A practical introduction to active automata learning

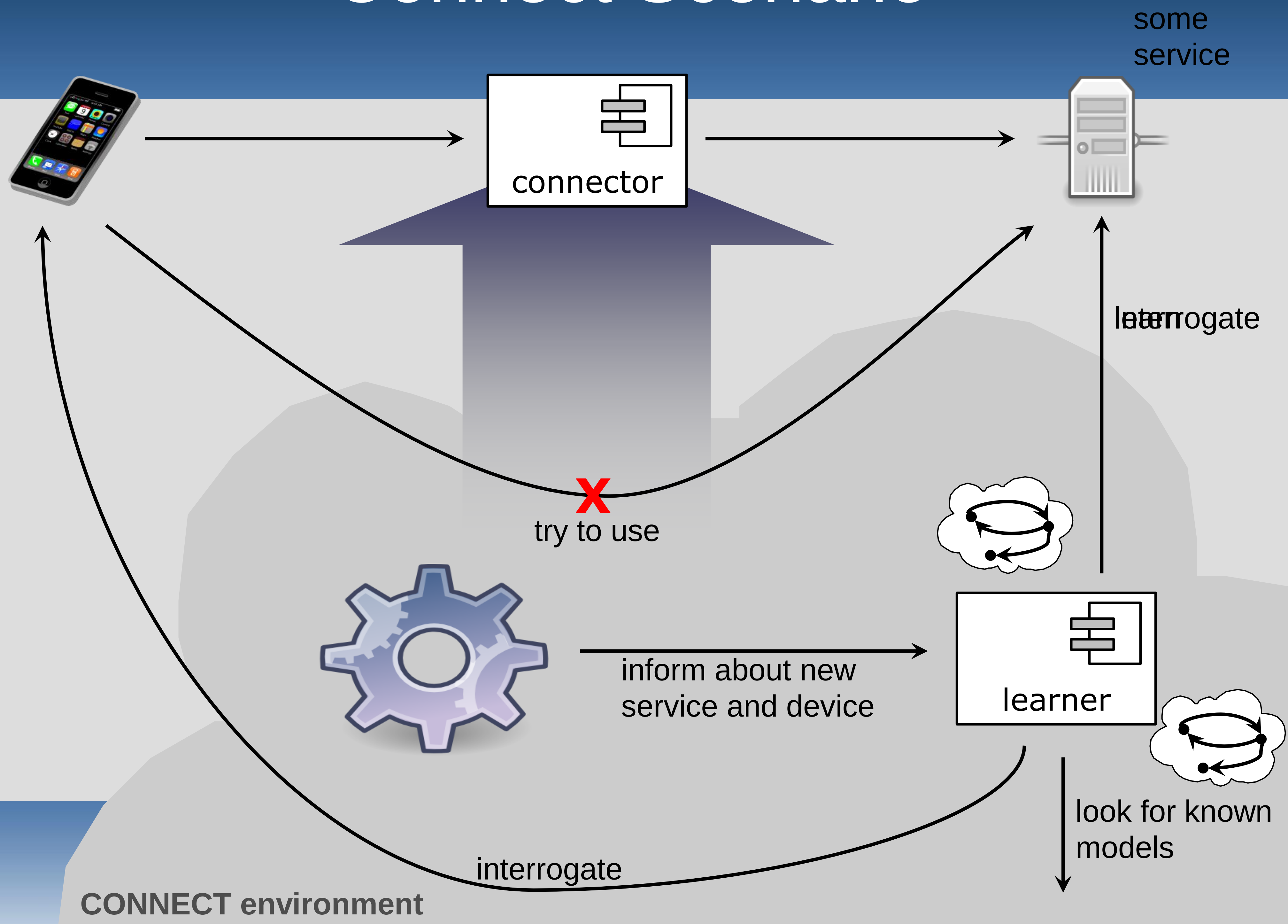
Bernhard Steffen, Falk Howar, Maik Merten
TU Dortmund

SFM2011

Overview

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

Connect Scenario



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

Metrics of interest

WP4 Learning Enabler

Rich & abstract models:

- Data parameters & state variables
- Pre- and post-conditions
- Non-functional properties

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
  - Σ finite input-alphabet
  - **Γ finite output-alphabet**
  - σ: $(S \times \Sigma) \rightarrow S$ transition-function
  - **γ: $(S \times \Sigma) \rightarrow \Gamma$ output-function**

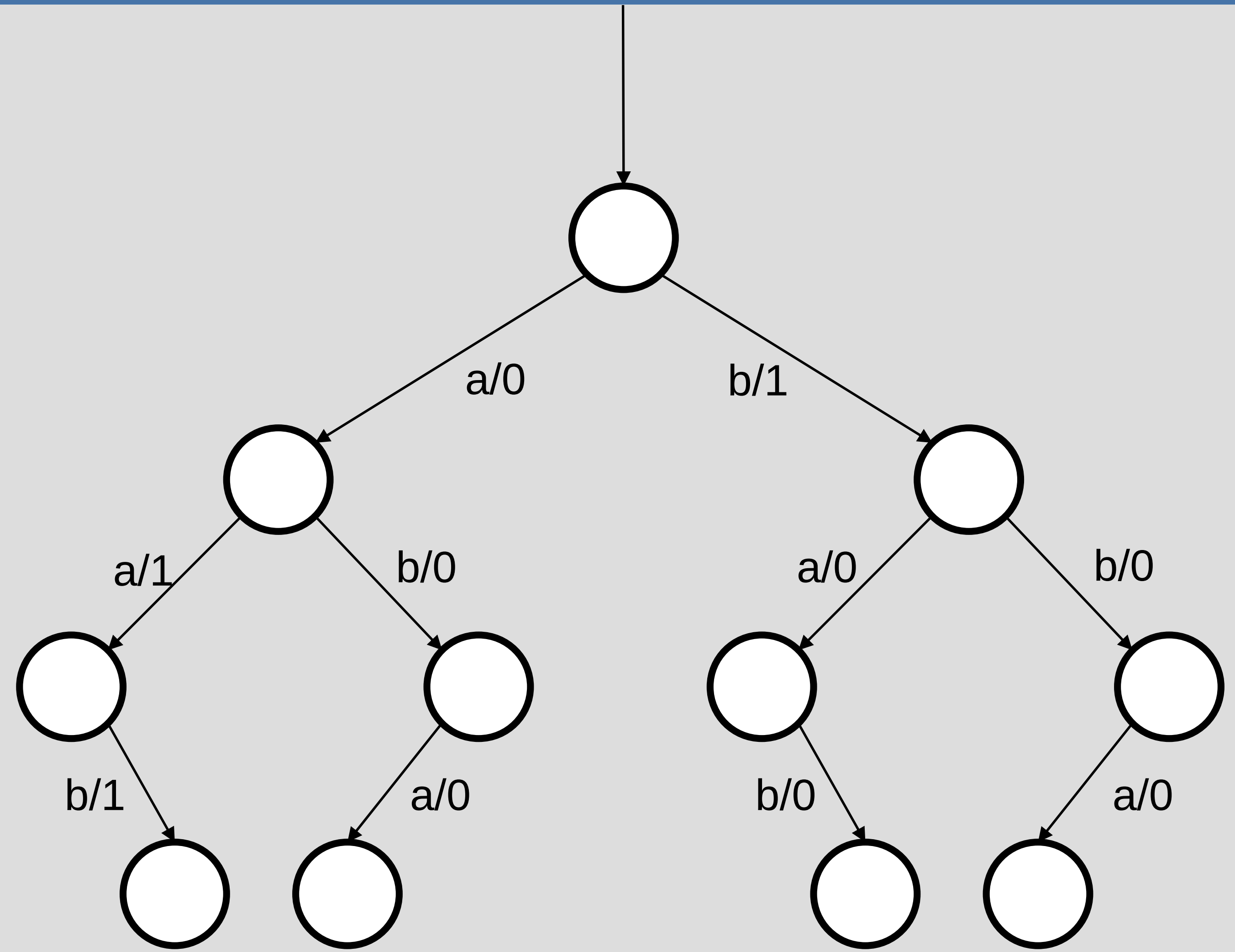
- Words Σ^* for ($s \in S$, $a \in \Sigma$, $w \in \Sigma^*$)
  - $\sigma:(S \times \Sigma^*) \rightarrow S$, $\sigma(s,\varepsilon)=s$, $\sigma(s,aw) =\sigma(\sigma(s,a),w)$
  - γ: $(S \times \Sigma^*) \rightarrow \Gamma^*$, $\gamma(s,\varepsilon)=\varepsilon$, $\gamma(s,aw) = \gamma(s,a).\gamma(\sigma(s,a),w)$

Passive learning or learning with traces

- tape-record communication
- Create observation tree
- Construct automaton without contradiction

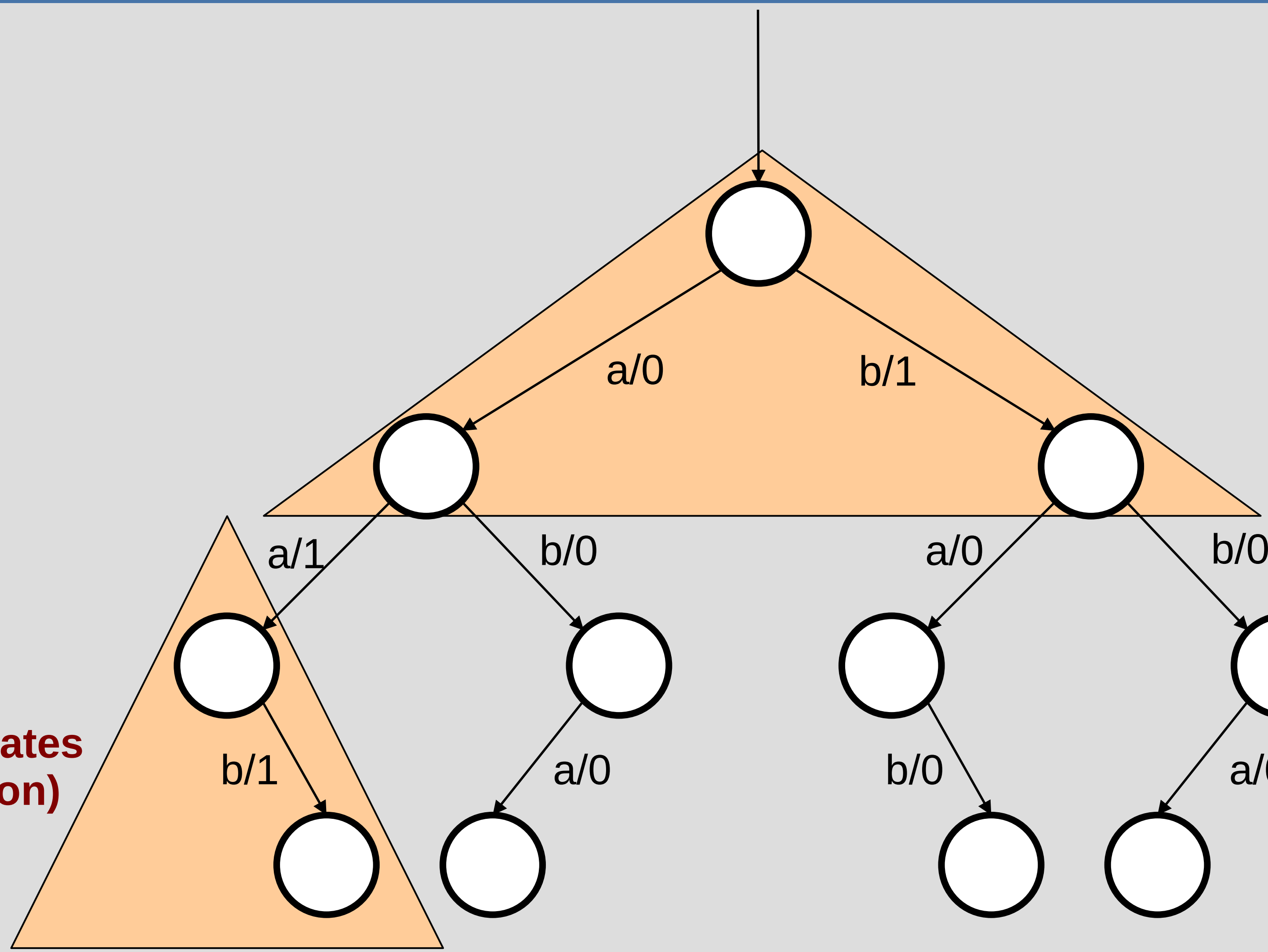
Passive learning or learning with traces

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

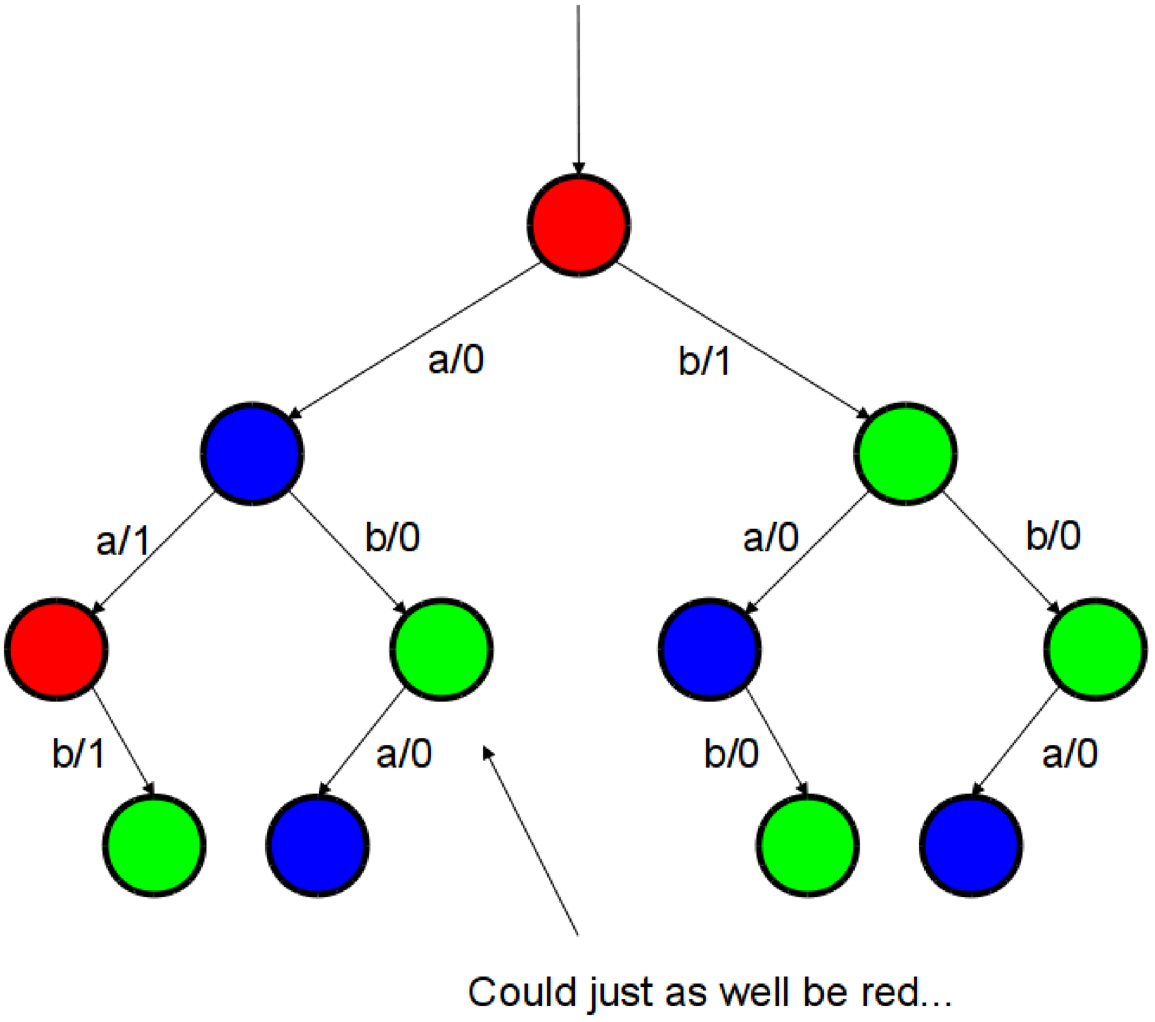
a/0 a/1 b/1
a/0 b/0 a/0
b/1 a/0 b/0
b/1 b/0 a/0



