LABORATOIRE D'INFORMATIQUE
DE NANTES ATLANTIQUE

UMR CNRS 6241
Université de Nantes
Ecole des Mines de Nantes

Grammatical inference

Colin de la Higuera
University of Nantes

PASCAL
Pattern Analysis, Statistical Modelling and Computational Learning

dépasser les frontières

Région
PAYS DE LA LOIRE

Acknowledgements

- Laurent Miclet, Jose Oncina, Tim Oates, Anne-Muriel Arigon, Leo Becerra-Bonache, Rafael Carrasco, Paco Casacuberta, Pierre Dupont, Rémi Eyraud, Philippe Ezequel, Henning Fernau, Jean-Christophe Janodet, Satoshi Kobayachi, Thierry Murgue, Frédéric Tantini, Franck Thollard, Enrique Vidal, Menno van Zaanen,...

http://pages-perso.univ-nantes.fr/~cdlh/

http://videolectures.net/colin_de_la_higuera/

Outline

1. What is learning automata about?
2. A (detailed) introductory example
3. Validation issues
4. Some criteria
5. Learning from an informant
6. Learning from text
7. Learning by observing
8. Learning actively
9. Extensions (PFA, transducers, tree automata)
10. Conclusions

1 Grammatical inference

is about learning a grammar given information about a language

- Information is strings, trees or graphs
- Information can be
  - Text: only positive information
  - Informant: labelled data
  - Actively sought (query learning, teaching)

Above lists are not exclusive

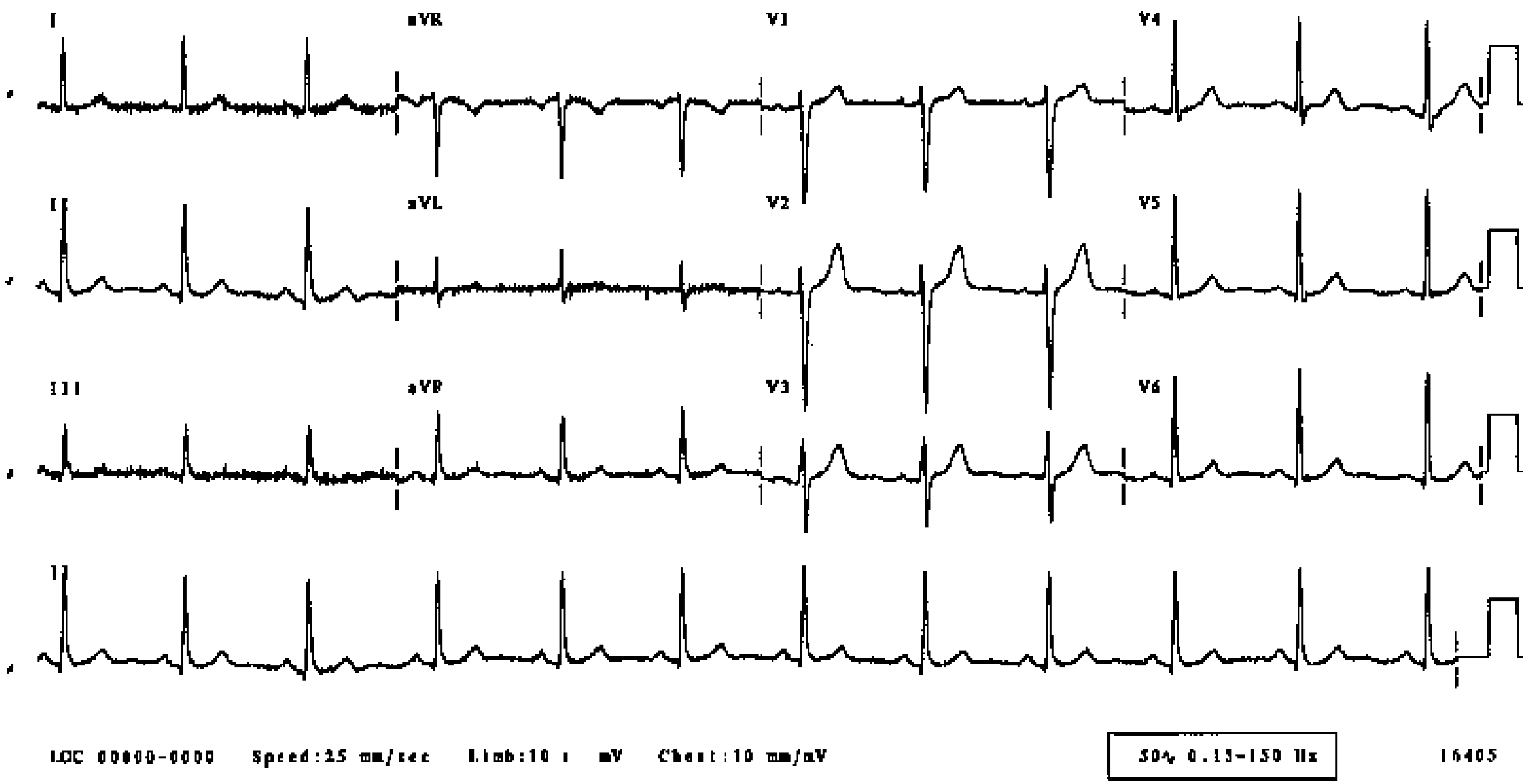
The functions/goals

- Languages and grammars from the Chomsky hierarchy
- Probabilistic automata and context-free grammars
- Hidden Markov Models
- Patterns
- Transducers
- …

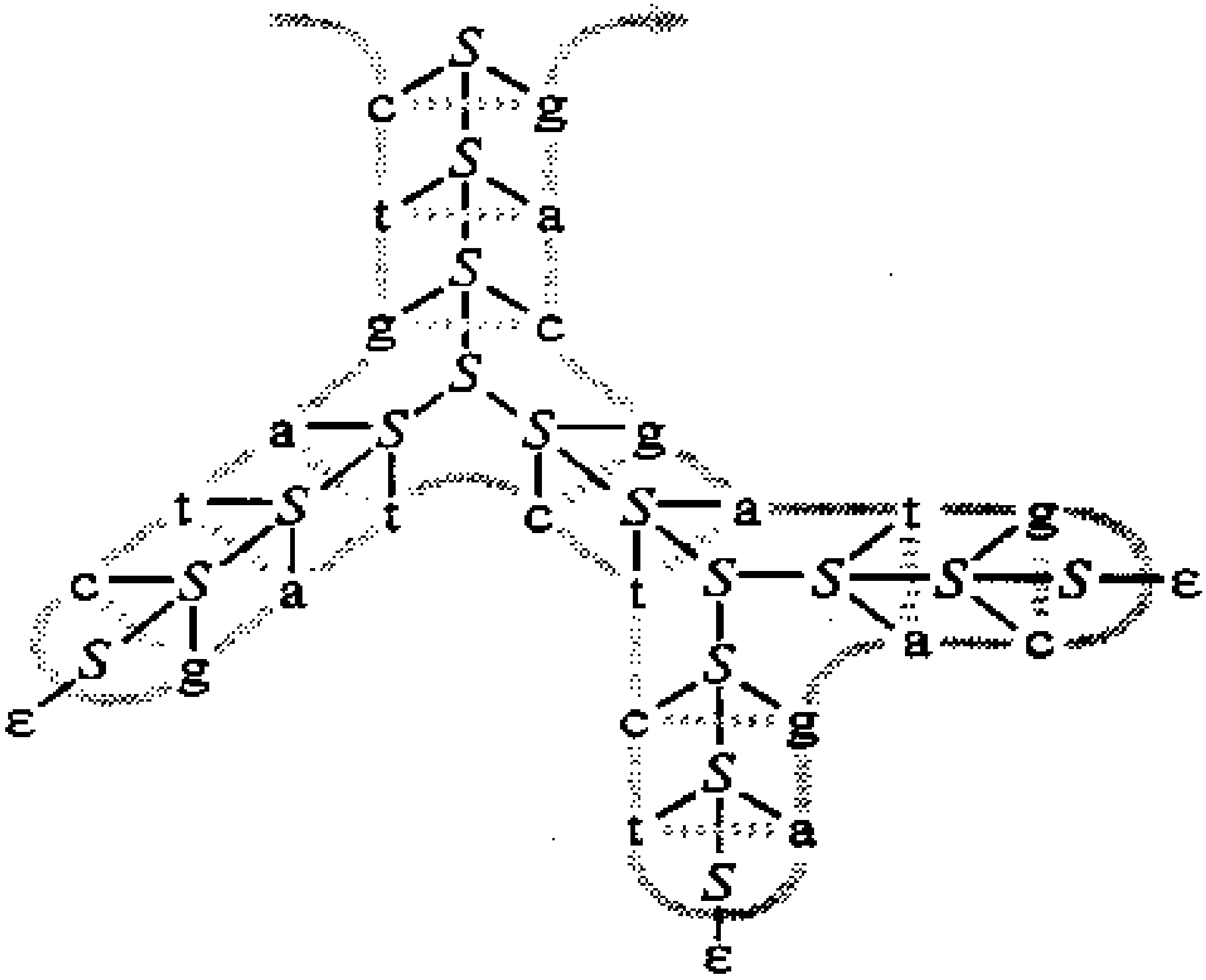
The data: examples of strings

A string in Gaelic and its translation to English:

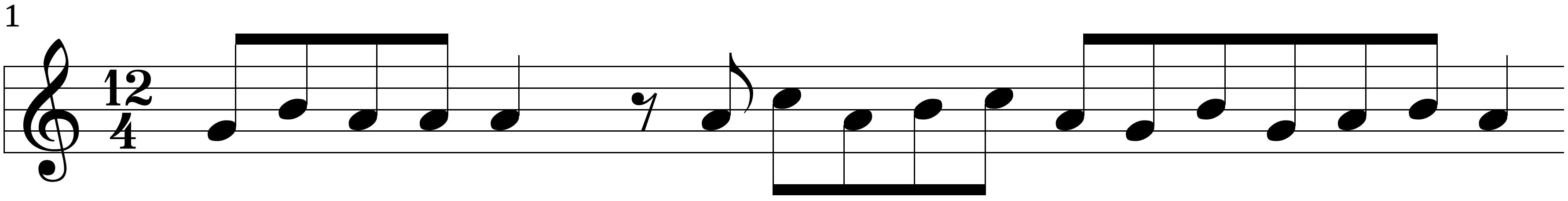
- *Tha thu cho duaichnidh ri èarr àirde de a' coisich deas damh*
- *You are as ugly as the north end of a southward traveling ox*

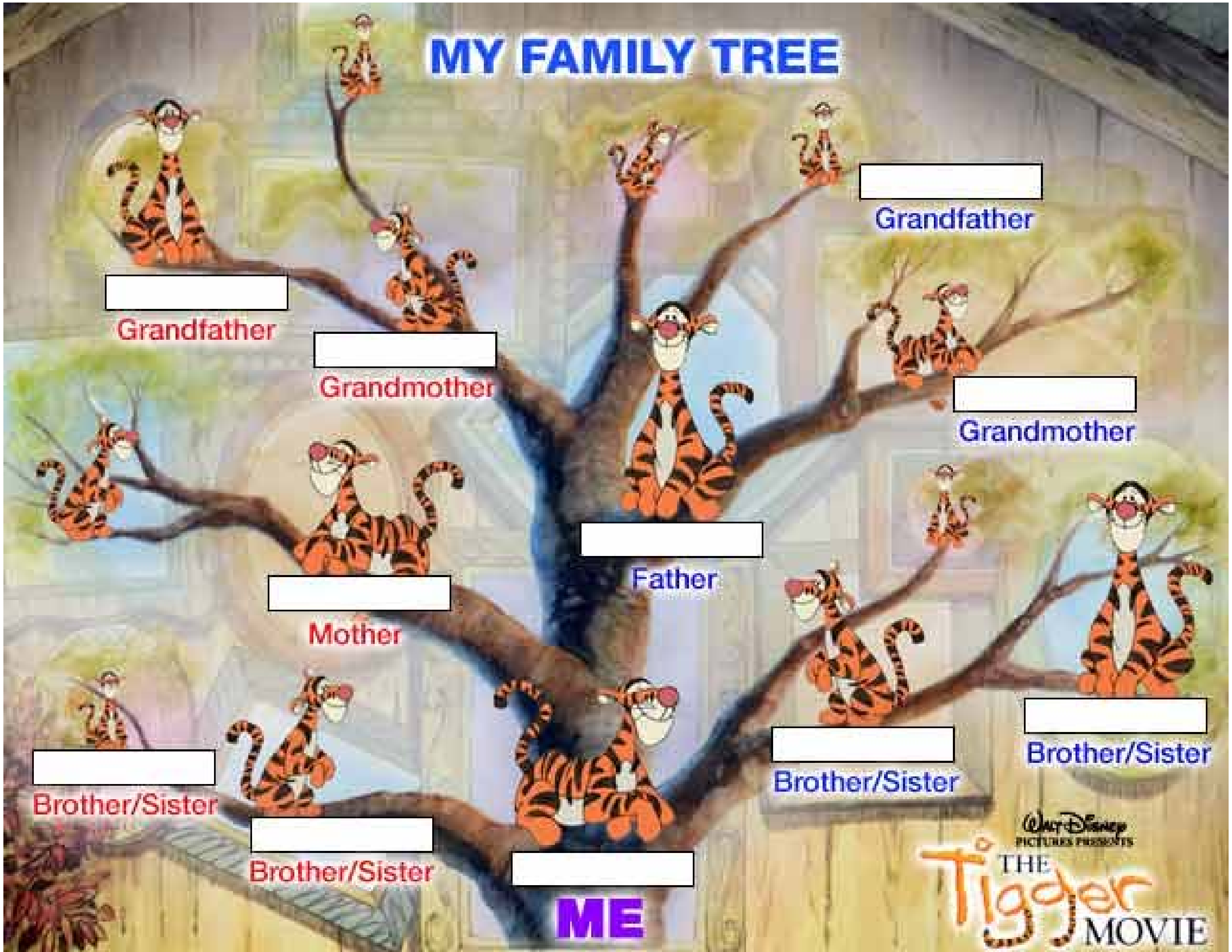


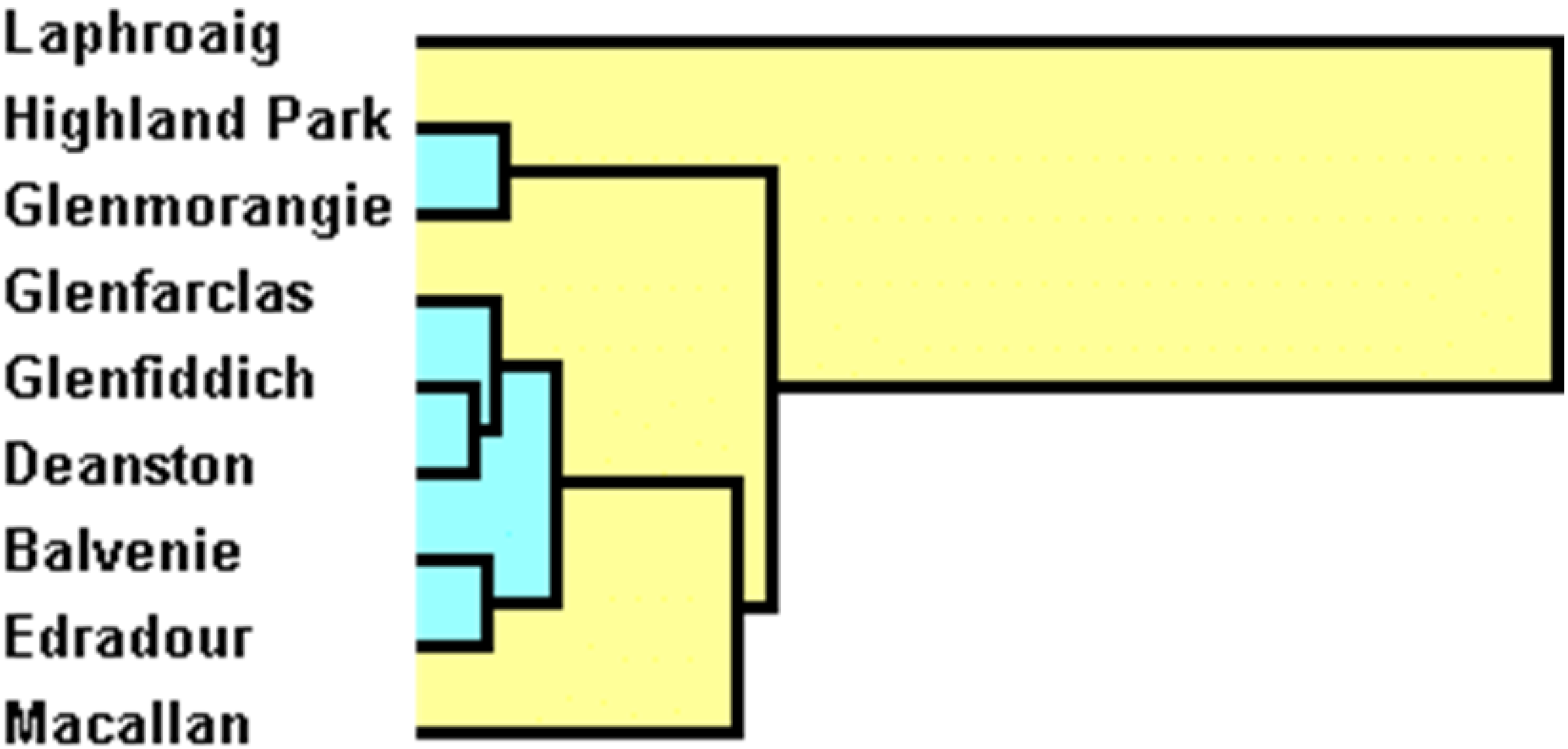
I
aVR
V1
V4
II
aVL
V2
V5
III
aVF
V3
V6
II
Speed:25 mm/sec
Chest:10 mm/mV
0.15-150 Hz
16405



>A BAC=41M14 LIBRARY=CITB_978_SKB
AAGCTTATTCAATAGTTTATTAAACAGCTTCTTAAATAGGATATAAGGCAGTGCCATGTA
GTGGATAAAAGTAATAATCATTATAATATTAAGAACTAATACATACTGAACACTTTCAAT
GGCACTTTACATGCACGGTCCCTTTAATCCTGAAAAAATGCTATTGCCATCTTTATTTCA
GAGACCAGGGTGCTAAGGCTTGAGAGTGAAGCCACTTTCCCCAAGCTCACACAGCAAAGA
CACGGGGACACCAGGACTCCATCTACTGCAGGTTGTCTGACTGGGAACCCCCATGCACCT
GGCAGGTGACAGAAATAGGAGGCATGTGCTGGGTTTGGAAGAGACACCTGGTGGGAGAGG
GCCCTGTGGAGCCAGATGGGGCTGAAAACAAATGTTGAATGCAAGAAAAGTCGAGTTCCA
GGGGCATTACATGCAGCAGGATATGCTTTTTAGAAAAAGTCCAAAAACACTAAACTTCAA
CAATATGTTCTTTTGGCTTGCATTTGTGTATAACCGTAATTAAAAAGCAAGGGGACAACA
CACAGTAGATTCAGGATAGGGGTCCCCTCTAGAAAGAAGGAGAAGGGGCAGGAGACAGGA
TGGGGAGGAGCACATAAGTAGATGTAAATTGCTGCTAATTTTTCTAGTCCTTGGTTTGAA
TGATAGGTTCATCAAGGGTCCATTACAAAAACATGTGTTAAGTTTTTTAAAAATATAATA
AAGGAGCCAGGTGTAGTTTGTCTTGAACCACAGTTATGAAAAAAATTCCAACTTTGTGCA
TCCAAGGACCAGATTTTTTTAAAATAAAGGATAAAAGGAATAAGAAATGAACAGCCAAG
TATTCACTATCAAATTTGAGGAATAATAGCCTGGCCAACATGGTGAAACTCCATCTCTAC
TAAAAATACAAAAATTAGCCAGGTGTGGTGGCTCATGCCTGTAGTCCCAGCTACTTGCGA
GGCTGAGGCAGGCTGAGAATCTCTTGAACCCAGGAAGTAGAGGTTGCAGTAGGCCAAGAT
GGCGCCACTGCACTCCAGCCTGGGTGACAGAGCAAGACCCTATGTCCAAAAAAAAAAAAA
AAAAAAAGGAAAAGAAAAAGAAAGAAAACAGTGTATATATAGTATATAGCTGAAGCTCCC
TGTGTACCCATCCCCAATTCCATTTCCCTTTTTTGTCCCAGAGAACACCCCATTCCTGAC
TAGTGTTTTATGTTCCTTTGCTTCTCTTTTTAAAAACTTCAATGCACACATATGCATCCA
TGAACAACAGATAGTGGTTTTTGCATGACCTGAAACATTAATGAAATTGTATGATTCTAT

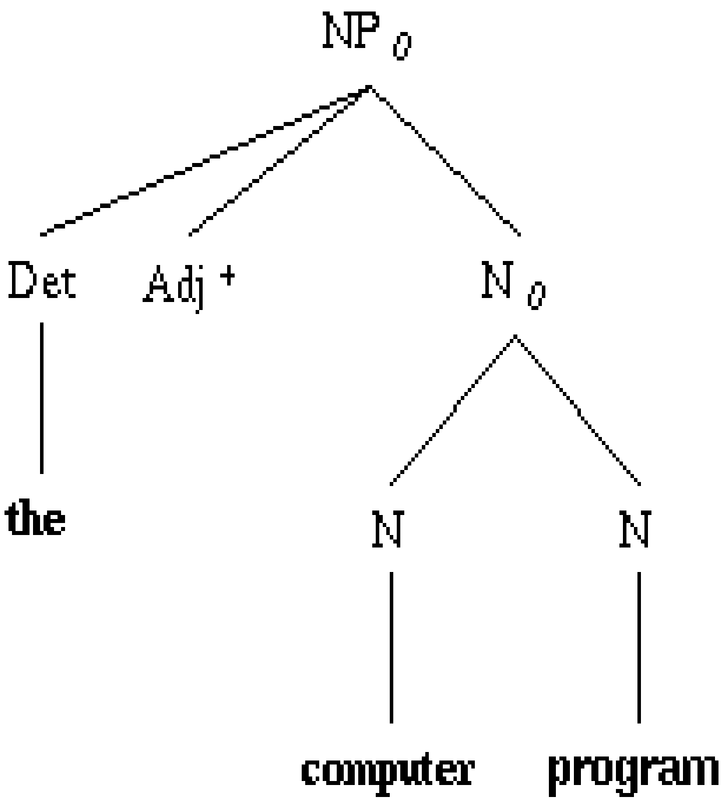

1


MY FAMILY TREE
Grandfather
Grandfather
Grandmother
Grandmother
Father
Mother
Brother/Sister
Brother/Sister
Brother/Sister
Brother/Sister
ME
Walt Disney
PICTURES PRESENTS
THE
Tigger
MOVIE


Laphroaig
Highland Park
Glenmorangie
Glenfarclas
Glenfiddich
Deanston
Balvenie
Edradour
Macallan

```xml
<book>
    <part>
      <chapter>
        <sect1/>
        <sect1>
          <orderedlist numeration="arabic">
            <listitem/>
            <f:fragbody/>
          </orderedlist>
        </sect1>
      </chapter>
    </part>
  </book>
```

```
<?xml version="1.0"?>
<?xml-stylesheet href="carmen.xsl" type="text/xsl"?>
<?cocoon-process type="xslt"?>
<!DOCTYPE pagina [
<!ELEMENT pagina (titulus?, poema)>
<!ELEMENT titulus (#PCDATA)>
<!ELEMENT auctor (praenomen, cognomen, nomen)>
<!ELEMENT praenomen (#PCDATA)>
<!ELEMENT nomen (#PCDATA)>
<!ELEMENT cognomen (#PCDATA)>
<!ELEMENT poema (versus+)>
<!ELEMENT versus (#PCDATA)>
]>
<pagina>
<titulus>Catullus II</titulus>
<auctor>
<praenomen>Gaius</praenomen>
<nomen>Valerius</nomen>
<cognomen>Catullus</cognomen>
</auctor>
```



[NP {subs 0}
[Det [{bold the}]]
[Adj {sups 8 +}]
[{norm12 N}{subs 0}
[N [{bold computer}]]
[N [{sans program}]]]]

And also

- Business processes
- Bird songs
- Images (contours and shapes)
- Robot moves
- Web services
- Malware
- ...

