Learning languages from bounded resources: the case of the DFA and the balls of strings

ICGI -Saint Malo

September 2008

de la Higuera, Janodet and Tantini

The authors

Colin de la Higuera

Jean Christophe Janodet

Frédéric Tantini

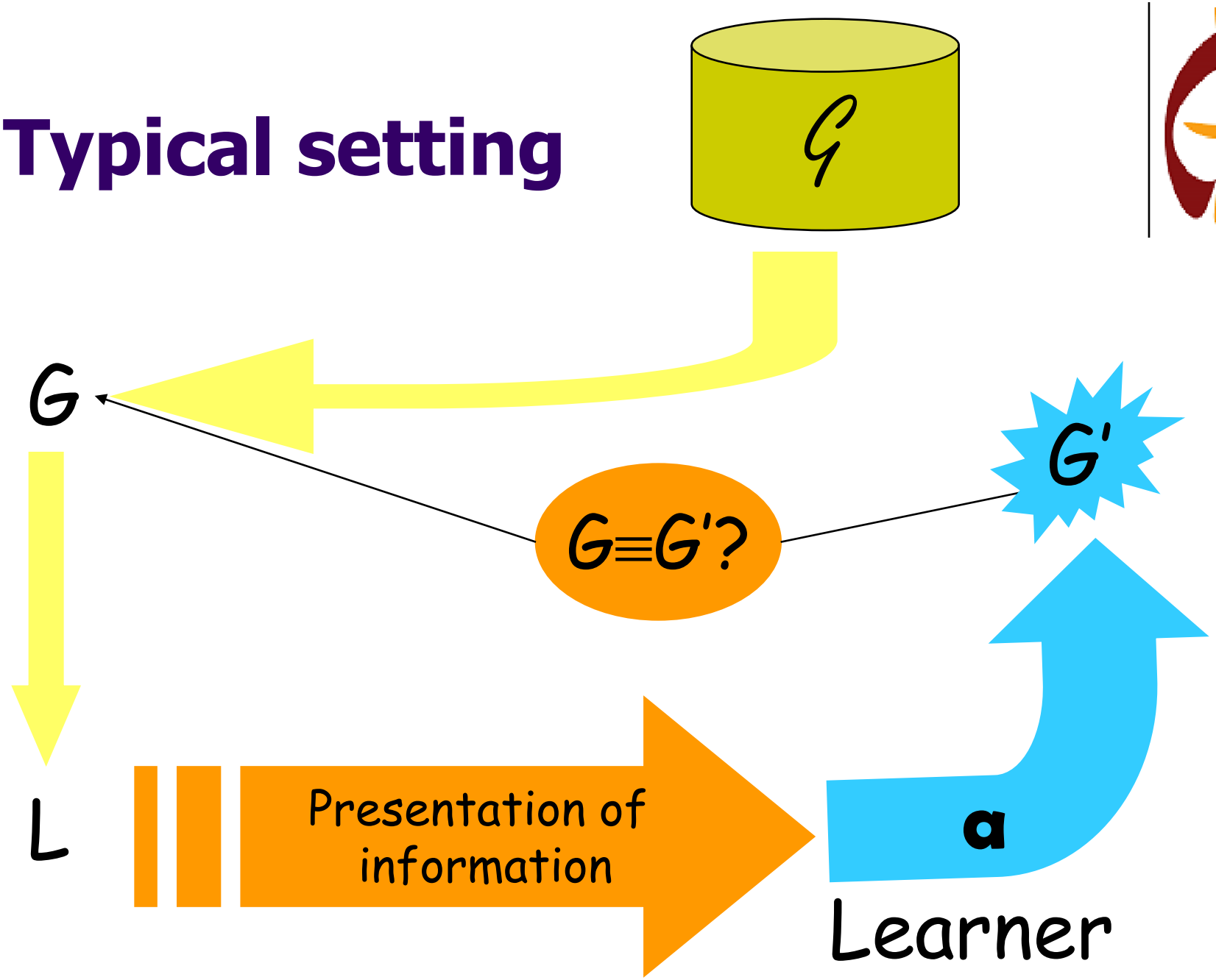
Outline

1. What and why
2. Balls and automata
3. General results
4. Selected result 1: it can be easier to learn without counter-examples
5. Selected result 2: PAC-learning balls
6. Selected result 3: can we get a result in combinatorics?
7. Conclusion

