Probabilistic Automata

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

Acknowledgements

- Laurent Miclet, Jose Oncina and Tim Oates for 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

1 Introduction: grammatical inference

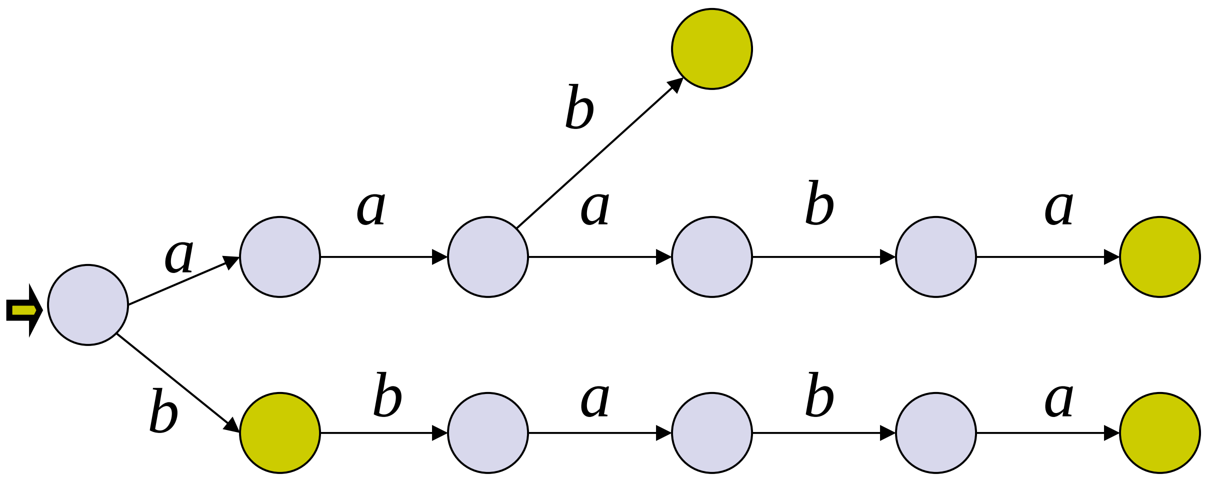
- Find an automaton which explains my data
- Given information about a language, find the good grammar / automaton

Typical case

- $S_+ = \{aab, b, aaaba, bbaba\}$
- $S_- = \{aa, ba, aaa\}$

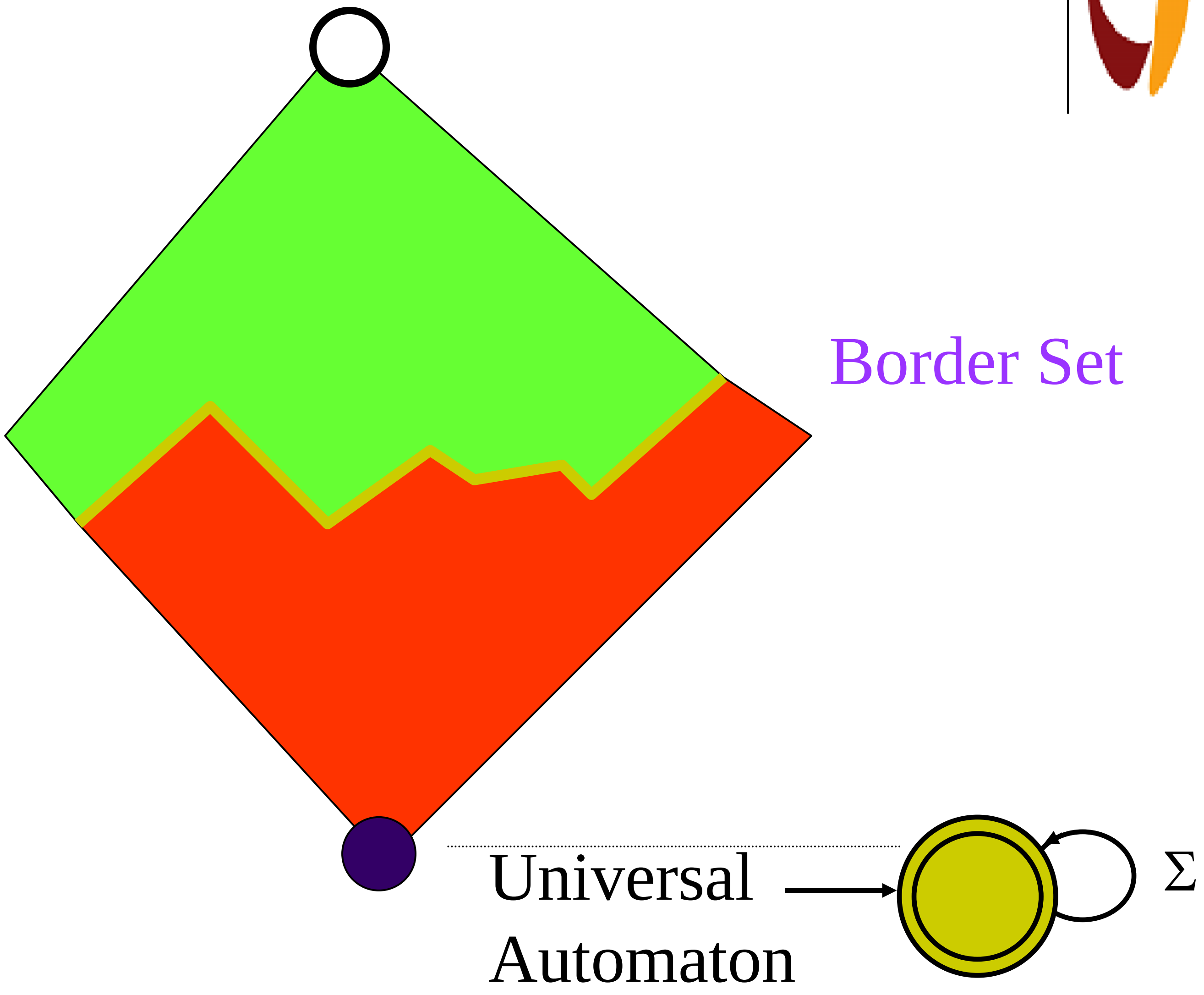
Prefix Tree Acceptor

$S_+ = \{aab, b, aaaba, bbaba\}$



The *PTA* is the smallest *DFA* accepting X_+ where every state has at most one predecessor.

$CA\ (X_+)$ or $PTA(X_+)$

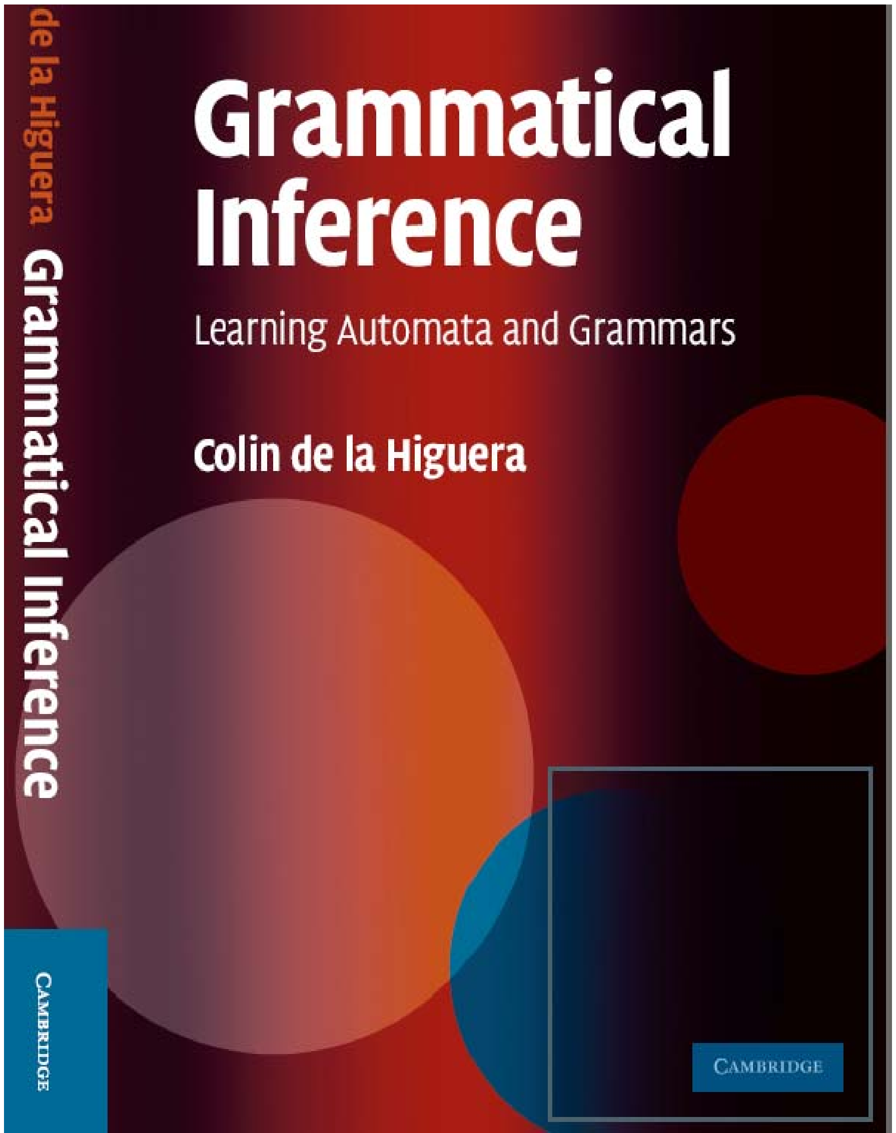


A combinatorial version of the problem

Some ideas that work

- Have both positive and negative examples (learn from an informant)
- Be allowed to ask questions (queries): active learning

If you want to know more…



In practice

(Computational biology, speech recognition, web services, automatic translation, image processing …)

