Automata Learning and Galois Connections

Frits Vaandrager

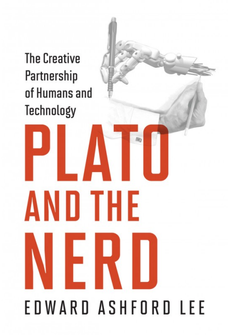
Radboud University Nijmegen

ICALP'19, Patras, Greece, July 2019

Outline

1. Introduction
2. Learning Unions of k-Testable Languages
3. Active Learning of DFAs and Mealy Machines
4. Active Learning Using Mappers
5. Conclusions and Future Work

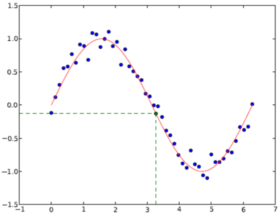
Plato and the Nerd

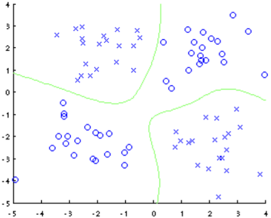


- Model: any description of a system that is not the thing-in-itself.
- Engineering perspective:
  "Can we build a system whose behavior matches that of a given model?"
- Science perspective:
  "Can we build a model whose behavior matches that of a given system?"
- This talk:
  By properly combining both perspectives we can build better systems.

Machine Learning in General

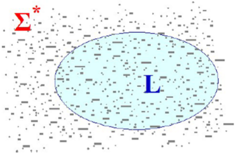
- Given a sample $M = \{(x, y) \mid x \in X, y \in Y\}$
- Find $f : X \to Y$ such that $f(x) = y, \forall(x, y) \in M$
- Predict $f(x)$ for all $x \in X$





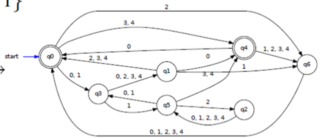
Learning Regular Languages

Let Σ be an alphabet and let $L \subseteq \Sigma^*$ be a regular language (*the target language*)



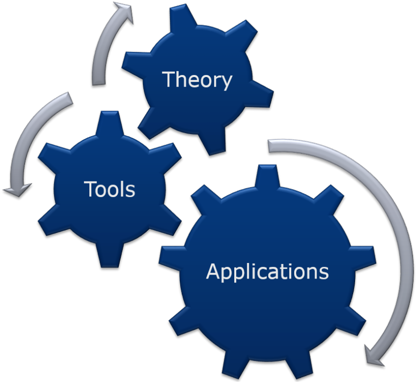
$\chi_L : \Sigma^* \to \{0, 1\}$

?



- Edward F Moore, *Gedanken-experiments on sequential machines*, 1956
- E. Mark Gold, *System Identification via State Characterization*, 1972
- Dana Angluin, *Learning regular sets from queries and counterexamples*, 1987

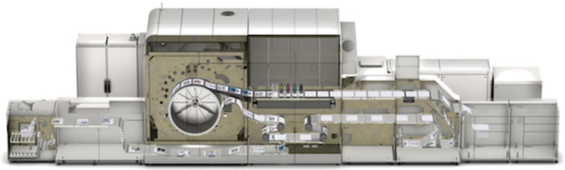
Our Research Method



Galois Connections

- Particular correspondence between two partially ordered sets
- Many applications in mathematics
- Adjoint functors in category theory
- Describe many forms of abstraction in theory of abstract interpretation of programming languages

A Problem from Océ



Identify patterns in logs of printer behavior:[1]

$$S = \{abab, ababab, abababab, abba, abbba, abbbba, aaaa, aaaaaa, aaaaaaaa\}$$

[1] Based on work of Linard, Vaandrager & De La Higuera (LATA'19).

Tackling the Océ Problem

- To solve Océ problem we need to learn a union of regular languages from positive examples only
- But it is impossible to learn regular languages in the limit from positive examples! (Gold, 1967)
- Window languages (a.k.a. k-testable languages) (McNaughton & Papert, 1971) are learnable in the limit from positive examples.
- Can we learn unions of window languages? And if so, does this provide the patterns Océ is looking for?

Window Languages

Definition (k-test vector)

Let $k > 0$. A k-test vector is a tuple $Z = \langle I, F, T, C \rangle$ where:

- $I \subseteq \Sigma^{k-1}$ is a set of allowed prefixes
- $F \subseteq \Sigma^{k-1}$ is a set of allowed suffixes
- $T \subseteq \Sigma^{k}$ is a set of allowed segments
- $C \subseteq \Sigma^{<k}$ is a set of allowed short strings

We write $\mathcal{T}_k$ to denote the set of k-test vectors.

Window Languages

Window of size 3

Words matching *k-test vector* $Z = \langle I, F, T, C \rangle$ where:

- prefixes $I = \{ab\}$
- suffixes $F = \{ab, ba\}$
- segments $T = \{aba, abb, bab, bba\}$
- short strings $C = \{ab\}$

Window Languages

Window of size 3

Words matching *k-test vector* $Z = \langle I, F, T, C \rangle$ where::

- prefixes $I = \{ab\}$
- suffixes $F = \{ab, ba\}$
- segments $T = \{aba, abb, bab, bba\}$
- short strings $C = \{ab\}$

ababababab

Window Languages

Window of size 3

Words matching *k-test vector* $Z = \langle I, F, T, C \rangle$ where::

- prefixes $I = \{ab\}$
- suffixes $F = \{ab, ba\}$
- segments $T = \{aba, abb, bab, bba\}$
- short strings $C = \{ab\}$

$\boxed{\text{ab}}\,$abababab

ab $\in I$

Window Languages

Window of size 3

Words matching *k-test vector* $Z = \langle I, F, T, C \rangle$ where::

- prefixes $I = \{ab\}$
- suffixes $F = \{ab, ba\}$
- segments $T = \{aba, abb, bab, bba\}$
- short strings $C = \{ab\}$

ababab $\boxed{\text{ab}}$

ab $\in F$

Window Languages

Window of size 3

Words matching *k-test vector* $Z = \langle I, F, T, C \rangle$ where::

