LABORATOIRE D'INFORMATIQUE
DE NANTES ATLANTIQUE
UMR CNRS 6241
Université de Nantes
Ecole des Mines de Nantes

Active Learning

2010

Colin de la Higuera

Pattern Analysis, Statistical Modelling and Computational Learning

dépasser les frontières

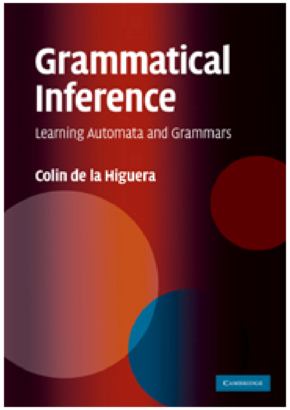
Région
PAYS DE LA LOIRE

Acknowledgements

- Laurent Miclet, Jose Oncina and Tim Oates for collaboration previous versions of these slides.
- Rafael Carrasco, Paco Casacuberta, Rémi Eyraud, Philippe Ezequel, Henning Fernau, Thierry Murgue, Franck Thollard, Enrique Vidal, Frédéric Tantini,...
- List is necessarily incomplete. Excuses to those that have been forgotten.

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

Book, chapters 9 and13



Outline

1. Motivations and applications
2. The learning model
3. Some negative results
4. Algorithm L*
5. Some implementation issues
6. Extensions
7. Conclusion

0 General idea

- The learning algorithm (he) is allowed to interact with its environment through queries
- The environment is formalised by an oracle (she)

- Also called learning from queries or oracle learning

1 Motivations

Goals

- define a credible learning model
- make use of additional information that can be measured
- explain thus the difficulty of learning certain classes
- solve real life problems

Application: robotics

- A robot has to find a route in a maze
- The maze is represented as a graph
- The robot finds his way and can experiment
- The robot gets feedback

- *Dean, T., Basye, K., Kaelbling, L., Kokkevis, E., Maron, O., Angluin, D., Engelson, S.: Inferring finite automata with stochastic output functions and an application to map learning. In Swartout, W., ed.: Proceedings of the 10th National Conference on Artificial Intelligence, San Jose, CA, Mit Press (1992) 208–214*
- *Rivest, R.L., Schapire, R.E.: Inference of finite automata using homing sequences. Information and Computation 103 (1993) 299–347*

Application: web wrapper induction

- System SQUIRREL learns tree automata
- Goal is to learn a tree automaton which, when run on XML, returns selected items

Carme, J., Gilleron, R., Lemay, A., Niehren, J.: Interactive learning of node selecting tree transducer. Machine Learning Journal 66(1) (2007) 33–67

Applications: under resourced languages

- When a language does not have enough data for statistical methods to be of interest, use of human expert for labelling
- This is the case for most languages
- Examples
  - Interactive predictive parsing
  - Computer aided translation

Checking models

- An electronic system can be modelled by a finite graph (a DFA)
- Checking if a chip meets its specification can be done by testing or by trying to learn the specification with queries

- *Bréhélin, L., Gascuel, O., Caraux, G.: Hidden Markov models with patterns to learn boolean vector sequences and application to the built-in self-test for integrated circuits. Pattern Analysis and Machine Intelligence 23(9) (2001) 997–1008*
- *Berg, T., Grinchtein, O., Jonsson, B., Leucker, M., Raffelt, H., Steffen, B.: On the correspondence between conformance testing and regular inference. In: Proceedings of Fundamental Approaches to Software Engineering, 8th International Conference, FASE 2005. Volume 3442 of Lncs., Springer-Verlag (2005) 175–189*
- *Raffelt, H., Steffen, B.: Learnlib: A library for automata learning and experimentation. In: Proceedings of Fase 2006. Volume 3922 of Lncs., Springer-Verlag (2006) 377–380*

Playing games

- *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*

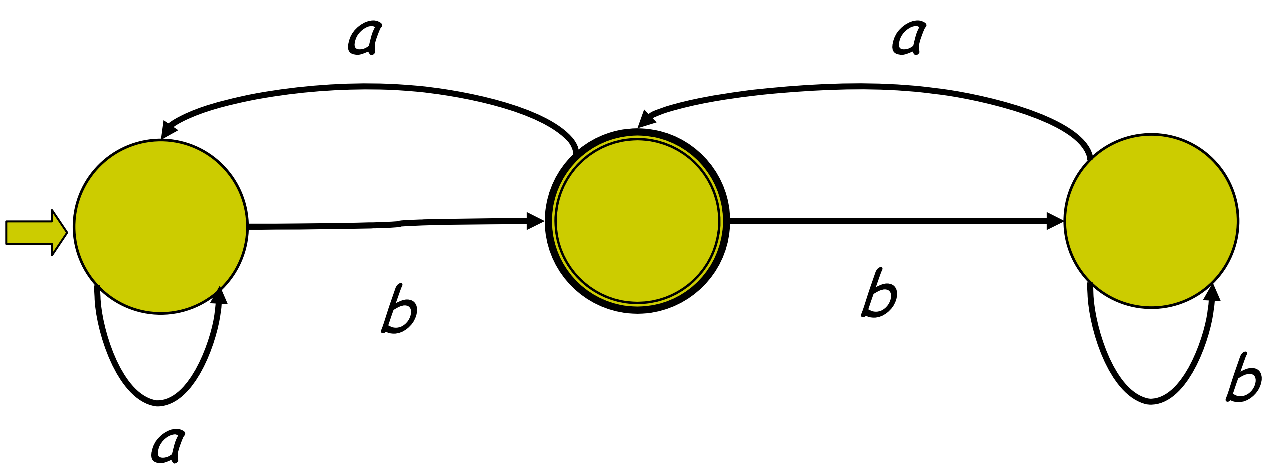
2. The model

Notations

- We denote by T the target grammar or automaton
- We denote by H the current hypothesis
- We denote by $\mathbf{L}(H)$ and $\mathbf{L}(T)$ the corresponding languages
- Examples are x, y, z, with labels $\ell_T(x)$, $\ell_T(y)$ $\ell_T(z)$ for the real labels and $\ell_H(x)$, $\ell_H(y)$ $\ell_H(z)$ for the hypothesized ones.
- The computation of $\ell_I(x)$ must take place in time polynomial in $|x|$

Running example

- Suppose we are learning DFA
- The *running example* target is:



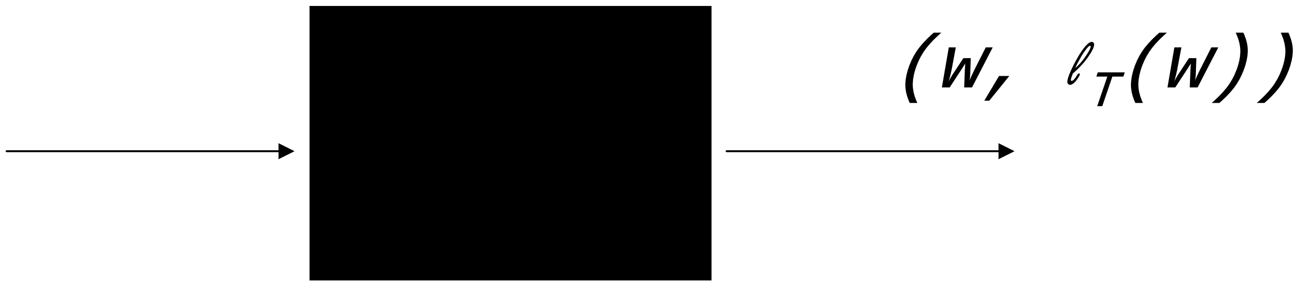
The Oracle

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

Some *queries*

1. sampling *queries*
2. presentation queries
3. membership *queries*
4. equivalence *queries* (weak or strong)
5. inclusion *queries*
6. correction *queries*
7. specific sampling *queries*
8. translation *queries*
9. probability *queries*

2.1 Sampling *queries* (Ex)

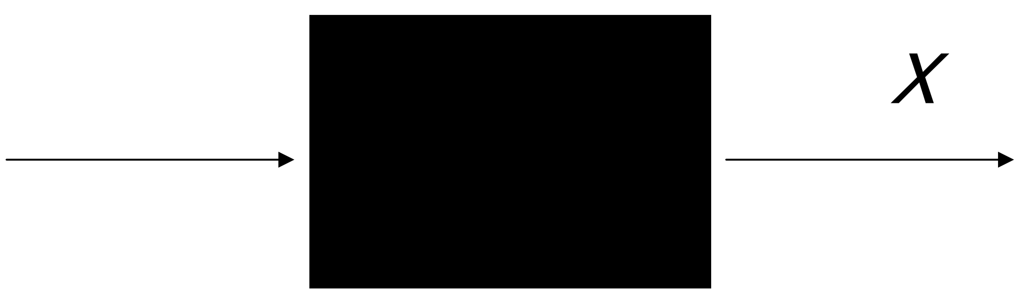


w is drawn following some unknown distribution

Sampling *queries* (Pos)

x is drawn following some unknown distribution, restricted to $\mathbb{L}(T)$

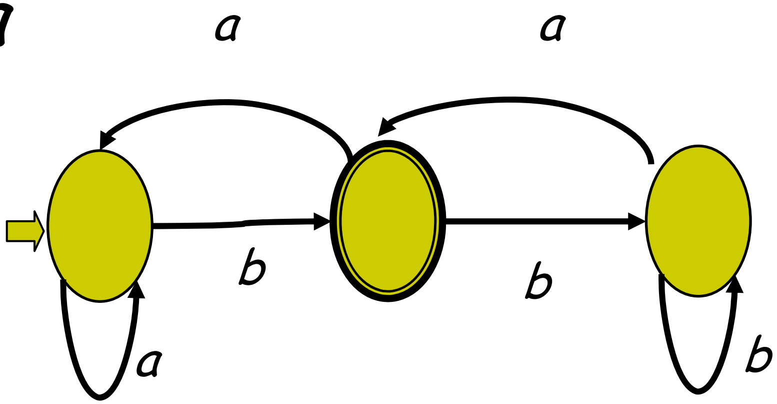
Sampling *queries* (Neg)



X is drawn following some unknown distribution, restricted to $\Sigma^* \setminus \mathbb{L}(T)$

Example

- Ex() might return (*aabab*,1) or (λ,0)
- Pos() might return *abab*
- Neg() might return *aa*



Needs a distribution over Σ^*, $\mathbf{L}(T)$ or $\Sigma^* \setminus \mathbf{L}(T)$