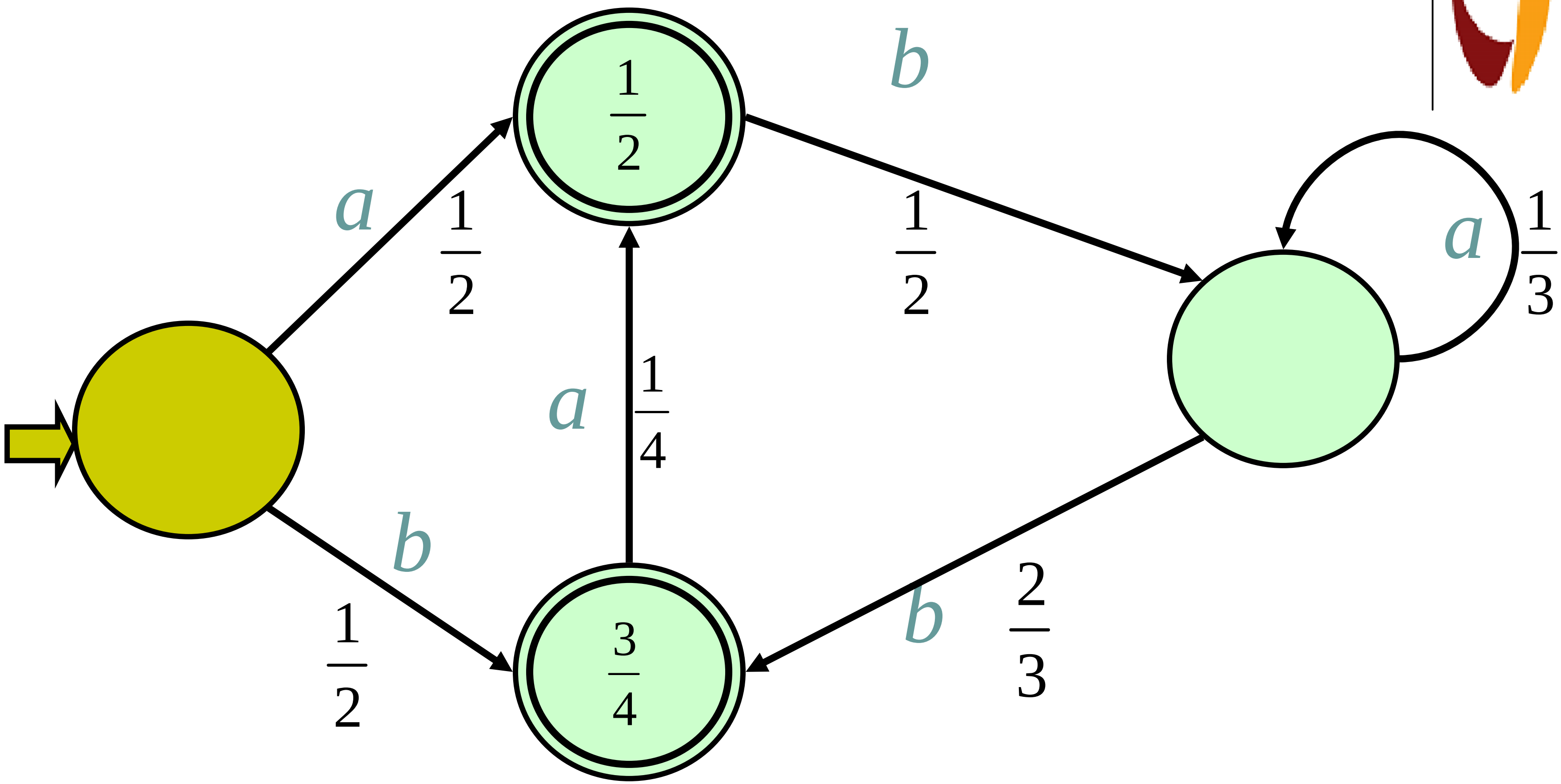
- A lot of positive data
- Not necessarily any negative data
- No ideal target
- Noise

The problem, revisited

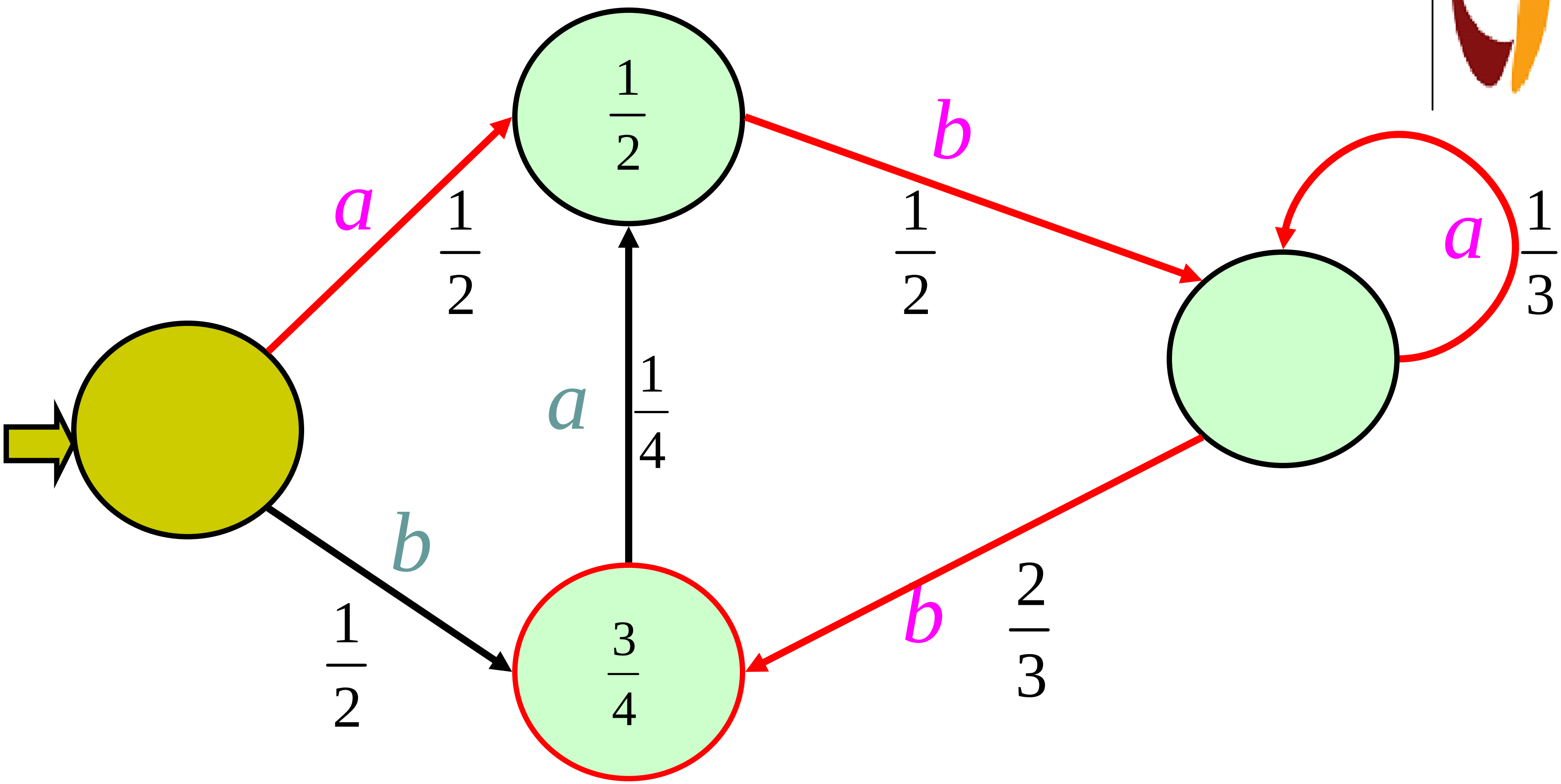
- The data consists of positive strings, «generated» following an unknown distribution
- The goal is now to find (learn) this distribution
- Or the FSM that is used to generate the strings
- Learning is about giving a meaning to finding…

Success of the probabilistic models

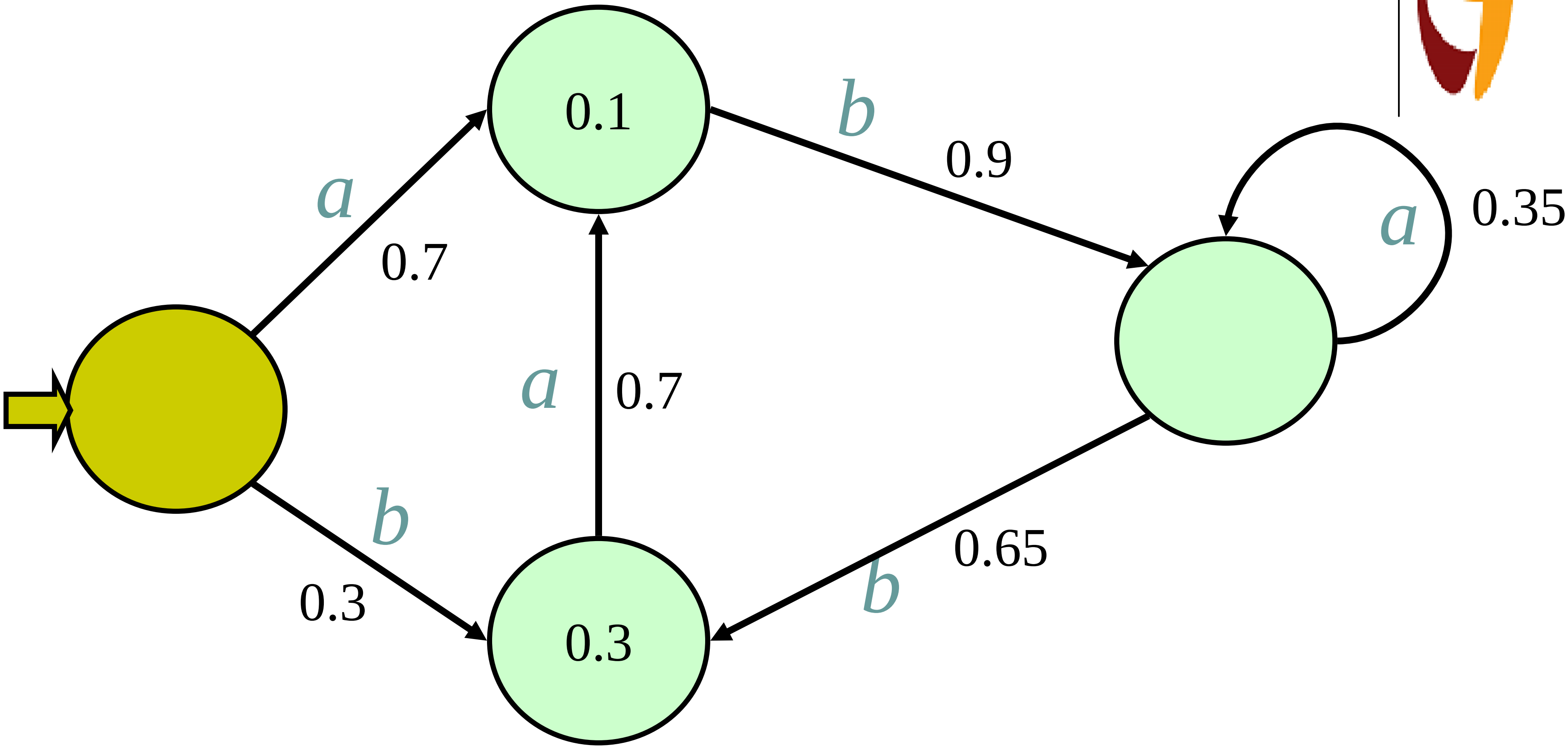
- *n*-grams
- Hidden Markov Models
- Probabilistic grammars



