A practical introduction to active automata learning

Bernhard Steffen, Falk Howar, Maik Merten
TU Dortmund

SFM2011

Overview

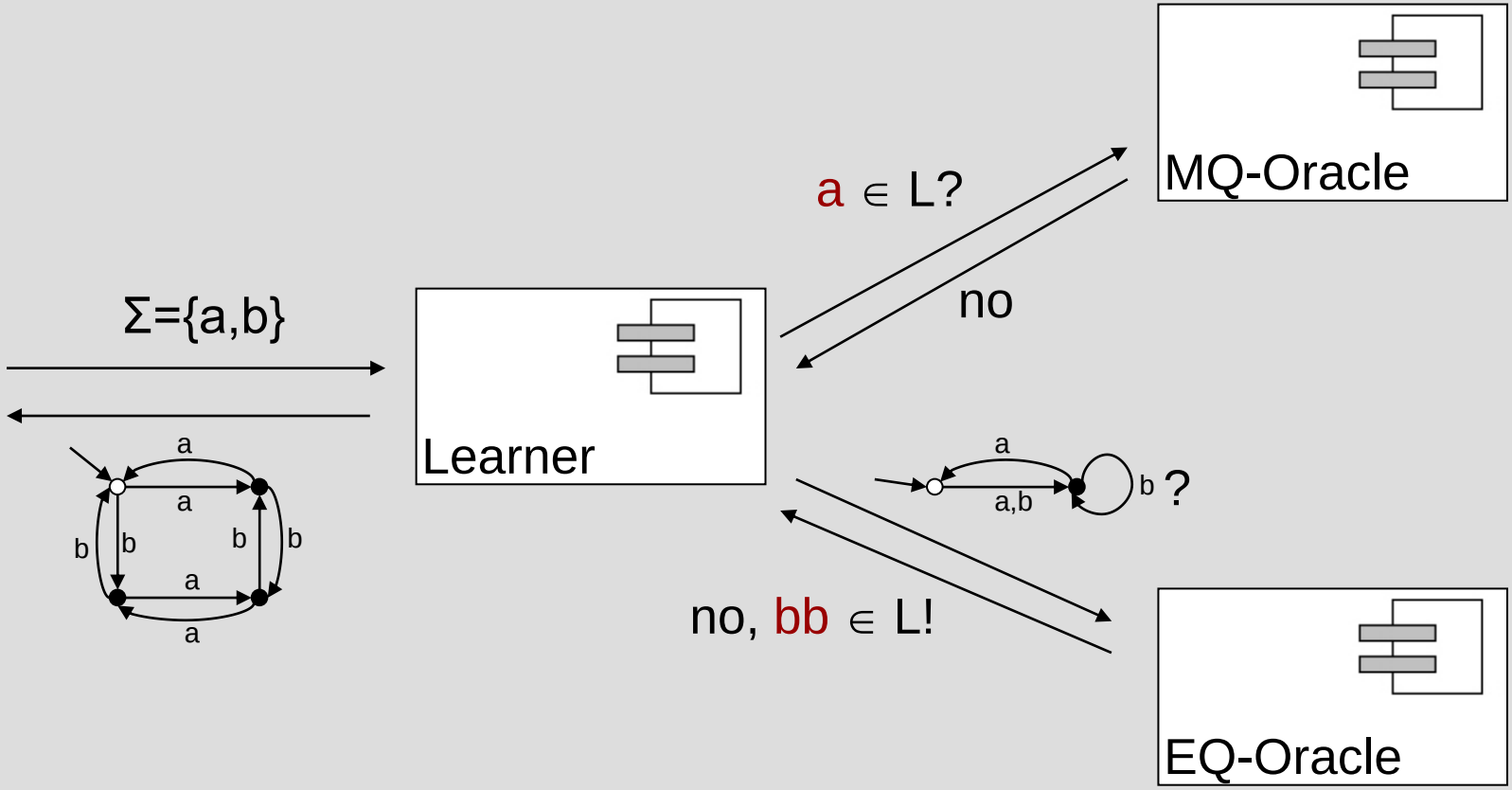
- Motivation
- Introduction to **active automata learning**
- **Practical aspects** in active automata learning
- Conclusions

Mealy machines

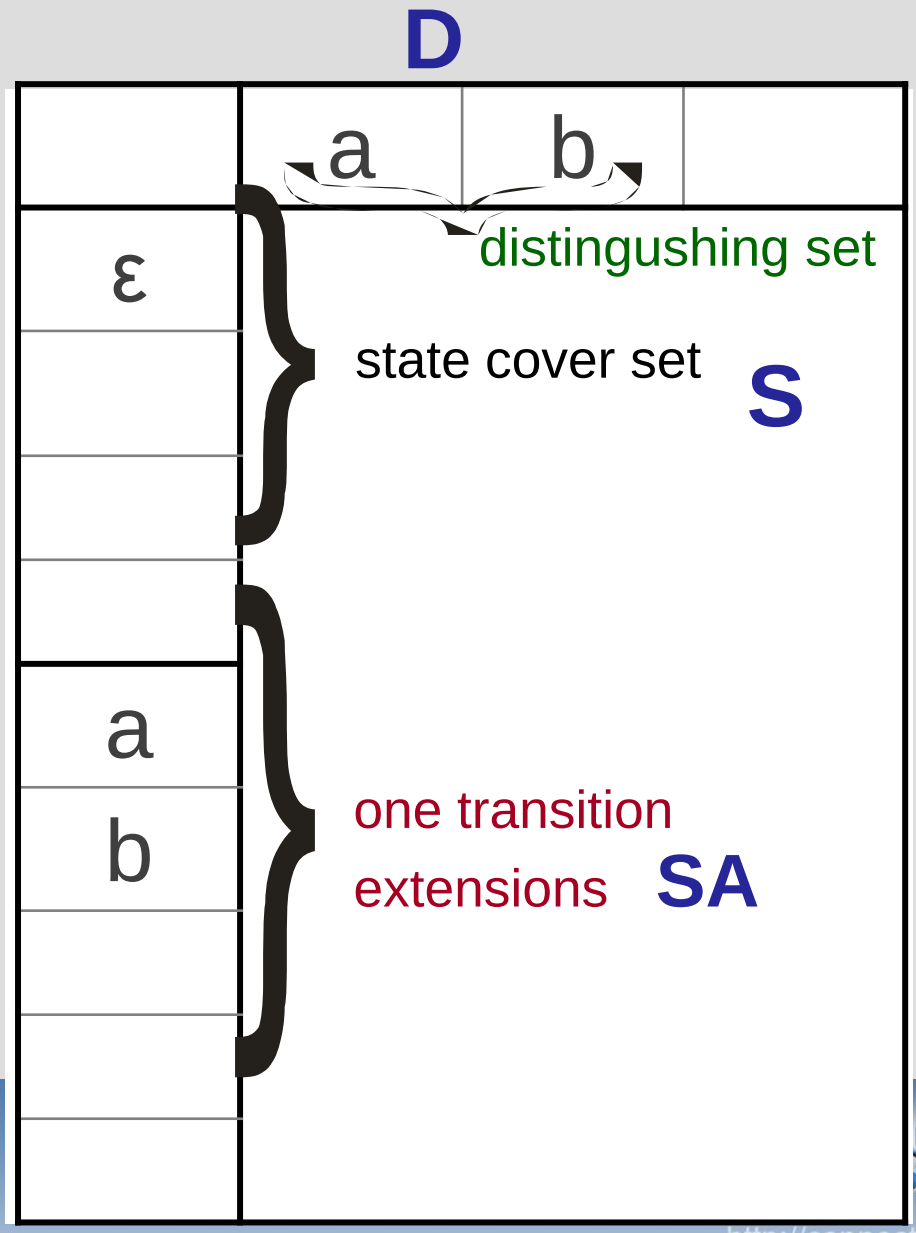
- Mealy machine $M=(S,\Sigma,\Gamma,\sigma,\gamma)$
  - S finite set of states
  - Σ finite input-alphabet
  - **Γ finite output-alphabet**
  - σ: $(S \times \Sigma) \rightarrow S$ transition-function
  - **γ: $(S \times \Sigma) \rightarrow \Gamma$ output-function**

- Words Σ^* for ($s \in S$, $a \in \Sigma$, $w \in \Sigma^*$)
  - $\sigma:(S \times \Sigma^*) \rightarrow S$, $\sigma(s,\varepsilon)=s$, $\sigma(s,aw) = \sigma(\sigma(s,a),w)$
  - γ: $(S \times \Sigma^*) \rightarrow \Gamma^*$, $\gamma(s,\varepsilon)=\varepsilon$, $\gamma(s,aw) = \gamma(s,a).\gamma(\sigma(s,a),w)$

Active automata learning



Angluin's alg. for Mealy machines

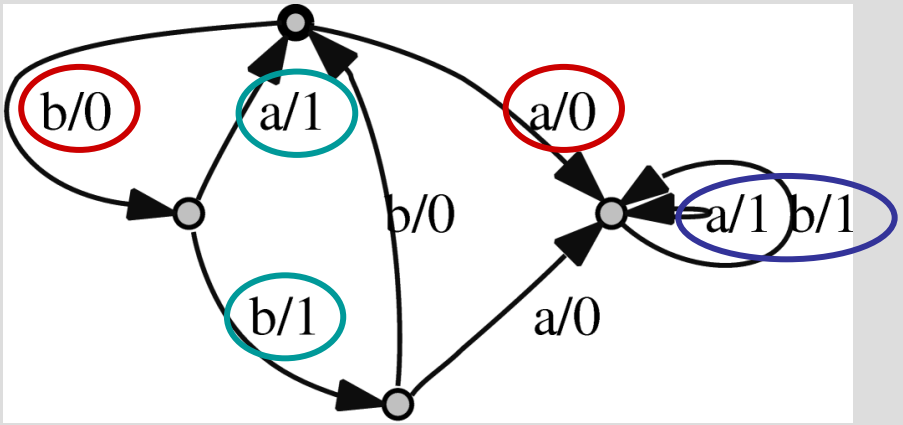


- Initialize **Distinguishing Set** **D** with alphabet of inputs

Angluin's alg. for Mealy machines

	a	b	
ε	0	0	
a	1	1	
b	1	1	

Unknown system:



Angluin's alg. for Mealy machines

	a	b	
ε	0	0	
a	1	1	
b	1	1	

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

Angluin's alg. for Mealy machines

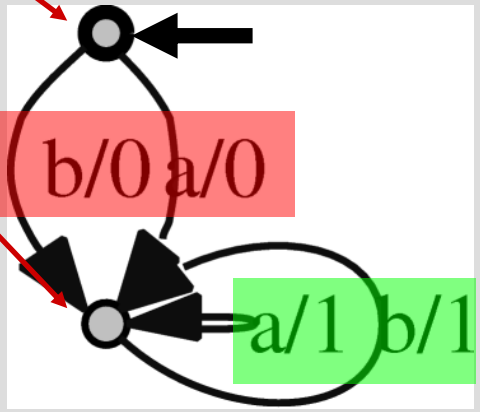
	a	b	
ε	0	0	
a	1	1	
b	1	1	
aa	1	1	
ab	1	1	

