Natural Language Processing

Spring 2017

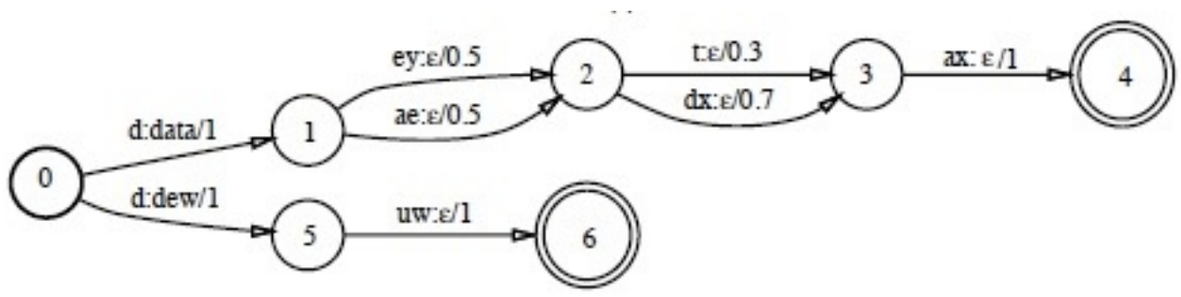
Unit 1: Sequence Models

Lecture 4a: Probabilities and Estimations

Lecture 4b: Weighted Finite-State Machines

required

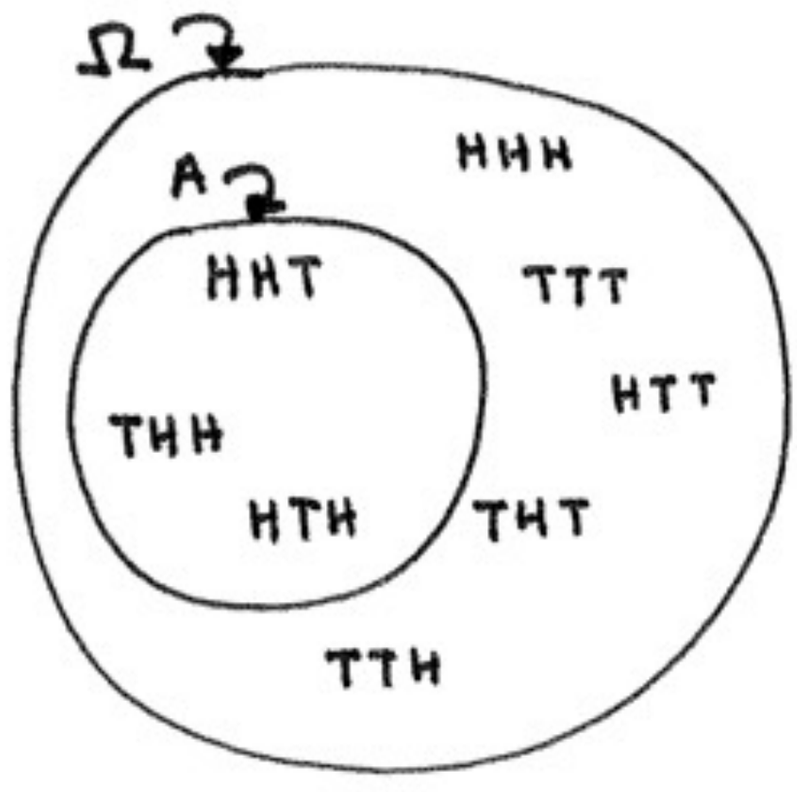
optional



Liang Huang

Probabilities

- experiment (e.g., "toss a coin 3 times")
- basic outcomes Ω (e.g., Ω={HHH, HHT, HTH, ..., TTT})
- event: some subset A of Ω (e.g., A = "heads twice")
- probability distribution
  - a function p from Ω to [0, 1]
  - $\sum_{e \in \Omega} p(e) = 1$
- probability of events (marginals)
  - p (A) = $\sum_{e \in A} p(e)$



Joint and Conditional Probs

joint probability of A and B

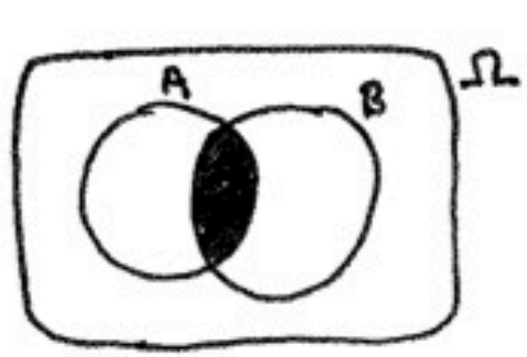
$$P(A,B) = P(A \cap B)$$

Conditional probability of A given B

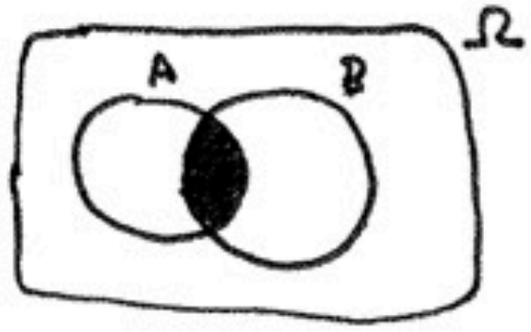
$$P(A|B) \equiv \frac{P(A,B)}{P(B)}$$

{ NOTE:

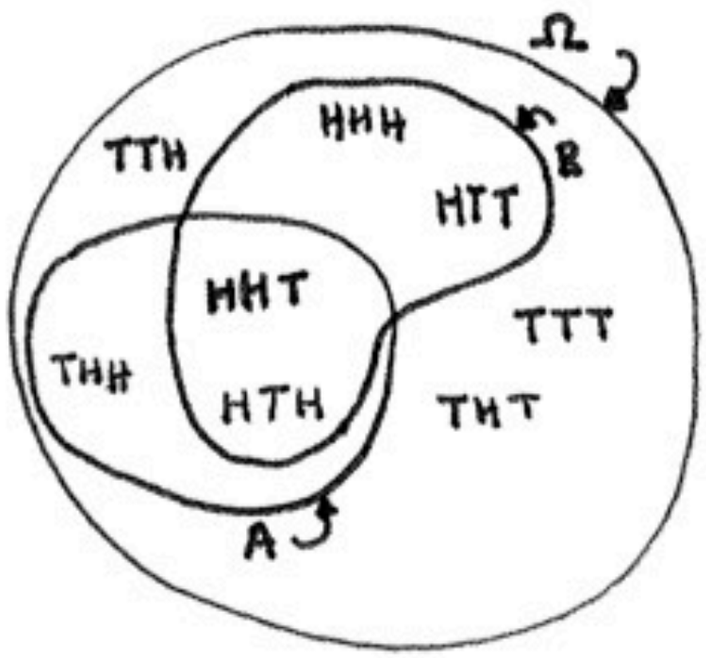
$$P(B|A) \equiv \frac{P(A,B)}{P(A)}$$ }



proportion of Ω taken up by ●.



proportion of B taken up by ●.



A = "heads twice" P(A) = 3/8

B = "first flip = heads" P(B) = 1/2

P(B|A) = "if you got two heads, what is chance the first flip was heads?"

$$P(A,B) \neq P(A) + P(B) \text{ !}$$

Multiplication Rule

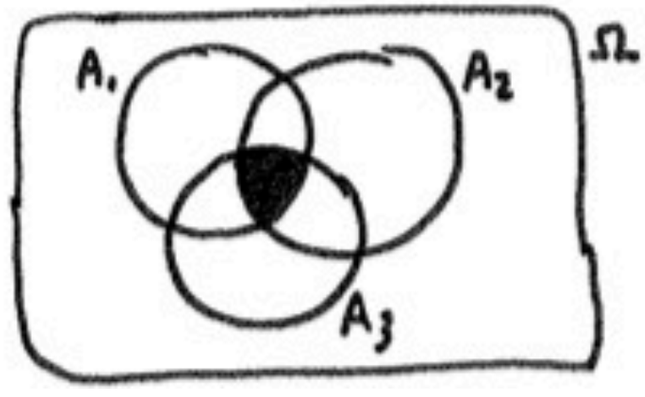
multiplication rule

$$P(A,B) = P(A) \cdot P(B|A) \quad \text{from def}^{n} \text{ of conditional prob.}$$
$$= P(B) \cdot P(A|B)$$

chain rule generalization

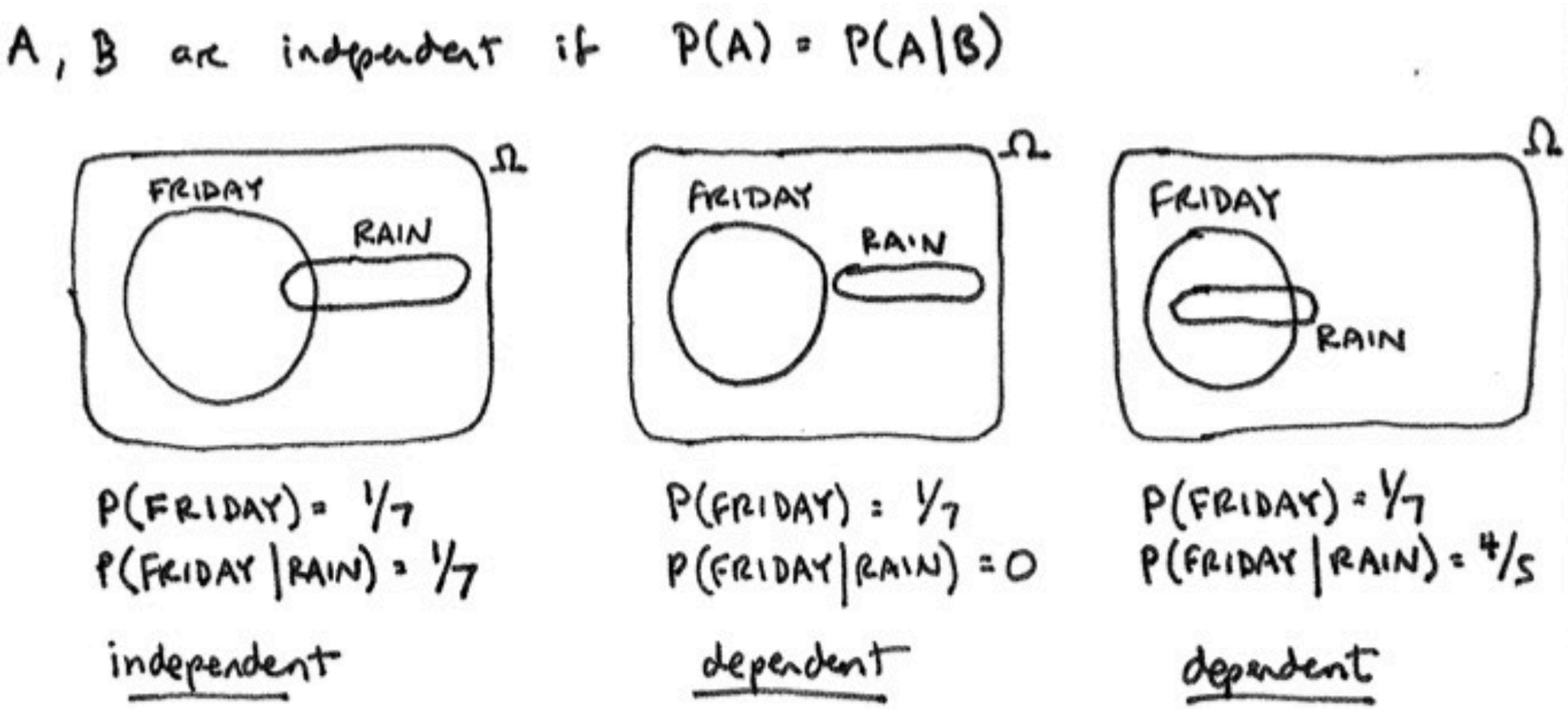
$$P(A_1 \cdots A_n) = P(A_1) \cdot P(A_2|A_1) \cdot P(A_3|A_1, A_2) \cdots P(A_n|A_1 \cdots A_{n-1})$$