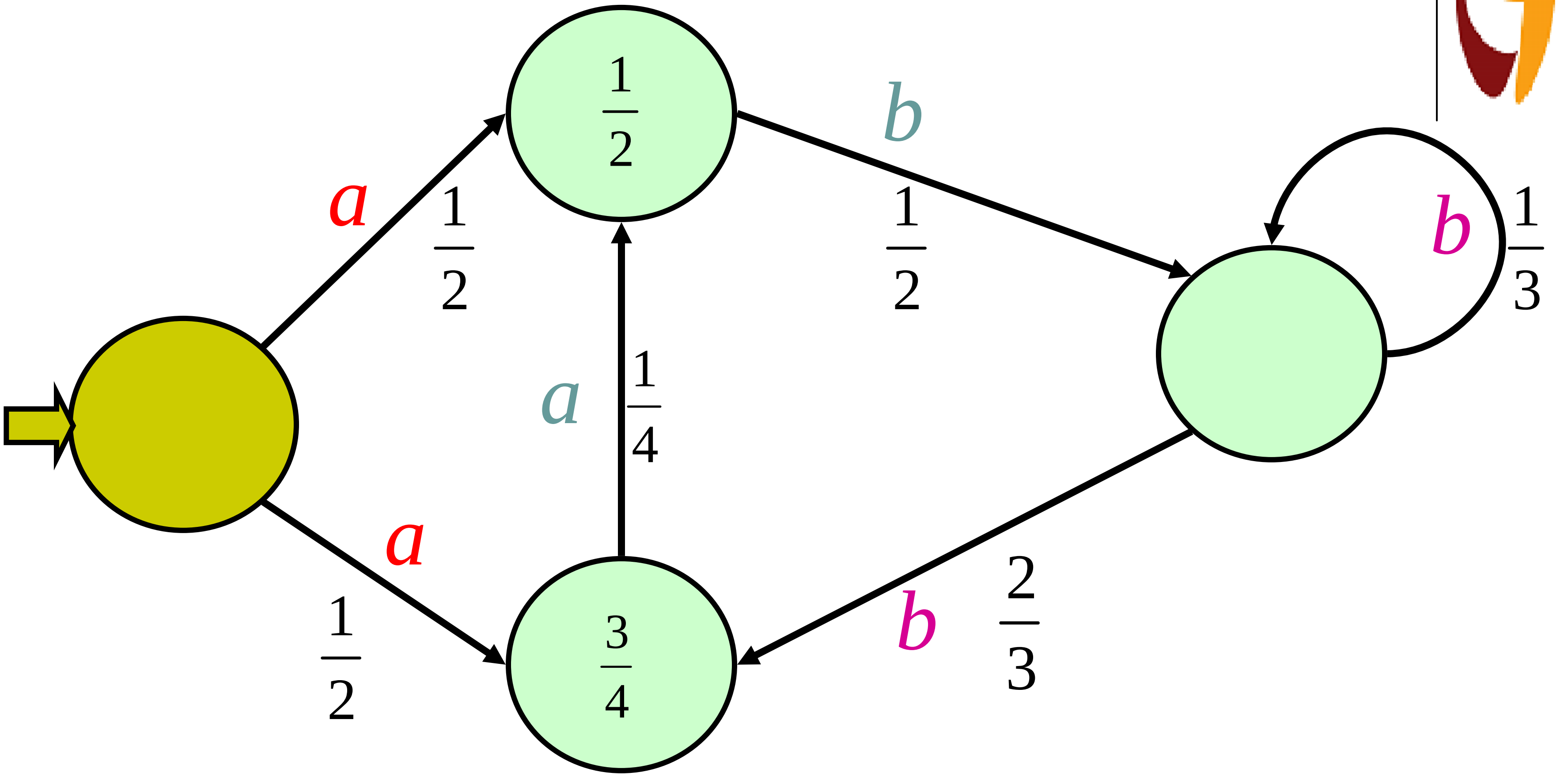
DPFA: Deterministic Probabilistic Finite Automaton



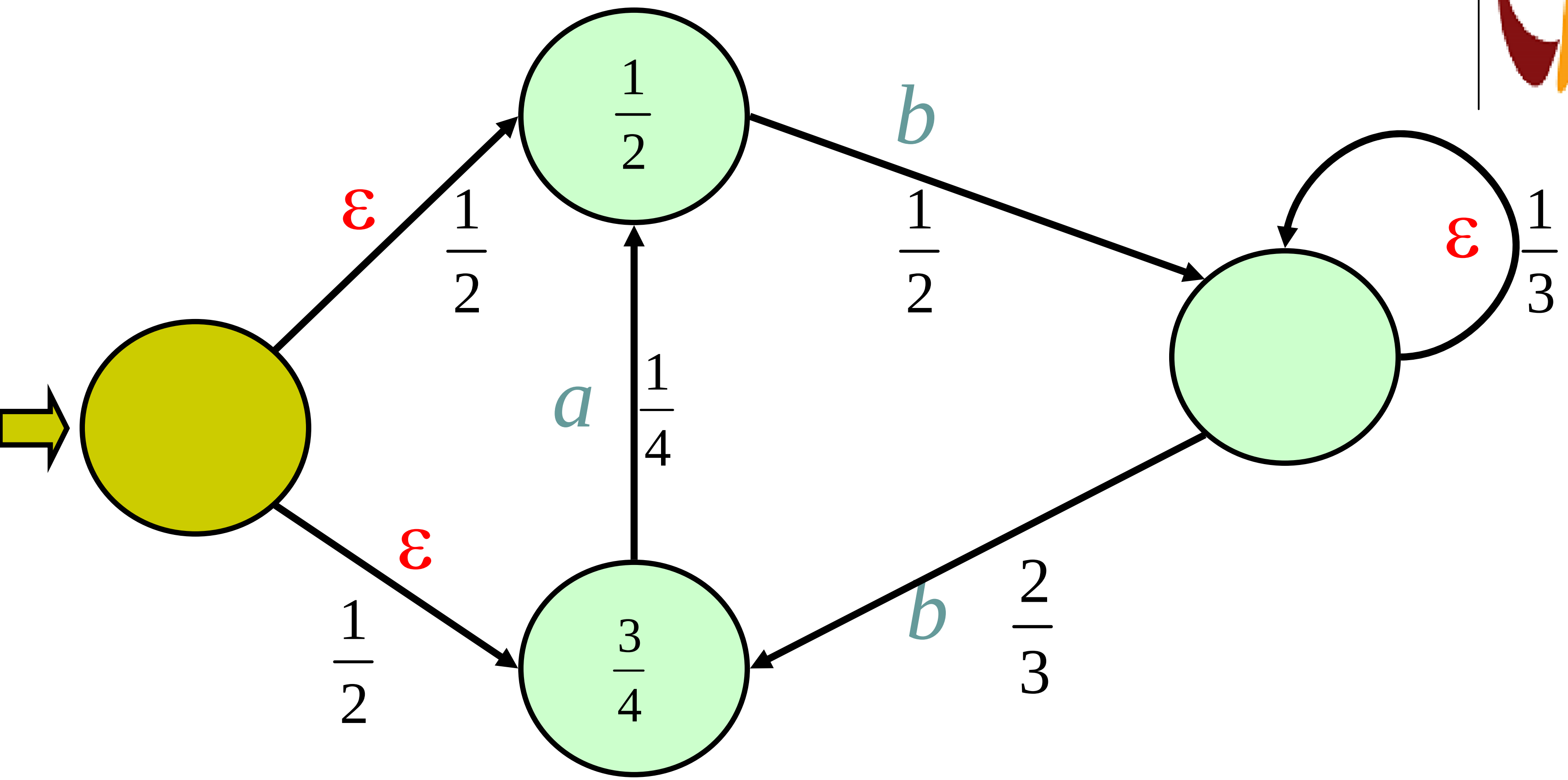
$$\Pr_A(abab) = \frac{1}{2} \times \frac{1}{2} \times \frac{1}{3} \times \frac{2}{3} \times \frac{3}{4} = \frac{1}{24}$$



a
0.7
0.1
b
0.9
a
0.35
a
0.7
b
0.3
0.3
b
0.65



PFA: Probabilistic Finite (state) Automaton



ε-*PFA*: Probabilistic Finite (state) Automaton with ε-transitions

How useful are these automata?

- They can define a distribution over Σ^*;
- They do not tell us if a string belongs to a language;
- They are good candidates for grammar induction;
- There was (is?) not that much written theory.

Basic references

- The *HMM* literature
- Azaria Paz 1973: **Introduction to probabilistic automata**
- Chapter 5 of my book
- Probabilistic Finite-State Machines, Vidal, Thollard, cdlh, Casacuberta & Carrasco
- *Grammatical Inference* papers

2 Automata, definitions

Let D be a distribution over Σ^*.

$$0 \leq Pr_D(w) \leq 1$$

$$\sum_{w \in \Sigma^*} Pr_D(w) = 1$$

A Probabilistic Finite (state) Automaton is a $\langle Q, \Sigma, I, F, P \rangle$

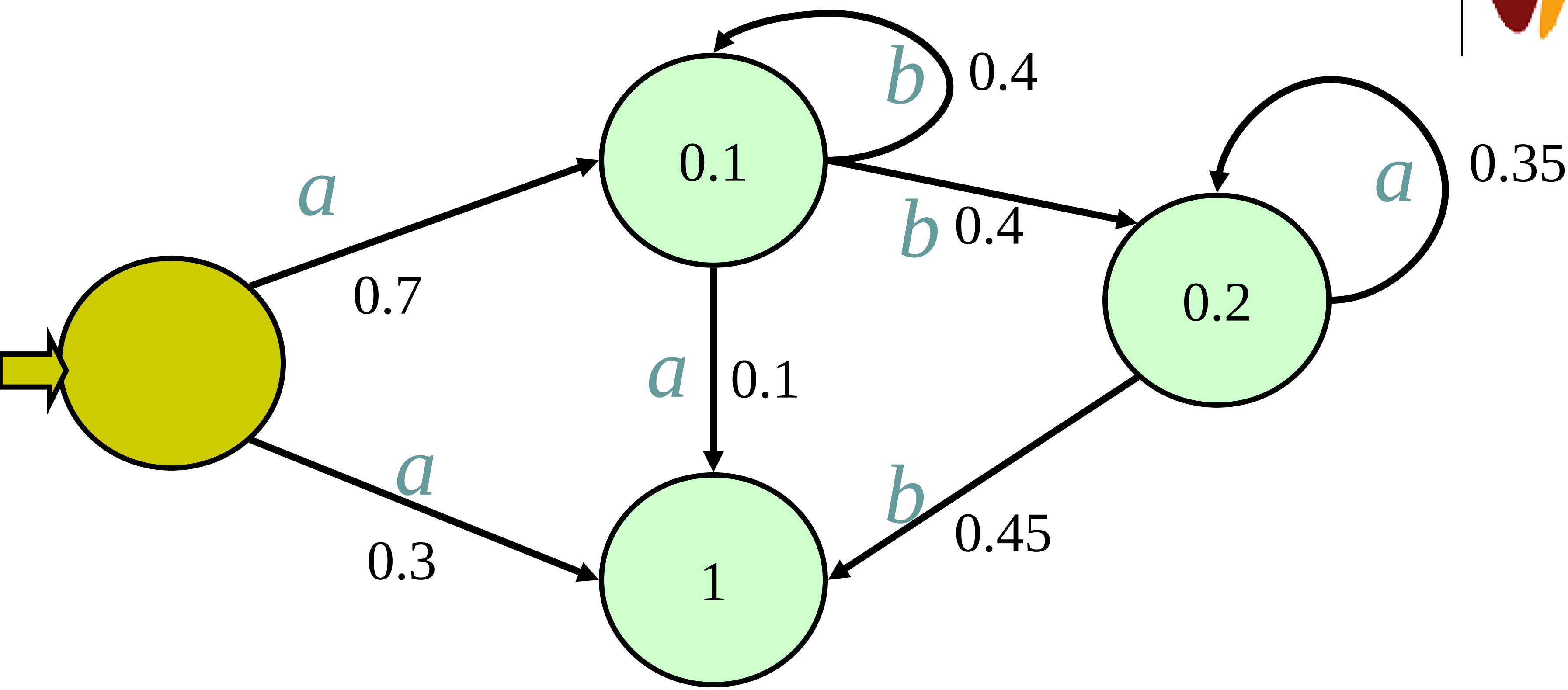
- Q set of states
- $I : Q \rightarrow [0;1]$
- $F : Q \rightarrow [0;1]$
- $P : Q \times \Sigma \times Q \rightarrow]0;1]$

What does a PFA do?