2 An introductory example

- D. Carmel and S. Markovitch. Model-based learning of interaction strategies in multi-agent systems. *Journal of Experimental and Theoretical Artificial Intelligence*, 10(3):309–332, 1998
- D. Carmel and S. Markovitch. Exploration strategies for model-based learning in multiagent systems. *Autonomous Agents and Multi-agent Systems*, 2(2):141–172, 1999

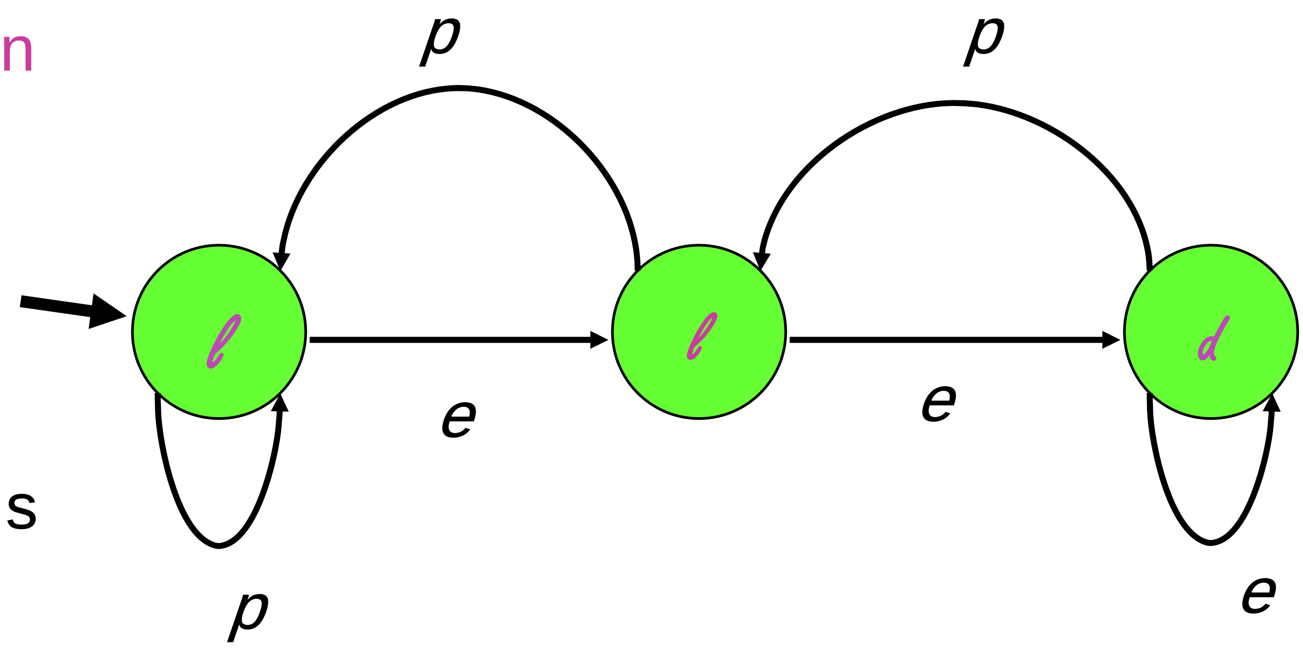
The problem:

- An agent must take cooperative decisions in a multi-agent world
- His decisions will depend:
  - on what he hopes to win or lose
  - on the actions of other agents

Hypothesis: the opponent follows a rational strategy (given by a *DFA*/Moore machine):

You: listen or doze

Me: equations or pictures



$e \ e \ p \ e \ p \quad \rightarrow \ \ell$

$e \ e \ e \quad \rightarrow \ d$

Example: (the prisoner's dilemma)

- Each prisoner can admit (*a*) or stay silent (*s*)

- If both admit: 3 years each
- If A admits but not B: A=0 years, B=5 years
- If B admits but not A: B=0 years, A=5 years
- If neither admits: 1 year each

A \ B	*a*	*s*
a	-3 / -3	0 / -5
s	-5 / 0	-1 / -1

- Here an iterated version against an opponent that follows a rational strategy
- Gain Function: limit of means
- A game is a string in

  $(\text{His\_moves} \times \text{My\_moves})^{*}!$

Example [*as*] [*as*] [*ss*] [*aa*]

The general problem

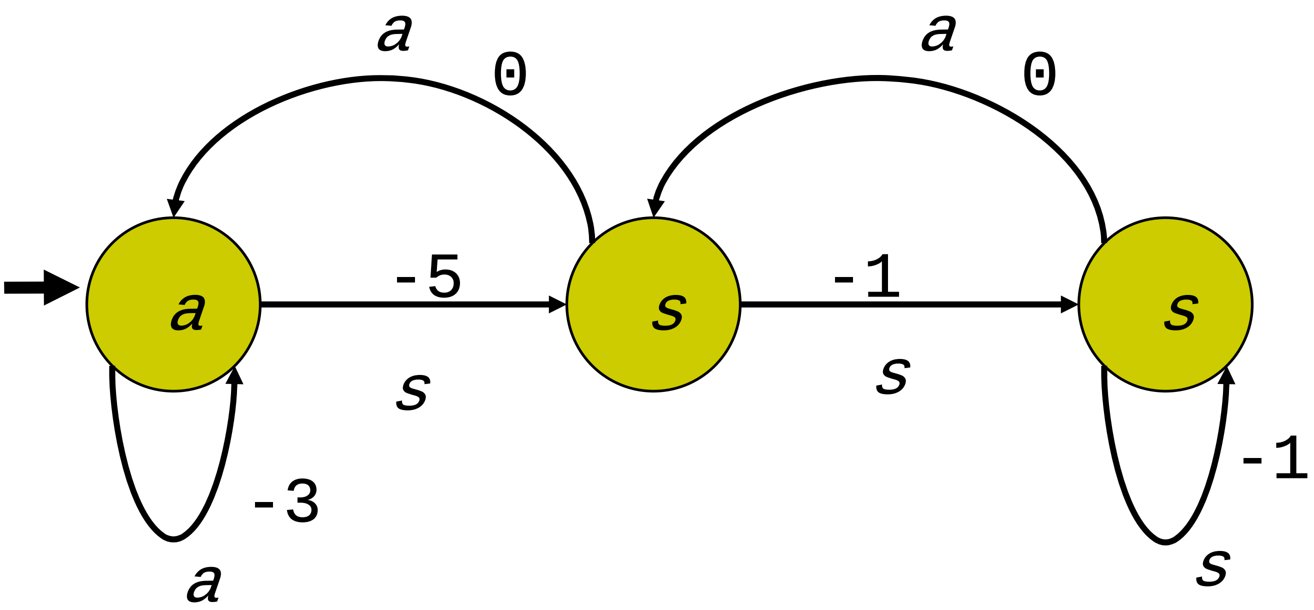
- We suppose that the strategy of the opponent is given by a deterministic finite automaton

- Can we imagine an optimal strategy?

Suppose we know the opponent's strategy:

- Then (game theory):
- Consider the opponent's graph in which we value the edges by *our own gain*

	a	*s*
a	-3	0
s	-5	-1

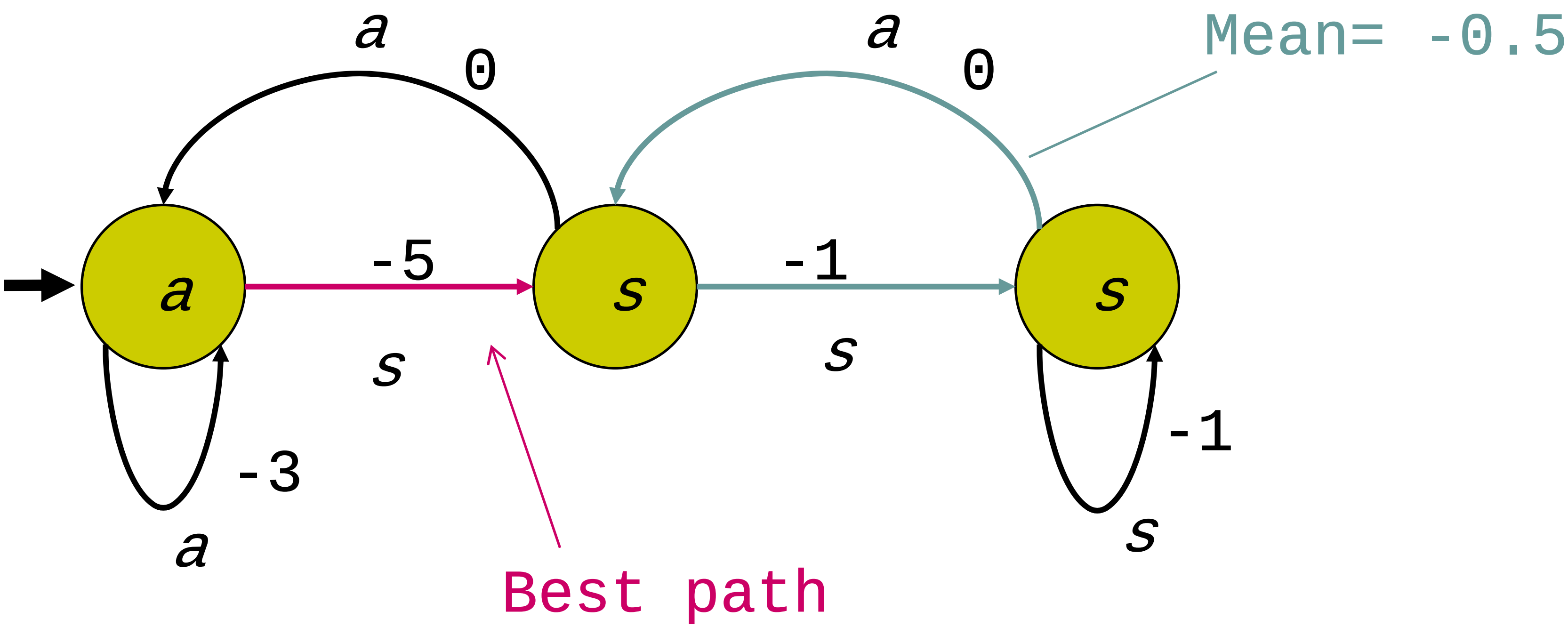


1 Find the cycle of maximum mean weight

2 Find the best path leading to this cycle of maximum mean weight

3 Follow the path and stay in the cycle

	a	*s*
a	-3	0
s	-5	-1



Question

- Can we play a game against this opponent and…

- can we reconstruct his strategy ?

Data *(him, me)*: {*aa as sa aa as ss ss ss sa*}

HIM	ME
a	*a*
a	*s*
s	*a*
a	*a*
a	*s*
s	*s*
s	*s*
s	*a*
s	*a*

I play *asa*,
his move is *a*

$\lambda \rightarrow a$

$a \rightarrow a$

$as \rightarrow s$

$asa \rightarrow a$

$asaa \rightarrow a$

$asaas \rightarrow s$

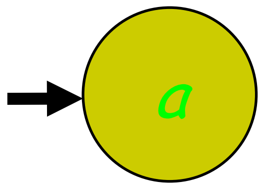
$asaass \rightarrow s$

First move: I play *a*, he plays *a*

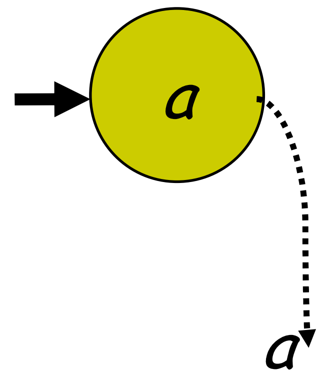
$\lambda \rightarrow a$

$a \rightarrow ?$

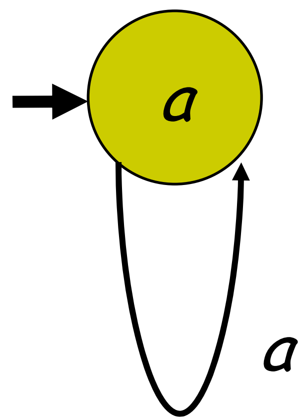
Sure:



Have to deal with:



Try:



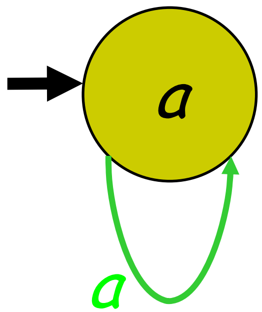
Second move: I play *s*, he plays *a*

$\lambda \rightarrow a$

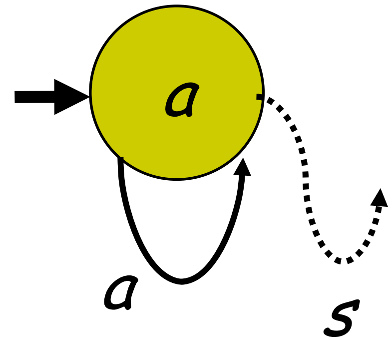
$a \rightarrow a$

$as \rightarrow ?$

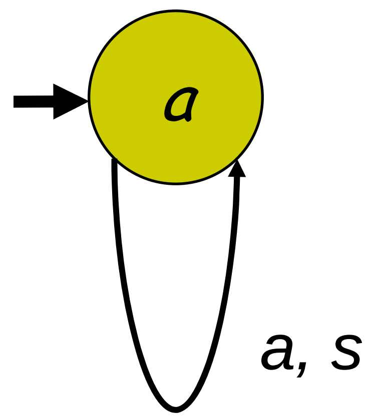
Confirmed:



Have to deal with:



Try:



Third move: I play a, he plays s

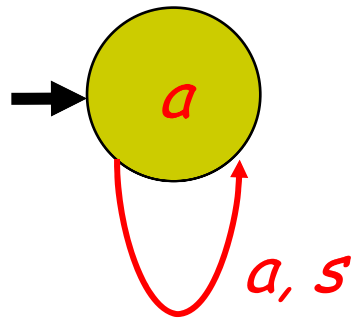
$\lambda \rightarrow a$

$a \rightarrow a$

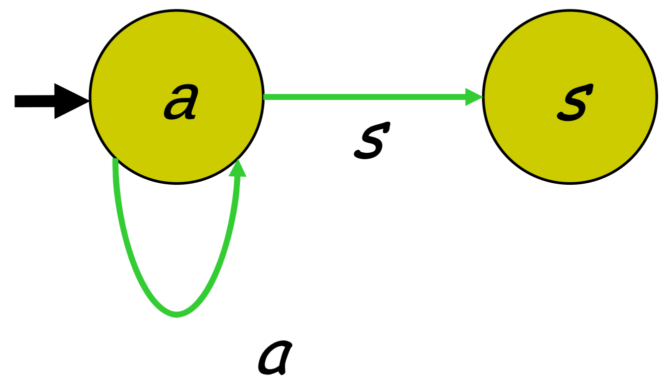
$as \rightarrow s$

$asa \rightarrow ?$

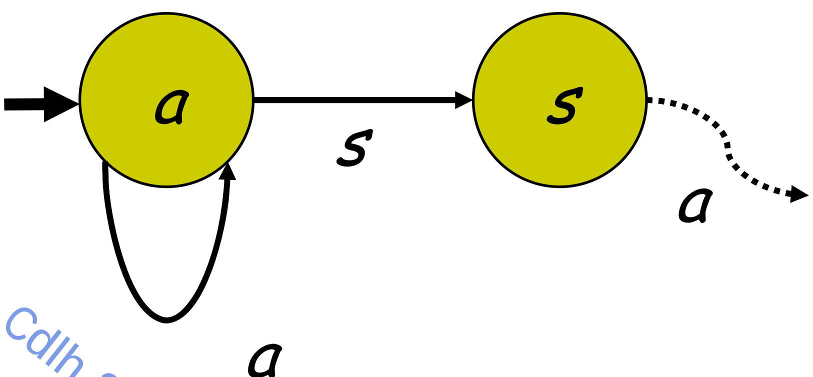
Inconsistent:



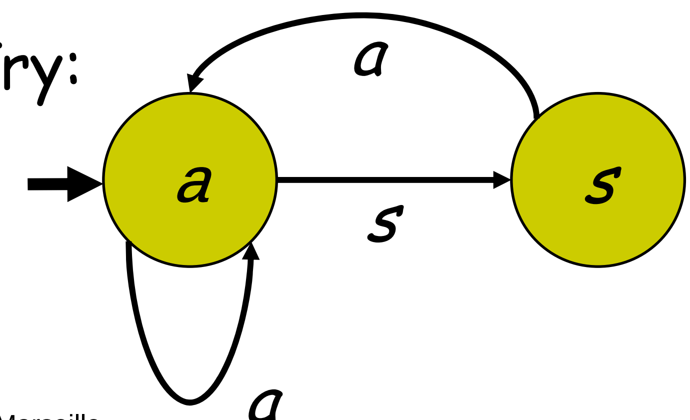
Consistent:



Have to deal with:



Try:



Fourth move: I play *a*, he plays *a*

$\lambda \rightarrow a$

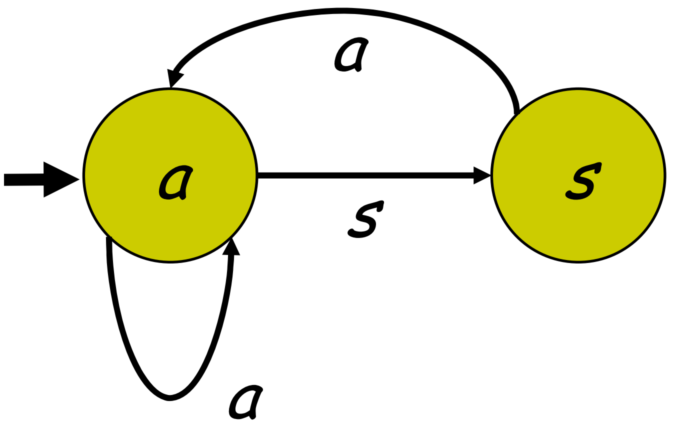
$a \rightarrow a$

$as \rightarrow s$

$asa \rightarrow a$

$asaa \rightarrow ?$

Consistent:



Fifth move: I play *s*, he plays *a*

$\lambda \rightarrow a$

$a \rightarrow a$

$as \rightarrow s$

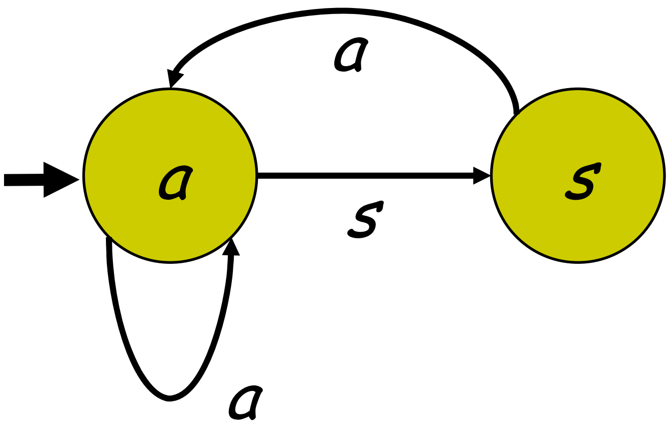
$asa \rightarrow a$

$asaa \rightarrow a$

$asaas \rightarrow ?$

$asaass \rightarrow s$

Consistent:



Sixth move: I play *s*, he plays *s*

$\lambda \to a$

$a \to a$

$as \to s$

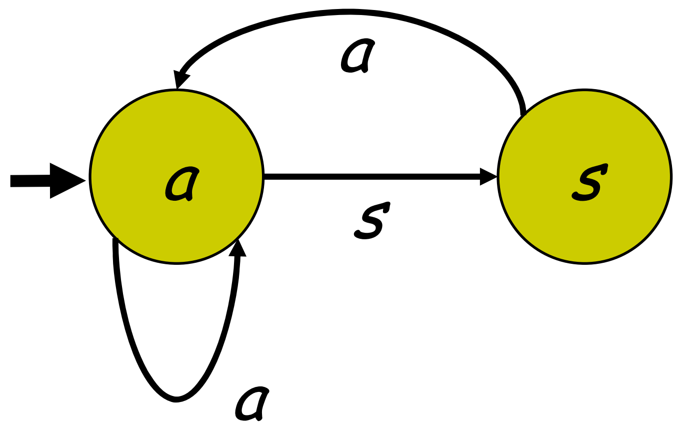
$asa \to a$

$asaa \to a$

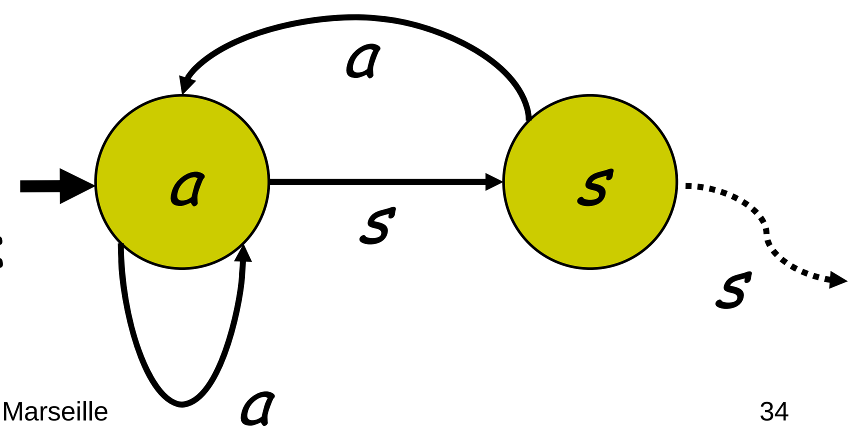
$asaas \to s$

$asaass \to ?$

Consistent:



But have to deal with:



Sixth move: I play s, he plays s

$\lambda \rightarrow a$

$a \rightarrow a$

$as \rightarrow s$

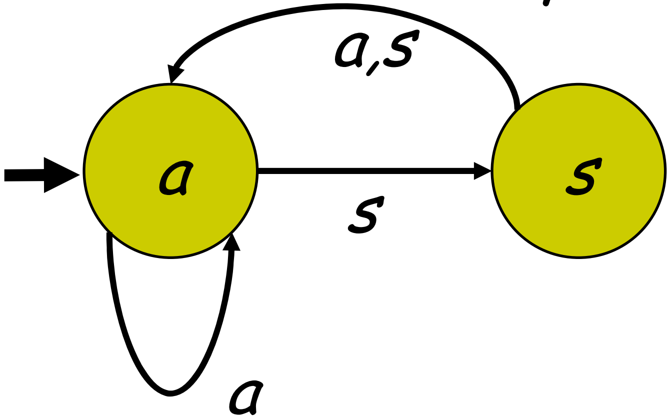
$asa \rightarrow a$

$asaa \rightarrow a$

$asaas \rightarrow s$

$asaass \rightarrow ?$

Try this:



Seventh move: I play *s*, he plays *s*

$\lambda \to a$

$a \to a$

$as \to s$

$asa \to a$

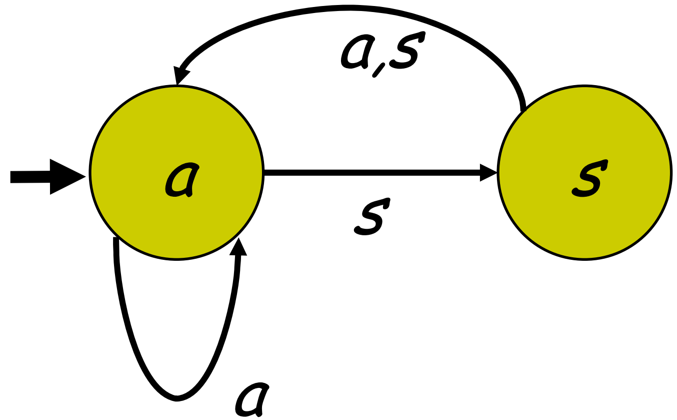
$asaa \to a$

$asaas \to s$

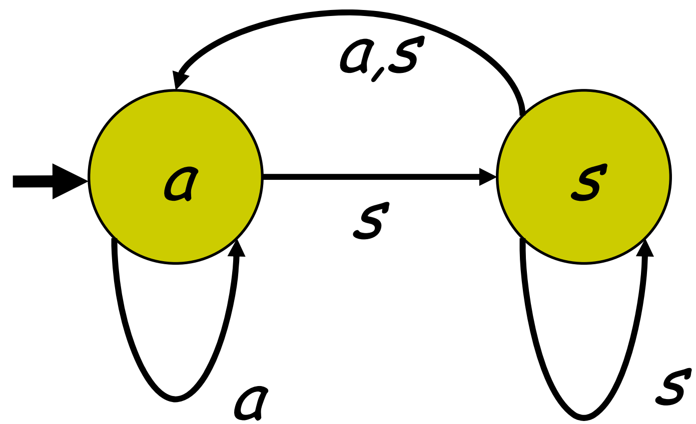
$asaass \to s$

$asaasss \to ?$

Inconsistent:



Consistent:



Eighth move: I play *a*, he plays *s*

$\lambda \rightarrow a$

$a \rightarrow a$

$as \rightarrow s$

$asa \rightarrow a$

$asaa \rightarrow a$

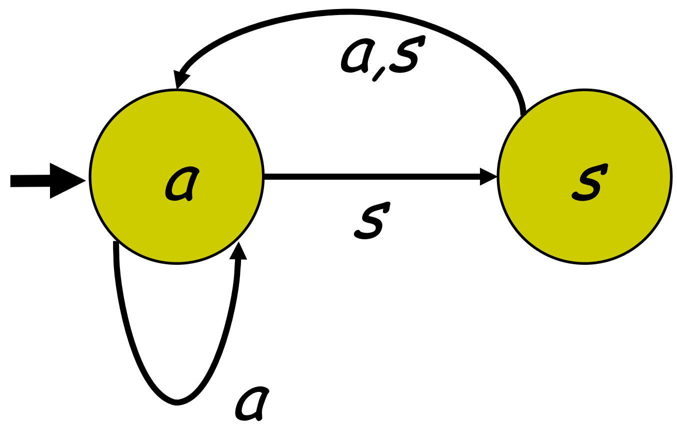
$asaas \rightarrow s$

$asaass \rightarrow s$

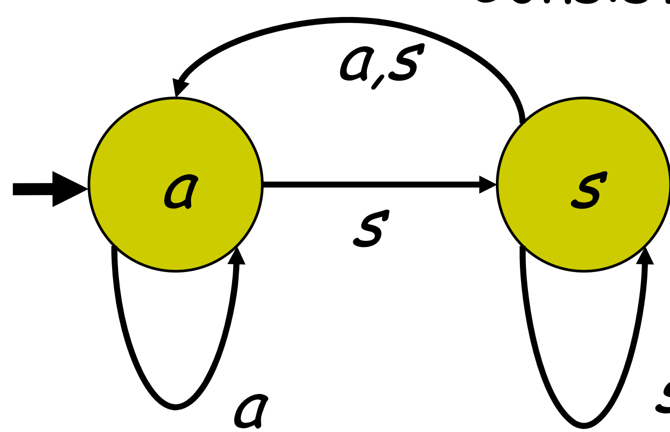
$asaasss \rightarrow s$

$asaasssa \rightarrow ?$

Inconsistent:



Consistent:



Ninth move: I play *a*, he plays *s*

$\lambda \rightarrow a$

$a \rightarrow a$

$as \rightarrow s$

$asa \rightarrow a$

$asaa \rightarrow a$

$asaas \rightarrow s$

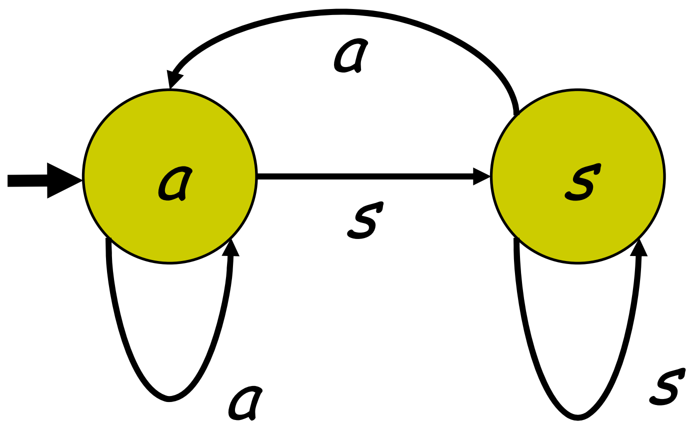
$asaass \rightarrow s$

$asaasss \rightarrow s$

$asaasssa \rightarrow s$

$asaasssas \rightarrow ?$

Inconsistent:



Consistent:

$\lambda \to a$

$a \to a$

$as \to s$

$asa \to a$

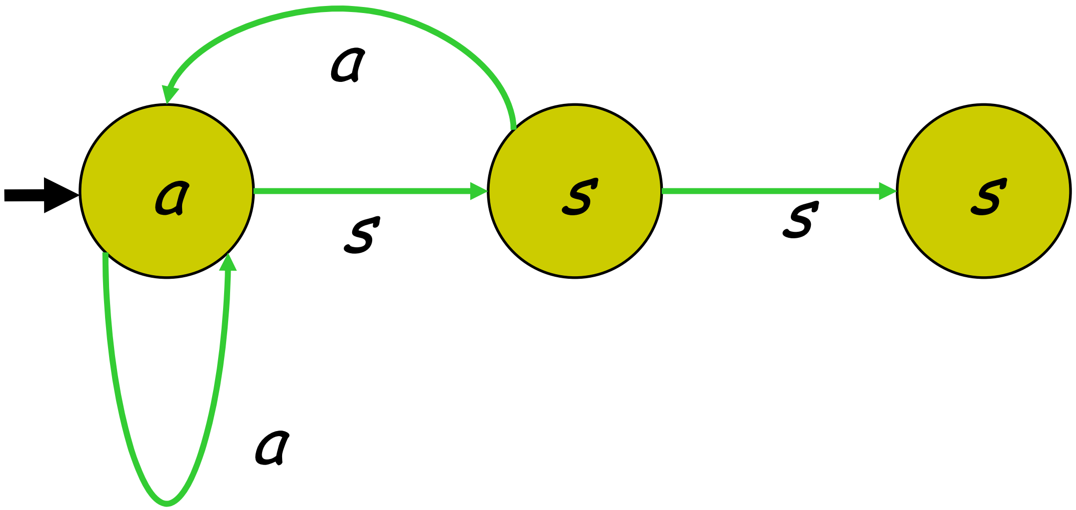
$asaa \to a$

$asaas \to s$

$asaass \to s$

$asaasss \to s$

$asaasssa \to s$



$\lambda \rightarrow a$

$a \rightarrow a$

$as \rightarrow s$

$asa \rightarrow a$

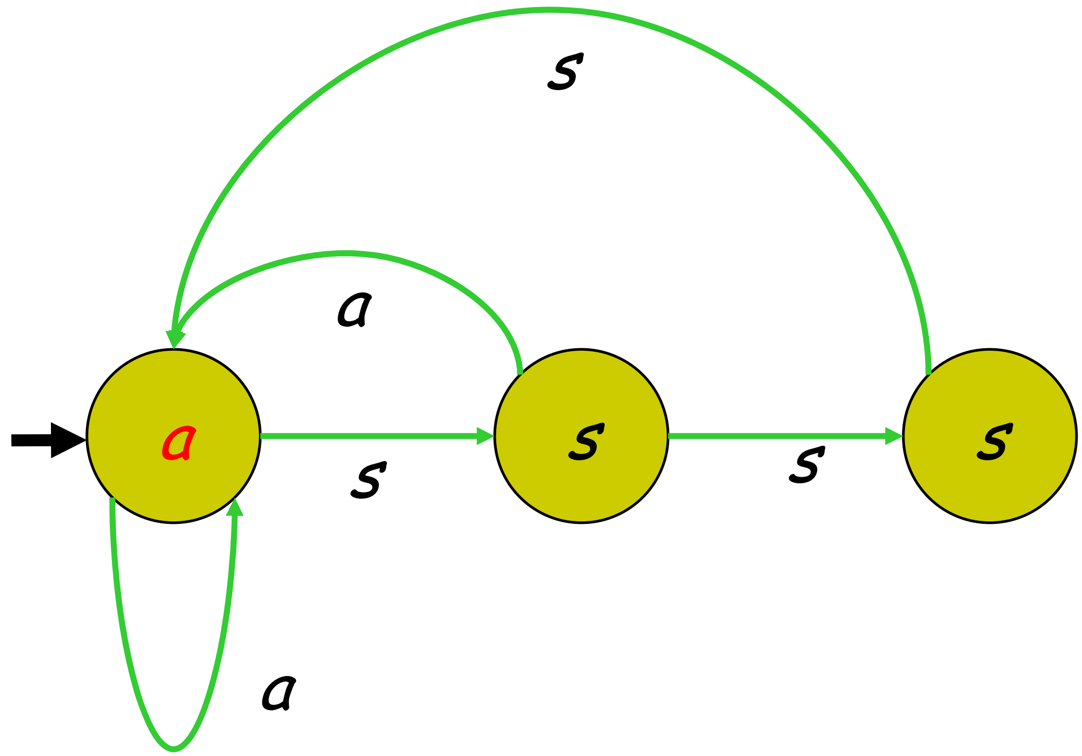
$asaa \rightarrow a$

$asaas \rightarrow s$

$asaass \rightarrow s$

$asaasss \rightarrow s$

$asaasssa \rightarrow s$



Cdlh 2010

$\lambda \rightarrow a$

$a \rightarrow a$

$as \rightarrow s$

$asa \rightarrow a$

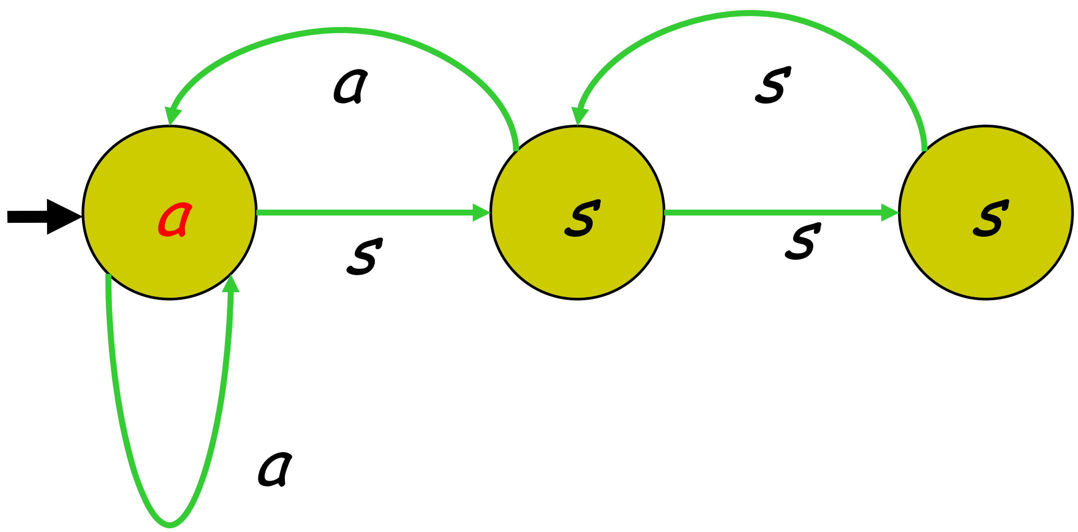
$asaa \rightarrow a$

$asaas \rightarrow s$

$asaass \rightarrow s$

$asaasss \rightarrow s$

$asaasssa \rightarrow s$



$\lambda \rightarrow a$

$a \rightarrow a$

$as \rightarrow s$

$asa \rightarrow a$

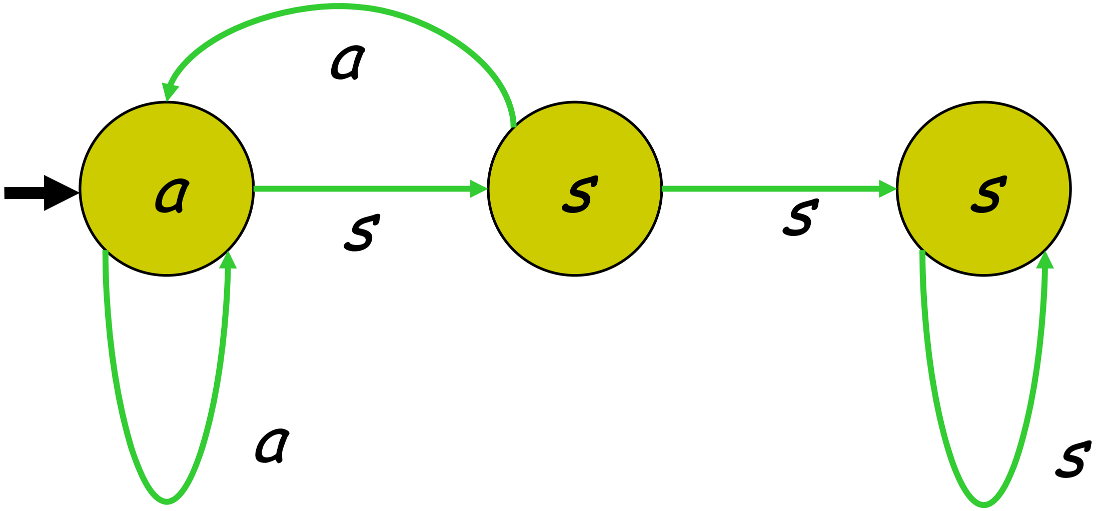
$asaa \rightarrow a$

$asaas \rightarrow s$

$asaass \rightarrow s$

$asaasss \rightarrow s$

$asaasssa \rightarrow s$



$\lambda \to a$

$a \to a$

$as \to s$

$asa \to a$

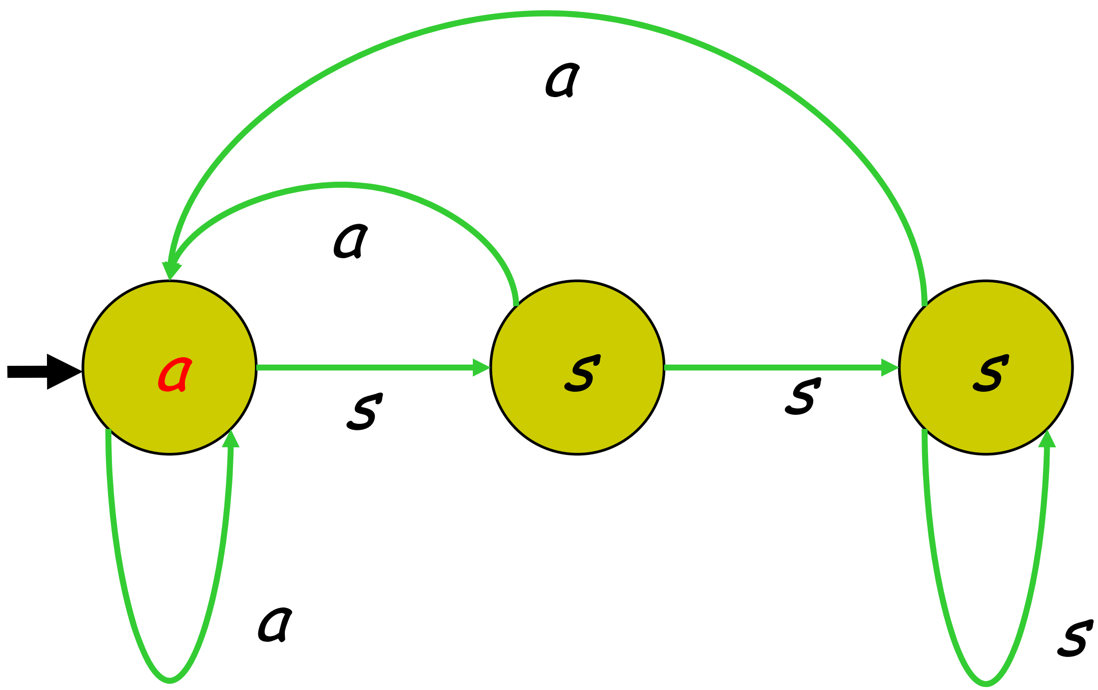
$asaa \to a$

$asaas \to s$

$asaass \to s$

$asaasss \to s$

$asaasssa \to s$



$\lambda \rightarrow a$

$a \rightarrow a$

$as \rightarrow s$

$asa \rightarrow a$

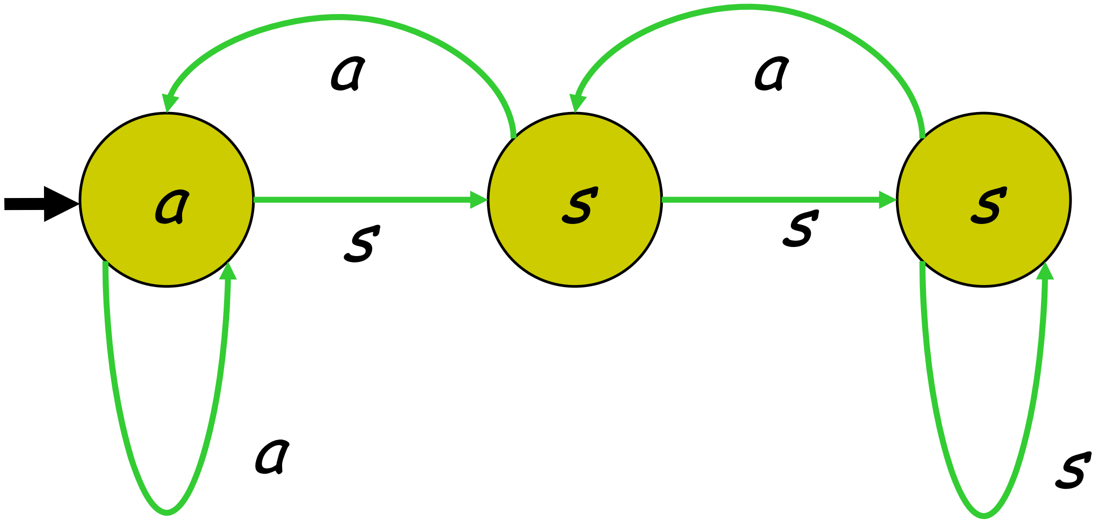
$asaa \rightarrow a$

$asaas \rightarrow s$

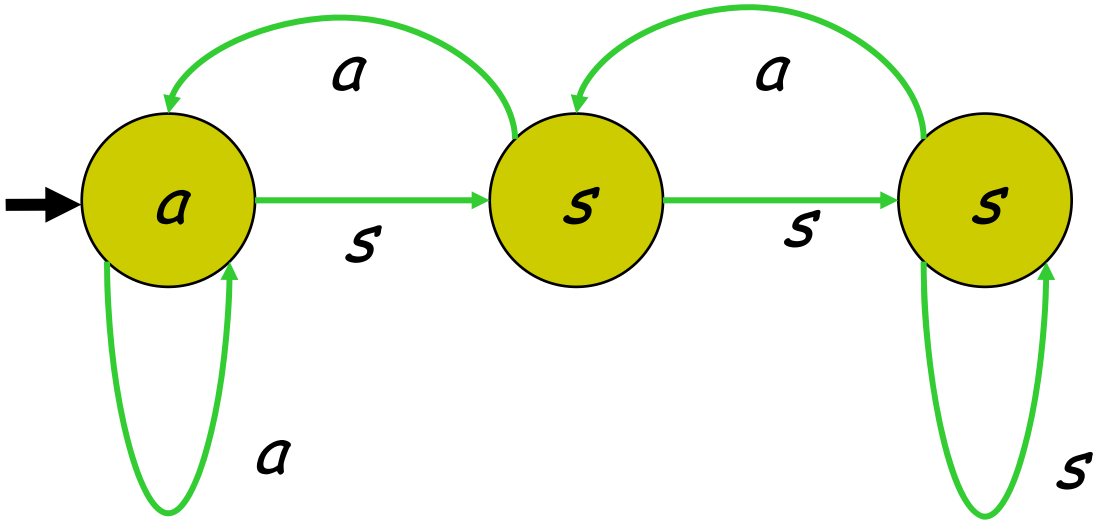
$asaass \rightarrow s$

$asaasss \rightarrow s$

$asaasssa \rightarrow s$



Result



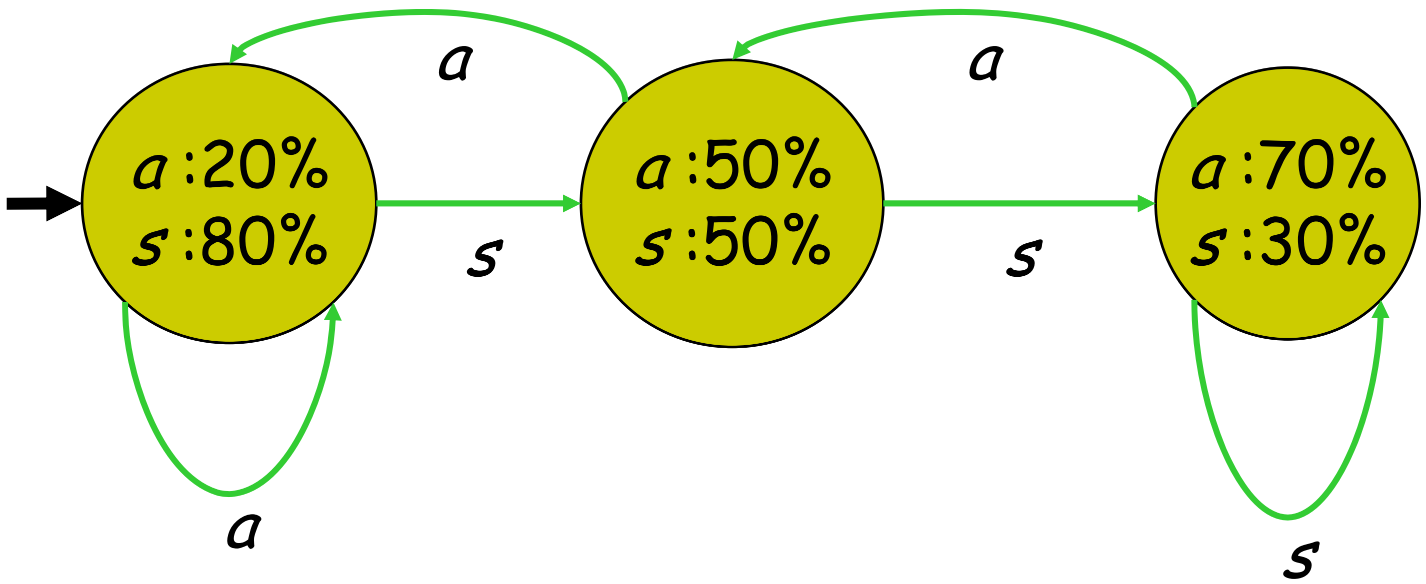
How do we get hold of the learning data?

a) through observation

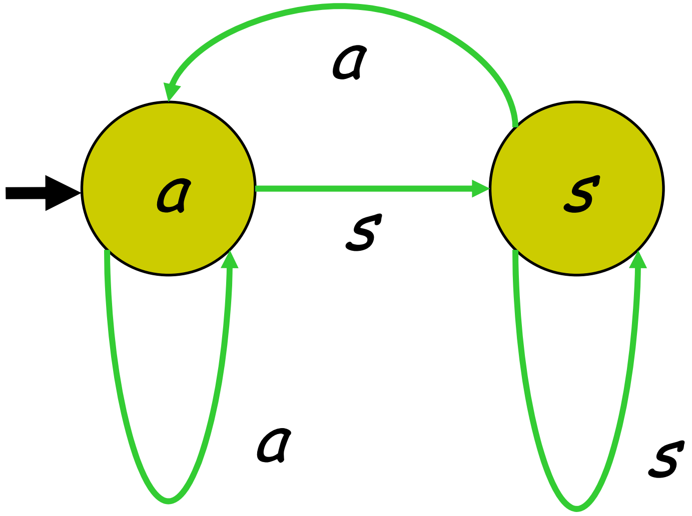
b) through exploration (like here)

An open problem

The strategy is probabilistic:



Tit for Tat



3 What does learning mean?

- Suppose we write a program that can learn FSM… are we done?
- The first question is: « why bother? »
- If my programme works, why do something more about it?
- Why should we do something when other researchers in Machine Learning are not?

Motivating question #1

- Is 17 a random number?
- Is 0110110110110101011000111101 a random sequence?

(Is FSM *A* the correct FSM for sample *S*?)

Motivating question #2

- Statement "I have learnt" does not make sense
- Statement "I am learning" makes sense

Motivating question #3

- In the case of languages, learning is an ongoing process.

- Is there a moment where we can say we have learnt a language?

What usually is called “having learnt”

- That the FSM is the smallest, best (re a score) → Combinatorial characterisation
- That some optimisation problem has been solved
- That the “learning“ algorithm has converged (EM)

What we would like to say

- That having solved some complex combinatorial question we have an Occam, Compression, MDL, Kolmogorov complexity like argument which gives us some guarantee with respect to the future
- Computational learning theory is full of such results

Why should we bother and those working in *statistical machine learning* not?

- Whether with numerical functions or with symbolic functions, we are all trying to do some sort of optimisation
- The difference is (perhaps) that numerical optimisation works much better than combinatorial optimisation!
- [they actually do bother, only differently]
- mbinatorics are harder (in this case) that optimisation

4 Some convergence criteria

- What would we like to say?
- That in the near future, given some string, we can predict if this string belongs to the language or not
- It would be nice to be able to bet €1000 on this

(if not) What would we like to say?

- That if the solution we have returned is not good, then that is because the initial data was bad (insufficient, biased)
- Idea: blame the data, not the algorithm

Suppose we cannot say anything of the sort?

- Then that means that we may be terribly wrong even in a favourable setting
- Thus there is a hidden bias

4.1 Non probabilistic setting

- Identification in the limit
- Resource bounded identification in the limit
- Active learning (query learning)

Example

23

2

3

5

31

7

11 103

{2}

{2, 3}

Fibonacci numbers

Prime numbers

Identification in the limit

- E. M. Gold. Language identification in the limit. *Information and Control*, 10(5):447-474, 1967
- E. M. Gold. Complexity of automaton identification from given data. *Information and Control*, 37:302–320, 1978

The general idea

- Information is presented to the learner who updates its hypothesis after each piece of data
- At some point, always, the learner will have found the correct concept and not move from it

A presentation is

a function $\varphi : \mathbb{N} \rightarrow X$

- where X is some set,
- and such that φ is associated to a language L through a function *yields*: $yields(\varphi) = L$.
- If $\varphi(\mathbb{N}) = \psi(\mathbb{N})$ then $yields(\varphi) = yields(\psi)$

Some types of presentations (1)

- A *text* presentation of a language $L \subseteq \Sigma^*$ is a function $\varphi : \mathbb{N} \rightarrow \Sigma^*$ such that $f(\mathbb{N})=L$
- φ is an infinite succession of all the elements of L
- (note : small technical difficulty with $\emptyset$)

Some types of presentations (2)

- An *informed* presentation (or an informant) of $L \subseteq \Sigma^*$ is a function $\varphi : \mathbb{N} \to \Sigma^* \times \{-,+\}$ such that $\varphi(\mathbb{N})=(L,+)\cup(\overline{L},-)$
- φ is an infinite succession of all the elements of Σ^* labelled to indicate if they belong or not to L

Presentation for $\{a^n b^n : n \in \mathbb{N}\}$

- Legal presentation from text: λ, a^2b^2, a^7b^7…
- Illegal presentation from text: ab, ab, ab,…
- Legal presentation from informant : $(\lambda,+)$, $(abab,-)$, $(a^2b^2,+)$, $(a^7b^7\ldots,+)$, $(aab,-)$,…

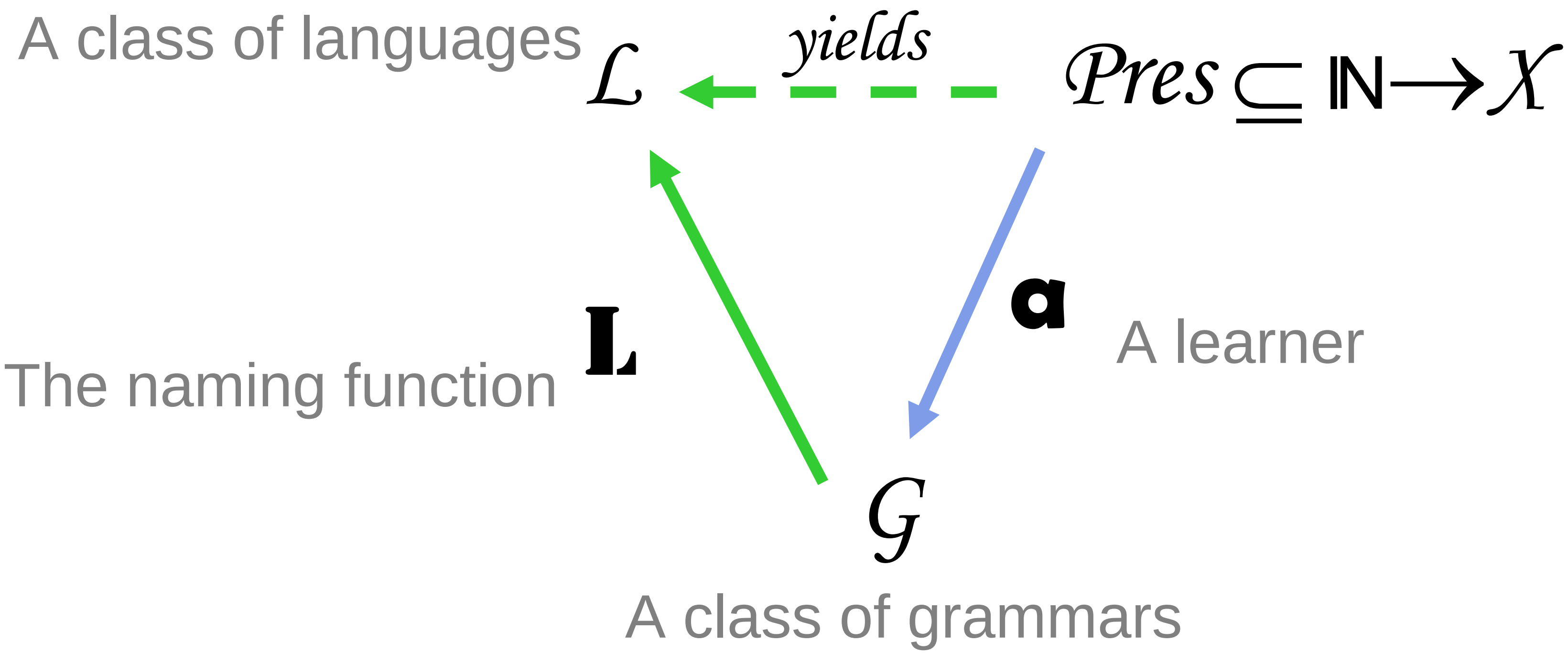
Learning function

- Given a presentation φ, φ_n is the set of the first n elements in f
- A learning algorithm $\mathbf{a}$ is a function that takes as input a set φ_n and returns a representation of a language
- Given a grammar G, $\mathbb{L}(G)$ is the language generated/recognised/ represented by G

Convergence to a hypothesis

- Let L be a language from a class $\mathcal{L}$, let φ be a presentation of L and let φ_n be the first n elements in f,
- **a** *converges* to G with φ if:
  - $\forall n \in \mathbb{N}$: $\mathbf{a}(\varphi_n)$ halts and gives an answer
  - $\exists n_0 \in \mathbb{N}$: $n \geq n_0 \Rightarrow \mathbf{a}(\varphi_n) = G$

Identification in the limit



$\mathbf{L}(\mathfrak{a}(\varphi)) = yields(\varphi)$ $\quad \varphi(\mathbb{N}) = \psi(\mathbb{N}) \Rightarrow yields(\varphi) = yields(\psi)$

Consistency

- We say that the learning function **a** is *consistent* if φ_n is consistent with **a**(φ_n) $\forall n$

- A consistent learner is always consistent with the past

Conservatism

- We say that the learning function $\mathbf{a}$ is *conservative* if whenever $\varphi(n+1)$ is consistent with $\mathbf{a}(\varphi_n)$, we have $\mathbf{a}(\varphi_n)= \mathbf{a}(\varphi_{n+1})$

- A conservative learner doesn't change his mind needlessly

What about efficiency?

- We can try to bound
  - global time
  - update time
  - errors before converging (IPE)
  - mind changes (MC)
  - queries
  - good examples needed

More precise definition of convergence

$\exists n \in \mathbb{N}$ such that $\forall k \geq n$ $\mathbf{L}(\mathbf{a}(\varphi_k)) = \mathbf{L}(\mathbf{a}(\varphi_n)) = yields(\varphi)$

φ_k *is the sequence of the first* k elements of presentation φ

Resource bounded identification in the limit

- Definitions of IPE, CS, MC, update time, etc…
- What should we try to measure?
  - The size of $\mathcal{M}$?
  - The size of $\mathcal{L}$?
  - The size of f ?
  - The size of φ_n ?