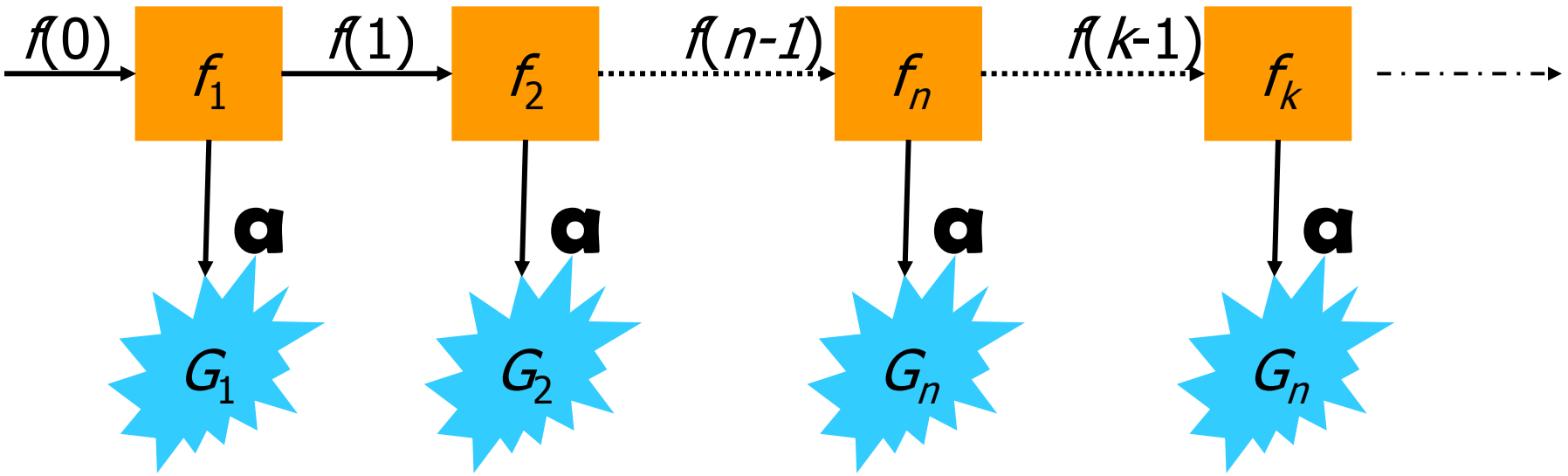
1 The problem

- Grammatical inference is about learning grammars given information about the language
- Pragmatic point of view (do something!)
- Theoretical point of view (study the convergence)

Typical setting



Online process



Summarising theory

A. Describe things with respect to finding a target
B. An idealised situation but allows to study:
    A. When we fail, that means that learning is hard
    B. We can compute how many resources are needed to identify (time, examples)
    C. One can also talk about approximate learning
C. Obviously, some targets are harder than others!

Many results

- Angluin 1980 & 1981
- Angluin and Smith 1983
- Pitt 1989
- cdlh 1997
- Becerra-Bonache et al. 2008
- …

2 The classes under scrutiny

- Balls for the edit distance
- Deterministic finite automata (DFA)

Balls for the edit distance

- Edit distance measures the number of edit operations to get from one string to another.

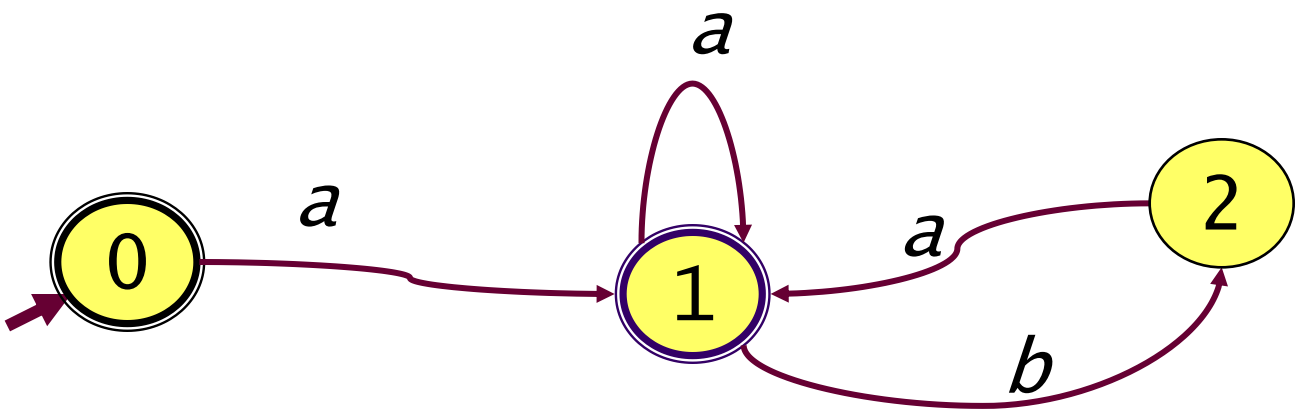
- here, each operation has cost 1
- a ball with centre x and radius k is a $B_k(x) = \{ w \in \Sigma^* : d_e(x,w) \leq k \}$

B_2(aba)

- {a, b, aa, ab, ba, bb, aaa aab aba abb, baa, bab, bba, bbb, aaaa, aaab, aaba, aabb, abaa, abab, abba, abbb, baaa, baba, babb, bbaa, bbab, bbba, aaaba, aabaa, abaaa, ababa, aabba, aabab, abaab , baaba, babaa, abbaa, bbaba, babba, babab, abbba, abbab, ababb}

DFA

- *Parsing* in $\mathcal{O}(n)$
- Equivalence in $\mathcal{O}(n \log n)$



Classes, for an alphabet Σ

- $GB(\Sigma)$ is the set of all good balls over Σ: $B_k(x)$ such that $|x| \geq k$
- $DFA(\Sigma)$ is the set of DFA over Σ