- It defines the probability of each string w as the sum (over all paths reading w) of the products of the probabilities

- $Pr_A(w)=\sum_{paths} p(w)\prod_{ai} P(q,a,q')$

- Note that if ε-transitions are allowed the sum may be infinite



$\Pr(\textcolor{teal}{aba}) = 0.7*0.4*0.1*1 + 0.7*0.4*0.45*0.2$
$= 0.028+0.0252=0.0532$

- non deterministic *PFA*: many initial states/only one initial state;
- an ε-*PFA*: a *PFA* with ε-transitions and perhaps many initial states;
- *DPFA*: a deterministic *PFA*.

Consistency

A PFA is consistent if

- $Pr_A(\Sigma^*)=1$
- $\forall x \in \Sigma^*\ 0 \leq Pr_A(x) \leq 1$

Consistency theorem

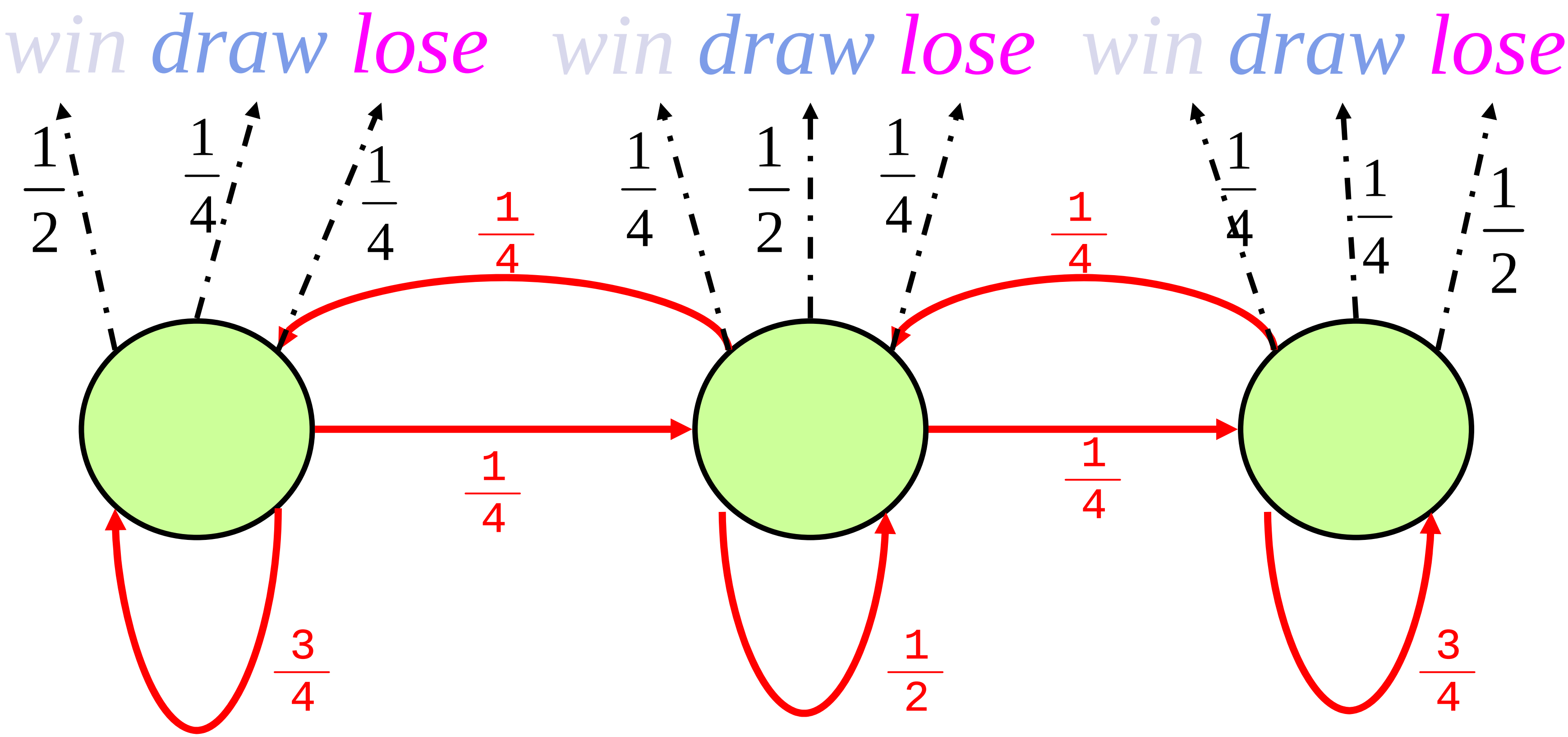
$\mathcal{A}$ is consistent *if* every state is useful (accessible and co-accessible)

$$\forall q \in Q\text{: } F(q) + \sum_{q' \in Q, a \in \Sigma} P(q,a,q') = 1$$

3 Equivalence between models

- Equivalence between *PFA* and *HMM*…
- But the *HMM* usually define distributions over each Σ^n

A *football* HMM



3.1 Equivalence between *PFA* with ε-transitions and *PFA* without ε-transitions

cdlh 2003, Hanneforth& cdlh 2009

- Many initial states can be transformed into one initial state with ε-transitions;
- ε-transitions can be removed in polynomial time;
- Strategy:
  - number the states
  - eliminate first ε-loops, then the transitions with highest ranking arrival state

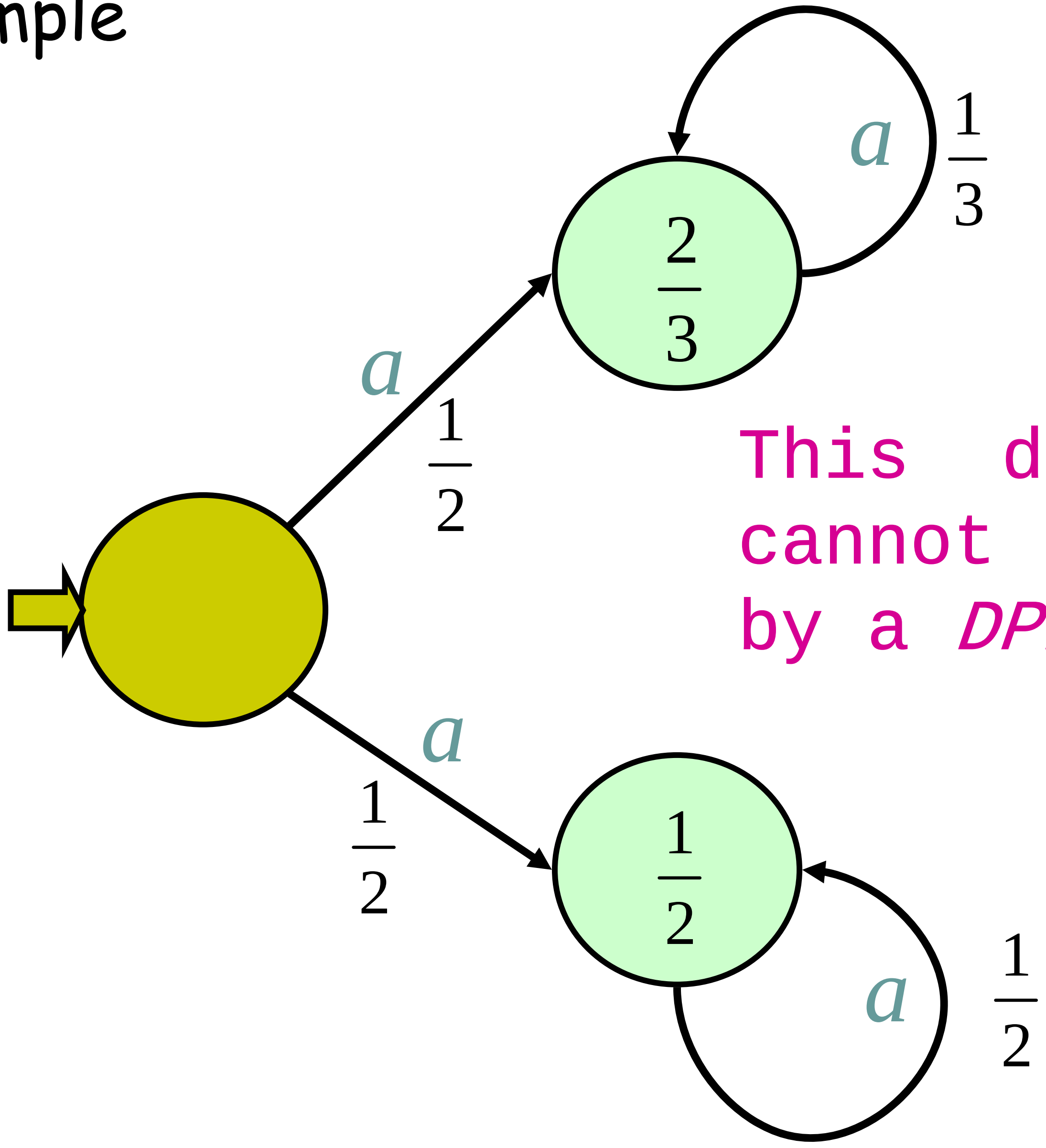
3.2 *PFA* are strictly more powerful than *DPFA*

Folk theorem

(and) You can't even tell in advance if you are in a good case or not.