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

Angluin's alg. for Mealy machines

	a	b	
ε	0	0	
a	1	1	
b	1	1	
aa	1	1	
ab	1	1	

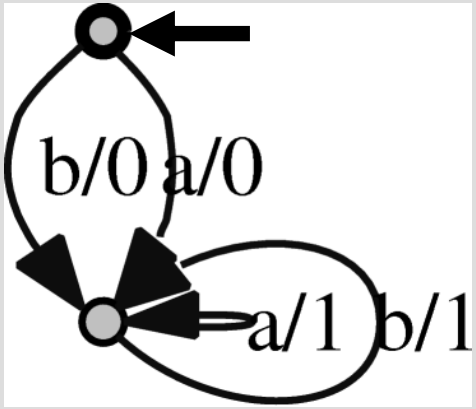
- **Conjecture**:
- Unique rows in **S** become states
- Rows in **S** and **SA** become transitions



Angluin's alg. for Mealy machines

	a	b	
ε	0	0	
a	1	1	
b	1	1	
aa	1	1	
ab	1	1	

- **Counterexample**: bbb / 010



Angluin's alg. for Mealy machines

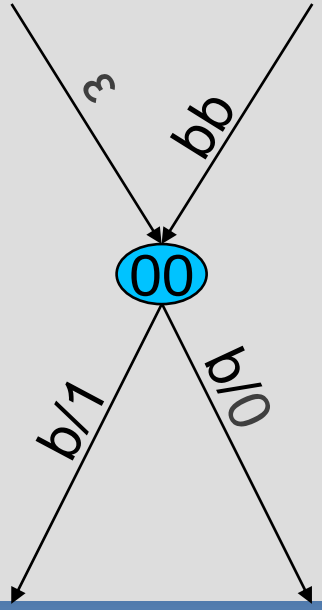
	a	b	
ε	0	0	
a	1	1	
b	1	1	
bb	0	0	
bbb	0	0	
aa	1	1	
ab	1	1	
ba	0	0	
…	…	…	

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

Angluin's alg. for Mealy machines

	a	b
ε	0	0
a	1	1
b	1	1
bb	0	0
bbb	0	0
aa	1	1
ab	1	1
ba	0	0
…	…	…

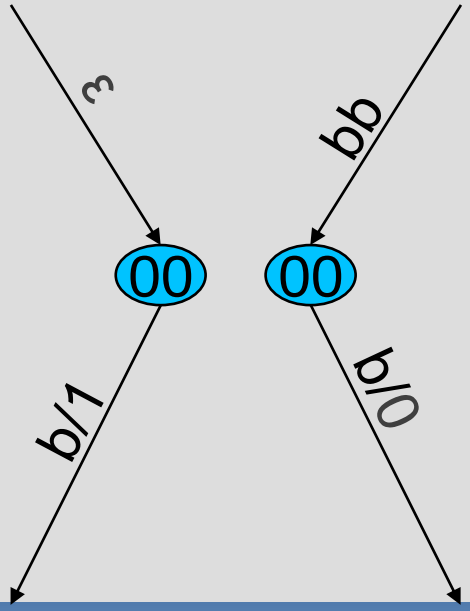
- **Inconsistency**:
- Equal rows in upper part have ‚different extensions‘



Angluin's alg. for Mealy machines

	a	b
ε	0	0
a	1	1
b	1	1
bb	0	0
bbb	0	0
aa	1	1
ab	1	1
ba	0	0
…	…	…

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



Angluin's alg. for Mealy machines

	a	**b**
ε	0	**0**
a	1	1
b	1	**1**
bb	0	**0**
bbb	0	**0**
aa	1	1
ab	1	1
ba	0	0
…	…	…

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

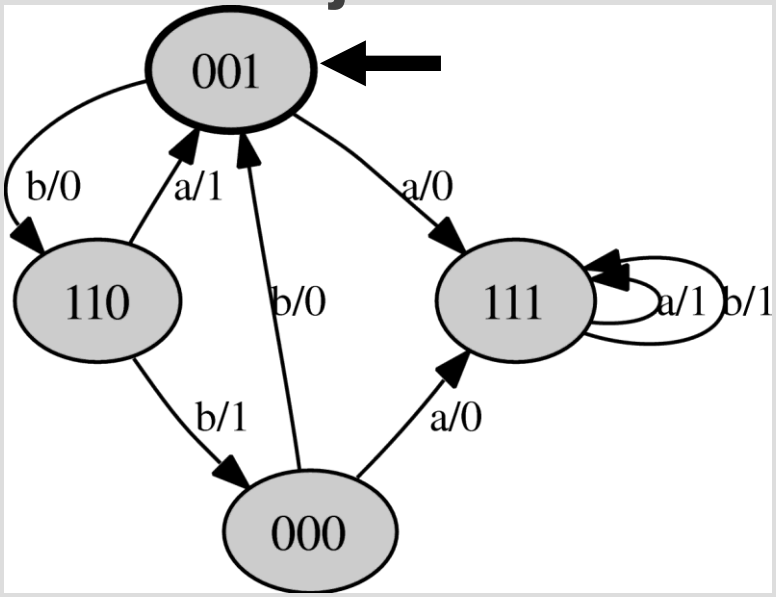
=> **ε** and **bb** will differ for suffix **bb**

Angluin's alg. for Mealy machines

	a	**b**	bb
ε	0	0	**1**
a	1	1	1
b	1	1	0
bb	0	0	**0**
bbb	0	0	1
aa	1	1	1
ab	1	1	1
ba	0	0	1
…	…	…	…

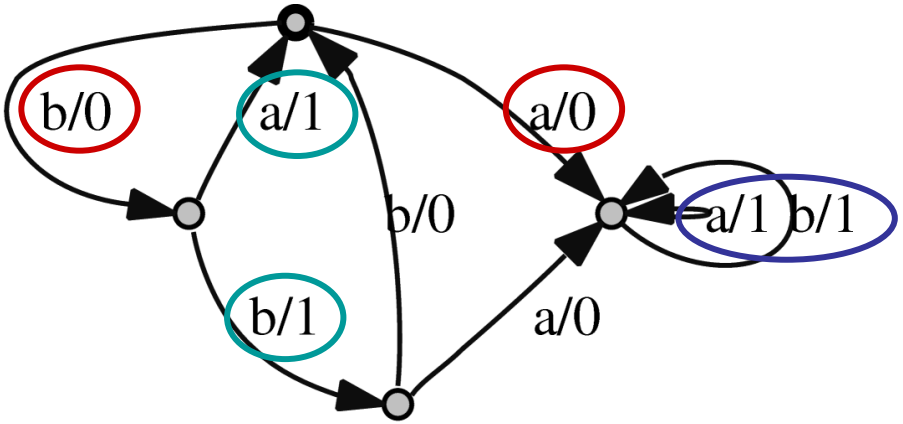
- **Inconsistencies** lead to new columns

New Conjecture

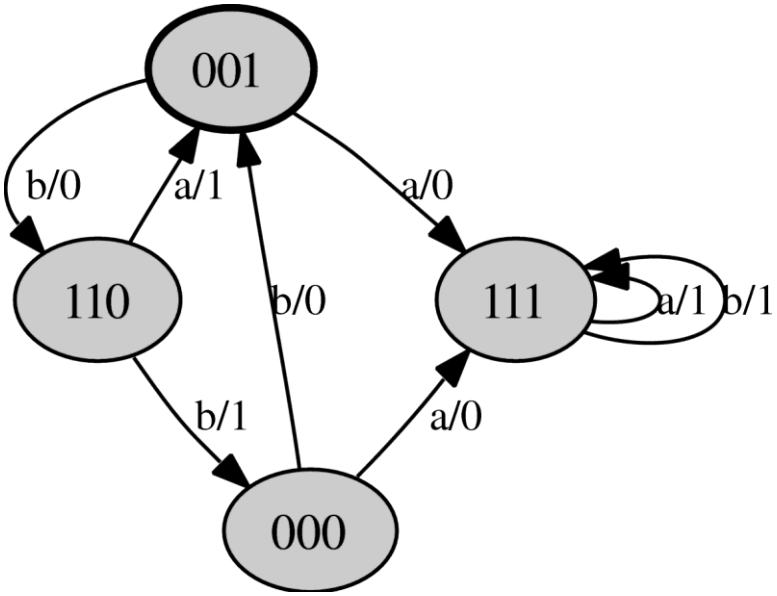


Angluin's alg. for Mealy machines

Target System



Learned System



Summarized Observations (1)

- Systematic completition 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**

Summarized Observations (2)

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 – **á la Nerode**).

Myhill–Nerode

Nerode relation:

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

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

Myhill-Nerode Theorem:

- Minimal number of states of an accepting deterministic automaton equals the number of equivalence classes of R_L

Summarized Observations (4)

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

```
<xs:complexType name="Login">
 <xs:sequence>
  <xs:element minOccurs="0" name="
  <xs:element minOccurs="0" name="
 </xs:sequence>
</xs:complexType>
<xs:complexType name="BeginTransac
 <xs:sequence>
  <xs:element minOccurs="0" name="
  <xs:element minOccurs="0" name="
  <xs:element minOccurs="0" name="
```

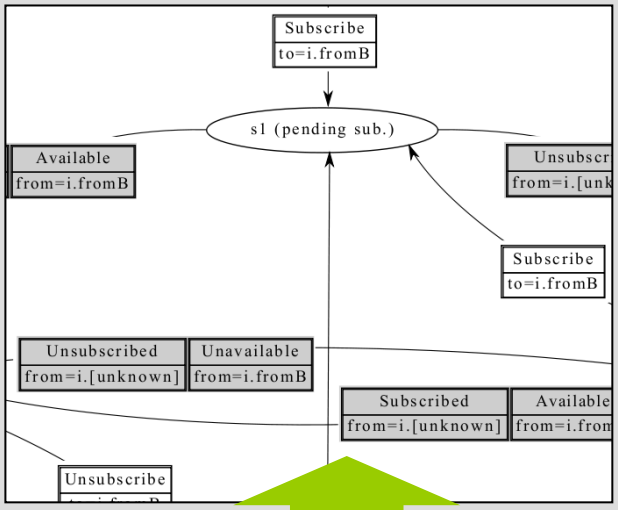
Interface description

etc.

