A Myhill-Nerode Theorem for Register Automata and Symbolic Trace Languages

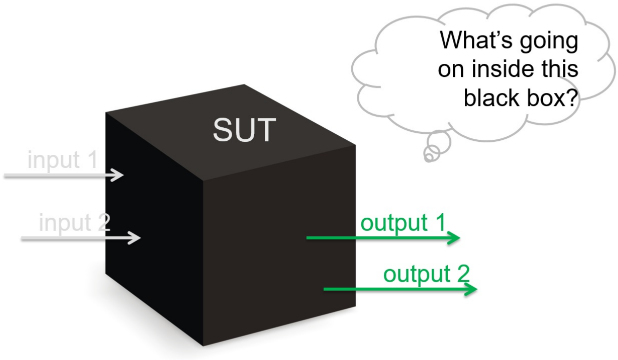
Frits Vaandrager Abhisek Midya

Radboud University Nijmegen

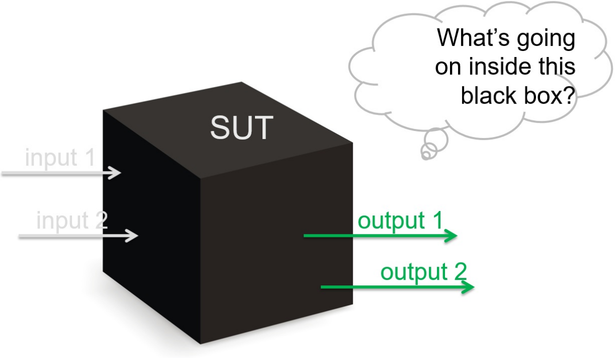
ICTAC, December 3, 2020

Outline

Black-box Automata Learning

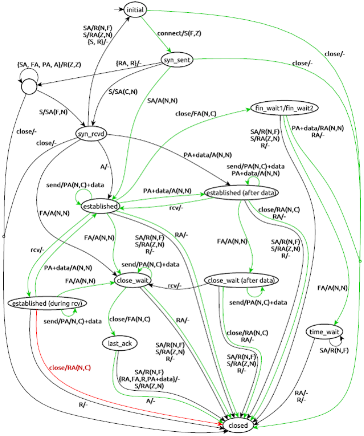


Black-box Automata Learning



System Under Test (SUT) behaves like deterministic state machine

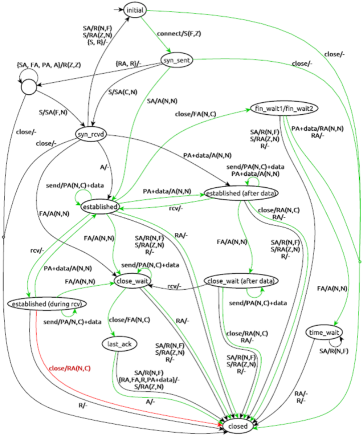
Success Stories Automata Learning



Standard violations found in implementations of major protocols:

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

Success Stories Automata Learning



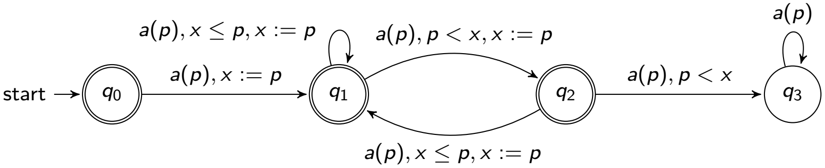
Standard violations found in implementations of major protocols:

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

These findings led to bug fixes in implementations.

Register Automata

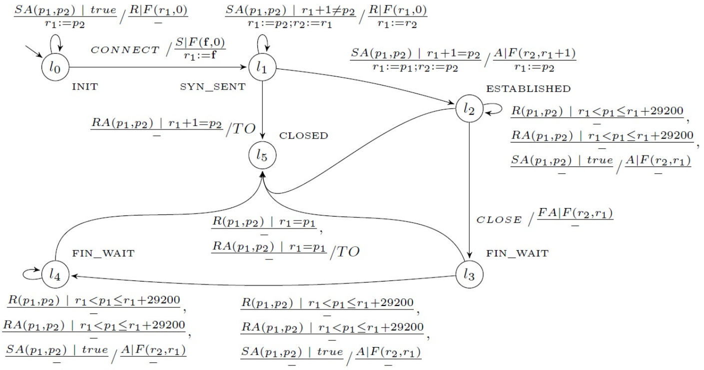
In applications inputs and outputs carry data parameters. State-of-the-art learning techniques either manually construct mappers to abstract from data parameters, or infer register automata (RAs) from black-box observations.