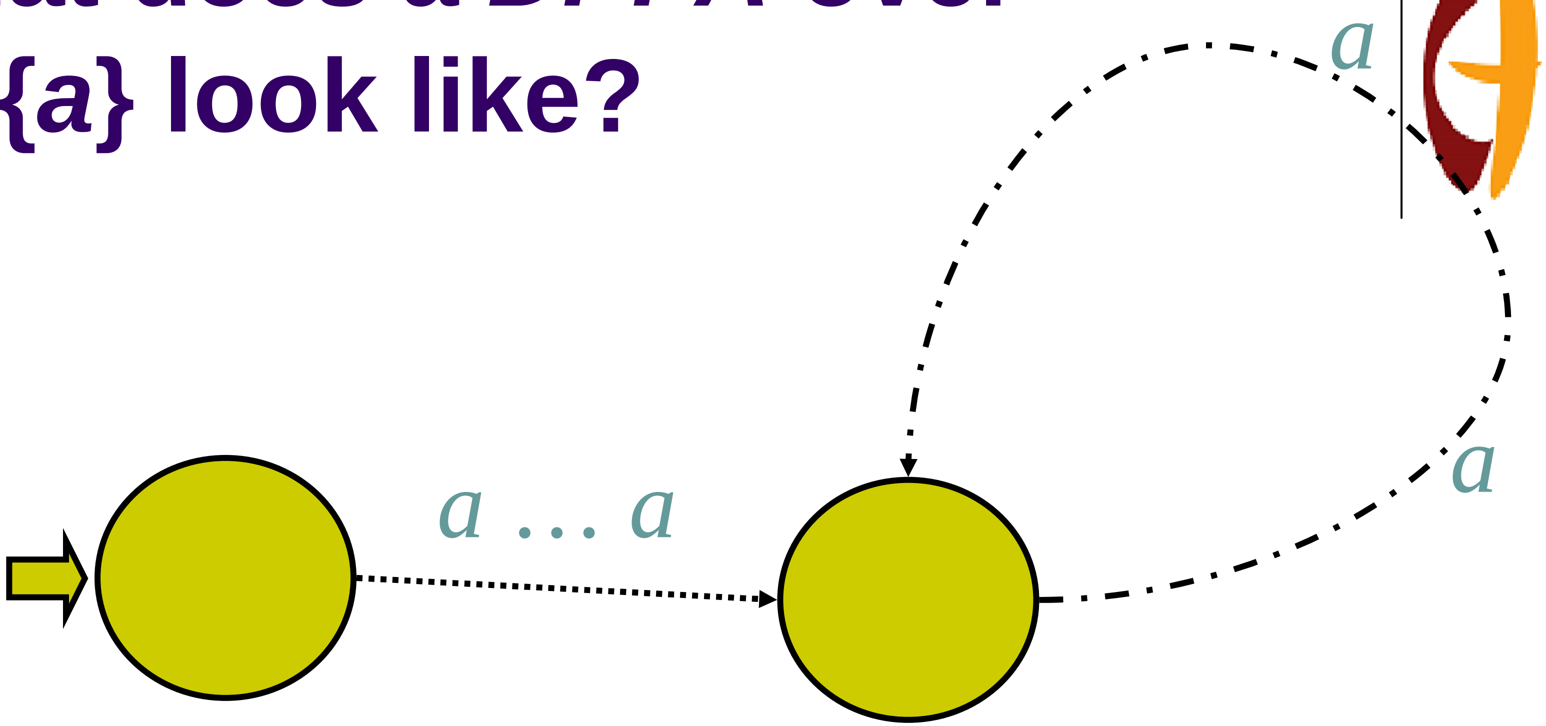
(see: Denis & Esposito 2004)

Example



This distribution cannot be modelled by a *DPFA*

What does a *DPFA* over $\Sigma = \{a\}$ look like?



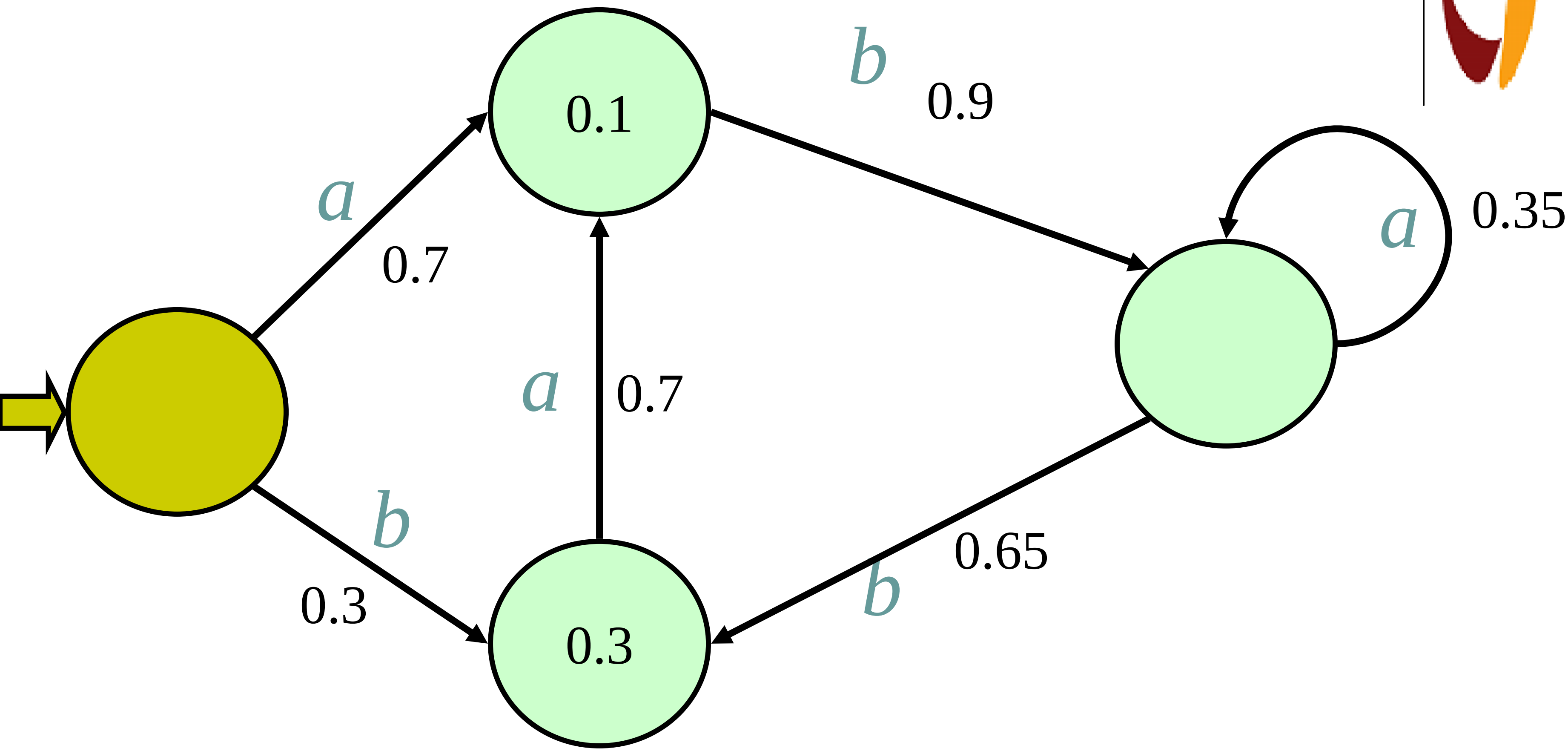
- And with this architecture you cannot generate the previous one.

4 Parsing issues

- Computation of the probability of a string or of a set of strings
- Properties of most probable strings

4.1 Deterministic case

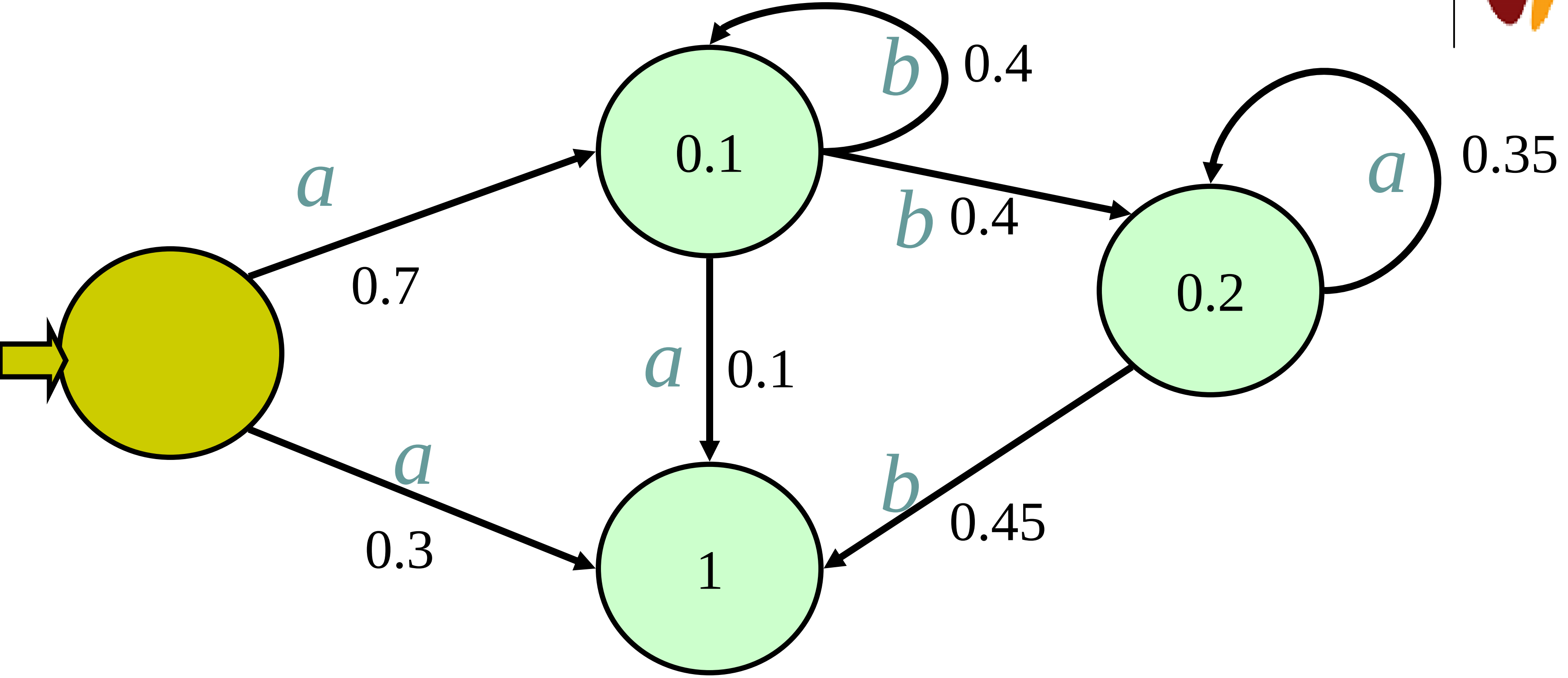
- Simple: apply definitions.
- Technically, rather sum up logs: this is easier, safer and cheaper.