interfacing real systems:
- alphabet generation
- abstraction
- data

equivalence queries

Behavioral models



<presence type=… />

<iq type= “result“ />

Test-driver

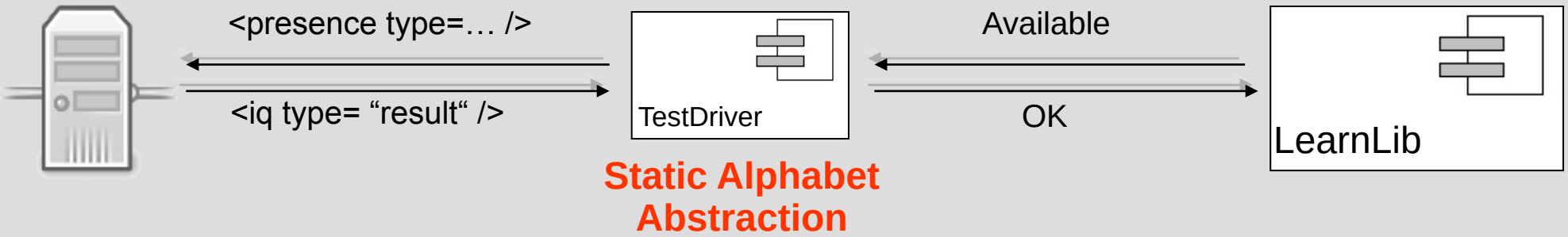
Available

OK

reset

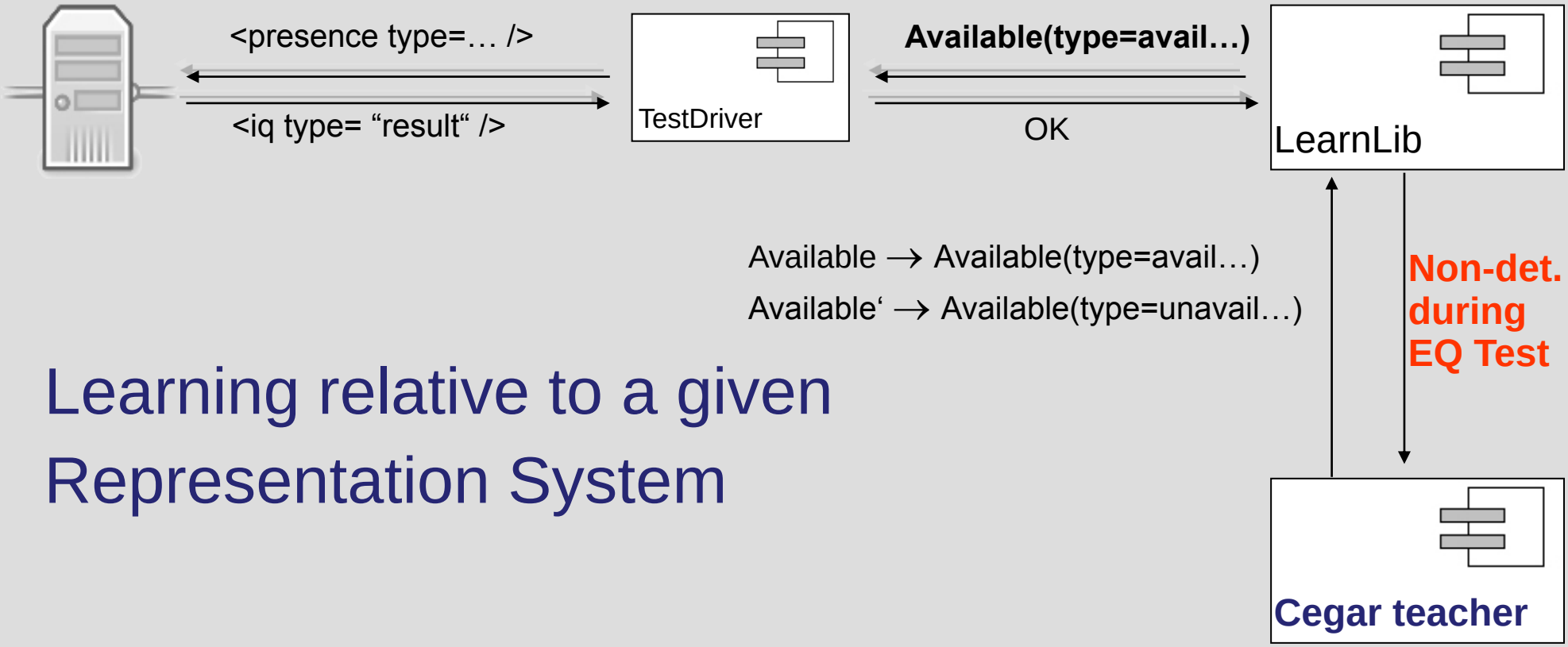
membership queries

Learning setup in Practice



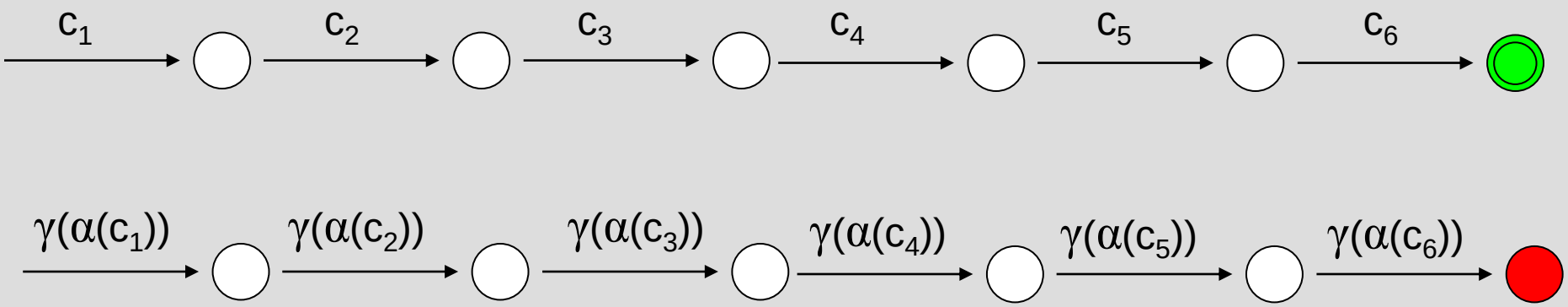
Static Alphabet Abstraction

A cegar teacher

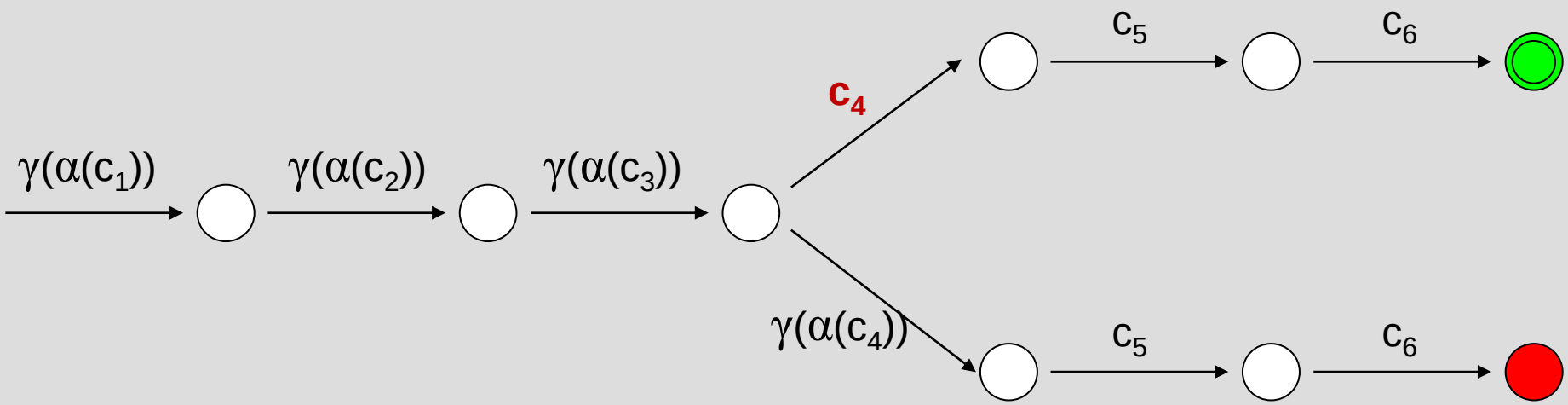


Learning relative to a given Representation System

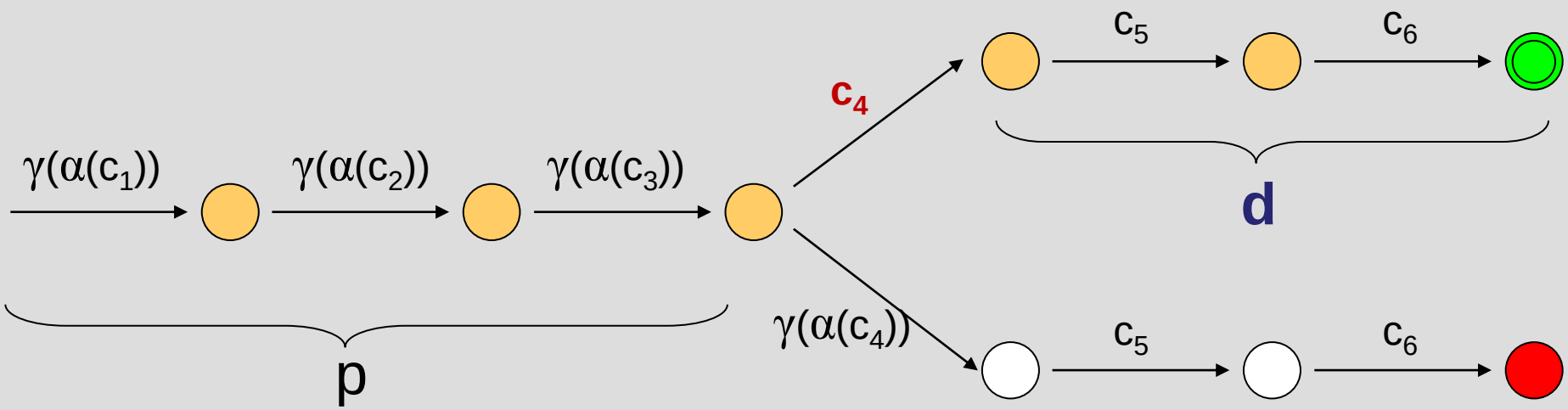
Counter Examples and Witnesses



Counter Examples and Witnesses



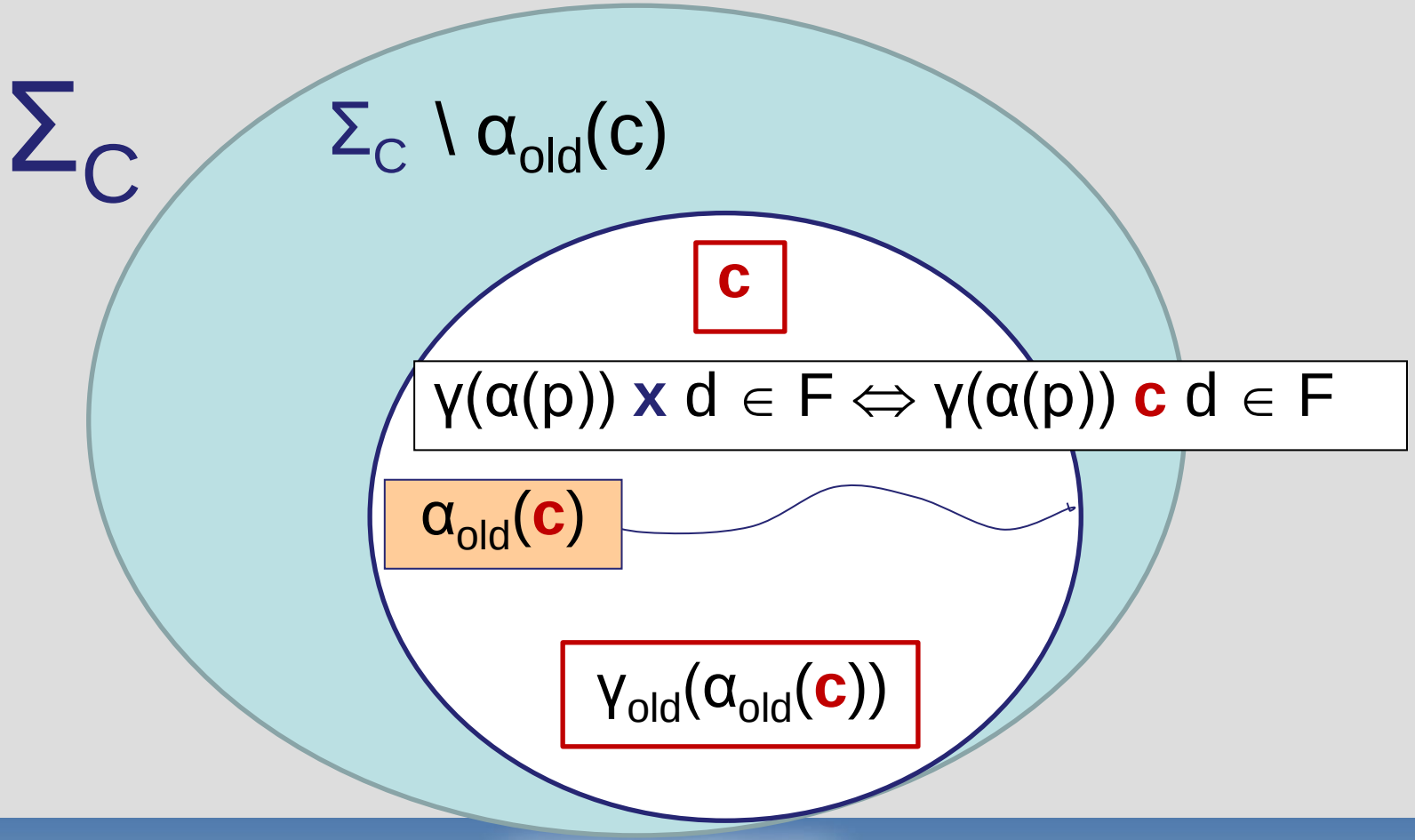
Counter Examples and Witnesses



Separating Pattern

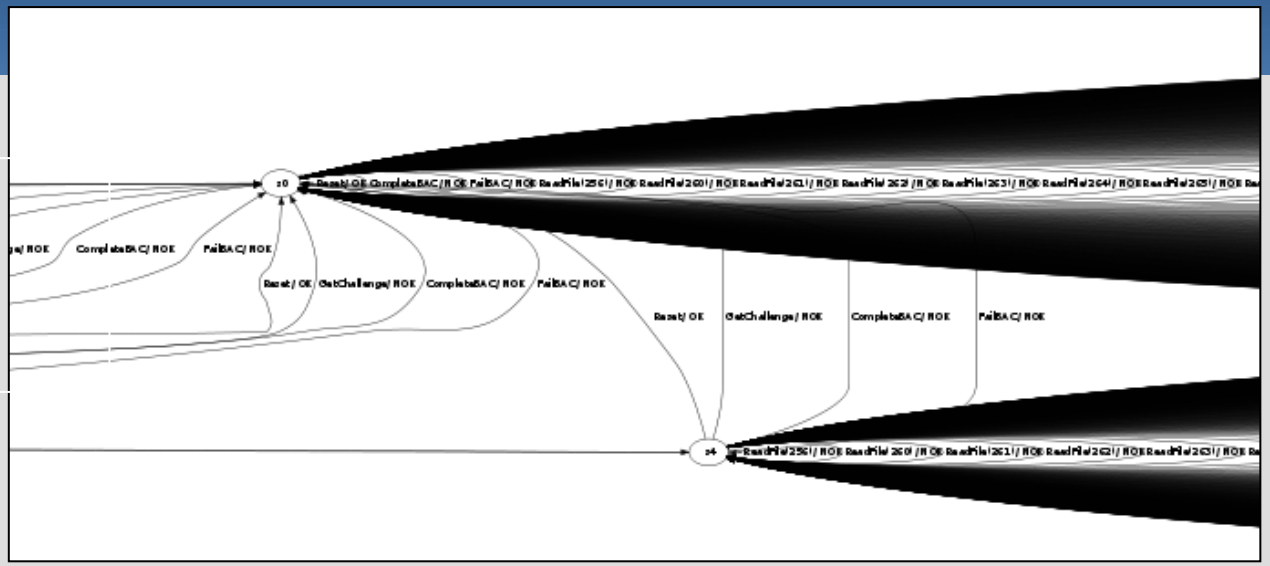