"equivalent" states (no contradiction)

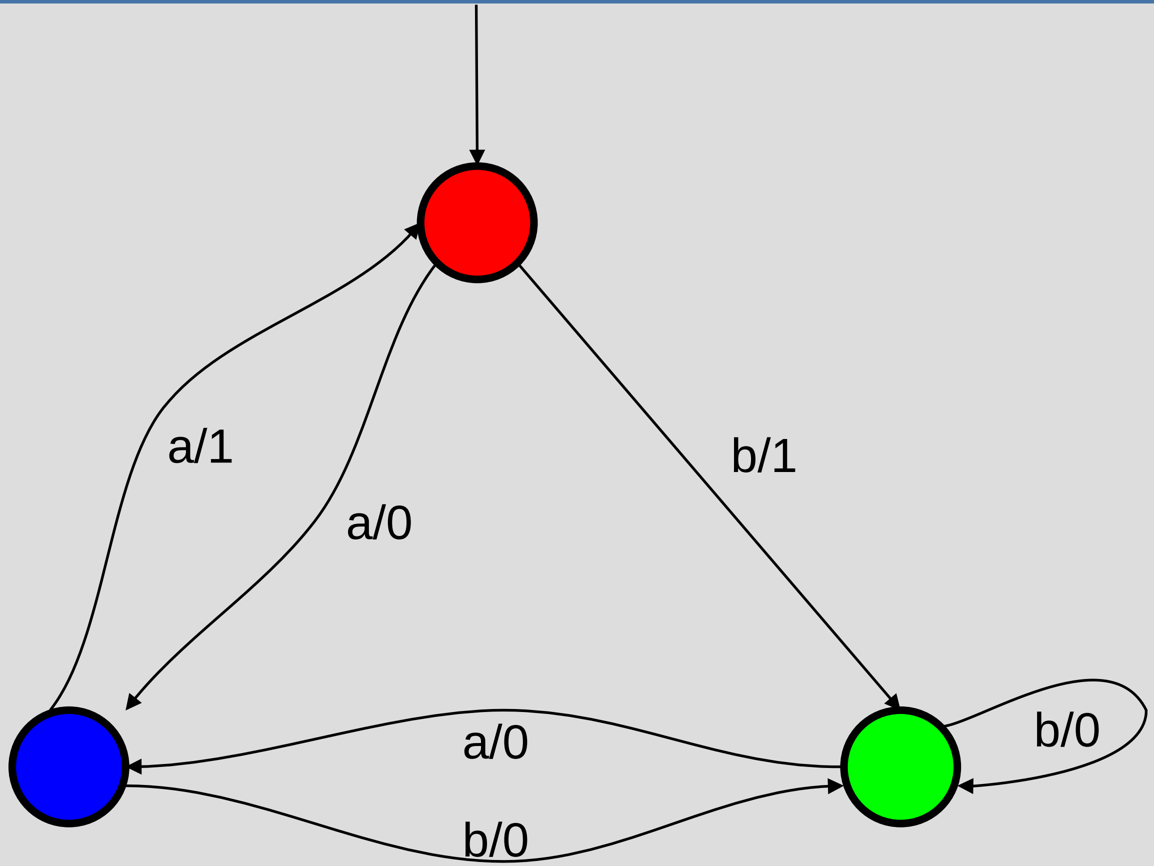
Passive learning or learning with traces

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

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 eqivalence relations
  - Computing the best choice of equivalence is:
    - Expensive for criteria like state minimality
    - Impossible in terms of adequacy for the problem.

Active automata learning

Idea 1: Ask where infomation 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

Angluin's algorithm

Idea 2: Enforce **consistency!**

- Refine the ‚not in conflict‘ relation
- Also consider whether the t**ransitions 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!**

Angluin's algorithm

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?)

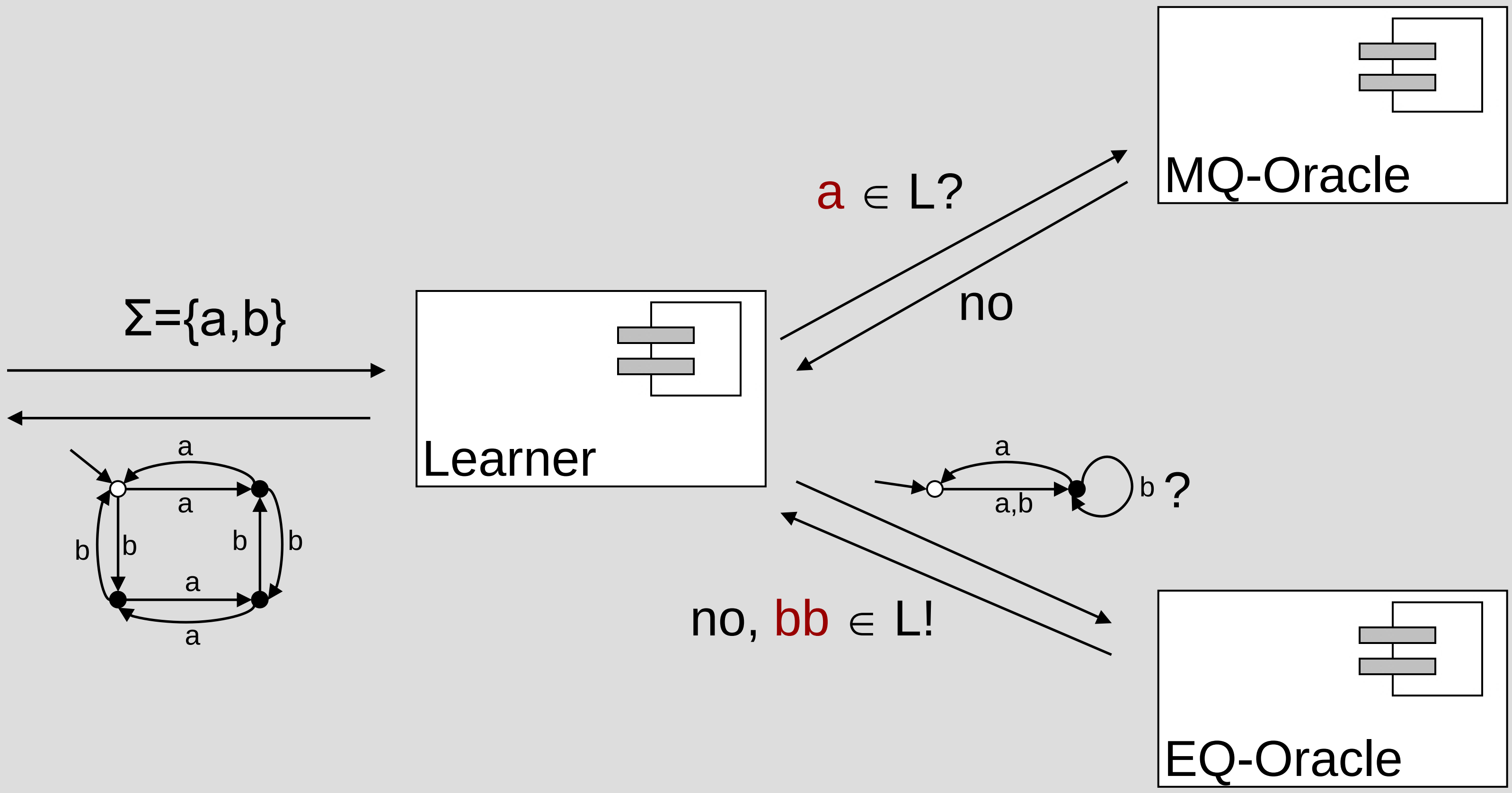
Angluin's algorithm

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



L* learner

the oracle

(queries)

should word w be included in L(A)?

yes / no

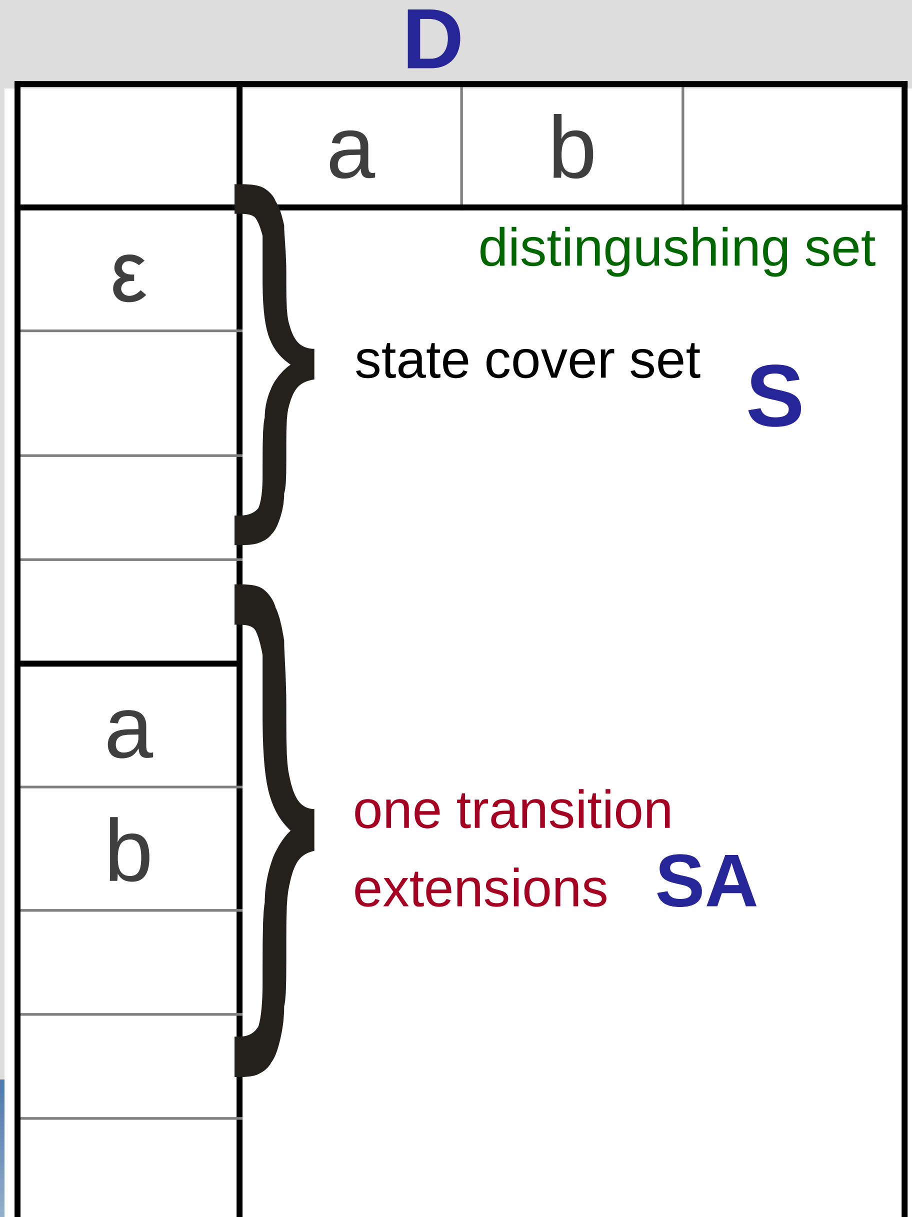
(conjectures)

here is an A – is L(A) = U?

yes!

no: word *w* should (not) be in L(A)

Angluin's alg. for Mealy machines

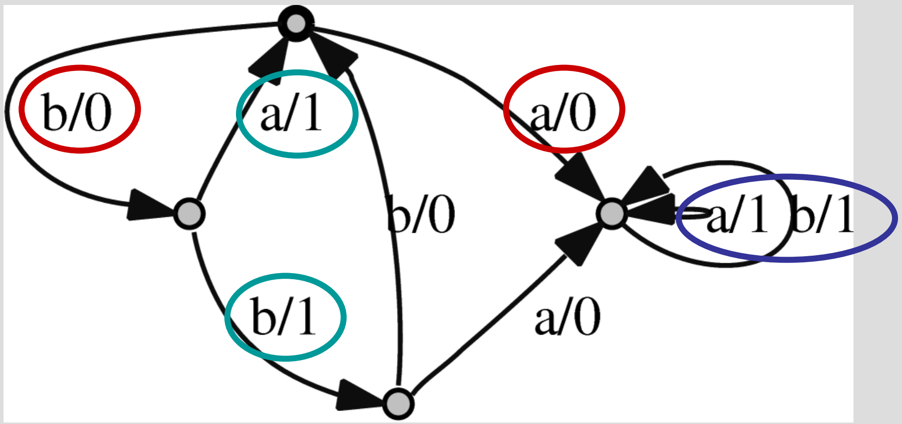


- Initialize **Distinguishing Set** D with alphabet of inputs

Angluin's alg. for Mealy machines

	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

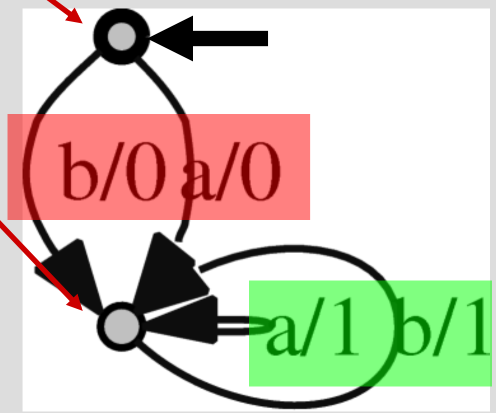
	a	b	
ε	0	0	
a	1	1	
b	1	1	
aa	1	1	
ab	1	1	

- **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	

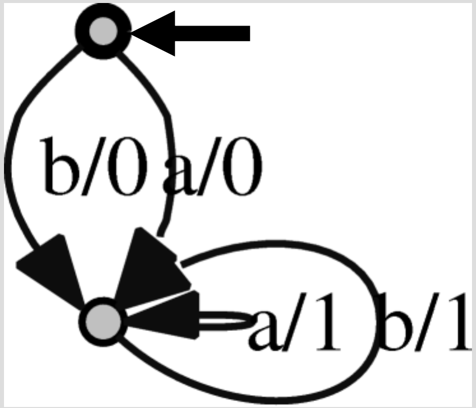
- **Conjecture**:
- Unique rows in **S** become states
- Rows in **S** and **SA** become transitions



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	
…	…	…	

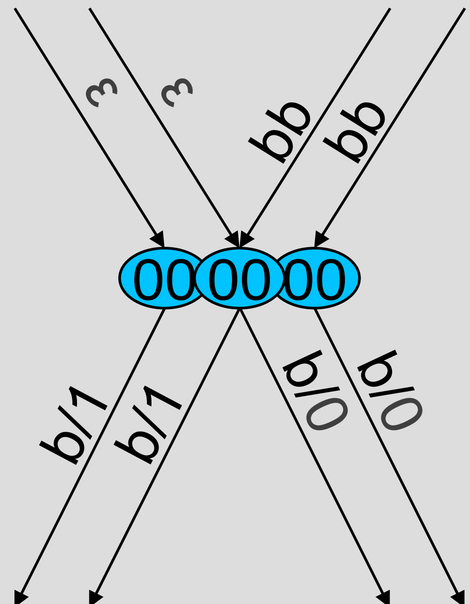
- **Counterexample**: bbb / 010

- 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'
- **b** and **bbb** differ, e.g., for suffix **b**