4.2 Probabilistic settings

- PAC learning
- Identification with probability 1
- PAC learning distributions

Learning a language from sampling

- We have a distribution over Σ^*
- We sample twice:
  - Once to learn
  - Once to see how well we have learned
- The PAC setting

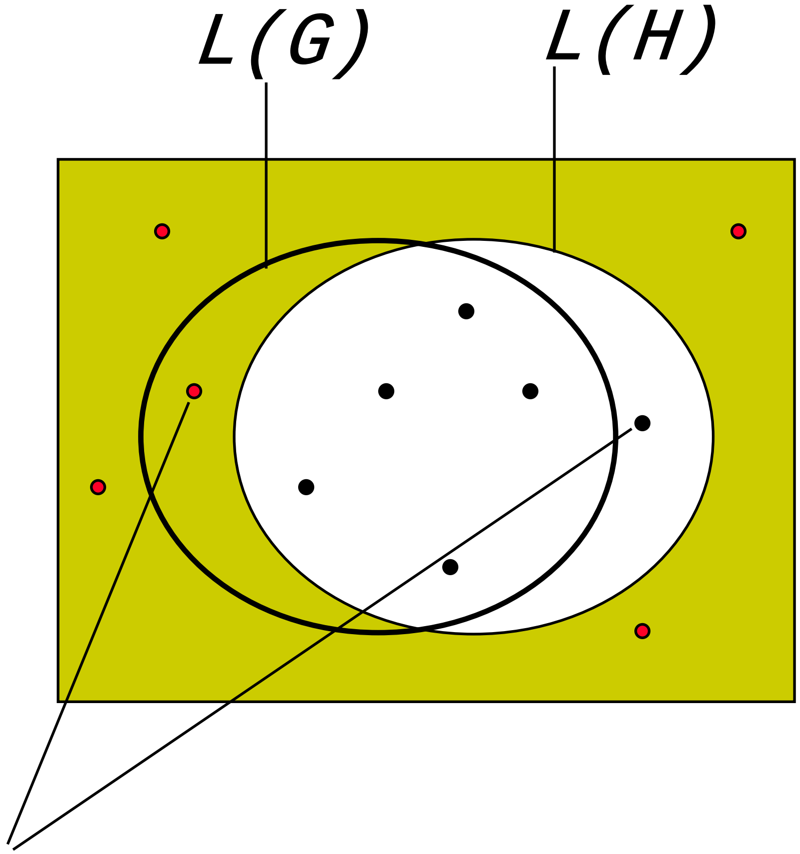
PAC-learning
(Valiant 84, Pitt 89)

- $\mathcal{L}$ a class of languages
- $\mathcal{M}$ a class of machines
- $\varepsilon > 0$ and $\delta > 0$
- m a maximal length over the strings
- n a maximal size of machines

$\mathcal{H}$ is ε -**AC** (approximately correct)*

if

$$Pr_D[\mathcal{H}(x) \neq \mathcal{G}(x)] < \varepsilon$$



Errors: we want $L_1(D(G), D(H)) < \varepsilon$

(French radio)

- Unless there is a surprise there should be no surprise
- (after the last primary elections, on 3rd of June 2008)

Results

- Using cryptographic assumptions, we cannot PAC-learn DFA
- Cannot PAC-learn NFA, CFGs with membership queries either

Alternatively

- Instead of learning classifiers in a probabilistic world, learn directly the distributions!
- Learn probabilistic finite automata (deterministic or not)

No error

- This calls for identification in the limit with probability 1
- Means that the probability of not converging is 0

Results

- If probabilities are computable, we can learn with probability 1 finite state automata
- But not with bounded (polynomial) resources
- Or it becomes very tricky (with added information)

With error

- PAC definition
- But error should be measured by a distance between the target distribution and the hypothesis
- L_1, L_2, L_∞ ?

Results

- Too easy with L_∞
- Too hard with L_1
- Nice algorithms for biased classes of distributions

Conclusion

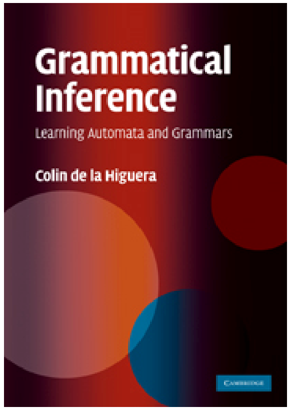
- A number of paradigms to study identification of learning algorithms
- Some to learn classifiers
- Some to learn distributions

5 Learning from an informant

- Algorithm RPNI
- Regular Positive and Negative Grammatical Inference

Inferring regular languages in polynomial time. Jose Oncina & Pedro García. *Pattern recognition and image analysis, 1992*

http://pagesperso.lina.univ-nantes.fr/~cdlh/slides/

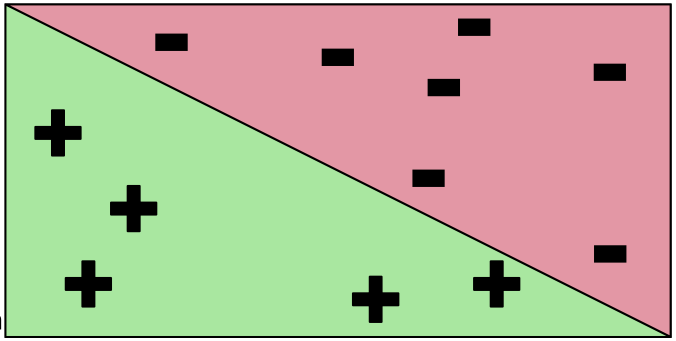


Chapter 12

Motivation

- We are given a set of strings S_+ and a set of strings S_-
- Goal is to build a classifier
- This is a traditional (or typical) machine learning question
- How should we solve it?

Σ^*

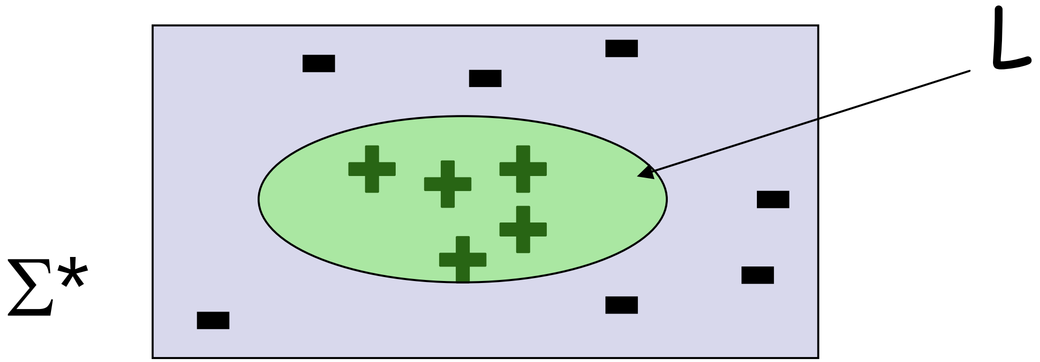


Ideas

- Use a distance between strings and try k-NN
- Embed strings into vectors and use some off-the-shelf technique (decision trees, SVMs, other kernel methods)

Alternative

- Suppose the classifier is some grammatical formalism
- Thus we have L and $\Sigma^*\backslash L$



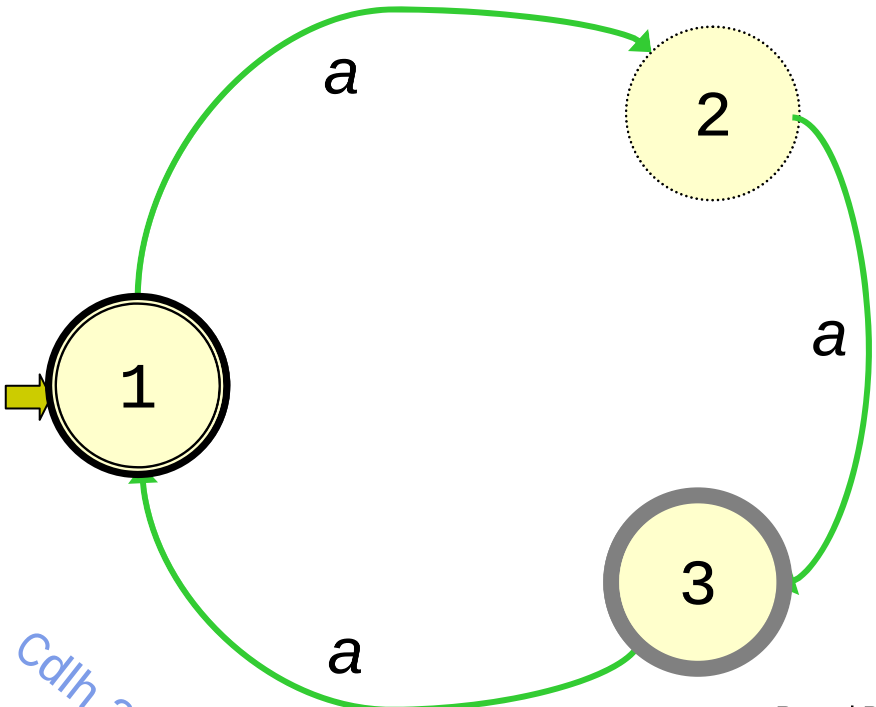
Obviously many possible candidates

- Any Grammar G such that
  - $S_+ \subseteq \mathbf{L}(G)$
  - $S_- \cap \mathbf{L}(G) = \emptyset$

Two types of final states

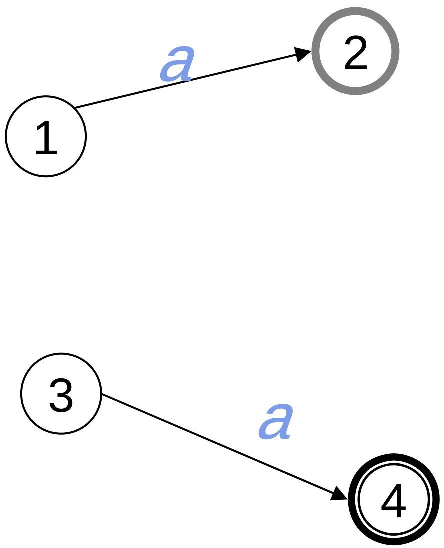
$S_+=\{\lambda,\ aaa\}$

$S_-=\{aa,\ aaaaa\}$

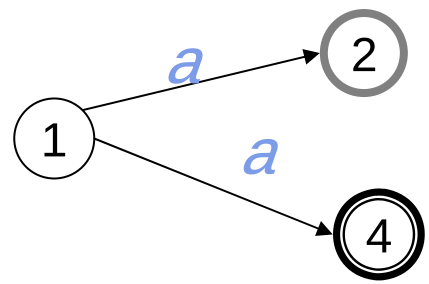


1 is accepting
3 is rejecting
What about state 2?

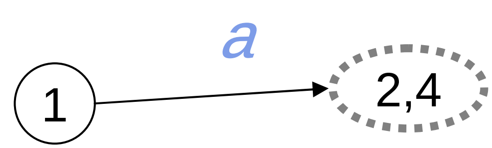
What is determinism about?



Merge 1 and 3?



But…

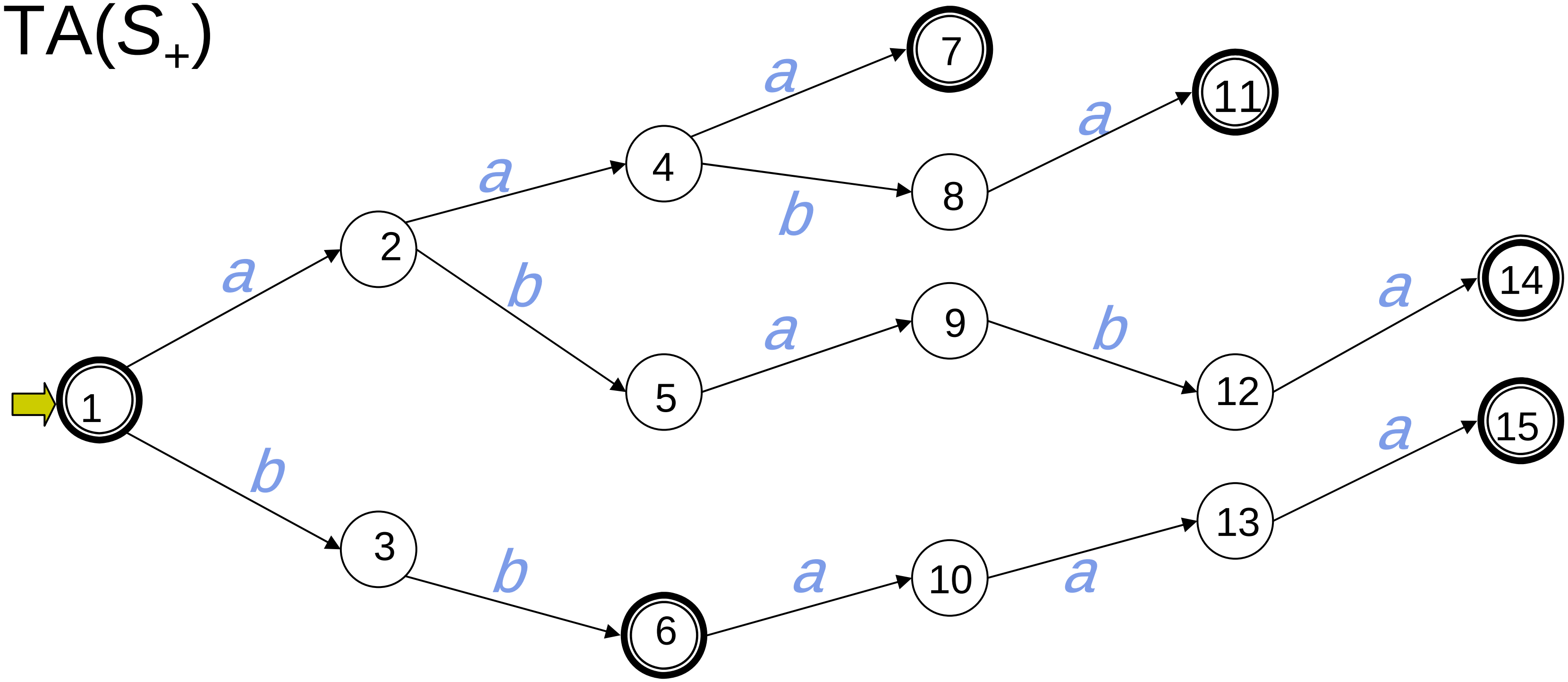


The prefix tree acceptor

- The smallest tree-like DFA consistent with the data
- Is a solution to the learning problem
- Corresponds to a rote learner

From the sample to the PTA

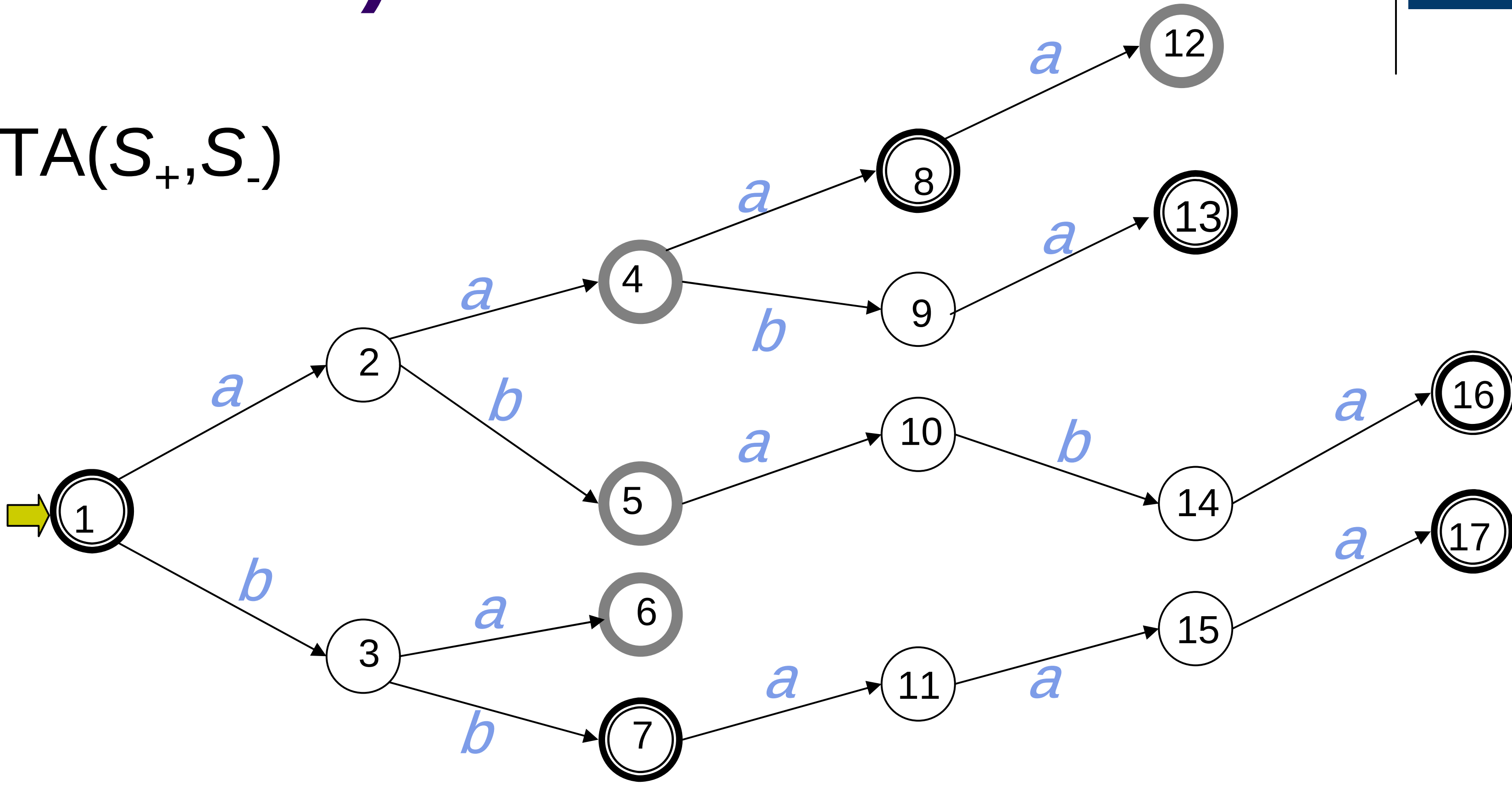
PTA(S_+)



$S_+=\{\lambda,\ aaa,\ aaba,\ ababa,\ bb,\ bbaaa\}$
$S_-=\{aa,\ ab,\ aaaa,\ ba\}$

From the sample to the PTA (full PTA)

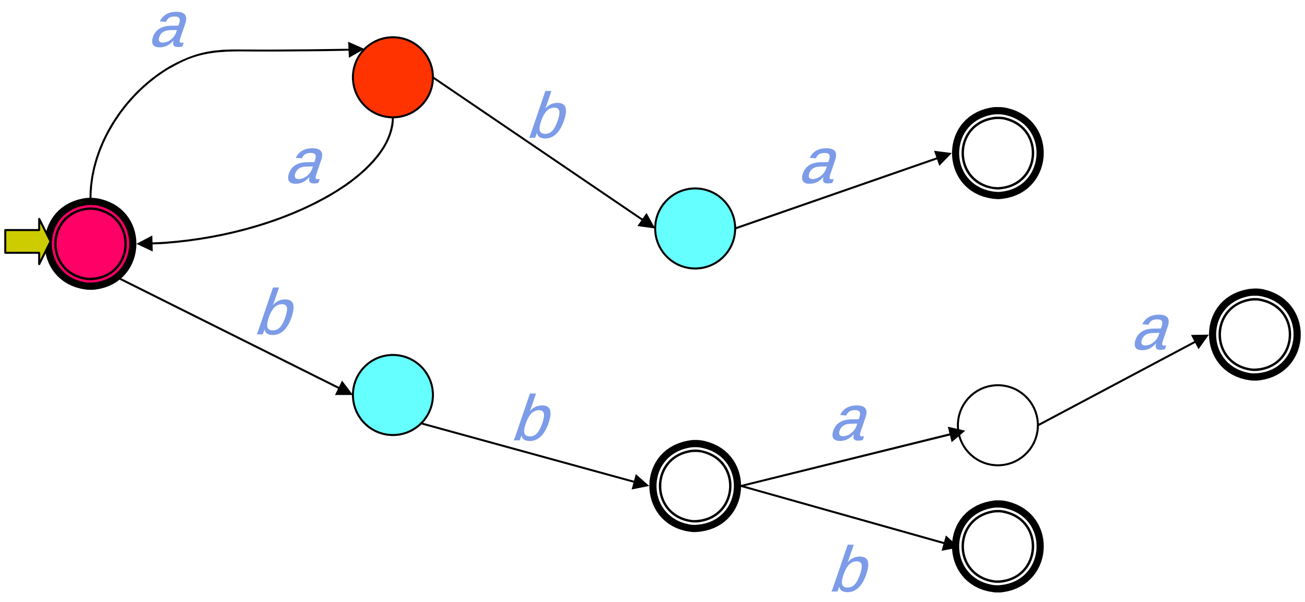
PTA(S_+,S_-)



S_+={λ, *aaa*, *aaba*, *ababa*, *bb*, *bbaaa*}
S_-={*aa*, *ab*, *aaaa*, *ba*}

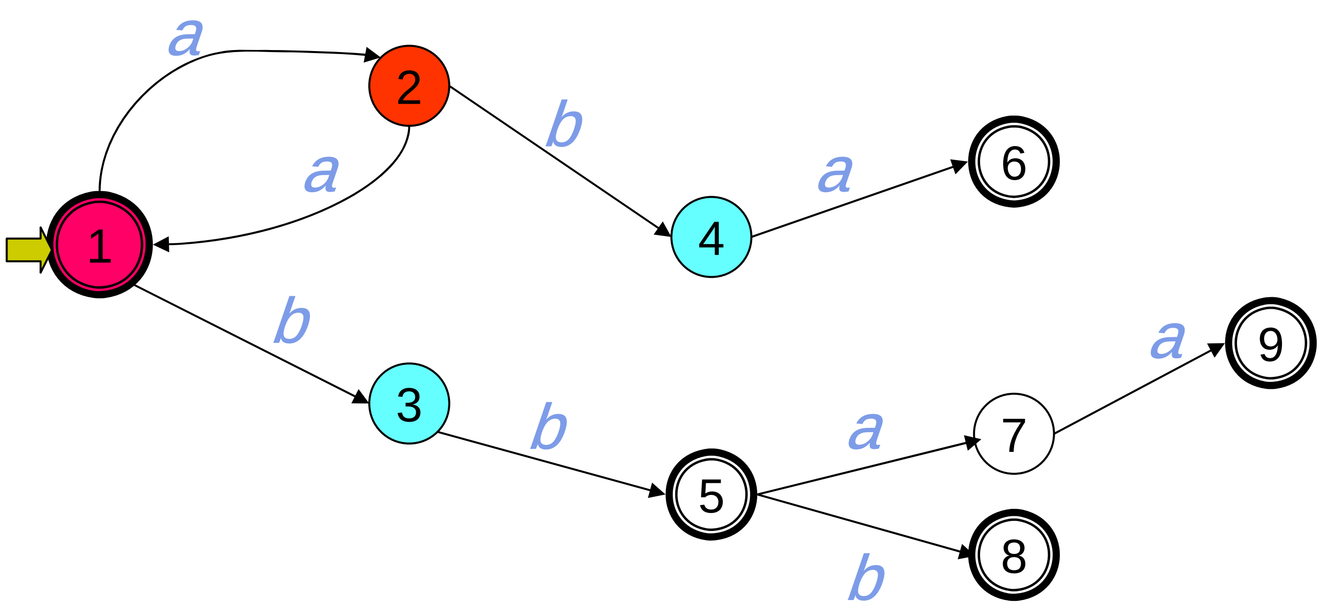
Red, Blue and White states

-Red states are confirmed states
-Blue states are the (non Red) successors of the Red states
-White states are the others



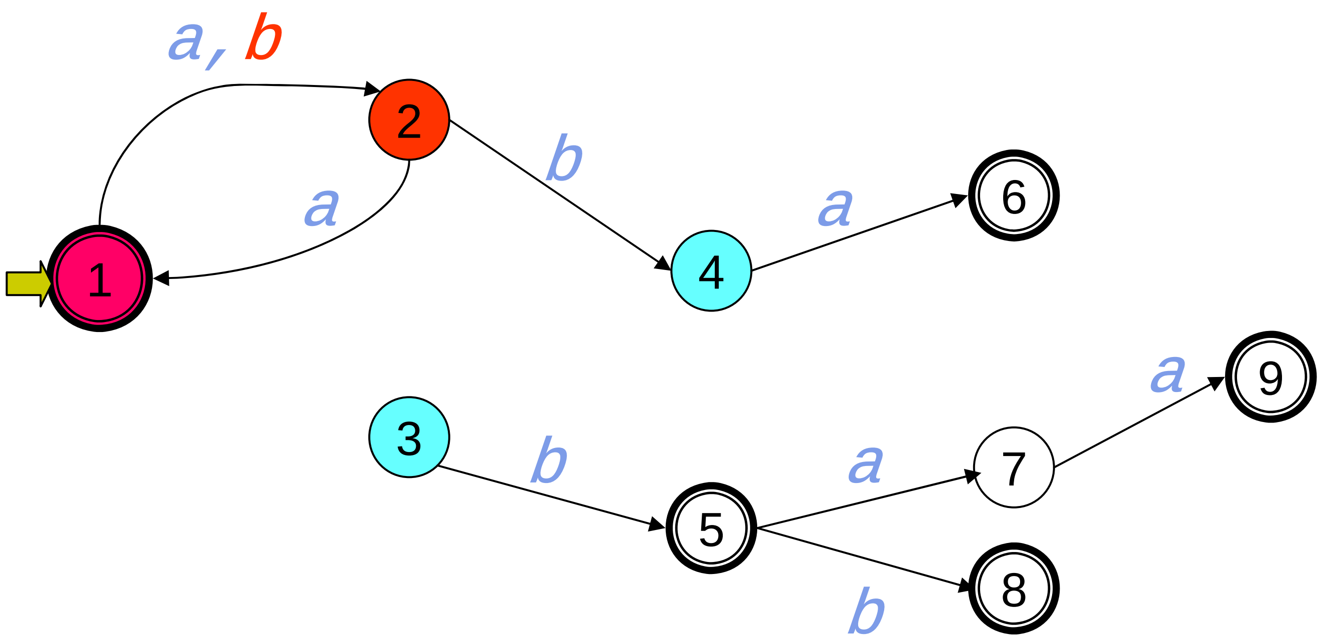
Merge and fold

Suppose we want to merge state 3 with state 2



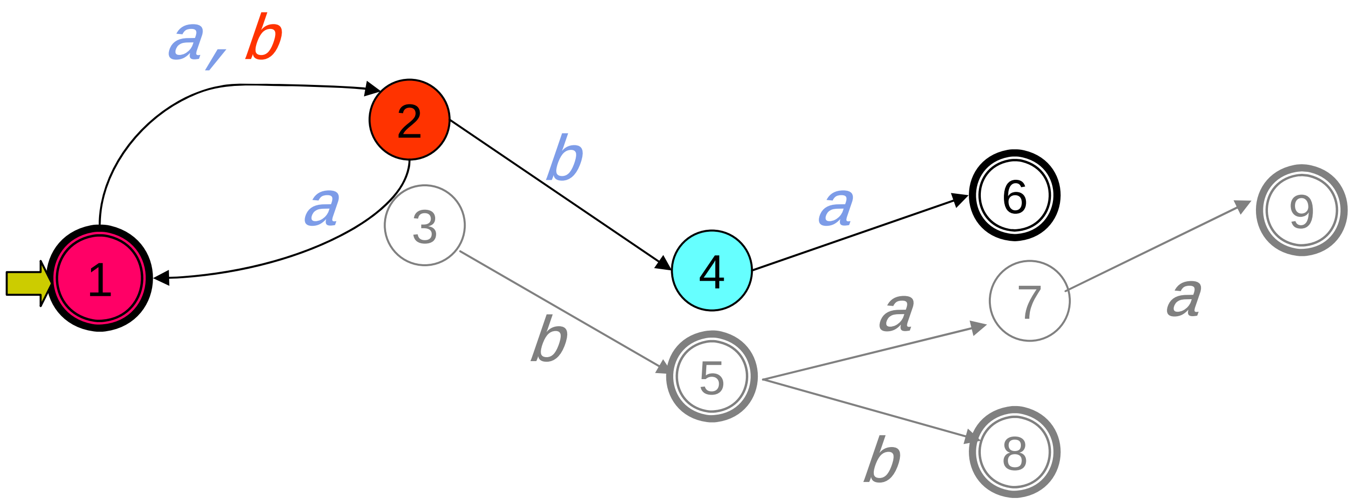
Merge and fold

First disconnect 3
and reconnect to 2



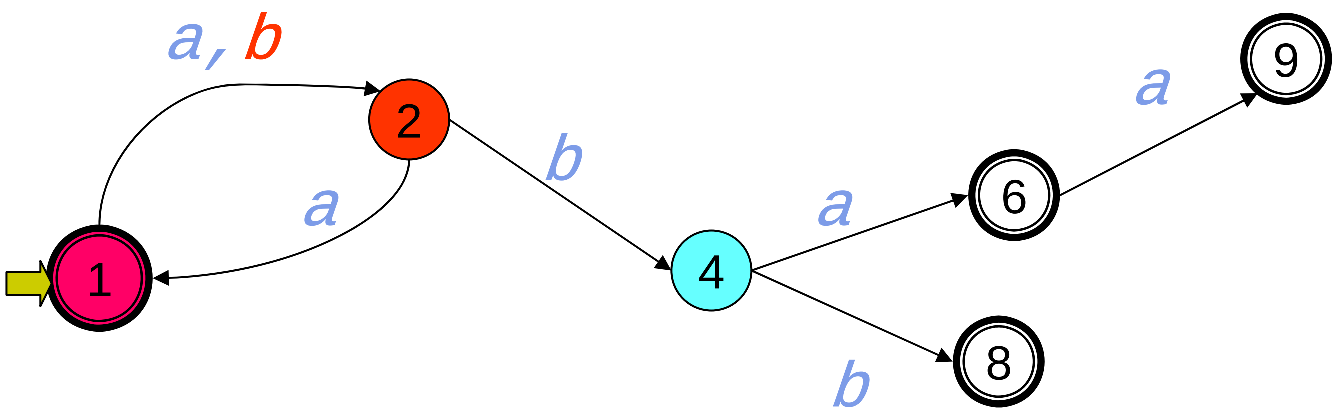
Merge and fold

Then fold subtree rooted in 3 into the DFA starting in 2



Merge and fold

Then fold subtree rooted in 3 into the DFA starting in 2



- RPNI is a state merging algorithm
- RPNI identifies any regular language in the limit
- RPNI works in polynomial time
- RPNI admits polynomial characteristic sets