- prefixes $I = \{ab\}$
- suffixes $F = \{ab, ba\}$
- segments $T = \{aba, abb, bab, bba\}$
- short strings $C = \{ab\}$

$\boxed{\text{aba}}$bababab

aba $\in T$

Window Languages

Window of size 3

Words matching *k-test vector* $Z = \langle I, F, T, C \rangle$ where::

- prefixes $I = \{ab\}$
- suffixes $F = \{ab, ba\}$
- segments $T = \{aba, abb, bab, bba\}$
- short strings $C = \{ab\}$

a $\boxed{\text{bab}}$ ababab

bab $\in T$

Window Languages

Window of size 3

Words matching *k-test vector* $Z = \langle I, F, T, C \rangle$ where::

- prefixes $I = \{ab\}$
- suffixes $F = \{ab, ba\}$
- segments $T = \{aba, abb, bab, bba\}$
- short strings $C = \{ab\}$

ab $\boxed{\text{aba}}$ babab

aba $\in T$

Window Languages

Window of size 3

Words matching *k-test vector* $Z = \langle I, F, T, C \rangle$ where::

- prefixes $I = \{ab\}$
- suffixes $F = \{ab, ba\}$
- segments $T = \{aba, abb, bab, bba\}$
- short strings $C = \{ab\}$

aba $\boxed{\text{bab}}$ abab

bab $\in T$

Window Languages

Window of size 3

Words matching *k-test vector* $Z = \langle I, F, T, C \rangle$ where::

- prefixes $I = \{ab\}$
- suffixes $F = \{ab, ba\}$
- segments $T = \{aba, abb, bab, bba\}$
- short strings $C = \{ab\}$

abab $\boxed{\text{aba}}$ bab

aba $\in T$

Window Languages

Window of size 3

Words matching *k-test vector* $Z = \langle I, F, T, C \rangle$ where::

- prefixes $I = \{ab\}$
- suffixes $F = \{ab, ba\}$
- segments $T = \{aba, abb, bab, bba\}$
- short strings $C = \{ab\}$

ababa $\boxed{\text{bab}}$ ab

bab $\in T$

Window Languages

Window of size 3

Words matching *k-test vector* $Z = \langle I, F, T, C \rangle$ where::

- prefixes $I = \{ab\}$
- suffixes $F = \{ab, ba\}$
- segments $T = \{aba, abb, bab, bba\}$
- short strings $C = \{ab\}$

ababab $\boxed{\text{aba}}$ b

aba $\in T$

Window Languages

Window of size 3

Words matching *k-test vector* $Z = \langle I, F, T, C\rangle$ where::

- prefixes $I = \{ab\}$
- suffixes $F = \{ab, ba\}$
- segments $T = \{aba, abb, bab, bba\}$
- short strings $C = \{ab\}$

ababab$\boxed{\text{a bab}}$

bab $\in T$

Window Languages

Window of size 3

Words matching *k-test vector* $Z = \langle I, F, T, C \rangle$ where::

- prefixes $I = \{ab\}$
- suffixes $F = \{ab, ba\}$
- segments $T = \{aba, abb, bab, bba\}$
- short strings $C = \{ab\}$

abaaba

Window Languages

Window of size 3

Words matching *k-test vector* $Z = \langle I, F, T, C \rangle$ where::

- prefixes $I = \{ab\}$
- suffixes $F = \{ab, ba\}$
- segments $T = \{aba, abb, bab, bba\}$
- short strings $C = \{ab\}$

$\boxed{\text{ab}}\text{aaba}$

ab $\in I$

Window Languages

Window of size 3

Words matching *k-test vector* $Z = \langle I, F, T, C \rangle$ where::

- prefixes $I = \{ab\}$
- suffixes $F = \{ab, ba\}$
- segments $T = \{aba, abb, bab, bba\}$
- short strings $C = \{ab\}$

abaa $\boxed{\text{ba}}$

ba $\in F$

Window Languages

Window of size 3

Words matching *k-test vector* $Z = \langle I, F, T, C \rangle$ where::

- prefixes $I = \{ab\}$
- suffixes $F = \{ab, ba\}$
- segments $T = \{aba, abb, bab, bba\}$
- short strings $C = \{ab\}$

$\boxed{\text{aba}}\text{aba}$

aba $\in T$

Window Languages

Window of size 3

Words matching *k-test vector* $Z = \langle I, F, T, C \rangle$ where::

- prefixes $I = \{ab\}$
- suffixes $F = \{ab, ba\}$
- segments $T = \{aba, abb, bab, bba\}$
- short strings $C = \{ab\}$

a $\boxed{\text{baa}}$ ba

baa $\notin T$

k-Testable Languages

Definition (from k-test vectors to languages)

Let $Z = \langle I, F, T, C \rangle$ be a k-test vector, for some $k > 0$. Then

$$\gamma_k(Z) \quad = \quad C \cup ((I\Sigma^* \cap \Sigma^* F) \setminus (\Sigma^*(\Sigma^k \setminus T)\Sigma^*)).$$

A language $L \subseteq \Sigma^*$ is k-testable in the strict sense (k-TSS) if there exists a k-test vector Z such that $L = \gamma_k(Z)$.

Note that k-TSS languages are regular.

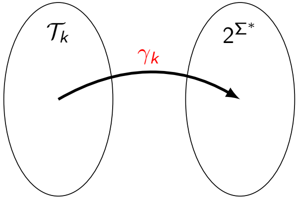
k-Testable Languages

Definition (from k-test vectors to languages)

Let $Z = \langle I, F, T, C \rangle$ be a k-test vector, for some $k > 0$. Then

$$\gamma_k(Z) \quad = \quad C \cup ((I\Sigma^* \cap \Sigma^* F) \setminus (\Sigma^*(\Sigma^k \setminus T)\Sigma^*)).$$

A language $L \subseteq \Sigma^*$ is *k-testable in the strict sense (k-TSS)* if there exists a k-test vector Z such that $L = \gamma_k(Z)$.



k-Testable Languages

Definition (from Languages to k-test vectors)

Let $L \subseteq \Sigma^*$ be a language and $k > 0$. Then $\alpha_k(L)$ is the k-test vector $\langle I_k(L), F_k(L), T_k(L), C_k(L) \rangle$ where