Sizes

- $||B_k(x)||$ is $|x|$ (+log k)
- $||A||$ is n, number of states

3 Results in the paper

Criteria (poly)	GB	DFA	BALL
PAC-informant	No	No	No
PAC-text	No	No	No
IPE-informant	No	No	No
IPE-text	Yes	No	No
MC-informant	Yes	Yes	Yes
MC-text	Yes	No	No
CS-informant	Yes	Yes	No
CS-text	Yes	No	No
MQ	No	No	No
MQ+EQ	No	Yes	No
CQ_{edit}	Yes	No	No

New, known, derived

4 Selected result (1)

- In some cases it is *easier* to learn without counter examples than with them!
- If we count errors, the algorithm should wait with a safe hypothesis before making a better guess.
- Unless we pay a penalty for over-generalising!
- But if we are not given any counter-examples, then there is no penalty!

IPE model

- Count errors of prediction.
- The algorithm has to identify in the limit.
- When learning from an informant, have to update the hypothesis to something tight.
- When learning from text, can afford to overgeneralize until having a "certificate".

Criteria	GB	DFA	BALL
PAC-informant	No	No	No
PAC-text	No	No	No
IPE-informant	**No**	No	No
IPE-text	**Yes**	No	No
MC-informant	Yes	Yes	Yes
MC-text	Yes	No	No
CS-informant	Yes	Yes	No
CS-text	Yes	No	No
MQ	No	No	No
MQ+EQ	No	Yes	No
CQ_{edit}	Yes	No	No

5 Selected result (2)

- Learning balls of strings in a PAC setting
- Setting: data arrive depending of a fixed unknown distribution
- Distribution is used to compute a distance between the target and the hypothesis.

Typical technique

- If the problem « find a grammar G in class C consistent with the data » is NP-hard,

  then the class C is not PAC learnable

- Usually, in grammatical inference, this is not very useful since consistency is a trivial problem.

Suppose GB(Σ) is PAC learnable

- Then, we could build a poly-time randomised algorithm which would solve the consistency problem.
- But finding a consistent ball is NP-hard.

Criteria	GB	DFA	BALL
PAC-informant	No	No	No
PAC-text	No	No	No
IPE-informant	No	No	No
IPE-text	Yes	No	No
MC-informant	Yes	Yes	Yes
MC-text	Yes	No	No
CS-informant	Yes	Yes	No
CS-text	Yes	No	No
MQ	No	No	No
MQ+EQ	No	Yes	No
CQ_{edit}	Yes	No	No

6 Selected result (3)

- Why should it be interesting to study both these classes?
- What if each ball of size k could be represented by a DFA of size $p(k)$?

* $p()$ is a fixed polynomial

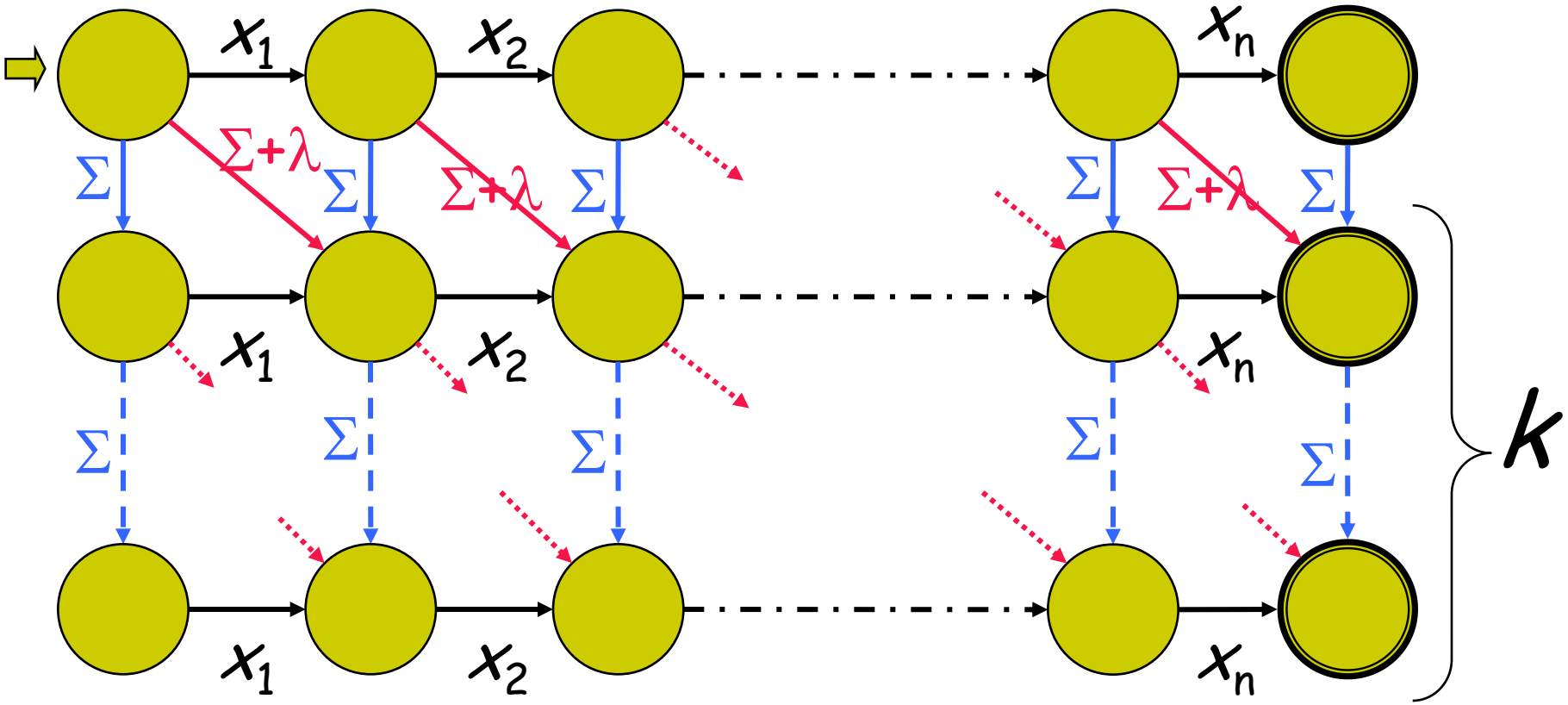
The question of *polynomial determinism for balls*