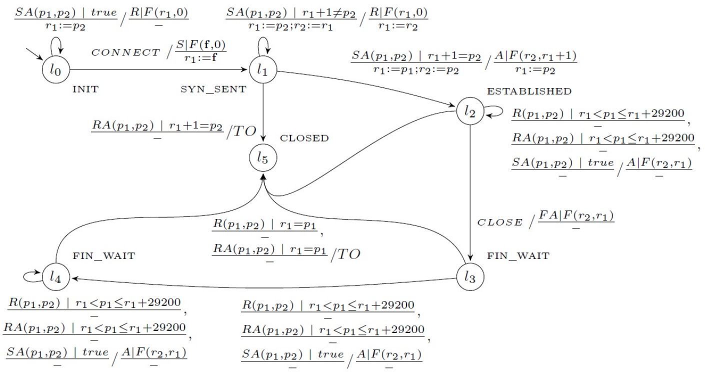
Learning Tools for Register Automata

- Tomte, Radboud University, only handles equality predicates
- LearnLib, TU Dortmund, only handles equality predicates
- RALib, Uppsala/Dortmund, handles a few other predicates

RALib TCP Case Study (FMICS-AVoCS'17)

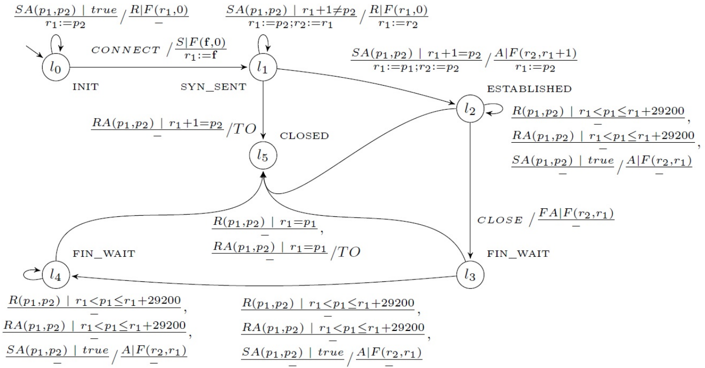


RALib TCP Case Study (FMICS-AVoCS'17)



These findings led to bug fix in Linux TCP implementation!

RALib TCP Case Study (FMICS-AVoCS'17)



These findings led to bug fix in Linux TCP implementation!

... but > 200.000 inputs were needed to learn model

Limits of Black-box Learning?

- Black-box learning is highly effective bug finding technique

Limits of Black-box Learning?

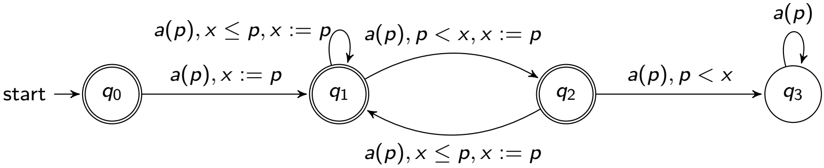
- Black-box learning is highly effective bug finding technique
- ... but it has scalability problems

Limits of Black-box Learning?

- Black-box learning is highly effective bug finding technique
- ... but it has scalability problems
- ... and fundamental restrictions on supported data types
- Challenge: use white-box information while preserving extensionality of black-box models

Data Words

In black-box learning, we observe (accepted) data words.

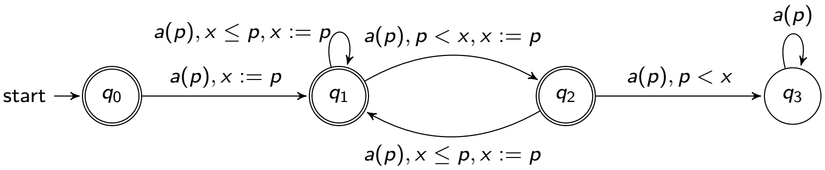


Register automaton accepts data word $a(1)a(4)a(0)a(7)$ via run

$$(q_0, []) \xrightarrow{a(1)} (q_1, [x \mapsto 1]) \xrightarrow{a(4)} (q_1, [x \mapsto 4]) \xrightarrow{a(0)} (q_2, [x \mapsto 0]) \xrightarrow{a(7)} (q_1, [x \mapsto 7])$$

Symbolic Words

Symbolic words record constraints on data parameters encountered during a run, using variable v_i as a marker for the i-th input value.



RA accepts symbolic word $a \top a\ v_1 \leq v_2\ a\ v_3 < v_2\ a\ v_3 \leq v_4$ via symbolic run

$$(q_0, []) \xrightarrow{a,\top,x:=p} (q_1, [x \mapsto v_1]) \xrightarrow{a,x\leq p,x:=p} (q_1, [x \mapsto v_2])$$

$$\xrightarrow{a,p<x,x:=p} (q_2, [x \mapsto v_3]) \xrightarrow{a,x\leq p,x:=p} (q_1, [x \mapsto v_4])$$

Symbolic Languages

- Symbolic words can be observed using white-box techniques like tainting, symbolic execution and concolic execution
- Can we adapt learning algorithms to this grey-box setting?