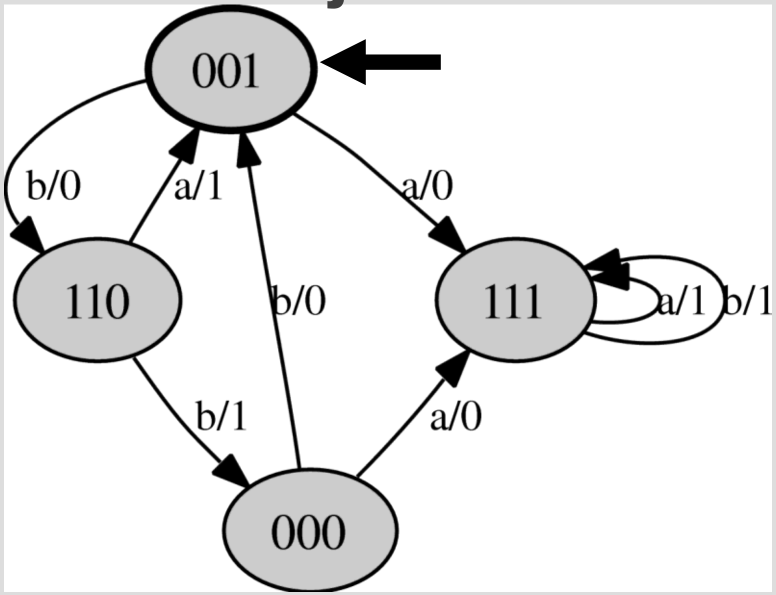
=> **ε** and **bb** will differ for suffix **bb**

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
…	…	…	…

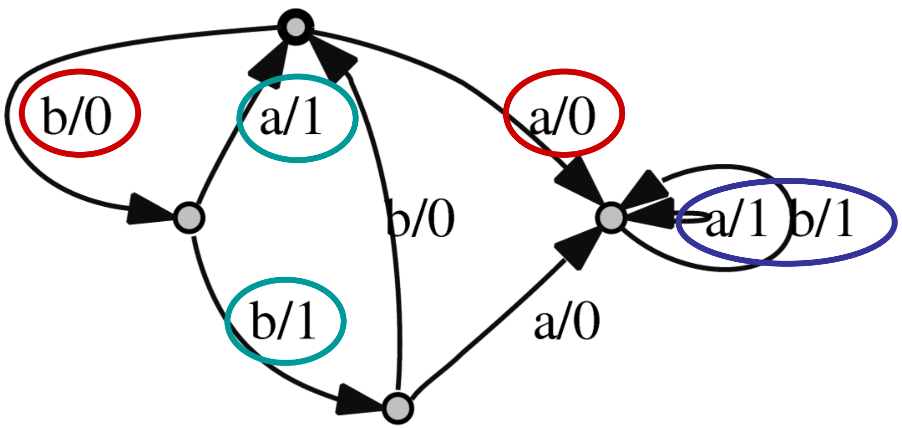
- **Inconsistencies** lead to new columns

New Conjecture

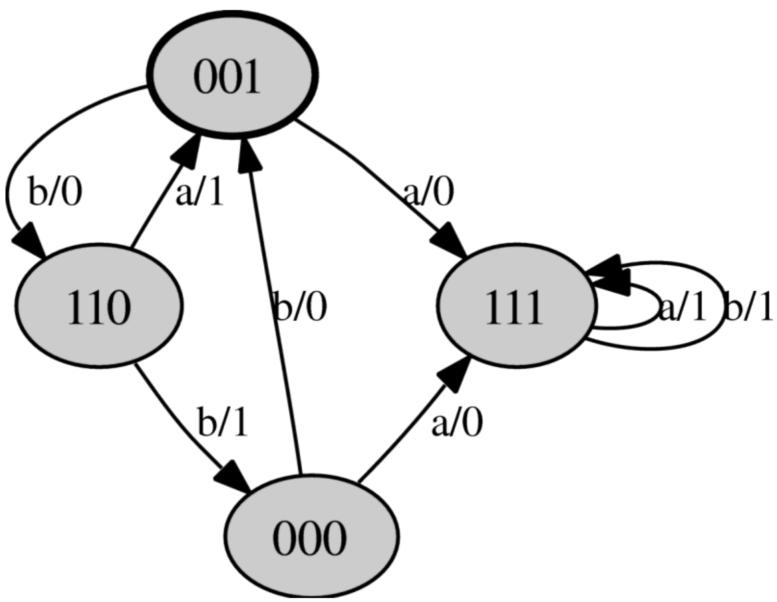


Angluin's alg. for Mealy machines

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**

Angluin's algorithm

Hypothesis models or **conjectures**:

- Closed and consistent models (projective quotients) of the so far expanded
  - *homogeously extended, and*
  - *consistent*

  execution tree.

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.

Complexity of Angluin

Equivalence Queries

At most **$|Q|$**

Membership Queries

At most **$O(m\ |Q|\ |\Sigma_A|\ )$** per **EQ** (m = 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

Conceptual Improvements 1

one erssential suffix

All prefixes of counterexample …

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

Reduced observation table

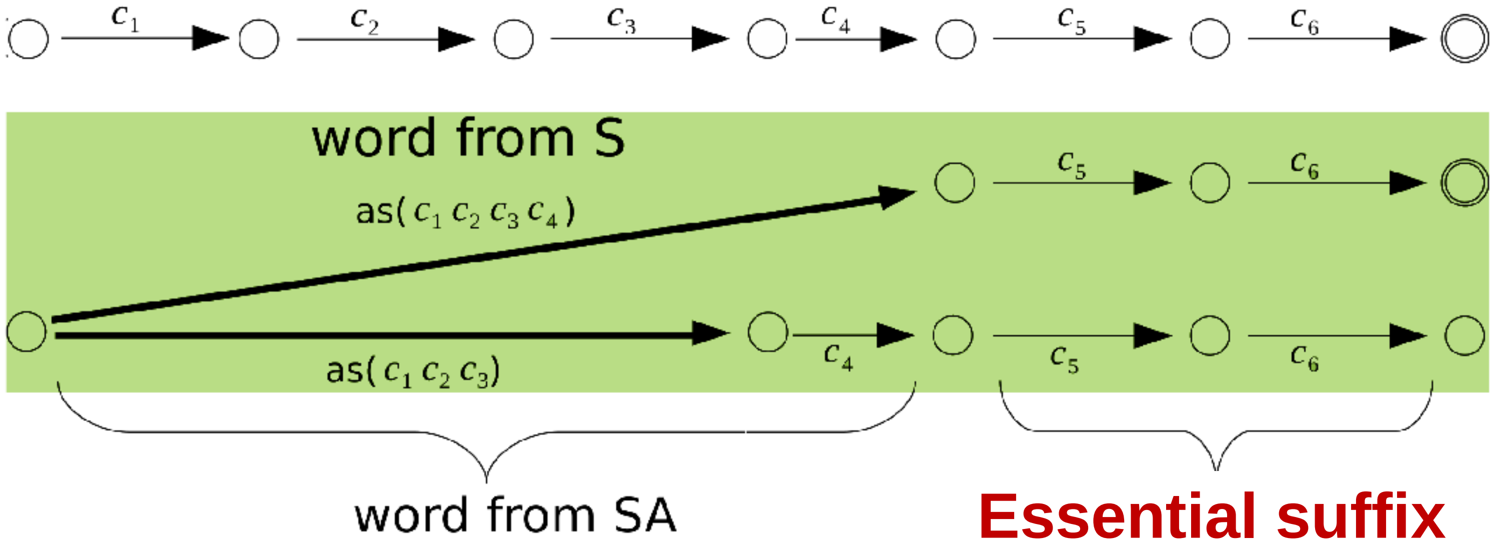
- **Rivest and Shapire**: Analyze counterexample separately (not in the table)
  - Only add **one** ‚essential‘ suffix (i.e., witness), as column label to the table

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)

Complexity (reduced 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

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

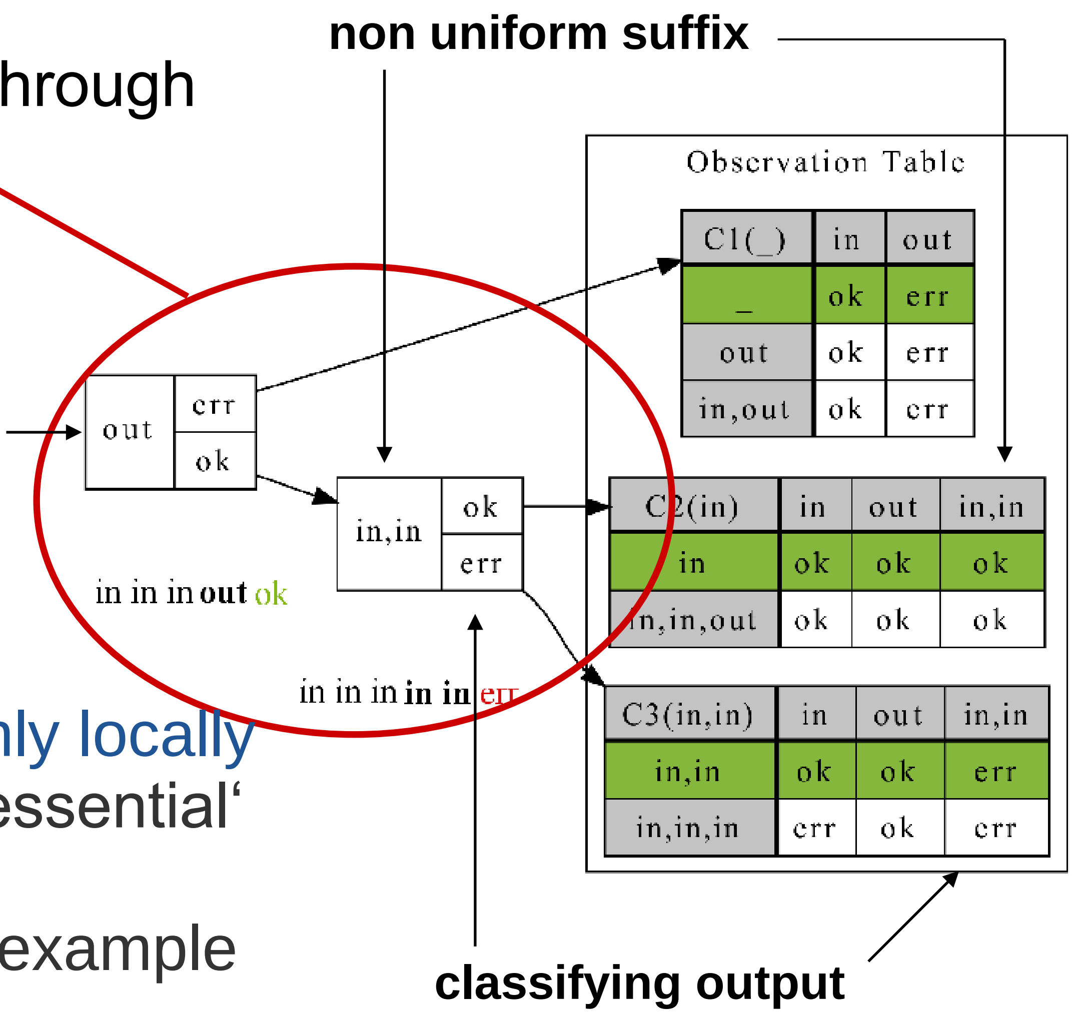
All rows are filled completely, even if unnecessary

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



Discimination 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

Correctness pattern (maintained)

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 determinsitc automaton** for the considered language / set of traces.

Overview

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

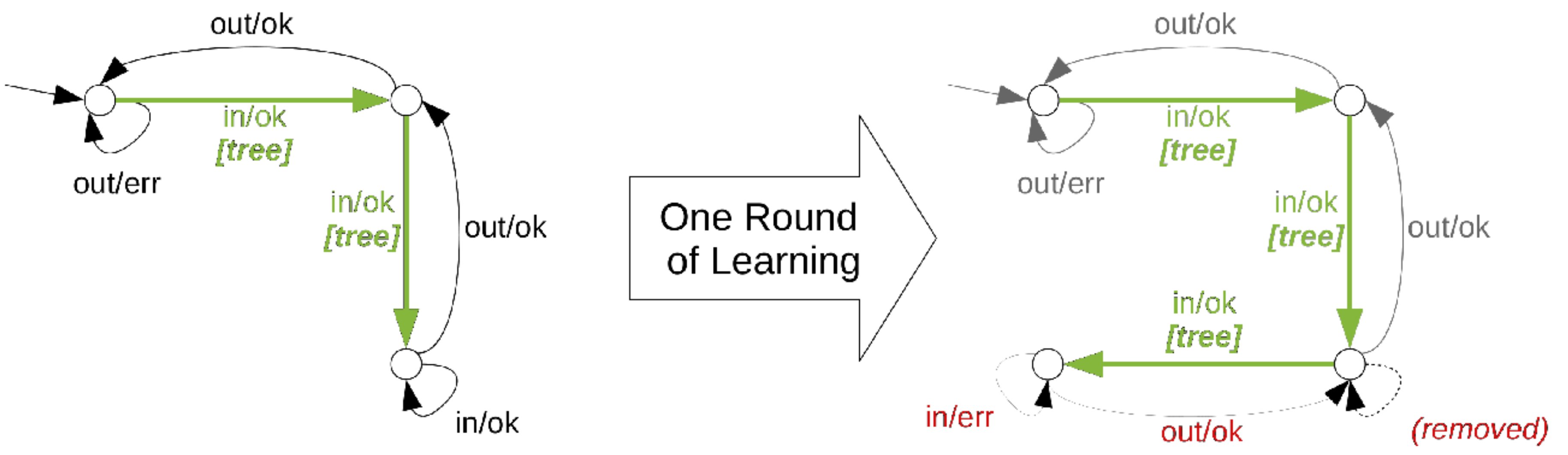
Practical results II

The ZULU competition

The ZULU challenge

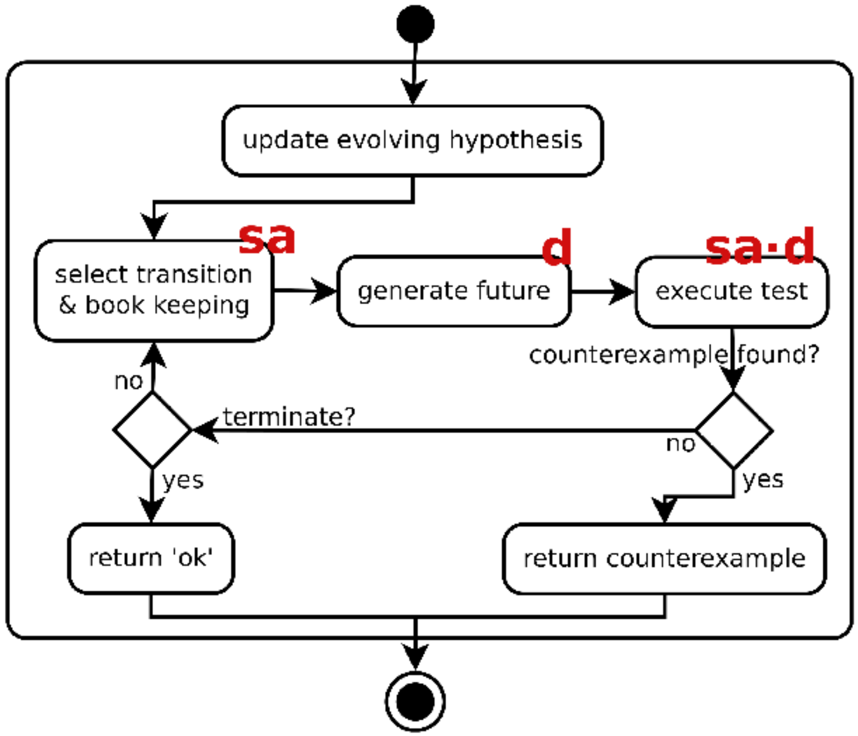
- Competition in active learning (2010)
- No **equivalence queries** allowed, limited amount of membership queries
- **Randomly** generated automata
- Test-based evaluation
- http://labh-curien.univ-st-etienne.fr/zulu/

Evolving hypothesis



- 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

ZULU competition results

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

Kearns & Vazirani: High impact even here!