A=PTA($S+$); *Blue* ={$\delta(q_I,a)$: $a\in\Sigma$ };

Red ={q_I}

While *Blue*$\neq\emptyset$ do

choose q from *Blue*

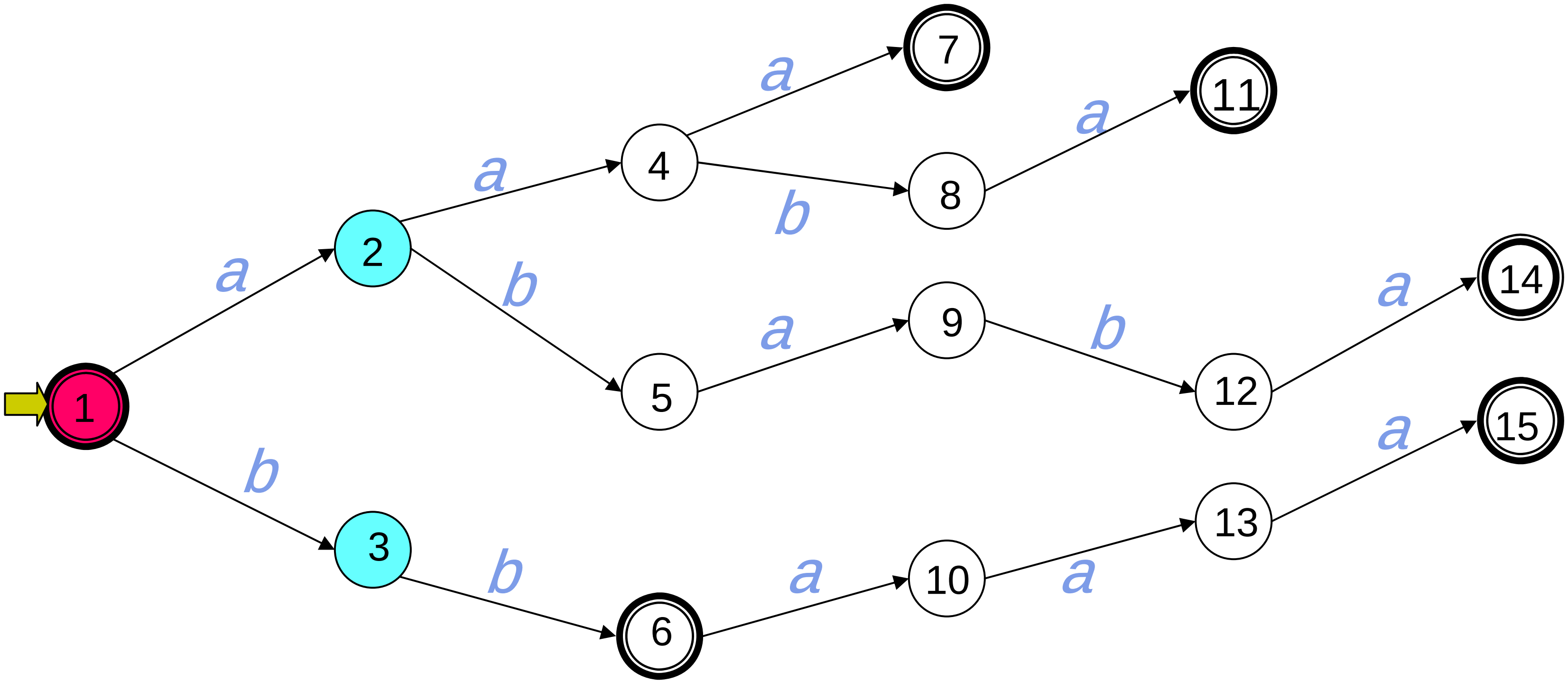
if $\exists p\in$ *Red*: $\mathbb{L}$(merge_and_fold(A,p,q))$\cap S-=\emptyset$

then A = merge_and_fold(A,p,q)

else *Red* = *Red* $\cup$ {q}

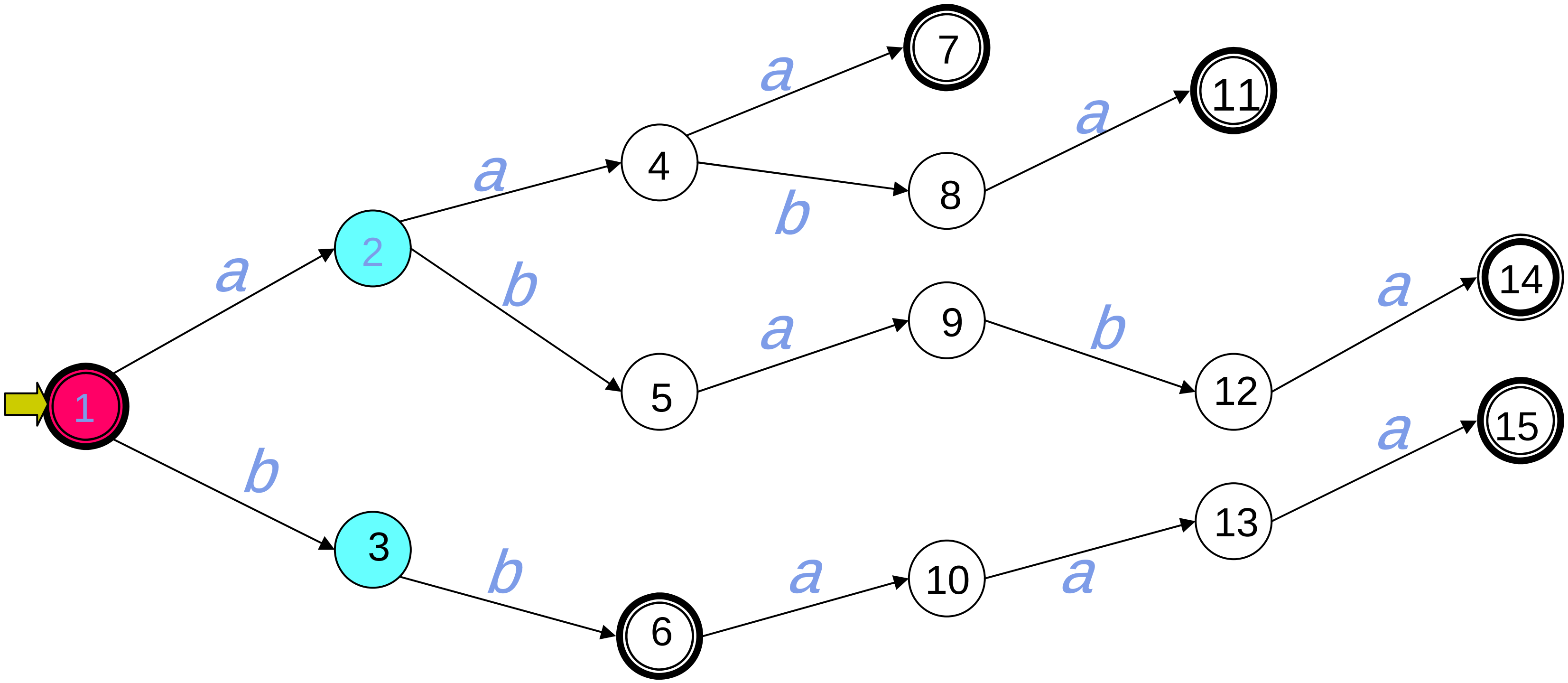
Blue = {$\delta(q,a)$: $q\in$ *Red*} - {*Red*}

$S_+=\{\lambda,\ aaa,\ aaba,\ ababa,\ bb,\ bbaaa\}$



$S_-=\{aa,\ ab,\ aaaa,\ ba\}$

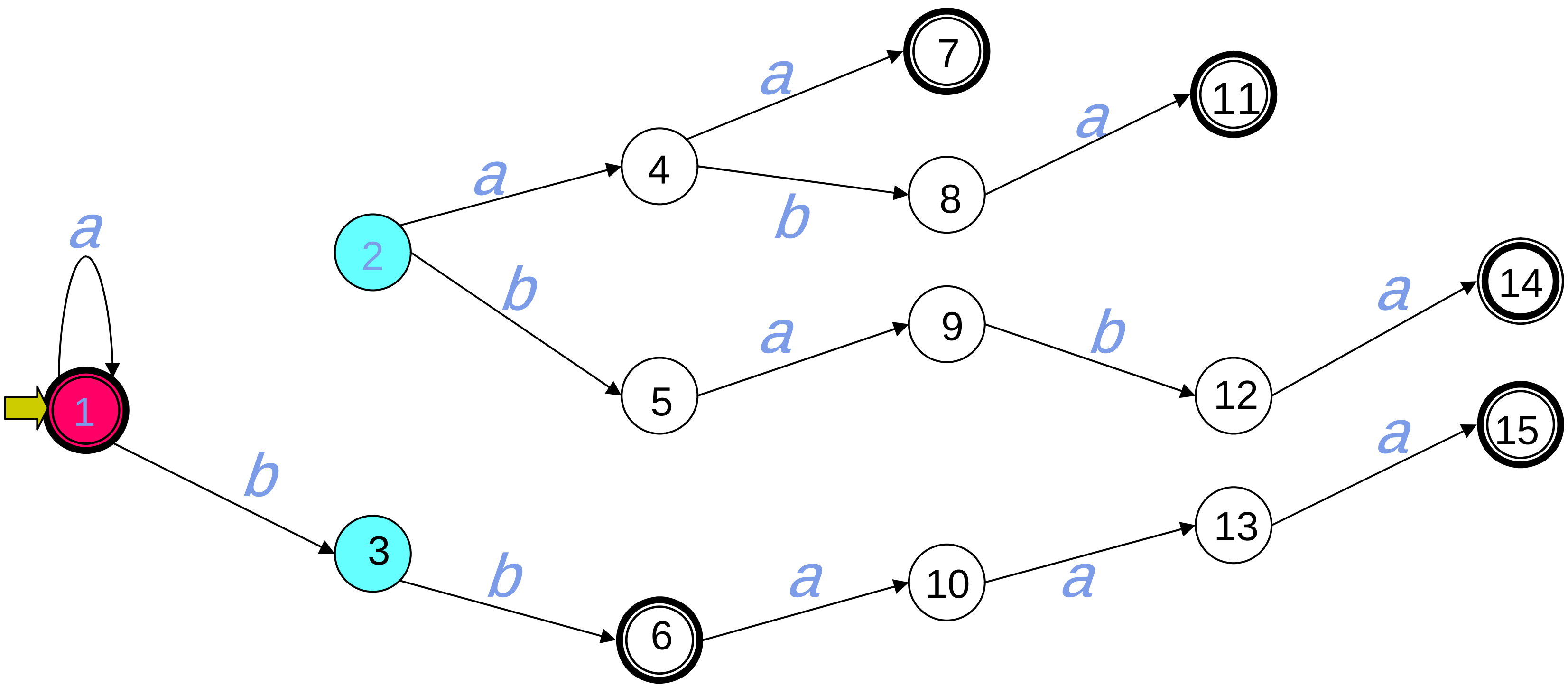
Try to merge 2 and 1



$S_-=\{aa,\ ab,\ aaaa,\ ba\}$

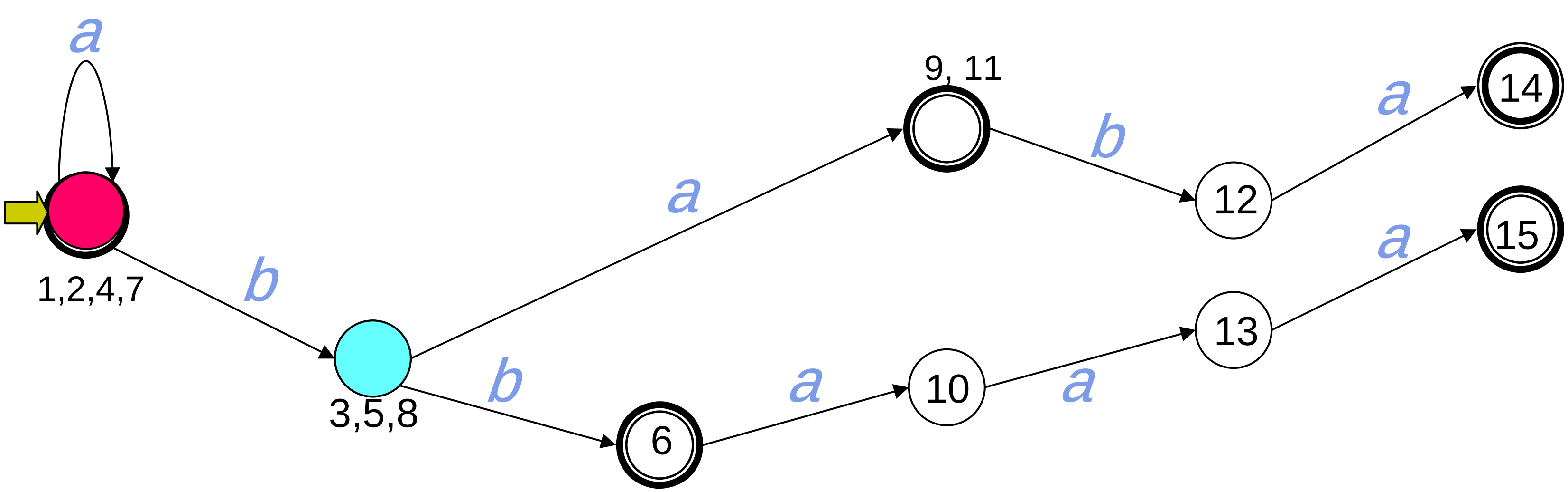
Cdlh 2010

First merge, then fold



$S_- = \{aa, ab, aaaa, ba\}$

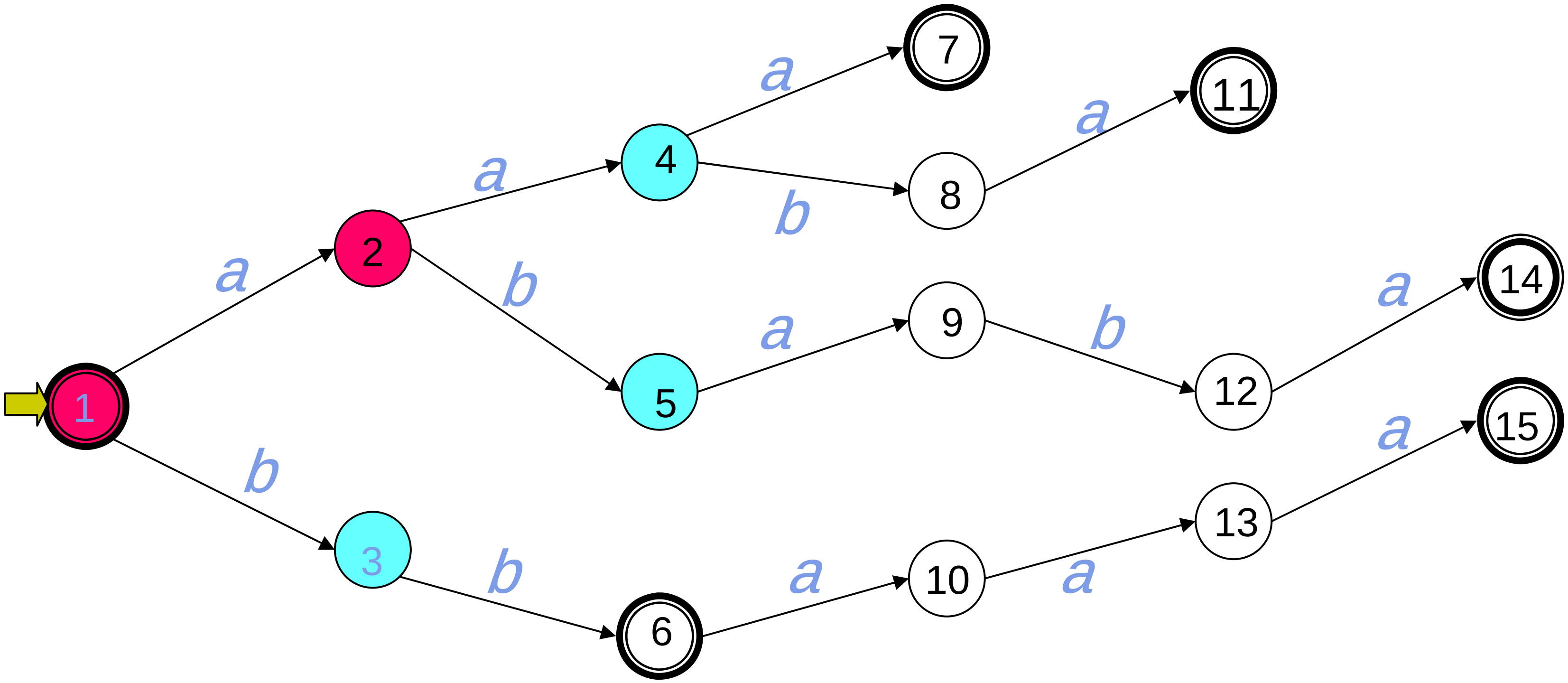
But now string aaaa is accepted, so the merge must be rejected, and state 2 is promoted