$$\frac{P(A_1, A_2, A_3)}{P(A_1, A_2)}$$



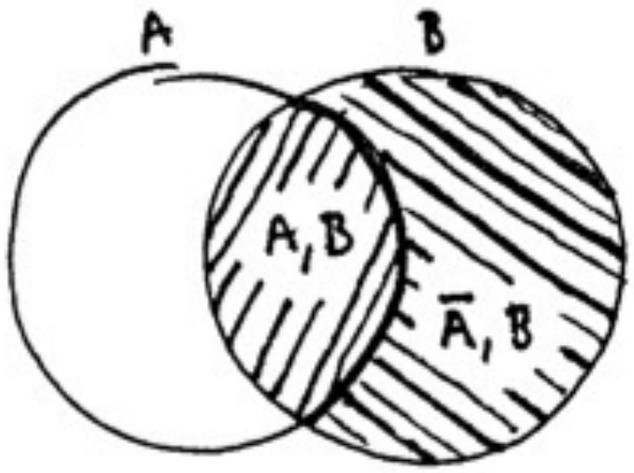
Independence

- P(A, B) = P(A) P(B) or P(A) = P(A|B)
- disjoint events are always dependent! P(A,B) = 0
  - unless one of them is "impossible": P(A)=0
- conditional independence: P(A, B|C) = P(A|C) P(B|C)
  P(A|C) = P (A|B, C)



Marginalization

- compute marginal probs from joint/conditional probs



$$P(B) = P(A,B) + P(\bar{A},B)$$

$$= P(A) \cdot P(B|A) + P(\bar{A}) \cdot P(B|\bar{A})$$

Bayes Rules

Bayes Rule

$$P(A|B) = \frac{P(A) \cdot P(B|A)}{P(B)}$$

proof: $P(A,B) = P(A) \cdot P(B|A)$

$= P(B) \cdot P(A|B)$

⇑ equate & divide by $P(B)$

alternative bayes rule by partition

Most Likely Event

most likely event

$$\underset{A_i}{\text{argmax}} \; P(A_i) \qquad\qquad \underset{A_i}{\text{argmax}} \; P(A_i|B)$$

(e.g., if $A_i \in \{\text{rain}, \text{snow}, \text{sun}\}$
and $P(\text{rain}) = 0.1$
$P(\text{snow}) = 0.1$
$P(\text{sun}) = 0.8$
then
$\underset{A_i}{\text{argmax}} \; P(A_i) = \text{sun}$)

Most Likely Given ...

$$\underset{A_i}{\operatorname{argmax}} \; P(A_i \mid B) = \underset{A_i}{\operatorname{argmax}} \; P(A_i) \cdot P(B \mid A_i)$$

the right-hand side is often easier to work with.

e.g., let B be a misspelled character sequence
let $A_1 \cdots A_n$ be possible corrections

$$P(A_i \mid B) \Rightarrow P(\text{knight} \mid \text{kmight})$$

$$P(A_i) \cdot P(B \mid A_i) \Rightarrow \underbrace{P(\text{knight})}_{} \cdot P(\text{kmight} \mid \text{knight})$$

the "prior" term allows us to bring a lot of background knowledge to bear.

Estimating Probabilities

- how to get probabilities for basic outcomes?
  - do experiments
  - count stuff
- e.g. how often do people start a sentence with "the"?
  - P (A) = (# of sentences like "the ..." in the sample) / (# of all sentences in the sample)
  - P (A | B) = (count of A, B) / (count of B)
- we will show that this is Maximum Likelihood Estimation

Model

- what is a MODEL?
  - a general theory of how the data is *generated*,
  - along with a set of parameter estimates
- e.g., given this statistics

result	count
1	0
2	7
3	26
4	45
5	119
6	70
7	31
8	14
9	6
10	0
11	4
12	2

  - we can "guess" it's generated by a 12-sided die
    - along with 11 free parameters p(1), p(2), ..., p(11)
  - alternatively, by two tosses of a single 6-sided die
    - along with 5 free parameters p(1), p(2), ..., p(5)
- which is better *given* the data? which better *explains* the data?

$$\text{argmax}_m\ p(m|d) = \text{argmax}_m\ p(m)\ p(d|m)$$

Maximum Likelihood Estimation

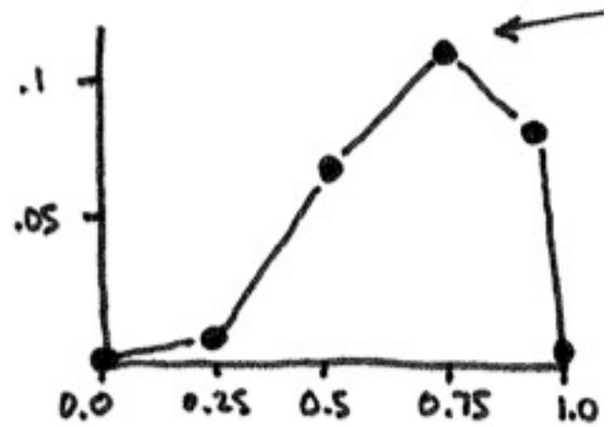
- always maximize posterior: what's the best m given d?
- when do we use maximum likelihood estimation?
- with uniform prior, same as likelihood (explains data)
  - $\text{argmax}_m\ p(m|d) = \text{argmax}_m\ p(m)\ p(d|m)$ bayes, and p(d)=1
  - $= \text{argmax}_m\ p(d|m)$ when p(m) uniform

Suppose d = H H T H

m_1 coin is unbiased $P(d|m) = \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} = 0.066$

m_2 coin is biased so that $P(H) = 3/4$ $P(d|m) = \frac{3}{4} \cdot \frac{3}{4} \cdot \frac{1}{4} \cdot \frac{3}{4} = 0.105$

m_3 coin is biased so that $P(H) = 9/10$ $P(d|m) = \frac{9}{10} \cdot \frac{9}{10} \cdot \frac{1}{10} \cdot \frac{9}{10} = 0.073$



How do we rigorously derive this?

- assuming any $p_m(H) = \theta$ is possible, what's the best θ?
- e.g.: data is still H, H, T, H.
- $\text{argmax}_\theta\ p(d|m; \theta) = \text{argmax}_\theta\ \theta^3 (1-\theta)$
  - take derivatives, make it zero: $\theta = 3/4$.
  - works in the general case: $\theta = n / (n+m)$ (n heads, m tails)
- this is why MLE is just count & divide in the discrete case

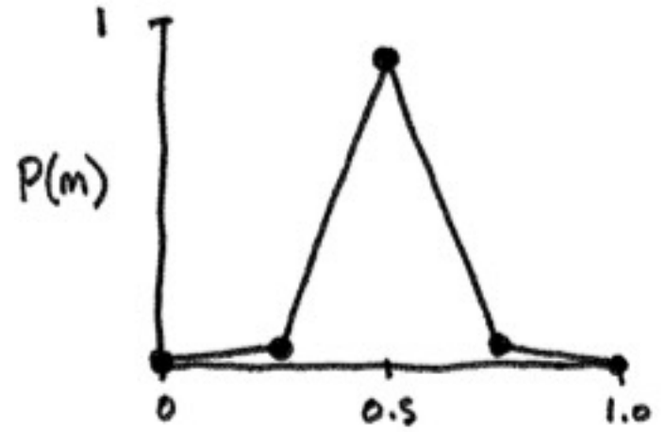
What if we have some prior?

m_1 coin is unbiased $P(m) = 0.90$

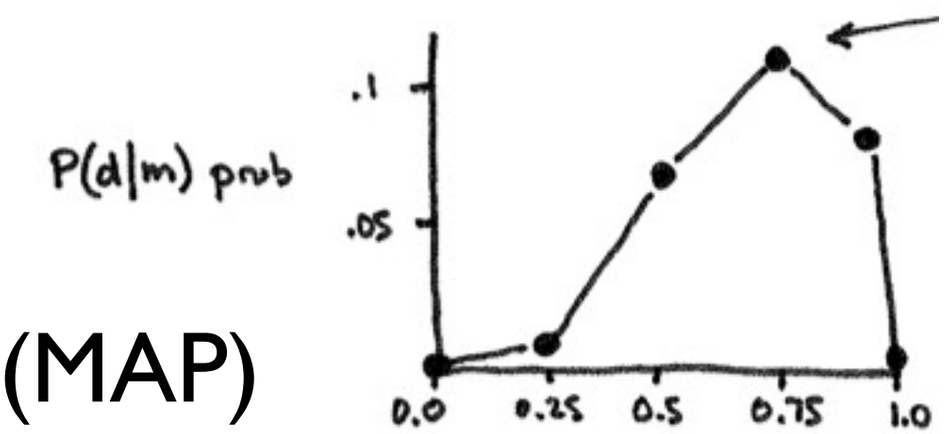
m_2 coin is biased 3/4 $P(m) = 0.01$

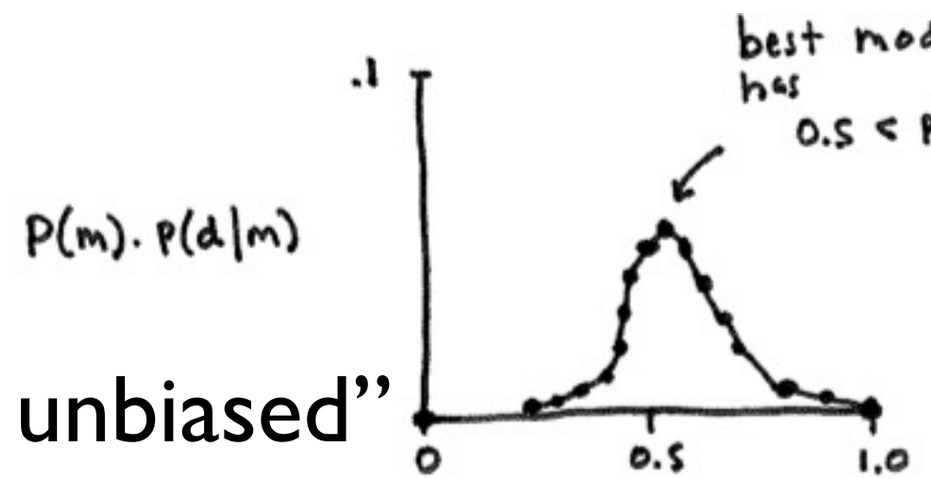
m_3 coin is biased 1/4 $P(m) = 0.01$

} just "made up" !



