Weighted Finite State Automata

Bill Byrne

Lent 2022

Neural Machine Translation and Dialogue Systems

MPhil in Machine Learning and Machine Intelligence

Cambridge University Engineering Department

v2022–02–28

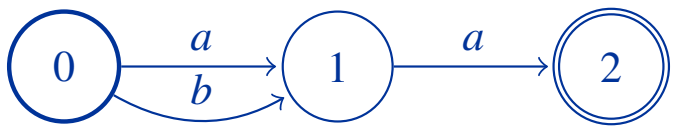
Formal Languages[1]

A formal language is a set of strings, each string composed of symbols drawn from a set Σ called the *alphabet* or the *vocabulary*.

A model that can *generate* and *recognize* all and only the strings of a formal language acts as the definition of the formal language.

An automata can be thought of as either an *acceptor* or a *generator*.

For $\Sigma = \{a, b\}$ this Finite State Automaton defines a language $L = \{aa, ba\}$



- Acceptor: only aa or ba lead from the start state to the final state
- Generator: valid paths generate either aa or ba

[1] Aho and Ullman. "The Theory of Parsing, Translation, and Compiling" 1972

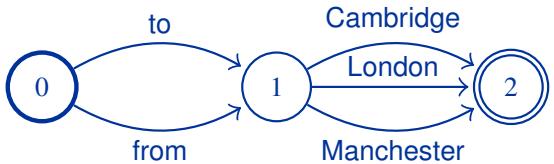
Finite Languages

Languages that have a finite vocabulary and strings of a fixed maximum length are finite

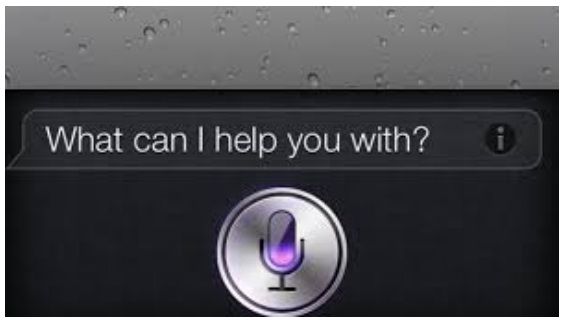
- It is possible to enumerate every sentence in a finite language (eventually)

Finite languages can arise in a variety of ways:

- By design:



- Through practical constraints:
  - Suppose Siri will process speech only in 10 sec chunks...
  - and that Siri has a 50K word vocabulary...
  - and that a person speaks at most at 2 words per second...
  - then an ASR system needs (at most) a language of 50000^{20} strings
  - Too big to list, but still finite



Weighted Finite State Acceptors - Definition

A WFSA is a directed graph specified as $M = (Q, \Sigma, E, q_0, F, \rho)$

An edge $e \in E$ now has a weight $w(e)$: $s(e) \xrightarrow{i(e)/w(e)} f(e)$

- Final states $f \in F$ also have weights $\rho(f)$
- The weights take values in $\mathbb{K}$

A complete path through an acceptor can be written as $p = e_1 \cdots e_{n_p}$, where :
- the path p has n_p edges
- the path starts at state $i_p = s(e_1)$ where i_p is an initial state
- the path ends at state $f_p = f(e_{n_p})$ where f_p is a final state

The arc weights and initial and final weights combine to form the path weight

$$w(p) = w(e_1) \otimes \cdots \otimes w(e_{n_p}) \otimes \rho(f_p)$$

$\otimes$ is the product of two weights (to be defined shortly)

Notation: $\otimes_{j=1}^{n_p} w(e_j) = w(e_1) \otimes \cdots \otimes w(e_{n_p})$ so that $w(p) = (\otimes_{j=1}^{n_p} w(e_j)) \otimes \rho(f_p)$

Weights Assigned to Strings by Acceptors

It is straightforward to associate paths through the acceptor with input sequences.
- A path $p = e_1 \cdots e_{n_p}$ produces the string $x = i(e_1) \cdots i(e_{n_p})$

If a string was generated by a single path, its weight would simply be the path weight.

However, since strings can be generated by multiple paths, the acceptor combines the weights of all paths which might have generated a string, as follows:

- Let x be a string constructed from symbols in the input alphabet Σ : $x \in \Sigma^*$
- Let $P(x)$ be the set of complete paths which generate x, i.e. $x = i(e_1) \cdots i(e_{n_p})$ and $s(e_1) = q_0$
- Let $\oplus$ be the *sum* of two weight values
- Define $[\![A]\!](x)$ as the cost assigned to the string x by the acceptor

$$
\begin{aligned}
[\![A]\!](x) &= \bigoplus_{p \in P(x)} (\otimes_{j=1}^{n_p} w(e_j)) \otimes \rho(f_p)) \\
&= \bigoplus_{p \in P(x)} w(p)
\end{aligned}
$$

$[\![A]\!](x)$ is the 'Sum' of the weights of the complete paths in A which can generate x

Weights and Operations on Weights

The *product* operation $\otimes$ computes the weight of a single path from the weights of its edges

The *sum* operation $\oplus$ computes the weight of a sequence by summing over all the distinct paths which could have generated that sequence

Semirings : sum $\oplus$ and product $\otimes$ with identity elements $\bar{0}$ and $\bar{1}$
- For a weight $k \in \mathbb{K}$: $\bar{0} \oplus k = k$; $\bar{1} \otimes k = k$; $\bar{0} \otimes k = \bar{0}$
- $\oplus$ and $\otimes$ distribute and commute in the familiar way

Three useful semirings:

Semiring	$\mathbb{K}$	$\oplus$	$\otimes$	$\bar{0}$	$\bar{1}$
Probability	$\mathbb{R}_+$	$+$	$\times$	0	1
Log	$\mathbb{R} \cup \{-\infty, \infty\}$	$\oplus_{\log}$	$+$	∞	0
Tropical	$\mathbb{R} \cup \{-\infty, \infty\}$	$\min$	$+$	∞	0