$$\Pr(aba) = 0.7*0.9*0.35*0 = 0$$

$$\Pr(abb) = 0.7*0.9*0.65*0.3 = 0.12285$$

4.2 Non-deterministic case



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

In the literature

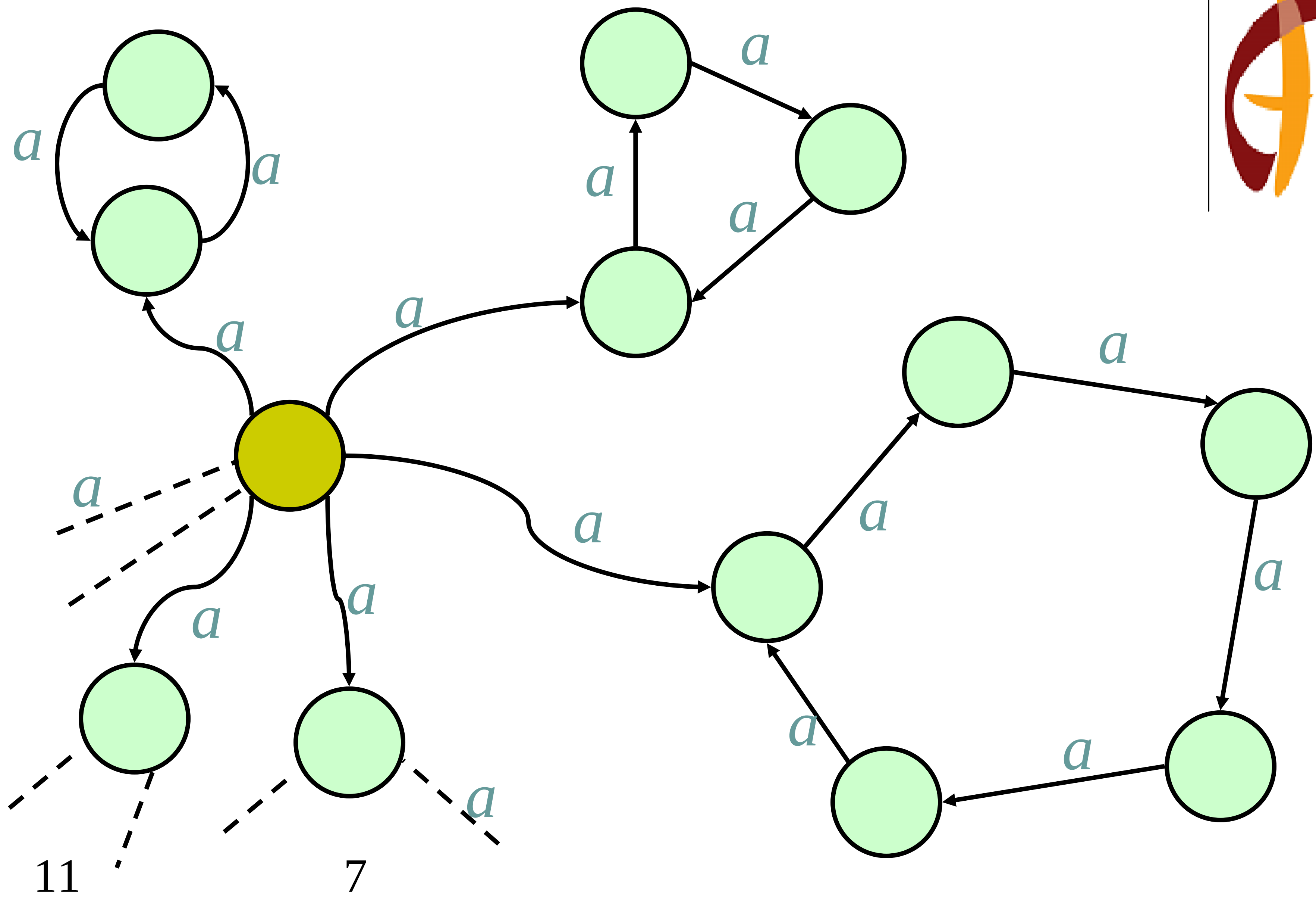
- The computation of the probability of a string is by dynamic programming : $O(n^2m)$
- 2 algorithms : *backward* and *forward*
- If we want the most probable derivation to define the probability of a string, then we can use the *Viterbi* algorithm.

Forward algorithm

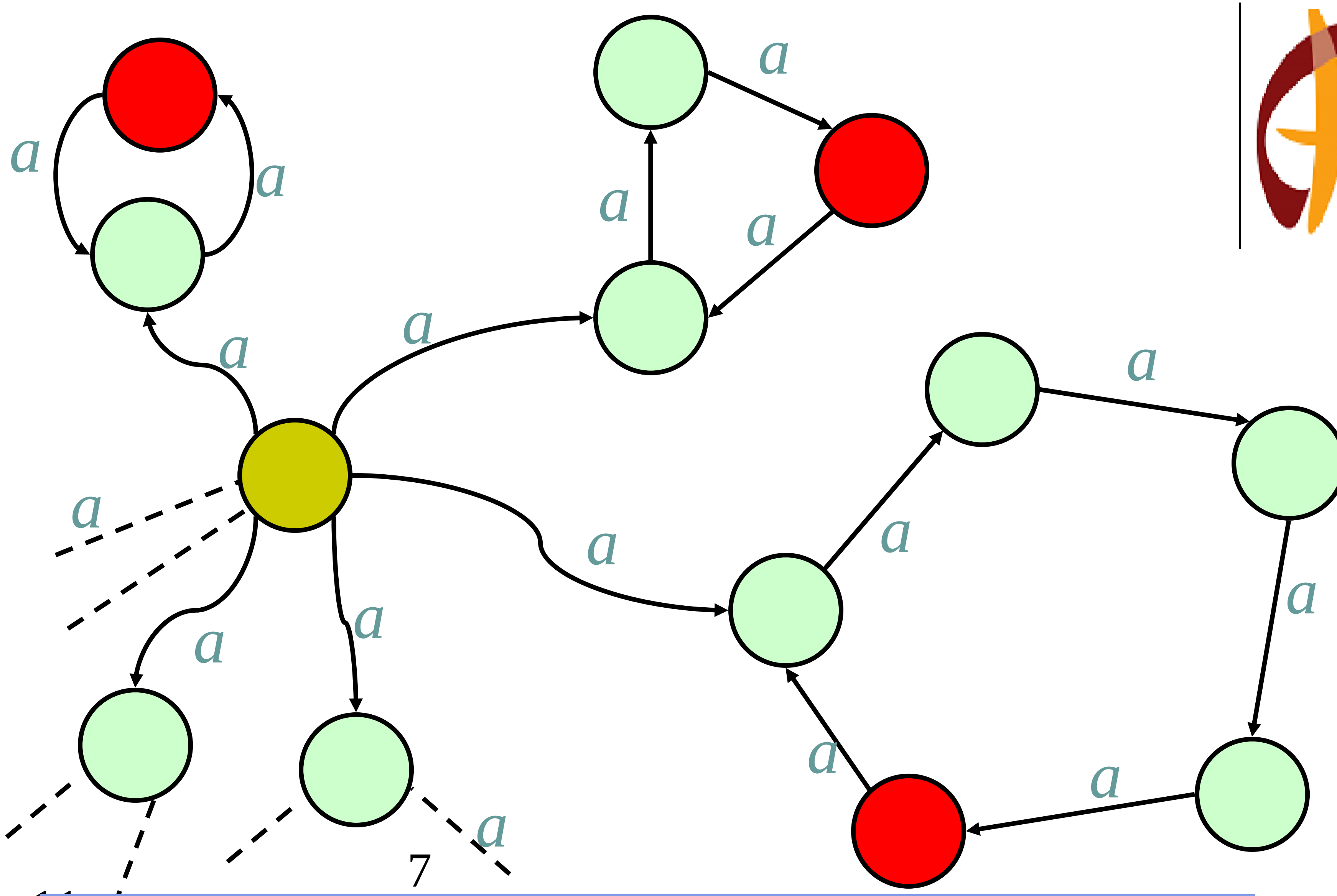
- $A[i,j]=\Pr(q_i|a_1..a_j)$
  (The probability of being in state q_i after having read $a_1..a_j$)
- $A[i,0]=I(q_i)$
- $A[i,j+1]= \sum_{k\leq|Q|} A[k,j] \,.\, P(q_k,a_{j+1},q_i)$
- $\Pr(a_1..a_n)= \sum_{k\leq|Q|} A[k,n] \,.\, F(q_k)$

4.3 Most probable string

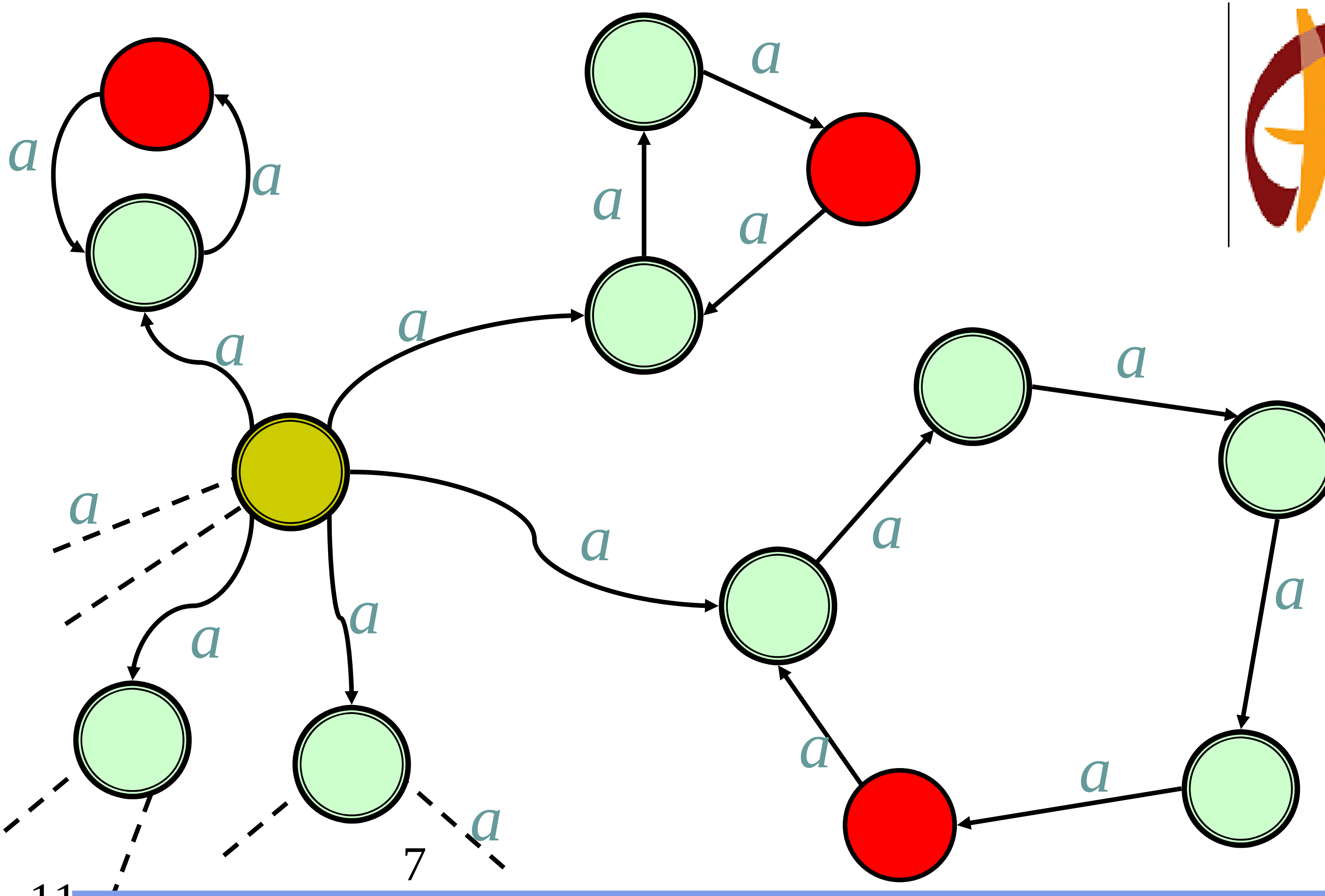
- We prove (Casacuberta & cdlh 2000) that finding the most probable string for a given *PFA* is **NP**-hard.