- Is each ball encodable by a polynomial DFA?
- Is there a polynomial $p()$ such that
  $\forall x \in \Sigma^*, \forall k \geq 0, || Dfa(k,x)|| \leq p(||B_k(x)||)$?
- Denote by $Dfa(k,u)$ the minimal DFA such that $Dfa(k,u)=B_k(u)$.

Answer

- Researchers working on dictionaries or *approximate string matching* believe that the answer is no.
- References: Ukkonen 85, Melichar 96, Navarro 97 (and a number of private communications)

Note: easy for NFA



But we can try…

- If in some paradigm we had

Criteria	GB	DFA	BALL
…	…	…	…
paradigm	No	Yes	-
…	…	…	…

- Then (perhaps) the question of *polynomiality of determinism for balls* could have a negative answer!

Criteria	GB	DFA	BALL
PAC-informant	No	No	No
PAC-text	No	No	No
IPE-informant	No	No	No
IPE-text	Yes	No	No
MC-informant	Yes	Yes	Yes
MC-text	Yes	No	No
CS-informant	Yes	Yes	No
CS-text	Yes	No	No
MQ	No	No	No
MQ+EQ	**No**	**Yes**	No
CQ_{edit}	Yes	No	No

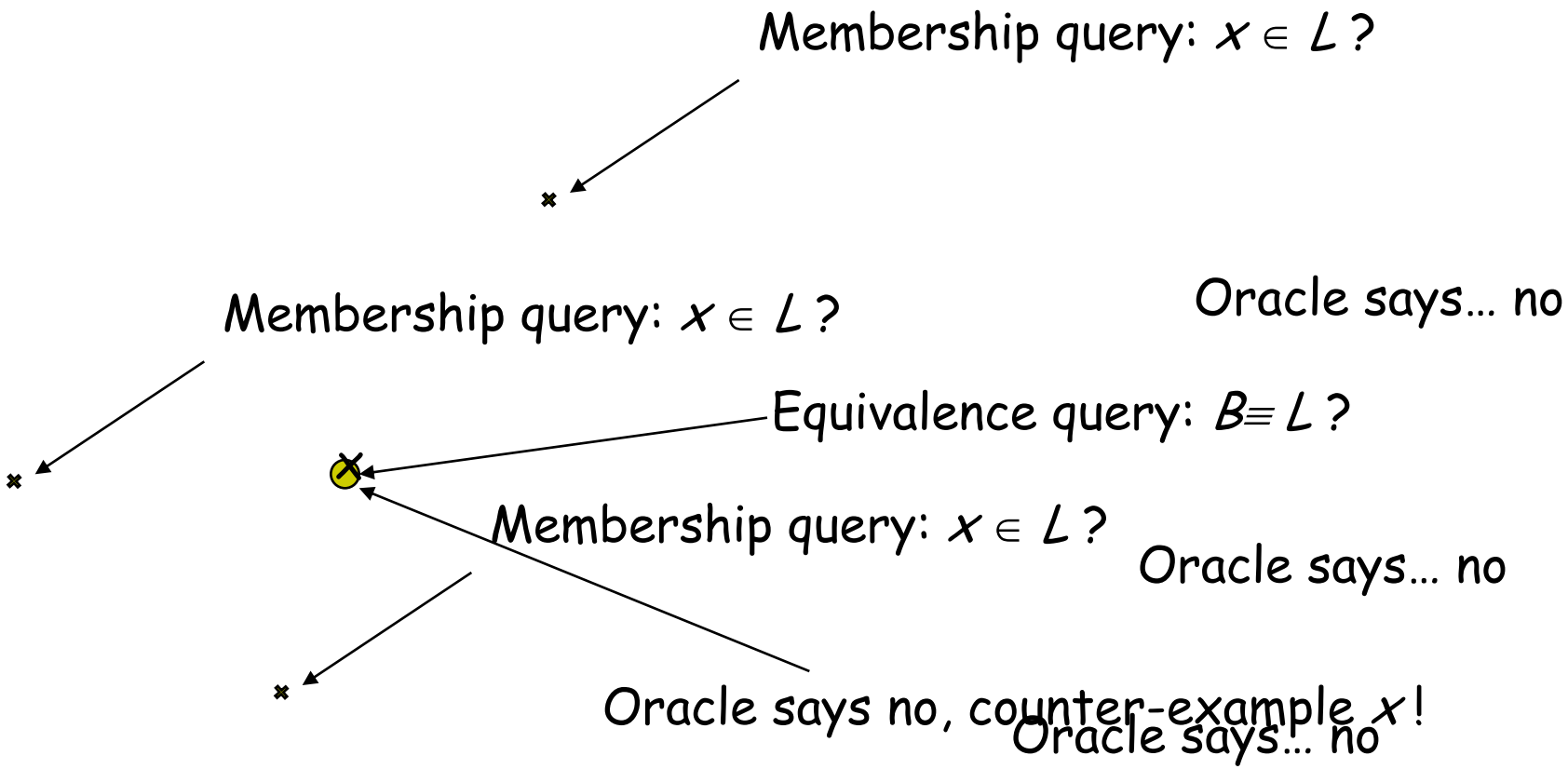
Could we…

- Learn a DFA instead of a ball and at the end of the learning transform the DFA into a ball ?

But… doesn't help

- EQs have to be proper!
- Can only query the Oracle with DFA.
- Yet the empty language if submitted to the Oracle, allows to get one (first) string very cheaply!

Find a ball in space....



So