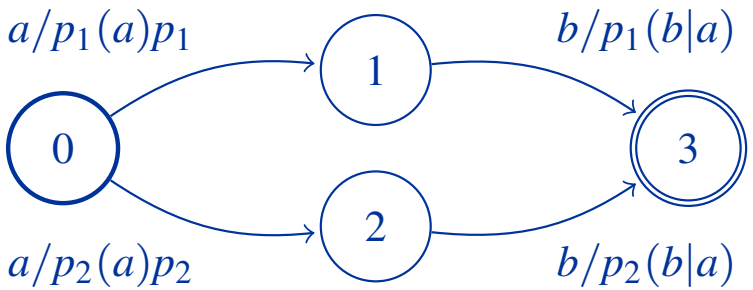
$$\oplus_{\log} : k_1 \oplus_{\log} k_2 = -\log(e^{-k_1} + e^{-k_2})$$

Unless otherwise stated, the tropical semiring is used by default

Note: if a string x is not accepted by A, then $[[A]](x) = \bar{0}$

Example - Weights Under the Probability Semiring

Find the weight assigned to the string ‘a b’ by the following acceptor:

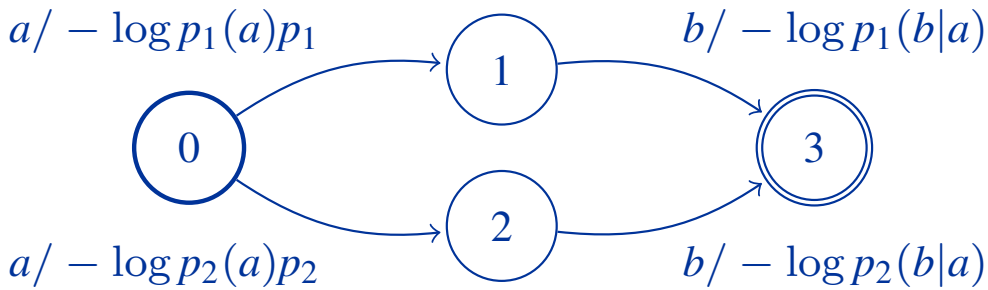


- Operations on weights via ‘usual’ multiplication and addition: $(\oplus, \otimes) = (+, \times)$

$$
\begin{aligned}
[\![A]\!](\text{‘a b’}) &= \underbrace{w(e_1) \otimes w(e_2)}_{\text{top path}} \oplus \underbrace{w(e_3) \otimes w(e_4)}_{\text{bottom path}} \\
&= p_1(a)p_1 \times p_1(b|a) + p_2(a)p_2 \times p_2(b|a) \\
&= p_1(\text{‘a b’})\, p_1 + p_2(\text{‘a b’})\, p_2 \Leftarrow \text{marginal probability}
\end{aligned}
$$

Example - Weights Under the Log Semiring

Find the weight assigned to the string ‘a b’ by the following acceptor:
- weights are negative log likelihoods



Weight operations: $(\oplus, \otimes) = (\oplus_{\log}, +)$ where $\oplus_{\log}$: $k_1 \oplus_{\log} k_2 = -\log(e^{-k_1} + e^{-k_2})$

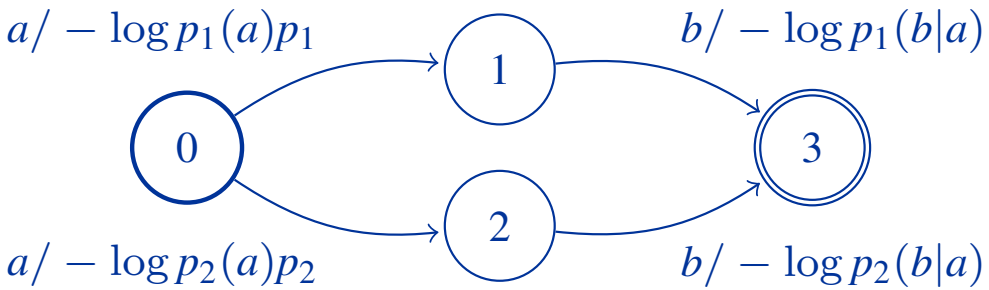
How $\oplus_{\log}$ operates on negative log probs:

$$(-\log p_1) \oplus_{\log} (-\log p_2) = -\log(e^{-(-\log p_1)} + e^{-(-\log p_2)}) = -\log(p_1 + p_2)$$

$$\begin{aligned}
[\![A]\!](\text{‘a b’}) &= w(e_1) \otimes w(e_2) \oplus w(e_3) \otimes w(e_4) \\
&= [-\log p_1(a)p_1 - \log p_1(b|a)] \oplus_{\log} [-\log p_2(a)p_2 - \log p_2(b|a)] \\
&= [-\log p_1(\text{‘a b’})p_1] \oplus_{\log} [-\log p_2(\text{‘a b’})p_2] \\
&= -\log[\exp(\log p_1(\text{‘a b’})p_1) + \exp(\log p_2(\text{‘a b’})p_2)] \\
&= -\log[\,p_1(\text{‘a b’})\,p_1 + p_2(\text{‘a b’})p_2\,] \Leftarrow \text{negative log marginal prob}
\end{aligned}$$

Example - Weights Under the Tropical Semiring

Find the weight assigned to the string ‘a b’ by the following acceptor:
- weights are negative log likelihoods



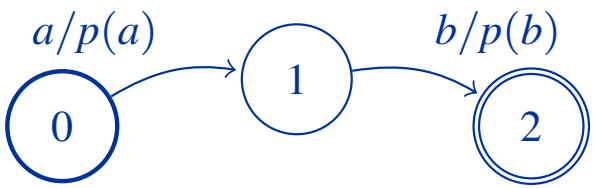
Weight operations: $(\oplus, \otimes) = (\min, +)$, where $\min$ is the minimum, i.e. $\min(k_1, k_2)$

$$
\begin{aligned}
[\![A]\!](\text{‘a b’}) &= w(e_1) \otimes w(e_2) \oplus w(e_3) \otimes w(e_4) \\
&= \min[-\log p_1(\text{‘a b’})\, p_1 \,,\, -\log p_2(\text{‘a b’})\, p_2] \\
&= \underbrace{-\max\left[\log\, p_1(\text{‘a b’})\, p_1 \,,\, \log\, p_2(\text{‘a b’})\, p_2\right]}_{\text{negative log Viterbi likelihood}}
\end{aligned}
$$

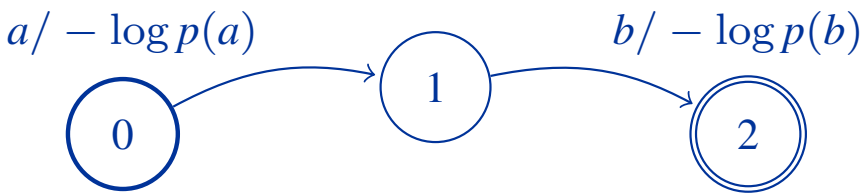
Tropicalization [2]

Tropicalization means replacing the arithmetic operations $(+, \times)$ by the operations $(\min, +)$.