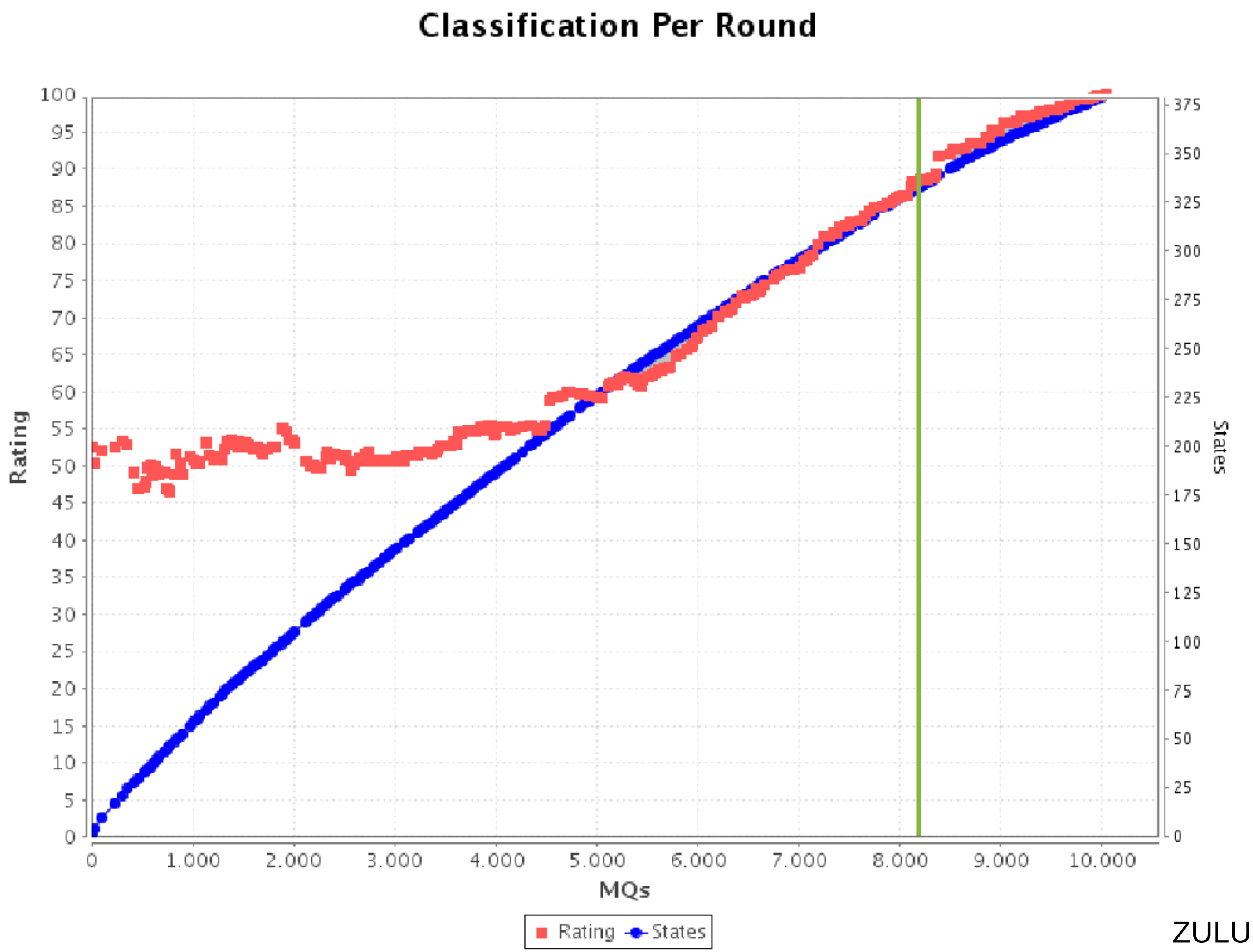
- Uniform DFA: ca. 83 , non-uniform Mealy: ca. 85

Detailed results

Algorithm	New Membership Queries			Rounds	States	Score
	Close	Analyze	Search			
E.H.Blocking	6,744	358	999	259	352	94.11
E.H.Weighted	6,717	349	1,035	262	351	94.61
Random	6,586	519	996	228	332	93.28
run_random	8,080	14	7	5	312	74.89
run_blocking1	8,074	11	16	6	319	73.06
run_weighted1	8,077	9	15	6	319	74.39

- 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

Asymtotic costs per state

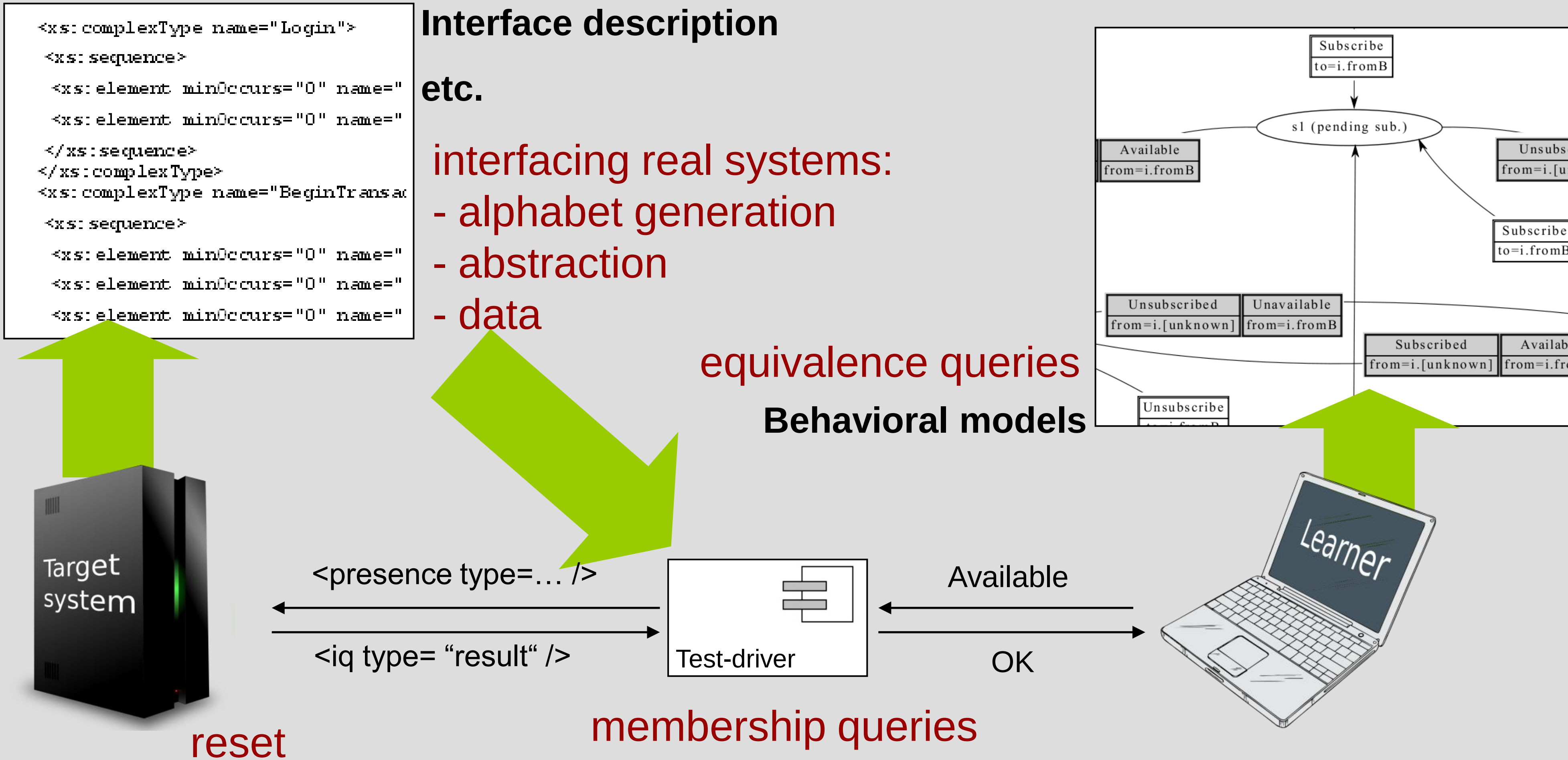


ZULU Problem 49763507

Practical results I

More Applications

Practical challenges

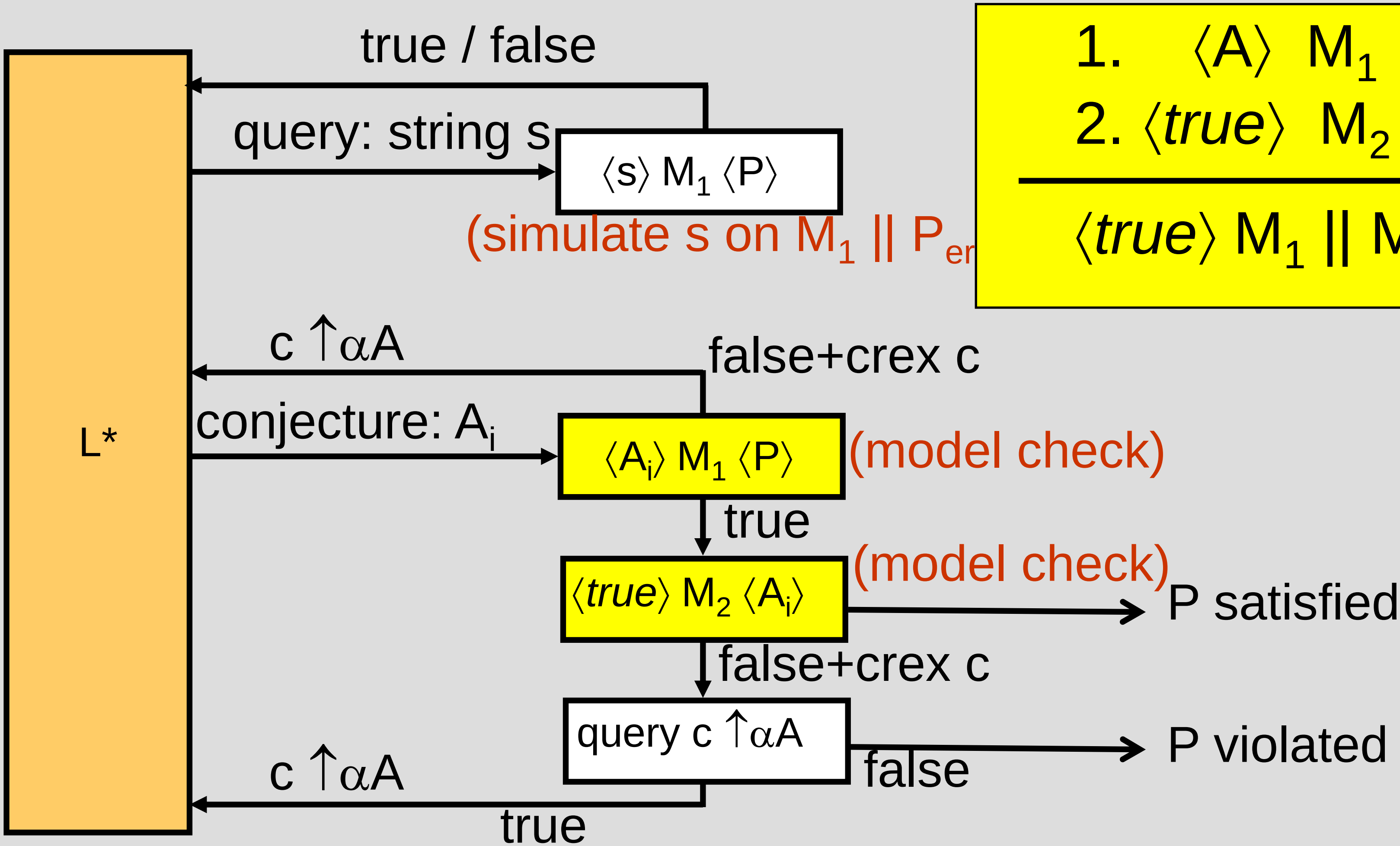


Practical results I

Learning assumptions

oracle for WA in assume-guarantee reasoning

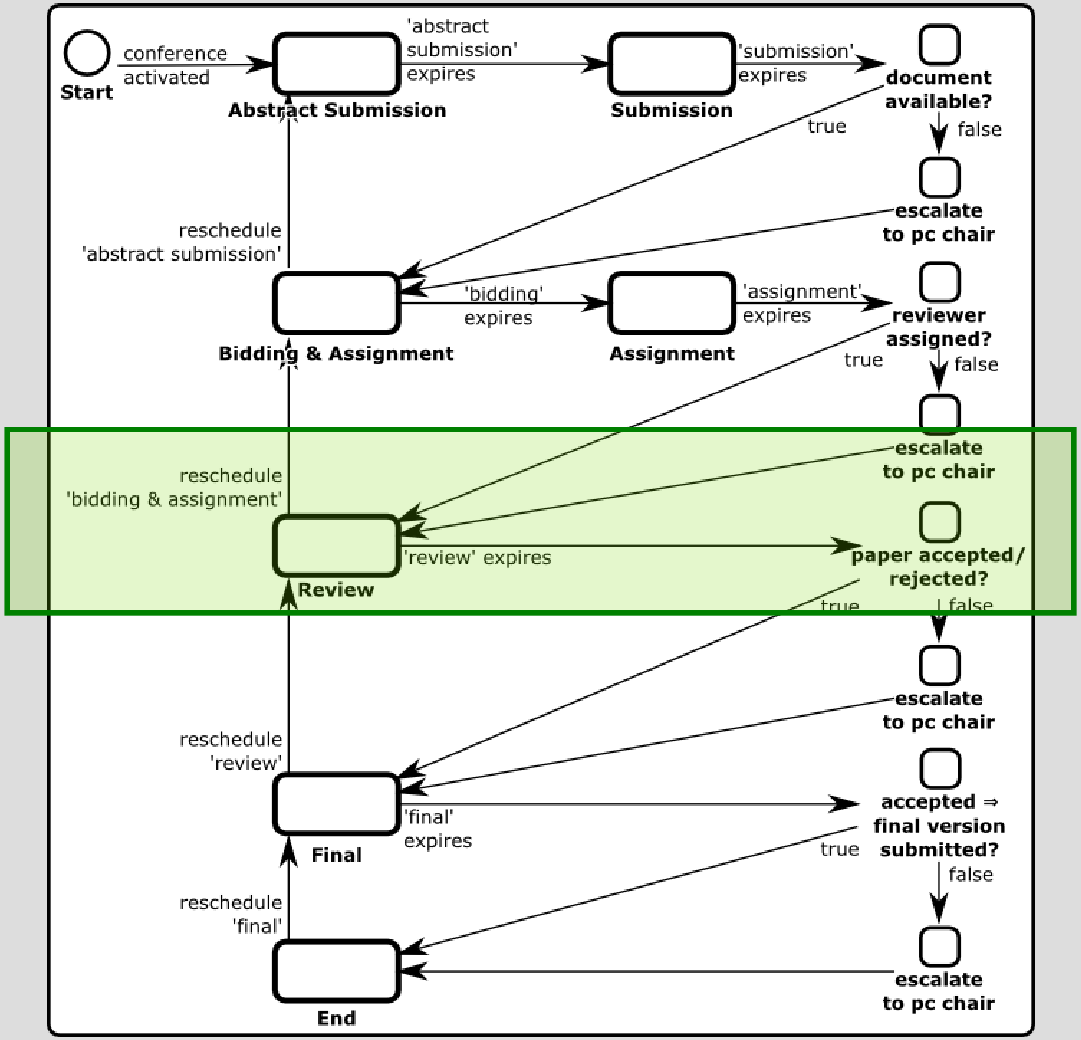
$$\frac{\begin{array}{l}1.\ \langle A\rangle\ M_1\ \langle P\rangle \\ 2.\ \langle \mathit{true}\rangle\ M_2\ \langle A\rangle\end{array}}{\langle \mathit{true}\rangle\ M_1 \parallel M_2\ \langle P\rangle}$$