p	c_4	d
state	representation	future

Alphabet Abstraction Refinement

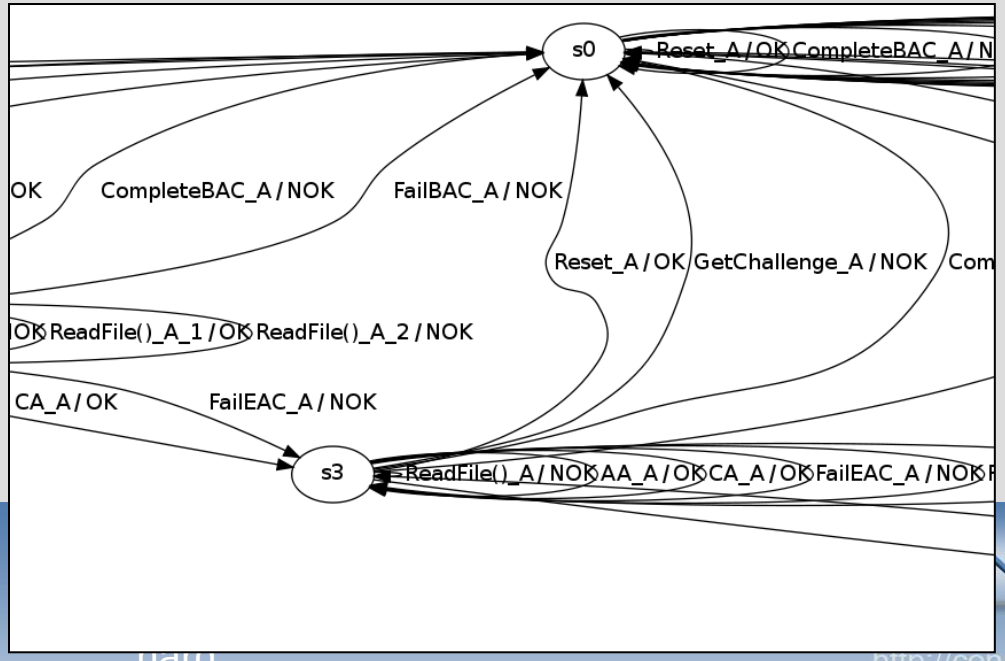


Case Study

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

‚read file(i)‘ *aggregated according to the required* ***Authentication***

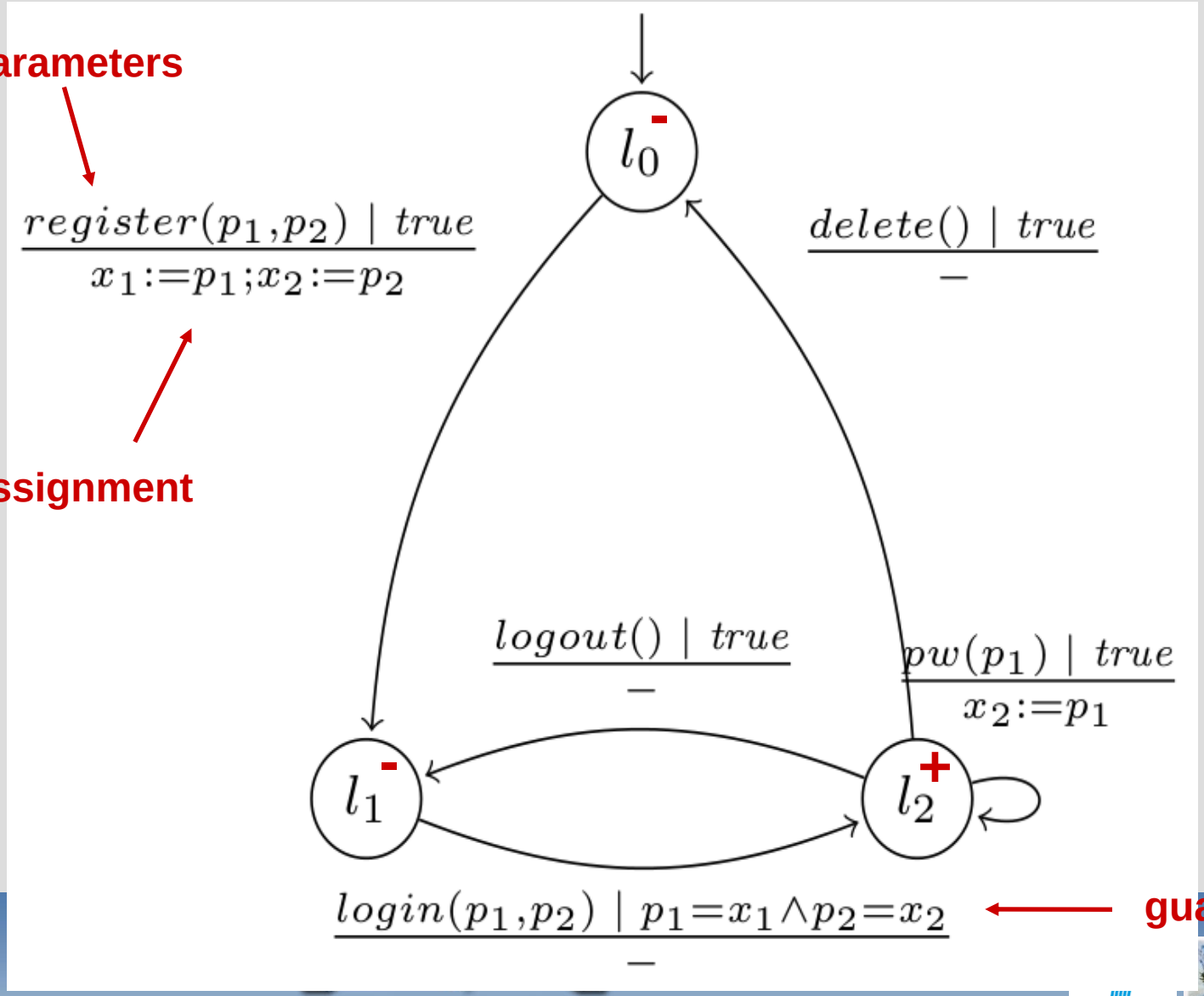
Further Developments

Learning Side-Effects

Register automata (RA)

action with formal parameters

parallel assignment

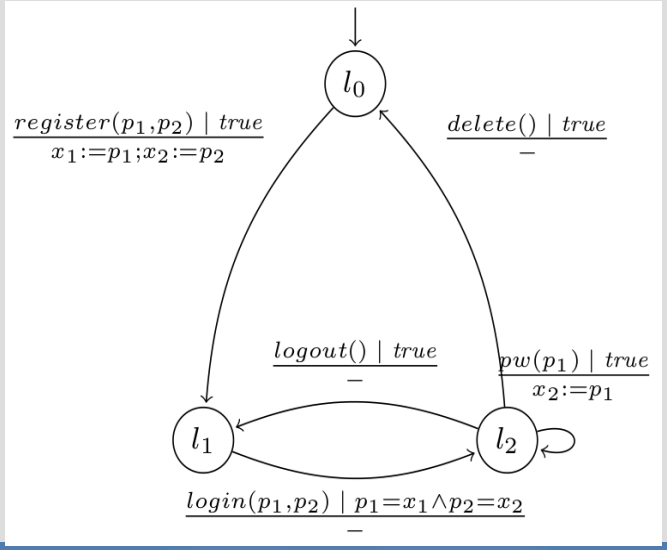


guard

Register (X):
x_1. username
x_2. password

Experimental results

Setup	# Loc.	# Trans.	MQs	EQs
RA learning algorithm	3	16	329	3
L^*, symmetry reduction, $\lvert D\rvert = 6$)	73	5,913	2,776	2
L^*, no optimization, $\lvert D\rvert = 6$)	73	5,913	415,333	72