This process captures the essence of what happens when the joint probabilities are replaced by their (negative) logarithms.



$A[[\text{'a b'}]] = p(a)p(b)$



$B[[\text{'a b'}]] = -\log(\, p(a)p(b)\,)$

B is the 'tropicalized' version of the machine A:

- B is derived from A simply by replacing probabilities on arcs by their negative log values
- The process is consistent (i.e. invertible) for arc weights and path weights

[2] *Tropical Geometry of Statistical Models*, Pachter and Sturmfels, 2004
http://www.pnas.org/content/101/46/16132.short

WFSA Operations

Basic operations can be performed over WFSAs

Some operations affect the languages defined by WFSAs :

- Intersection
- Union
- Concatenation (or Product)
- ...

Other operations correspond to operations on the WFSA itself :

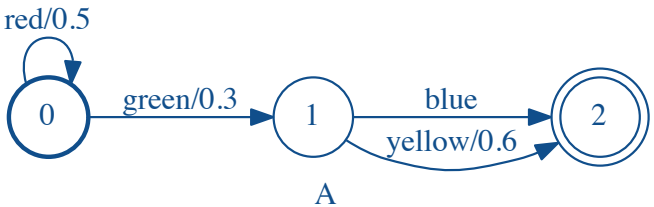
- Determinization
- Shortest distance calculations
- ...

The tropical semiring will be used by default in what follows

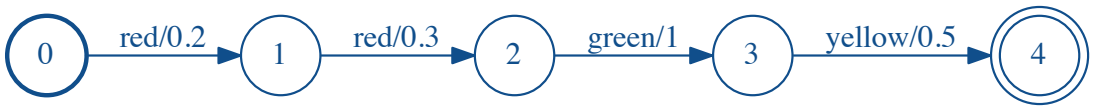
WFSA Operations - Intersection

A string x is accepted by $C = A \cap B$ if x is accepted by A and by B

$$[\![C]\!](x) = [\![A]\!](x) \otimes [\![B]\!](x)$$



A



B

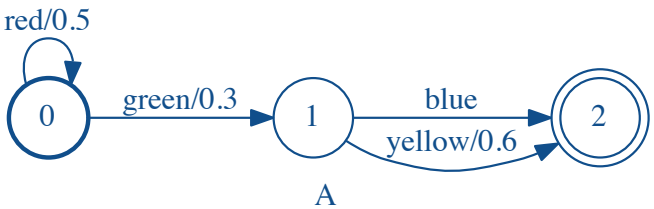
$C = A \cap B$

In this example $x =$ 'red red green yellow' and $(\oplus, \otimes) = (\min, +)$.
Verify that $[\![A \cap B]\!](x) = [\![C]\!](x)$:
$[\![A]\!](x) = 0.5 + 0.5 + 0.3 + 0.0 = 1.3$
$[\![B]\!](x) = 0.2 + 0.3 + 1 + 0.5 = 2.0$
$[\![C]\!](x) = 0.7 + 0.8 + 1.3 + 0.5 = 3.3$
$[\![A \cap B]\!](x) = [\![A]\!](x) \otimes [\![B]\!](x) = [\![A]\!](x) + [\![B]\!](x) = 1.3 + 2.0 = 3.3$

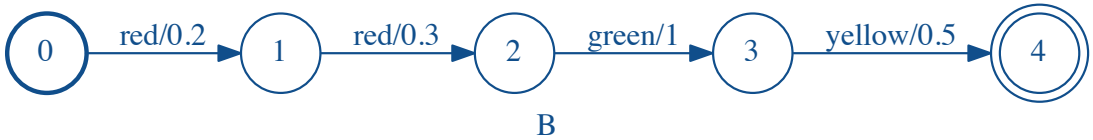
WFSA Operations - Union

A string x is accepted by $C = A \cup B$ if x is accepted by A or by B

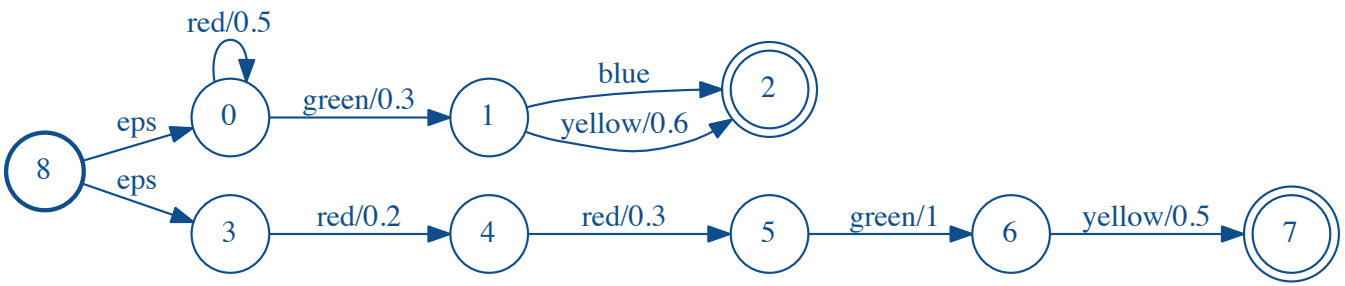
$$[\![C]\!](x) = [\![A]\!](x) \oplus [\![B]\!](x)$$