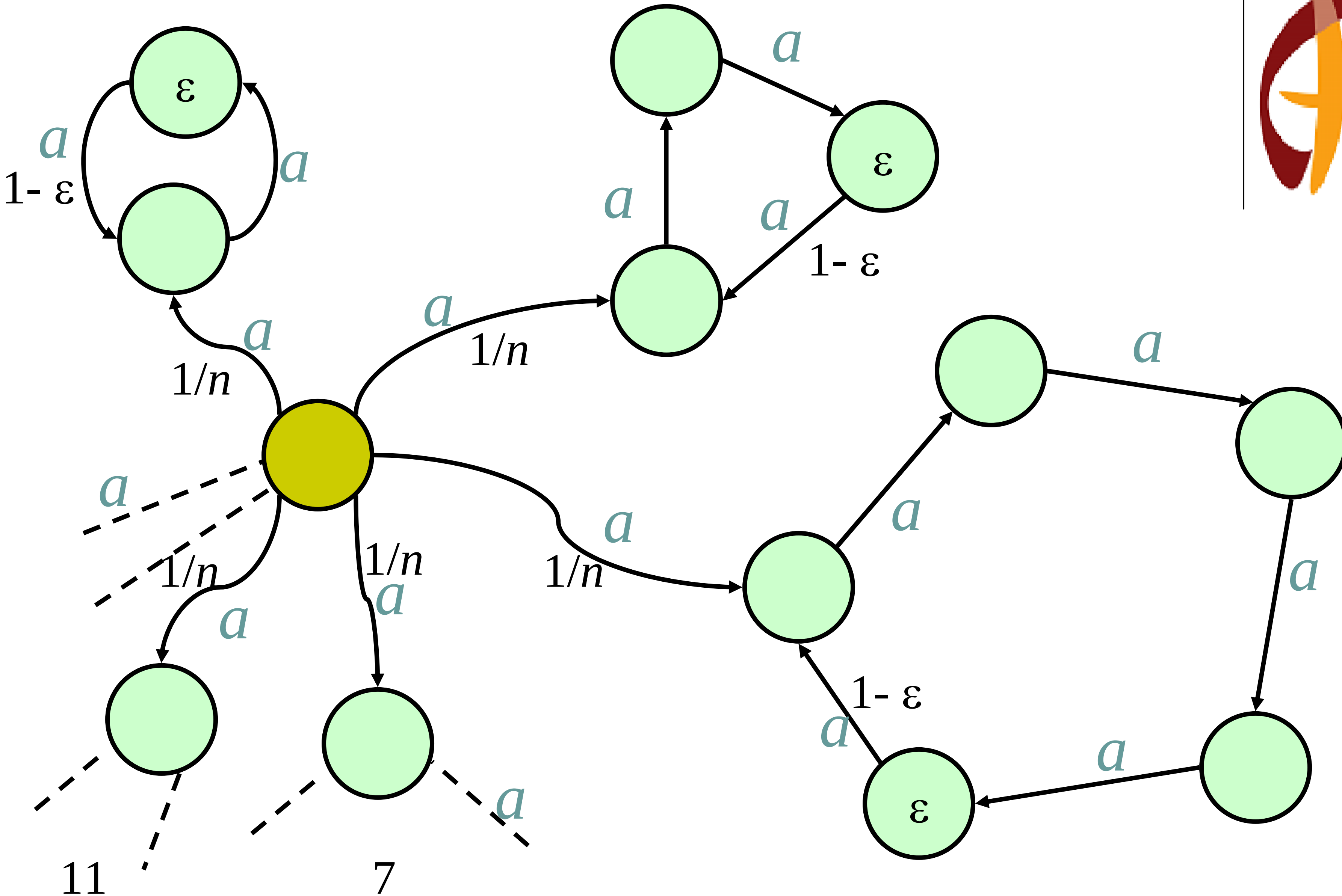
a
a
a
a
a
a
a
a
a
a
a
a
a
a
a
a
a
a
11
7



The shortest string finishing in all the states is exponentially long.



Which is the shortest string not in the language?


ε
a
1- ε
a
a
ε
a
a
1- ε
a
1/n
a
1/n
a
a
a
a
1/n
1/n
a
1/n
a
a
a
a
a
1- ε
ε
a
11
7

A strange problem

Question: Given a *PFA* A, and a rational $r<1$, is there a string w such that $Pr_A(w) \geq r$?

Status : NP hard (Casacuberta & cdlh 2000)

Yet there exists a nice polynomial randomized algorithm

ie in $O(1/\varepsilon, |A|)$ where $\varepsilon=1/r$

4.4 The weight of a language

Question: let A be a stochastic automaton and B be a "normal" one. Compute the weight of B in A.

$$\Pr_A(L(B)) = \sum_{w \in L(B)} \Pr_A(w)$$

Fred, ICGI 2000

5 Distances

- What for?
  - Estimate the quality of a language model;
  - Have an indicator of the convergence of learning algorithms;
  - Construct kernels.

5.1 Entropy

- How many bits do we need to correct our model?
- Two distributions over Σ^*: D et D'

Kullback Leibler divergence (or relative entropy) between D and D':

$$\sum_{w \in \Sigma^*} Pr_D(w) \times \left| \log Pr_D(w) - \log Pr_{D'}(w) \right|$$

5.2 Perplexity

- The idea is to allow the computation of the divergence, but relatively to a test set (*T*).
- An approximation (*sic*) is perplexity: inverse of the geometric mean of the probabilities of the elements of the test set.

$$\prod_{w \in T} Pr_D(w)^{-1/|T|}$$

$$= \frac{1}{\sqrt[|T|]{\prod_{w \in T} Pr_D(w)}}$$

Problem if some probability is null...

Why multiply?

- Suppose we have two predictors for a coin toss.
- Predictor 1: heads 60%, tails 40%
- Predictor 2: heads 100%
- The tests are H: 6, T 4
- Arithmetic mean P1: 36%+16%=0,52
- P2: 0,6
- Predictor 2 is the better predictor ;-)

5.3 Distance d_2

$$d_2(D, D') = \sqrt{\sum_{w \in \Sigma^*} (Pr_D(w) - Pr_{D'}(w))^2}$$

Can be computed in polynomial time if *D* and *D'* are given by *PFA* (Carrasco & cdlh 2002)

This also means that equivalence of PFA is in P.

6 Learning

- From a multi-set of strings, discover, infer, learn the (*D*)*PFA* that could have generated this data.
- What can we say about an algorithm that would do this?

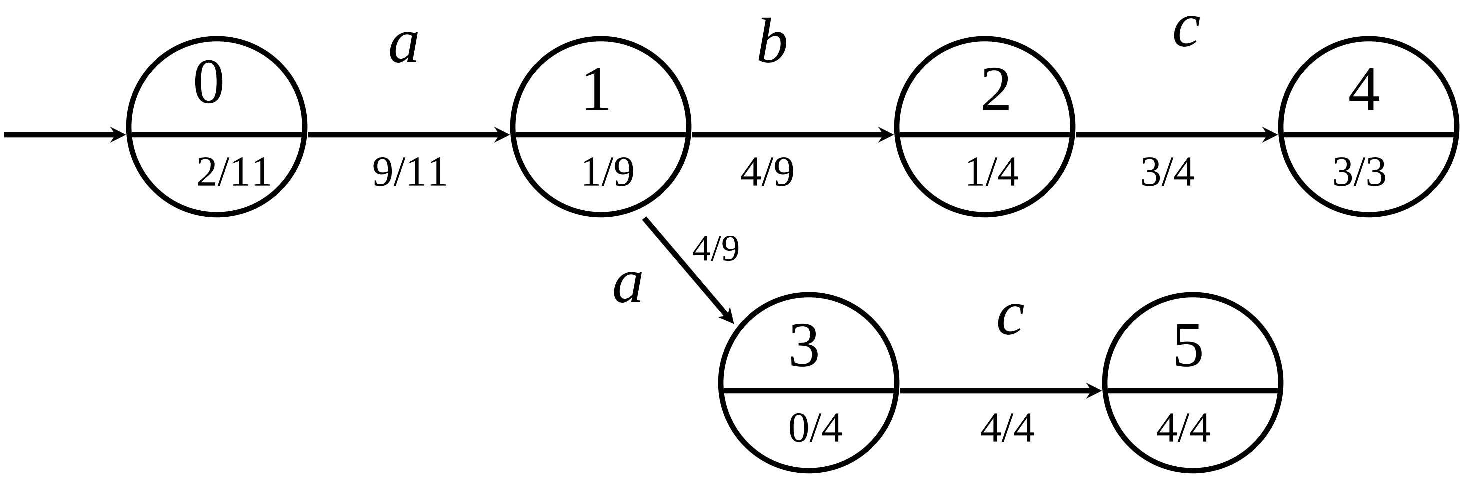
6.1 *Alergia*

Algorithm by Carrasco and Oncina (1994)