- No simple reduction class to class
- Reassuring: in learning theory inclusion doesn't preserve learnability!

6 Conclusion

- The question of *polynomial determinism for balls* remains open
- No conjecture, an opinion: false
- No new result "easier to learn in paradigm P than in paradigm Q", but some incompatibility results
- Several new techniques to avoid mind changes or prediction errors

Appendix

- Some definitions

Identification in the limit

Poly-CS

Efficiency issues

Poly-MC

Poly-IPE

Appendix, some technicalities

Size of *f*

Size of *G*

Size of *L*

#MC

Runtimes

#IPE

PAC

#CS

Non probabilistic setting

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

Identification in the limit

- The definitions, presentations
- The alternatives
  - Order free or not
  - Randomised algorithm

The intuition

- Online setting:
- Data arrives one at the time. Learner modifies its previous hypothesis.
- Learner only makes a finite number of modifications
- After the last modification, the result is correct.

A presentation is

a function $f: \mathrm{N} \rightarrow X$ where X is any set,

- *yields*: $\mathcal{P}resentations \rightarrow \mathcal{L}anguages$

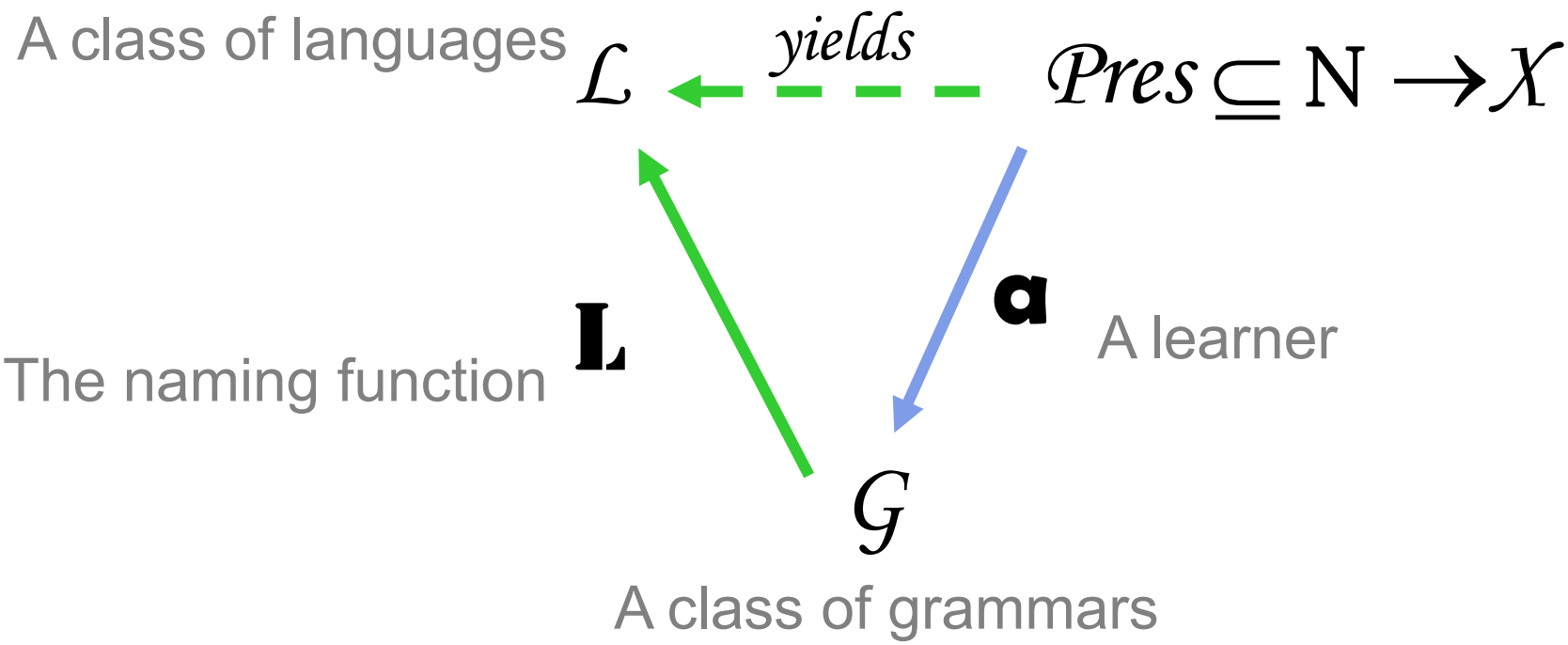
- If $f(\mathrm{N})=g(\mathrm{N})$ then *yields*(f)= *yields*(g)