A



B

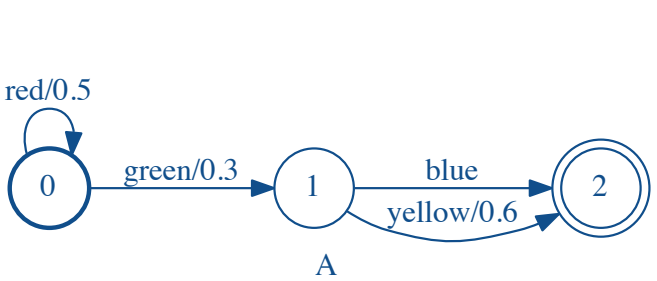


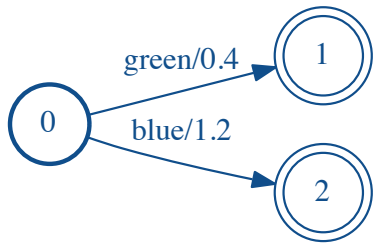
$C = A \cup B$

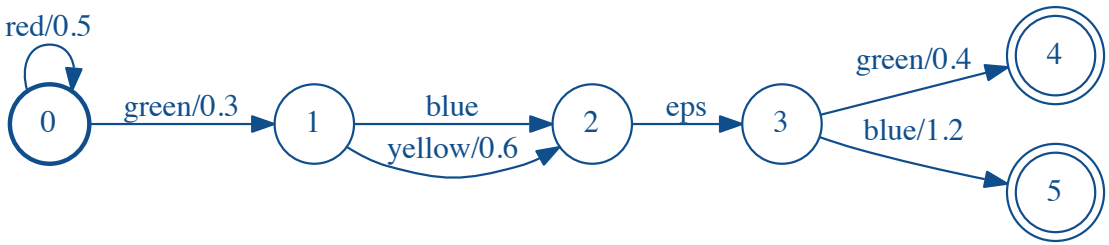
WFSA Operations - Concatenation (or Product)

A string x is accepted by $C = A \otimes B$ if x can be split into $x = x_1 x_2$ such that x_1 is accepted by A and x_2 is accepted by B

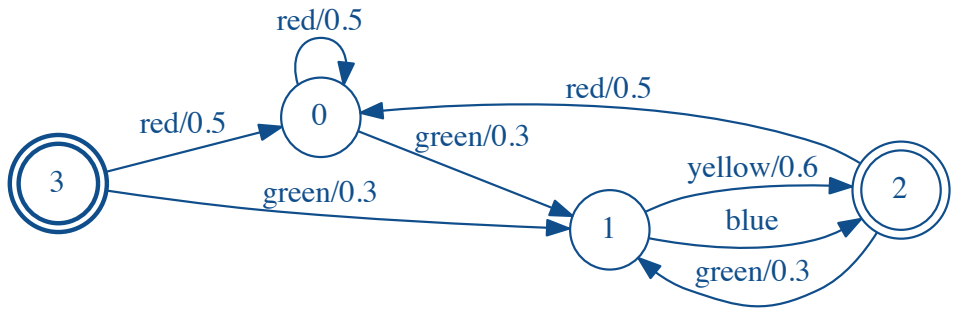
$$[\![C]\!](x) = \bigoplus_{x_1, x_2 : x = x_1 x_2} [\![A]\!](x_1) \otimes [\![B]\!](x_2)$$





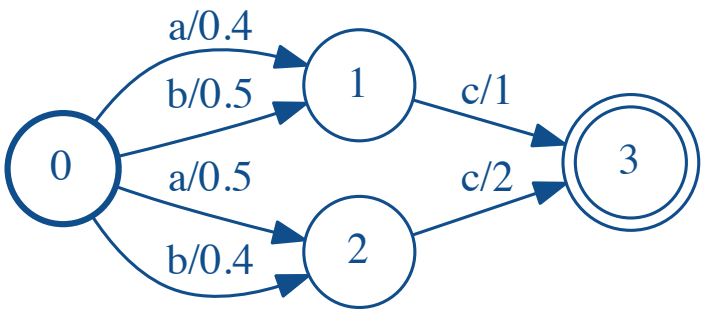


$C = A \otimes B$

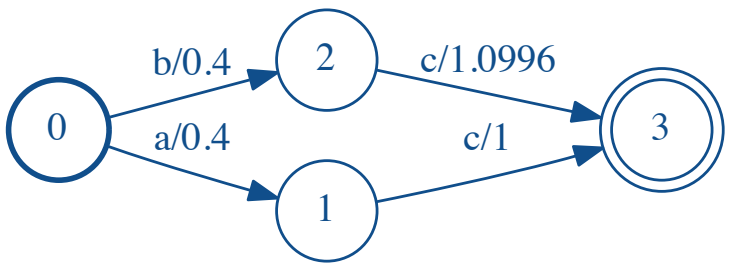


WFSA Operations - Determinization

Some WFSAs (in some semirings) can be **determinized**. After determinization:
- there is a unique starting state
- no two transitions leaving a state share the same input label
- arc weights may change, but weights assigned to strings are unchanged
- there may be many new epsilon arcs



Before Determinization

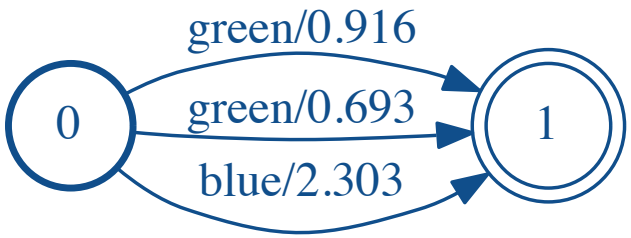


After Determinization

- determinization can be followed by **minimization** which finds an equivalent machine with a minimal number of states and arcs

Determinization is done with respect to the semiring

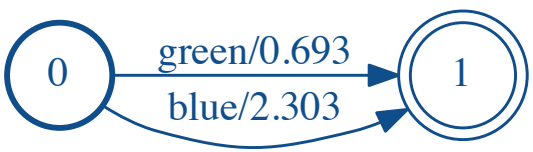
Determinization Under the Tropical and Log Semirings



Weights are negative log probs

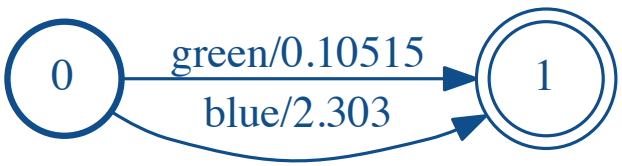
$$-\log 0.4 = 0.9163, -\log 0.5 = 693, -\log 0.1 = 2.3025$$

In the determinized machine, there will be one edge for **green** with weight $0.916 \oplus 0.693$