Symbolic Languages

- Symbolic words can be observed using white-box techniques like tainting, symbolic execution and concolic execution
- Can we adapt learning algorithms to this grey-box setting?
- Yes, see our iFM'20 paper! Approach effective, but ad hoc and limited to predicates like $=$ and $<$
- Can we do better?

A Classic Result

Definition (Nerode equivalence)

The equivalence relation $\sim_L$ on Σ^* induced by a language $L \subseteq \Sigma^*$:

$$u \sim_L v \quad \textit{iff} \quad \forall w \in \Sigma^* : u \cdot w \in L \Leftrightarrow v \cdot w \in L$$

Theorem (Myhill-Nerode, 1958)

Language L is regular iff $\sim_L$ has finitely equivalence classes. Moreover, the number of states in the smallest deterministic finite automaton (DFA) recognizing L is equal to the number of equivalence classes (index) of $\sim_L$.

Importance of Myhill-Nerode

Myhill-Nerode Theorems (MNTs) are crucial for model learning:

- Angluin's classical L^* algorithm for learning regular languages is based on MNT and approximates Nerode congruence
- Maler & Steiger establish MNT for ω-languages that serves as basis for learning algorithm
- SL^* learning algorithm of Cassel et al based on MNT for data languages and register automata
- Francez & Kaminski, Benedikt et al, and Bojańczyk et al present MNTs for data languages

Importance of Myhill-Nerode

Myhill-Nerode Theorems (MNTs) are crucial for model learning:

- Angluin's classical L^* algorithm for learning regular languages is based on MNT and approximates Nerode congruence
- Maler & Steiger establish MNT for ω-languages that serves as basis for learning algorithm
- SL^* learning algorithm of Cassel et al based on MNT for data languages and register automata
- Francez & Kaminski, Benedikt et al, and Bojańczyk et al present MNTs for data languages

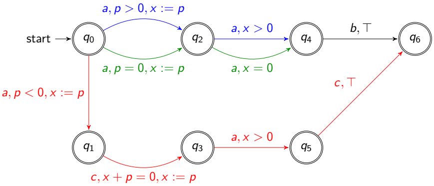
Can we come up with MNT for symbolic languages?

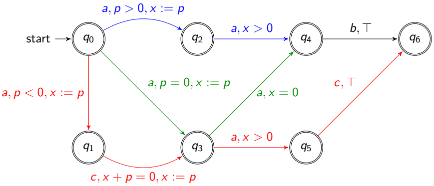
No Unique Minimal Register Automaton

Consider symbolic language L with 3 symbolic words and their prefixes:

$$
\begin{aligned}
w &= a\ v_1 > 0\ a\ v_1 > 0\ b\ \top \\
u &= a\ v_1 = 0\ a\ v_1 = 0\ b\ \top \\
z &= a\ v_1 < 0\ c\ v_1 + v_2 = 0\ a\ v_2 > 0\ c\ \top
\end{aligned}
$$

Both register automata below accept L:





Nerode Relations

We look for symbolic versions of what Kozen calls Nerode relations:

Definition

A relation $\equiv_l$ on Σ^* is a Nerode relation if it satisfies the following three conditions, for $u, v \in \Sigma^*$ and $\alpha \in \Sigma$,

$$
\begin{array}{rcll}
u \equiv_l v & \Rightarrow & (u \in L \Leftrightarrow v \in L) & \\
u \equiv_l v & \Rightarrow & u\alpha \equiv_l v\alpha & \text{(“right invariance”)} \\
\equiv_l & \text{has} & \text{finite index} &
\end{array}
$$

Symbolic Languages

Definition (Feasible)

Let $w = \alpha_1 G_1 \cdots \alpha_n G_n$ be a symbolic word. Then w is feasible if

1. $guard(w) = G_1 \wedge \cdots \wedge G_n$ is satisfiable, and
2. $Var(G_i) \subseteq \{v_1, \ldots, v_i\}$, for each $i \in \{1, \ldots, n\}$.

A symbolic language is feasible if it is prefix closed and consists of feasible symbolic words.

Regular Symbolic Languages

Definition (Regularity)

A feasible symbolic language L is regular iff there exist three relations:

- an equivalence $\equiv_l$ on L, called location equivalence,
- an equivalence $\equiv_t$ on $L \setminus \{\epsilon\}$, called transition equivalence,
- a partial equivalence $\equiv_r$ on $\{(w, v_i) \in L \times \mathcal{V} \mid i \leq \mathit{length}(w)\}$, called register equivalence; w stores v if $(w, v) \equiv_r (w, v)$.

We require $\equiv_l$ and $\equiv_t$, as well as the equivalence induced by $\equiv_t$ to have finite index. Furthermore, we require ...