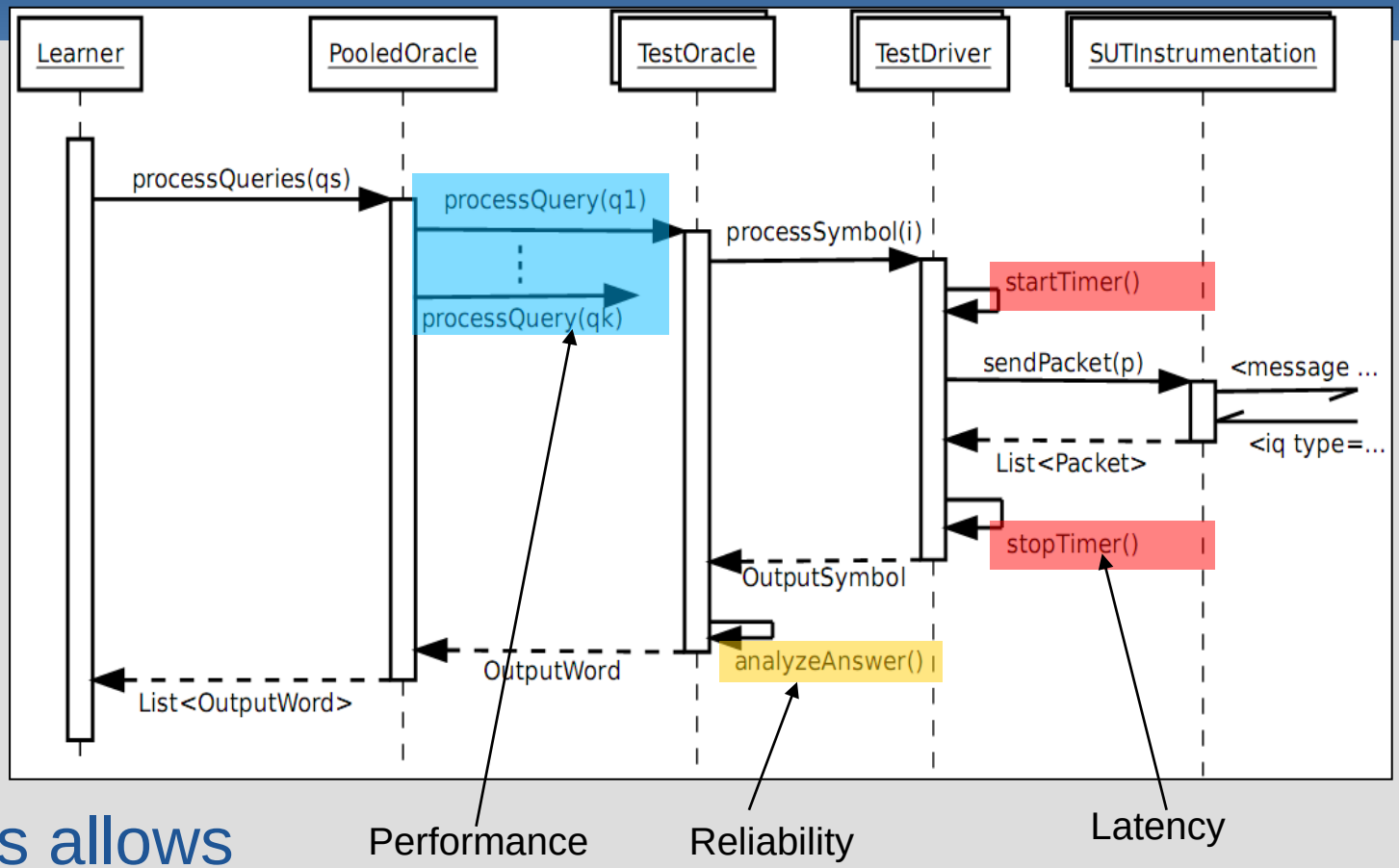


Overview

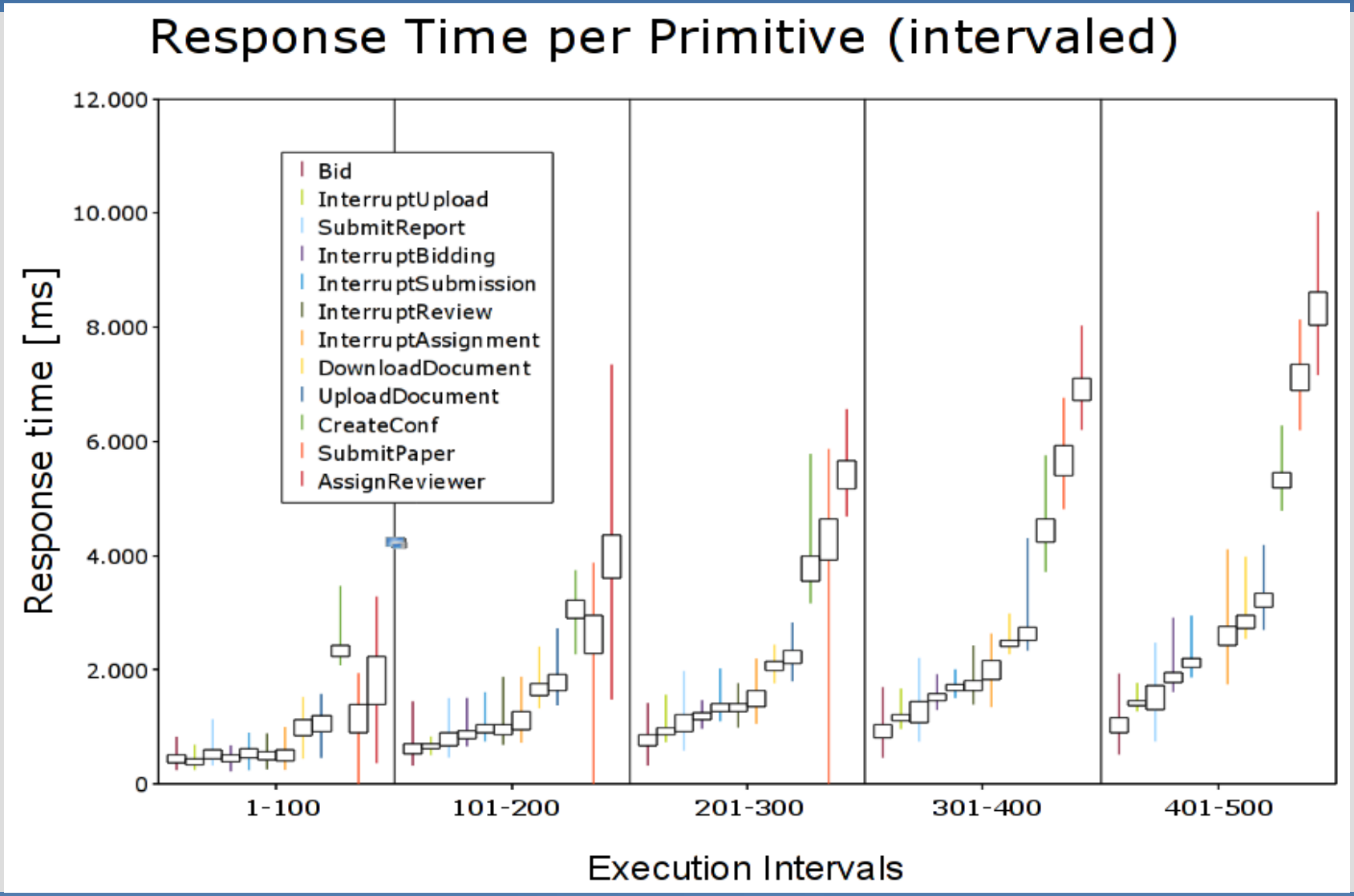
- Motivation
- Introduction to **active automata learning**
- **Practical aspects** in active automata learning
- Conclusions

Learning non-functional properties

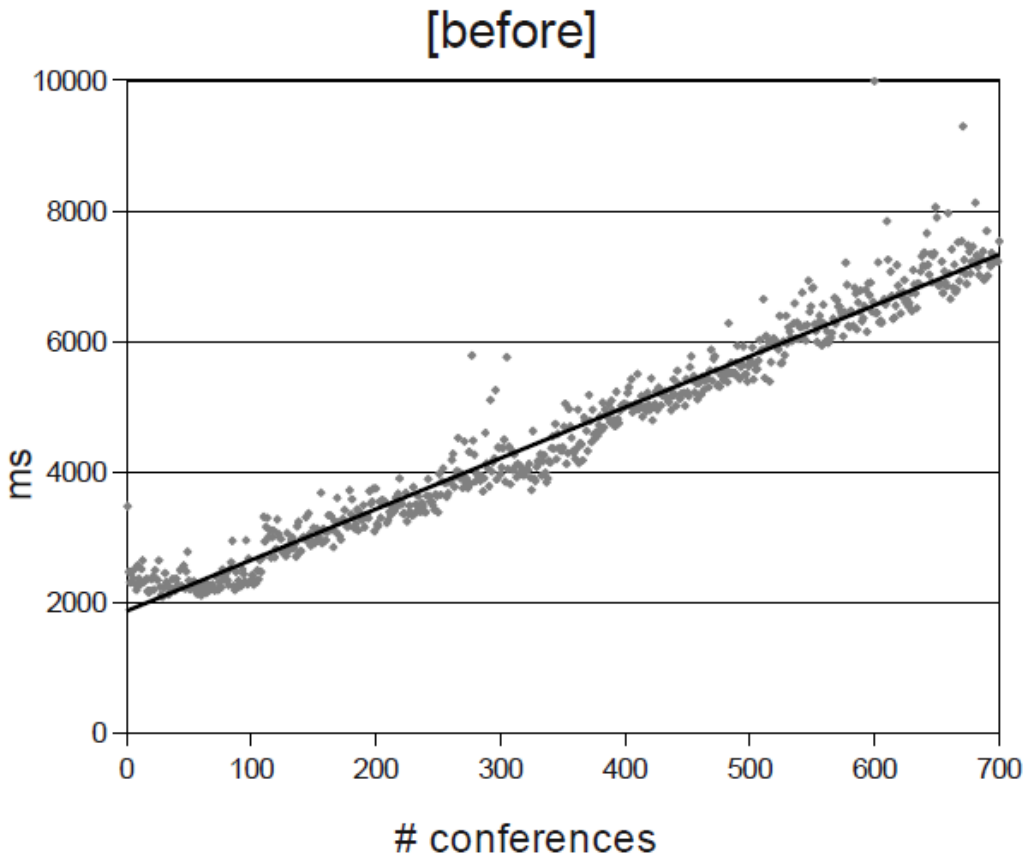
- New test-driver architecture
- Allows learning of non-functional properties
- Bundling of queries allows application of performance testing methods