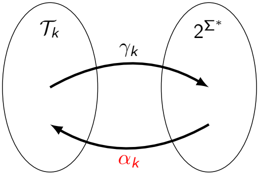
- $I_k(L) = \{u \in \Sigma^{k-1} \mid \exists v \in \Sigma^* : uv \in L\}$,
- $F_k(L) = \{w \in \Sigma^{k-1} \mid \exists v \in \Sigma^* : vw \in L\}$,
- $T_k(L) = \{v \in \Sigma^k \mid \exists u, w \in \Sigma^* : uvw \in L\}$, and
- $C_k(L) = (L \cap \Sigma^{<k-1}) \cup (I_k(L) \cap F_k(L))$.

k-Testable Languages

Definition (from Languages to k-test vectors)

Let $L \subseteq \Sigma^*$ be a language and $k > 0$. Then $\alpha_k(L)$ is the k-test vector $\langle I_k(L), F_k(L), T_k(L), C_k(L)\rangle$ where

- $I_k(L) = \{u \in \Sigma^{k-1} \mid \exists v \in \Sigma^* : uv \in L\}$,
- $F_k(L) = \{w \in \Sigma^{k-1} \mid \exists v \in \Sigma^* : vw \in L\}$,
- $T_k(L) = \{v \in \Sigma^k \mid \exists u, w \in \Sigma^* : uvw \in L\}$, and
- $C_k(L) = (L \cap \Sigma^{<k-1}) \cup (I_k(L) \cap F_k(L))$.



k-Test Vector Inclusion

Definition

Let $k > 0$. The relation $\sqsubseteq$ on $\mathcal{T}_k$ is given by

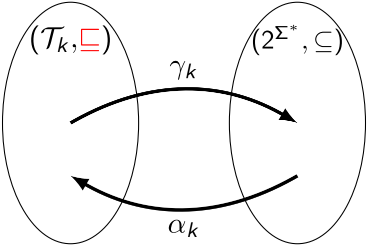
$$\langle I, F, T, C \rangle \sqsubseteq \langle I', F', T', C' \rangle \quad \Leftrightarrow \quad I \subseteq I' \wedge F \subseteq F' \wedge T \subseteq T' \wedge C \subseteq C'.$$

k-Test Vector Inclusion

Definition

Let $k > 0$. The relation $\sqsubseteq$ on $\mathcal{T}_k$ is given by

$$\langle I, F, T, C \rangle \sqsubseteq \langle I', F', T', C' \rangle \;\Leftrightarrow\; I \subseteq I' \wedge F \subseteq F' \wedge T \subseteq T' \wedge C \subseteq C'.$$



Order Preservation

Lemma

For $k > 0$ and for all languages L, L',

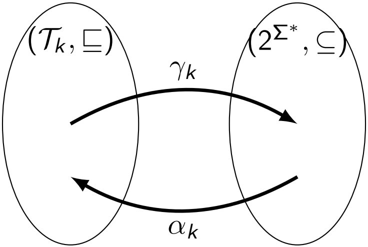
$$L \subseteq L' \quad \Rightarrow \quad \alpha_k(L) \sqsubseteq \alpha_k(L').$$

Lemma

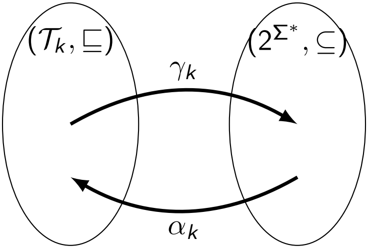
For all $k > 0$ and for all k-test vectors Z and Z',

$$Z \sqsubseteq Z' \quad \Rightarrow \quad \gamma_k(Z) \subseteq \gamma_k(Z').$$

Galois Connection



Galois Connection



Theorem (Galois Connection)

Let $k > 0$, $L \subseteq \Sigma^*$ a language, and Z a k-test vector. Then

$$\alpha_k(L) \sqsubseteq Z \quad \Leftrightarrow \quad L \subseteq \gamma_k(Z).$$

Galois Connection

Corollary

For all $k > 0$, $\gamma_k \circ \alpha_k$ and $\alpha_k \circ \gamma_k$ are monotone and idempotent.

Previously established as Theorem 3.2 in Garcia and Vidal (1990) and as Lemma 3.3 in in Yokomori and Kobayashi (1998).

Galois Connection

Corollary

For all $k > 0$, $L \subseteq \Sigma^*$ *and* $Z \in \mathcal{T}_k$,

$$\begin{aligned} \alpha_k \circ \gamma_k(Z) &\sqsubseteq Z \\ L &\subseteq \gamma_k \circ \alpha_k(L) \end{aligned}$$

Previously established as Lemma 3.1 in Garcia and Vidal (1990) and as Lemma 3.1 in Yokomori and Kobayashi (1998).

Galois Connection

Corollary

For all $k > 0$, $L \subseteq \Sigma^*$, and $Z \in \mathcal{T}_k$,

$$L \subseteq \gamma_k(Z) \quad \Rightarrow \quad \gamma_k \circ \alpha_k(L) \subseteq \gamma_k(Z).$$

Previously established as Theorem 3.1 in Garcia and Vidal (1990).

Galois Connection

Corollary

For all $k > 0$ and $Z \in \mathcal{T}_k$, $\gamma_k \circ \alpha_k \circ \gamma_k(Z) = \gamma_k(Z)$. Moreover, for any $Z' \in \mathcal{T}_k$,

$$\gamma_k(Z) = \gamma_k(Z') \quad \Rightarrow \quad \alpha_k \circ \gamma_k(Z) \sqsubseteq Z'.$$

Previously established as Lemma 1 in Yokomori and Kobayashi (1998).

Learning k-Testable Languages

Theorem (Garcia & Vidal (1990))

Any k-testable language can be identified in the limit from positive examples.