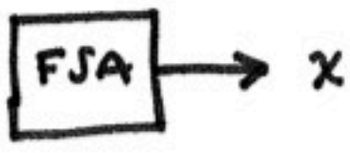
- what if we have arbitrary prior
  - like $p(\theta) = \theta\ (1-\theta)$
- maximum a posteriori estimation (MAP)
- MAP approaches MLE with infinite
- MAP = MLE + smoothing
  - this prior is just "extra two tosses, unbiased"
  - you can inject other priors, like "extra 4 tosses, 3 Hs"





Probabilistic Finite-State Machines

- adding probabilities into finite-state acceptors (FSAs)
  - FSA: a set of strings; WFSA: a distribution of strings



accepts/generates some strings,
rejects others.

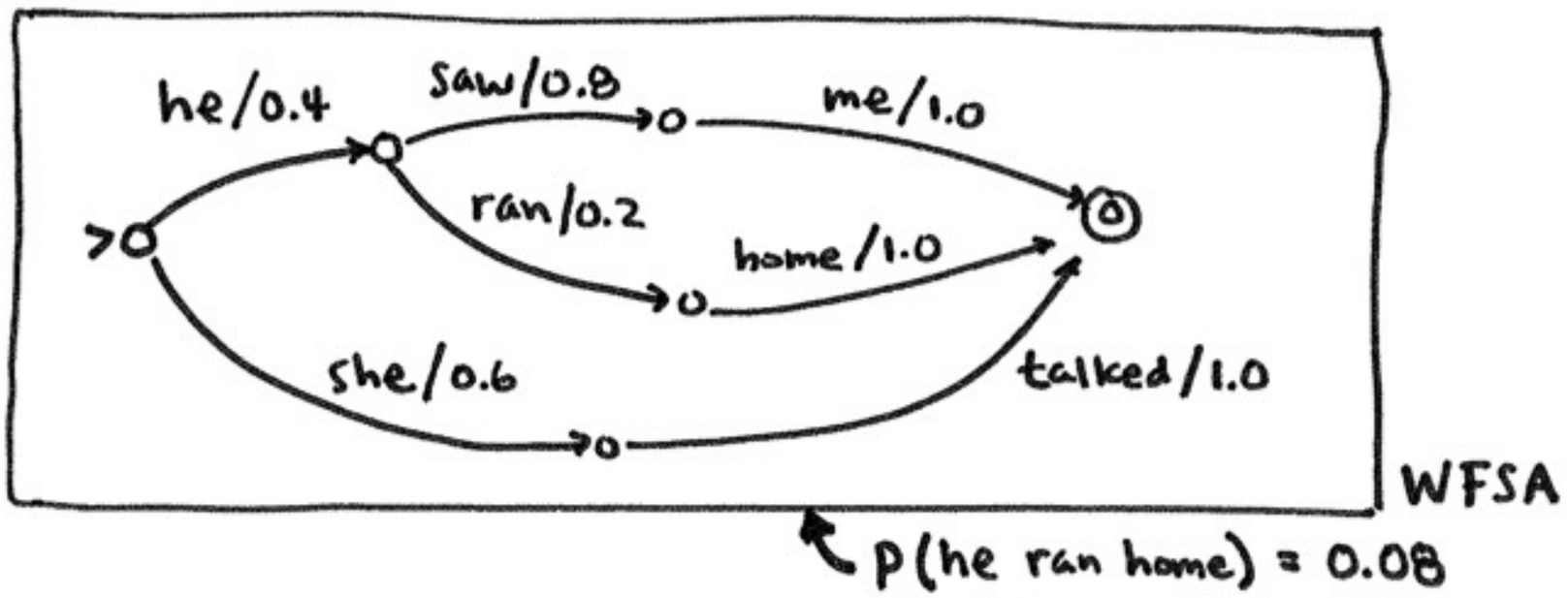
associates a prob. P(x) w/each x.
if P(x) = 0, x is rejected.

weighted FSA represents set of pairs, e.g.:

{<you too, 10^{-6}>,
<you two, 10^{-7}>,
<you to, 10^{-21}>,
<he the, 0>,
<i eat fish, 10^{-8}>
...}

WFSA

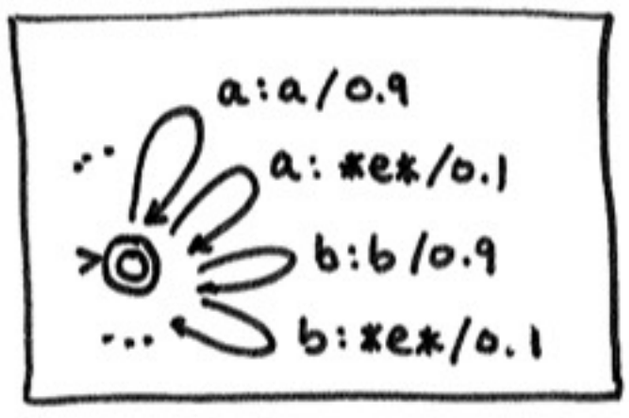
- normalization: transitions leaving each state sum up to 1
  - defines a distribution over strings?
  - or a distribution over paths?
    - => also induces a distribution over strings



WFSTs

- FST: a relation over strings (a set of string pairs)
- WFST: a probabilistic relation over strings (a set of <s, t, p>: strings pair <s, t> with probability p)
  - what is p representing?

{<they can fish, PRO AUX V, 0.99>,
<they can fish, PRO V N, 0.01>,
<they can fish, DT DT DT, 0.00>,
<they sing, PRO V, 1.00>,
...}



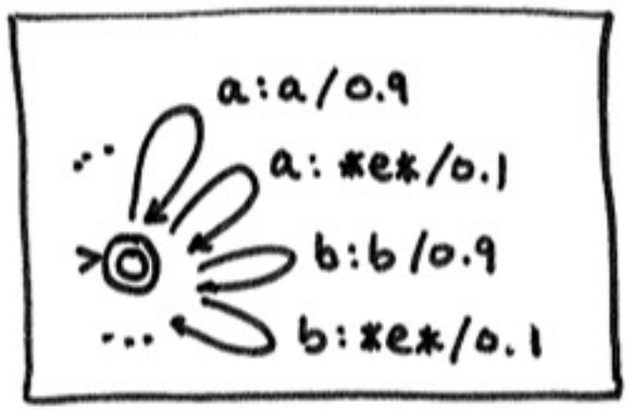
WFST

deletes any input letter w/ probability 0.1

P(ade | acdet) = 0.9 · 0.1 · 0.9 · 0.9 · 0.1 = .00729

Edit Distance as WFST

- this is simplified edit distance
- real edit distance as an example of WFST, but not PFST



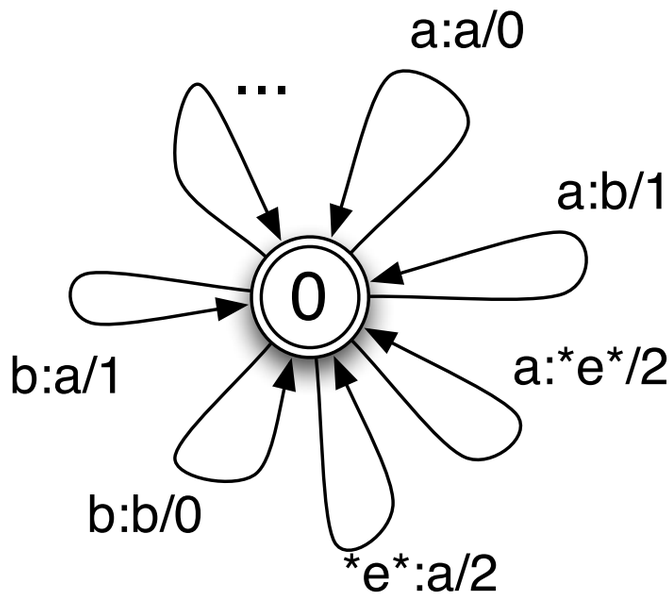
WFST

deletes any input letter w/ probability 0.1

$P(\text{ade} \mid \text{acdet}) = 0.9 \cdot 0.1 \cdot 0.9 \cdot 0.9 \cdot 0.1 = .00729$