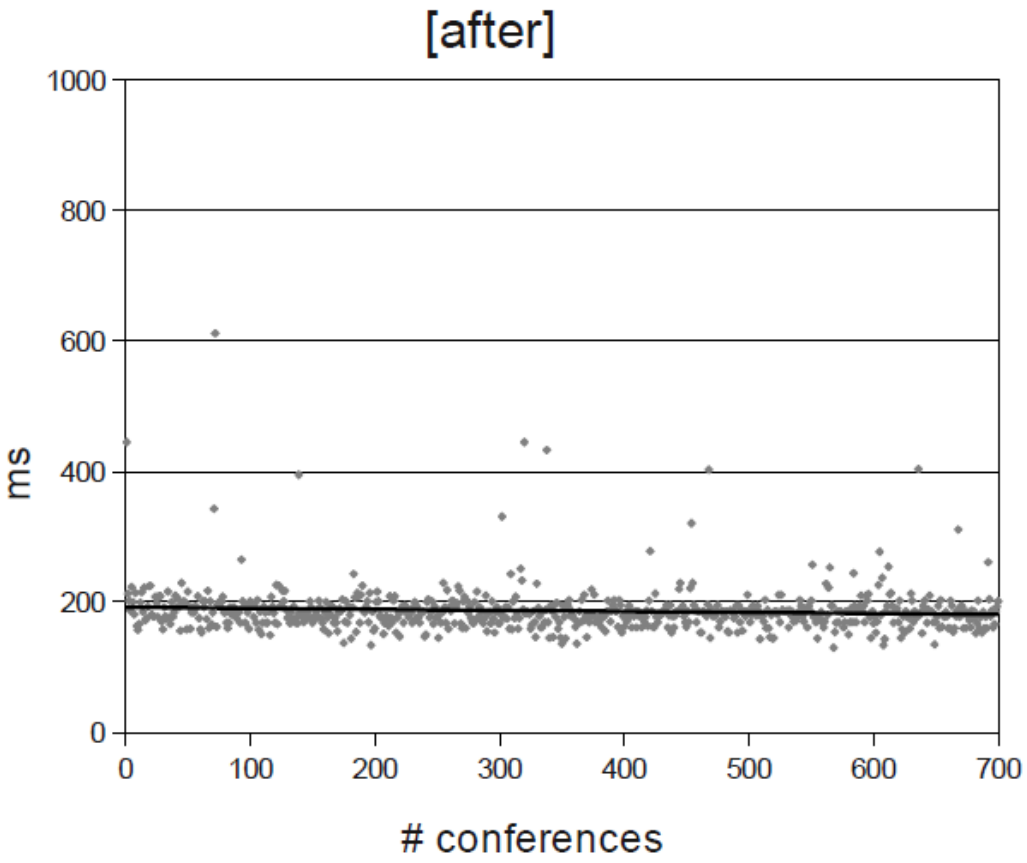


Non-functional properties: First results



Profiling example: creating conferences

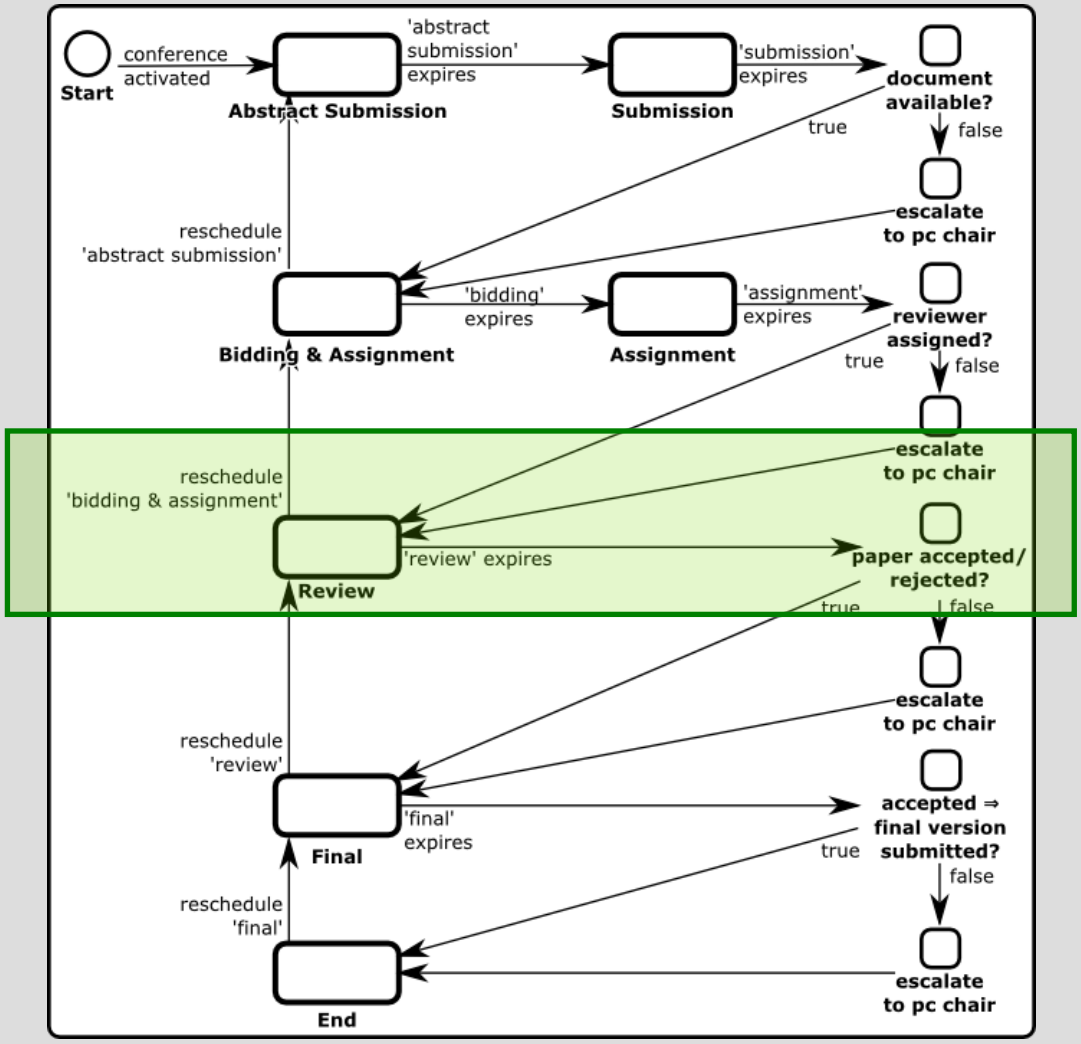


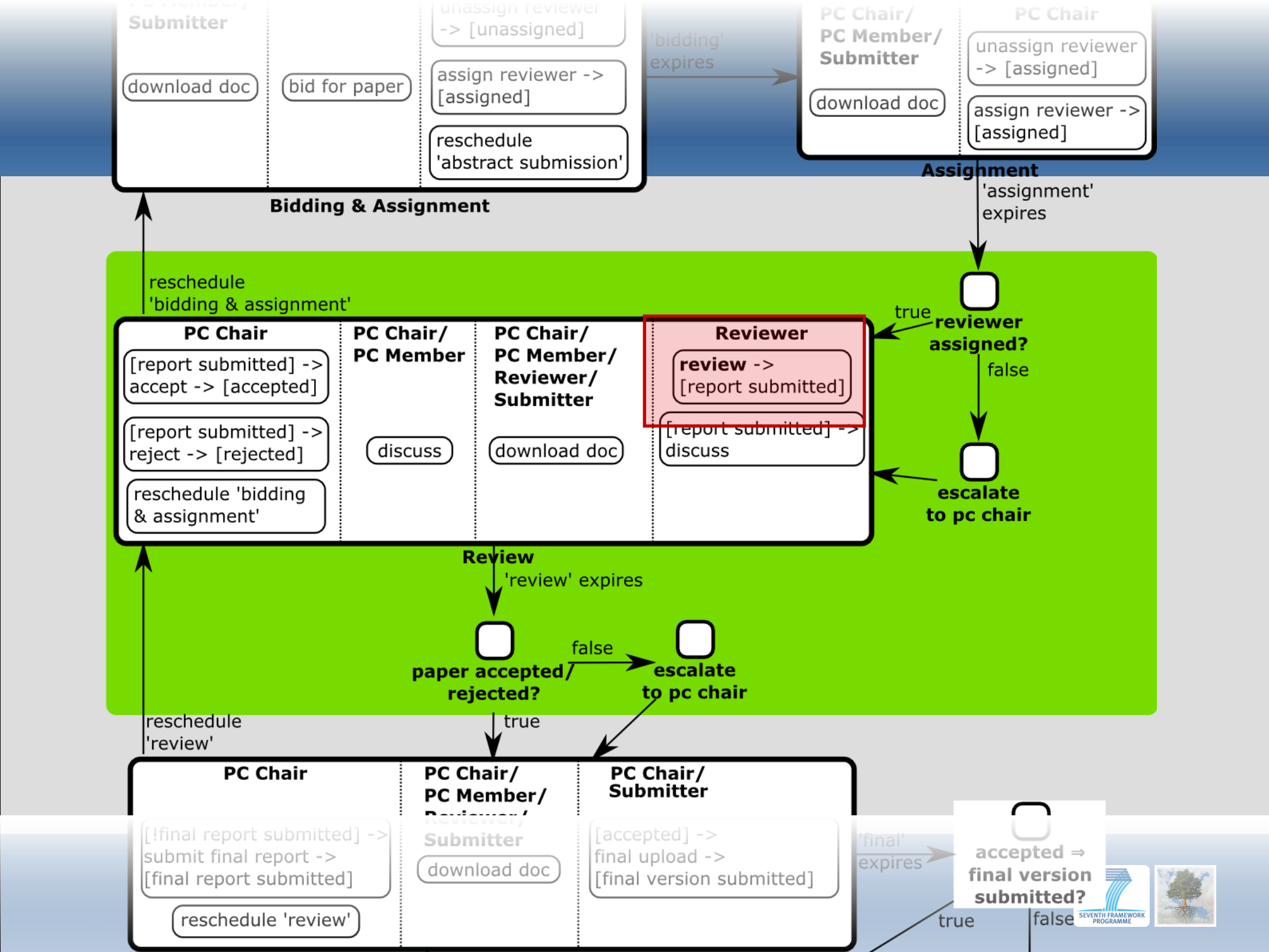


Practical results I

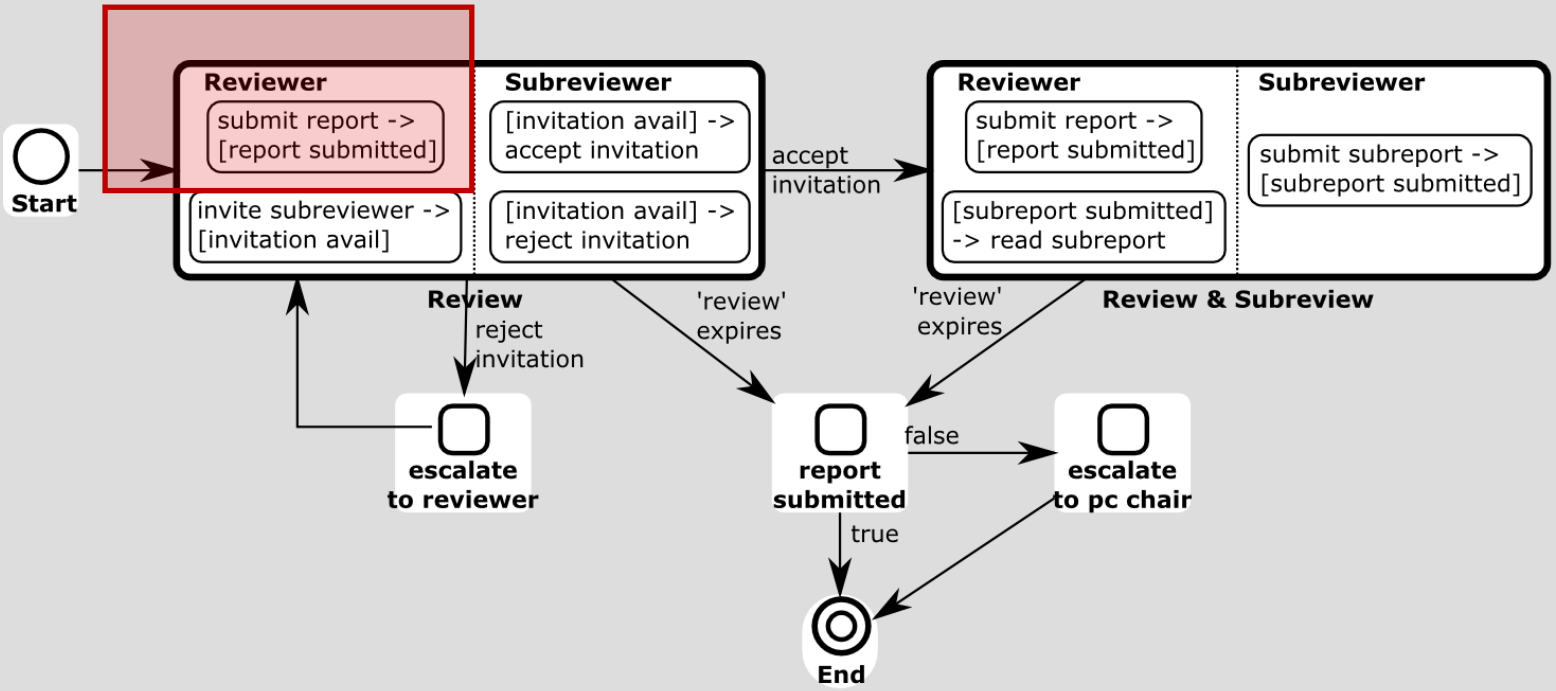
Learning the OCS

User model: paper workflow

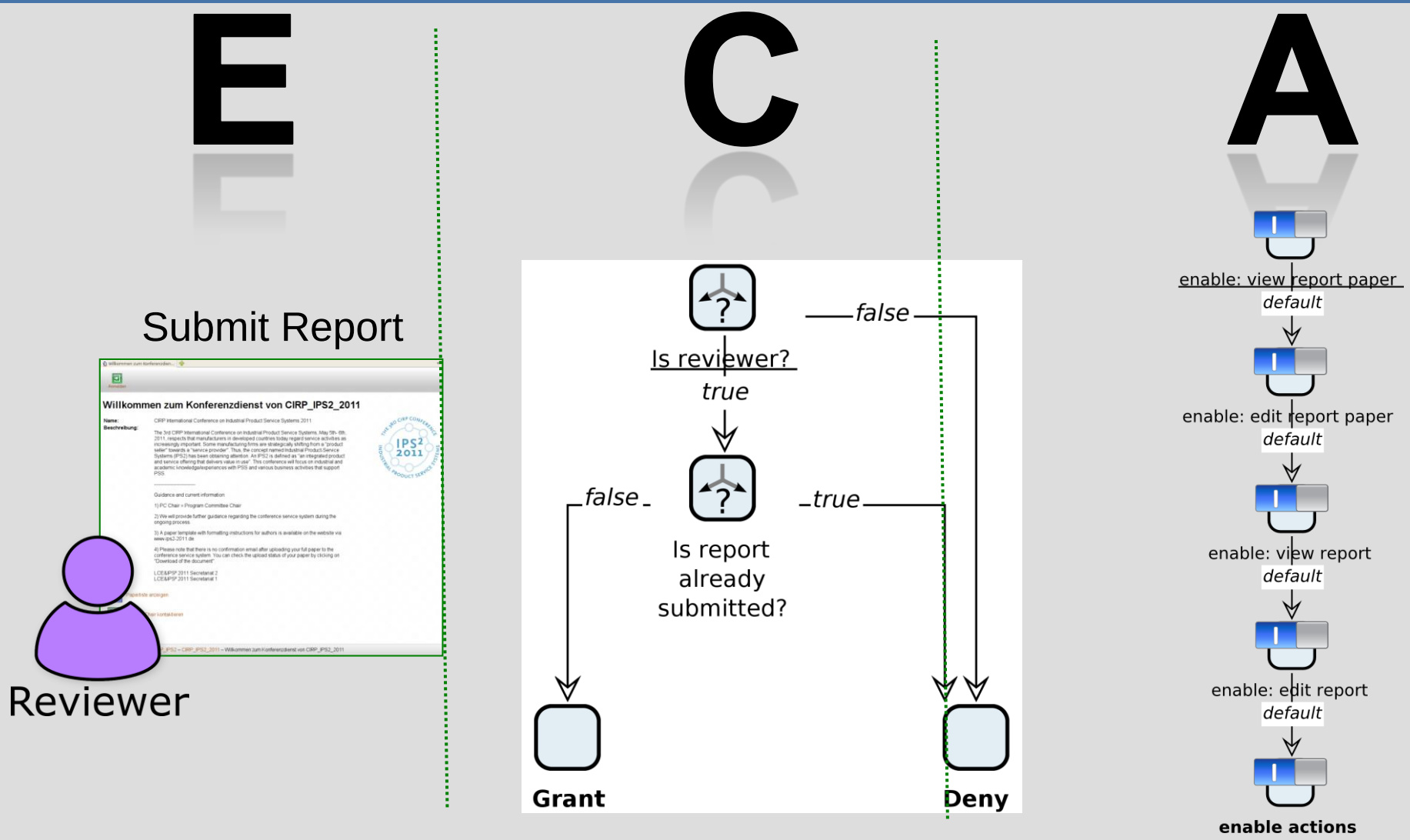



Submitter
download doc
bid for paper
unassign reviewer -> [unassigned]
assign reviewer -> [assigned]
reschedule 'abstract submission'
Bidding & Assignment
'bidding' expires
PC Chair/ PC Member/ Submitter
download doc
PC Chair
unassign reviewer -> [assigned]
assign reviewer -> [assigned]
Assignment
'assignment' expires
reschedule 'bidding & assignment'
true
reviewer assigned?
false
escalate to pc chair
PC Chair
[report submitted] -> accept -> [accepted]
[report submitted] -> reject -> [rejected]
reschedule 'bidding & assignment'
PC Chair/ PC Member
discuss
PC Chair/ PC Member/ Reviewer/ Submitter
download doc
Reviewer
