A practical introduction to active automata learning

Bernhard Steffen, Falk Howar, Maik Merten
TU Dortmund

SFM2011
Overview

- Motivation
- Introduction to active automata learning
- Practical aspects in active automata learning
- Conclusions
Connect Scenario

- CONNECT environment
- Interrogate
- Learn
- Look for known models
- Some service
- Connect Scenario
- Connector
- Try to use
- Inform about new service and device
- Learner
Learning in CONNECT

Develop techniques for learning ... models of ... behavior of networked peers and middleware through exploratory interaction...

Interface descriptions

Semantic information on
- Data domains
- Data dependencies
- Effects

WP4 Learning Enabler

Rich & abstract models:
- Data parameters & state variables
- Pre- and post-conditions
- Non-functional properties

Metrics of interest

Counterexamples through monitoring
Overview

- Motivation
- Introduction to **active automata learning**
- **Practical aspects** in active automata learning
- Conclusions
Mealy machines

- Mealy machine $M = (S, \Sigma, \Gamma, \sigma, \gamma)$
  - $S$ finite set of states
  - $\Sigma$ finite input-alphabet
  - $\Gamma$ finite output-alphabet
  - $\sigma : (S \times \Sigma) \rightarrow S$ transition-function
  - $\gamma : (S \times \Sigma) \rightarrow \Gamma$ output-function

- Words $\Sigma^*$ for $(s \in S, a \in \Sigma, w \in \Sigma^*)$
  - $\sigma : (S \times \Sigma^*) \rightarrow S$, $\sigma(s, \varepsilon) = s$, $\sigma(s, aw) = \sigma(\sigma(s, a), w)$
  - $\gamma : (S \times \Sigma^*) \rightarrow \Gamma^*$, $\gamma(s, \varepsilon) = \varepsilon$, $\gamma(s, aw) = \gamma(s, a) \cdot \gamma(\sigma(s, a), w)$
Passive learning or learning with traces

- tape-record communication
- Create observation tree
- Construct automaton without contradiction
Passive learning or learning with traces

a/0 a/1 b/1
a/0 b/0 a/0
b/1 a/0 b/0
b/1 b/0 a/0
Passive learning or learning with traces

"equivalent" states (no contradiction)
Passive learning or learning with traces

Could just as well be red...
Passive learning or learning with traces

a/0 a/1 b/1
a/0 b/0 a/0
b/1 a/0 b/0
b/1 b/0 a/0
Observations

- The relation "not in conflict" is very weak:
  - Reflexive, symmetric, but not transitive!
  - `Not in conflict` clusters typically overlap
  - The relation contain various equivalence relations
  - Computing the best choice of equivalence is:
    - Expensive for criteria like state minimality
    - Impossible in terms of adequacy for the problem.
Idea 1: Ask where information is incomplete!

- This requires an active testing mechanism:
  - **Membership Queries**: Check the reaction of the system to input sequences.
  - Checking all inputs at all positions makes `not in conflict` an equivalence relation.
Angluin's algorithm

Consequence:
- The underlying tree is homogeneous in the sense that all nodes treat the same set of inputs.
- As the not in conflict relation is now an equivalence relation, the corresponding clustering is unique.

Problem:
- The clustered graph may be non-deterministic in general.
Idea 2: Enforce consistency!

- Refine the 'not in conflict' relation
- Also consider whether the target of the transitions of each cluster are unique for each input
  - I.e.: consider the largest congruence wrt. The Transition relation inside the not in conflict relation).

This yields determinism!
Consequence:
- Clustering yields an (input) deterministic graph / model
- The **projective quotient model** of a consistent and homogeneous abstraction

This simplifies the situation a lot:

**Termination Lemma 1**

Given some execution tree, realizing the two ideas via Membership Queries provides a closed, consistent, and deterministic Hypothesis Model

**(Quality? Termination?)**
Idea 3: Introduce **qualitative termination!**

- **Equivalence Queries:** Check for equivalence with the target system, and produce a distinguishing test in case of failure.