Union and Symmetric Difference

Definition

The union and symmetric difference of two k-test vectors $Z = \langle I, F, T, C \rangle$ and $Z' = \langle I', F', T', C' \rangle$ are given by:

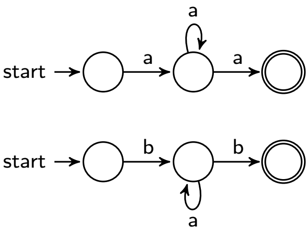
$$\begin{aligned} Z \sqcup Z' &= \langle I \cup I', F \cup F', T \cup T', C \cup C' \cup (I \cap F') \cup (I' \cap F) \rangle \\ Z \triangle Z' &= \langle I \triangle I', F \triangle F', T \triangle T', C \triangle C' \triangle (I' \cap F) \triangle (I \cap F') \rangle \end{aligned}$$

Window Languages Not Closed Under Union

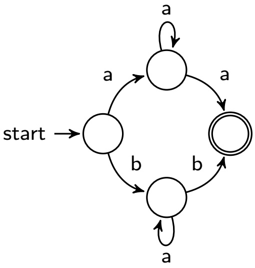
$$
\begin{aligned}
Z &= \langle\{aa\},\{aa\},\{aaa\},\{aa\}\rangle \\
Z' &= \langle\{ba, bb\},\{ab, bb\},\{baa, bab, aaa, aab\},\{bb\}\rangle
\end{aligned}
$$

Window Languages Not Closed Under Union

$$
\begin{aligned}
Z &= \langle \{aa\}, \{aa\}, \{aaa\}, \{aa\} \rangle \\
Z' &= \langle \{ba, bb\}, \{ab, bb\}, \{baa, bab, aaa, aab\}, \{bb\} \rangle
\end{aligned}
$$



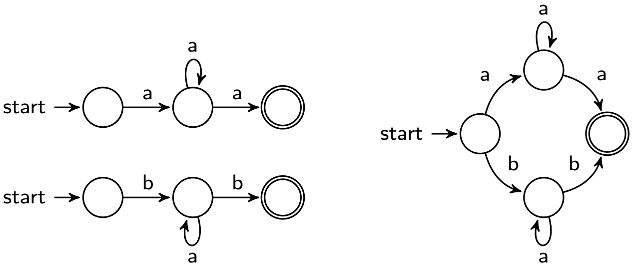
(a) $\gamma_3(Z)$ and $\gamma_3(Z')$.



(b) $\gamma_3(Z) \cup \gamma_3(Z')$.

Window Languages Not Closed Under Union

$$
\begin{aligned}
Z &= \langle \{aa\}, \{aa\}, \{aaa\}, \{aa\} \rangle \\
Z' &= \langle \{ba, bb\}, \{ab, bb\}, \{baa, bab, aaa, aab\}, \{bb\} \rangle
\end{aligned}
$$



(a) $\gamma_3(Z)$ and $\gamma_3(Z')$.

(b) $\gamma_3(Z) \cup \gamma_3(Z')$.

$aab \in \gamma_3(Z \sqcup Z')$ but $aab \notin \gamma_3(Z) \cup \gamma_3(Z')$.

Learning Unions of k-Testable Languages

Theorem (Identification of unions in the limit)

Any language that is a union of k-testable languages can be identified in the limit from positive examples.

Distance

Definition (Size)

The size of a k-test vector $Z = \langle I, F, T, C \rangle$ is defined by:

$$|Z| \quad = |I| + |F| + |T| + |C \cap \Sigma^{<k-1}|.$$

Definition (Distance)

We define the distance between a pair of k-test vectors as:

$$d(Z, Z') \quad = |Z \triangle Z'|$$

Lemma (Metric)

Distance function is a metric on the set of k-test vectors.

Hierarchical Clustering Algorithm

Given a set $\mathcal{S}$ of words:

Hierarchical Clustering Algorithm

Given a set $\mathcal{S}$ of words:

1. compute k-test vectors $s = \{\alpha_k(\{x\}) \mid x \in \mathcal{S}\}$

Hierarchical Clustering Algorithm

Given a set $\mathcal{S}$ of words:

1. compute k-test vectors $s = \{\alpha_k(\{x\}) \mid x \in \mathcal{S}\}$
2. compute distance matrix D of vectors in s

Hierarchical Clustering Algorithm

Given a set $\mathcal{S}$ of words:

1. compute k-test vectors $s = \{\alpha_k(\{x\}) \mid x \in \mathcal{S}\}$
2. compute distance matrix D of vectors in s
3. until no more merges are possible:

Hierarchical Clustering Algorithm

Given a set $\mathcal{S}$ of words:

1. compute k-test vectors $s = \{\alpha_k(\{x\}) \mid x \in \mathcal{S}\}$
2. compute distance matrix D of vectors in s
3. until no more merges are possible:
   1. find closest pair of vectors Z and Z' s.t. $\gamma_k(Z \sqcup Z') = \gamma_k(Z) \cup \gamma_k(Z')$

Hierarchical Clustering Algorithm

Given a set $\mathcal{S}$ of words:

1. compute k-test vectors $s = \{\alpha_k(\{x\}) \mid x \in \mathcal{S}\}$
2. compute distance matrix D of vectors in s
3. until no more merges are possible:
    1. find closest pair of vectors Z and Z' s.t. $\gamma_k(Z \sqcup Z') = \gamma_k(Z) \cup \gamma_k(Z')$
    2. replace Z and Z' by $Z \sqcup Z'$ in s

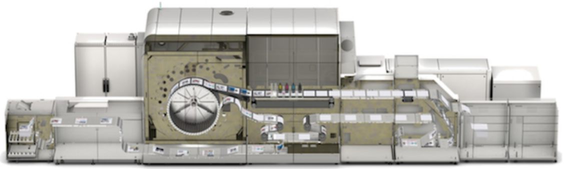
Hierarchical Clustering Algorithm

Given a set $\mathcal{S}$ of words:

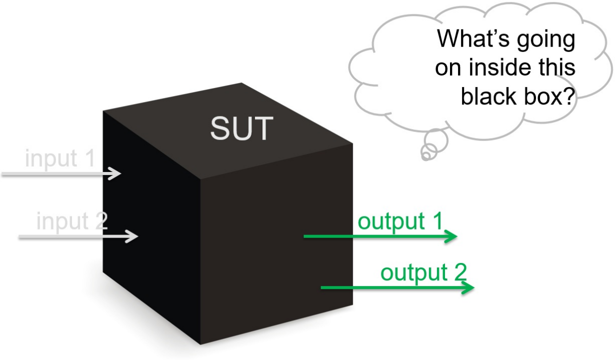
1. compute k-test vectors $s = \{\alpha_k(\{x\}) \mid x \in \mathcal{S}\}$
2. compute distance matrix D of vectors in s
3. until no more merges are possible:
   1. find closest pair of vectors Z and Z' s.t. $\gamma_k(Z \sqcup Z') = \gamma_k(Z) \cup \gamma_k(Z')$
   2. replace Z and Z' by $Z \sqcup Z'$ in s
   3. update distance between $Z \sqcup Z'$ and remaining vectors in s

Case Study Océ

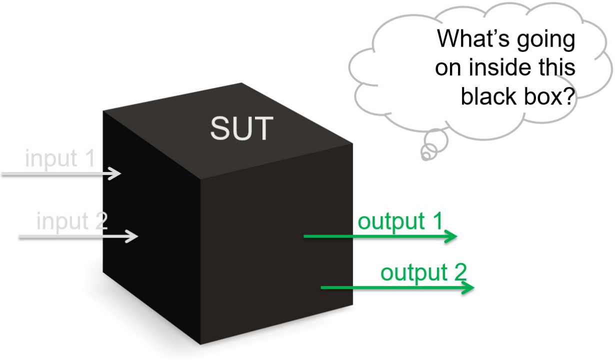
job	pattern	3-test vector	type of job
aaaaa	a^+	$Z = \langle\{aa\}, \{aa\}, \{aaa\}, \{a, aa\}\rangle$	homogeneous
aaaaaaaaaa			
aaaaa . . . aaa			
abababab	$(ab)^+$	$Z = \langle\{ab\}, \{ab\}, \{aba, bab\}, \{ab\}\rangle$	heterogeneous
ababababab			
abcabcabc	$(abc)^+$	$Z = \langle\{ab\}, \{bc\}, \{abc, bca, cab\}, \{\}\rangle$	
abcabcabcabcabc			
abcbcbcbca	$a(bc)^+a$	$Z = \langle\{ab\}, \{ca\}, \{abc, bcb, cbc, cba\}, \{\}\rangle$	booklet



A Common Problem for Software Engineers

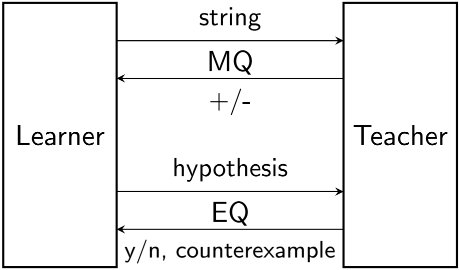


A Common Problem for Software Engineers



We assume SUT behaves deterministically and can be reset.

Minimally Adequate Teacher (Angluin)

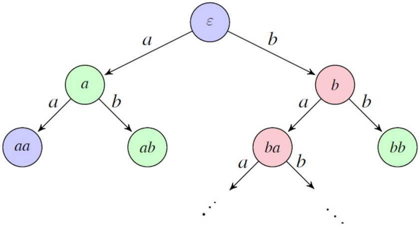


Learner asks membership queries and equivalence queries

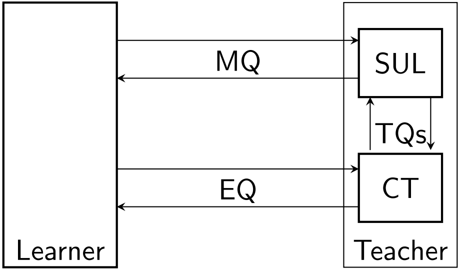
Angluin's L^* Algorithm

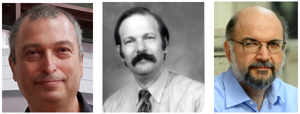
		E: ε	E: a
S	ε	$-$	$+$
S	a	$+$	$-$
S	b	$-$	$-$
S	ba	$-$	$-$
R	aa	$-$	$+$
R	ab	$+$	$-$
R	bb	$+$	$-$
R	baa	$-$	$-$
R	bab	$+$	$-$

- S *states* of the canonical automaton
- The words/paths correspond to a spanning tree
- R cross- and back-edges/transitions



Black Box Checking (Peled, Vardi & Yannakakis)

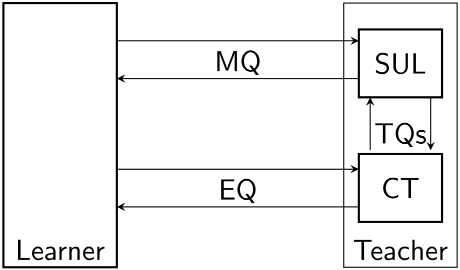


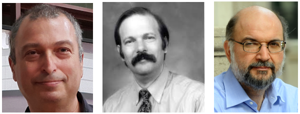


Learner: Formulate hypotheses

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
a framework for automata learning

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

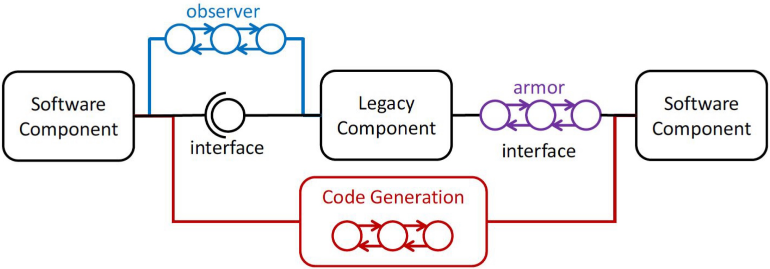
Prof. Dr. Bernhard Steffen

Implements MAT framework for DFAs and Mealy machines

Engine Status Manager in Océ Printer (ICFEM'15)

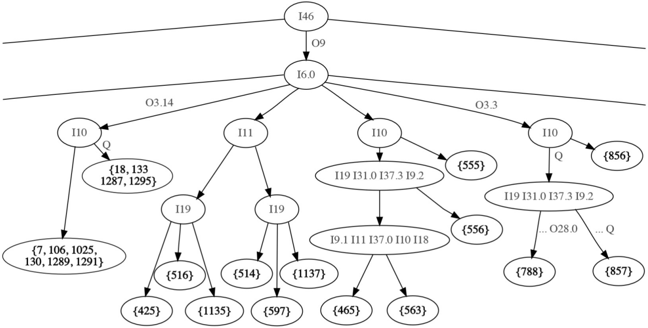
Can we learn interface models of realistic printer controllers?