- polynomial
- identifies in the limit with probability 1

Inferring stochastic automata

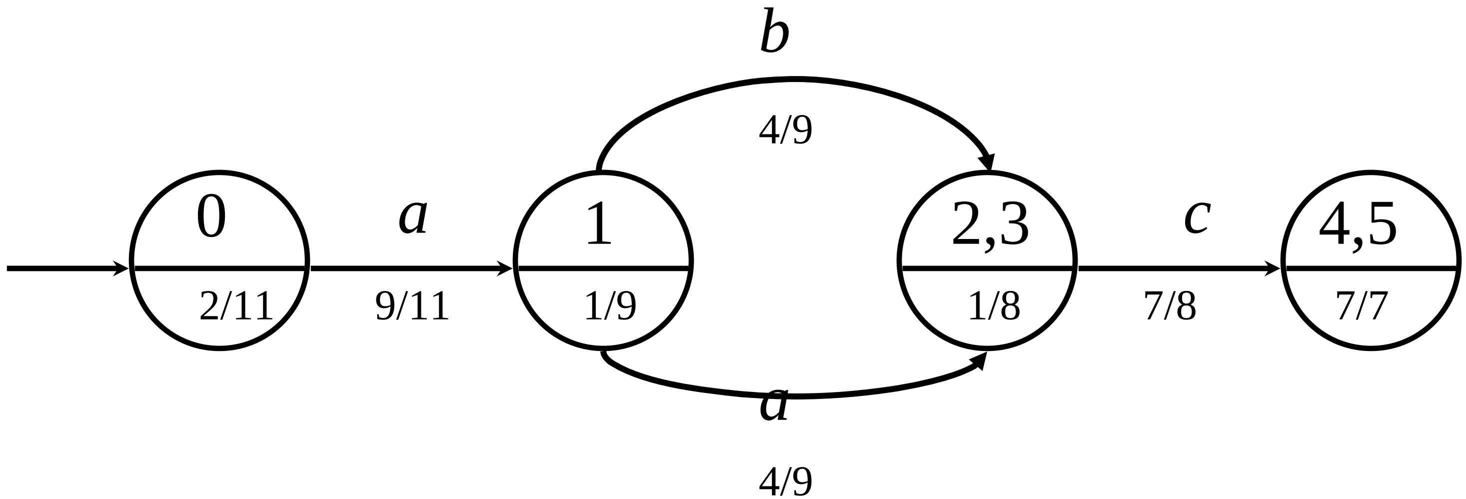
S+ **= {λ (2) ,** ***a*** **(1) ,** ***ab*** **(1) ,** ***aac*** **(4) ,** ***abc*** **(3) }**



Pr(*aac*) = 4/11 = 0.36

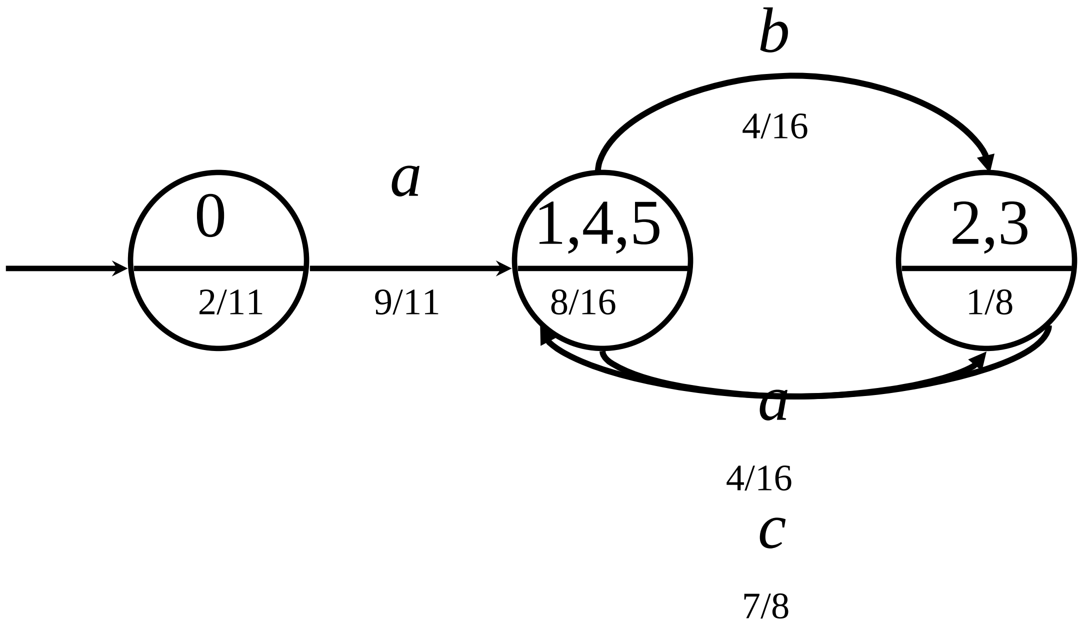
Cascade merging : 2 and 3, 4 and 5

Pr(*aac*) = (9/11).(4/9).(7/8).(7/7) = 0.31



Merging 1 and {4,5}

Pr(*aac*) = (9/11).(4/16).(7/8).(8/16) = 0.08



6.2 MDI

- Thollard, Dupont and cdlh 2000, also Thollard 2001
- Idea: compute the divergence between the model and the data, and accept the merge if the ratio loss of entropy/ gain in size is favorable
- Current results are better than Alergia on classical benchmarks.

State of the art

- Interesting Results (Kermorvant & Dupont 2002) in protein classification task
- In speech, comparable results to state of the art statistical techniques
- Best idea is to mix all good ideas: MDI, heuristics, domain background knowledge...

6.3 Smoothing

- Not allowed to propose null probabilities:
  - Because of perplexity
  - Also because an unseen event should not have null probability (probabilities multiply…)
- You have to probabilize more than Σ^*
- Hard problem…

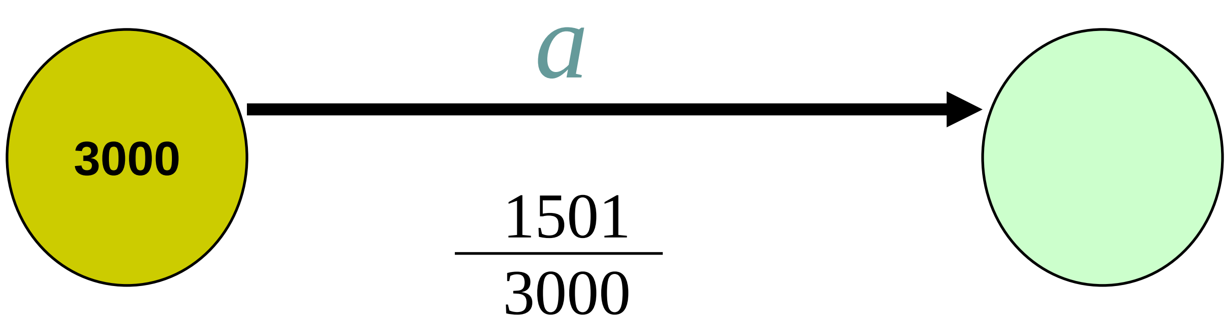
Smoothing

- Through sampling I may not be able to see all possible examples (for example all possible strings).
- Should these unseen events have probability 0?
- How should I adapt the other probabilities to take into account these unseen events?

6.4 Identification of probabilities

- The objective is to identify stochastic automata or grammars.
- If we were able to discover the structure, how do we identify the probabilities?

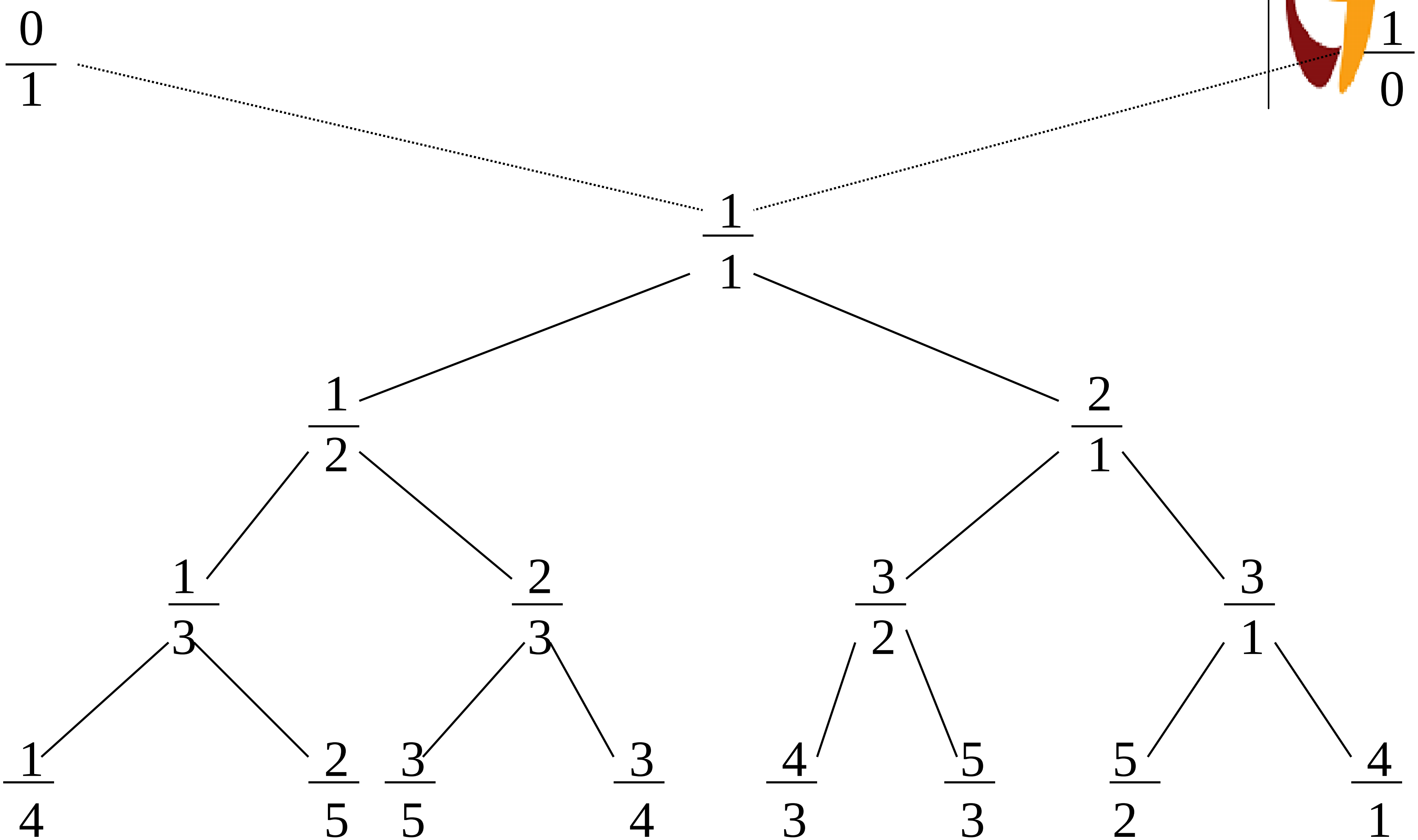
- By estimation: the edge is used 1501 times out of 3000 passages through the state :



Stern-Brocot trees: (Stern 1858, Brocot 1860)

Can be constructed from two simple adjacent fractions by the «mean» operation

$$\frac{a}{b} \; m \; \frac{c}{d} = \frac{a+c}{b+d}$$


0/1
1/0
1/1
1/2
2/1
1/3
2/3
3/2
3/1
1/4
2/5
3/5
3/4
4/3
5/3
5/2
4/1

Idea:

- Instead of returning $c(x)/n$, search the Stern-Brocot tree to find a good simple approximation of this value.

Iterated Logarithm:

With probability 1, for a co-finite number of values of *n* we have:

$$\left| \frac{c(x)}{n} - \frac{a}{b} \right| < \sqrt{\frac{\lambda \log \log n}{n}}$$

$$\forall \lambda > 1$$

7 Open problems

- The extension to probabilistic context-free grammars
- The consensus string problem revisited
- Computation of margins