Determinization in the tropical semiring

$$0.916 \oplus 0.693 = \min(0.916, 0.693)$$



Determinization in the log semiring

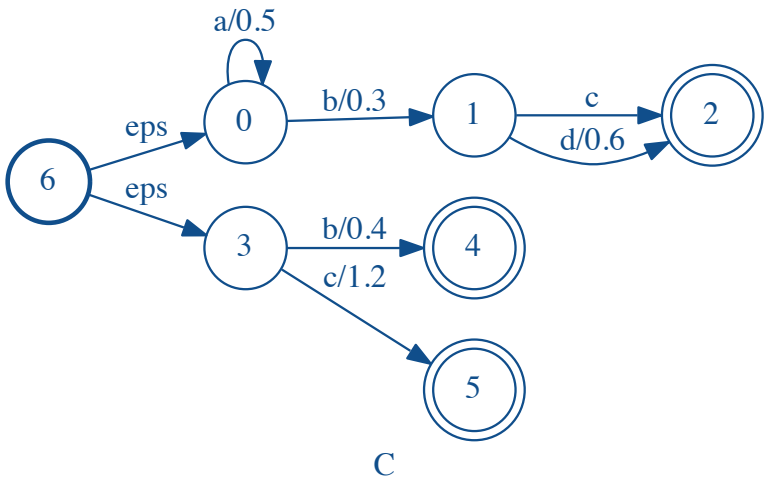
$$\begin{aligned} 0.916 \oplus 0.693 &= \oplus_{\log}(0.916, 0.693) \\ &= -\log(0.4 + 0.5) \\ &= -\log 0.9 = 0.105 \end{aligned}$$

WFSA Operations - Shortest Distance and Shortest Path Algorithms

Let $P(q, F)$ be the set of paths from any state q to any final state in F
- $d[q]$ is the sum of the weights of all paths from q to any final state in F

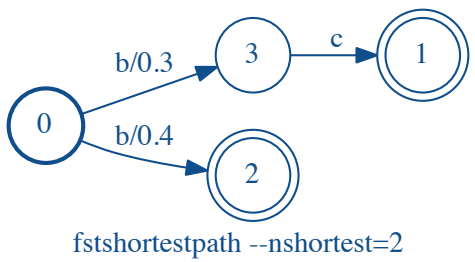
$$d[q] = \bigoplus_{p \in P(q,F)} w(p)$$

- costs $d[q]$ can be computed efficiently, and trace-back can reconstruct shortest-distance paths




```
> fstshortestdistance --reverse C.fst
0 0.30
1 0
2 0
3 0.40
4 0
5 0
6 0.30
```

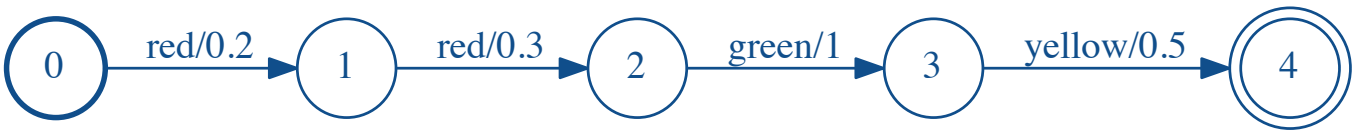
For semirings with the path property, it is also possible to compute shortest paths.
These are generated as a WFSA.



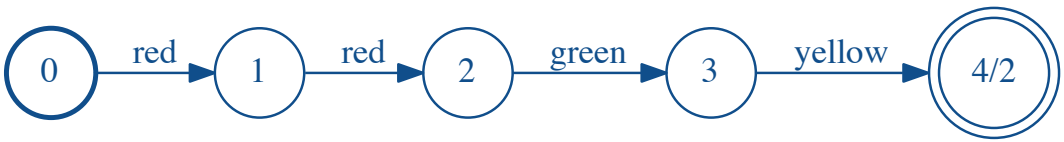
Pushing

Arc weights and labels can be moved if weights assigned to strings are preserved.

Pushing moves weights and/or labels towards the start or the end state
- pushing towards the start state can improve pruning
- pushing towards the end states can help accumulating costs over paths



B



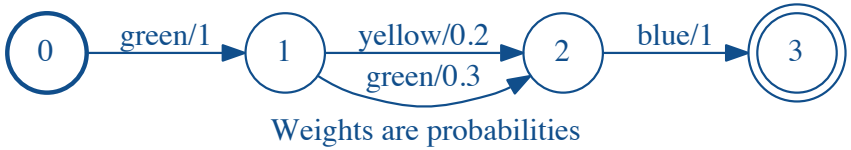
Weight pushing – Tropical Semiring

Pushing in the Log Semiring

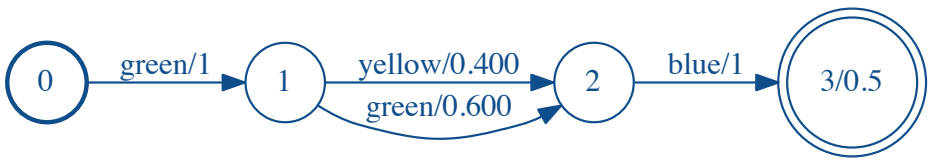
In pushing weights towards the final state:

- Final state has sum of all path weights, i.e. the forward probability
- Arc weights have posterior probabilities
- Pushing makes the WFSA stochastic, in its semiring

Real Semiring



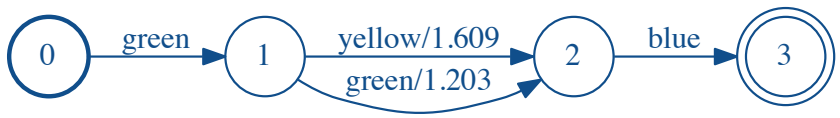
Weights are probabilities



Weight Pushing -- Real Semiring

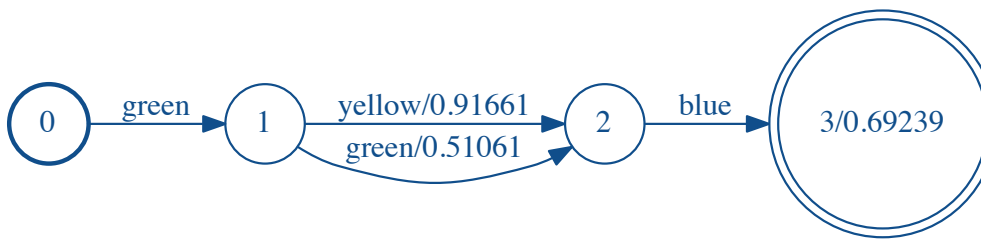
N.B. OpenFst does not support the real semiring.