Learning function

- Given a presentation f, f_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 $f_n = \{f(0),\ldots,f(n-1)\}$ and returns a grammar.
- Given a grammar G, $\mathbf{L}(G)$ is the language generated/recognised/ represented by G.

Identification in the limit

$$f(\mathbb{N})=g(\mathbb{N}) \Rightarrow yields(f)=yields(g)$$



$$\exists n \in \mathbb{N} : k > n \Rightarrow \mathbf{L}(\mathbf{a}(f_k)) = yields(f)$$

What about efficiency?

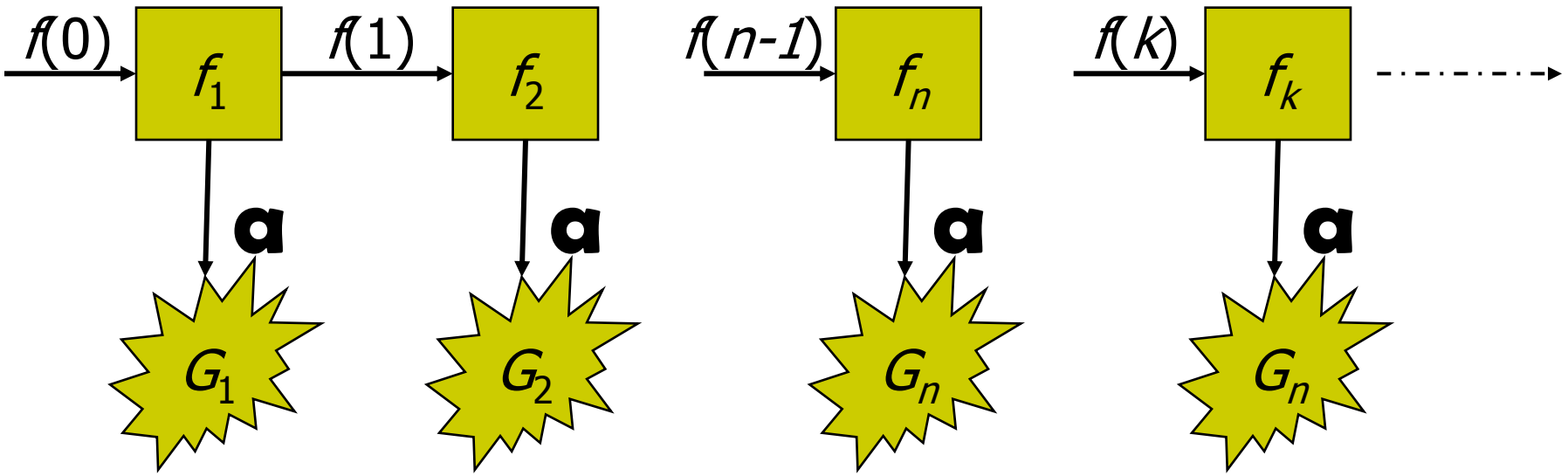
- We can try to bound
  - global time
  - update time
  - errors before converging
  - mind changes
  - queries
  - good examples needed

What should we try to measure?

- The size of G?
- The size of L?
- The size of f?
- The size of f_n?