  - **Conceptually** a nice idea that leads to a very elegant correctness proof.
  - **Practically** typically not implementable.
Active automata learning

$\Sigma = \{a, b\}$

Learner

$\begin{align*}
a \in L? \\
\text{no} \\
\text{no, } bb \in L!
\end{align*}$

MQ-Oracle

$a \in L? \\
\text{no}$

EQ-Oracle

$b \in L? \\
a, b \in L$
(queries) should word w be included in L(A)?

yes / no

(conjectures) here is an A – is L(A) = U?

yes!

no: word w should (not) be in L(A)
Angluin's alg. for Mealy machines

- Initialize

**Distinguishing Set**

\( D \) with alphabet of inputs
### Angluin's alg. for Mealy machines

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>ε</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>b</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Unknown system:**

```
(1) → (2) a/0 → (3) b/0
(2) → (3) a/1 → (1) b/0
(3) b/1 → (2) a/0
```

[Diagram showing transitions between states for the unknown system.]
### Angluin's alg. for Mealy machines

- **Unclosure**: Rows in lower part that are not in upper part

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>ε</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>b</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
**Angluin's alg. for Mealy machines**

- **Unclosure**: Rows in lower part that are not in upper part.

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\varepsilon$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>b</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>aa</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>ab</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Angluin's alg. for Mealy machines

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>ε</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>b</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>aa</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>ab</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

- **Conjecture:**
  - Unique rows in $S$ become states
  - Rows in $S$ and $SA$ become transitions
Angluin's alg. for Mealy machines

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\varepsilon$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>b</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>aa</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>ab</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Counterexample:**

```
bbb / 010
```
Angluin’s alg. for Mealy machines

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>ε</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>b</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>bb</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>bbb</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>aa</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>ab</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>ba</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

- **Counterexample**: bbb / 010
- Insert all prefixes of the counterexample to upper part
- Extend SA accordingly
Inconsistency:

Equal rows in upper part have 'different extensions.'

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\varepsilon$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>b</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>bb</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>bbb</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>aa</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>ab</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>ba</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Angluin's alg. for Mealy machines

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\varepsilon$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>b</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>bb</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>bbb</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>aa</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>ab</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>ba</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

- **Inconsistency:**
- Equal rows in upper part have 'different extensions'
- b and bbb differ, e.g., for suffix b

$\Rightarrow \varepsilon$ and bb will differ for suffix bb
Angluin's alg. for Mealy machines

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>bb</th>
</tr>
</thead>
<tbody>
<tr>
<td>ε</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>a</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>b</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>bb</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>bbb</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>aa</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>ab</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>ba</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

- **Inconsistencies** lead to new columns

**New Conjecture**
Angluin's alg. for Mealy machines

Target System

Learned System
Summarized Observations (1)

- Systematic completion of the observation table
- **New states** arise as targets of transitions or from counter examples of the equivalence queries. Technically: *prefixes* are added to $S$
- **Closure** procedure extends $SA$
- **Consistency** is enforced by *enlarging* the Distinguishing Set $D$
Hypothesis models or conjectures:

- Closed and consistent models \(\text{(projective quotients)}\) of the so far expanded
  - *homogeneously extended, and*
  - *consistent*

execution tree.
Invariance Lemma:

- All hypothesis models are 
  - **totally defined**: each input is considered at each state,
  - **input deterministic**: there is only one transition per input at each state,
  - **transition covered**: each transition lies on a path of the original system,
  - **state minimal**: two different states in a hypothesis model always have a separating future – *à la Nerode*).
Nerode relation:

For language $L$ define relation $R_L$ (for $u, u' \in \Sigma^*$)

$$u R_L u' \iff \text{for all } v \in \Sigma^*: \ (uv \in L \iff uv \in L)$$

Myhill-Nerode Theorem:

Minimal number of states of an accepting deterministic automaton equals the number of equivalence classes of $R_L$
This (Nerode‘s theorem) directly yields:

- **Corollary**: Hypothesis automata have at most as many states as the **smallest deterministic** equivalent **automaton**.

- We will denote the number of states by \( n \).
Summarized Observations (4)

- **Lemma**: The number of states of the hypothesis model increases in response to a counterexample.

- **Theorem**: Angluin´s algorithm terminates after at most $n$ equivalence queries with the smallest deterministic system representing the behaviour of the system to be learned.
**Equivalence Queries**
At most $|Q|$

**Membership Queries**
At most $O(m |Q| |\Sigma_A|)$ per EQ ($m = \text{length of max. counter example}$)

Max. size of table = $O(m |Q|^2 |\Sigma_A|)$.

**Theorem** (Complexity for const. Time MQs and EQs).
$O( m |Q|^2 |\Sigma_A| )$.

For $m$ in $O(|Q|)$ the complexity result reads: $O(|Q|^3 |\Sigma_A|)$.
Remaining Problems

- High Computational Complexity
- Even worse: equivalence queries in general undecidable.