Log Semiring



Weights are negative log probs

$$\log 0.3 = -1.203, \log 0.2 = -1.609$$



Weight Pushing -- Log Semiring

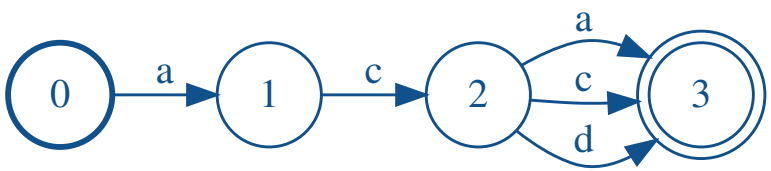
$$\log 0.6 = -0.511, \log 0.4 = -0.916,$$
$$\log 0.5 = -0.693$$

Failure Transitions – Special Symbols

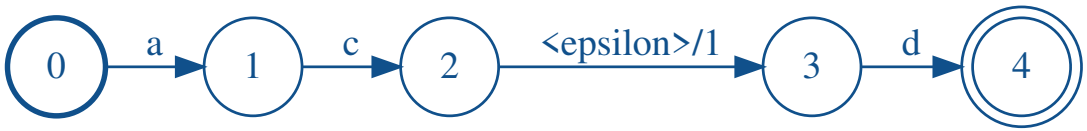
OpenFST provides *special symbols* used in matching symbols in composition and intersection.

	Consumes no symbol	Consumes symbol
Matches all	ϵ	σ
Matches 'rest'	ϕ	ρ

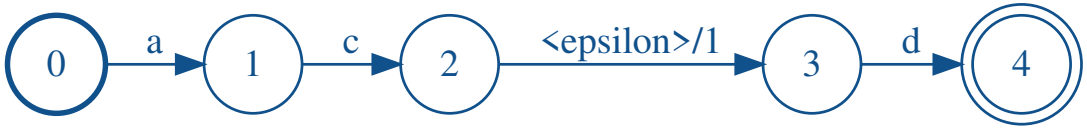
ϵ Arcs



A



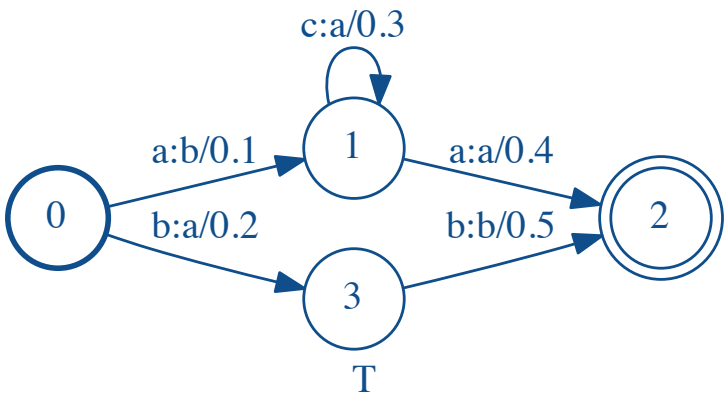
B



$A \circ B$

Weighted Finite State Transducers

WFSTs can be used to transform one string to another string
- this is done via symbol-to-symbol mappings
- arcs are modified to have an 'output' symbol
- the interpretation is 'read a symbol x , write a symbol y'
- weights are applied analogously to weighted acceptors



Input String x	Output String y	Cost $[\![T]\!](x, y)$
'b b'	'a b'	0.7
'a a'	'b a'	0.5
'a c a'	'b a a'	0.8
'a c c a'	'b a a a'	1.1
'a c c c a'	'b a a a a'	1.4
⋮	⋮	

In a weighted transducer, arcs have the form:

$$q \overset{x:y/k}{\rightarrow} q'$$

- e.g. the WFST T has an arc with $q = 0$, $q' = 3$, $x = b$, $y = a$, $k = 0.2$

Weighted Finite State Transducer – Definition

A WFST is a directed graph specified as $M = (Q, \Sigma, \Delta, E, q_0, F)$

The definition of the acceptor is extended to support output operations

- Two alphabets: an input alphabet: Σ and an output alphabet: Δ
- Each arc (edge) e has an input symbol $i(e) \in \Sigma$ and an output symbol $o(e) \in \Delta$

$$e : \; s(e) \xrightarrow{i(e):o(e)/w(e)} f(e)$$

For strings $x \in \Sigma^*$ and $y \in \Delta^*$, define $P(x, y)$ to be the set of all complete paths $p = e_1 \cdots e_{n_p}$ which have x as an input sequence and y as an output sequence

$$p \in P(x, y) : x = i(e_1) \cdots i(e_{n_p}) \,, \; y = o(e_1) \cdots o(e_{n_p})$$

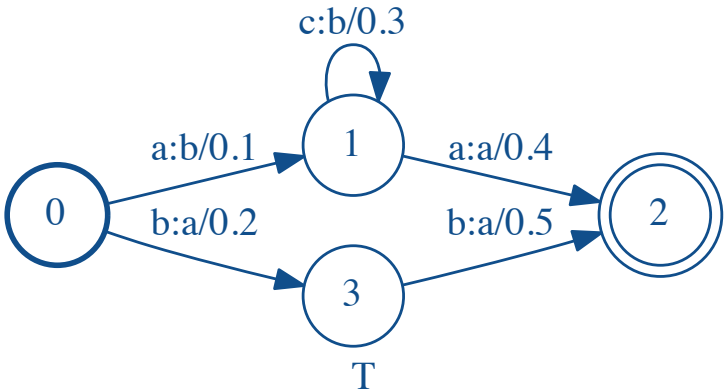
- Path weights are computed as in acceptors: $w(p) = \otimes_{j=1}^{n_p} w(e_j) \otimes \rho(f_p)$

The transducer T implements a **weighted mapping** of string x to string y :
- the weight is the sum of all path weights along which x is mapped to y

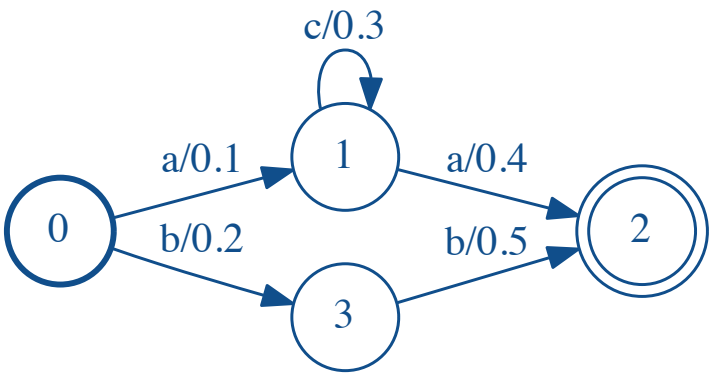
$$[\![T]\!](x, y) = \bigoplus_{p \in P(x,y)} w(p)$$

WFST Operations – Projection

Transform a WFST to a WFSA by projecting onto the input arcs or the output arcs.