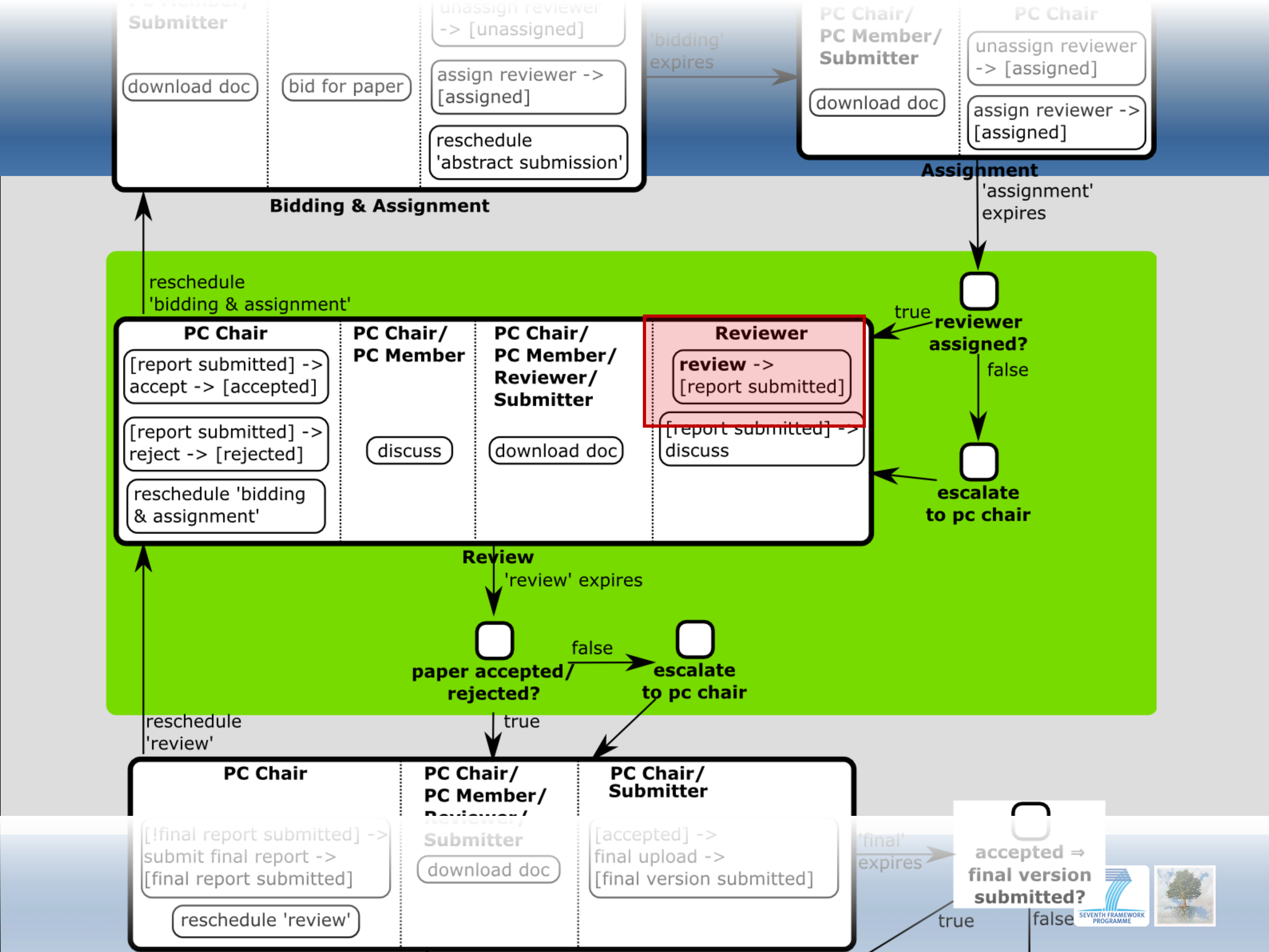


Practical results I

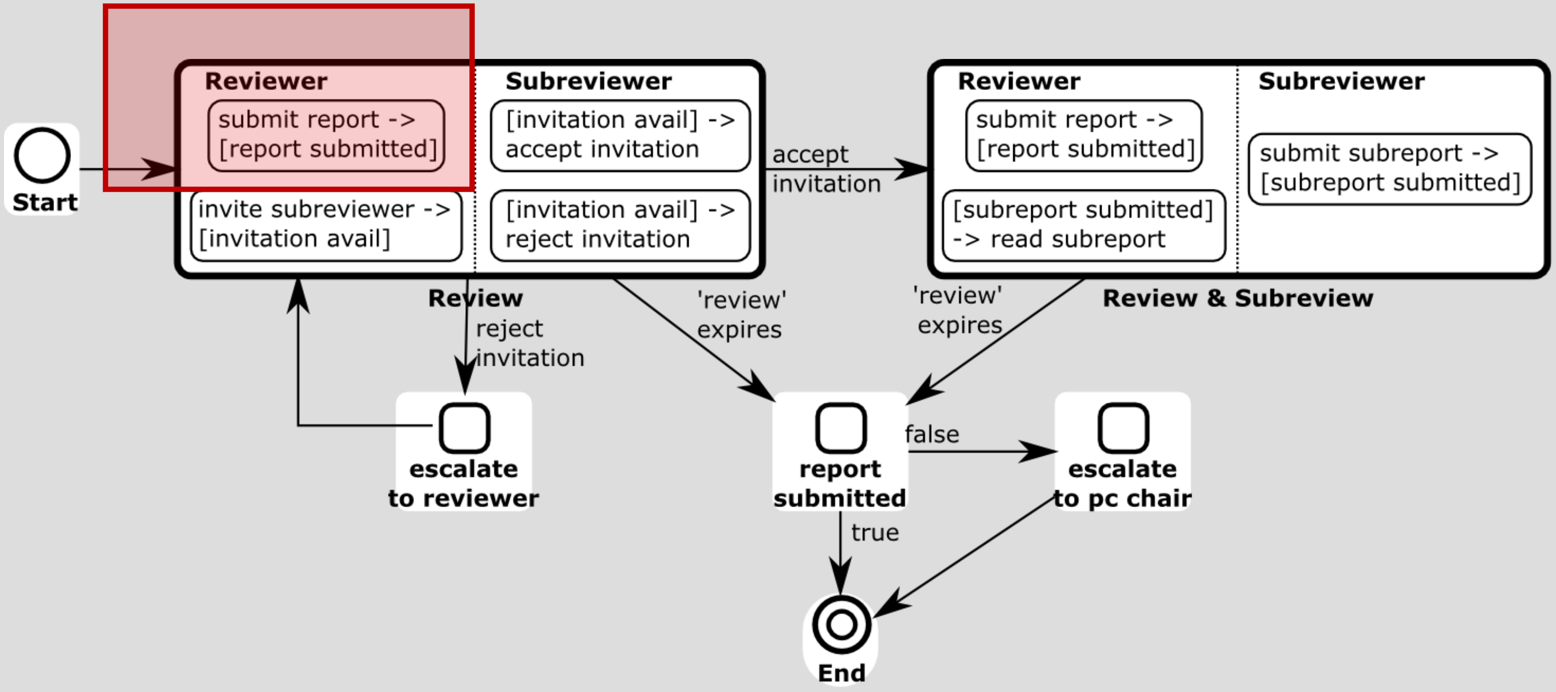
Learning the OCS

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
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
escalate to pc chair
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
SEVENTH FRAMEWORK PROGRAMME

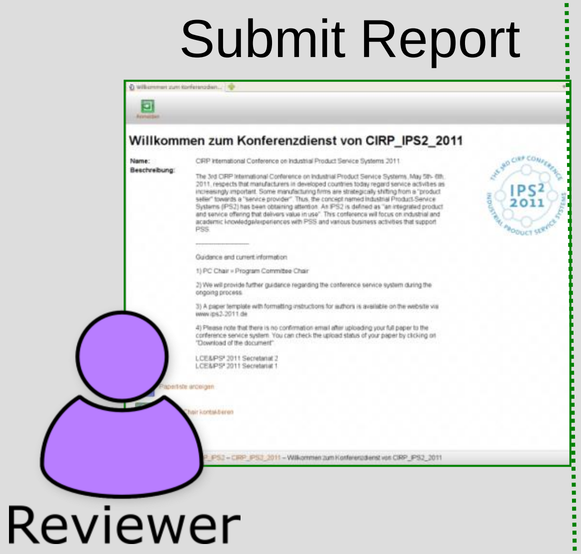
Hierarchy

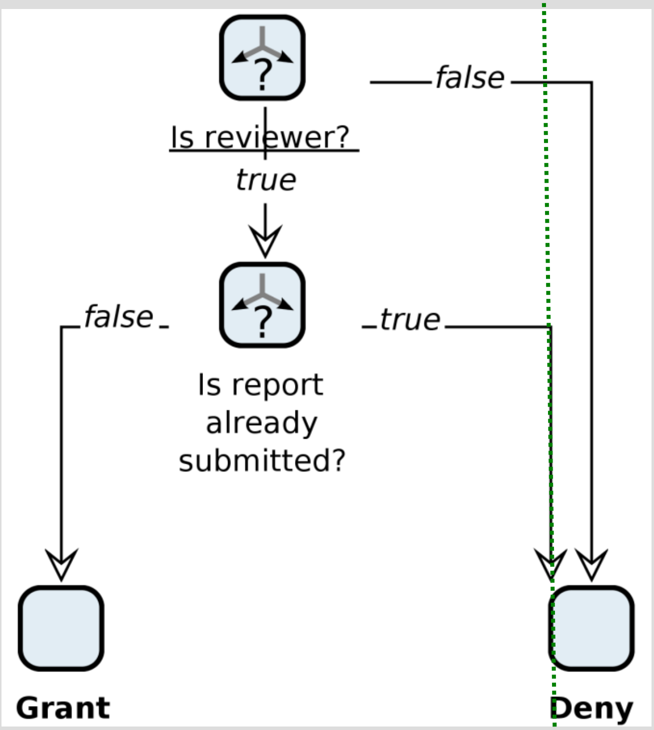


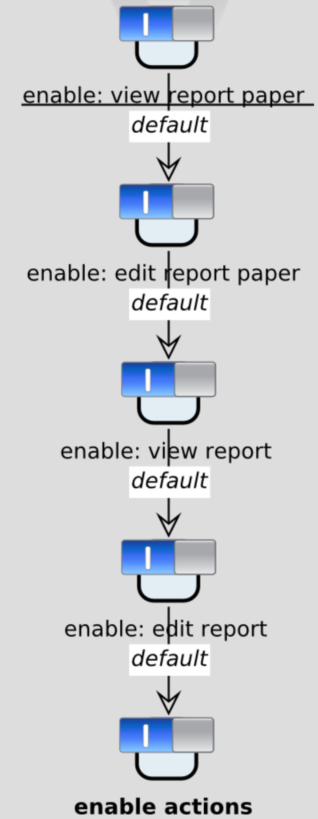
Reviewer
submit report -> [report submitted]
invite subreviewer -> [invitation avail]
Subreviewer
[invitation avail] -> accept invitation
[invitation avail] -> reject invitation
Review
Start
accept invitation
Reviewer
submit report -> [report submitted]
[subreport submitted] -> read subreport
Subreviewer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