2.2 Presentation *queries*

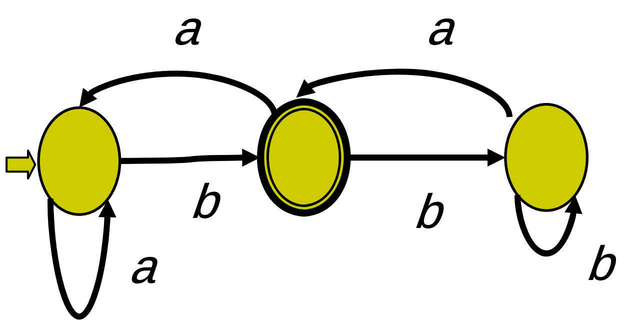
- A presentation of a language is an enumeration of all the strings in Σ^*, with a label indicating if a string belongs or not to $\mathbf{L}(T)$ (informed presentation),
- or an enumeration of all the strings in $\mathbf{L}(T)$ (text presentation)
- There can be repetitions

Presentation *queries*

f is a valid (unknown) presentation. Sub-cases can be ***text*** *or* ***informed*** *presentations*

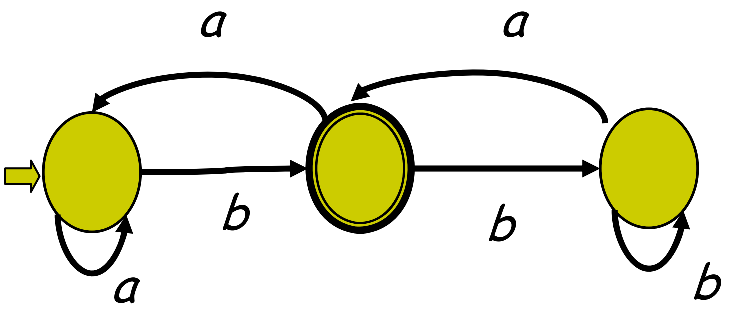
Example

- $\mathrm{Pres}_{text}(3)$ could be *bba*
- $\mathrm{Pres}_{text}(17)$ could be *abbaba*
- (the « selected » presentation being *b*, *ab*, *aab*, *bba*, *aaab*, *abab*, *abba*, *bbab*,…)

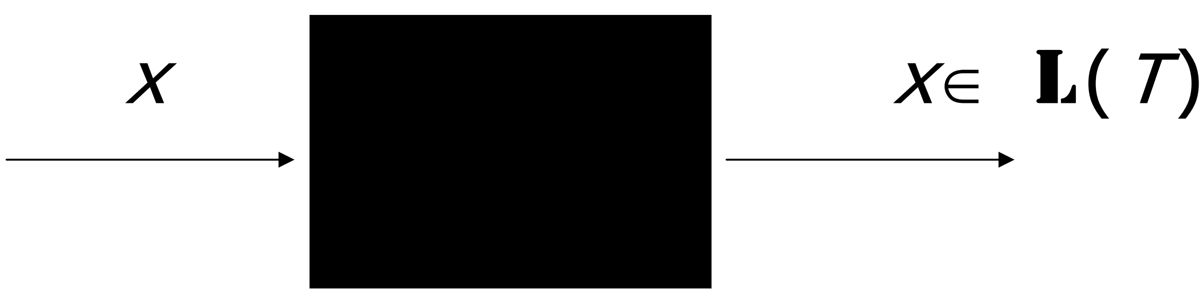


Example

- $Pres_{informed}(3)$ could be (*aab*,1)
- $Pres_{informed}(1)$ could be (*a*,0)
- (the « selected » presentation being (*b*,1),(*a*,0),(*aaa*,0),(*aab*,1),(*bba*,1),(*a*,0)…)



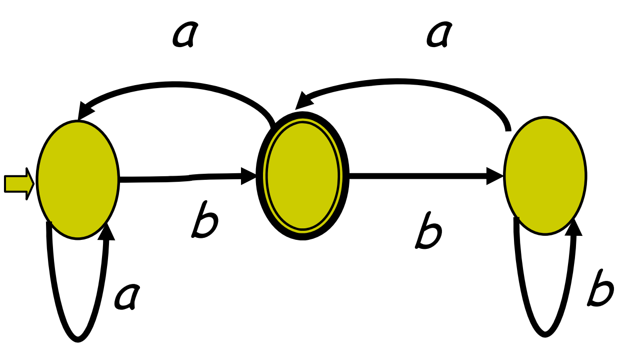
2.3 Membership *queries*.



$\mathbb{L}(T)$ is the target language

Example

- MQ(*aab*) returns 1 (or true)
- MQ(*bbb*) returns 0 (or false)



2.4 Equivalence (weak) *queries*.

Yes if $\mathbf{L}(T) = \mathbf{L}(H)$

No if $\exists x \in \Sigma^* : x \in \mathbf{L}(H) \oplus \mathbf{L}(T)$

$A \oplus B$ is the symmetric difference

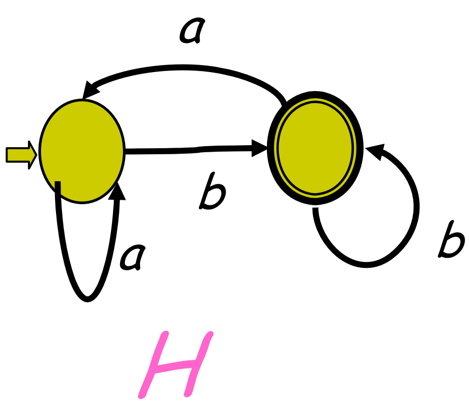
Equivalence (strong) *queries*.

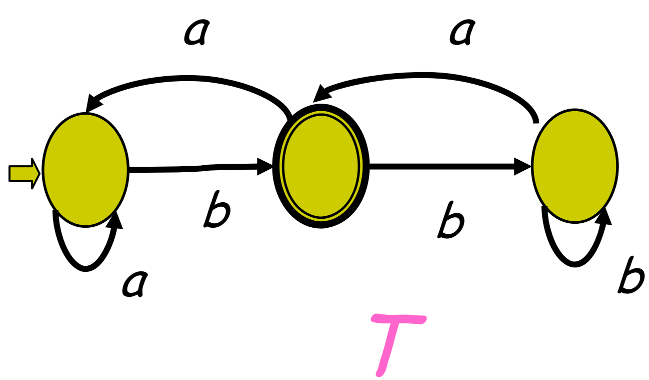
Yes if $T \equiv H$

$x \in \Sigma^*$: $x \in \mathbf{L}(H) \oplus \mathbf{L}(T)$ if not

Example

- EQ(*H*) returns *abbb* (or *abba*…)
- WEQ(*H*) returns false





2.5 Subset *queries*.

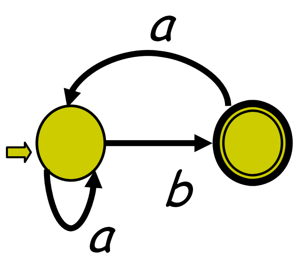
Yes if $\mathbf{L}(H) \subseteq \mathbf{L}(T)$

$x \in \Sigma^*$: $x \in \mathbf{L}(H) \wedge x \notin \mathbf{L}(T)$ if not

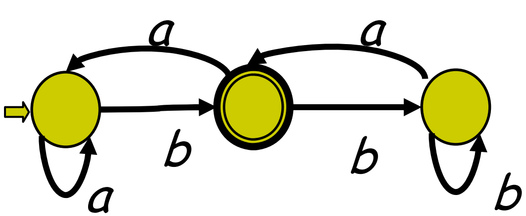
Example

- SSQ(H_1) returns true
- SSQ(H_2) returns *abbb*

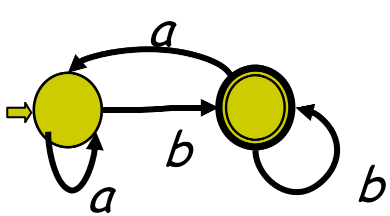
H_1



T



H_2



2.6 Correction *queries*.

Yes if $x \in \mathbb{L}(T)$

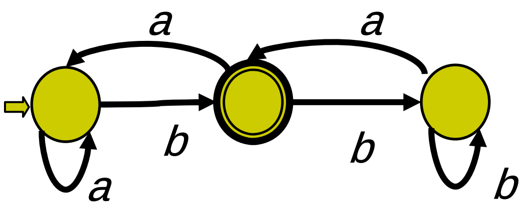
$y \in \mathbb{L}(T)$: y is a correction of x if not

Becerra-Bonache, L., de la Higuera, C., Janodet, J.C., Tantini, F.: Learning balls of strings from edit corrections. Journal of Machine Learning Research 9 (2008) 1841–1870

Kinber, E.B.: On learning regular expressions and patterns via membership and correction queries. [33] 125–138

Example

- CQ_{suff}(*bb*) returns *bba*
- CQ_{edit}(*bb*) returns any string in {*b*,*ab*,*bba*}
- CQ_{edit}(*bba*) and CQ_{suff}(*bba*) return true



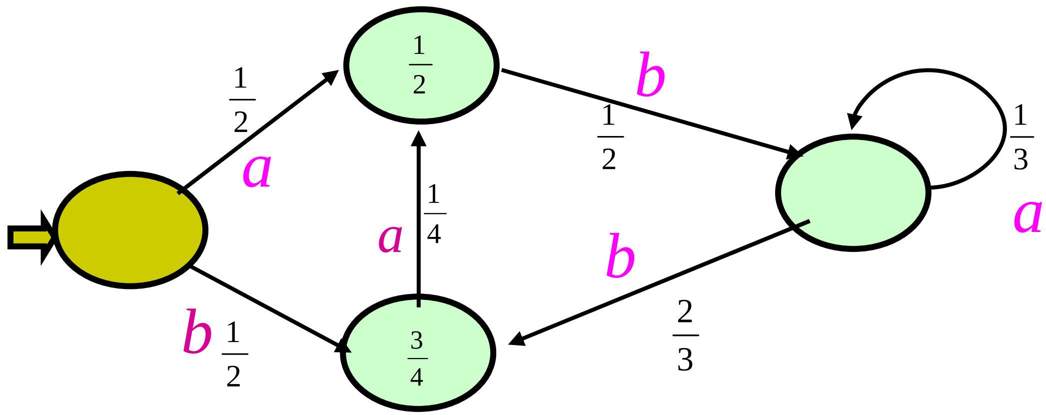
2.7 Specific sampling *queries*

- Submit a grammar G
- Oracle draws a string from $\mathbb{L}(G)$ and labels it according to T
- Requires an unknown distribution

- Allows for example to sample string starting with some specific prefix

2.8 Probability queries

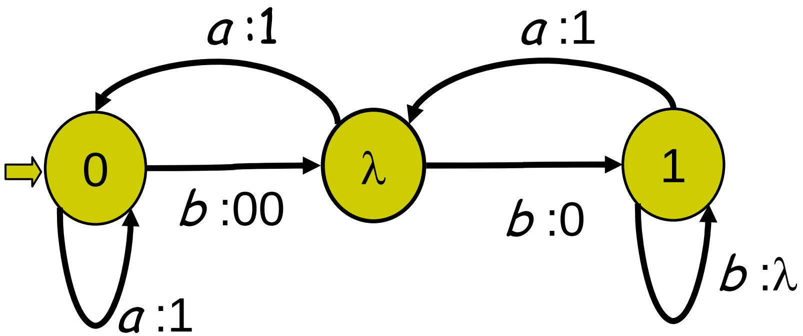
- Target is a PFA.
- Submit *w*, Oracle returns $Pr_T(w)$



String *ba* should have a relative frequency of 1/16

2.9 Translation queries

- Target is a transducer
- Submit a string. Oracle returns its translation



Tr(*ab*) returns 100
Tr(*bb*) returns 0001

Learning setting

Two things have to be decided

- The exact queries the learner is allowed to use
- The conditions to be met to say that learning has been achieved

What queries are we allowed?