Potential Applications Interface Models

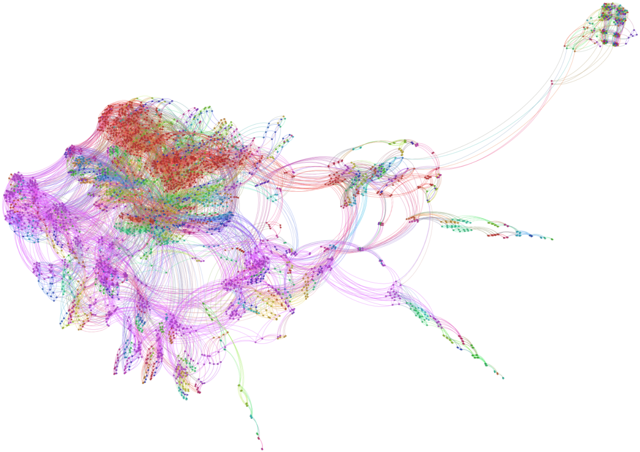


Conformance Testing Becomes Bottleneck!

No existing conformance testing methods (W, Wp, HSI, ADS, UIOv, P, H, SPY,..) was able to find counterexamples for some hypotheses models of the printer software. We had to develop a new hybrid ADS method, based on work of Lee & Yannakakis.



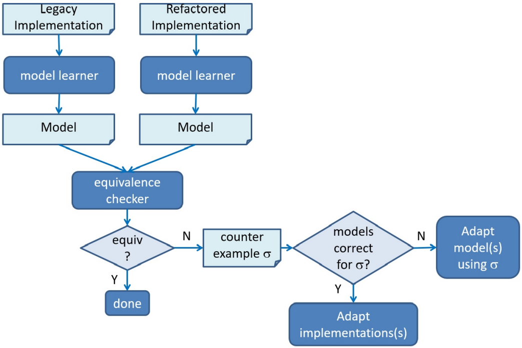
Mealy Machine for Engine Status Manager



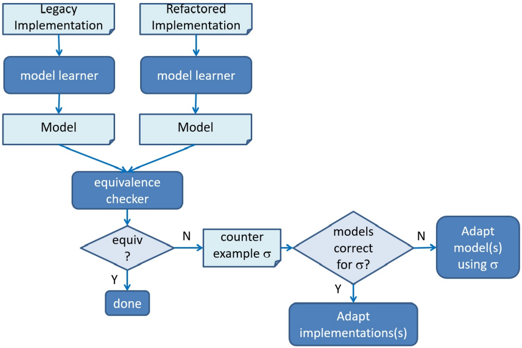
Power Control Service from Philips Healthcare (iFM'16)

Are legacy component and refactored implementation equivalent?

Refactoring Legacy Implementations



Refactoring Legacy Implementations



This approach allowed us to find several bugs in refactored implementations of power control service.

ASML Twinscan

ASML Challenge

Can active automata learning be used to support refactoring of legacy software at ASML?

ASML machines run on legacy software. Recent components have been designed using model-based techniques. Can we learn those?

Can we learn the hundreds of design and interface models used for high level control of the wafer flow during lot operation?

ASML Challenge

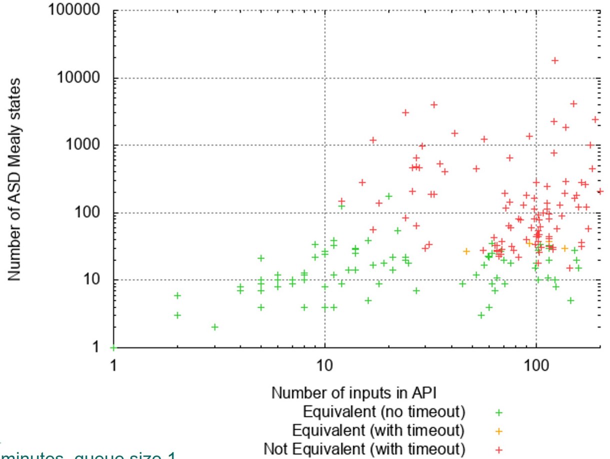
Can active automata learning be used to support refactoring of legacy software at ASML?

ASML machines run on legacy software. Recent components have been designed using model-based techniques. Can we learn those?

Can we learn the hundreds of design and interface models used for high level control of the wafer flow during lot operation?

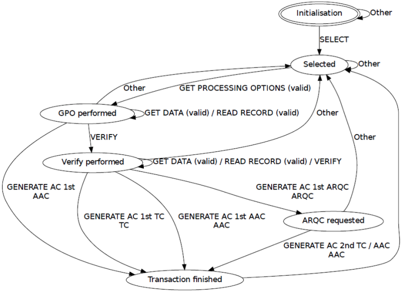
$\Rightarrow$ RERS @ TOOLympics'19

Results LearnLib on ASML Benchmarks



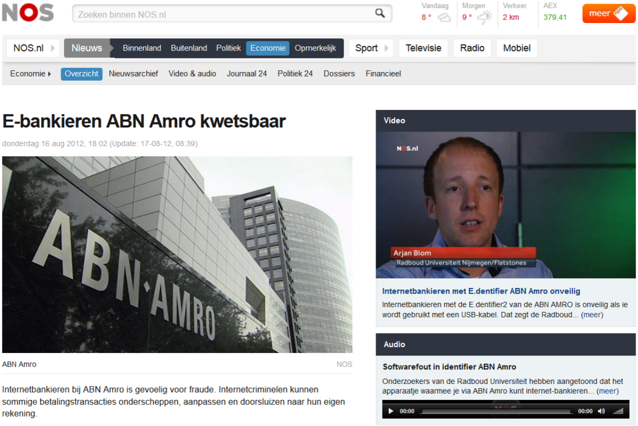
60 minutes, queue size 1

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

Arjan Blom
Radboud Universiteit Nijmegen/Flatstones

Internetbankieren met E.dentifier ABN Amro onveilig

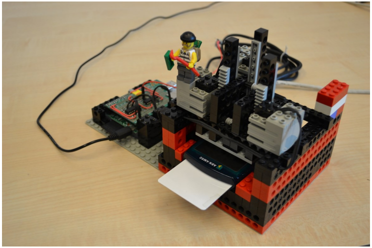
Internetbankieren met de E.dentifier2 van de ABN AMRO is onveilig als ie wordt gebruikt met een USB-kabel. Dat zegt de Radboud... (meer)