review -> [report submitted]
[report submitted] -> discuss
Review
'review' expires
paper accepted/ rejected?
false
escalate to pc chair
true
reschedule 'review'
PC Chair
[!final report submitted] -> submit final report -> [final report submitted]
reschedule 'review'
PC Chair/ PC Member/ Reviewer/ Submitter
download doc
PC Chair/ Submitter
[accepted] -> final upload -> [final version submitted]
'final' expires
accepted ⇒ final version submitted?
true
false

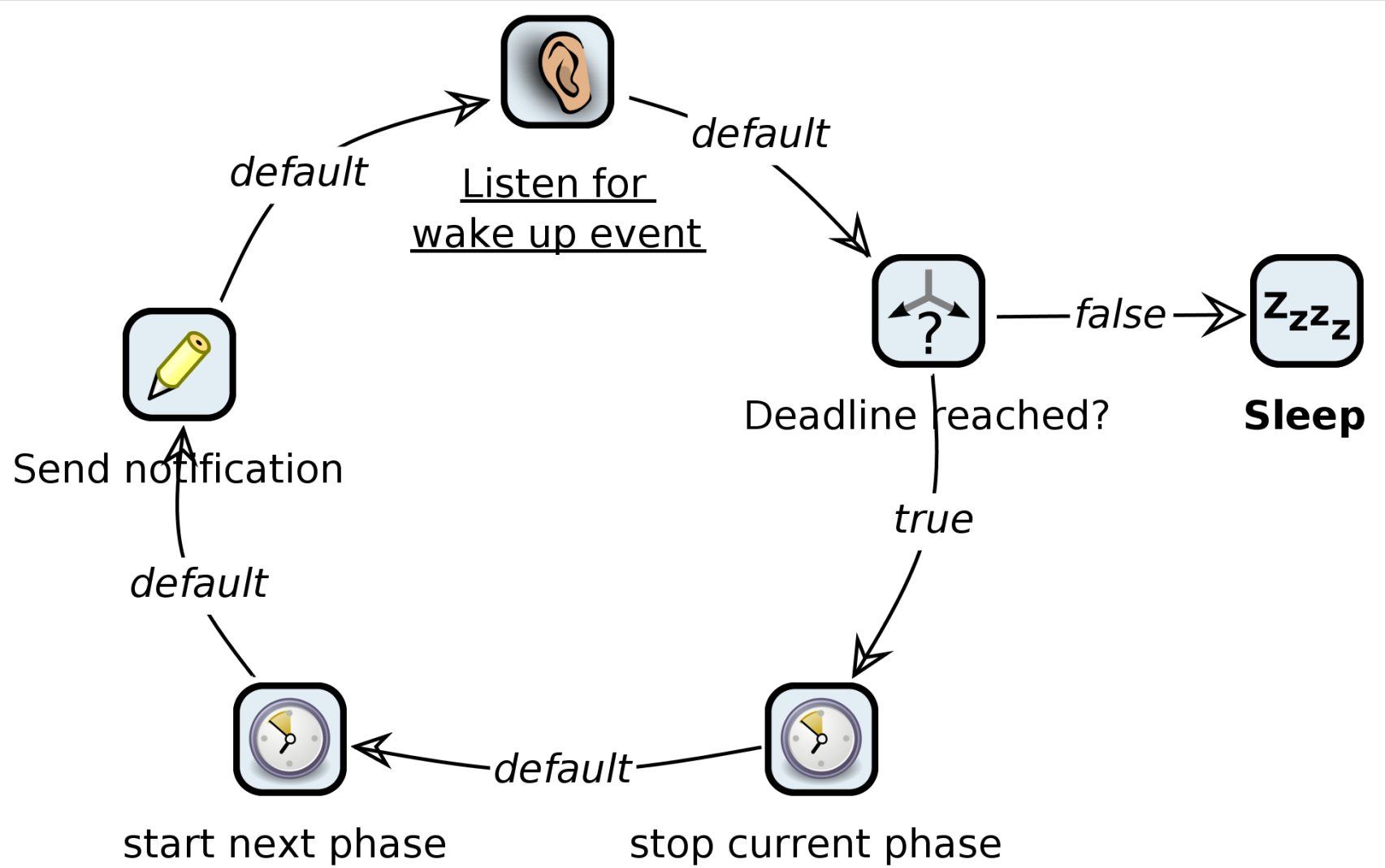
Hierarchy


Start
Reviewer
submit report -> [report submitted]
invite subreviewer -> [invitation avail]
Subreviewer
[invitation avail] -> accept invitation
[invitation avail] -> reject invitation
Review
accept invitation
Reviewer
submit report -> [report submitted]
[subreport submitted] -> read subreport
Subreviewer
submit subreport -> [subreport submitted]
Review & Subreview
reject invitation
'review' expires
'review' expires
escalate to reviewer
report submitted
false
escalate to pc chair
true
End