Regular Symbolic Languages (cnt)

$$(w, v) \equiv_r (w, v') \Rightarrow v = v' \tag{1}$$

$$w\alpha G \equiv_t w'\alpha' G' \;\Rightarrow\; w \equiv_l w' \tag{2}$$

$$w\alpha G \equiv_t w'\alpha' G' \;\Rightarrow\; \alpha = \alpha' \tag{3}$$

$$w\alpha G \equiv_t w'\alpha G' \wedge \sigma = matching(w, w') \;\Rightarrow\; G[\sigma] \equiv G' \tag{4}$$

$$w \equiv_t w' \Rightarrow w \equiv_l w' \tag{5}$$

$$w \equiv_t w' \wedge w \text{ stores } v_m \Rightarrow (w, v_m) \equiv_r (w', v_n) \tag{6}$$

$$u \equiv_t u' \wedge u = w\alpha G \wedge u' = w'\alpha G' \wedge (w, v) \equiv_r (w', v') \wedge u \text{ stores } v$$
$$\Rightarrow (u, v) \equiv_r (u', v') \tag{7}$$

$$u \equiv_t u' \wedge u = w\alpha G \wedge u' = w'\alpha G' \wedge (u, v) \equiv_r (u', v') \wedge v \neq v_{m+1}$$
$$\Rightarrow (w, v) \equiv_r (w', v') \tag{8}$$

$$w \equiv_l w' \wedge w\alpha G \in L \wedge v \in Var(G) \setminus \{v_{m+1}\} \Rightarrow \exists v' : (w, v) \equiv_r (w', v') \tag{9}$$

$$w \equiv_l w' \wedge w\alpha G \in L \wedge \sigma = matching(w, w')$$
$$\wedge\, Sat(guard(w') \wedge G[\sigma]) \Rightarrow w'\alpha G[\sigma] \in L \qquad \text{“right invariance”} \tag{10}$$

$$w \equiv_l w' \wedge w\alpha G \in L \wedge w'\alpha G' \in L \wedge \sigma = matching(w, w')$$
$$\wedge\, Sat(G[\sigma] \wedge G') \Rightarrow w\alpha G \equiv_t w'\alpha G' \qquad \text{“determinism”} \tag{11}$$

Regular Symbolic Languages (cnt)

Definition (Matching)

We define *matching*(w, w') as the variable renaming σ that maps each marker of w to the marker of w' stored in the same register (if any):

$$\sigma(v) = \begin{cases} v' & \text{if } (w, v) \equiv_r (w', v') \\ v_{n+1} & \text{if } v = v_{m+1} \\ \text{undefined} & \text{otherwise} \end{cases}$$

Here m and n denote number of inputs in w and w', respectively.

Main Result

Theorem (Soundness)

Suppose $\mathcal{A}$ is a register automaton. Then $L_s(\mathcal{A})$ is regular.

Theorem (Completeness)

Suppose L is a regular symbolic language over Σ. Then there exists a register automaton $\mathcal{A}$ such that $L = L_s(\mathcal{A})$.

Conclusions and Future Work

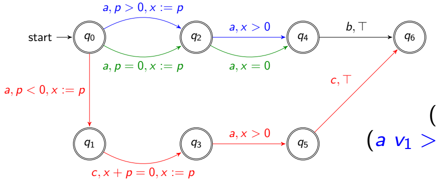
1. Register automata can be defined directly from a regular symbolic language, with locations materializing as equivalence classes of $\equiv_l$, transitions as equivalence classes of $\equiv_t$, and registers as equivalences classes of $\equiv_r$
2. No restrictions on allowed data types!
3. Challenge: develop learning algorithm based on our result
4. Refactoring of legacy software excellent application domain

Example

Consider symbolic language with following symbolic words (plus prefixes):

$$
\begin{aligned}
w &= a\ v_1 > 0\ a\ v_1 > 0\ b\ \top \\
u &= a\ v_1 = 0\ a\ v_1 = 0\ b\ \top \\
z &= a\ v_1 < 0\ c\ v_1 + v_2 = 0\ a\ v_2 > 0\ c\ \top
\end{aligned}
$$

Words are $\equiv_l$ equivalent if they lead to same location in RA below, $\equiv_t$ equivalent if they share final transition, and $(y, v) \equiv_r (y', v')$ if symbolic values v and v' are stored in same register after words y and y', resp.



$$
\begin{array}{rcl}
w & \equiv_t & u \\
w & \not\equiv_t & z \\
w & \equiv_l & z \\
(w, v_2) & \not\equiv_r & (w, v_2) \\
(a\ v_1 > 0, v_1) & \equiv_r & (a\ v_1 < 0\ c\ v_1 + v_2 = 0, v_2)
\end{array}
$$