Audio

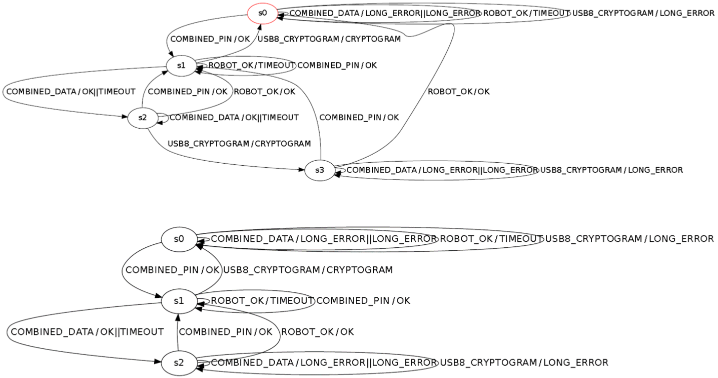
Softwarefout in identifier ABN Amro

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

Learning a Model of the E.dentifier2



State Machines for Old and New E.dentifier2



A Simple Galois Connection for Handling Subalphabets

Theorem

Let, for $i = 1, 2$, $\mathcal{M}_i = \langle I_i, O_i, Q_i, q_i^0, \rightarrow_i \rangle$ be (nondeterministic) Mealy machines with $I_1 \supseteq I_2$ and $O_1 = O_2$. Then

$$\mathcal{M}_1 \downarrow I_2 \leq \mathcal{M}_2 \quad \Leftrightarrow \quad \mathcal{M}_1 \leq \mathcal{M}_2 \uparrow I_1.$$

Here $\mathcal{M}_1 \downarrow I_2$ removes all transitions with input label not in I_2, and $\mathcal{M}_2 \uparrow I_1$ adds transitions to a chaos state for all inputs not in I_2.

A Galois Connection for Action Refinement

Assume we have sets X and Y of abstract inputs and outputs, and sets I and O of concrete inputs and outputs. An action refinement ρ is a pair of injective functions

$$\rho_i : X \to I^+ \qquad \rho_o : Y \to O^+$$

such that $\rho_i(x) \leq \rho_i(x') \Rightarrow x = x'$.

Galois Connection

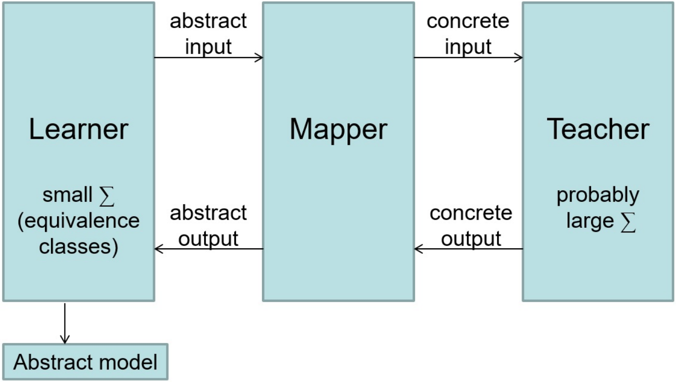
Then we can define monotone abstraction operators α_ρ and concretization operators γ_ρ such that:

Theorem

Let $\mathcal{M}$ be a Mealy machine over I and O, and let $\mathcal{N}$ be a Mealy machine over X and Y. If $\mathcal{M}$ and $\mathcal{N}$ "respect" refinement ρ then

$$\alpha_\rho(\mathcal{M}) \leq \mathcal{N} \quad \Leftrightarrow \quad \mathcal{M} \leq \gamma_\rho(\mathcal{N})$$

A Theory of Mappers (AJUV, 2015)



Transducers

Definition (Mapper)

A mapper for a set of inputs I and a set of outputs O is a deterministic Mealy machine $\mathcal{A} = \langle I \cup O, X \cup Y, R, r_0, \delta, \lambda \rangle$, where

- I and O are disjoint sets of concrete input/output symbols,
- X and Y are finite sets of abstract input/output symbols, and
- $\lambda : R \times (I \cup O) \to (X \cup Y)$, the abstraction function, respects inputs and outputs, that is, for all $a \in I \cup O$ and $r \in R$, $a \in I \Leftrightarrow \lambda(r, a) \in X$.

We assume that mappers are surjective: for every state r and every abstract symbol z there is a concrete symbol a with $\lambda(r, a) = z$. (Otherwise we get a useful theory, but no Galois connection.)

Abstraction

Definition (Abstraction)

Let $\mathcal{M} = \langle I, O, Q, q_0, \rightarrow \rangle$ be a Mealy machine and let $\mathcal{A} = \langle I \cup O, X \cup Y, R, r_0, \delta, \lambda \rangle$ be a mapper. Then $\alpha_{\mathcal{A}}(\mathcal{M})$, the abstraction of $\mathcal{M}$ via $\mathcal{A}$, is the Mealy machine $\langle X, Y, Q \times R, (q_0, r_0), \rightarrow \rangle$, where $\rightarrow$ is given by

$$\frac{q \xrightarrow{i/o} q', \;\; r \xrightarrow{i/x} r' \xrightarrow{o/y} r''}{(q, r) \xrightarrow{x/y} (q', r'')}$$

Concretization

Definition (Concretization)