Event Condition Action

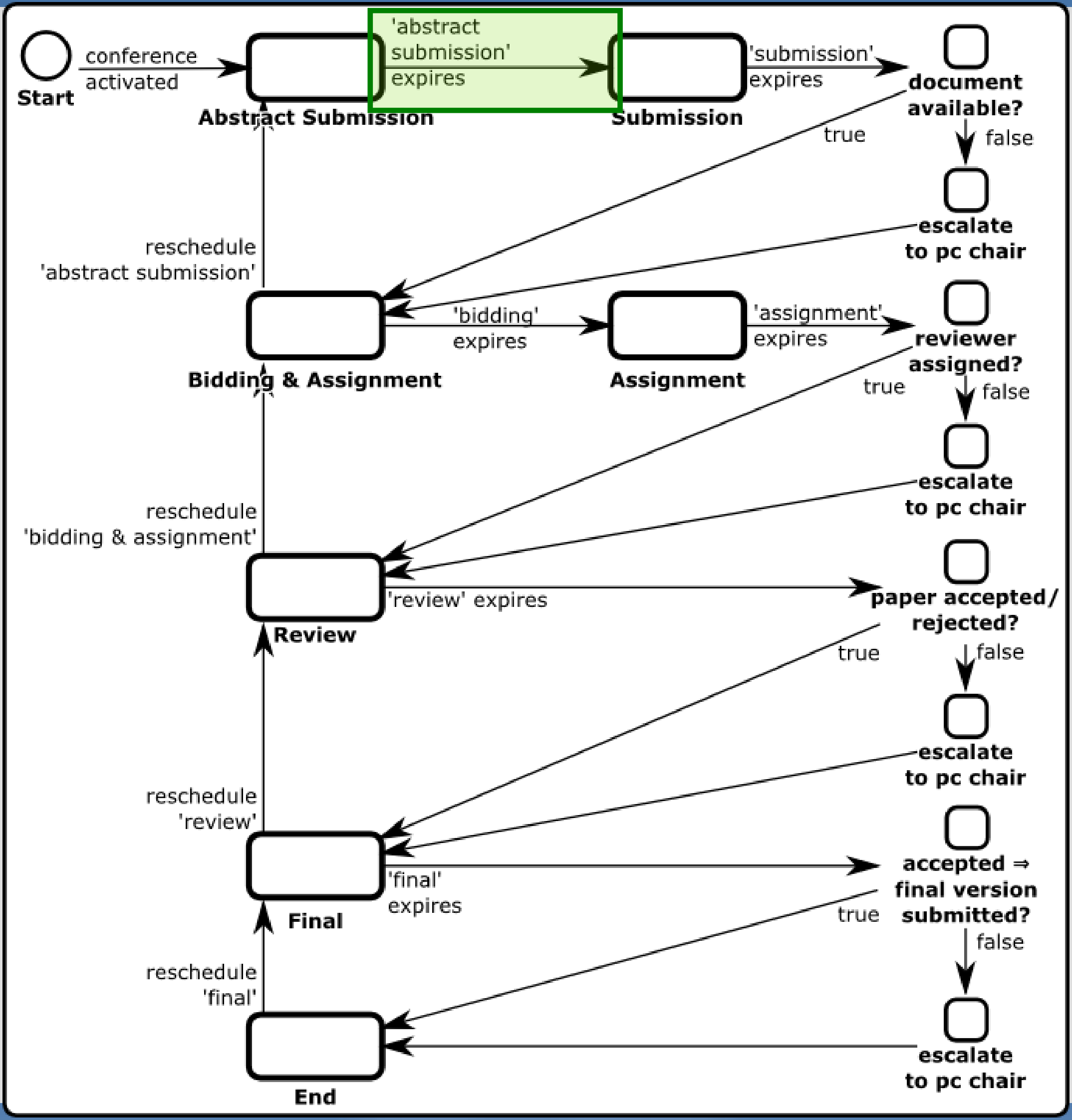
E **C** **A**



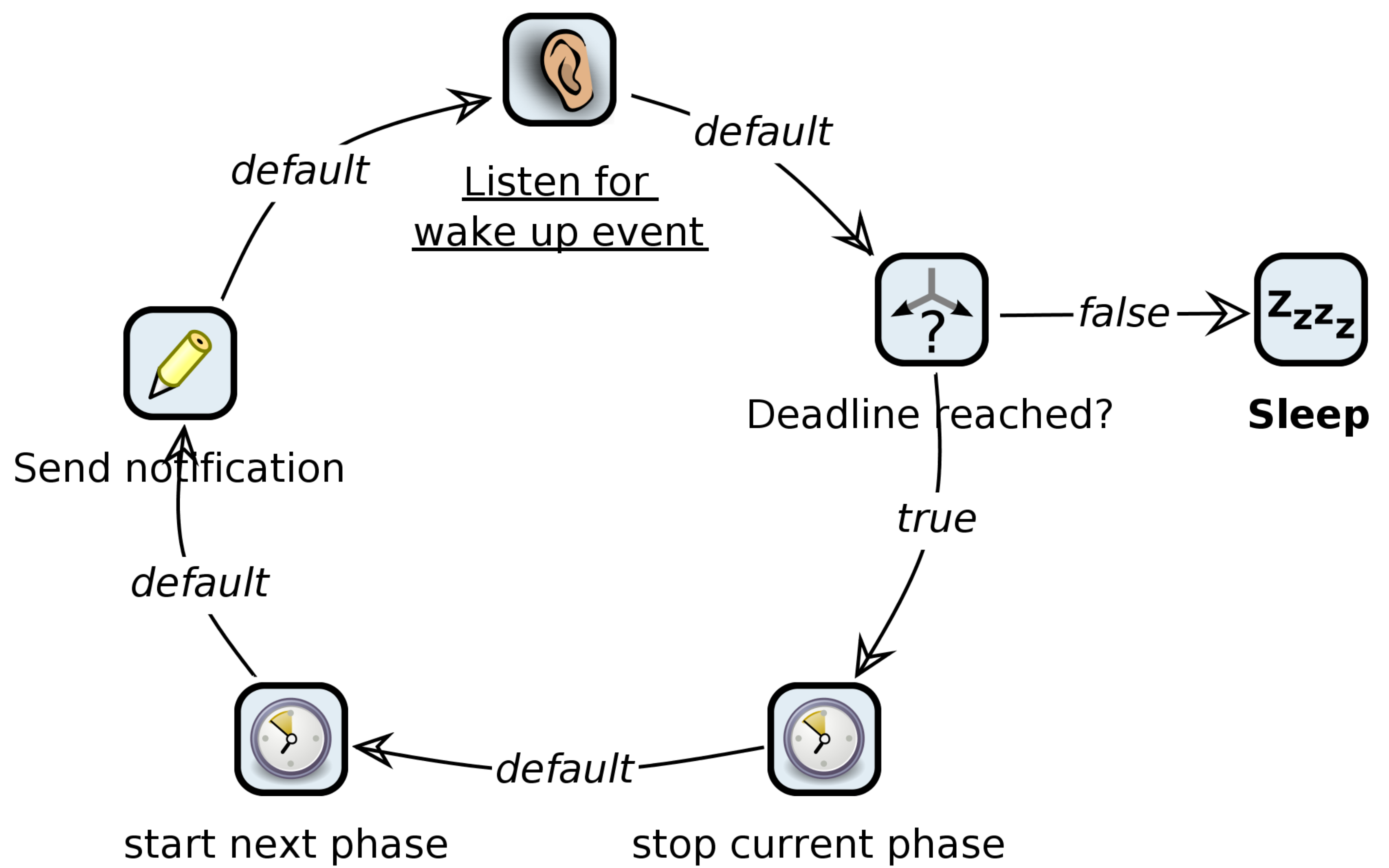




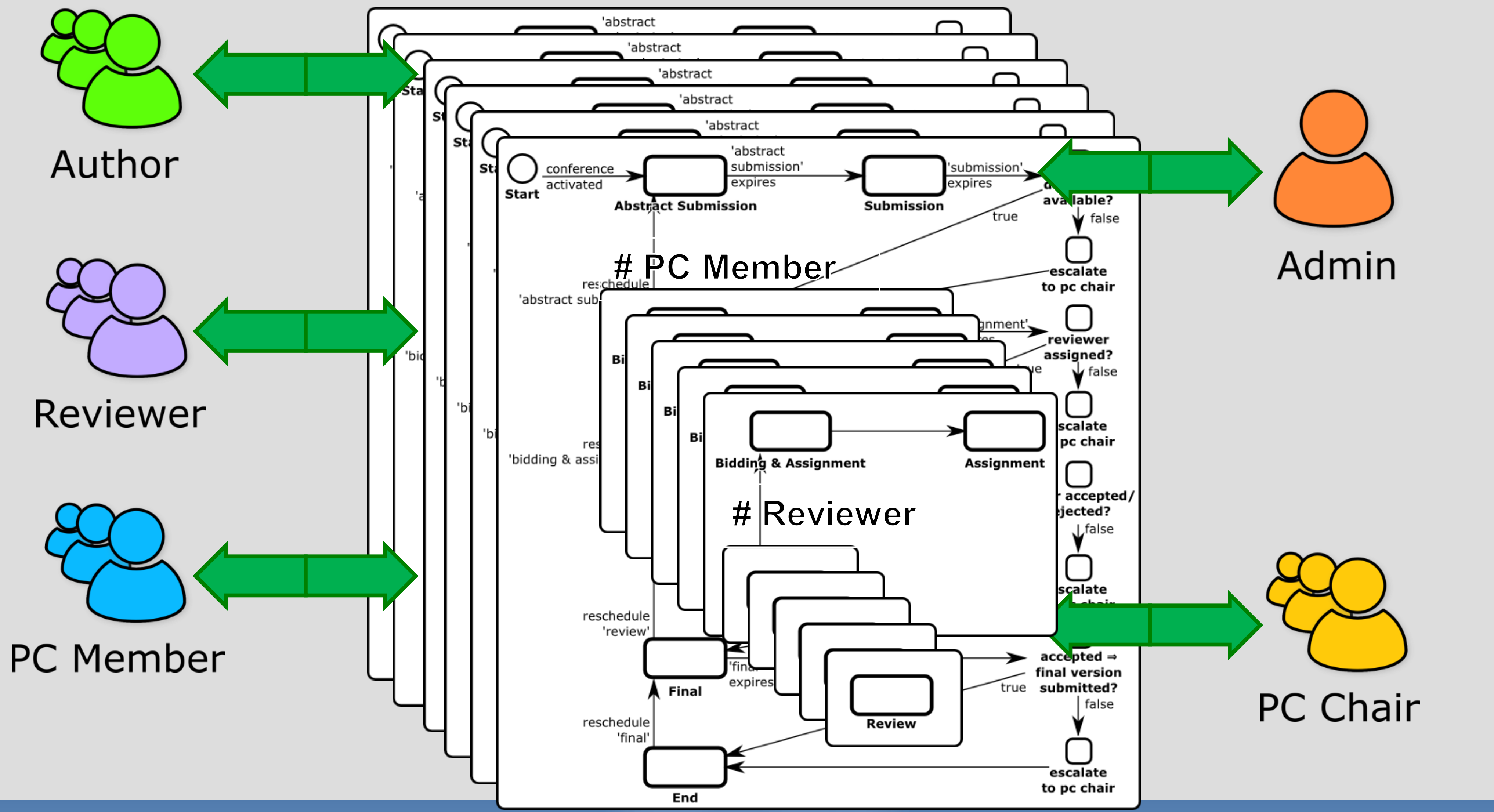
Semantics of "phase expires"-edges (1)



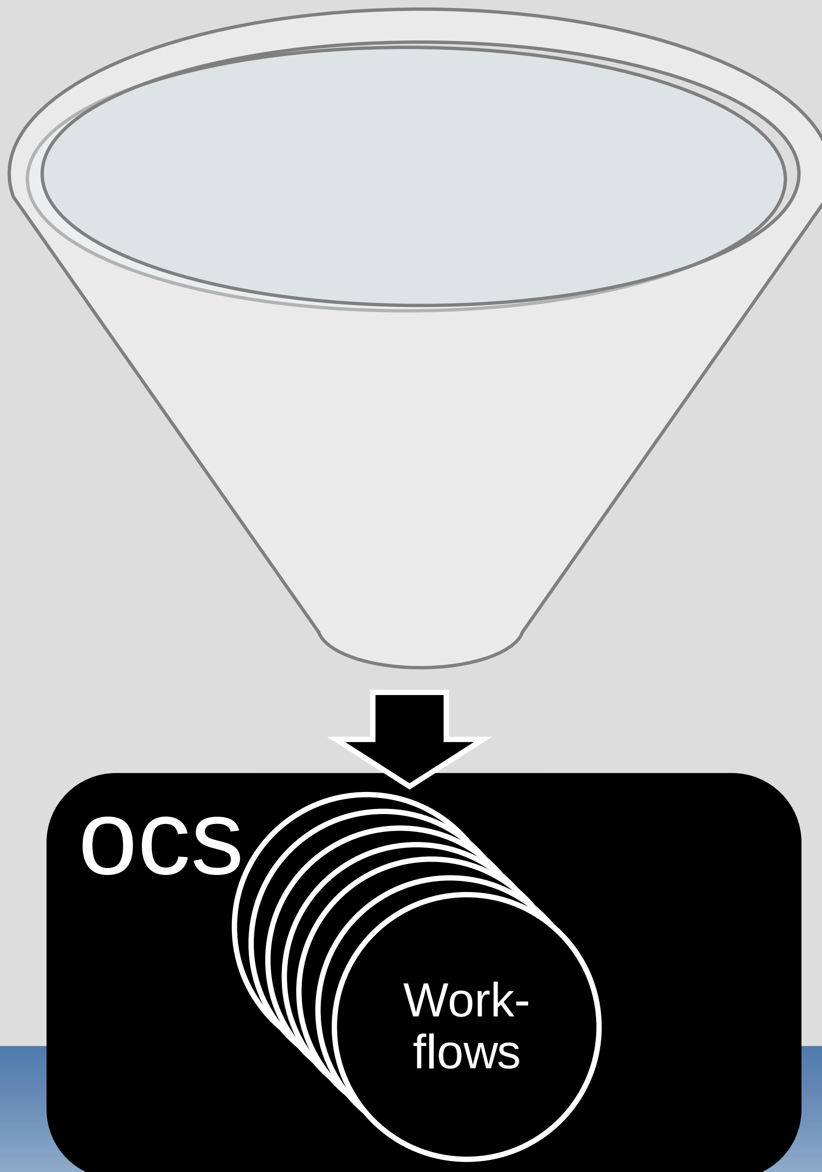
Semantics of "phase expires"-edges (2)



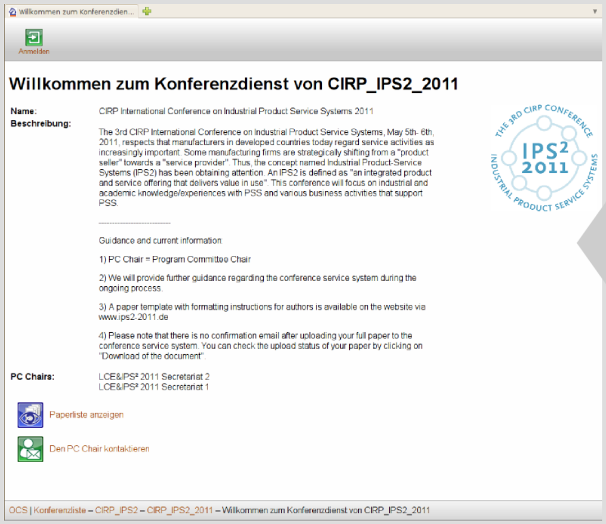
Many participants



Putting it all together



Regular extrapolation


Willkommen zum Konferenzdien...

Anmelden

Willkommen zum Konferenzdienst von CIRP_IPS2_2011

Name: CIRP International Conference on Industrial Product Service Systems 2011

Beschreibung:

The 3rd CIRP International Conference on Industrial Product Service Systems, May 5th- 6th, 2011, respects that manufacturers in developed countries today regard service activities as increasingly important. Some manufacturing firms are strategically shifting from a "product seller" towards a "service provider". Thus, the concept named Industrial Product-Service Systems (IPS2) has been obtaining attention. An IPS2 is defined as "an integrated product and service offering that delivers value in use". This conference will focus on industrial and academic knowledge/experiences with PSS and various business activities that support PSS.

Guidance and current information:

1) PC Chair = Program Committee Chair

2) We will provide further guidance regarding the conference service system during the ongoing process.

3) A paper template with formatting instructions for authors is available on the website via www.ips2-2011.de

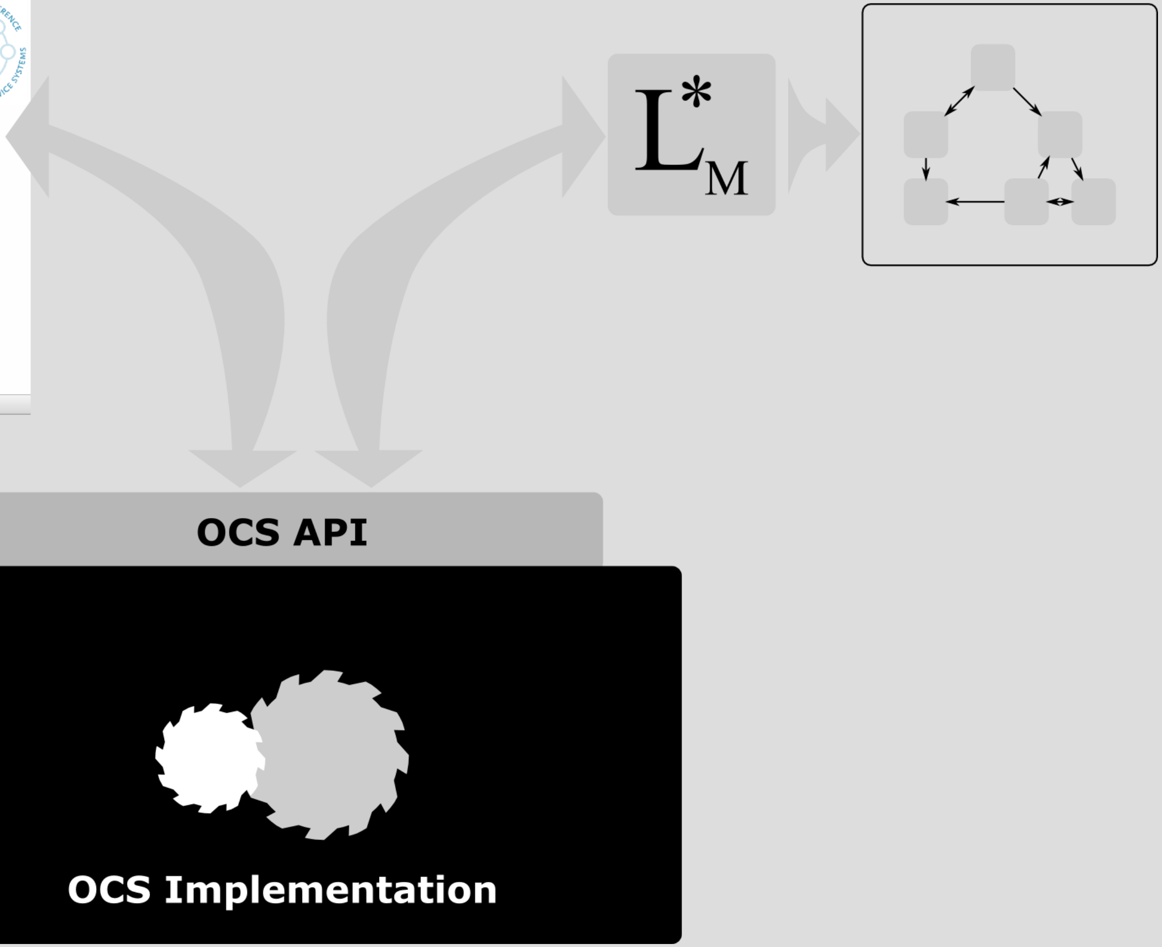
4) Please note that there is no confirmation email after uploading your full paper to the conference service system. You can check the upload status of your paper by clicking on "Download of the document".