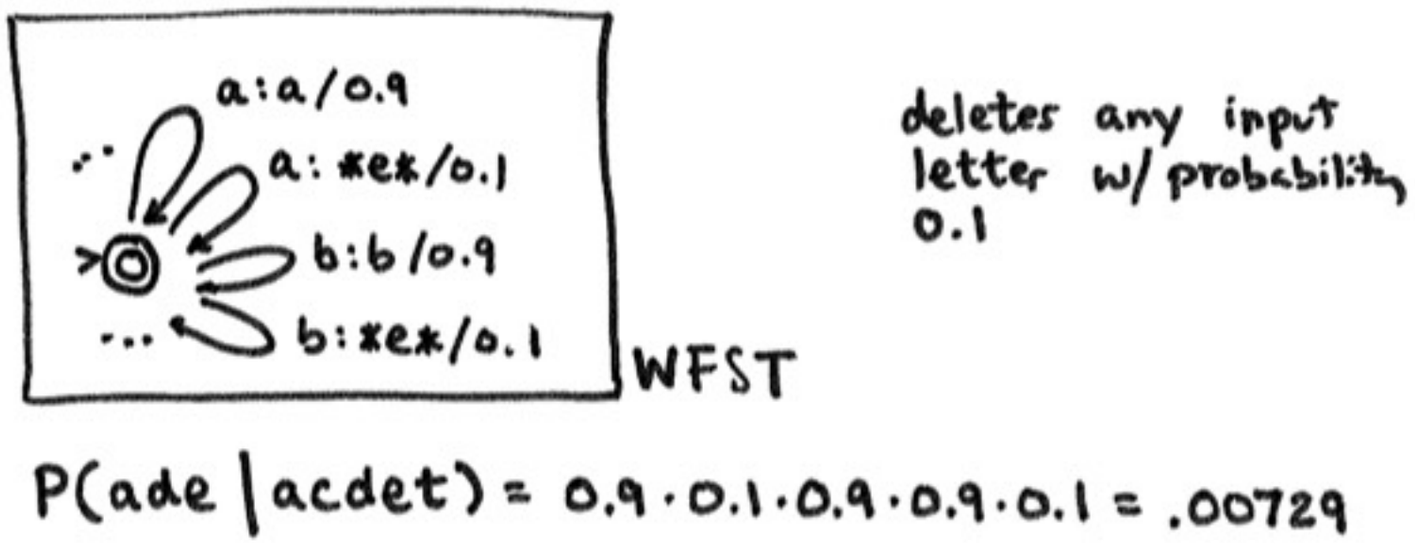
WFST: real edit distance

costs:
replacement: 1
insertion: 2
deletion: 2



Normalization

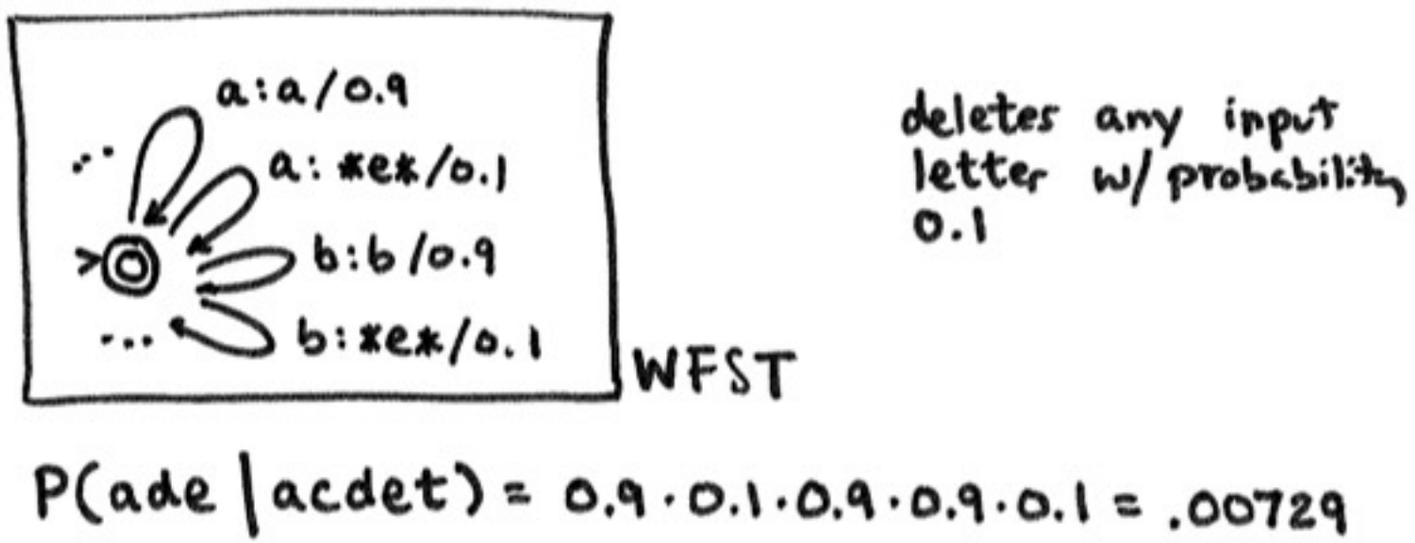
- if transitions leaving each state and each input symbol sum up to 1, then...
  - WFST defines conditional prob p(y|x) for x => y
- what if we want to define a joint prob p(x, y) for x=>y?
- what if we want p(x | y)?



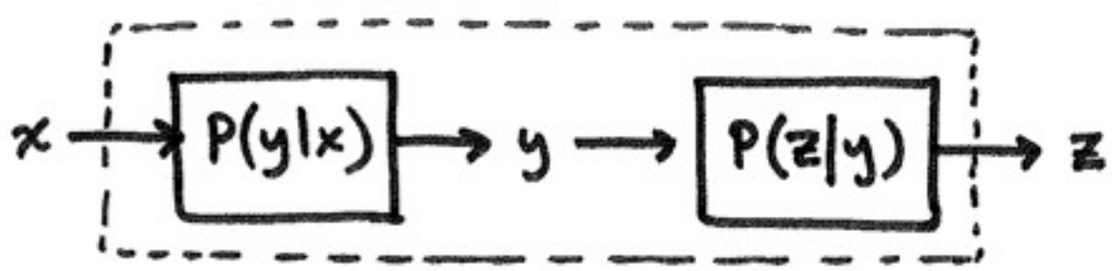
Questions of WFSTs

- given x, y, what is p(y|x) ?
- for a given x, what's the y that maximizes p(y|x) ?
- for a given y, what's the x that maximizes p(y|x) ?
- for a given x, supply all output y w/ respective p(y|x)

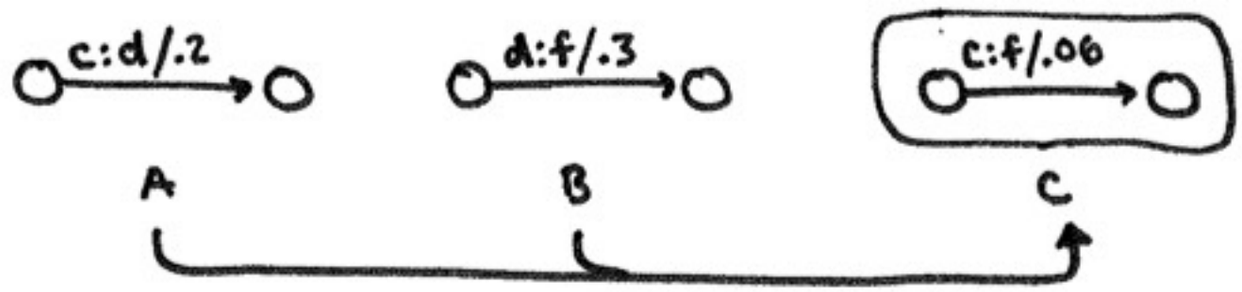
- for a given y, supply all input x w/ respective p(y|x)



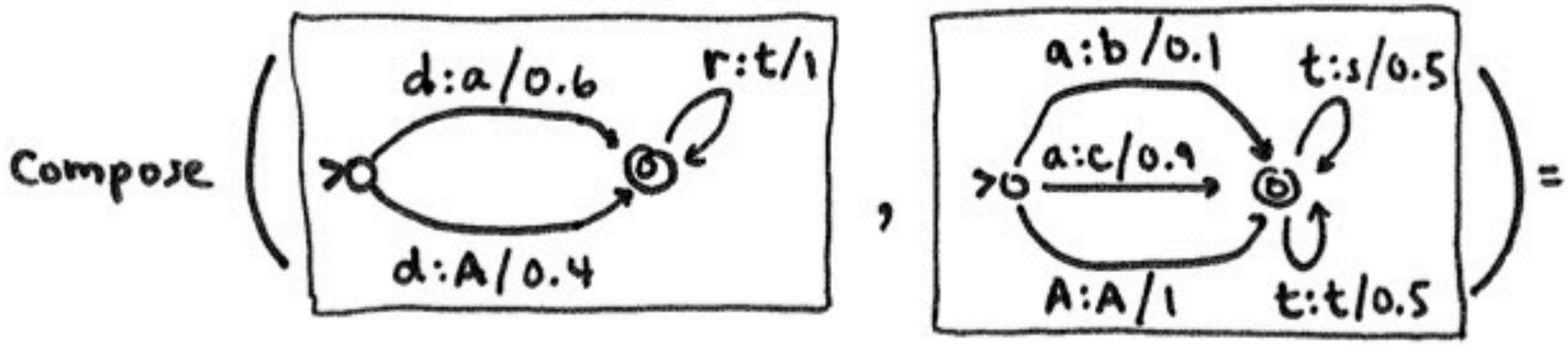
Answer: Composition



- p (z | x) = p (y | x) p (z | y) ???
  - = sum_y p (y | x) p (z | y) have to sum up y
    - given y, z & x are independent in this cascade - Why?
- how to build a composed WFST C out of WFSTs A, B?
  - again, like intersection
  - sum up the products
  - (+, x) semiring