S_-={*aa*, *ab*, *aaaa*, *ba*}

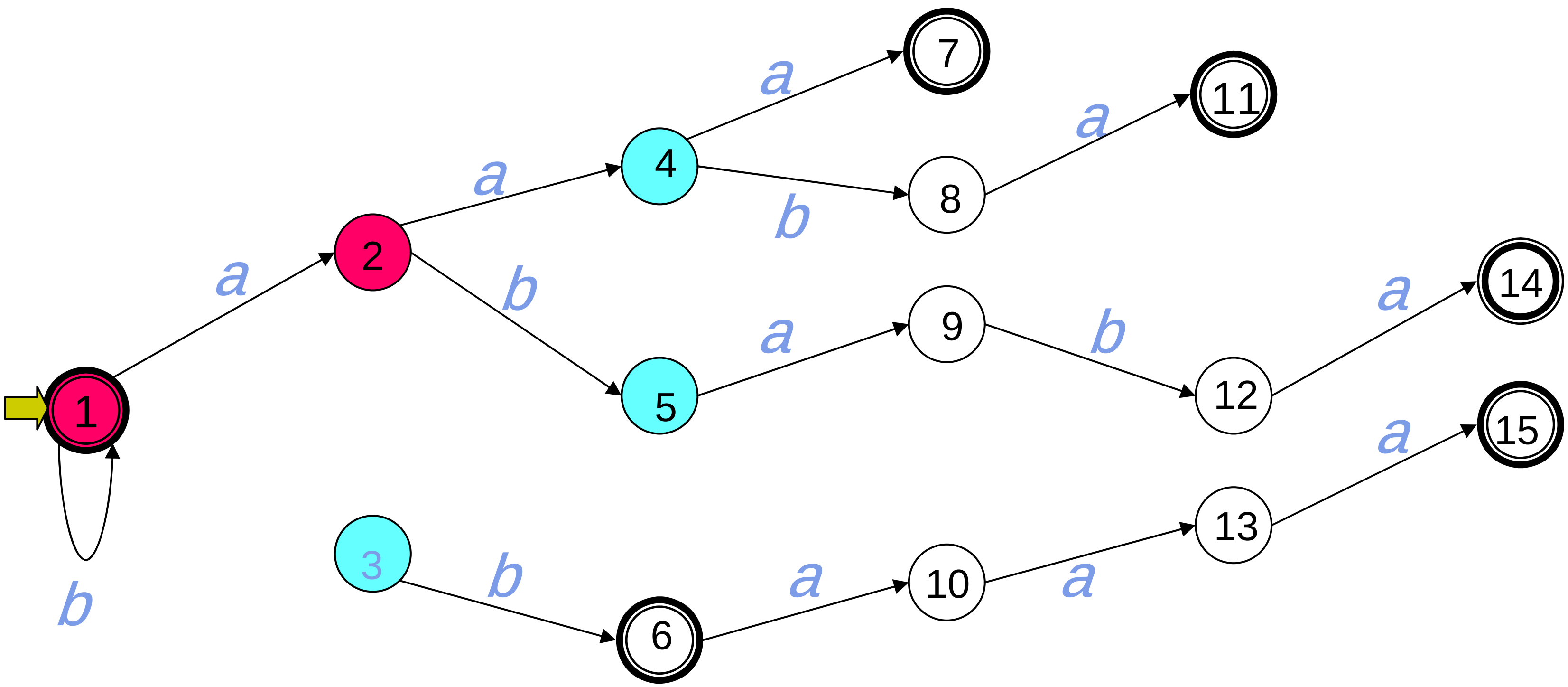
Cdlh 2010

Try to merge 3 and 1



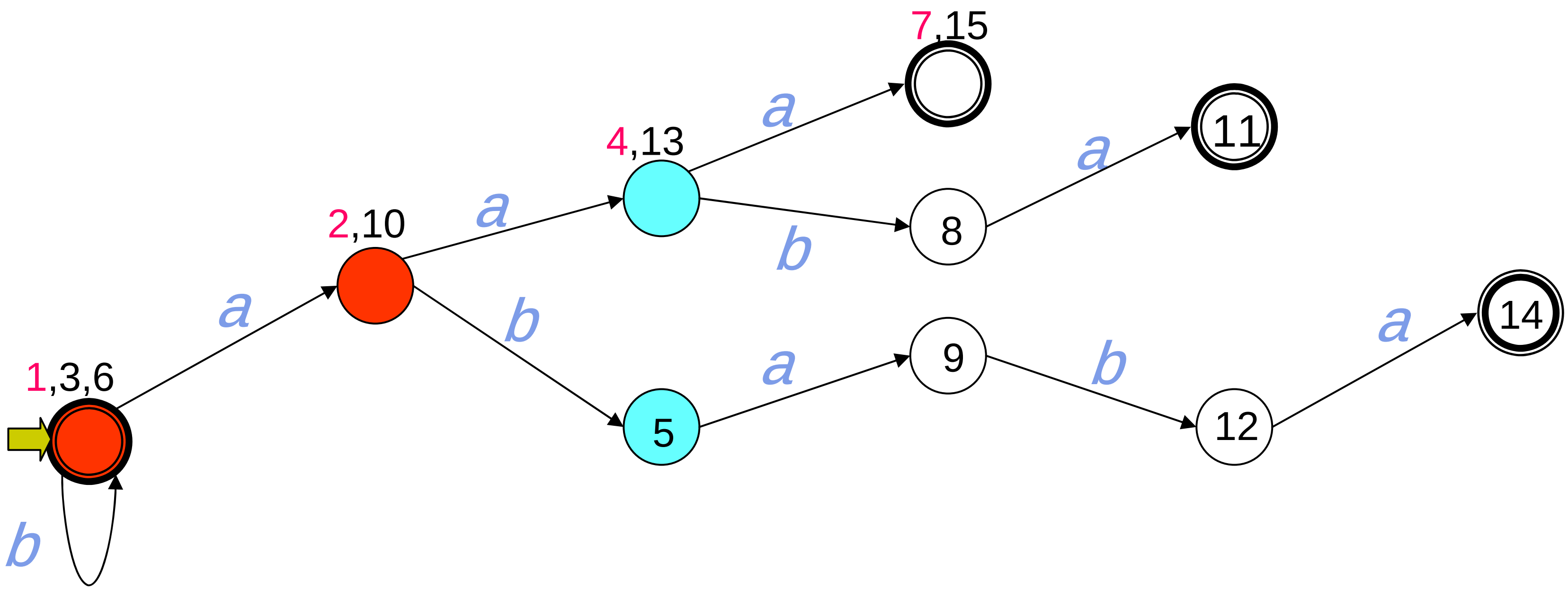
$S_- = \{aa, ab, aaaa, ba\}$

First merge, then fold



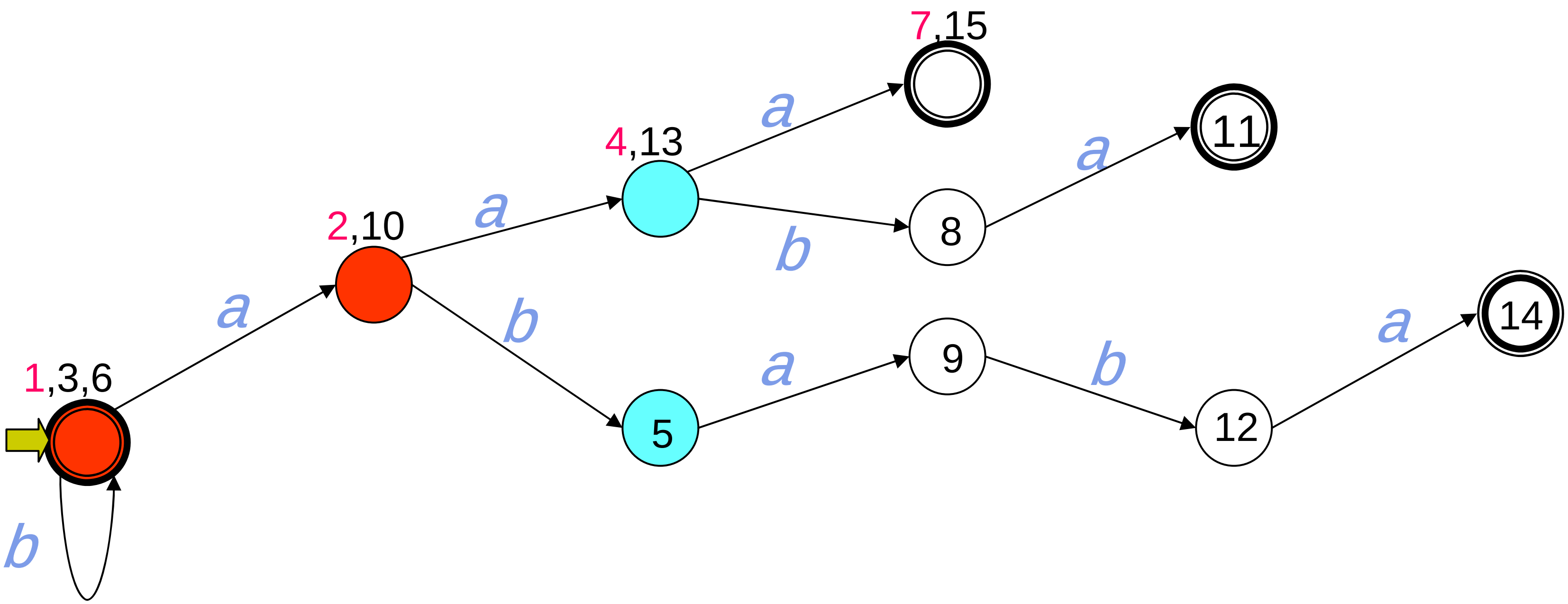
$S_- = \{aa, ab, aaaa, ba\}$

No counter example is accepted so the merge is kept



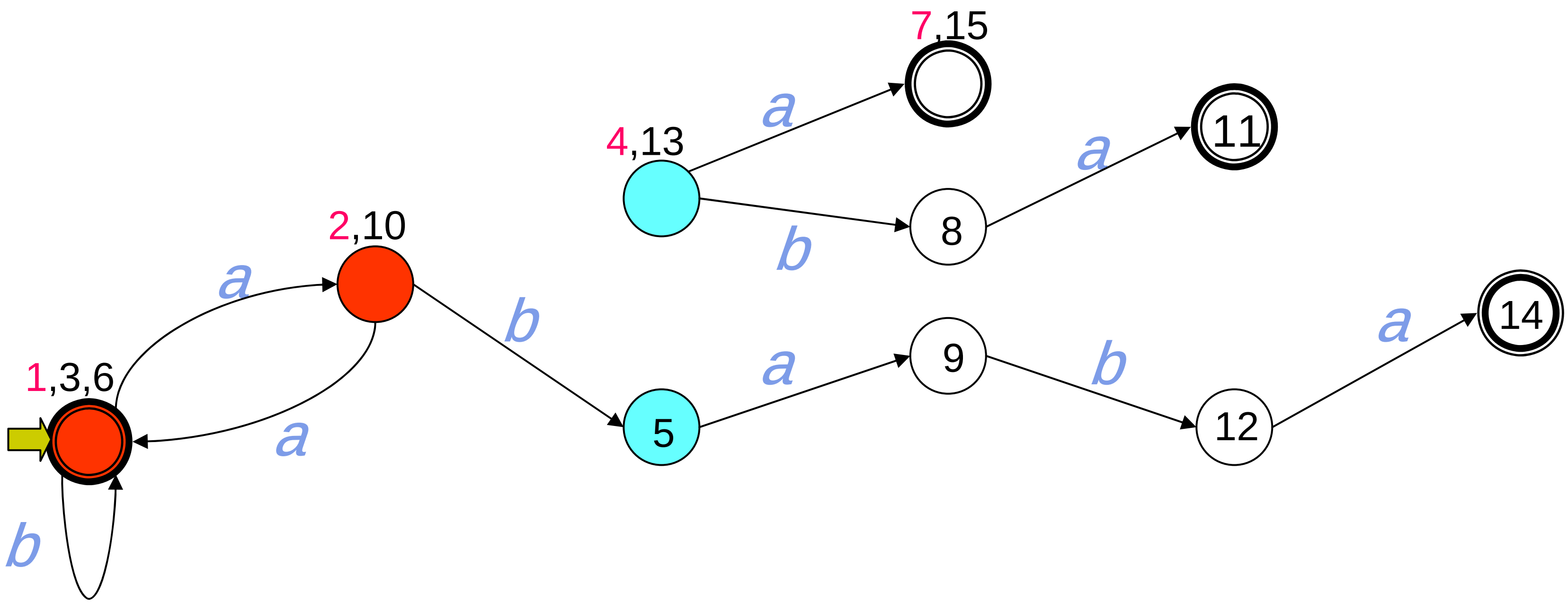
$S_- = \{aa, ab, aaaa, ba\}$

Next possible merge to be checked is {4,13} with {1,3,6}



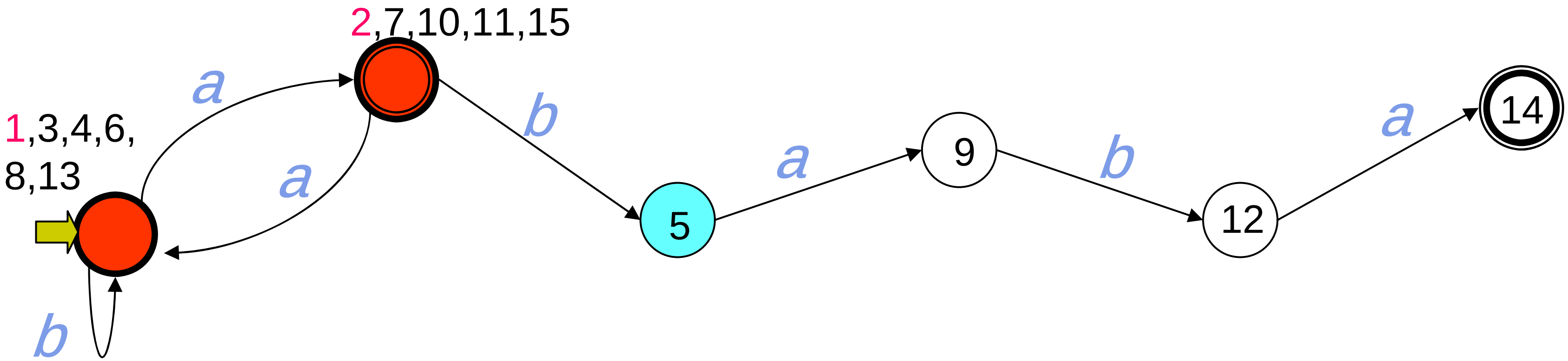
$S_- = \{aa,\ ab,\ aaaa,\ ba\}$

Merged. Needs folding subtree in {4,13} with {1,3,6}



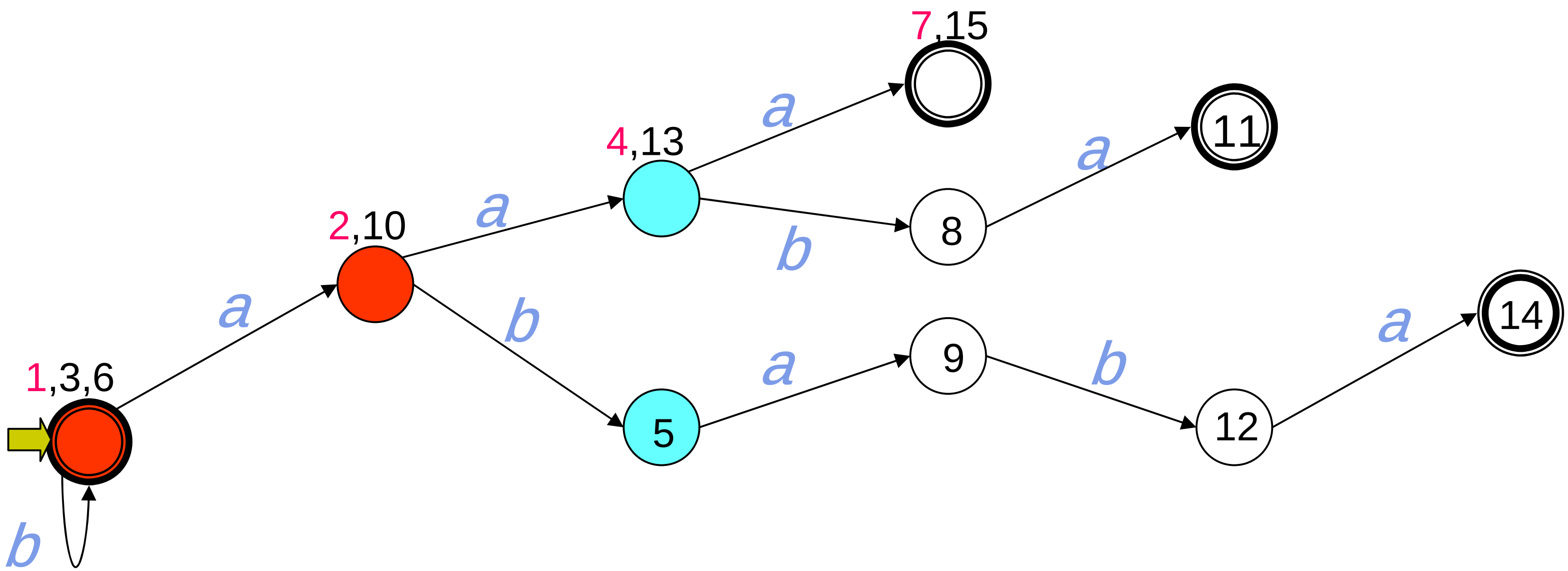
$S_- = \{aa, ab, aaaa, ba\}$

But now aa is accepted



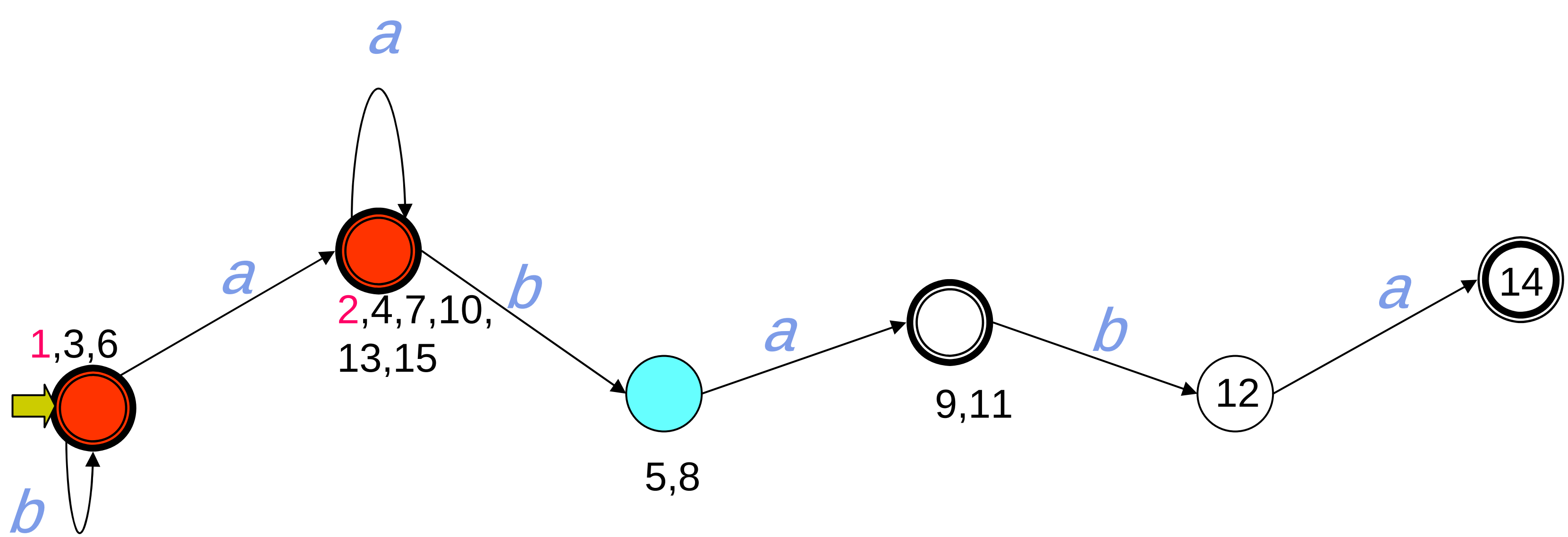
S_{-}={*aa*, *ab*, *aaaa*, *ba*}

So we try {4,13} with {2,10}



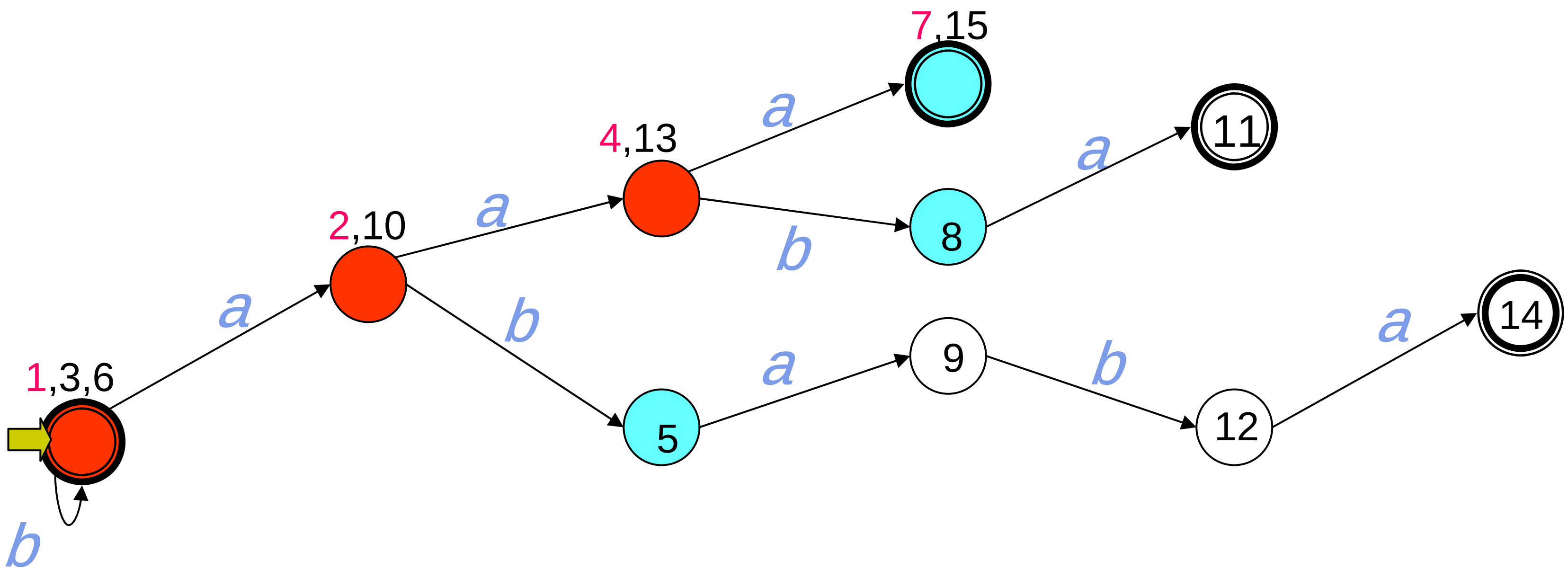
$S_- = \{aa,\ ab,\ aaaa,\ ba\}$

Negative string aa is again accepted. Since we have tried all Red for merging, state 4 is promoted.