In essence:
- Active automata learning always remains at the level of hypotheses:
  - neither correct nor complete
Further Developments
All prefixes of counterexample... one essential suffix

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>bb</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\varepsilon$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>bb</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>bbb</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>aa</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>ab</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>ba</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Rivest and Shapire: Analyze counterexample separately (not in the table)
  - Only add one 'essential' suffix (i.e., witness), as column label to the table

Consequence: Guaranteed Consistency!

BUT: Hypothesis Automata are no longer guaranteed to be minimal!

(cf. Pnueli / Mahler‘s criticism)
Reduced observation table (contd.)

- Saves membership queries! (by saving rows in the observation table)
**Equivalence Queries**
At most $|Q|$

**Membership Queries for guaranteed progress after Eqs**
At most $O(\log_2(m) + |\Sigma_A| |Q|)$ per EQ ($m =$ length of max. counter example)

Max. size of table = $O(|Q|^2 |\Sigma_A|)$.

**Theorem** (Complexity for const. Time MQs and EQs).
$O( |Q|^2 |\Sigma_A| + |Q| \log_2(m) )$.

For $m$ in $O(|Q|)$ the complexity result reads: $O(|Q|^2 |\Sigma_A|)$
## Conceptual Improvements 2

All rows are filled completely, even if unnecessary.
Discrimination tree

‘Sink’ words into table through discrimination tree

**Angluin**: Add suffix globally to all rows
- leads to unclosedness
- resolved by new elements in $S$

**Kearns & Vazirani**: Add suffix only locally
- Suffix only added to one ‘essential’ sub-table.
- Prefix known from counterexample
Discrimination tree (contd.)

Kearns & Vazirani + discrimination tree

- **Saves membership queries!** (by saving entries in the observation table)
- **More equivalence queries!** (using suffixes globally may be a good heuristic sometimes)
- Worst case complexity unchanged
**Lemma.** Each counterexample leads to at least one new state.

**Lemma.** The hypothesis automata are guaranteed to have fewer states than the minimal deterministic finite automaton for the considered language.

**Theorem (for perfect equivalence oracle)**

The algorithm terminates with the smallest deterministic automaton for the considered language / set of traces.
Overview

- Motivation
- Introduction to **active automata learning**
- **Practical aspects** in active automata learning
- Conclusions
Practical results II

The ZULU competition
The ZULU challenge

- Competition in active learning (2010)
- No equivalence queries allowed, limited amount of membership queries
- Randomly generated automata
- Test-based evaluation
- The set $S \cup SA$ defines a monotonically growing spanning tree of the target automaton.
- Usually only local modifications between two equivalence queries (especially for non-uniform sets of distinguishing suffixes)
Continuous equivalence queries

- **Select transition:** randomly from set of non-blocked
- **Generate future:** randomly with increasing length. Initially $\max\left\{ \frac{\log(n)}{2}, 3 \right\}$.
- **Book keeping:**
  - **E.H.Blocking:** transitions excluded from subsequent tests.
  - **E.H.Weighted:** weights on transitions are increased.
- **Termination:** ZULU limit

![Diagram of the process](image)
## ZULU competition results

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Dist. Set</th>
<th>Equivalence</th>
<th>Training (Avg.)</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>E.H. Blocking</td>
<td>Init.</td>
<td>block transitions</td>
<td>89.38</td>
<td>1</td>
</tr>
<tr>
<td>E.H. Weighted Random</td>
<td>Uniform</td>
<td>weight transitions</td>
<td>89.26</td>
<td>2</td>
</tr>
<tr>
<td>E.H. Weighted Random</td>
<td>no</td>
<td>random walks</td>
<td>88.93</td>
<td>6</td>
</tr>
<tr>
<td>run_random</td>
<td>${\epsilon}$</td>
<td>random walks</td>
<td>80.17</td>
<td>14</td>
</tr>
<tr>
<td>run_blocking1</td>
<td>${\epsilon} \cup \Sigma$</td>
<td>block transitions</td>
<td>79.89</td>
<td>15</td>
</tr>
<tr>
<td>run_weighted1</td>
<td>yes</td>
<td>weight transitions</td>
<td>79.65</td>
<td>16</td>
</tr>
</tbody>
</table>

Kearns & Vazirani: High impact even here!