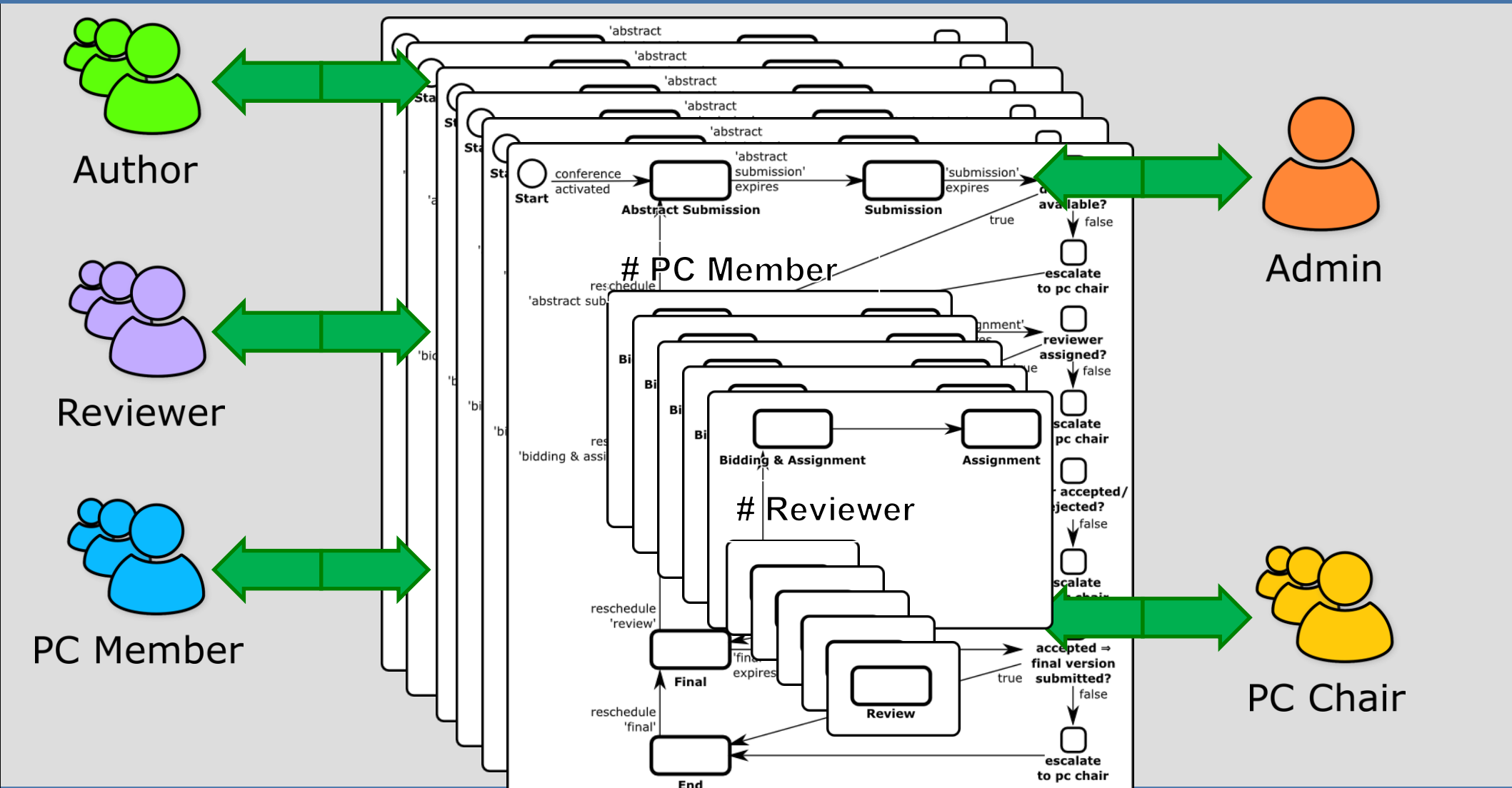
Event Condition Action



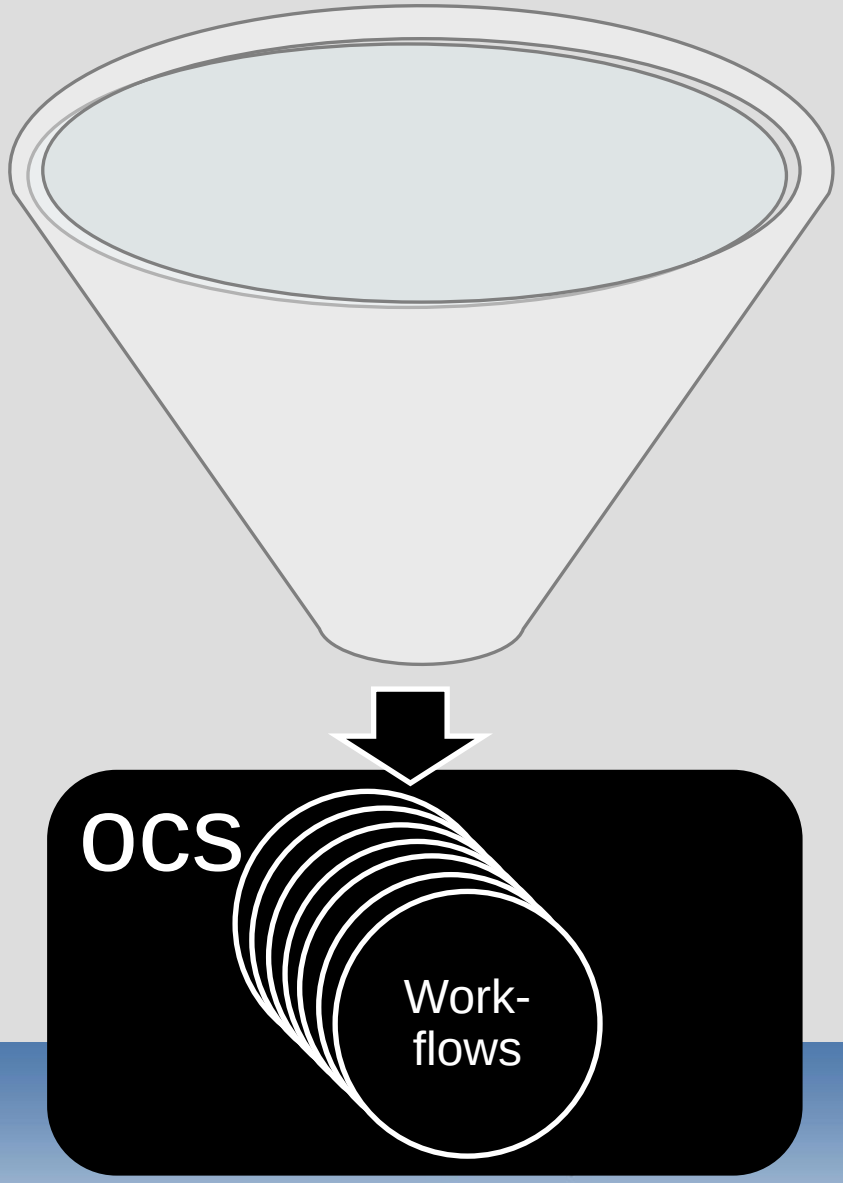
Semantics of "phase expires"-edges (2)



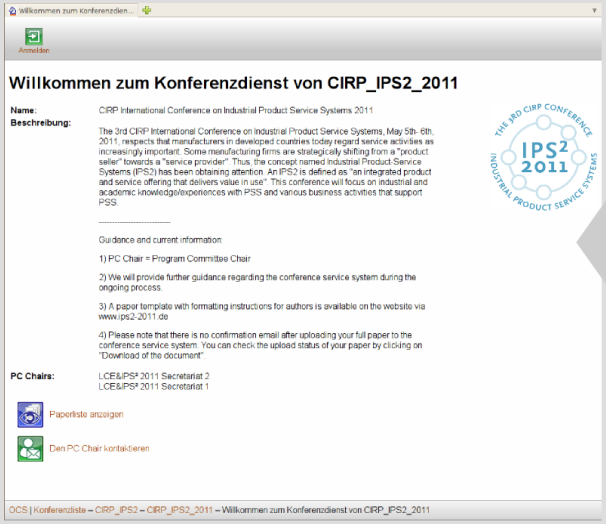
Many participants

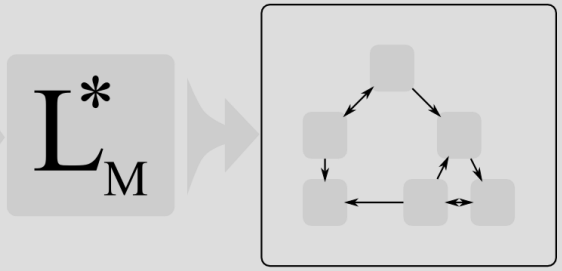


Putting it all together

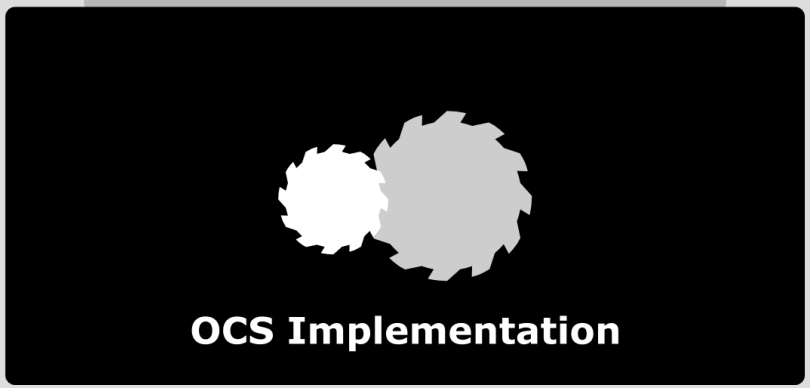


Regular extrapolation

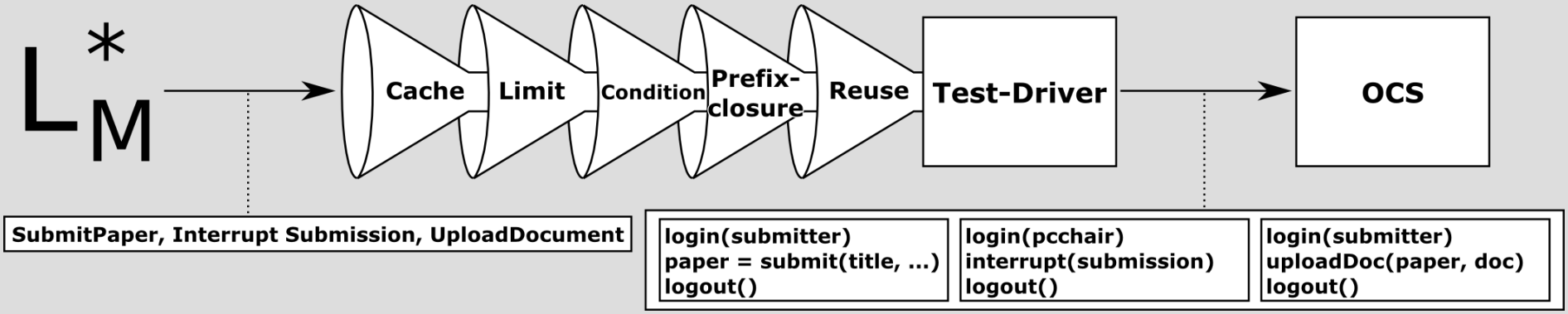




OCS API



Optimized learning setup



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