Some candidates for polynomial learning

- Total runtime polynomial in $\|L\|$
- Update runtime polynomial in $\|L\|$
- #MC polynomial in $\|L\|$
- #IPE polynomial in $\|L\|$
- Size of CS polynomial in $\|L\|$


f(0)
f1
f(1)
f2
f(n-1)
fn
f(k)
fk
a
a
a
a
G1
G2
Gn
Gn

Some selected results (1)

DFA	text	informant
Runtime	no	no
Update-time	"	yes
#IPE	"	no
#MC	"	yes
CS	"	yes

Some selected results (2)

CFG	text	informant
Runtime	no	no
Update-time	"	yes
#IPE	"	no
#MC	"	?
CS	"	no

Some selected results (3)

Good Balls	text	informant
Runtime	no	no
Update-time	yes	yes
#IPE	yes	no
#MC	yes	no
CS	yes	yes

4 Probabilistic setting

- Using the distribution to measure error
- Identifying the distribution
- Approximating the distribution

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

Probably approximately correct

PAC learning
(Valiant 84, Pitt 89)

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

Polynomially PAC learnable

- There is an algorithm that samples reasonably and returns with probability at least 1-δ a grammar that will make at most ε errors.

Results

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

Learning distributions

- No error
- Small error

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.

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.

Appendix, some technicalities

Size of *f*

Size of *G*

Size of *L*

#MC

Runtimes

#IPE

PAC

#CS

The size of L

- If no grammar system is given, meaningless
- If $\mathcal{G}$ is the class of grammars then $\|L\| = \min\{\|G\| : G \in \mathcal{G} \wedge \mathbf{L}(G)=L\}$
- Example: the size of a regular language when considering DFA is the number of states of the minimal DFA that recognizes it.

Is a grammar representation reasonable?

- Difficult question: typical arguments are that NFA are better than DFA because you can encode more languages with less bits.
- Yet redundancy is necessary!

Proposal

- A grammar class is reasonable if it encodes sufficient different languages.
- *Ie* with n bits you have 2^{n+1} candidate encodings so optimally you could have 2^{n+1} different languages.
- Allow for redundancy and syntaxic sugar, so $\Omega(2^{n+1})$ different languages.

But

- We should allow for redundancy and for some strings that do not encode grammars.
- Therefore a grammar representation is reasonable if there exists a polynomial $p()$ and for any n the number of different languages encoded by grammars of size n is in $\Omega(2^n)$.

The size of a presentation *f*

- Meaningless. Or at least no convincing definition comes up.
- But when associated with a learner **a** we can define the *convergence point* *Cp*(*f*,**a**) which is the point at which the learner **a** finds a grammar for the correct language *L* and does not change its mind.
- $Cp(f,\mathbf{a}) = n : \forall m \geq n,\ \mathbf{a}(f_m) = \mathbf{a}(f_n) \equiv L$

The size of a finite presentation f_n

- An easy attempt is n
- But then this does not represent the quantity of information we have received to learn.
- A better measure is $\sum_{i \leq n} |f(i)|$

Quantities associated with learner α

- The update runtime: time needed to update hypothesis h_{n-1} into h_n when presented with $f(n)$.
- The complete runtime: time needed to build hypothesis h_n from f_n. Also the sum of all update-runtimes.

Definition 1 (total time)

$\mathcal{G}$ is polynomially identifiable in the limit from $\mathcal{Pres}$ if there exists an identification algorithm $\mathbf{a}$ and a polynomial $p()$ such that given any G in $\mathcal{G}$, and given any presentation f such that $yields(f)=\mathbf{L}(G)$, $Cp(f,\mathbf{a}) \leq p(\|G\|)$.

(or $global\text{-}runtime(\mathbf{a}) \leq p(\|G\|)$)

Impossible

- Just take some presentation that stays useless until the bound is reached and then starts helping.

Definition 2 (update polynomial time)

$\mathcal{G}$ is polynomially identifiable in the limit from $\mathcal{Pres}$ if there exists an identification algorithm $\mathbf{a}$ and a polynomial $p()$ such that given any G in $\mathcal{G}$, and given any presentation f such that $yields(f)=\mathbf{L}(G)$, $update\text{-}runtime(\mathbf{a}) \leq p(\|G\|)$.

Doesn't work

- We can just differ identification
- Here we are measuring the time it takes to build the next hypothesis.

Definition 4: polynomial number of mind changes

$\mathcal{G}$ is polynomially identifiable in the limit from $\mathcal{Pres}$ if there exists an identification algorithm $\mathbf{a}$ and a polynomial $p()$ such that given any G in $\mathcal{G}$, and given any presentation f such that $yields(f)=\mathbf{L}(G)$,

$$\#\{i : \mathbf{a}(f_i) \neq \mathbf{a}(f_{i+1})\} \leq p(\|G\|).$$

Definition 5: polynomial number of implicit prediction errors (IPE)

- Denote by $G \blacklozenge x$ if G is incorrect with respect to an element x of the presentation (*i.e.* the algorithm producing G has made an IPE.

$\mathcal{G}$ is polynomially identifiable in the limit from $\mathcal{P}res$ if there exists an identification algorithm $\mathbf{a}$ and a polynomial $p()$ such that given any G in $\mathcal{G}$, and given any presentation f such that $yields(f)=\mathbf{L}(G)$,

$\#\{i : \mathbf{a}(f_i) \blacklozenge f(i+1)\} \leq p(\|G\|)$.