- Uniform DFA: ca. 83, non-uniform Mealy: ca. 85
Detailed results

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>New Membership</th>
<th>Queries</th>
<th>Rounds</th>
<th>States</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Close</td>
<td>Analyze</td>
<td>Search</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E.H. Blocking</td>
<td>6,744</td>
<td>358</td>
<td>999</td>
<td>259</td>
<td>352</td>
</tr>
<tr>
<td>E.H. Weighted</td>
<td>6,717</td>
<td>349</td>
<td>1,035</td>
<td>262</td>
<td>351</td>
</tr>
<tr>
<td>Random</td>
<td>6,586</td>
<td>519</td>
<td>996</td>
<td>228</td>
<td>332</td>
</tr>
<tr>
<td>run_random</td>
<td>8,080</td>
<td>14</td>
<td>7</td>
<td>5</td>
<td>312</td>
</tr>
<tr>
<td>run_blocking1</td>
<td>8,074</td>
<td>11</td>
<td>16</td>
<td>6</td>
<td>319</td>
</tr>
<tr>
<td>run_weighted1</td>
<td>8,077</td>
<td>9</td>
<td>15</td>
<td>6</td>
<td>319</td>
</tr>
</tbody>
</table>

- **ZULU limit:** 8,101
- **MQs / EQ:** 1-3 (uniform), ca. 3.9 (non-uniform), ca. 4.36 (random)
- **MQS / State:** ca. 25 (uniform), ca. 19 (non-uniform)
- Random Walks: higher costs for analyzing counterexamples
Asymptotic costs per state
More Applications
Practical challenges

Interface description
etc.

interfacing real systems:
- alphabet generation
- abstraction
- data

equivalence queries
Behavioral models

memberships queries
Learning assumptions
oracle for WA in assume-guarantee reasoning

Query $c$:

$\langle A \rangle M_1 \langle P \rangle$

(simulate $s$ on $M_1 \parallel P_{err}$)

Conjecture: $A_i$

$\langle A_i \rangle M_1 \langle P \rangle$

(model check)

True

$\langle true \rangle M_2 \langle A_i \rangle$

(model check)

True

$\langle true \rangle M_1 \parallel M_2 \langle P \rangle$

false + crex $c$

false

P satisfied

false + crex $c$

true

query $c \uparrow \alpha A$

true

false

P violated

1. $\langle A \rangle M_1 \langle P \rangle$

2. $\langle true \rangle M_2 \langle A \rangle$

http://connect-forever.eu/
Practical results I

Learning the OCS
User model: paper workflow
Event Condition Action

Submit Report

Reviewer

Is reviewer?

true

false

Is report already submitted?

false

true

Grant

Deny

enable: view report paper
default

enable: edit report paper
default

enable: view report paper
default

enable: edit report paper
default

enable actions

http://connect-forever.eu/
Semantics of "phase expires" - edges (1)
Semantics of "phase expires" - edges (2)

Send notification

default

default

Listen for wake up event

Deadline reached?

true

false

Sleep

start next phase

default

stop current phase
Many participants
Putting it all together
Regular extrapolation
Optimized learning setup

SubmitPaper, Interrupt Submission, UploadDocument

login(submitter)
paper = submit(title, ...)
logout()

login(pchair)
interrupt(submission)
logout()

login(submitter)
uploadDoc(paper, doc)
logout()
Learning algorithm

- Observation Table
- Mealy machine inference
- Regular extrapolation

First Hypothesis

<table>
<thead>
<tr>
<th></th>
<th>SP</th>
<th>UD</th>
<th>DD</th>
<th>DP</th>
</tr>
</thead>
<tbody>
<tr>
<td>λ</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SP</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>UD</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DD</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DP</td>
<td></td>
<td>✔</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPSP</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>SPUD</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>SPDD</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>SPDP</td>
<td>✔</td>
<td></td>
<td></td>
<td>✔</td>
</tr>
</tbody>
</table>
Reusing system states

<table>
<thead>
<tr>
<th>( \lambda )</th>
<th>SP</th>
<th>UD</th>
<th>DD</th>
<th>DP</th>
</tr>
</thead>
<tbody>
<tr>
<td>SP</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>UD</td>
<td>5</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DD</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Diagram:

- Step 1: SP
- Step 2: SP, UD
- Step 3: SP, UD, DD
- Step 4: SP, UD, DD, DP
- Step 5: SP, UD, DD, DP
- Step 6: SP, UD, DD, DP
Reuse tree on our example