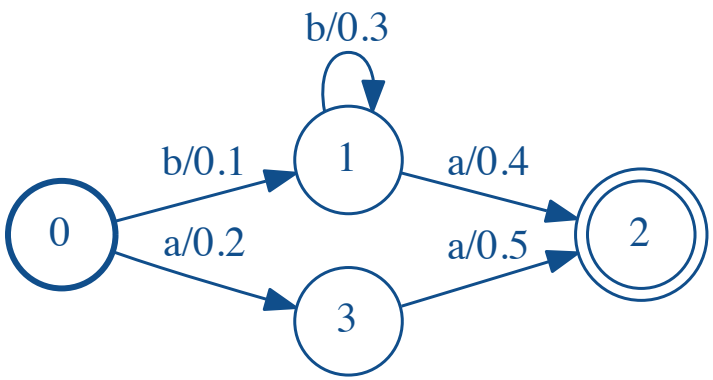


T_1 – Projection of T on its input



$$[\![T_1]\!](x) = \bigoplus_y [\![T]\!](x, y)$$

T_2 – Projection of T on its output



$$[\![T_2]\!](y) = \bigoplus_x [\![T]\!](x, y)$$

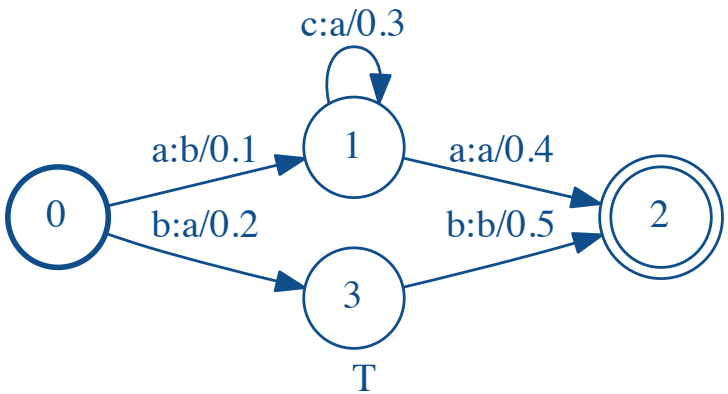
L_{T_1} is the input language and L_{T_2} is the output language

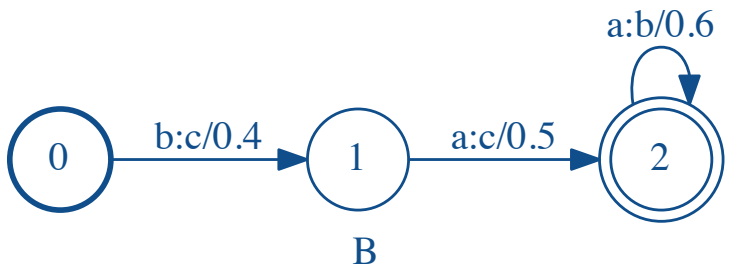
WFST Operations – Composition

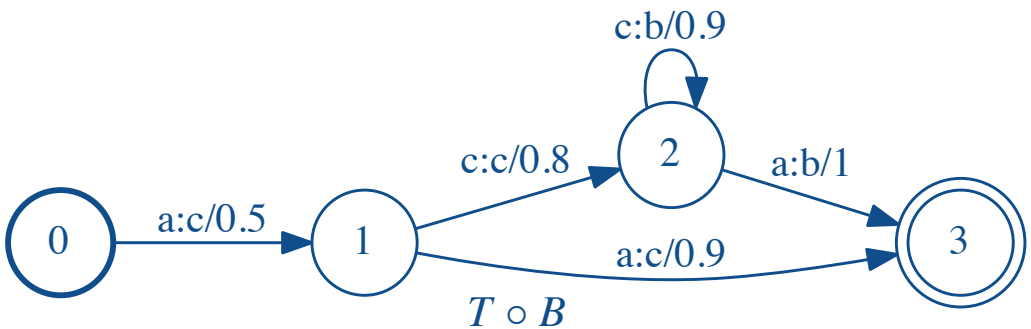
Suppose A and B are two WFSTs: A maps x to y ; B maps y to z.

$A \circ B$ is the composition of A with B which maps x to z

$$[\![A \circ B]\!](x,z) = \bigoplus_y [\![A]\!](x,y) \otimes [\![B]\!](y,z)$$





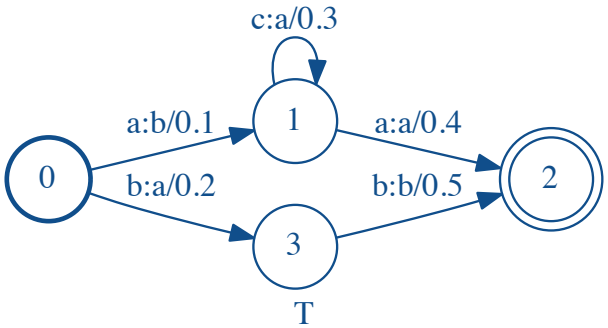


$$x \in L_{T_1} \Rightarrow y \in L_{T_2} \cap L_{B_1} \Rightarrow z \in L_{B_2}$$