PC Chairs: LCE&IPS² 2011 Secretariat 2
LCE&IPS² 2011 Secretariat 1

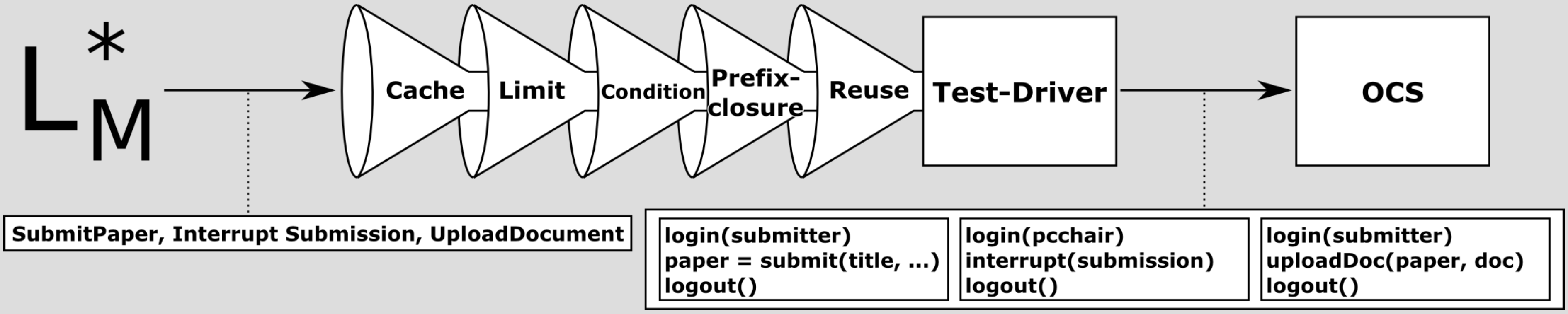
Paperliste anzeigen

Den PC Chair kontaktieren

OCS | Konferenzliste – CIRP_IPS2 – CIRP_IPS2_2011 – Willkommen zum Konferenzdienst von CIRP_IPS2_2011



Optimized learning setup



Learning algorithm

- Observation Table
- Mealy machine inference
- Regular extrapolation

	SP	UD	DD	DP
λ	√	↯	↯	↯
SP	↯	√	↯	√
UD	√	↯	↯	↯
DD	√	↯	↯	↯
DP	√	↯	↯	↯

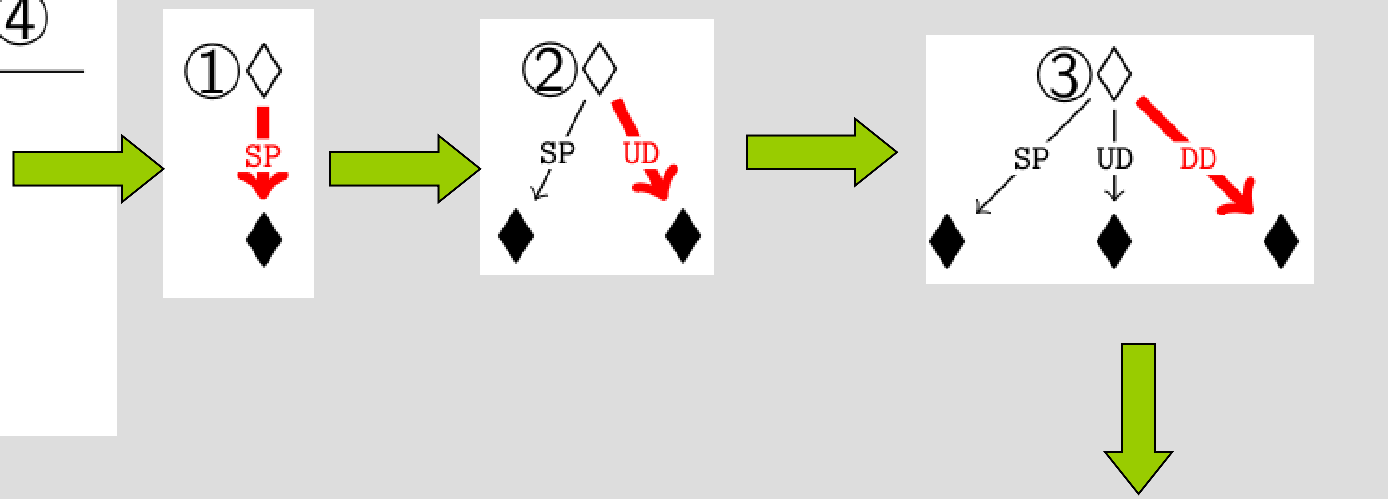
	SP	UD	DD	DP
λ	√	↯	↯	↯
SP	↯	√	↯	√
UD	√	↯	↯	↯
DD	√	↯	↯	↯
DP	√	↯	↯	↯
SPSP	↯	√	↯	√
SPUD	↯	√	√	√
SPDD	↯	√	↯	√
SPDP	√	↯	↯	↯

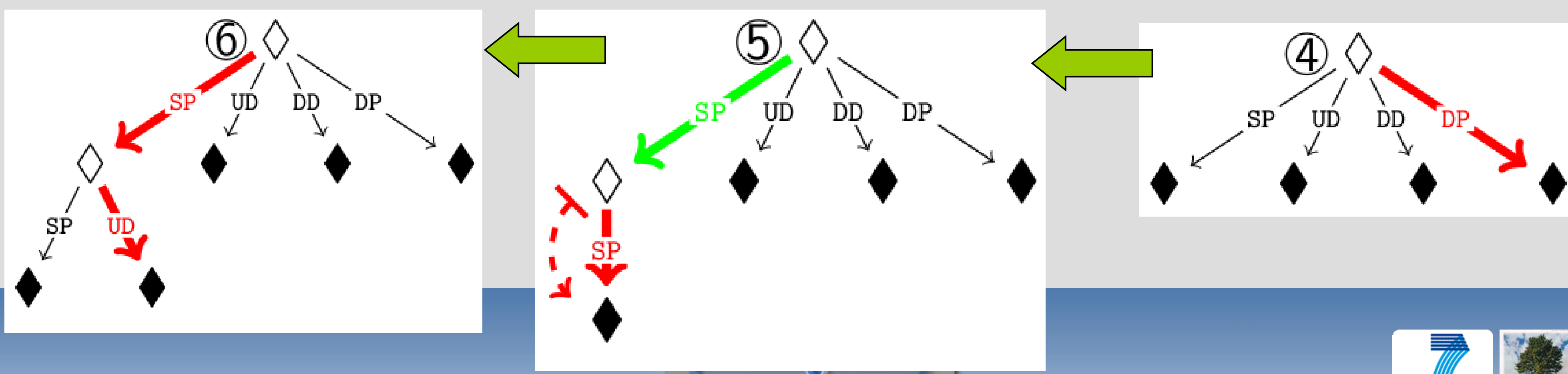
First Hypothesis

	SP	UD	DD	DP
λ	√	↯	↯	↯
SP	↯	√	↯	√
SPUD	↯	√	√	√
UD	√	↯	↯	↯
DD	√	↯	↯	↯
DP	√	↯	↯	↯
SPSP	↯	√	↯	√
SPDD	↯	√	↯	√
SPDP	√	↯	↯	↯
SPUDSP	↯	√	√	√
SPUDUD	↯	√	√	√
SPUDDD	↯	√	√	√
SPUDDP	√	↯	↯	↯

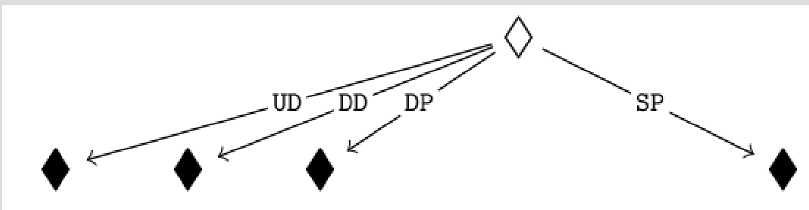
Reusing system states

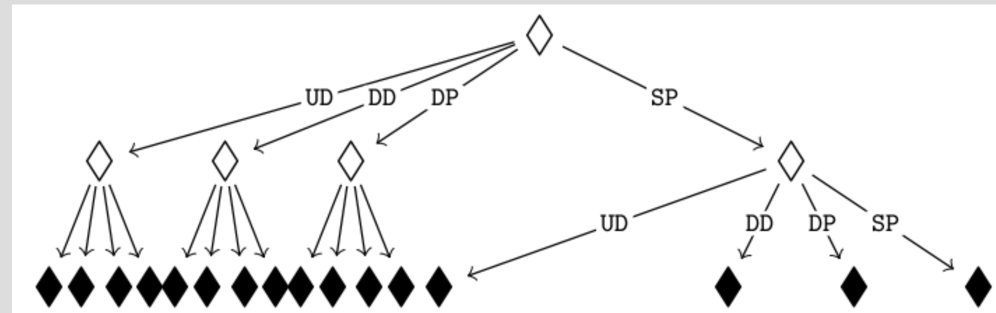
	SP	UD	DD	DP
λ	①	②	③	④
SP	⑤	⑥		
UD				
DD				
DP				



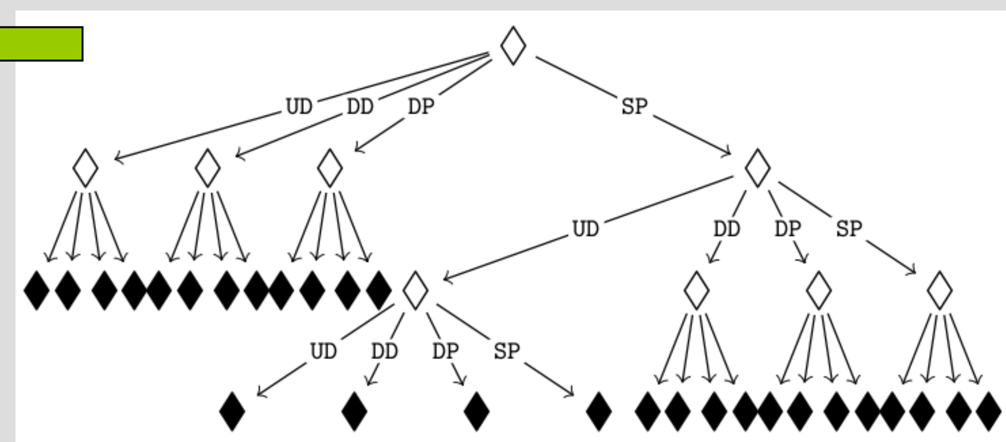


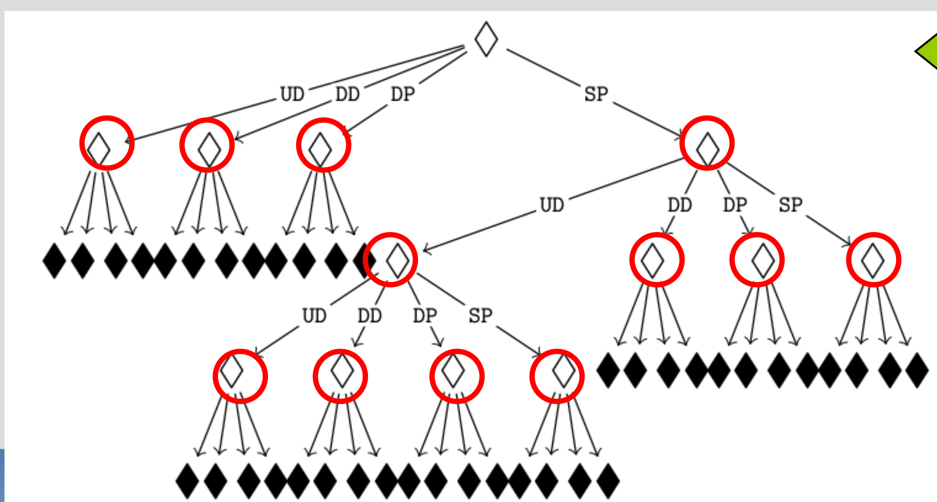
Reuse tree on our example





- 52 Membership Queries
- Saved 12 Resets

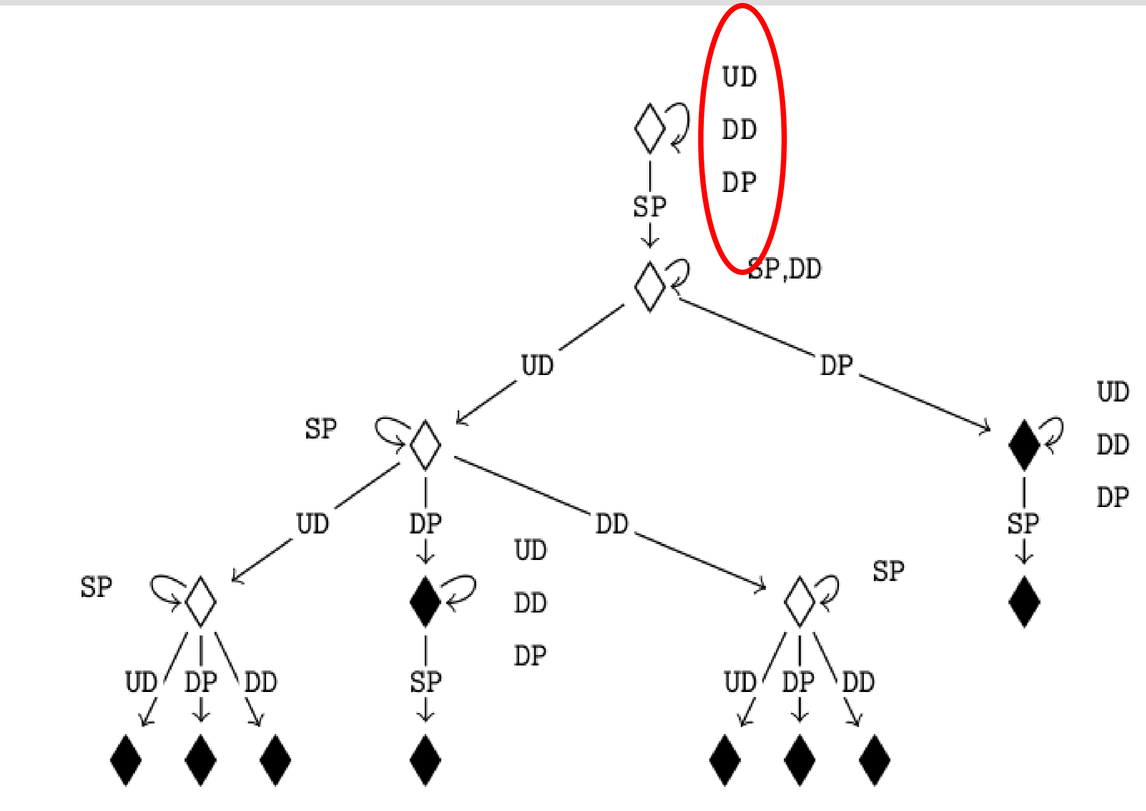




Exploitation: failure invariance

- Domainknowledge
- Failing actions due to missing permissions
- OCS is transaction secure

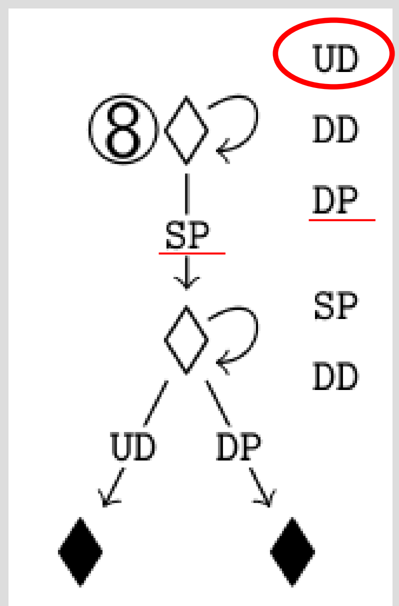
  (**roll back** in case of error)
- Partition output alphabet into successful and failed execution
- Reflexive edges indicate failure output



Pumping: Unfolding edges

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

	SP	UD	DD	DP
λ	①	②	③	④
SP	⑤	⑥	⑦	⑧
UD				
DD				
DP				



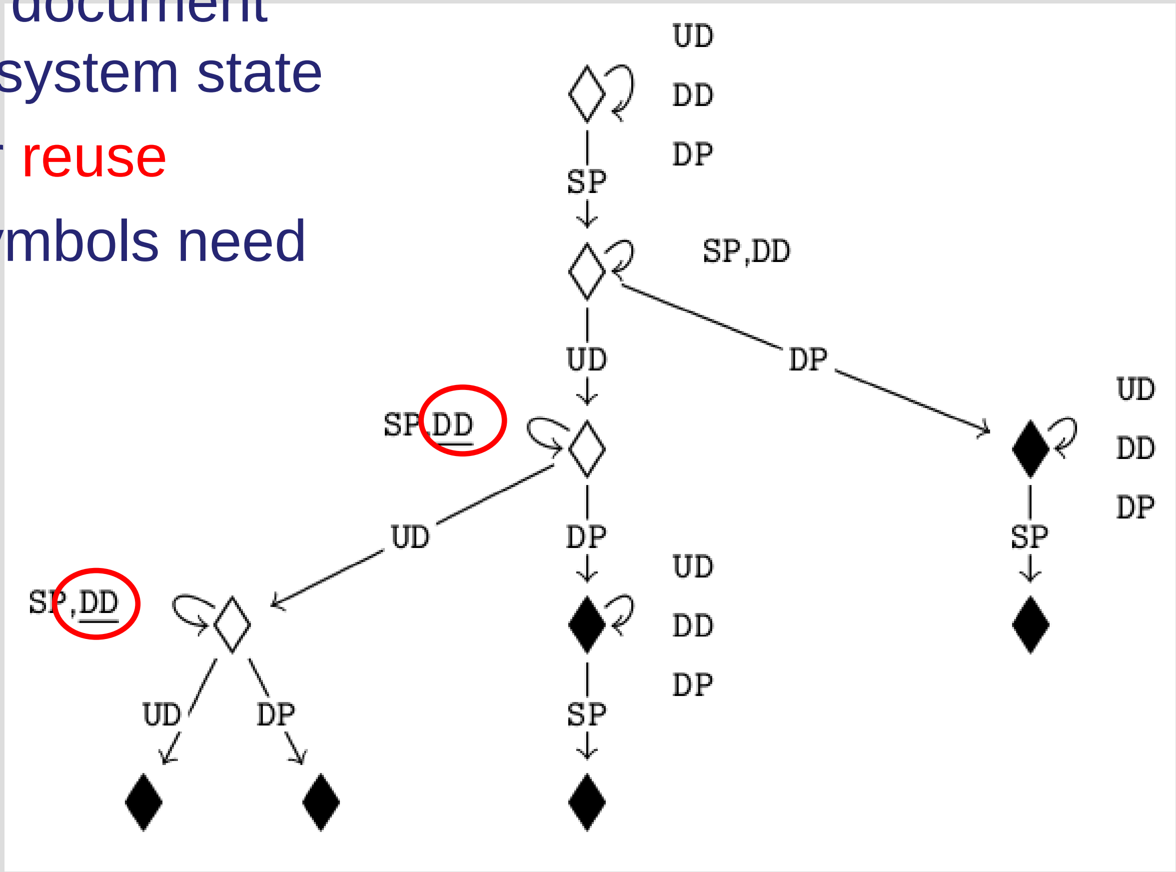
Queries 9 to 20 will be ‘pumped’, e.g.