- The combination of queries is declared
- Examples:
  - $\mathcal{Q}$={MQ}
  - $\mathcal{Q}$={MQ,EQ} (this is an MAT)

Defining learnablility

- Can be in terms of classes of languages or in terms of classes of grammars
- The size of a language is the size of the smallest grammar for that language
- Important issue: when does the learner stop asking questions?

You can't learn DFA with membership queries

- Indeed, suppose the target is a finite language
- Membership queries just add strings to the language
- But you can't stop and be sure of success

Correct learning

A class $\mathcal{C}$ is learnable with *queries* from $\mathcal{Q}$ if there exists an algorithm **a** such that:

$\forall L \in \mathcal{C}$, **a** makes a finite number of queries from $\mathcal{Q}$, halts and returns a grammar G such that $\mathbb{L}(G)=L$

We say that **a** learns $\mathcal{C}$ with queries from $\mathcal{Q}$

Received information

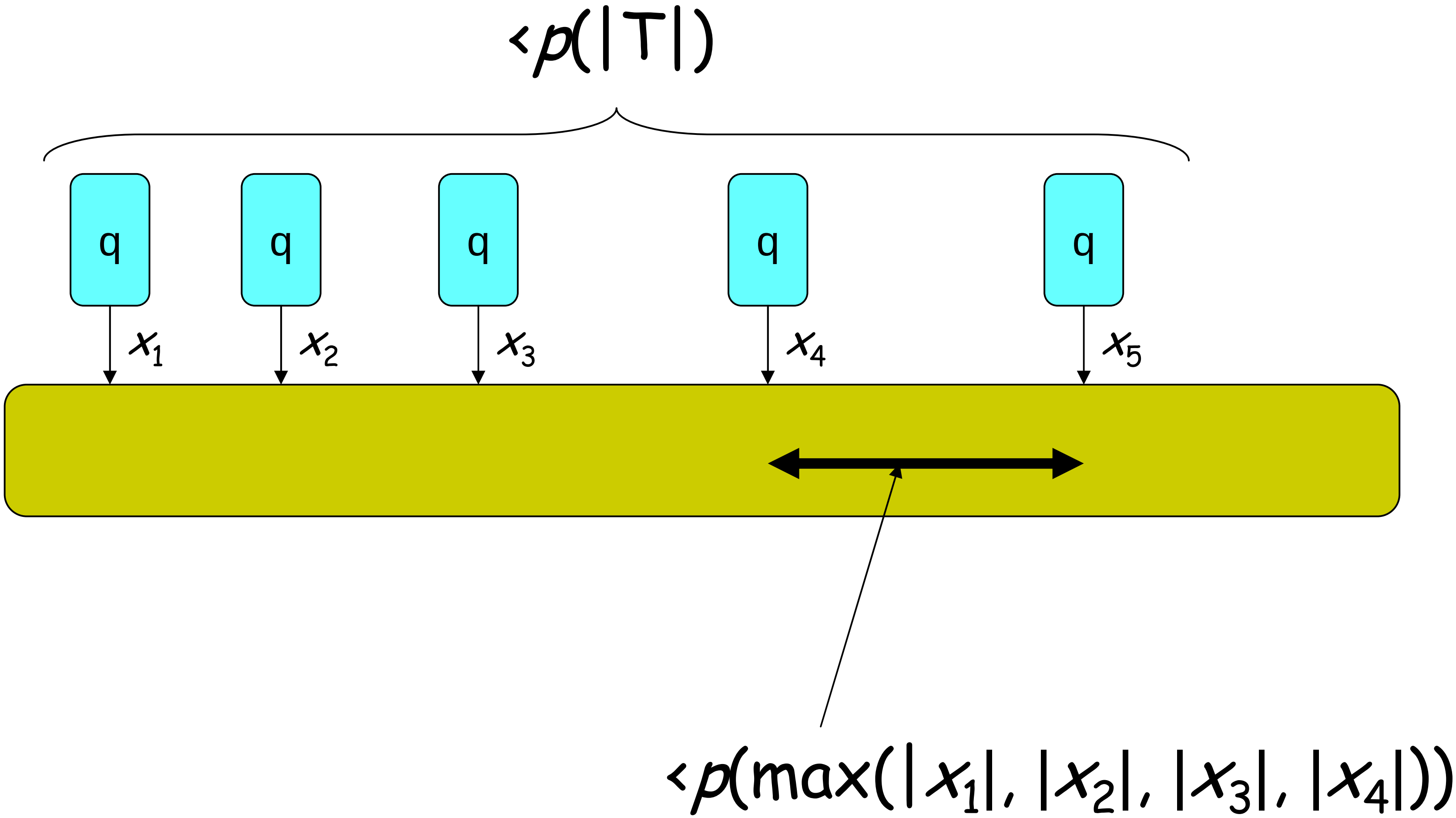
- Suppose that during a run ρ, the information received from the Oracle is stocked in a table $Info(\rho)$
- $Info_n(\rho)$ is the information received from the first n queries
- We denote by $mInfo_n(\rho)$ (resp. $mInfo(\rho)$) the size of the longest information received from the first n queries (resp. during the entire run ρ)

Polynomial update

- A polynomial $p(\cdot)$ is given
- After the n^{th} query in any run ρ, the runtime before the next query is in $O(p(m))$, where $m = mInfo_n(\rho)$

We say that **a** makes polynomial updates

Polynomial update



Correct learning

A class $\mathcal{C}$ is learnable with a polynomial number of *queries* from $\mathcal{Q}$ if there exists an algorithm **a** and a polynomial $q(\cdot, \cdot)$ such that:

1) $\forall L \in \mathcal{C}$ **a** learns L with queries from $\mathcal{Q}$
2) **a** makes polynomial updates
3) **a** uses during a run ρ at most $q(m, |L|)$ queries, where $m = mInfo(\rho)$

Comment

- In the previous definitions, the queries receive deterministic answers.
- When the queries have a probabilistic nature, things still have to be discussed as identification cannot be sure

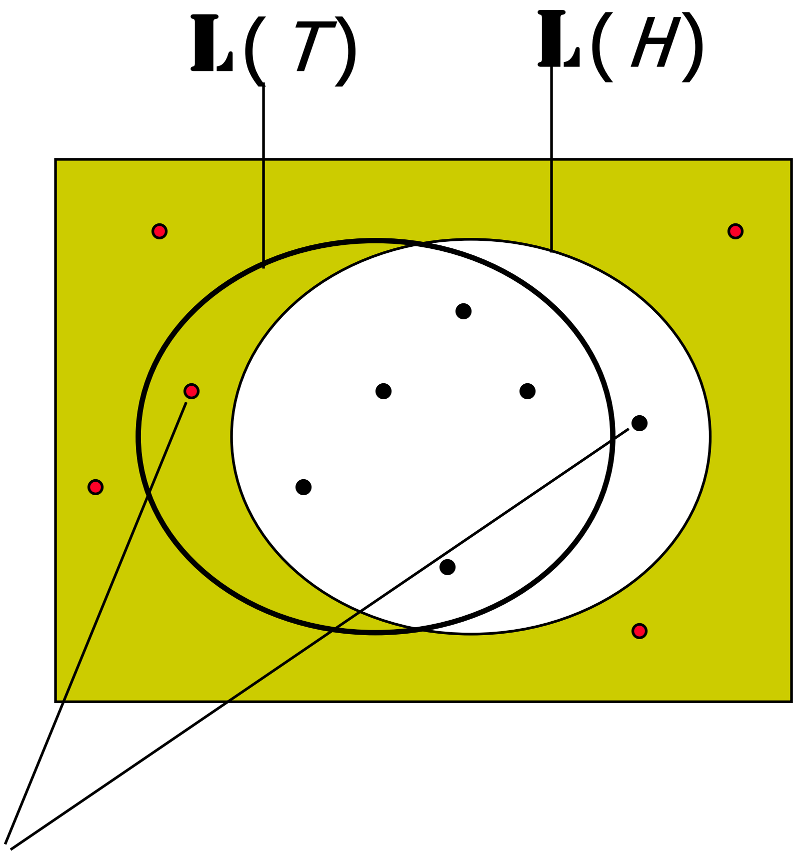
PAC learning
(Valiant 84, Pitt 89)

- $\mathcal{C}$ a set of languages
- $\mathcal{G}$ a set of grammars
- $\varepsilon>0$ and $\delta>0$
- m a maximal length over the strings
- n a maximal size of grammars

