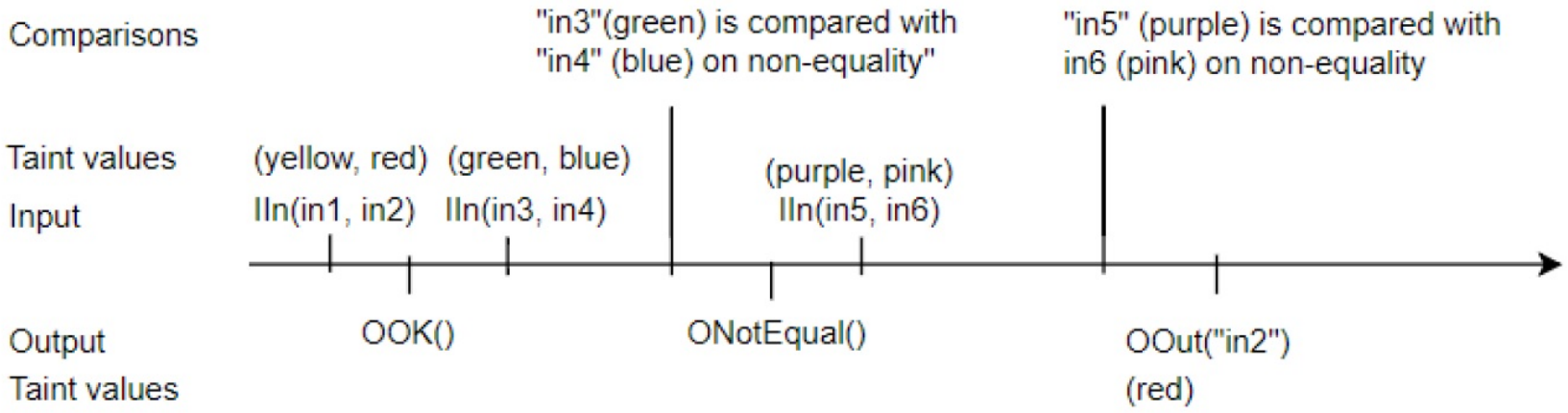
# Taint Analysis

- White-box technique for code analysis
- Instruments code to track input values
- Many tools focus on specific vulnerabilities, e.g. buffer overflows and sql injections
- Usually implemented using Dynamic Binary Analysis, e.g. Valgrind
- We use Python library from Pygmalion tool from Andreas Zeller et al.

# What Does Pygmalion Tool Do For Us?



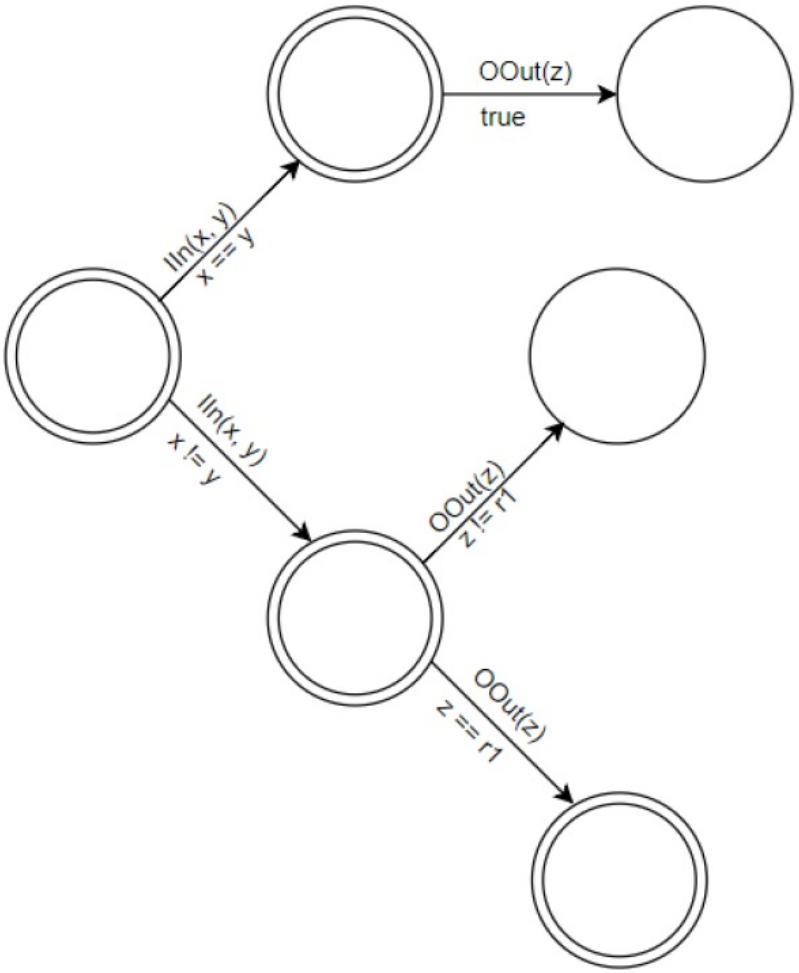
# What Does Pygmalion Tool Do For Us?




Potential of exponential gains during learning!

# Architecture RAlib Tool for Learning Register Automata

# Tree Oracle



# Ongoing Work

Replace tree oracle in RAlib by a version that uses taint analysis.

# Ongoing Work

Replace tree oracle in RAlib by a version that uses taint analysis.

First prototype finished (for integers with equality)!!!

# Conclusion

Active automata learning is emerging as a highly effective bug-finding technique, and slowly becoming a standard tool in the toolbox of the software engineer.

But much further research is needed!

# Future Work

1. Explore combinations of black-box and white-box learning
2. Further improvement of learning and testing algorithms for FSM models
3. Develop algorithms for models with time and probabilities
4. Refactoring of legacy software is potentially excellent application domain
5. Please contribute benchmarks to our repository!