Let $\mathcal{H} = \langle X, Y \cup \{\perp\}, H, h_0, \rightarrow\rangle$ be a Mealy machine and let $\mathcal{A} = \langle I \cup O, X \cup Y, R, r_0, \delta, \lambda\rangle$ be a mapper for I and O. Then $\gamma_{\mathcal{A}}(\mathcal{H})$, the concretization of $\mathcal{H}$ via $\mathcal{A}$, is the Mealy machine $\langle I, O, R \times H, (r_0, h_0), \rightarrow\rangle$, where $\rightarrow$ is given by

$$\frac{r \xrightarrow{i/x} r' \xrightarrow{o/y} r'', \;\; h \xrightarrow{x/y} h'}{(r, h) \xrightarrow{i/o} (r'', h')}$$

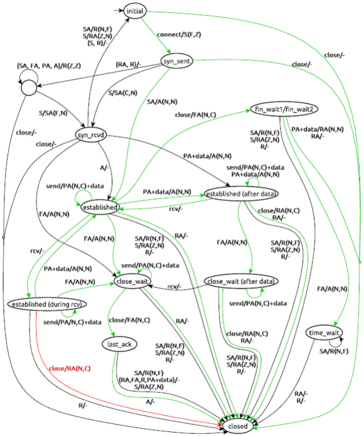
A Galois Connection that is Quite Useful

Theorem

For a mapper $\mathcal{A}$ and (nondeterministic) Mealy machines $\mathcal{M}$ and $\mathcal{H}$,

$$\alpha_{\mathcal{A}}(\mathcal{M}) \leq \mathcal{H} \quad \Leftrightarrow \quad \mathcal{M} \leq \gamma_{\mathcal{A}}(\mathcal{H})$$

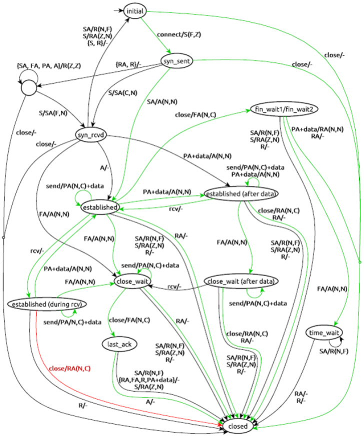
Bugs in Protocol Implementations



Standard violations found in implementations of major protocols:

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

Bugs in Protocol Implementations

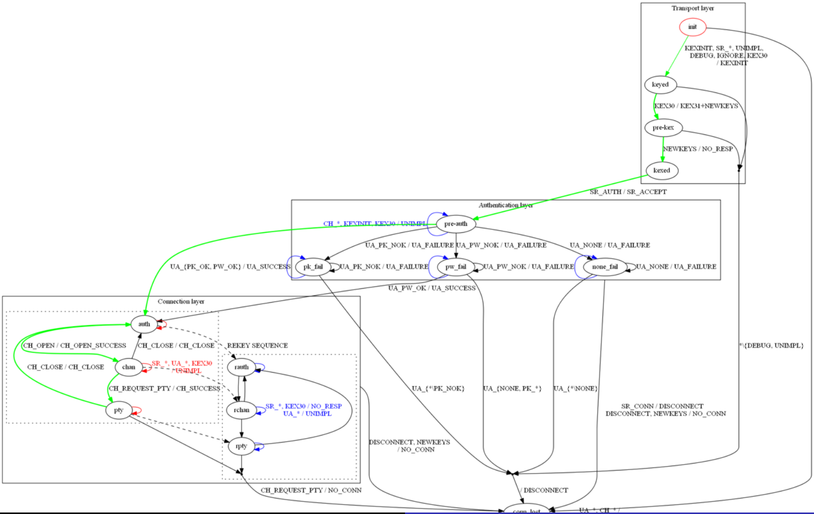


Standard violations found in implementations of major protocols:

- 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 Protocol Case Studies

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

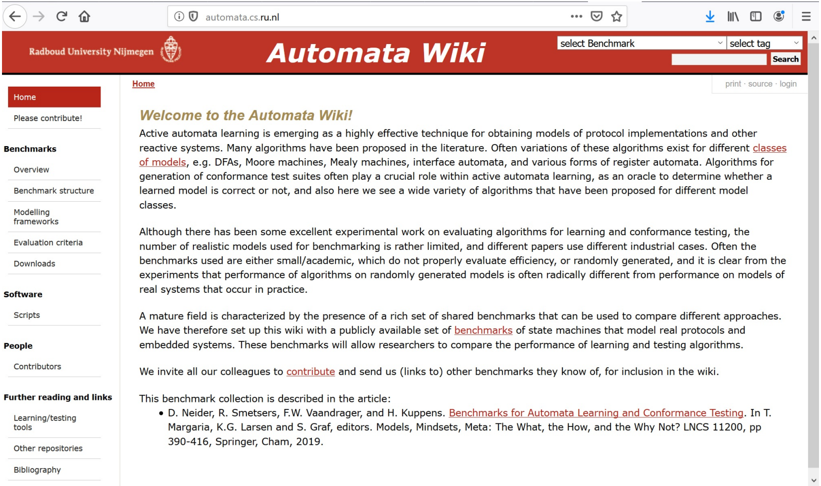
Lorentz Workshop

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

Working groups on e.g.,

- WiFi
- side channels in TLS
- LTE

Benchmark Wiki `automata.cs.ru.nl`



automata.cs.ru.nl

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
- Bibliography

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.

This benchmark collection is described in the article:

- D. Neider, R. Smetsers, F.W. Vaandrager, and H. Kuppens. Benchmarks for Automata Learning and Conformance Testing. In T. Margaria, K.G. Larsen and S. Graf, editors. Models, Mindsets, Meta: The What, the How, and the Why Not? LNCS 11200, pp 390-416, Springer, Cham, 2019.

Conclusions

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.

Galois connections provide useful abstractions.

Further research needed!

Future Work

1. Improved algorithms for black-box learning/testing FSMs
2. Better understanding of role Galois connections in learning; algorithms for finding Galois connections automatically
3. From Mealy machines to I/O automata
4. Learning EFSMs
5. Combinations of black-box and white-box learning
6. Algorithms for models with time and probabilities
7. Refactoring of legacy software excellent application domain