H is ε-**AC** (approximately correct)*

if

$\Pr_{D}[\ell_H(x) \neq \ell_T(x)] < \varepsilon$

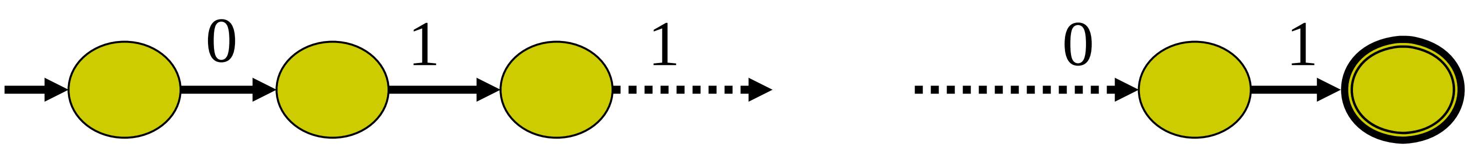


Errors: we want $Pr_D[\ell_H(x) \neq \ell_T(x)] < \varepsilon$

3 Negative results

3.1 Learning from membership queries alone

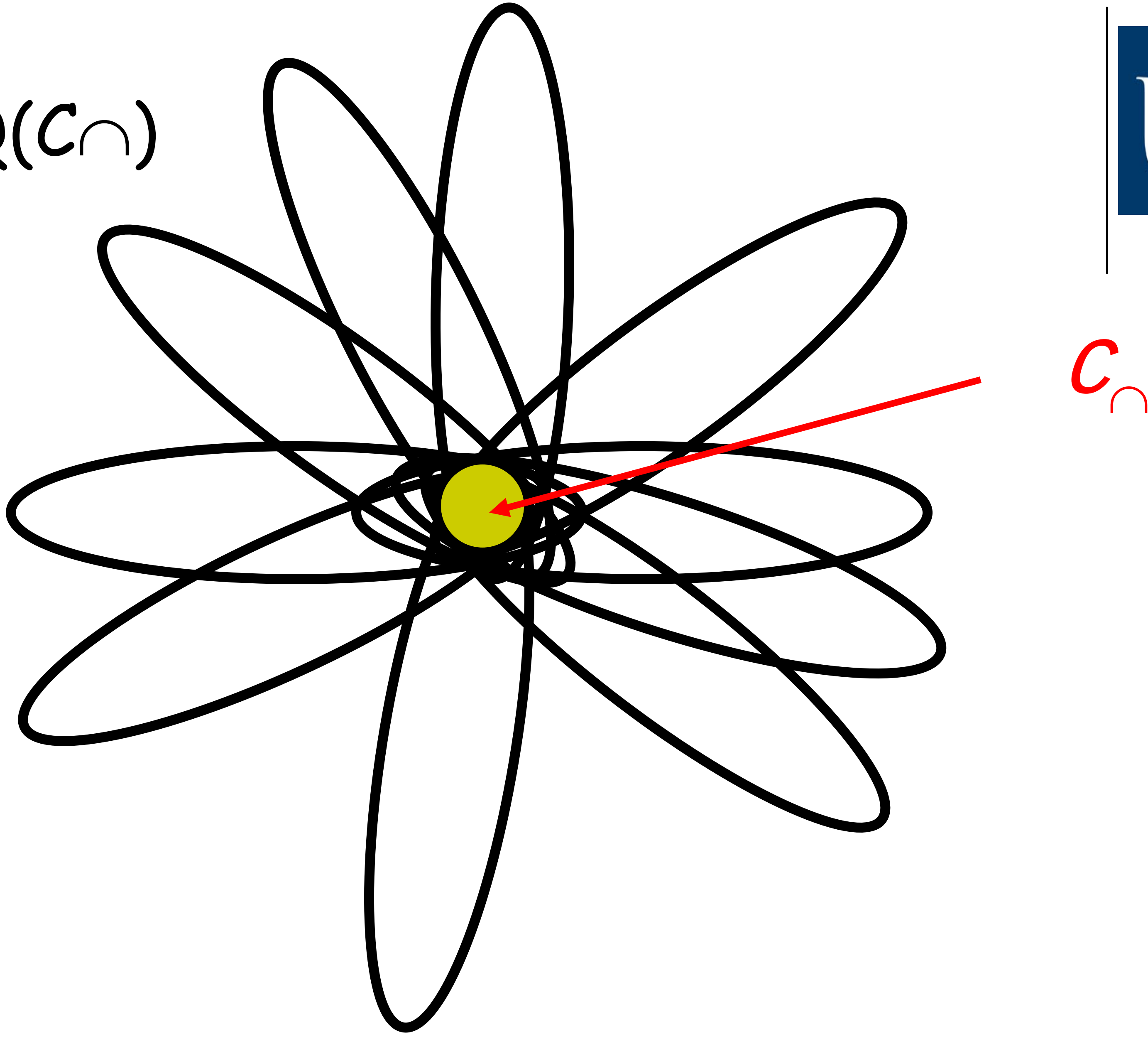
- Actually we can use subset queries and weak equivalence queries also, without doing much better.
- Intuition: keep in mind lock automata...



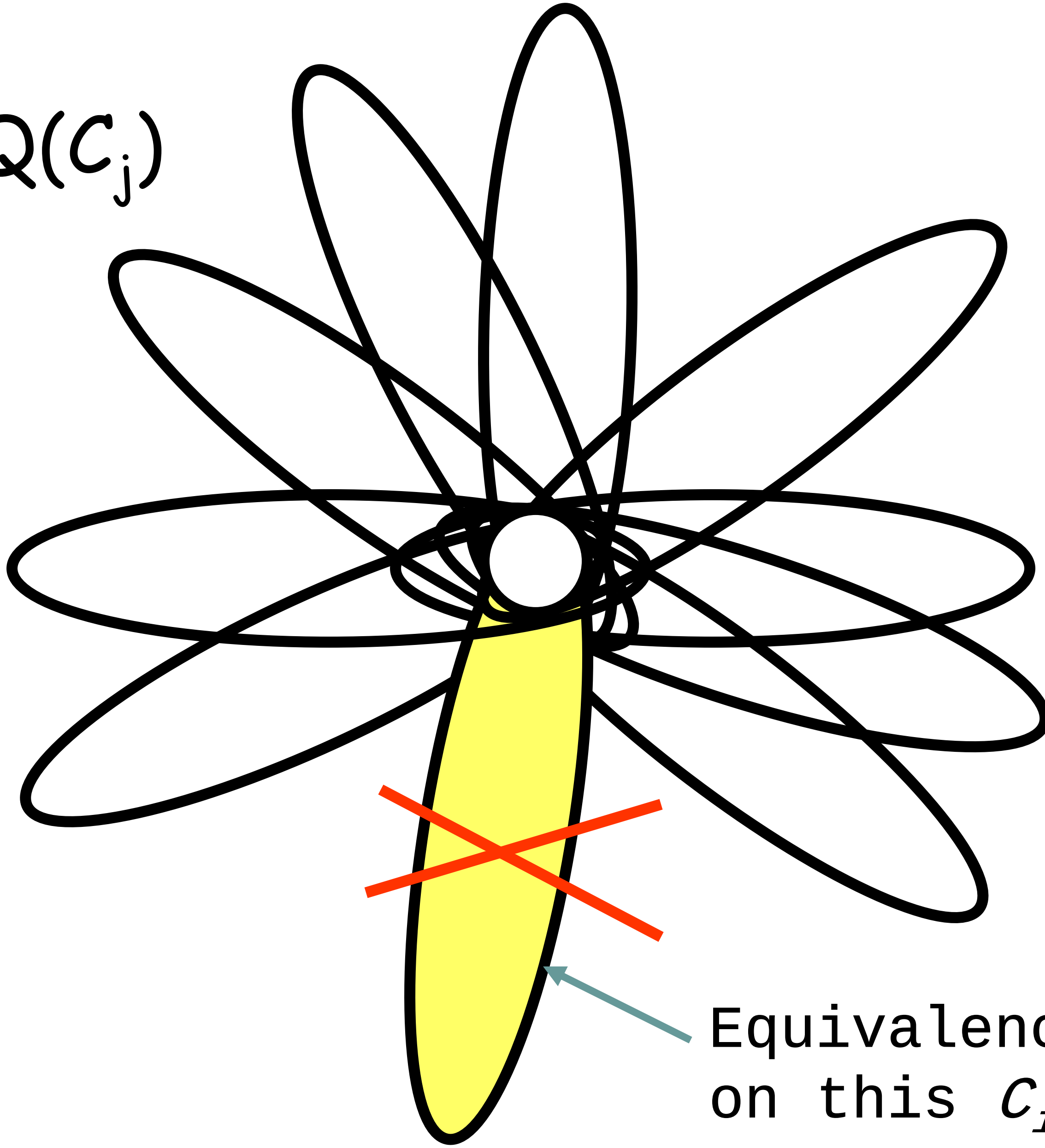
Lemma (Angluin 88)

- If a class $\mathcal{C}$ contains a set $C_\cap$ and n sets $C_1 \ldots C_n$ such that $\forall\ i, j \in [n] \quad C_i \cap C_j = C_\cap$, any algorithm using membership, weak equivalence and subset *queries* needs in the worse case to make n-1 *queries*.

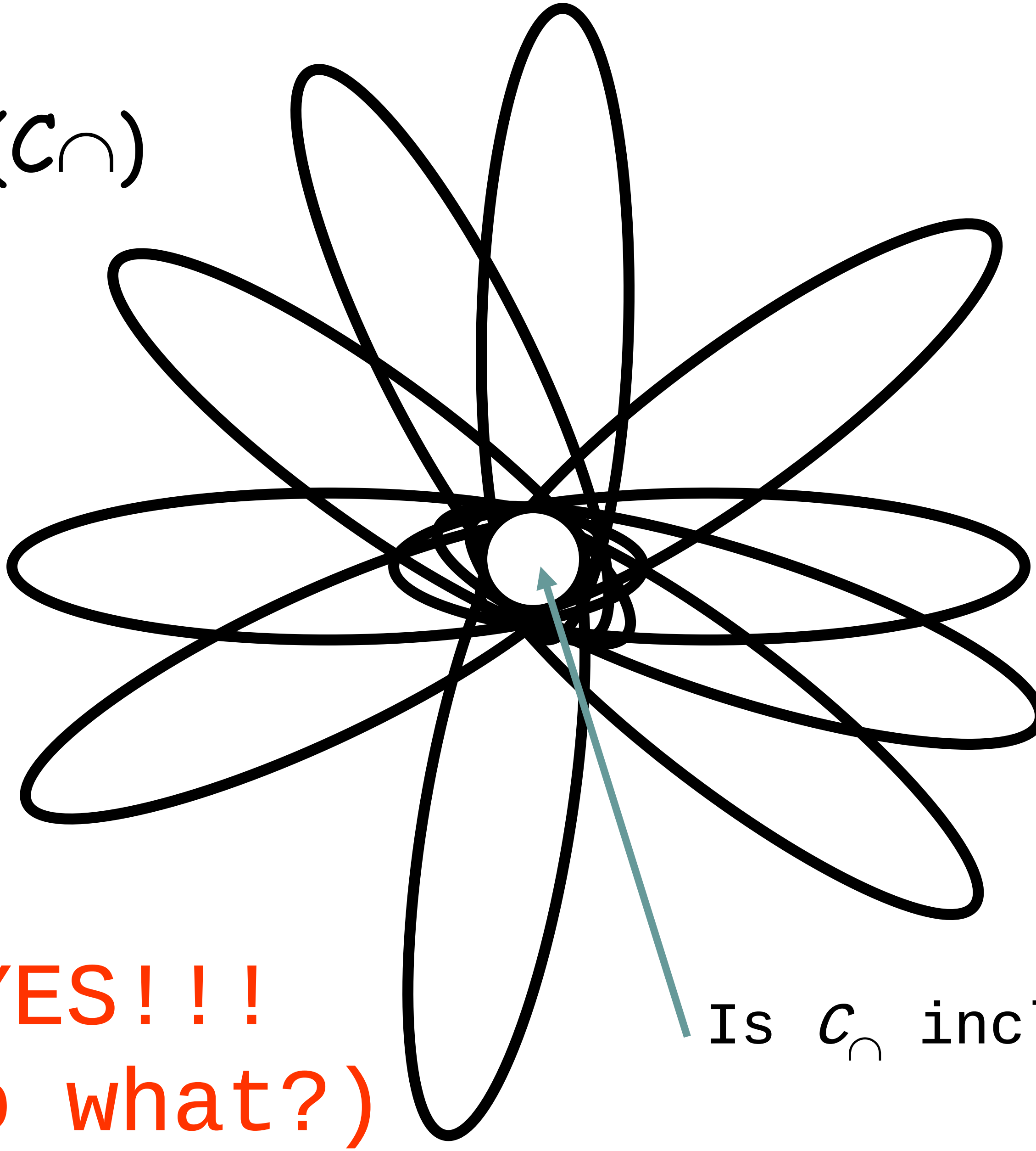
WEQ($\mathcal{C}_\cap$)



