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
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 $(s \in S, a \in \Sigma, w \in \Sigma^*)$
  - $\sigma : (S \times \Sigma^*) \rightarrow S$, \( \sigma(s, \varepsilon) = s \), \( \sigma(s, aw) = \sigma(\sigma(s, a), w) \)
  - $\gamma : (S \times \Sigma^*) \rightarrow \Gamma^*$, \( \gamma(s, \varepsilon) = \varepsilon \), \( \gamma(s, aw) = \gamma(s, a).\gamma(\sigma(s, a), w) \)
Active automata learning

\[ \Sigma = \{a, b\} \]

Learner

\[ a \in L? \]
\[ \text{no} \]

MQ-Oracle

\[ a \in L? \]
\[ \text{no} \]

EQ-Oracle

\[ b? \]
\[ a \in L? \]
\[ \text{no, } bb \in L! \]
Angluin's alg. for Mealy machines

- Initialize Distinguishing Set $D$ with alphabet of inputs

Diagram:
- $D$ is the distinguishing set
- $S$ is the state cover set
- $\varepsilon$ is the empty input
- One transition extensions $SA$
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:

http://connect-forever.eu/
Angluin's alg. for Mealy machines

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

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

- Unique rows in $S$ become states
- Rows in $S$ and $SA$ become transitions

<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
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>

**Counterexample:**

$bba / 010$
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
### Angluin's alg. for Mealy machines

#### 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>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>

## Diagram

**States:**
- 00
- 3

**Transitions:**
- b/1
- b/0
- bb

**Sums:**
- b/1
- b/0
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>

- **Inconsistency**: Equal rows in upper part have 'different extensions'

http://connect-forever.eu/
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 $b$
  - $\varepsilon$ 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>
</tbody>
</table>

- **Inconsistencies** lead to new columns

**New Conjecture**

![Diagram with states 001, 110, 111, and 000 connected by transitions](http://connect-forever.eu/)
Angluin's alg. for Mealy machines
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**
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 – \(\text{á la Nerode}\).
Myhill–Nerode

Nerode relation:
- For language $L$ define relation $R_L$ (for $u, u' \in \Sigma^*$)
  
  \[ u \mathrel{R_L} u' \iff \text{for all } v \in \Sigma^*: \ (uv \in L \iff uv \in L) \]

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.
Further Developments
Practical challenges

Interface description etc.

interfacing real systems:
- alphabet generation
- abstraction
- data

equivalence queries

Behavioral models

reset

membership queries
Learning setup in Practice

Static Alphabet Abstraction

<presence type=… /> → TestDriver → Available → LearnLib

<iq type=“result“ /> ← Available ← OK
Learning relative to a given Representation System
Counter Examples and Witnesses

\[ \gamma(\alpha(c_1)) \rightarrow \gamma(\alpha(c_2)) \rightarrow \gamma(\alpha(c_3)) \rightarrow \gamma(\alpha(c_4)) \rightarrow \gamma(\alpha(c_5)) \rightarrow \gamma(\alpha(c_6)) \]
Counter Examples and Witnesses

\[ \gamma(\alpha(c_1)) \rightarrow \gamma(\alpha(c_2)) \rightarrow \gamma(\alpha(c_3)) \rightarrow \gamma(\alpha(c_4)) \rightarrow c_5 \rightarrow c_6 \rightarrow \text{(Red Node)} \]
Counter Examples and Witnesses

\[ \gamma(\alpha(c_1)) \rightarrow \gamma(\alpha(c_2)) \rightarrow \gamma(\alpha(c_3)) \rightarrow \gamma(\alpha(c_4)) \rightarrow \delta \]

\[ p \rightarrow c_4 \rightarrow \delta \]

Separating Pattern

- **p**: state
- **c_4**: representation
- **d**: future
\[ \Sigma_C \setminus \alpha_{old}(c) \]

\[ \gamma(\alpha(p)) \ x \ d \in F \iff \gamma(\alpha(p)) \ c \ d \in F \]

\[ \alpha_{old}(c) \]

\[ \gamma_{old}(\alpha_{old}(c)) \]
Case Study

Biometric Passport
[Aarts et. al, 2010]

262 Concrete symbols, 256 x readFile(i).

- 1 initial abstract symbols
- 8 alphabet refinements, to split readFile
- 9 final abstract symbols

\textit{\textbf{read file(i)}} aggregated according to the required Authentication
Further Developments

Learning Side-Effects
Register automata (RA)

- Action with formal parameters
- Parallel assignment

Register (X):
- $x_1$: username
- $x_2$: password

Diagram:
- $\text{register}(p_1,p_2) \mid true$
- $x_1 := p_1; x_2 := p_2$
- $\text{logout}() \mid true$
- $\text{pw}(p_1) \mid true$
- $x_2 := p_1$
- $\text{login}(p_1,p_2) \mid p_1 = x_1 \wedge p_2 = x_2$

http://connect-forever.eu/
### Experimental results

<table>
<thead>
<tr>
<th>Setup</th>
<th># Loc.</th>
<th># Trans.</th>
<th>MQs</th>
<th>EQs</th>
</tr>
</thead>
<tbody>
<tr>
<td>RA learning algorithm</td>
<td>3</td>
<td>16</td>
<td>329</td>
<td>3</td>
</tr>
<tr>
<td>$L^*$, symmetry reduction, $</td>
<td>D</td>
<td>= 6)$</td>
<td>73</td>
<td>5,913</td>
</tr>
<tr>
<td>$L^*$, no optimization, $</td>
<td>D</td>
<td>= 6)$</td>
<td>73</td>
<td>5,913</td>
</tr>
</tbody>
</table>
Overview

- Motivation
- Introduction to active automata learning
- Practical aspects in active automata learning
- Conclusions
Learning non-functional properties

- New test-driver architecture
- Allows learning of non-functional properties
- Bundling of queries allows application of performance testing methods
Non-functional properties: First results

Response Time per Primitive (intervaled)

- Bid
- InterruptUpload
- SubmitReport
- InterruptBidding
- SubmitSubmission
- InterruptReview
- AssignReviewer
- DownloadDocument
- UploadDocument
- CreateConf
- SubmitPaper

Execution Intervals

Response time [ms]

- 0
- 1-100
- 101-200
- 201-300
- 301-400
- 401-500

- 0
- 1.2K
- 2.4K
- 3.6K
- 4.8K
- 6K
- 7.2K
- 8K
- 9.6K
- 12K

http://connect-forever.eu/

7th Framework Programme
Profiling example: creating conferences

[before]

[after]
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
Semantics of "phase expires"-edges (2)
Many participants
Putting it all together

OCS

Workflows

http://connect-forever.eu/
Regular extrapolation
Optimized learning setup

L^*_M

SubmitPaper, Interrupt Submission, UploadDocument

login(submitter)
paper = submit(title, ...)
logout()  
login(pchair)
interrupt(submission)
logout()  
login(submitter)
uploadDoc(paper, doc)
logout()  

Cache Limit Condition Prefix-closure Reuse Test-Driver OCS
Conclusions

Active Automata Learning in 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 with
  - Functional and Non-Function Properties

**Behavioural perspective to control system evolution**

http://connect-forever.eu/