- **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.

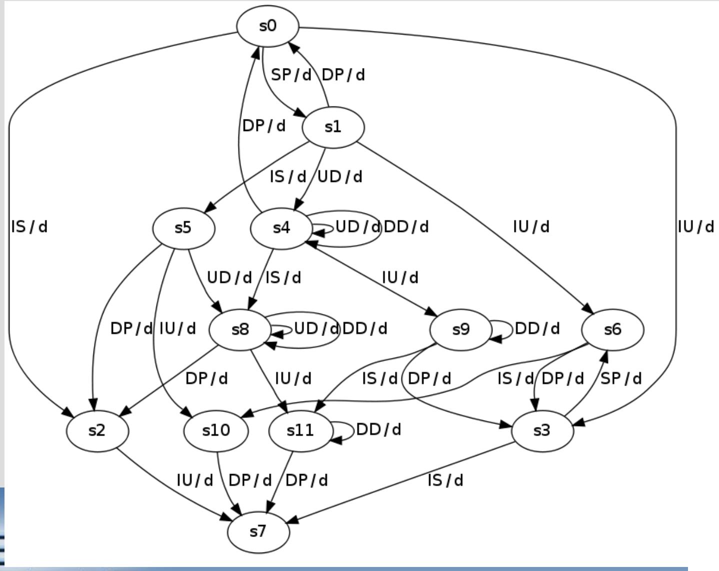
indicates allowed outputs.

Statistics: Learning the OCS

438 MQs containing 1734 Symbols:

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

- a) No reuse
- b) Only direct re-usage
- c) Exploit input knowledge
- d) Exploit output knowledge
- e) Exploit input and output knowledge



Statistics: Learning the OCS

	$\|Q\|$	$\|\Sigma\|$	*MQs*	Resets	Reuses	Pumped
(a)	3	3	30	21	9	0
(b)	11	5	280	31	84	165
(c)	11	5	280	17	78	185
(d)	40	9	3882	137	757	2988
(e)	66	13	12210	646	2514	9050
(f)	160	18	56284	5598	12168	38518

MQs = Resets + Reuses + Pumped

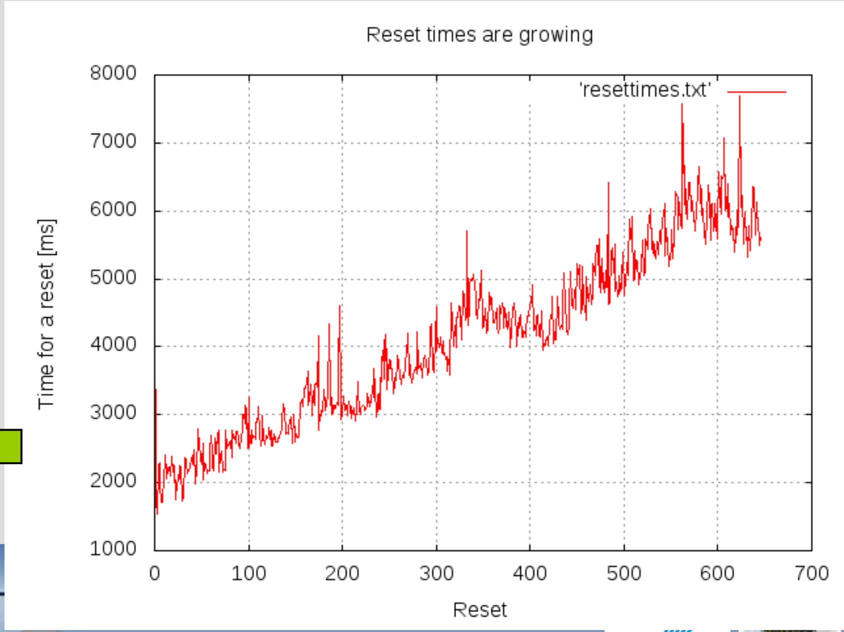
- a) Only direct reusage
- b) Exploit of failure outputs
- c) Like b) but one input marked as invariant
- d)-f) failure outputs and invariant for growing learn alphabets

Statistics: Learning the OCS

	Resets	Avg. reset	Acc. reset	Observed runtime
(a)	21	1.8s.	56s.	40s.
(b)	31	2.3s.	10m	1m 25s.
(c)	17	2.1s.	9m	48s.
(d)	137	3.2s.	3h 27m	10m 30s
(e)	646	4.1s.	13h 52m	53m 50s
(f)	5598	12.7s.	over 8 days	≈ 22h

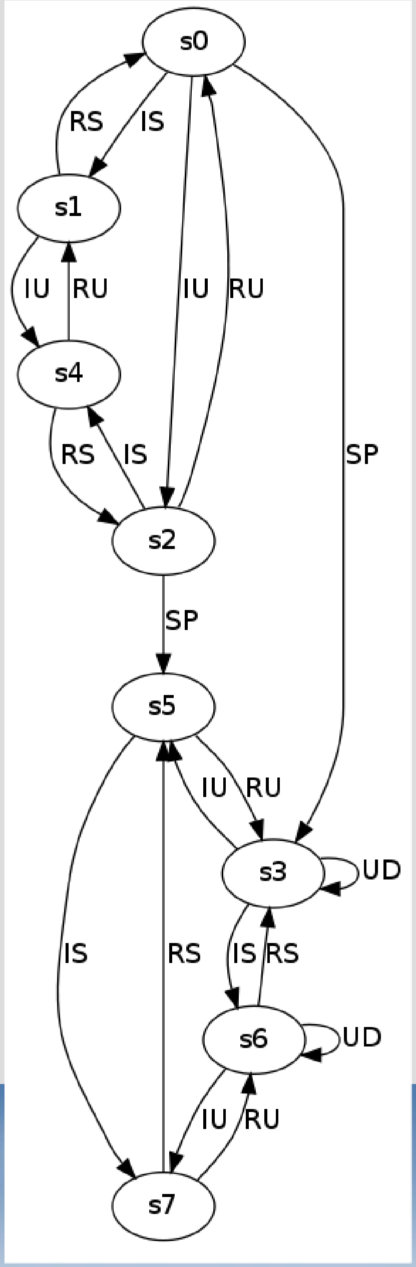
- Reset times are growing
- Observed runtime included execution of input symbols

Accumulated reset time is highly **optimistic**!

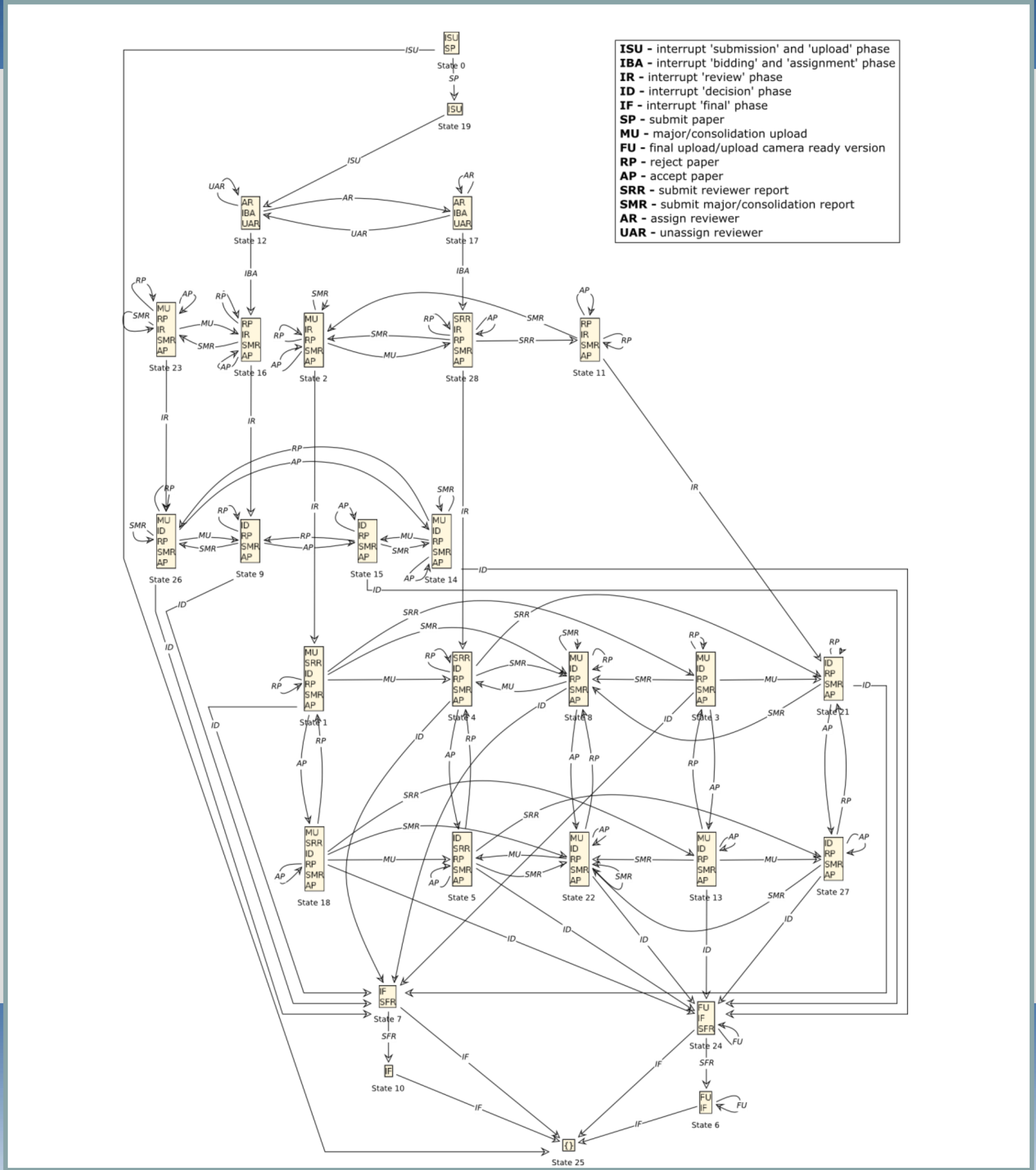


Simple User Process

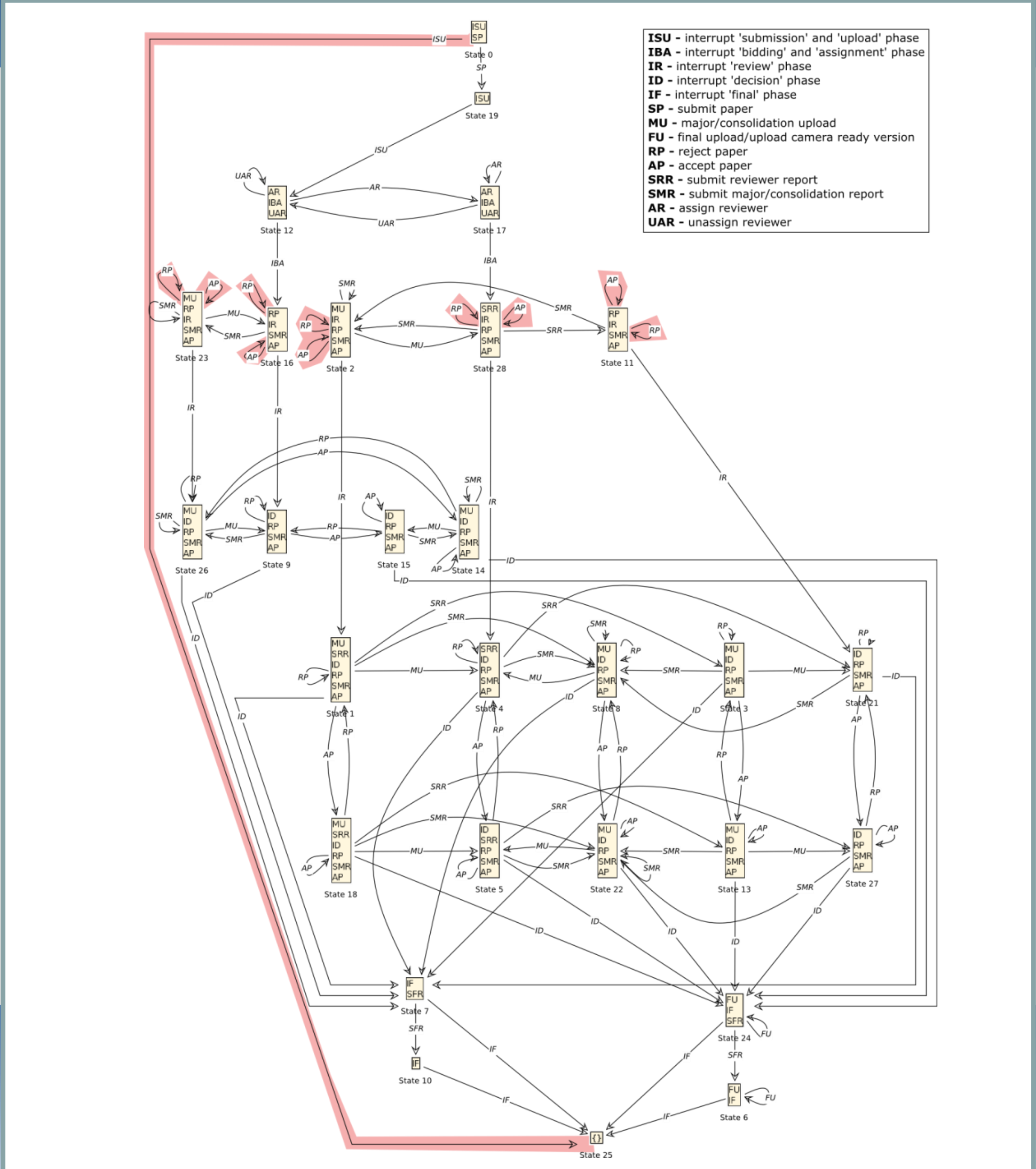
SP: Submit Paper
UD: Upload Document
IS: Interrupt Submission
IU: Interrupt Upload
RS: Restart Submission
RU: Restart Upload



Learned automaton



Verification



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' !