WEQ(C_j)



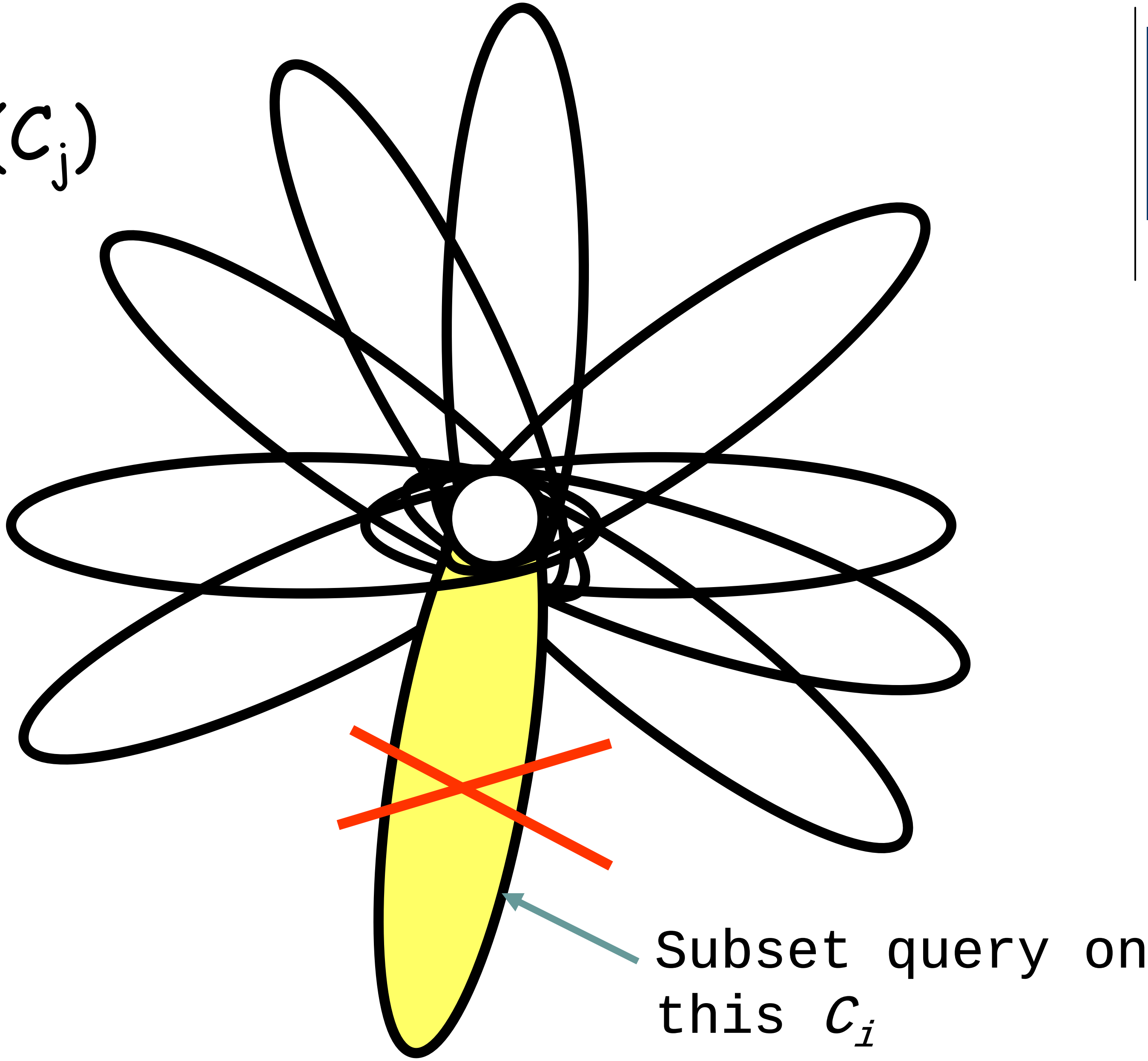
Sub($C_\cap$)



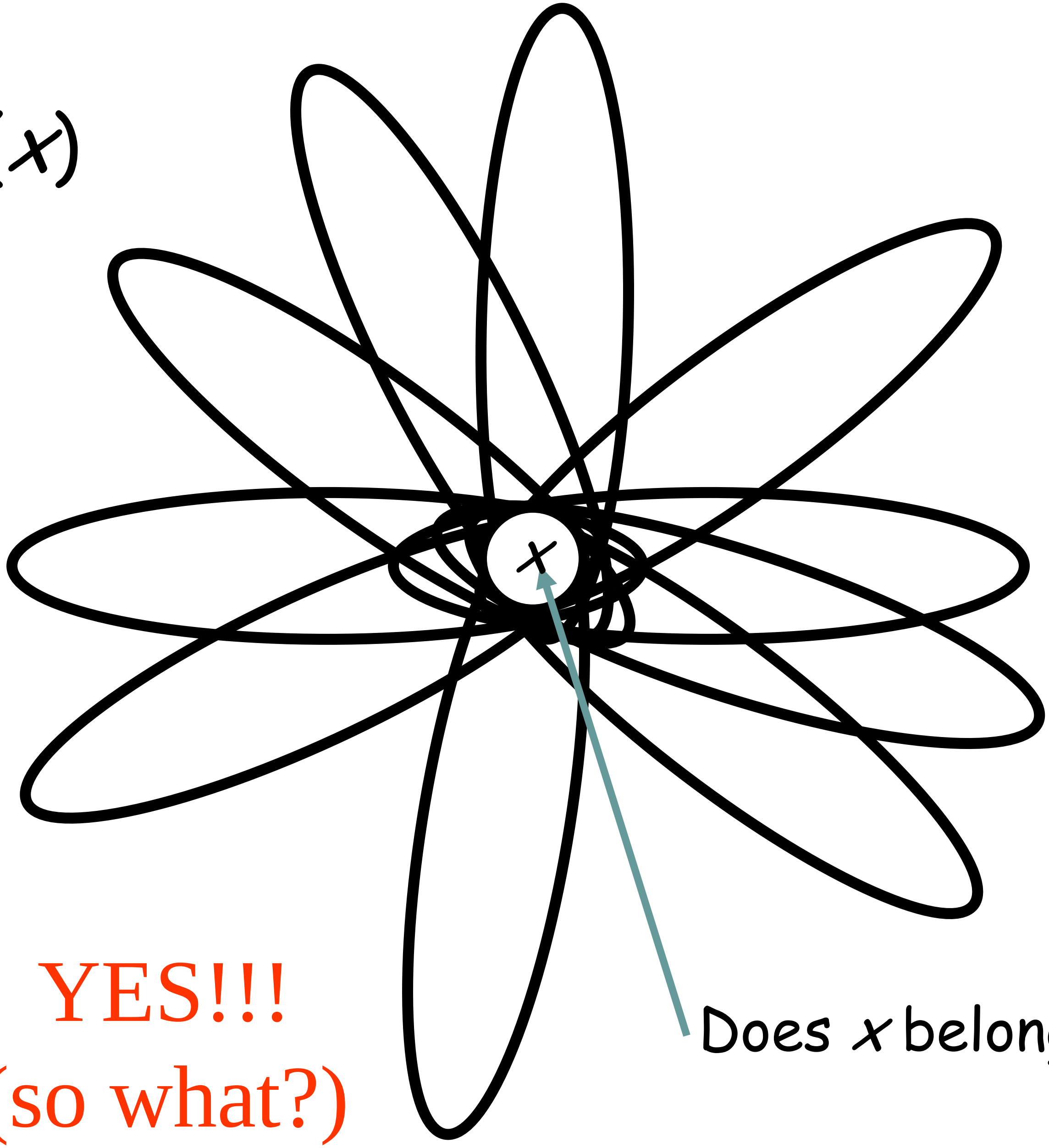
Is $C_\cap$ included?

YES!!!
(so what?)

Sub(C_j)



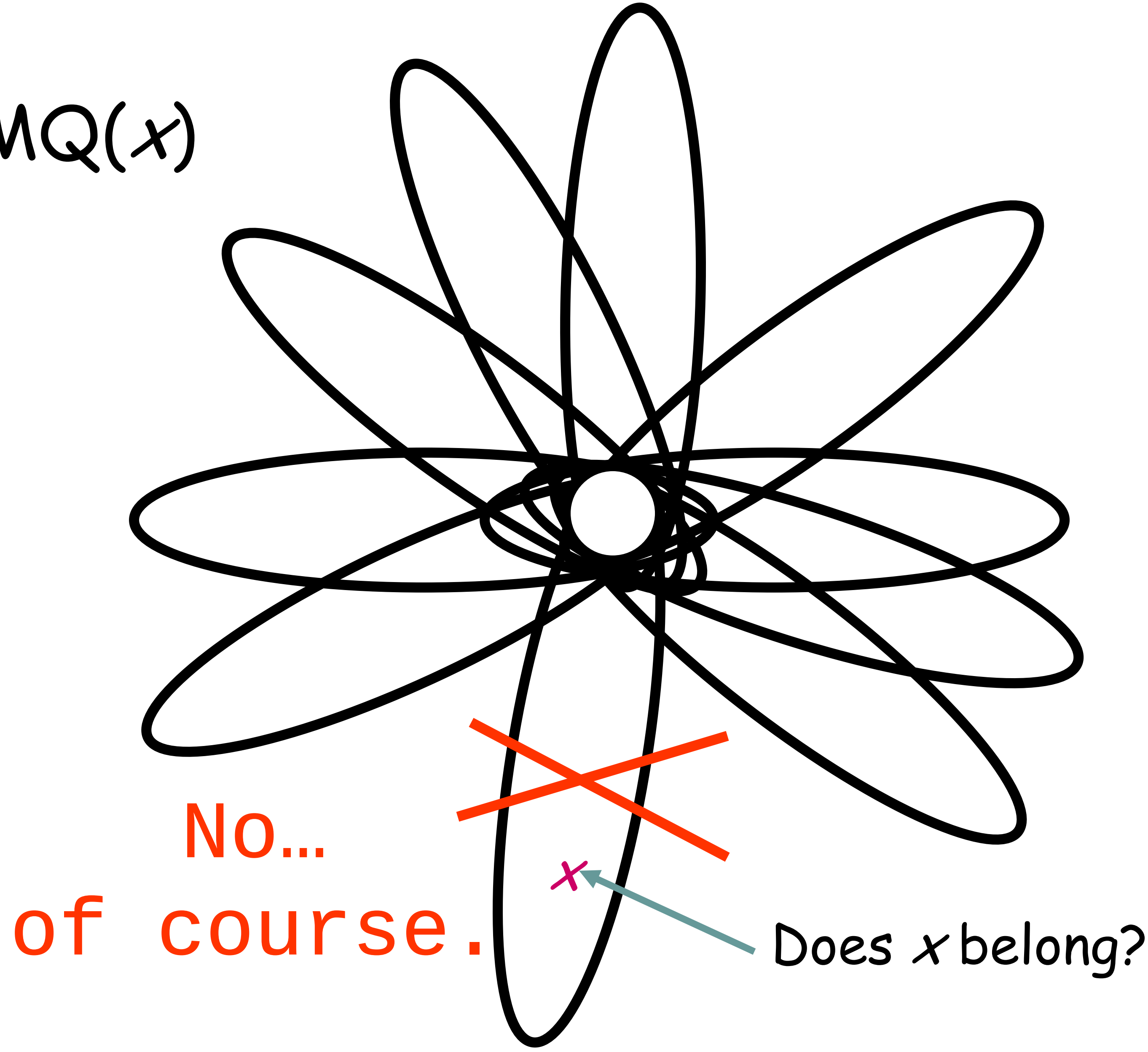
MQ(x)



Does x belong?