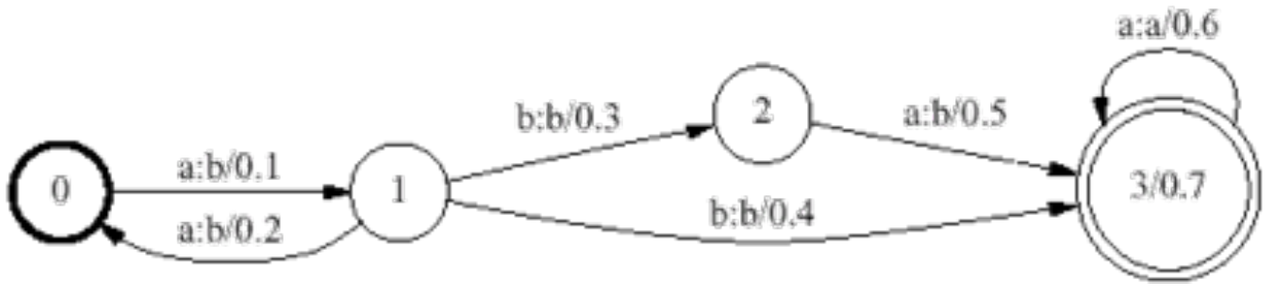
Example

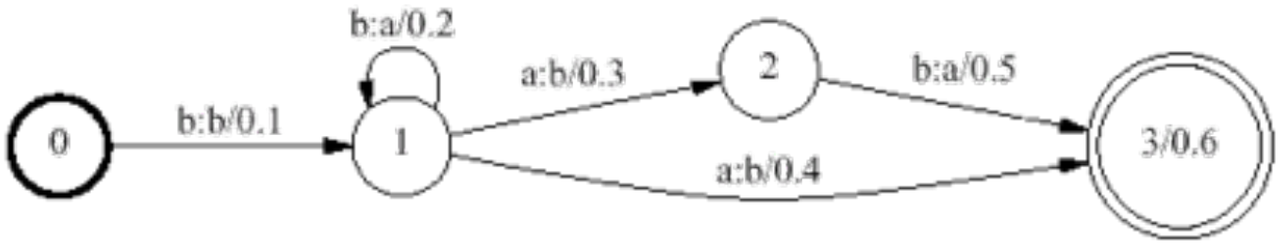


$P(bt|dr) = 0.06 \cdot 0.5 = 0.03$

Example

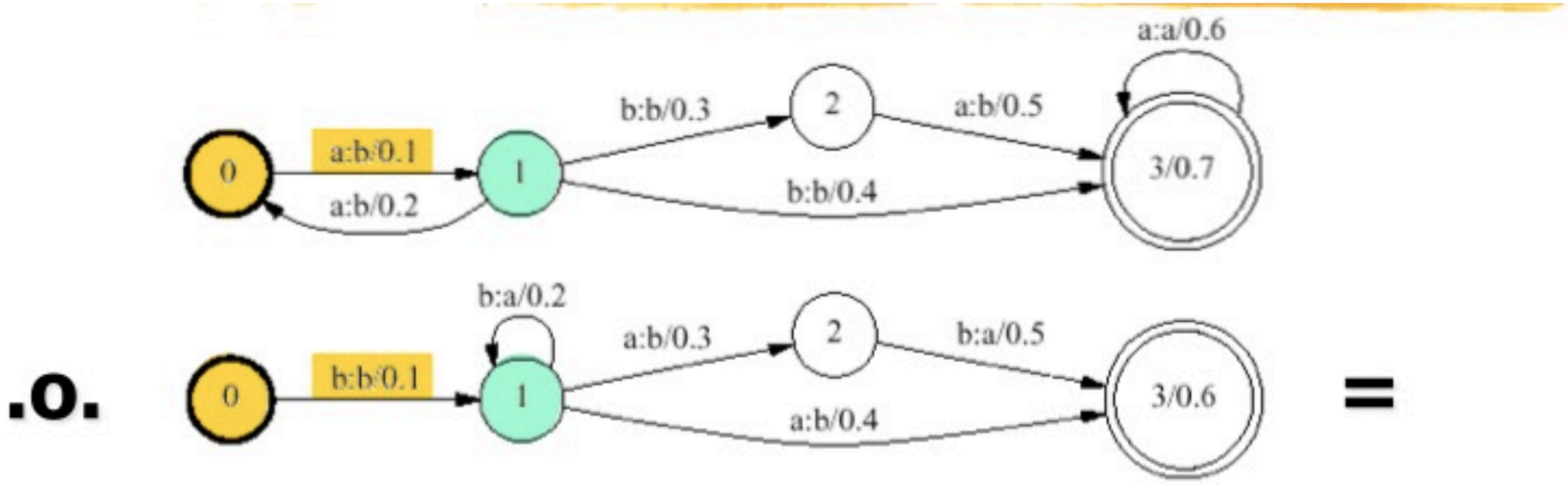
from M. Mohri and J. Eisner



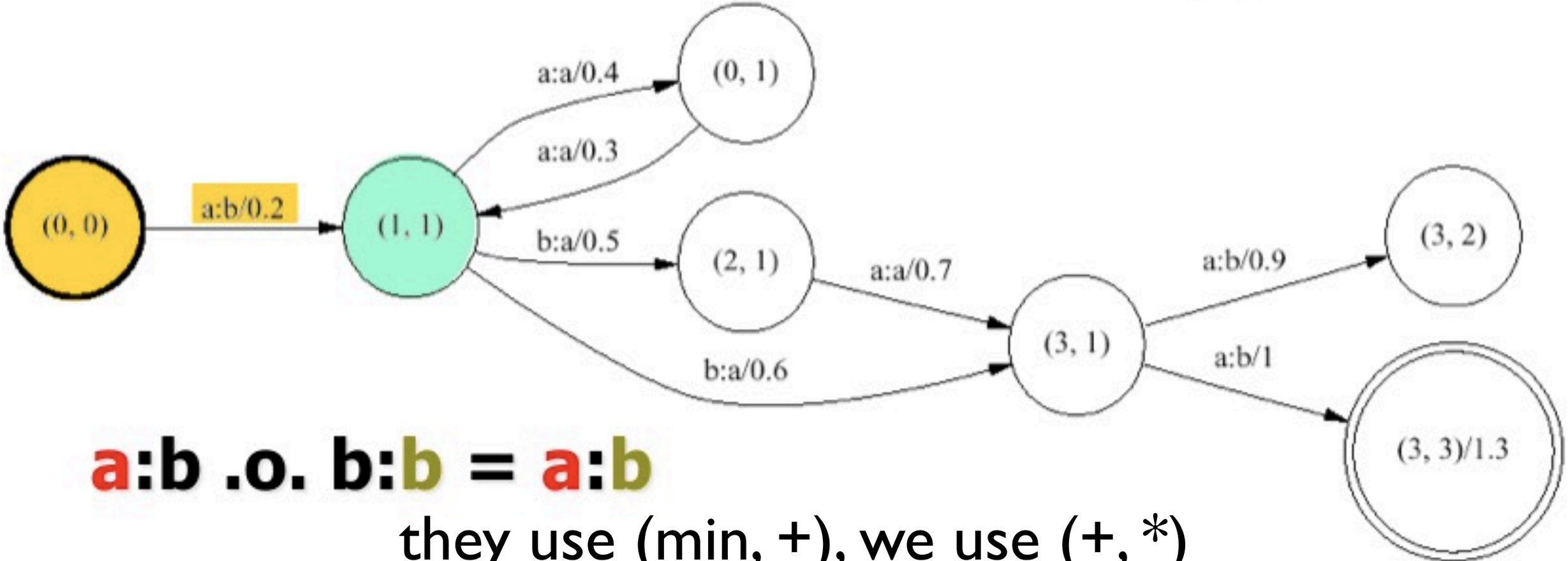


they use (min, +), we use (+, *)

Example



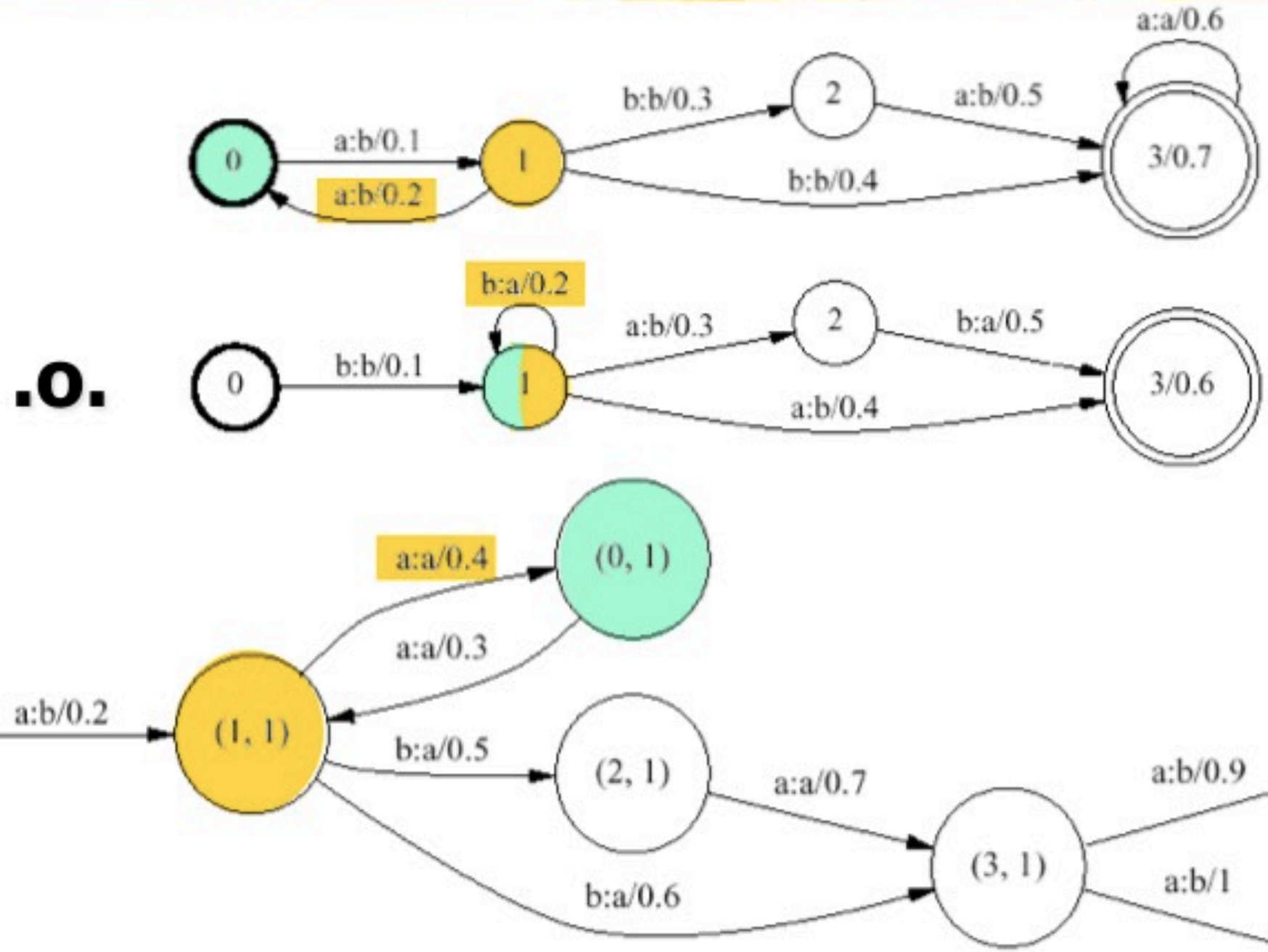
.o. =



a:b .o. b:b = a:b

they use (min, +), we use (+, *)

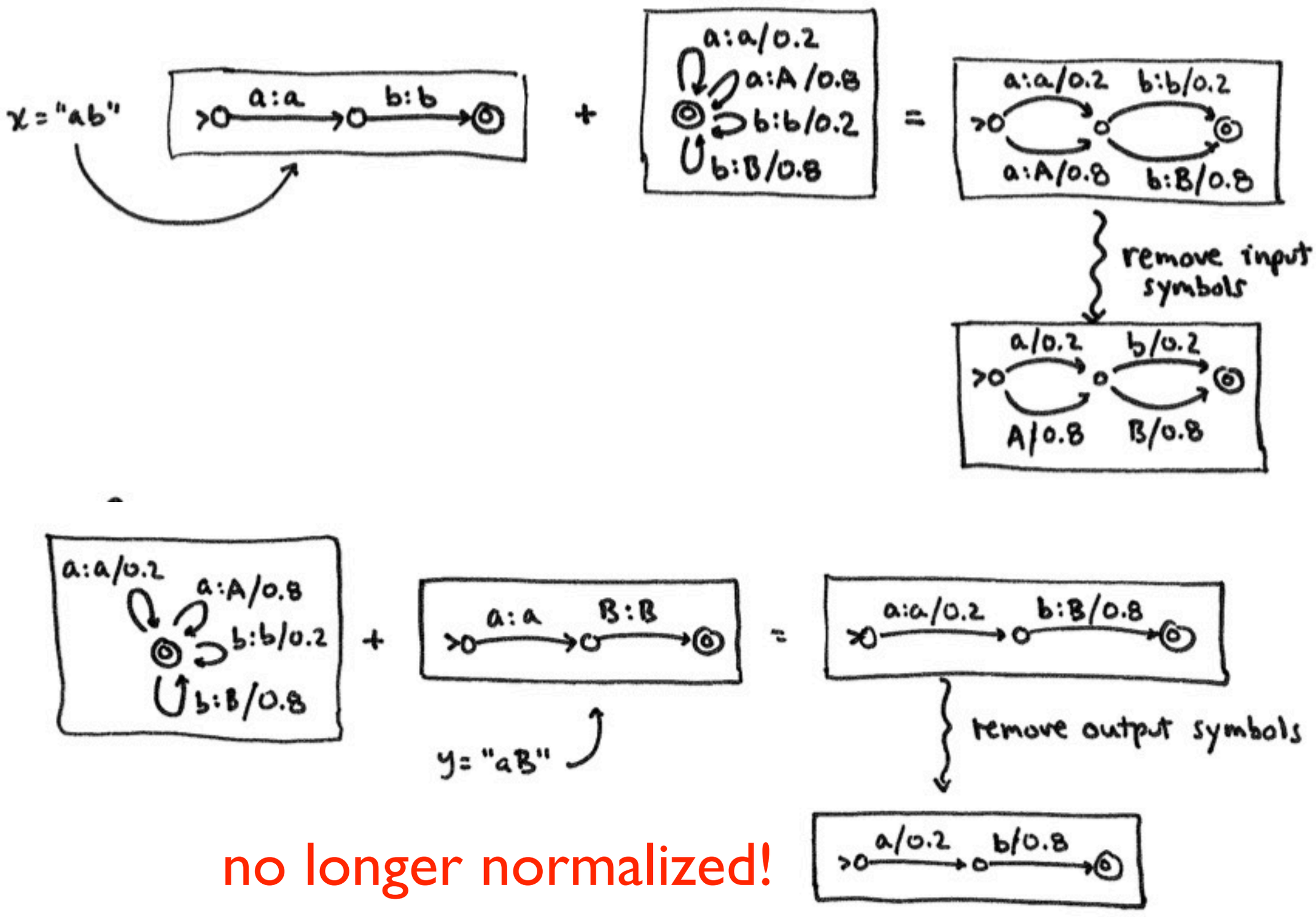
Example



a:b .o. b:a = a:a

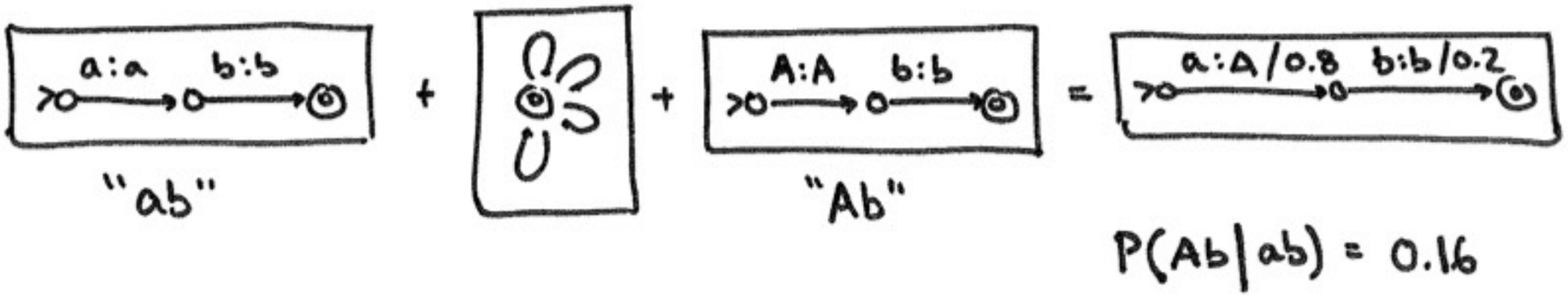
they use (min, +), we use (+, *)