Definition 6: polynomial characteristic sample (poly-CS)

$\mathcal{G}$ has polynomial characteristic samples for identification algorithm **a** if there exists an and a polynomial $p()$ such that: given any G in $\mathcal{G}$, $\exists Y$ correct sample for G, such that when $Y \subseteq f_n$, $\mathbf{a}(f_n) \equiv G$ and $\| Y \| \leq p(\| G \|)$.

3 Probabilistic setting

- Using the distribution to measure error
- Identifying the distribution
- Approximating the distribution

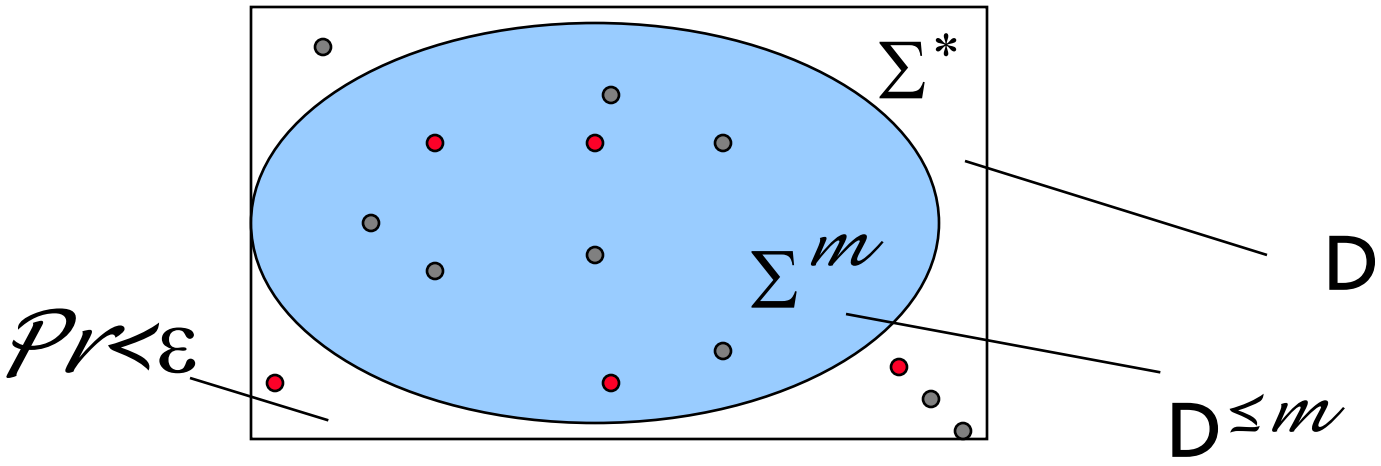
Probabilistic settings

- 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

How do we consider a finite set?



By sampling $1/\varepsilon$ ln $1/\delta$ examples we can find a safe m.

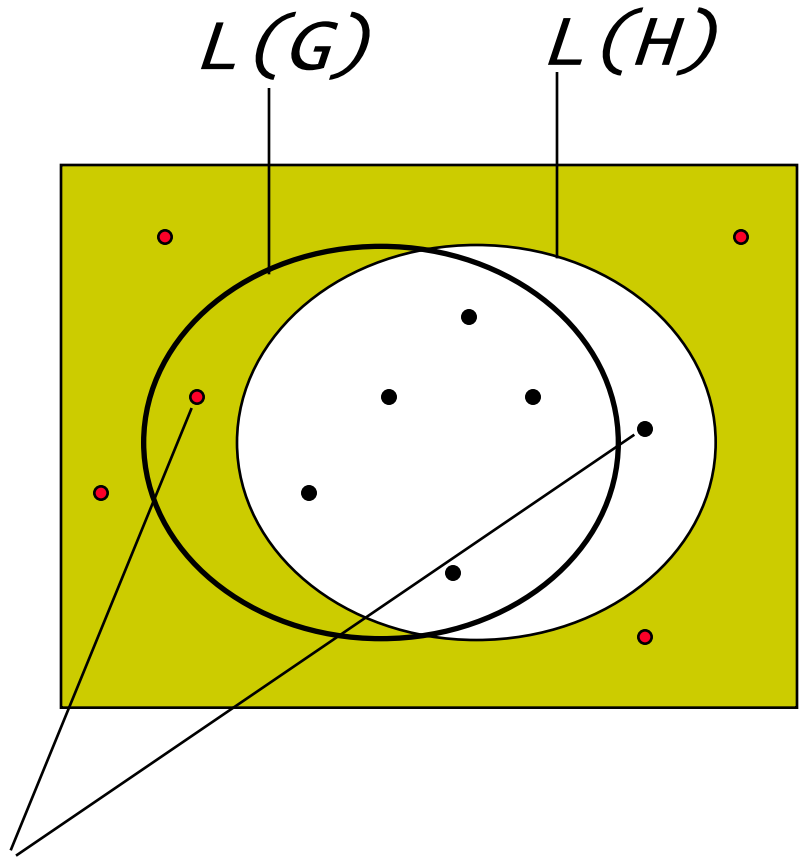
PAC learning
(Valiant 84, Pitt 89)

- L a set of languages
- 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

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



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