YES!!!
(so what?)

$MQ(x)$



Proof (summarised)

Query	Answer	Action
WEQ(C_i)	No	eliminates C_i
SSQ($C_\cap$)	Yes	eliminates nothing
SSQ(C_i)	No	eliminates nothing
MQ(x) ($\in C_\cap$)	Yes	eliminates nothing
MQ(x) ($\notin C_\cap$)	No	eliminates C_i such that $x \in C_i$

Corollary

Let DFA_n be the class of DFA with at most n states

DFA_n cannot be identified by a polynomial number of membership, weak equivalence and inclusion *queries*.

- $L_\cap=\emptyset$
- $L_i=\{w_i\}$ where w_i is i written in base 2.

3.2 What about equivalence queries?

- *Negative results for Equivalence Queries, D. Angluin, Machine Learning, 5, 121-150, 1990*

- Equivalence queries measure also the number of implicit prediction errors a learning algorithm might make

The halving algorithm (1)

- Suppose language consists of any subset of set of n strings $\{w_1,\dots,w_n\}$
- There are 2^n possible languages
- After k queries, candidate languages belong to set C_k
- w_i is a consensus string if it belongs to a majority of languages in C_k

The halving algorithm (2)

- Then consider consensus language H_k={w_i: w_i is a consensus string for C_k}.
- Submit EQ(H_k)
- A counterexample is therefore not a consensus string so less than half the languages in C will agree with the counterexample. Therefore $|C_{k+1}| \leq \frac{1}{2}|C_k|$
- Hence in n steps convergence is ensured!

Why is this relevant?

- The halving algorithm can work when $|C_0|=2^{p(n)}$
- This is always the case (because grammars are written with $p(n)$ characters
- That is why it is important that the equivalence queries are proper
- Proper: EQ(L) with L in C_0

3.3 Learning from equivalence queries alone

Theorem (Angluin 88)

DFA cannot be identified by a polynomial number of strong equivalence *queries*.

(Polynomial in the size of the target)

Proof (*approximate fingerprints*)

- $\forall n$, let $\mathcal{H}_n$ be a class of [size n] *DFA*
- $\forall A$ (perhaps not in $\mathcal{H}_n$), and $\forall q()$, and $\forall n$ sufficiently large $\exists w_A$: $|w_A| < q(n)$ such that

$|\{F \in \mathcal{H}_n : w_A \in L_A \Leftrightarrow w_A \in L_F\}| < |\mathcal{H}_n| / q(n)$

- Answering no to *A*? and returning w_A as a counter example only eliminates a sub-polynomial fraction of *DFA*...

4 Algorithm L*

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

4.1 The Minimal Adequate Teacher

- Learner is allowed:
  - strong equivalence queries
  - membership queries

General idea of *L**

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

4.2 An observation table

	λ	a
λ	1	0
a	0	0
b	1	0
aa	0	0
ab	1	0

The experiments (*EXP*)

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

The states (*RED*)

The transitions (*BLUE*)

Meaning

	λ	a
λ	1	0
a	0	0
b	1	0
aa	0	0
ab	1	0

$$\delta(q_0, \lambda.\lambda) \in F$$
$$\Leftrightarrow$$
$$\lambda \in L$$

Must have identical label (redundancy)

	λ	a
λ	1	0
a	0	0
b	1	0
aa	0	0
ab	1	0

$\delta(q_0,\ ab.a) \notin F$

$\Leftrightarrow$

$aba \notin L$

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

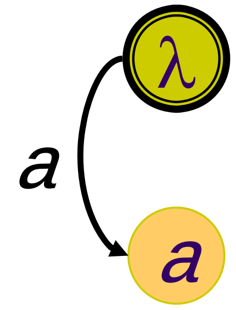
$$\delta(q_0,\lambda)= \delta(q_0,ab)$$

Equivalent prefixes are states

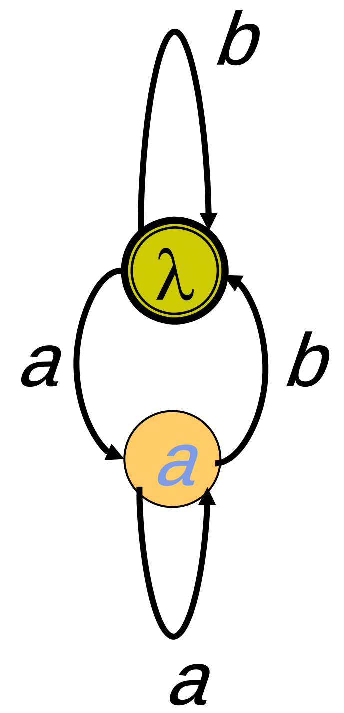
	λ	a
λ	1	0
a	0	0
b	1	0
aa	0	0
ab	1	0

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



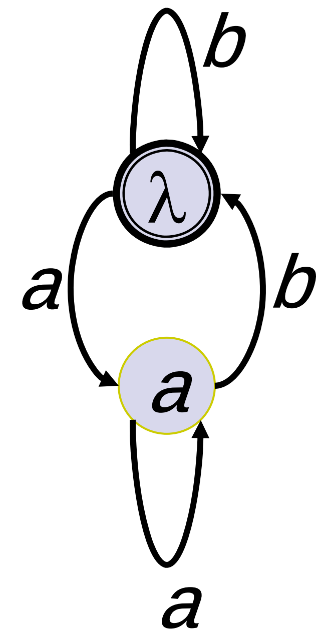
Some rules

This set is suffix-closed

	λ	a
λ	1	0
a	0	0
b	1	0
aa	0	0
ab	1	0

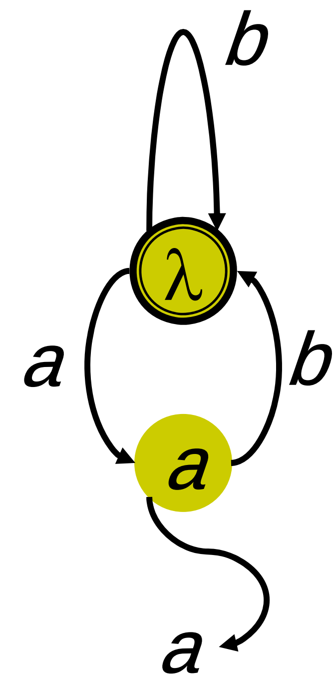
This set is prefix-closed

$RED\Sigma \setminus RED = BLUE$



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

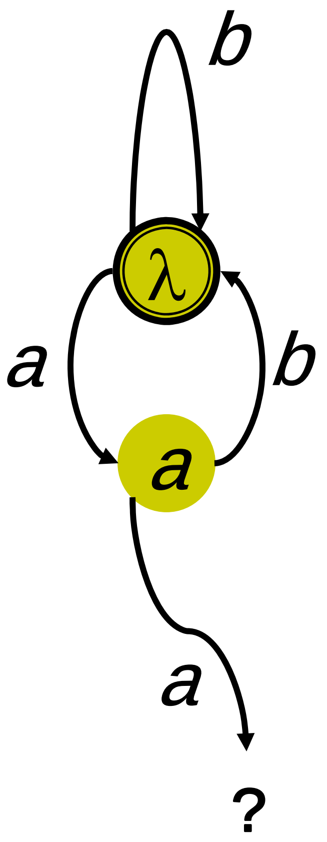
closed if any row of *BLUE* corresponds to some row in *RED*

	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 $\forall a \in \Sigma$, add 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* $\cup$ *BLUE* after appending any symbol

$$OT[s_1] = OT[s_2]$$

$$\Rightarrow$$

$$\forall a \in \Sigma,\ OT[s_1 a] = OT[s_2 a]$$

What do we do when we have an inconsistent table?

Let $a \in \Sigma$ be such that $OT[s_1]$=row(s_2] but $OT[s_1 a] \neq OT[s_2 a]$

- If $OT[s_1 a] \neq OT[s_2 a]$, 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*

4.3 Run of the algorithm

	λ
λ	1
a	1
b	1

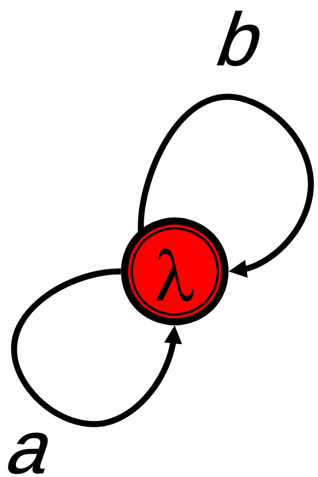
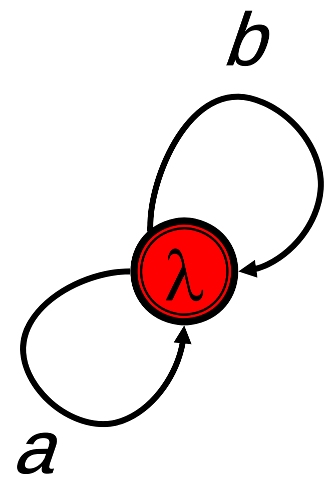


Table is now closed and consistent

An equivalence query is made!