Given x, supply all output y

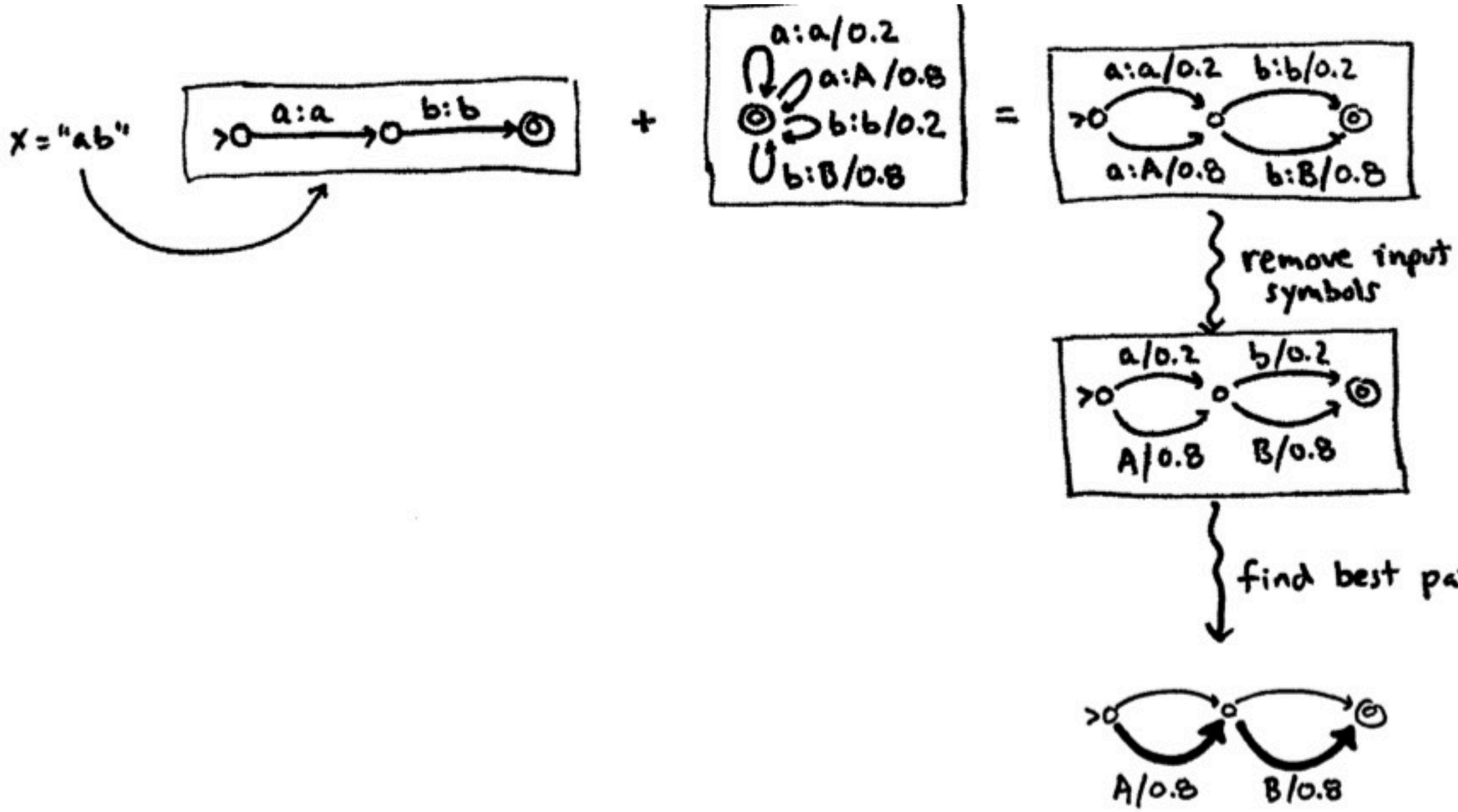


no longer normalized!

Given x, y, what's p(y|x)



Given x, what's max p(y|x)



so, $\underset{y}{\operatorname{argmax}}\ P(y|ab) = AB$

Part-of-Speech Tagging Again

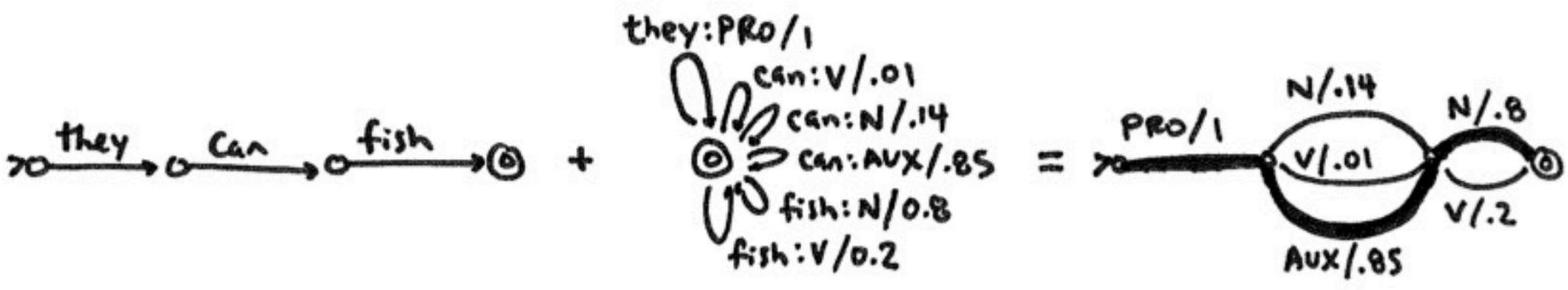
Before Probabilities

no preferences

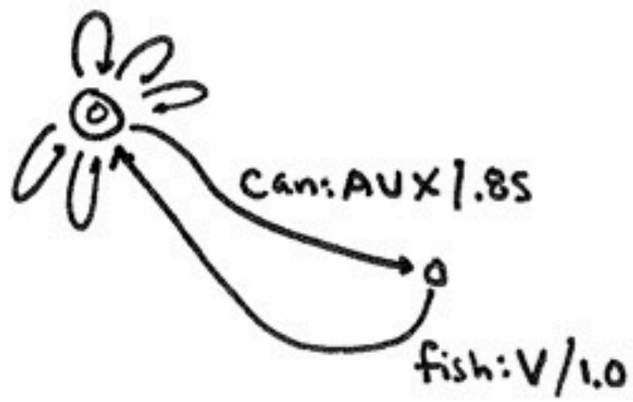
Part-of-Speech Tagging Again

Before Probabilities

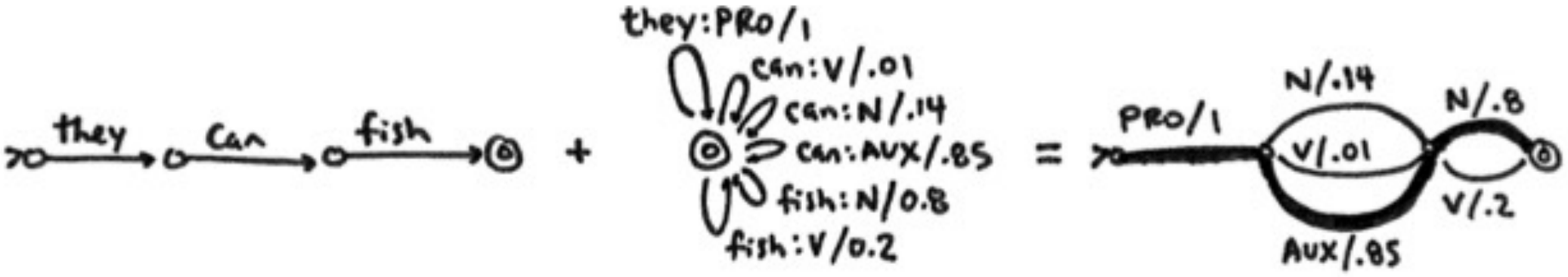
After Probabilities



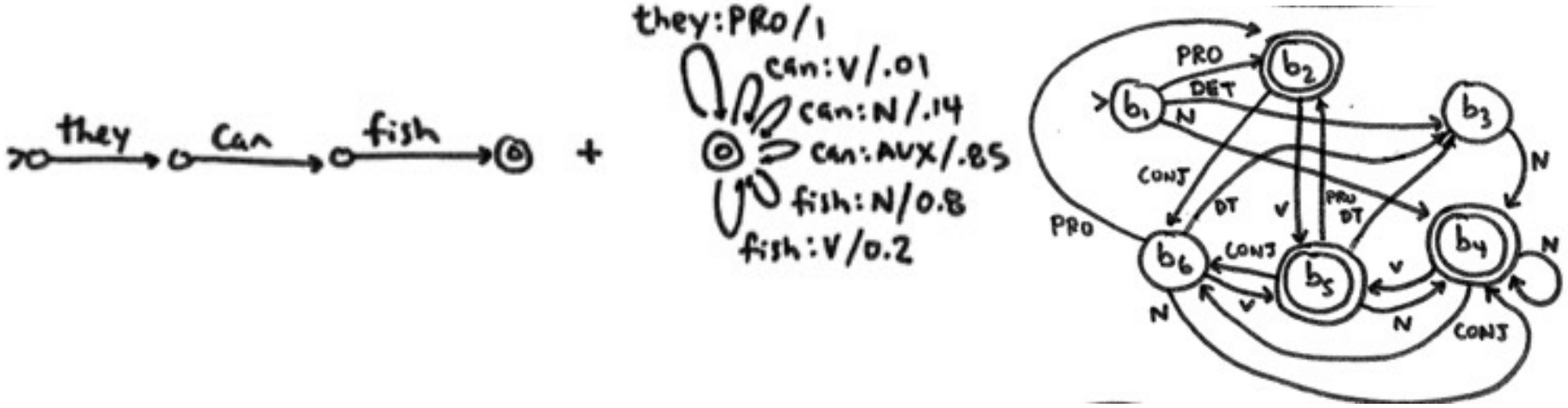
needs tweaking:



Adding a Tag Bigram Model (again)



FST C: POS bigram LM

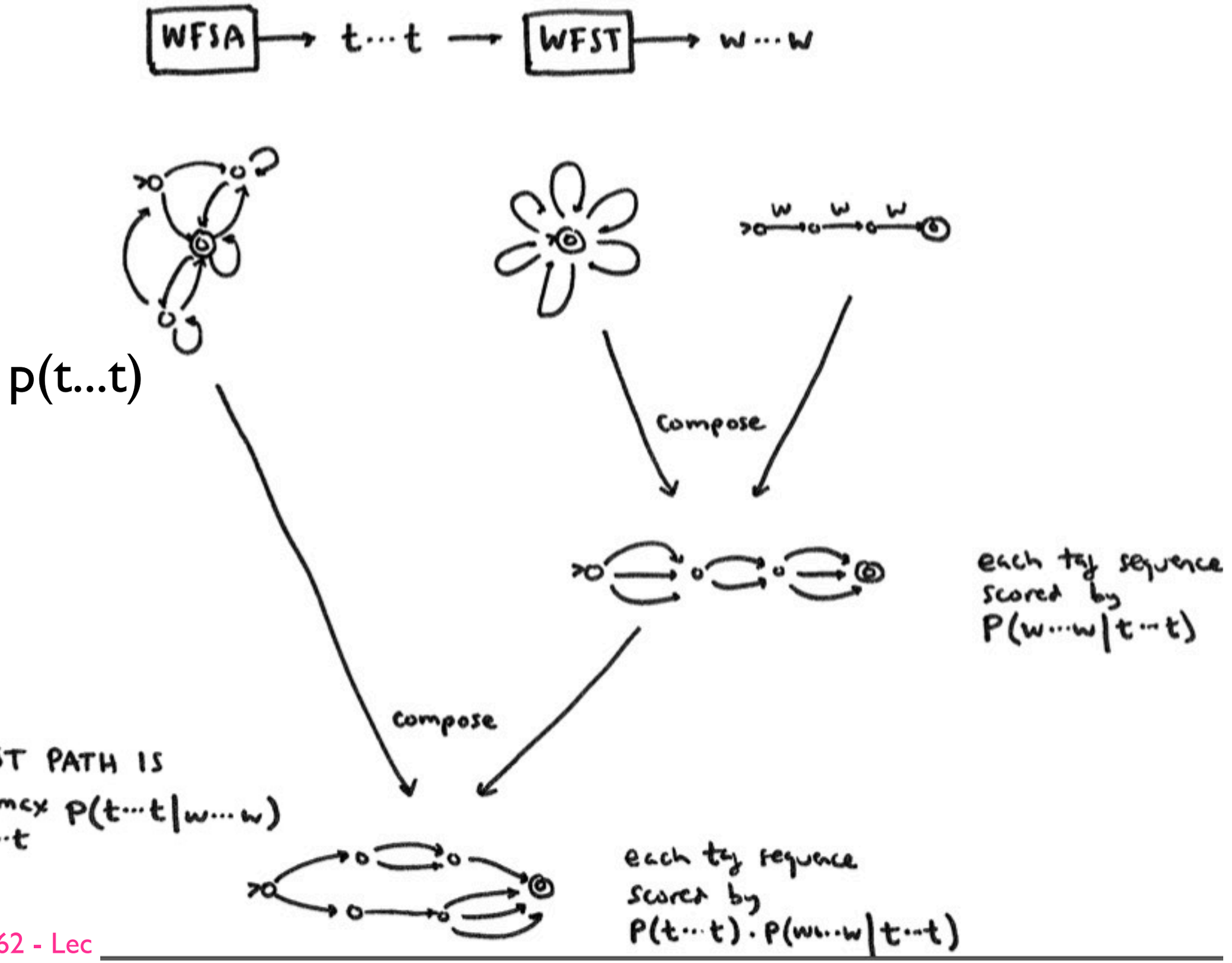


p(w...w) p(t...t|w...w) p(t...t) p(???)

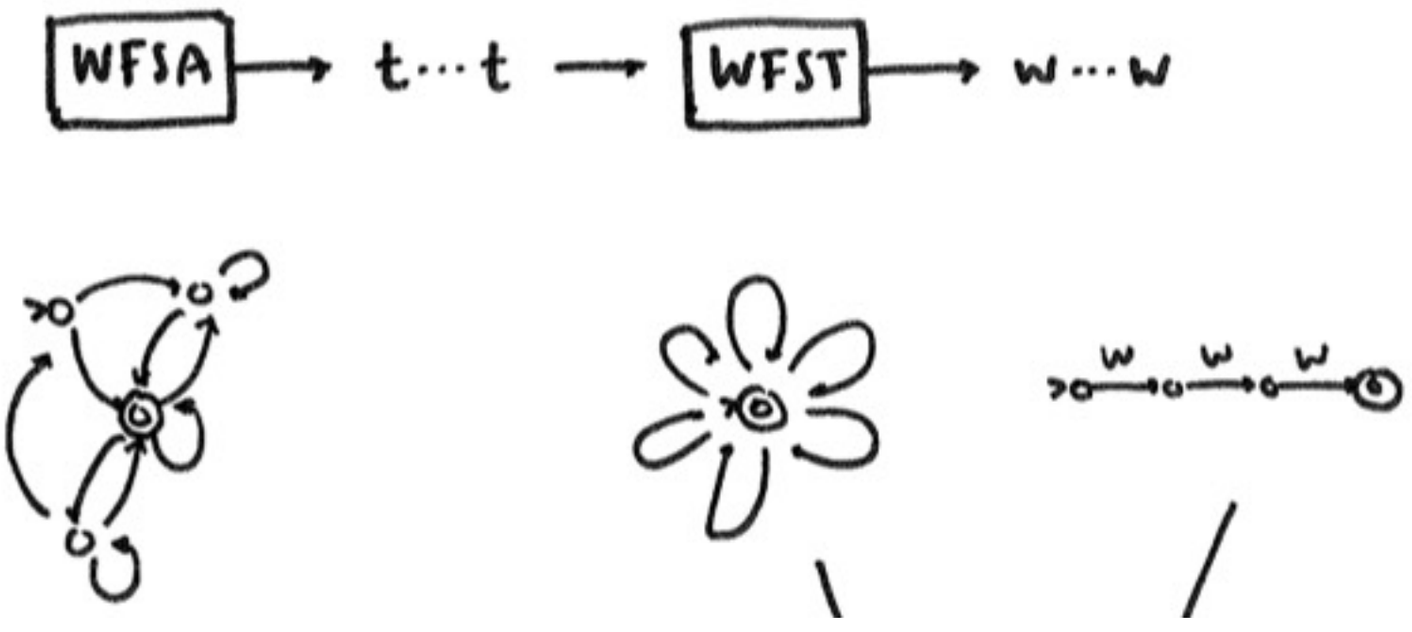
wait, is that right (mathematically)?

Noisy-Channel Model

Noisy-Channel Model



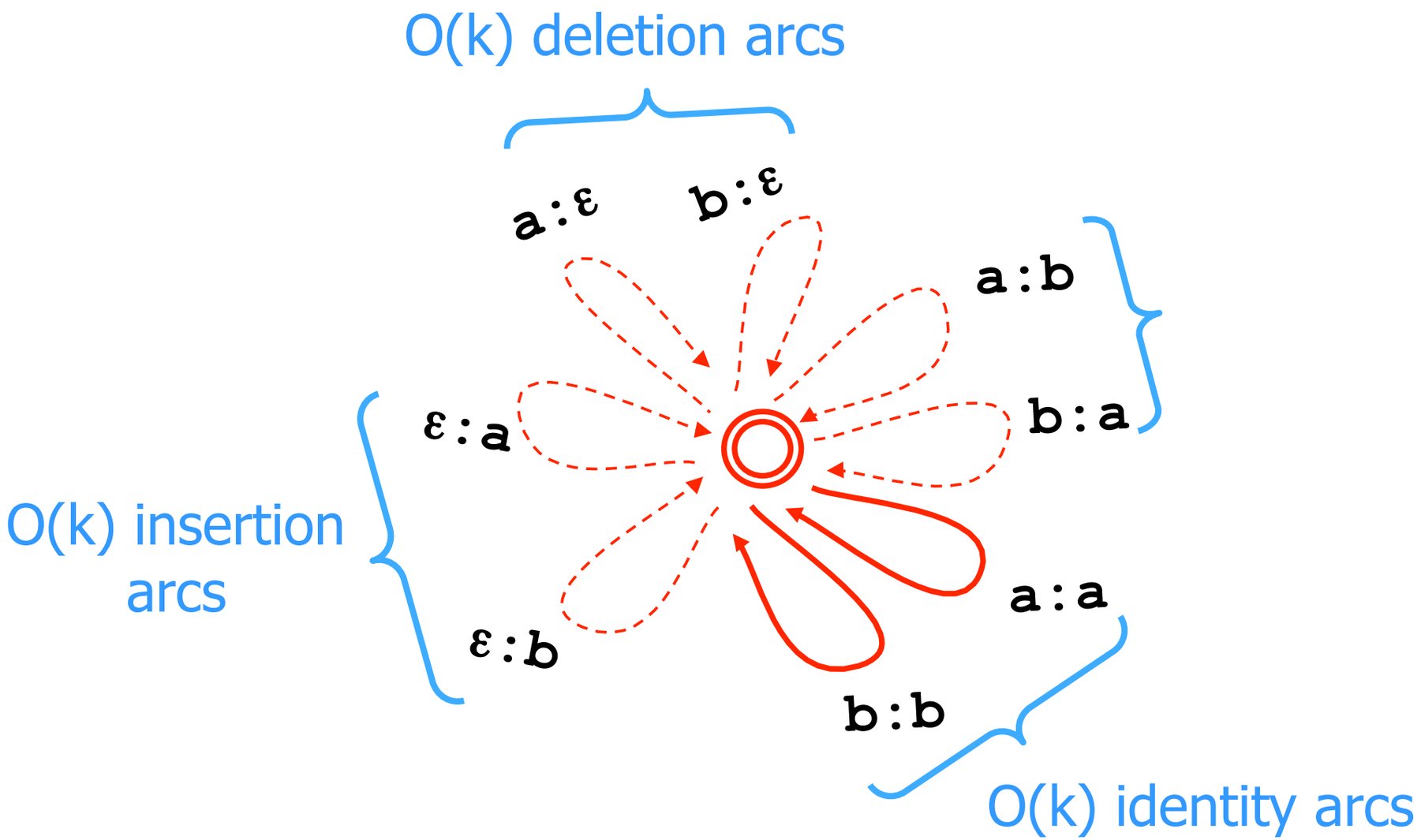
Applications of Noisy-Channel



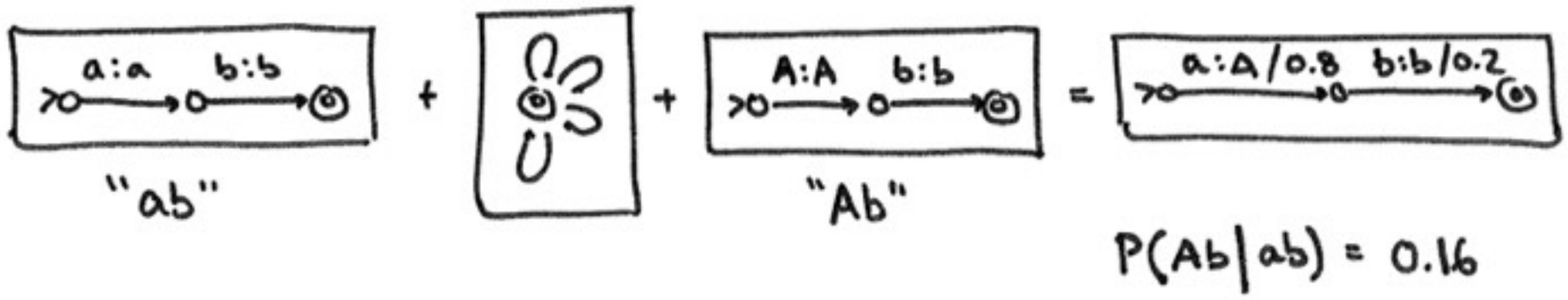
Application	Input	Output	p(i)	p($o|i$)
Machine Translation	L_1 word sequences	L_2 word sequences	p(L_1) in a language model	translation model
Optical Character Recognition (OCR)	actual text	text with mistakes	prob of language text	model of OCR errors
Part Of Speech (POS) tagging	POS tag sequences	English words	prob of POS sequences	p($w|t$)
Speech recognition	word sequences	speech signal	prob of word sequences	acoustic model