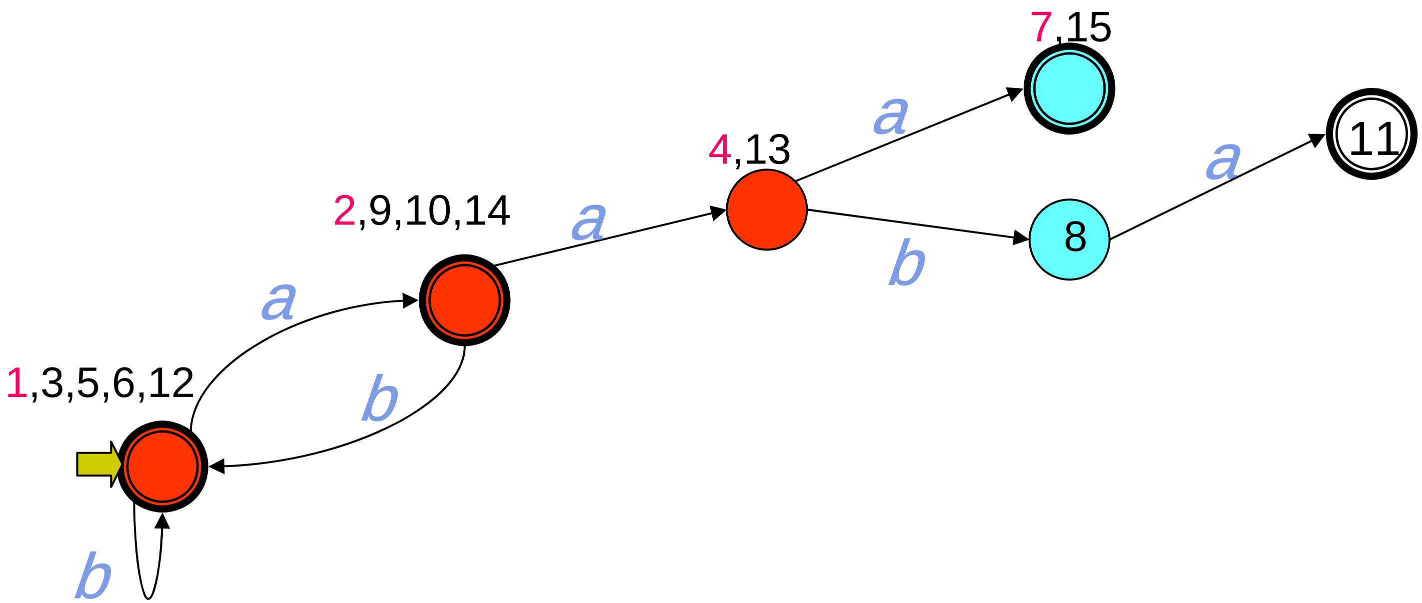
$S_-=\{aa, ab, aaaa, ba\}$

So we try 5 with {1,3,6}



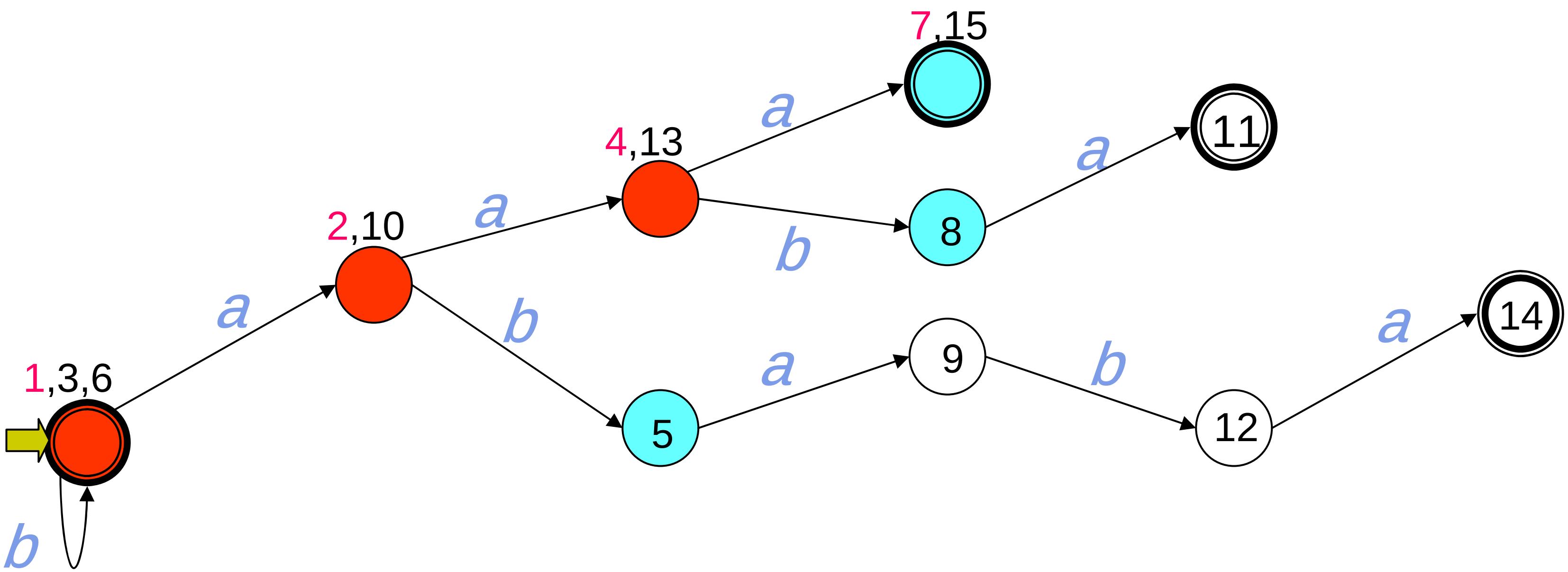
$S_- = \{aa, ab, aaaa, ba\}$

But again we accept ab



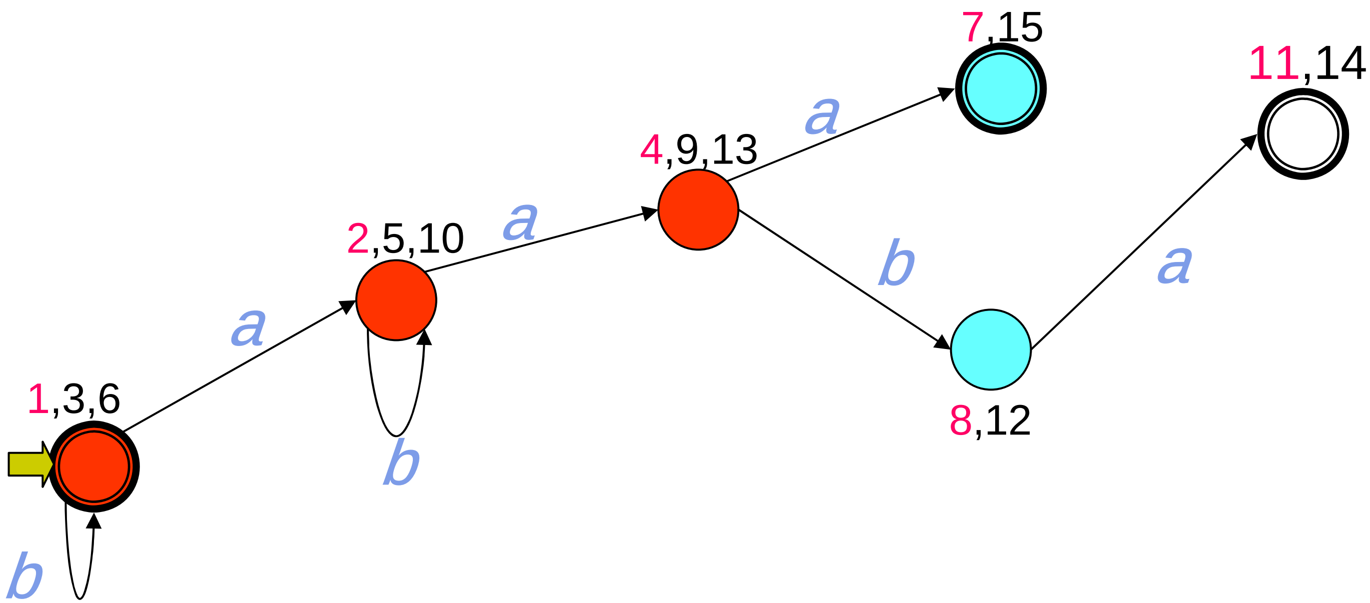
S_{-}={aa, ab, aaaa, ba}

So we try 5 with {2,10}



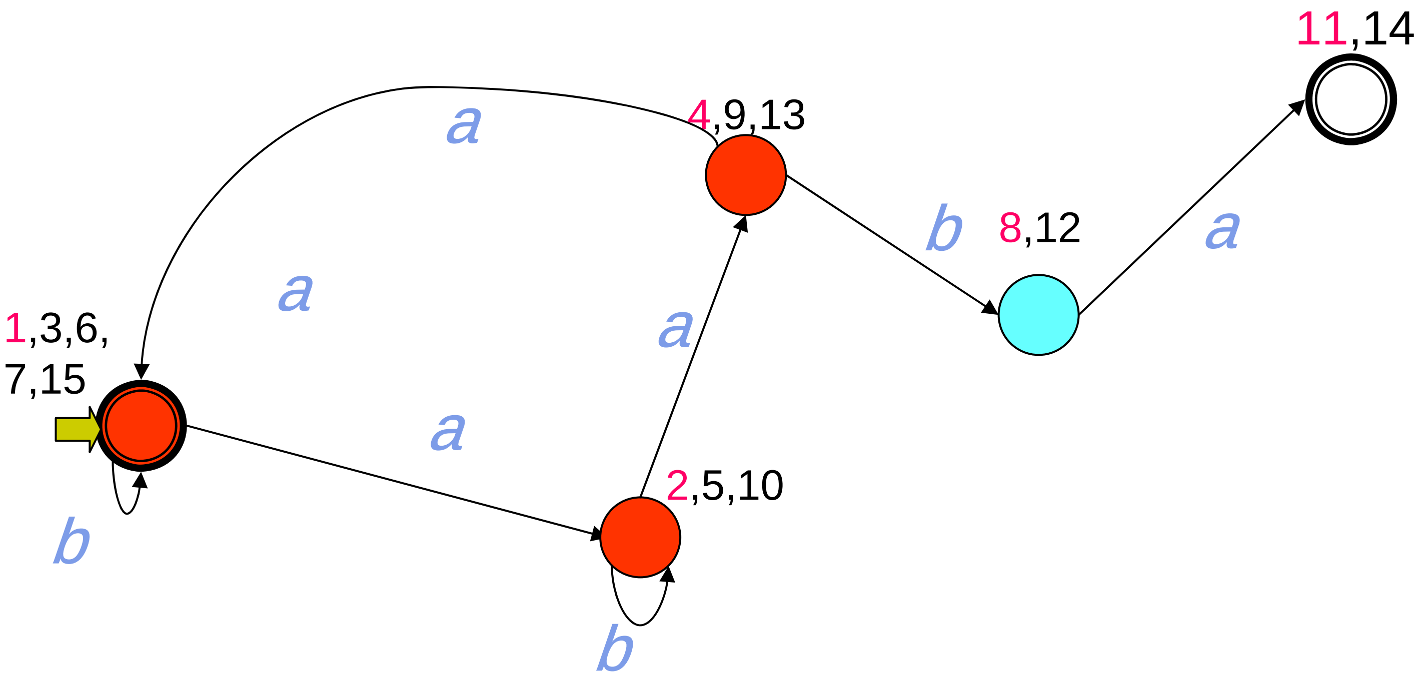
$S_- = \{aa,\ ab,\ aaaa,\ ba\}$

Which is OK. So next possible merge is {7,15} with {1,3,6}



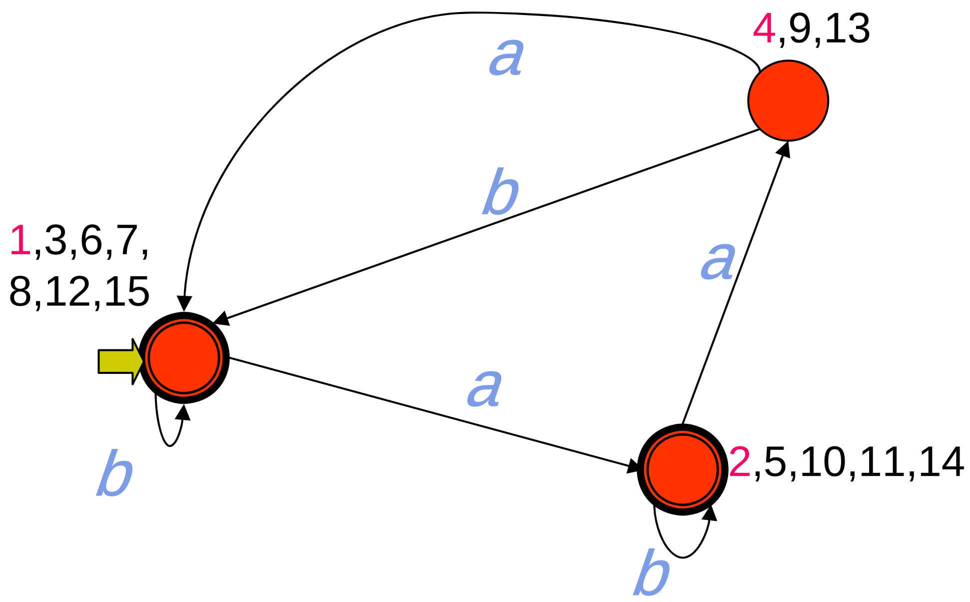
$S_- = \{aa, ab, aaaa, ba\}$

Which is OK. Now try to merge {8,12} with {1,3,6,7,15}



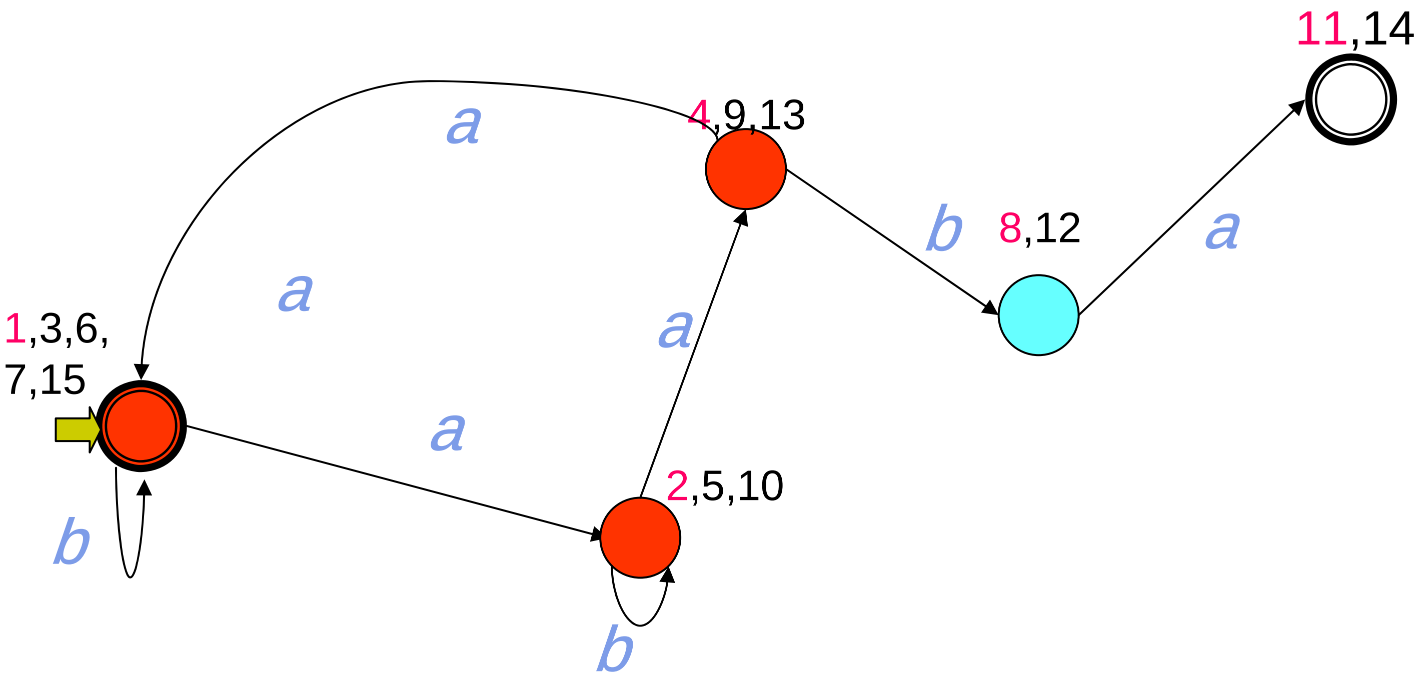
S_-={*aa*, *ab*, *aaaa*, *ba*}

And ab is accepted



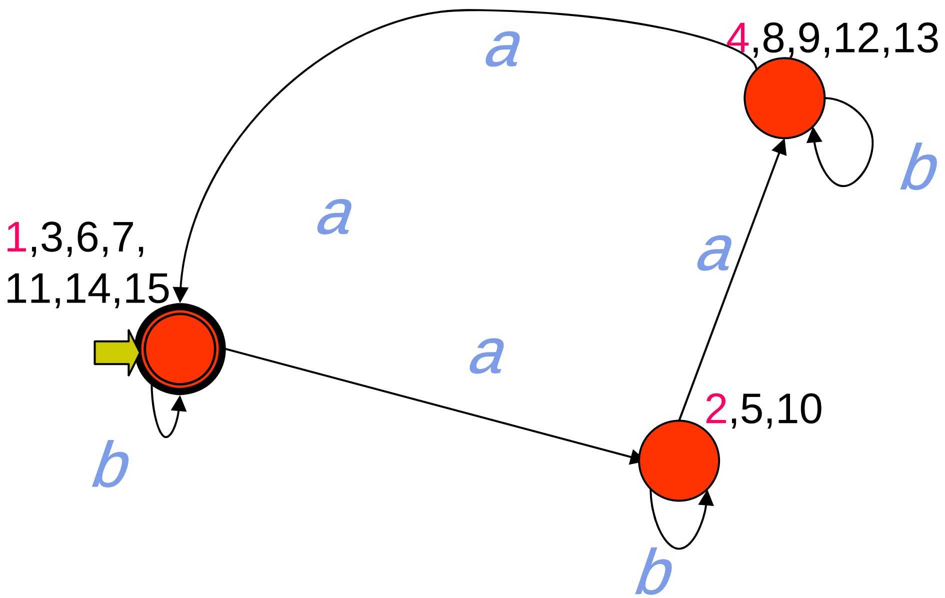
$S_-=\{aa, ab, aaaa, ba\}$

Now try to merge {8,12} with {4,9,13}



S_-={*aa*, *ab*, *aaaa*, *ba*}

This is OK and no more merge is possible so the algorithm halts



S_{-}={*aa*, *ab*, *aaaa*, *ba*}

A characteristic sample

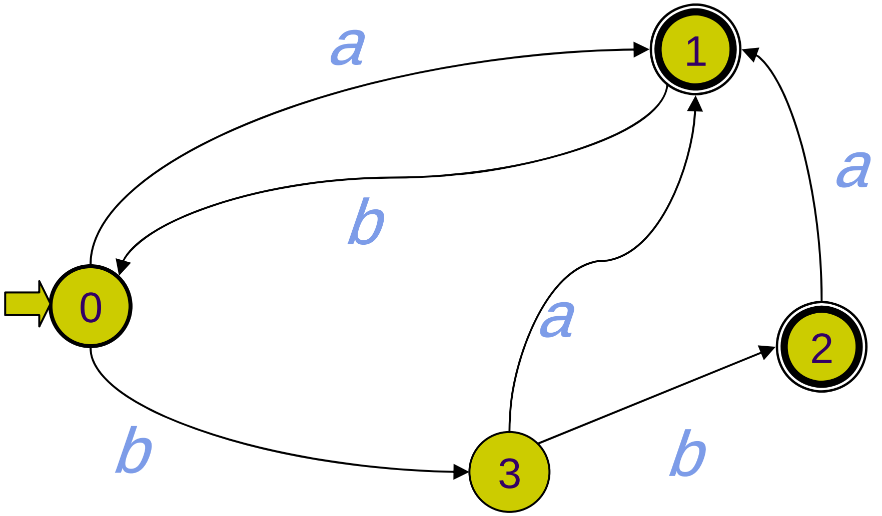
- A sample is characteristic (for RPNI) whenever, when included in the learning sample, the algorithm returns the correct DFA
- Particularity: the characteristic sample is of polynomial size
- There is an algorithm which given a DFA builds a characteristic sample

About characteristic samples

- If you add more strings to a characteristic sample it still is characteristic
- There can be many different characteristic samples (EDSM, tree version,…)
- Change the ordering (or the exploring function in RPNI) and the characteristic sample will change

Exercices

- Run RPNI on
  - $S_+=\{a,bba,bab,aabb\}$
  - $S_-=\{b,ab,baa,baabb\}$
- Find a characteristic sample for:



Open problems

- RPNI's complexity is not a tight upper bound. Find the correct complexity
- The definition of the characteristic sample is not tight either. Find a better definition

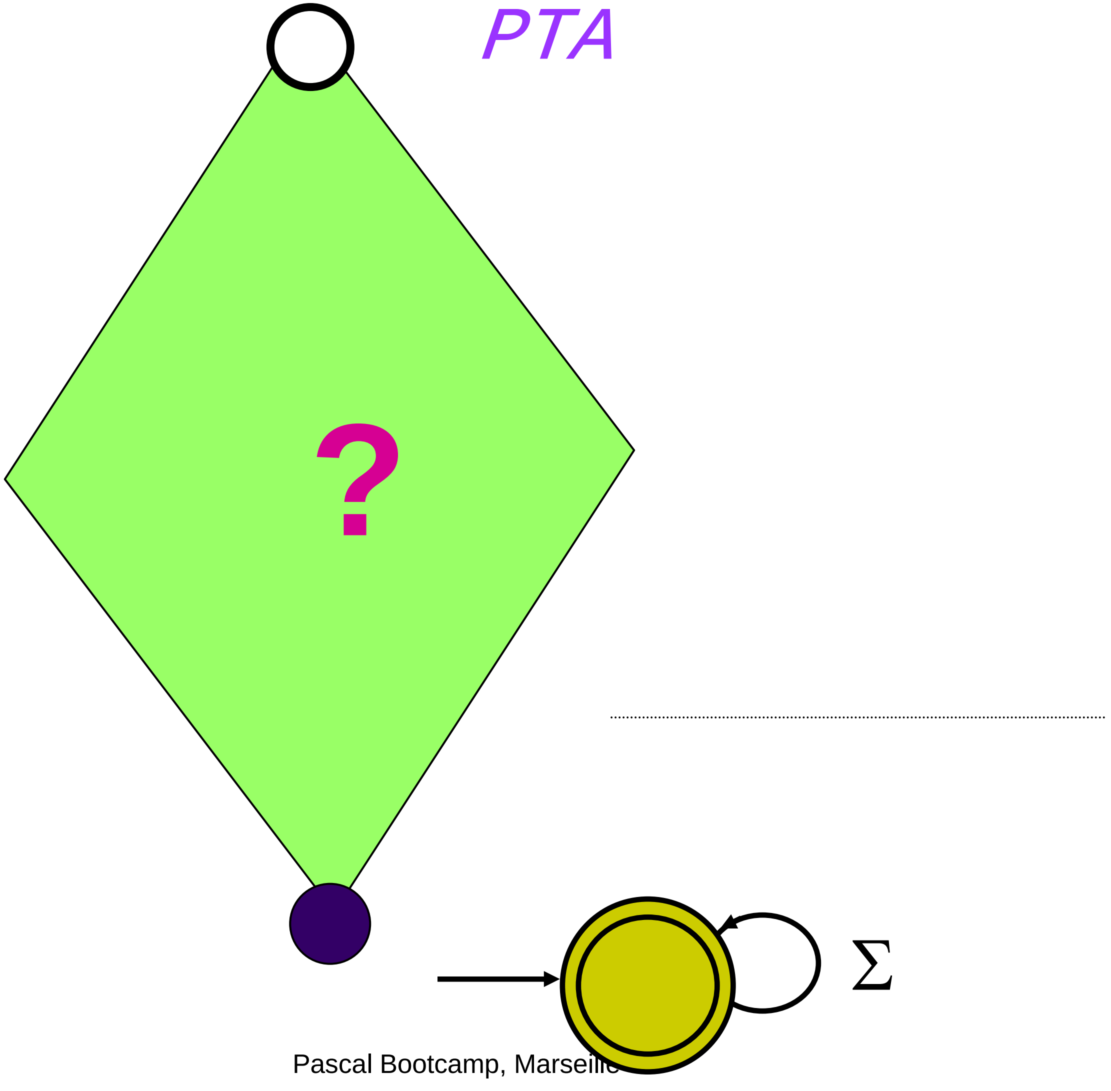
Conclusion

- RPNI identifies any regular language in the limit
- RPNI works in polynomial time
- There are many significant variants of RPNI
- Parallel version can be efficient
- RPNI can be extended to other classes of grammars

6 Learning from text

- Only positive examples are available
- Danger of over-generalization: why not return Σ^*?
- The problem is "basic":
  - Negative examples might not be available
  - Or they might be heavily biased: near-misses, absurd examples…
- Base line: all the rest is learning with help

GI as a search problem



The theory

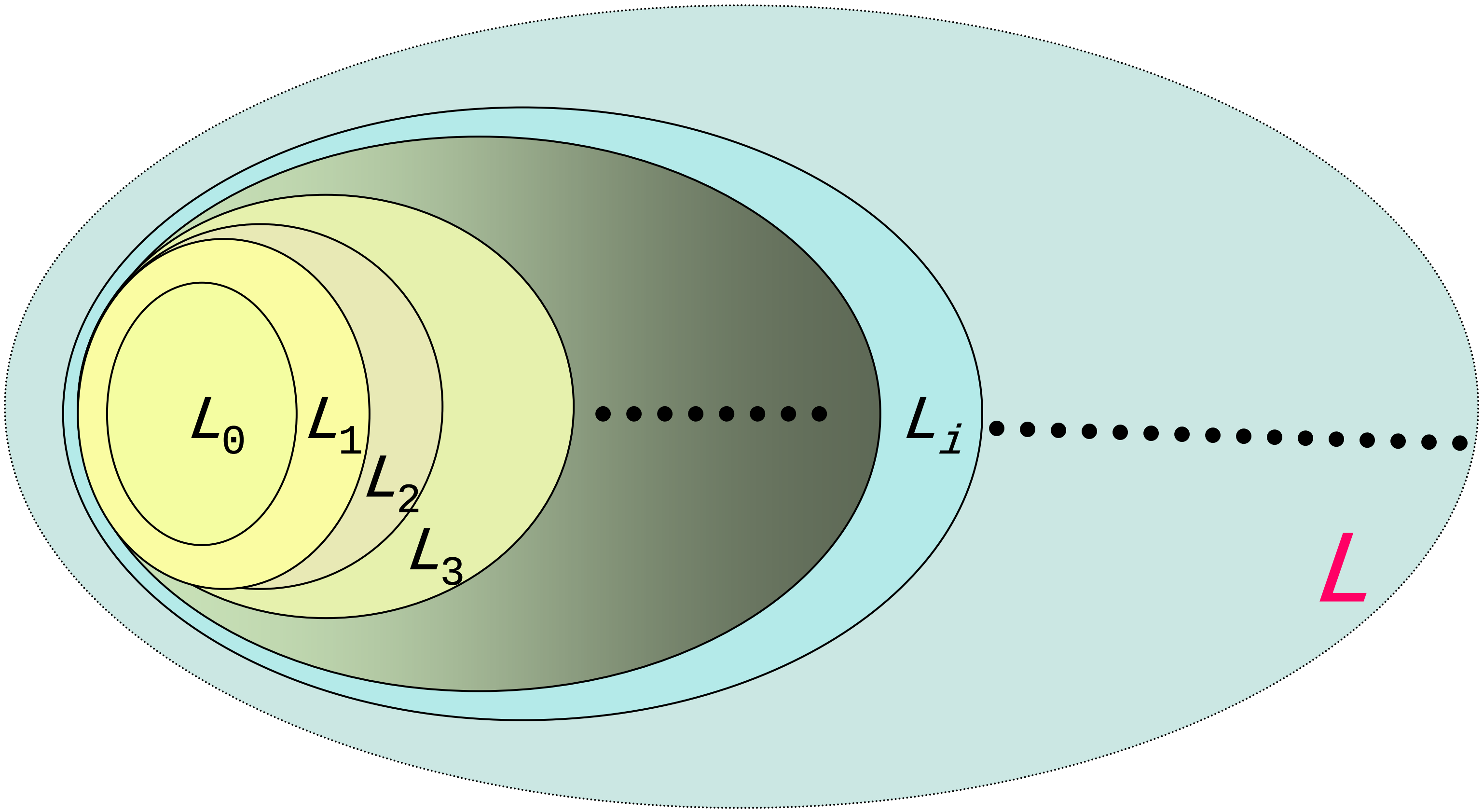
- Gold 67: No super-finite class can be identified from positive examples (or **text**) only

- Necessary and sufficient conditions for learning
- Literature:
  - inductive inference,
  - ALT series, …

Limit point

- A class $\mathcal{L}$ of languages has a limit point *iff* there exists an infinite sequence L_n $_{n\in\mathbb{N}}$ of languages in $\mathcal{L}$ such that $L_0 \subset L_1 \subset \ldots L_n \subset \ldots$, and there exists another language $L \in \mathcal{L}$ such that $L = \bigcup_{n\in\mathbb{N}} L_n$
- L is called a limit point of $\mathcal{L}$

L is a limit point



Theorem

If $\mathcal{L}$ admits a limit point, then $\mathcal{L}$ is not learnable from text

<u>Proof:</u> Let s^i be a presentation in length-lex order for L_i, and s be a presentation in length-lex order for L. Then $\forall n \in \mathbb{N}\ \exists i\ /\ \forall k \leq n\ s^i_k = s_k$

<u>Note:</u> having a limit point is a sufficient condition for non learnability; not a necessary condition

Mincons classes

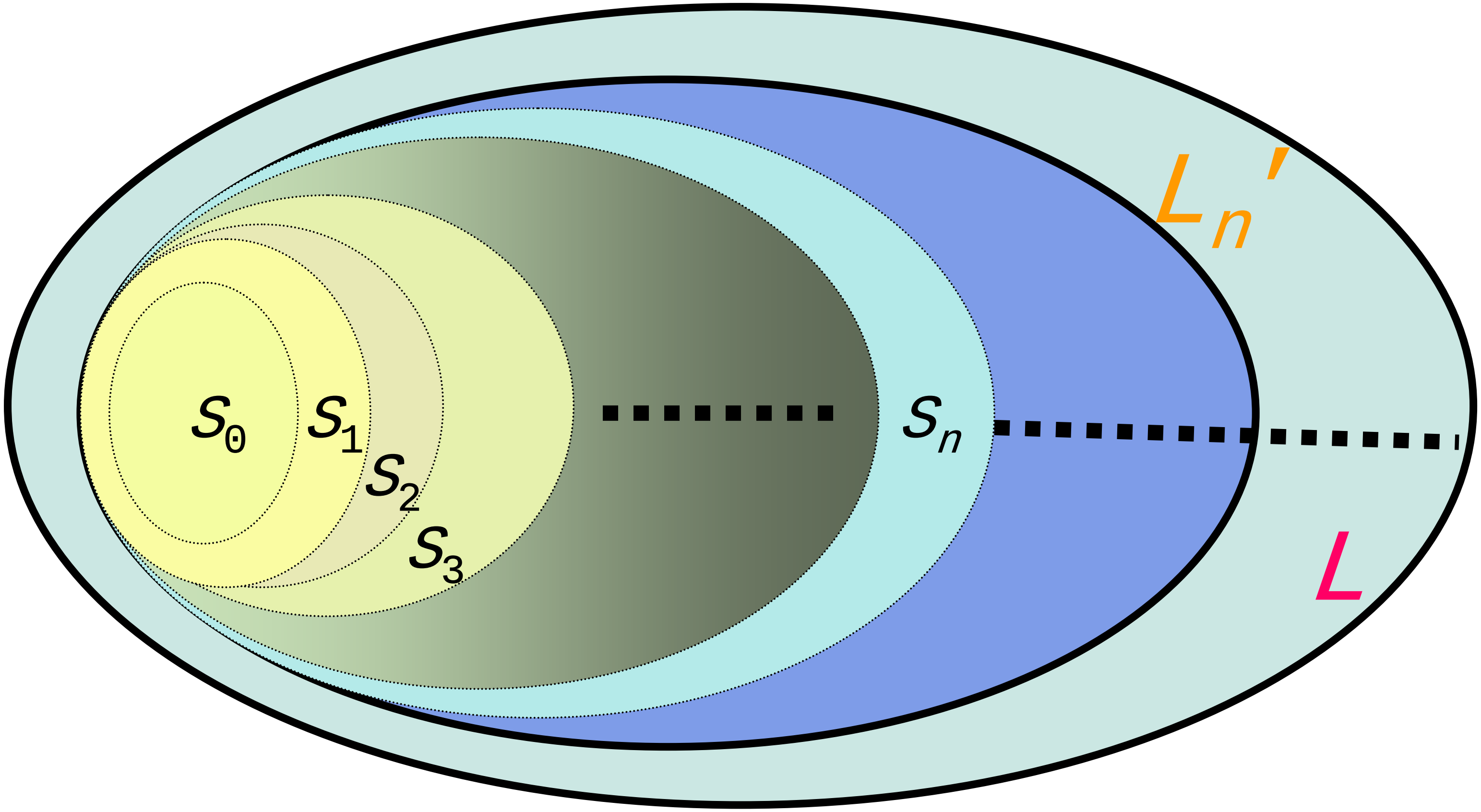
- A class is mincons if there is an algorithm which, given a sample S, builds a $G \in \mathcal{G}$ such that $S \subseteq L \subseteq \mathbb{L}(G) \Rightarrow L = \mathbb{L}(G)$

Existence of an accumulation point (Kapur 91)

A class $\mathcal{L}$ of languages has an accumulation point *iff* there exists an infinite sequence $S_{n\ n\in\mathbb{N}}$ of sets such that $S_0 \subseteq S_1 \subseteq \ldots S_n \subseteq \ldots$, and $L = \bigcup_{n\in\mathbb{N}} S_n \in \mathcal{L}$

…and for any $n\in\mathbb{N}$ there exists a language L_n' in $\mathcal{L}$ such that $S_n \subseteq L_n' \subset L$. The language L is called an accumulation point of $\mathcal{L}$

L is an accumulation point



Theorem (for Mincons classes)

$\mathcal{L}$ admits an accumulation point

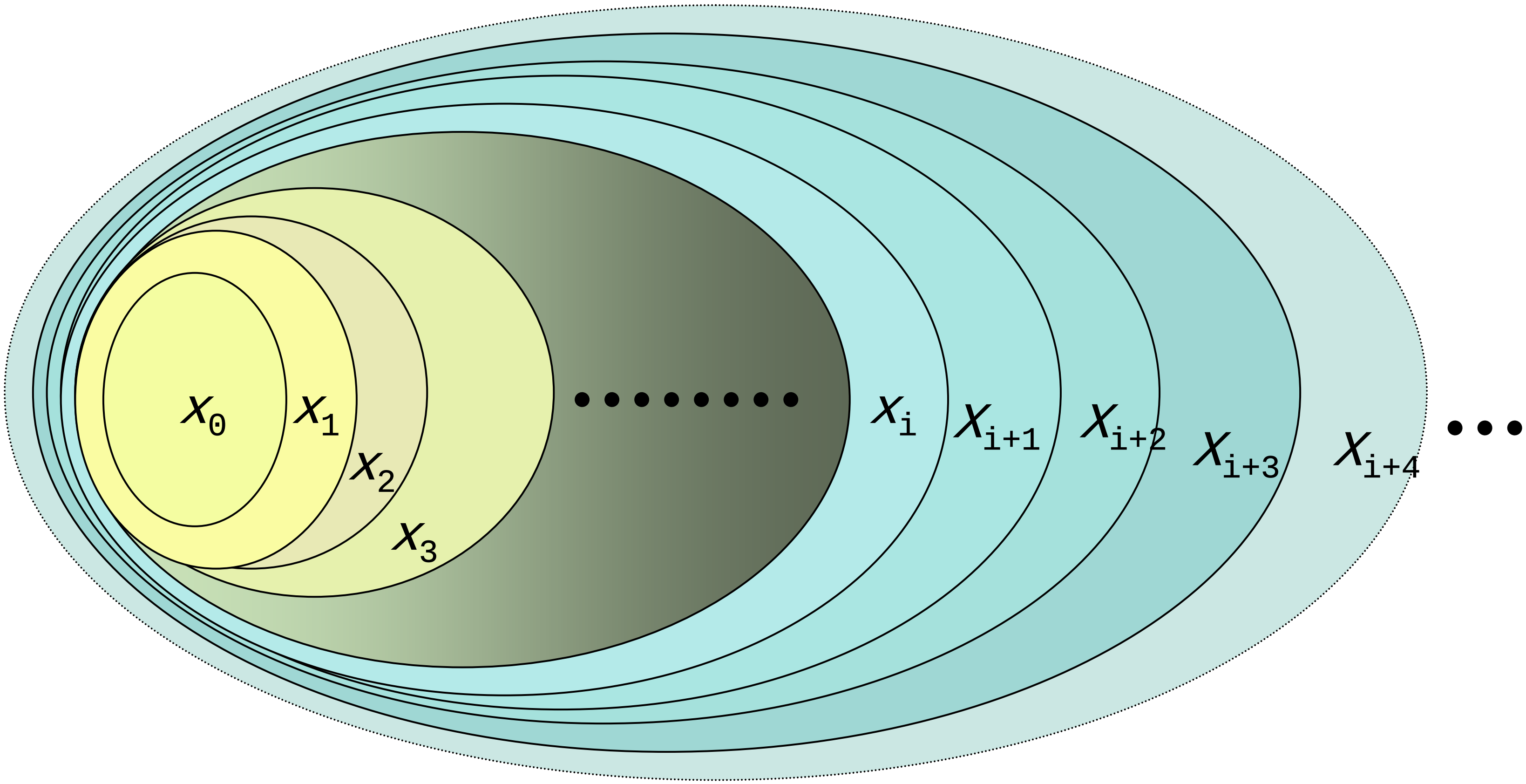
iff

$\mathcal{L}$ is not learnable from text

Infinite Elasticity

- If a class of languages has a limit point there exists an infinite ascending chain of languages $L_0 \subset L_1 \subset \ldots \subset L_n \subset \ldots$.
- This property is called infinite elasticity

Infinite Elasticity



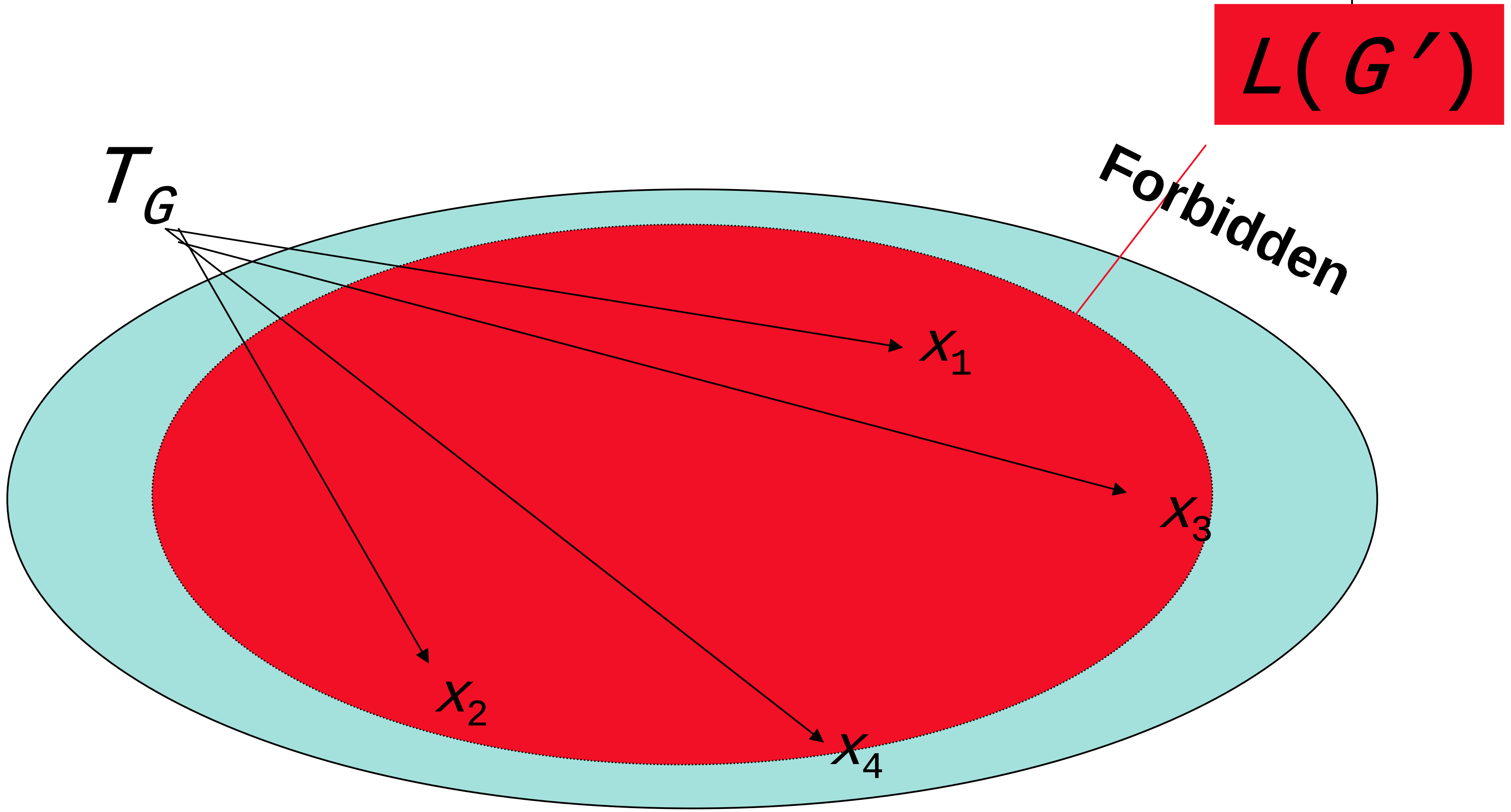
Finite elasticity

$\mathcal{L}$ has *finite elasticity* if it does not have infinite elasticity

Theorem (Wright)

If $\mathcal{L}(\mathcal{G})$ has finite elasticity and is mincons, then $\mathcal{G}$ is learnable.