Counter example ***baa*** is returned

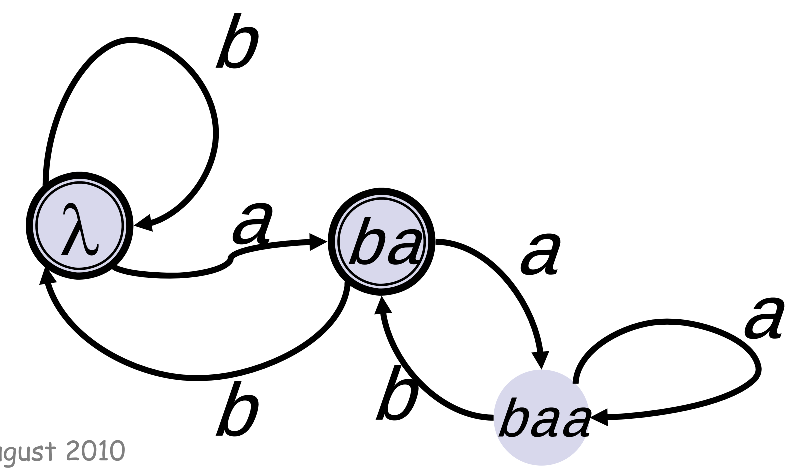
	λ
λ	1
b	1
ba	1
baa	0
a	1
bb	1
bab	1
baaa	0
baab	1

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



The algorithm

while not *A* do

while OT is not complete, consistent or closed do

if OT is not complete then make MQ

if OT is not consistent then add experiment

if OT is not closed then promote

$A \leftarrow$ EQ(OT)

4.4 Proof of the algorithm

Sketch only

Understanding the proof is important for further algorithms

Termination / Correctness

- For every regular language there is a unique minimal *DFA* that recognizes it
- Given a closed and consistent table, one can generate a consistent *DFA*
- A *DFA* consistent with a table has at least as many states as different rows in *H*
- If the algorithm has built a table with *n* different rows in *H*, then it is the target

Finiteness

- Each closure failure adds one different row to *RED*
- Each inconsistency failure adds one experiment, which also creates a new row in *RED*
- Each counterexample adds one different row to *RED*

Polynomial

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

Conclusion

- 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?

Discussion

- Are membership and equivalence queries realistic?
- Membership queries are plausible in a number of applications
- Equivalence queries are not
- A way around this it to do sampling

Good idea

- If we sample following $\mathcal{D}$ 100 strings, and we coincide in labelling with the Oracle, then how bad are we?
- Formula: suppose the error is more than e, then coinciding 100 times has probability at least $(1-\varepsilon)^{100}$. The chance this happens is less than 0,6% for ε =5%

About *PAC* learning and equivalence queries

To be convinced that equivalence queries can exist replace them by the following test:

- draw *m* random examples $x_1,\ldots x_m$
- if $\forall i\ \ell_T(x_i)=\ell_H(x_i)$ then the error is most likely small...

How small?

- Let us suppose that the true error is more than ε
- Then
  - the probability of selecting randomly one example where *T* and *H* coincide is at most $1-\varepsilon$
  - the probability of selecting randomly *m* examples where *T* and *H* coincide (all the time) is at most $(1-\varepsilon)^m$

And

- $(1-\varepsilon)^m \leq e^{-\varepsilon m}$
- So by making this δ we have:

$$\delta \geq e^{-\varepsilon m}$$

$$\Leftrightarrow \quad \log \delta \geq -\varepsilon m$$

$$\Leftrightarrow \quad \log(1/\delta) \leq \varepsilon m$$

$$\Leftrightarrow \quad m \geq 1/\varepsilon \log(1/\delta)$$

Conclusion

- If we can draw according to the true distribution, one can learn a approximately correct DFA from membership queries only.

5 Implementation issues

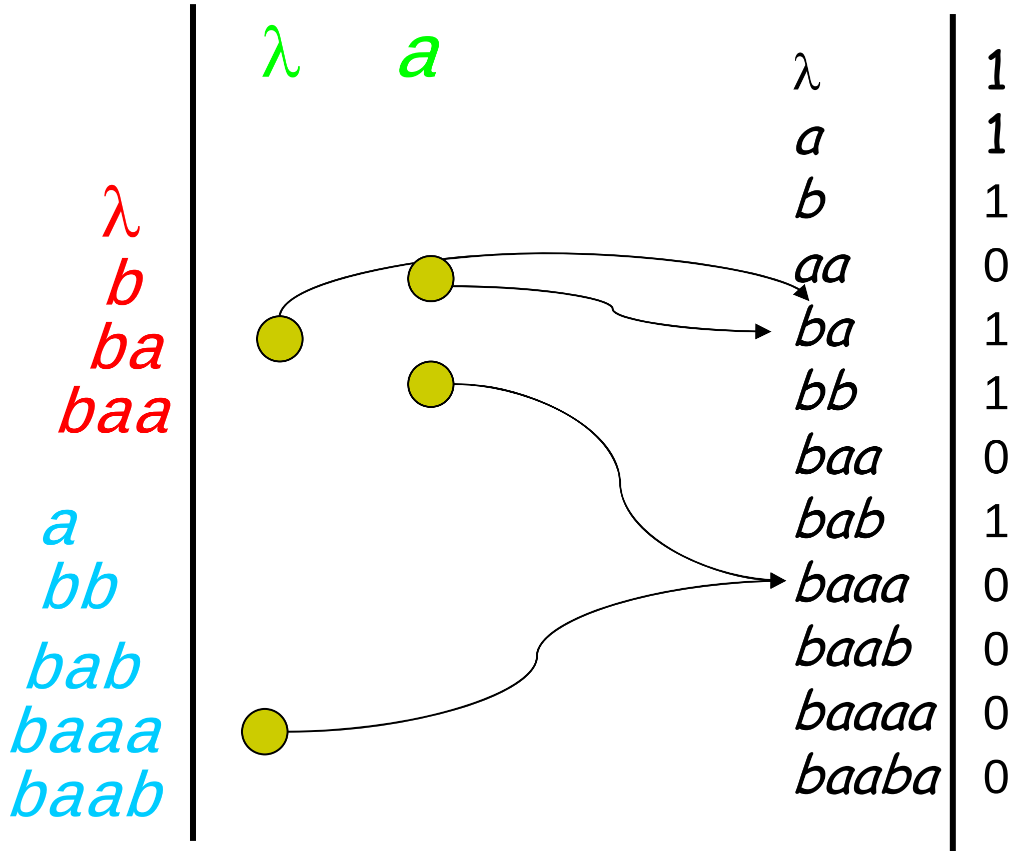
How to implement the table
About Zulu

	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

Instead of memorising a table of 0s and 1s, data structure should be a table of «knowledge»

Use pointers

	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



Zulu competition

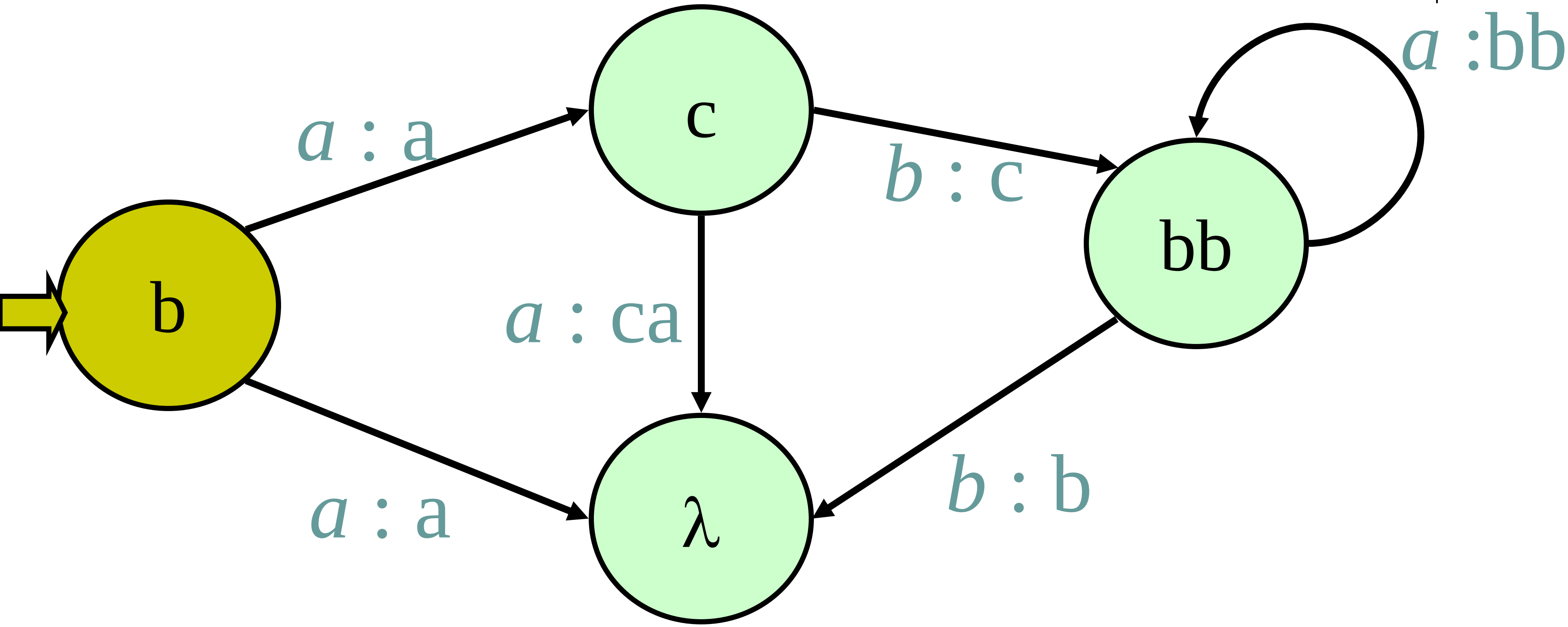
- http://labh-curien.univ-st-etienne.fr/zulu
- 23 competing algorithms, 11 players
- End of the competition a week ago
- Tasks:
- Learn a DFA, be as precise as possible, with *n* queries

Results

Task	queries	alphabet	Best%	states	Task	queries	alphabet	states	Best %
1	304	3	100,00	8	13	725	15	10	100,00
2	199	3	100,00	16	14	1365	15	17	100,00
3	1197	3	96,50	81	15	5266	15	60	100,00
4	1384	3	93,22	100	16	7570	15	71	100,00
5	1971	3	85,89	151	17	17034	15	147	100,00
6	3625	3	100,00	176	18	16914	15	143	87,94
7	429	5	100,00	15	19	1970	5	93	81,67
8	375	5	100,00	18	20	1329	5	61	70,00
9	2524	5	96,44	84	21	571	5	40	69,22
10	3021	5	100,00	90	22	735	5	57	65,11
11	5428	5	99,94	153	23	483	5	73	86,61
12	4616	5	100,00	123	24	632	5	78	100,00