Example: Edit Distance

from J. Eisner

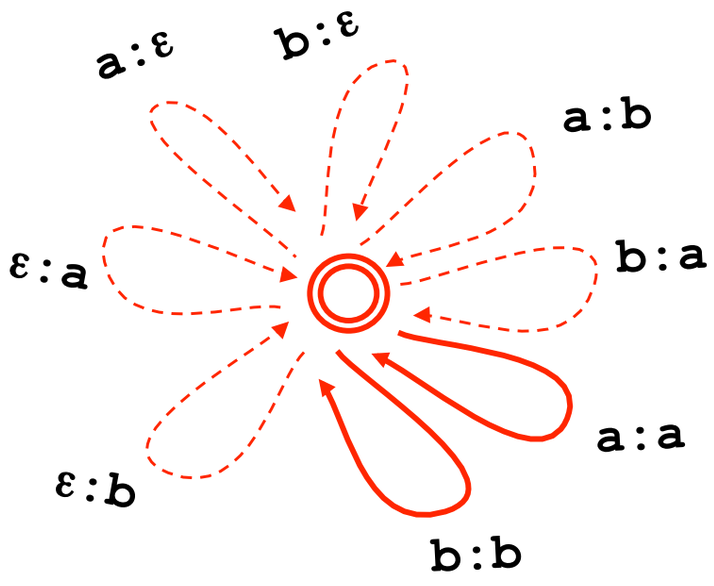


Example: Edit Distance



clara

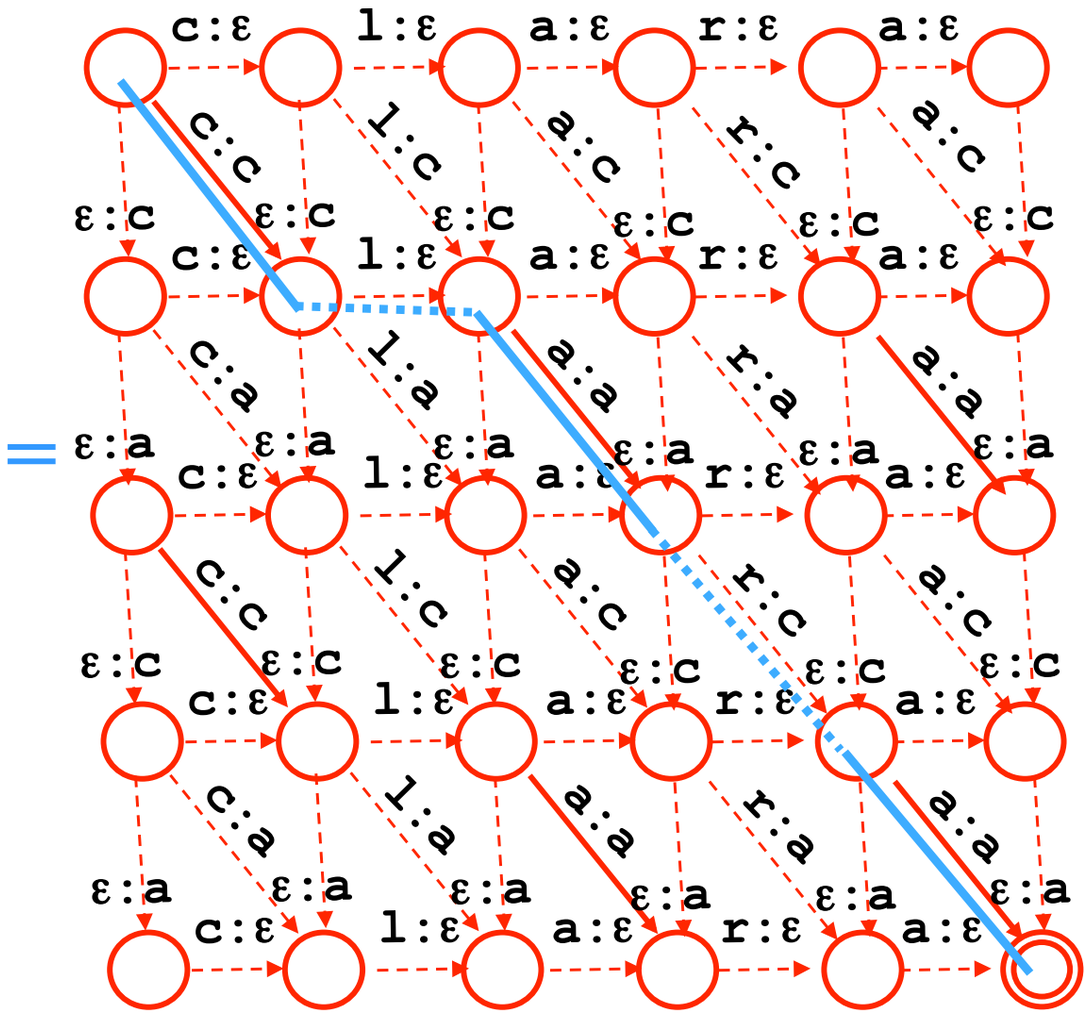
.0.



.0.

caca

Best path (by Dijkstra's algorithm)

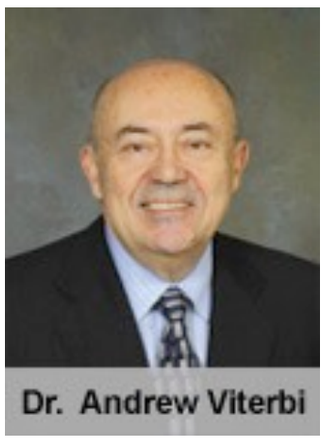


Max / Sum Probs

- in a WFSA, which string x has the greatest $p(x)$?
  - graph search (shortest path) problem
    - Dijkstra;
    - or Viterbi if the FSA is acyclic

Edsger Dijkstra
(1930-2002)
"GOTO considered harmful"

  - does it work for NFA?
    - best path much easier than best string
    - you can determinize it (with exponential cost!)
    - popular work-around: *n*-best list crunching



(b. 1932)
Viterbi Alg. (1967)
CMDA, Qualcomm

Dijkstra 1959 vs. Viterbi 1967

Numerische Mathematik 1, 269–271 (1959)

A Note on Two Problems in Connexion with Graphs

By

E. W. Dijkstra

We consider n points (nodes), some or all pairs of which are connected by a branch; the length of each branch is given. We restrict ourselves to the case where at least one path exists between any two nodes. We now consider two problems.

Problem 1. Construct the tree of minimum total length between the n nodes. (A tree is a graph with one and only one path between every two nodes.)

In the course of the construction that we present here, the branches are subdivided into three sets:

I. the branches definitely assigned to the tree under construction (they will form a subtree);

II. the branches from which the next branch to be added to set I, will be selected;

III. the remaining branches (rejected or not yet considered).

Edsger Dijkstra
(1930-2002)
"GOTO considered harmful"

that's min. spanning tree!

Jarnik (1930) - *Prim* (1957) - Dijkstra (1959)

Dijkstra 1959 vs. Viterbi 1967

that's shortest-path

Moore (1957) - *Dijkstra* (1959)

the data for at most n branches, viz. the branches in sets I and II and the branch under consideration in step 2.

Problem 2. Find the path of minimum total length between two given nodes P and Q.

We use the fact that, if R is a node on the minimal path from P to Q, knowledge of the latter implies the knowledge of the minimal path from P to R. In the solution presented, the minimal paths from P to the other nodes are constructed in order of increasing length until Q is reached.

In the course of the solution the nodes are subdivided into three sets:

A. the nodes for which the path of minimum length from P is known; nodes will be added to this set in order of increasing minimum path length from node P;

B. the nodes from which the next node to be added to set A will be selected; this set comprises all those nodes that are connected to at least one node of set A but do not yet belong to A themselves;

C. the remaining nodes.

Dijkstra 1959 vs. Viterbi 1967

IEEE TRANSACTIONS ON INFORMATION THEORY, VOL. IT-13, NO. 2, APRIL 1967

Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm

special case of dynamic programming (Bellman, 1957)

ANDREW J. VITERBI, SENIOR MEMBER, IEEE

Abstract—The probability of error in decoding an optimal convolutional code transmitted over a memoryless channel is bounded from above and below as a function of the constraint length of the code. For all but pathological channels the bounds are asymptotically (exponentially) tight for rates above R_0, the computational cutoff rate of sequential decoding. As a function of constraint length the performance of optimal convolutional codes is shown to be superior to that of block codes of the same length, the relative improvement increasing with rate. The upper bound is obtained for a specific probabilistic nonsequential decoding algorithm which is shown to be asymptotically optimum for rates above R_0 and whose performance bears certain similarities to that of sequential decoding algorithms.

Manuscript received May 20, 1966; revised November 14, 1966. The research for this work was sponsored by Applied Mathematics Division, Office of Aerospace Research, U. S. Air Force, Grant AFOSR-700-65.

The author is with the Department of Engineering, University of California, Los Angeles, Calif.

I. Summary of Results

SINCE Elias[1] first proposed the use of convolutional (tree) codes for the discrete memoryless channel, it has been conjectured that the performance of this class of codes is potentially superior to that of block codes of the same length. The first quantitative verification of this conjecture was due to Yudkin[2] who obtained

Sum Probs

- what is $p(x)$ for some particular x?
  - for DFA, just follow x
  - for NFA,
    - get a subgraph (by composition), then sum ??
      - acyclic => Viterbi
      - cyclic => compute strongly connected components
        - SCC-DAG cluster graph (cyclic locally, acyclic globally)
        - do infinite sum (matrix inversion) locally, Viterbi globally
        - *refer to extra readings on course website*