Tell tale sets



Theorem (Angluin)

$\mathcal{G}$ is learnable *iff* there is a computable partial function ψ: $\mathcal{G}\times\mathbb{N}\to\Sigma^*$ such that:

1) $\forall n\in\mathbb{N}$, $\psi(G,n)$ is defined *iff* $G\in\mathcal{G}$ and $\mathbb{L}(G)\neq\emptyset$;
2) $\forall G\in\mathcal{G}$, $T_G=\{\psi(G,n)$: $n\in\mathbb{N}\}$ is a finite subset of $\mathbb{L}(G)$ called a *tell-tale* subset;
3) $\forall G,G'\in\mathcal{G}$, if $T_G\subseteq\mathbb{L}(G')$ then $\mathbb{L}(G')\not\subset\mathbb{L}(G)$.

Proposition (Kapur 91)

A language L in $\mathcal{L}$ has a *tell-tale subset* *iff* L is not an accumulation point.

(for mincons)

7 Learning by observing

Inference of k-Testable Languages in the Strict Sense and Application to Syntactic Pattern Recognition. García & Vidal et al. 1990

Definition

Let $k \geq 0$, a k-testable language in the strict sense (k-TSS) is a 5-tuple $Z_k=(\Sigma, I, F, T, C)$ with:

- Σ a finite alphabet
- $I, F \subseteq \Sigma^{k-1}$ (allowed prefixes of length k-1 and suffixes of length k-1)
- $T \subseteq \Sigma^k$ (allowed segments)
- $C \subseteq \Sigma^{<k}$ contains all strings of length less than k
- Note that $I \cap F = C \cap \Sigma^{k-1}$

- The k-testable language is

$$\mathbf{L}(Z_k) = I\Sigma^* \cap \Sigma^* F - \Sigma^*(\Sigma^k - T)\Sigma^* \cup C$$

- Strings (of length at least k) have to use a good prefix and a good suffix of length k-1, and all sub-strings have to belong to T. Strings of length less than k should be in C
- Or: $\Sigma^k - T$ defines the prohibited segments
- Key idea: use a window of size k

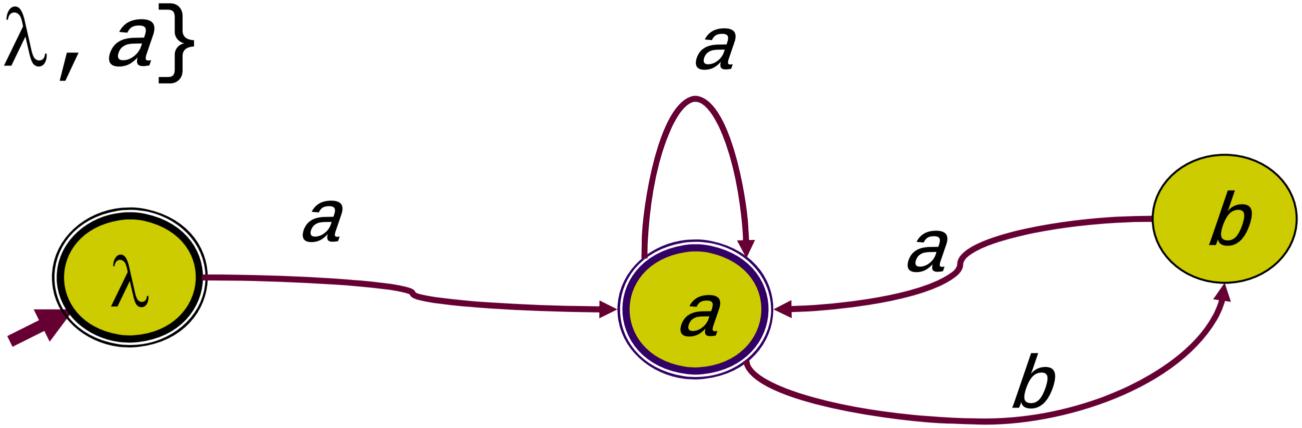
An example (2-testable)

I={*a*}

F={*a*}

T={*aa*, *ab*, *ba*}

C={λ, *a*}



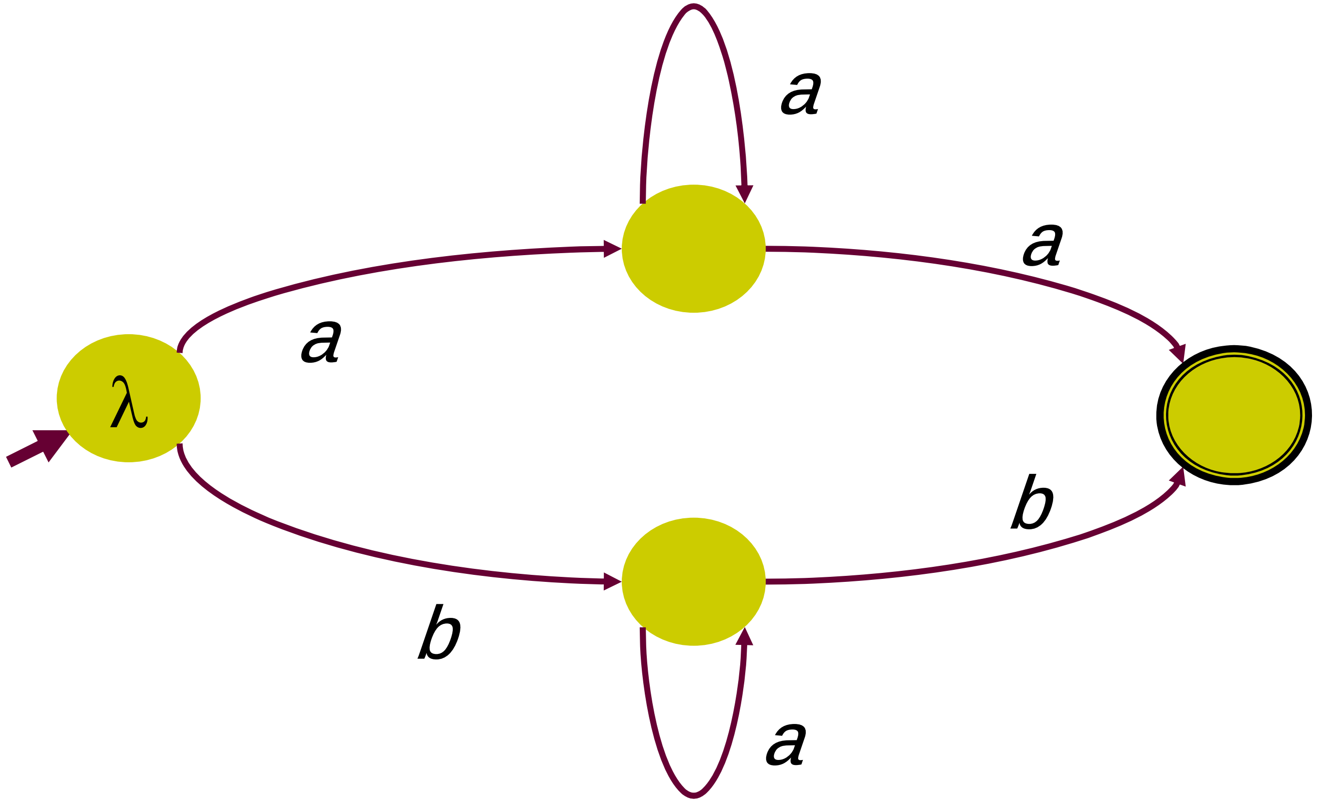
Window language

- By sliding a window of size 2 over a string we can parse
- *ababaaababababaaaab* OK
- *aaabbaaaababab* not OK

The hierarchy of *k-TSS* languages

- *k-TSS*(Σ)={$L \subseteq \Sigma^*$: L is *k-TSS*}
- All finite languages are in *k-TSS*(Σ) if k is large enough!
- *k-TSS*(Σ) $\subset$ [k+1]-*TSS*(Σ)
- $(ba^k)^* \in$ [k+1]-*TSS*(Σ)
- $(ba^k)^* \notin$ *k-TSS*(Σ)

A language that is not *k*-testable



K-TSS inference

Given a sample S, $\mathbf{a}_{k\text{-}TSS}(S)= L(Z_k)$ where $Z_k=(\Sigma(S), I(S), F(S), T(S), C(S))$ and

- $\Sigma(S)$ is the alphabet used in S
- $C(S)=\Sigma(S)^{<k}\cap S$
- $I(S)=\Sigma(S)^{k-1}\cap \mathrm{Pref}(S)$
- $F(S)= \Sigma(S)^{k-1}\cap \mathrm{Suff}(S)$
- $T(S)=\Sigma(S)^{k} \cap \{v: uvw\in S\}$

Example

- S={a, aa, $abba$, $abbbba$}
- Let k=3
  - $\Sigma(S)$={a, b}
  - $I(S)$= {aa, ab}
  - $F(S)$= {aa, ba}
  - $C(S)$= {a , aa}
  - $T(S)$={abb, bbb, bba}
- Hence $\mathbf{a}_{k\text{-}TSS}(S)= ab^*a+a$

Building the corresponding automaton

- Each string in $I \cup C$ and PREF($I \cup C$) is a state
- Each substring of length k-1 of strings in T is a state
- λ is the initial state
- Add a transition labeled b from u to ub for each state ub
- Add a transition labeled b from au to ub for each aub in T
- Each state/substring that is in F is a final state
- Each state/substring that is in C is a final state

Running the algorithm

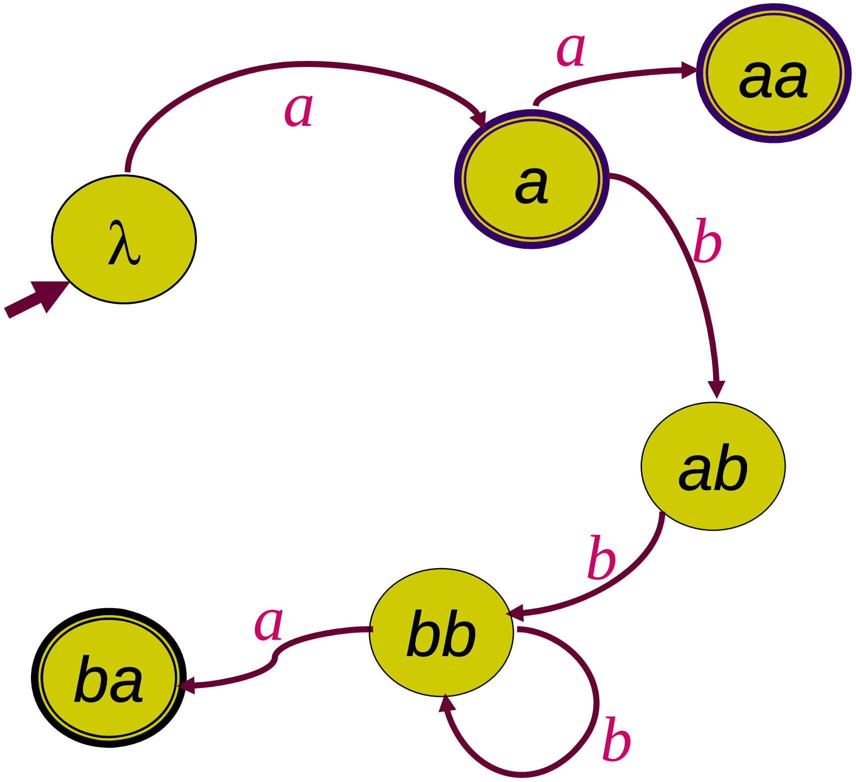
S={*a*, *aa*, *abba*, *abbbba*}

I={*aa*, *ab*}

F={*aa*, *ba*}

T={*abb*, *bbb*, *bba*}

C={*a*, *aa*}



Properties (1)

- $S \subseteq \mathbf{a}_{k\text{-}TSS}(S)$

- $\mathbf{a}_{k\text{-}TSS}(S)$ is the smallest k-TSS language that contains S
  - *If there is a smaller one, some prefix, suffix or substring has to be absent*

Properties (2)

- $\mathbf{a}_{k\text{-}TSS}$ identifies any k-TSS language in the limit from polynomial data
  - Once all the prefixes, suffixes and substrings have been seen, the correct automaton is returned
- If $Y \subseteq S$, $\mathbf{a}_{k\text{-}TSS}(Y) \subseteq \mathbf{a}_{k\text{-}TSS}(S)$

Properties (3)

- $\alpha_{k+1\text{-}TSS}(S) \subseteq \alpha_{k\text{-}TSS}(S)$
  - In I_{k+1} (resp. F_{k+1} and T_{k+1}) there are less allowed prefixes (resp. suffixes or substrings) than in I_k (resp. F_k and T_k)
- $\forall k > \max_{x \in S} |x|$, $\alpha_{k\text{-}TSS}(S) = S$
  - Because for a large k, $T_k(S) = \emptyset$

Extensions

- These languages have been studied and adapted to:
  - Local languages
  - *N*-grams
  - Tree languages

8 Learning actively

- *Learning regular sets from queries and counter-examples, D. Angluin, Information and computation, 75, 87-106, 1987*
- *Queries and Concept learning, D. Angluin, Machine Learning, 2, 319-342, 1988*
- *Negative results for Equivalence Queries, D. Angluin, Machine Learning, 5, 121-150, 1990*

8.1 About learning with *queries*

- Ideas:
  - define a credible learning model
  - make use of additional information that can be measured
  - explain thus the difficulty of learning certain classes

The Oracle

- knows the language and has to answer correctly
- no probabilities
- worse case policy: the Oracle does not want to help

Some *queries*

- membership *queries*
- equivalence *queries* (weak)
- equivalence *queries* (strong)
- inclusion *queries*

Membership *queries*.

L is the target language

Equivalence (weak) *queries*.

Yes if $L \equiv \mathbb{L}(h)$

No if $\exists x \in \Sigma^* : x \in \mathbb{L}(h) \oplus L$

$A \oplus B$ is the symmetric difference

Equivalence (strong) *queries*.

Yes if $\mathcal{L} \equiv h$

$x \in \Sigma^*$: $x \in \mathbb{L}(h) \oplus \mathcal{L}$ if not

Subset *queries*.

Yes if $\mathbf{L}(h) \subseteq L$

$x \in \Sigma^*$: $x \in \mathbf{L}(h) \wedge x \notin L$

if not

Correct learning

A class $\mathcal{C}$ is identifiable with a polynomial number of *queries* of type *T* if there exists an algorithm **a** that:

1) $\forall L \in \mathcal{C}$ identifies *L* with a polynomial number of *queries* of type *T*
2) does each update in time polynomial in $|f|$ and in $\Sigma|x_i|$, $\{x_i\}$ counter-examples seen so far

8.2 The Minimal Adequate Teacher

- You are allowed:
  - strong equivalence queries
  - membership queries

General idea of *L**

- find a consistent table (representing a *DFA*)
- submit it as an *equivalence query*
- use counterexample to update the table
- submit *membership queries* to make the table complete
- iterate

8.3 An observation table

	λ	a
λ	1	0
a	0	0
b	1	0
aa	0	0
ab	1	0

The experiments (E)

	λ	a
λ	1	0
a	0	0
b	1	0
aa	0	0
ab	1	0

The states (S)

The transitions (T)

Meaning

	λ	a
λ	1	0
a	0	0
b	1	0
aa	0	0
ab	1	0

$$\delta(q_I, \lambda.\lambda) \in F$$

$$\Leftrightarrow$$

$$\lambda \in L$$

	λ	a
λ	1	0
a	0	0
b	1	0
aa	0	0
ab	1	0

$$\delta(q_I,\ ab.a) \notin F$$

$$\Leftrightarrow$$

$$aba \notin L$$

Equivalent prefixes

	λ	a
λ	1	0
a	0	0
b	1	0
aa	0	0
ab	1	0

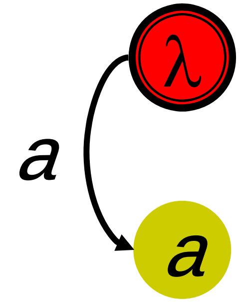
These two rows are equal, hence

$$\delta(q_I,\lambda) = \delta(q_I,ab)$$

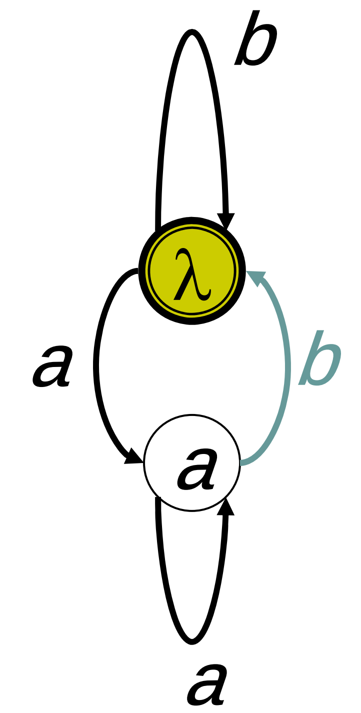
Cdlh 2010

Building a *DFA* from a table

	λ	*a*
λ	1	0
a	0	0
b	1	0
aa	0	0
ab	1	0



	λ	a
λ	1	0
a	0	0
b	1	0
aa	0	0
ab	1	0



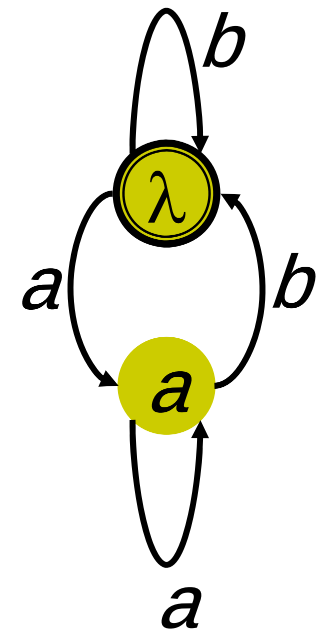
Some rules

This set is suffix-closed

This set is prefix-closed

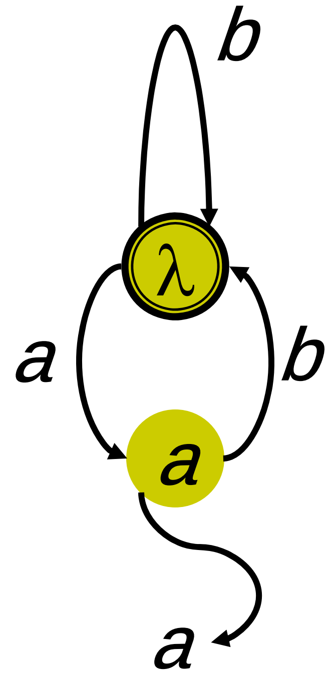
	λ	a
λ	1	0
a	0	0
b	1	0
aa	0	0
ab	1	0

$Red\Sigma \setminus Red = Blue$



An incomplete table

	λ	a
λ	1	0
a	0	
b	1	0
aa		0
ab	1	0



Good idea

We can complete the table by submitting membership queries...

		v
u		?

Membership query:

$uv \in L$?

A table is

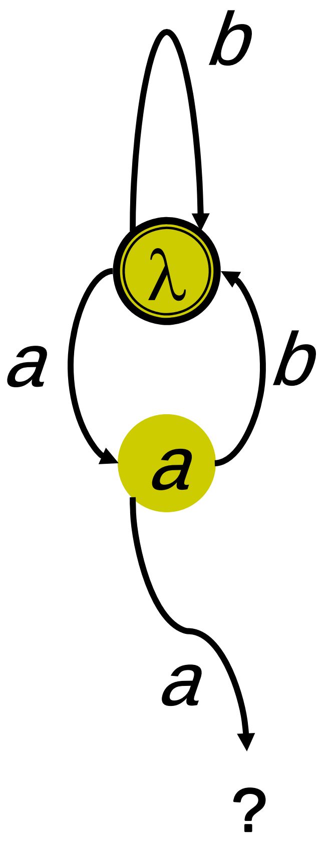
closed if any row of *Blue* corresponds to some row in *Red*

	λ	a
λ	1	0
a	0	0
b	1	0
aa	0	1
ab	1	0

Not closed

And a table that is not *closed*

	λ	a
λ	1	0
a	0	0
b	1	0
aa	0	1
ab	1	0



What do we do when we have a table that is not closed?

- Let s be the row (of *Blue*) that does not appear in *Red*
- Add s to *Red*, and $\forall a \in \Sigma$ sa to *Blue*

An inconsistent table

	λ	a
λ	1	0
a	0	0
b	0	0
aa	1	0
ab	1	0
ba	1	0
bb	0	0

Are *a* and *b* equivalent?

A table is consistent if

Every equivalent pair of rows in *Red* remains equivalent in *Red* ∪ *Blue* after appending any symbol

$$\text{row}(s_1)=\text{row}(s_2)$$

$$\Rightarrow$$

$$\forall a\in\Sigma,\ \text{row}(s_1a)=\text{row}(s_2a)$$

What do we do when we have an inconsistent table?

Let $a \in \Sigma$ be such that row(s_1)=row(s_2) but row(s_1a)$\neq$row(s_2a)

- If row(s_1a)$\neq$row(s_2a), it is so for experiment e
- Then add experiment ae to the table

What do we do when we have a closed and consistent table ?

- We build the corresponding *DFA*
- We make an equivalence query!!!

What do we do if we get a counter-example?

- Let u be this counter-example
- $\forall w \in \mathrm{Pref}(u)$ do
  - add w to *Red*
  - $\forall a \in \Sigma$, such that $wa \notin$ *Red* add wa to *Blue*

8.4 Run of the algorithm

	λ
λ	1
a	1
b	1

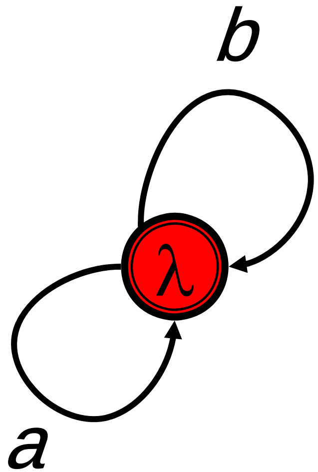
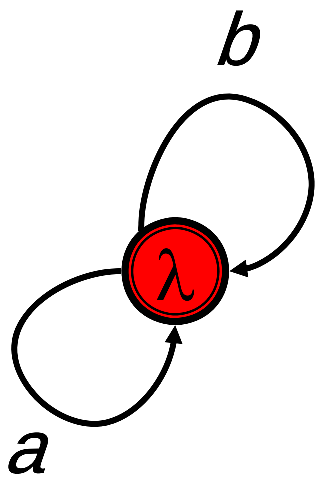


Table is now closed and consistent

An equivalence query is made!



Counter example ***baa*** is returned

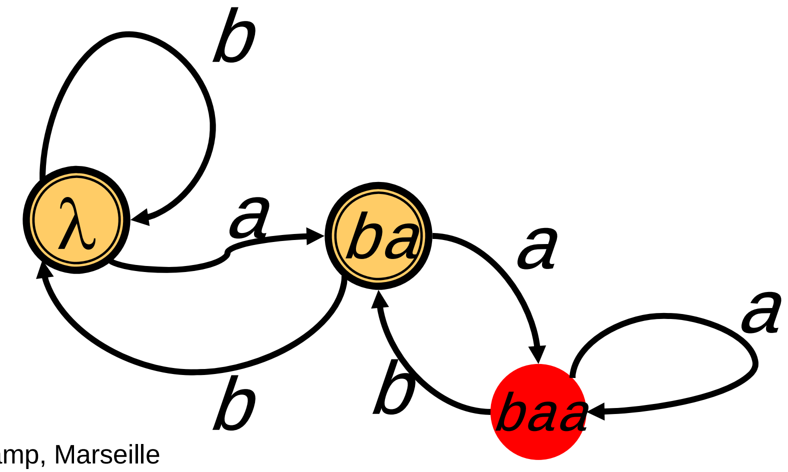
	λ
λ	1
b	1
ba	1
baa	0
a	1
bb	1
bab	1
baaa	0
baab	1

Not consistent

Because of

	λ	a
λ	1	1
b	1	1
ba	1	0
baa	0	0
a	1	0
bb	1	1
bab	1	1
$baaa$	0	0
$baab$	1	0

Table is now closed and consistent



Polynomial

- $|E| \leq n$
- at most n-1 equivalence queries
- $|membership\ queries| \leq n(n-1)m$ where m is the length of the longest counter-example returned by the oracle

Conclusion (1)

- With an *MAT* you can learn *DFA*
  - but also a variety of other classes of grammars
  - it is difficult to see how powerful is really an *MAT*
  - probably as much as *PAC* learning
  - Easy to find a class, a set of queries and provide and algorithm that learns with them
  - more difficult for it to be meaningful
- Discussion: why are these queries meaningful?

Conclusion (2)

- Active learning is an exciting topic, and good strategies for choosing the queries are still largely unexplored
- Zulu competition can be a great opportunity to start research in this area
- http://cian.univ-st-etienne.fr/zulu/

9 Extensions (PFA, transducers, tree automata)

- Theory, algorithms and applications have extended to:
  - Transducers
  - Probabilistic finite automata
  - Context free grammars (with special interest in linear grammars)
  - String kernels
  - Regular expressions
  - patterns

Main results for learning PFA

There are now several DPFA learning algorithms

ALERGIA (Carrasco & Oncina 94)

DSAI (Ron el al. 94)

MDI (Thollard et al. 99)

DEES (Denis et al. 05) [also PFA]

Main results for learning transducers

- One basic algorithm : OSTIA (Oncina et al. 93)
- State merging algorithm, based on a normal form for subsequencial transducers

10 Conclusions

Why should one pick up grammatical inference as a research topic?

- Nice community
- Broad field
- Can use ideas from algorithmics, formal language theory, combinatorics, statistics, machine learning, natural language processing, bio-informatics, pattern recognition…
- Theory and applications

Open problems

- C. de la Higuera. A bibliographical study of grammatical inference. *Pattern Recognition*, 38:1332-1348, 2005
- C. de la Higuera. Ten open problems in grammatical inference. In proceedings of ICGI 2006, pages 32-44

Some addresses to start working

http://pages-perso.univ-nantes.fr/~cdlh/
http://videolectures.net/colin_de_la_higuera/
http://cian.univ-st-etienne.fr/zulu/

Grammatical Inference: Learning Automata and Grammars, Colin de la Higuera, Cambridge University Press