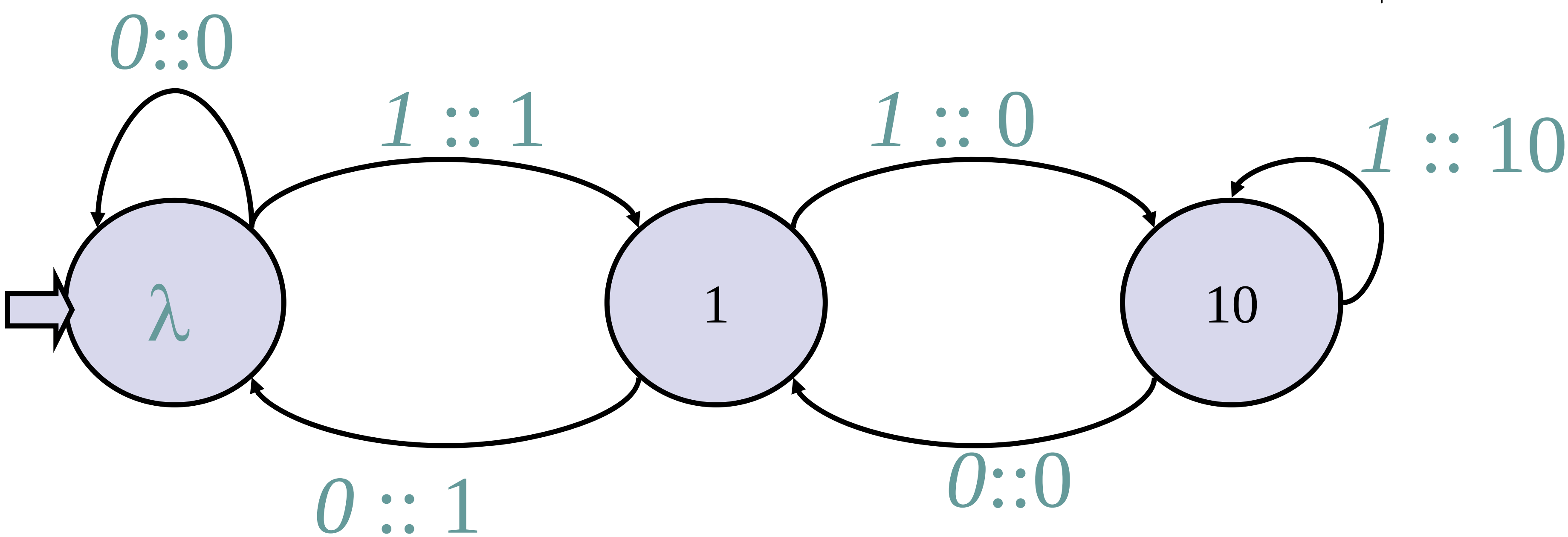
6 Further challenges

Transducer learning



$abaab \rightarrow \mathrm{acbbbbb}$

Transducer learning



Multiplication by 3

Typical queries

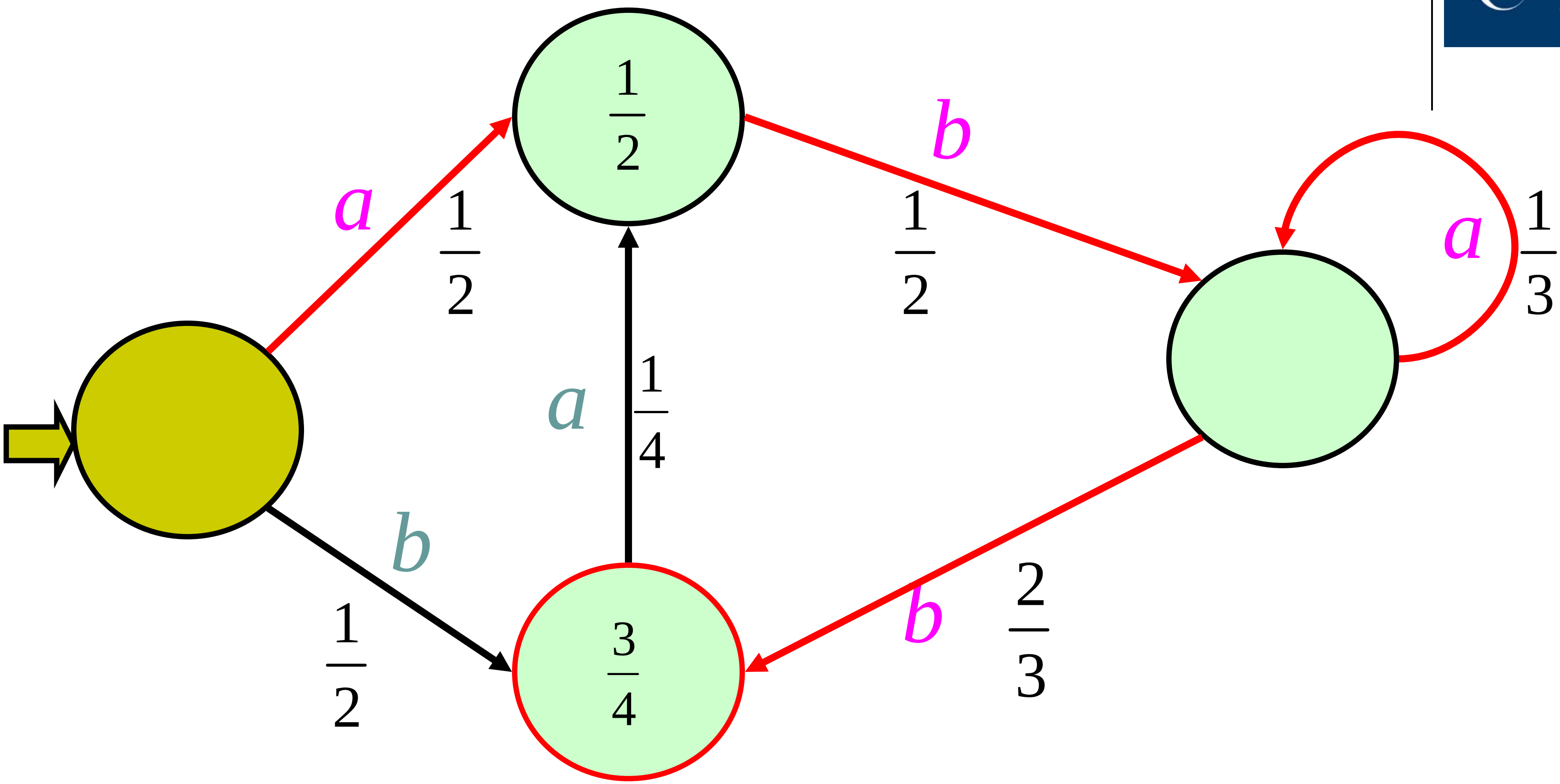
- Translation queries
- Tr(*w*)? Oracle answers with the translation of *w*

- *Vilar, J.M.: Query learning of subsequential transducers. In Miclet, L., de la Higuera, C., eds.: Proceedings of ICGI '96. Number 1147 in LNAI, Springer-Verlag (1996) 72–83*

PFA learning

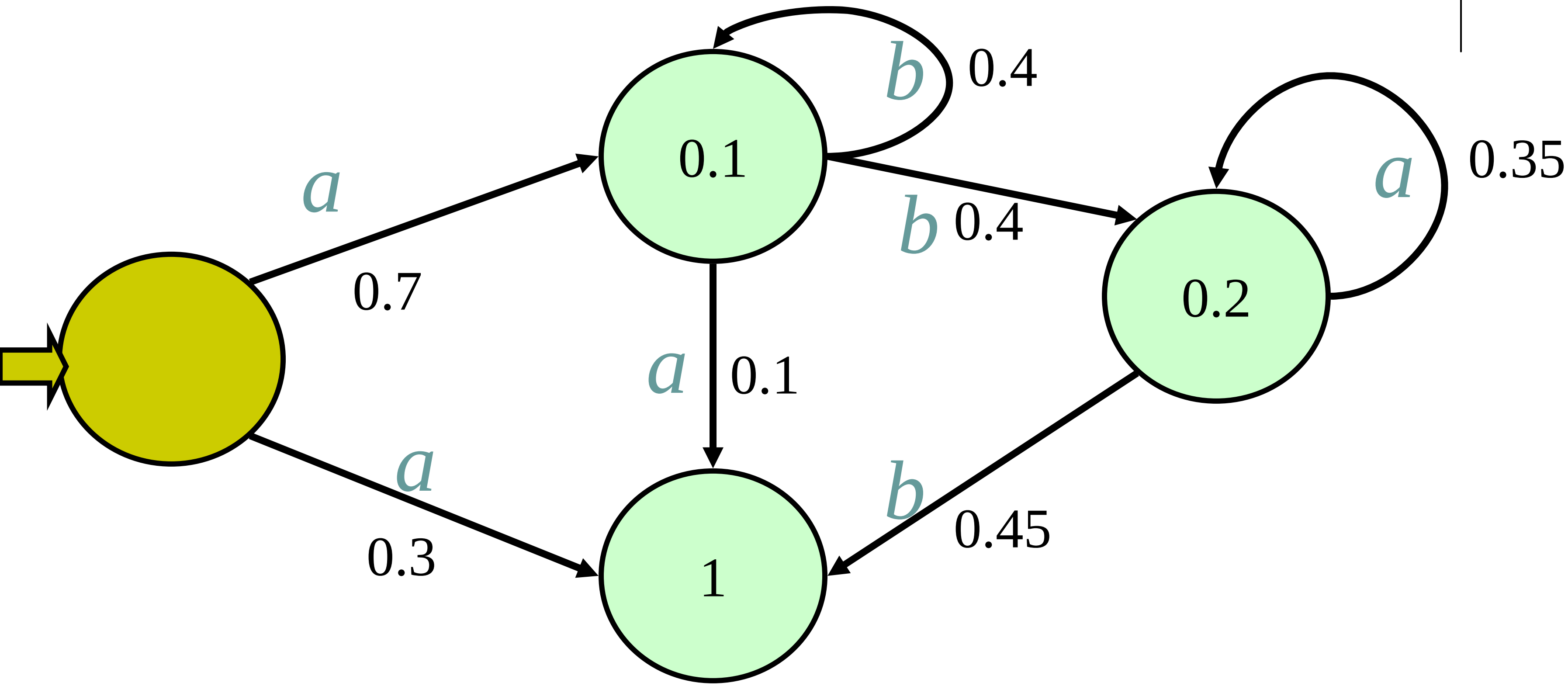
- Probabilistic finite automata can be
  - Deterministic
  - Non deterministic

A deterministic PFA



$$\Pr_A(abab)=\frac{1}{2}\times\frac{1}{2}\times\frac{1}{3}\times\frac{2}{3}\times\frac{3}{4}=\frac{1}{24}$$

A nondeterministic PFA



Pr(*aba*) = 0.7*0.4*0.1*1 +0.7*0.4*0.45*0.2
= 0.028+0.0252=0.0532

What queries should we consider?

- Probability queries
  - PQ(w)? Oracle returns $Pr_{\mathcal{D}}(w)$
- EX()? Oracle returns a string w randomly drawn according to $\mathcal{D}$
- Specific sampling query SSQ(L)
  - Oracle returns a string belonging to L sampled according to $\mathcal{D}$
  - Distribution is $Pr_{\mathcal{D}\,L}(w)=Pr_{\mathcal{D}}(w)/Pr_{\mathcal{D}}(L)$

Context-free grammar learning

- Typical query corresponds to using the grammar (structural query)

- In which case the goal is to identify the grammar, not the language !

7 General conclusion

Some open problems

- Find better definitions
- Do these definitions give a general framework for grammatical inference?
- How can we use resource bounded queries?
- Use Zulu or develop new tools for Zulu