- 52 Membership Queries
- Saved 12 Resets
Exploitation: failure invariance

- Domain knowledge
- Failing actions due to missing permissions
- OCS is transaction secure (roll back in case of error)
- Partition output alphabet into successful and failed execution
- Reflexive edges indicate failure output
Pumping: Unfolding edges

- For 52 Membership Queries only 10 Resets necessary
- 50 Symbols executed (of 148)

Queries 9 to 20 will be ‘pumped’, e.g.
- UD UD or
- DP SP
already known
Exploitation: Invariant symbols

- Downloading (reading) a document (DD) does not change a system state.
- The state can be kept for reuse.
- Only 6 Resets and 35 Symbols need to be executed.

Failure invariance + invariant input symbol DD.

○ indicates allowed outputs.
Statistics: Learning the OCS

438 MQs containing 1734 Symbols:

<table>
<thead>
<tr>
<th></th>
<th>Resets</th>
<th>Reuses</th>
<th>Pumped</th>
<th>Reset [t]</th>
<th>Symbols [t] (♯)</th>
<th>Observed [t]</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>438</td>
<td>0</td>
<td>0</td>
<td>7m 50s</td>
<td>8m 28s (1734)</td>
<td>16m 18s</td>
</tr>
<tr>
<td>(b)</td>
<td>366</td>
<td>72</td>
<td>0</td>
<td>7m 14s</td>
<td>7m 55s (1518)</td>
<td>15m 9s</td>
</tr>
<tr>
<td>(c)</td>
<td>328</td>
<td>86</td>
<td>24</td>
<td>5m 23s</td>
<td>5m 51s (1345)</td>
<td>11m 14s</td>
</tr>
<tr>
<td>(d)</td>
<td>56</td>
<td>130</td>
<td>252</td>
<td>0m 52s</td>
<td>1m 25s (344)</td>
<td>2m 17s</td>
</tr>
<tr>
<td>(e)</td>
<td>37</td>
<td>125</td>
<td>276</td>
<td>0m 34s</td>
<td>0m 59s (252)</td>
<td>1m 33s</td>
</tr>
</tbody>
</table>

- a) No reuse
- b) Only direct re-usage
- c) Exploit input knowledge
- d) Exploit output knowledge
- e) Exploit input and output knowledge
MQs = Resets + Reuses + Pumped

- a) Only direct reusage
- b) Exploit of failure outputs
- c) Like b) but one input marked as invariant
- d)-f) failure outputs and invariant for growing learn alphabets
Statistics: Learning the OCS

- Reset times are growing
- Observed runtime included execution of input symbols

<table>
<thead>
<tr>
<th></th>
<th>Resets</th>
<th>Avg. reset</th>
<th>Accum. reset</th>
<th>Observed runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>21</td>
<td>1.8s.</td>
<td>56s.</td>
<td>40s.</td>
</tr>
<tr>
<td>(b)</td>
<td>31</td>
<td>2.3s.</td>
<td>10m</td>
<td>1m 25s.</td>
</tr>
<tr>
<td>(c)</td>
<td>17</td>
<td>2.1s.</td>
<td>9m</td>
<td>48s.</td>
</tr>
<tr>
<td>(d)</td>
<td>137</td>
<td>3.2s.</td>
<td>3h 27m</td>
<td>10m 30s</td>
</tr>
<tr>
<td>(e)</td>
<td>646</td>
<td>4.1s.</td>
<td>13h 52m</td>
<td>53m 50s</td>
</tr>
<tr>
<td>(f)</td>
<td>5598</td>
<td>12.7s.</td>
<td>over 8 days</td>
<td>$\approx$ 22h</td>
</tr>
</tbody>
</table>

Accumulated reset time is highly **optimistic**!
Simple User Process

SP: Submit Paper
UD: Upload Document
IS: Interrupt Submission
IU: Interrupt Upload
RS: Restart Submission
RU: Restart Upload
Learned automaton
Overview

- Motivation
- Introduction to active automata learning
- Practical aspects in active automata learning
- Conclusions
Conclusions

Active Automata Learning:

- its practice has many facets:
  - Abstraction
  - Instrumentation
  - Reuse/Optimization

- It establishes a **new system perspective**

  Systems as **evolving ,beasts‘:**
  - to be observed continuously
  - difficult to control:

  **Forget the ‚Why/How‘ focus on the ‚What‘ !**