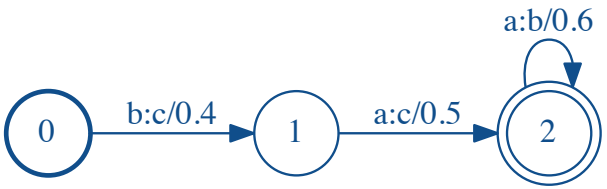
WFST Operations – Union

Suppose A and B are two WFSTs: $A \cup B$ is the union of A with B

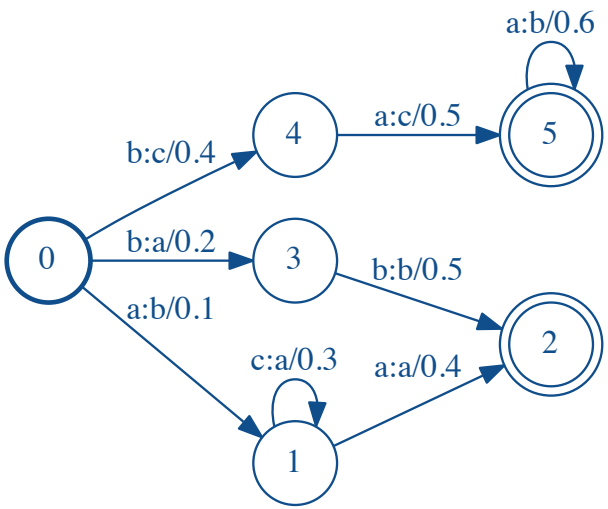
$$[\![A \oplus B]\!](x, z) = [\![A]\!](x, z) \oplus [\![B]\!](x, z)$$



T



B

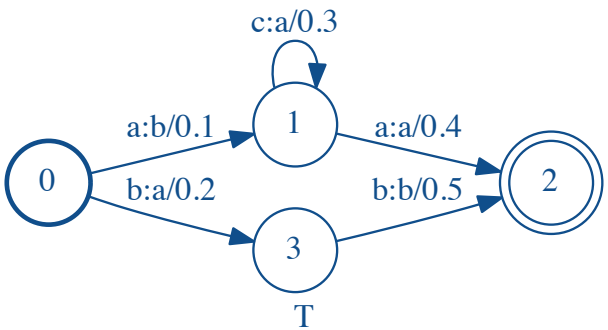


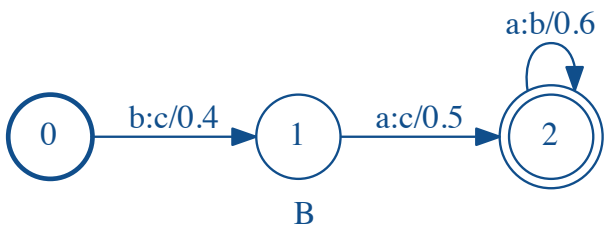
$T \cup B$

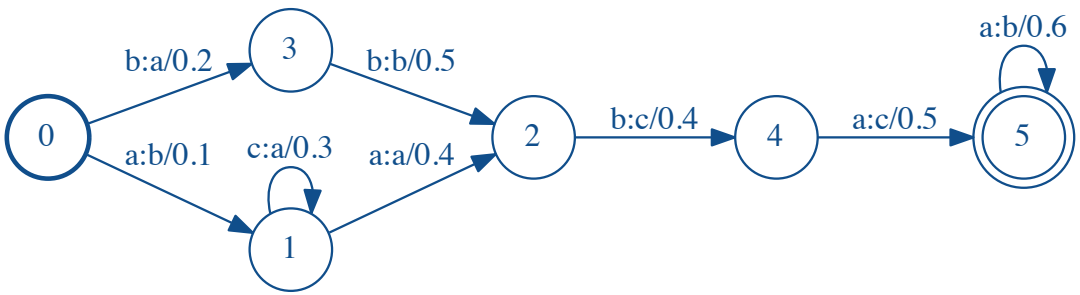
WFST Operations – Concatenation

Suppose A and B are two WFSTs: $A \otimes B$ is the concatentation of A with B which maps x to z

$$[\![A \otimes B]\!](x, y) = \bigoplus_{x=x_1x_2,\ y=y_1y_2} [\![A]\!](x_1, y_1) \otimes [\![B]\!](x_2, y_2)$$





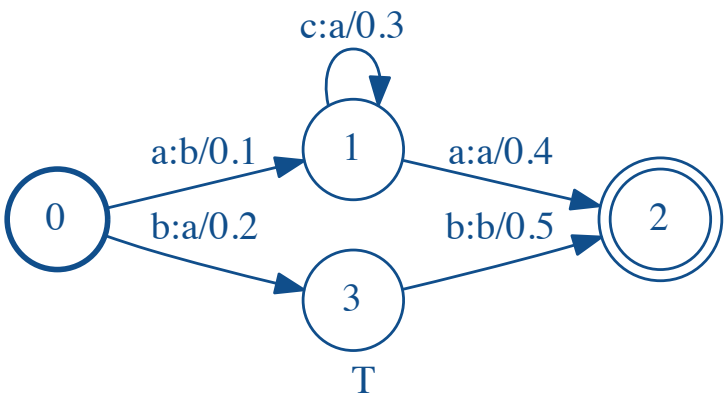


$T \otimes B$

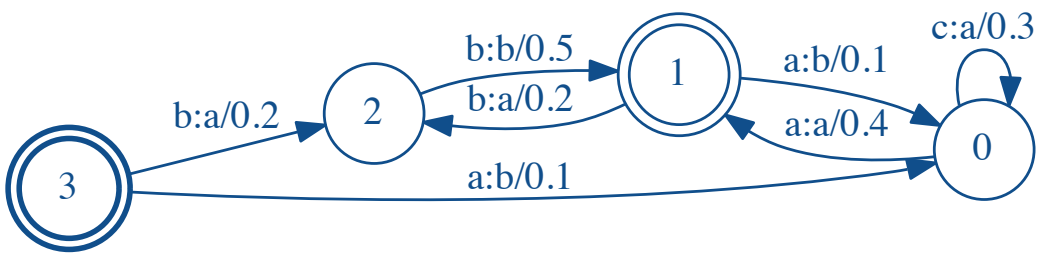
WFST Operations – Closure

Suppose T is a WFST. Its Closure T^* is

$$[\![T^*]\!](x,y) = \bigoplus_{n=0}^{\infty} [\![T^n]\!](x,y)$$



T



T^*

WFST Operations – Disambiguation

Transducers often arise through successive composition of a sequence of individual transducers. The result is a complicated machine within which any straightforward alignment between the input and output is lost [3]:

- there may be multiple paths that accept the same input string – ambiguous
- there may be multiple output paths for a single input string – non-functional

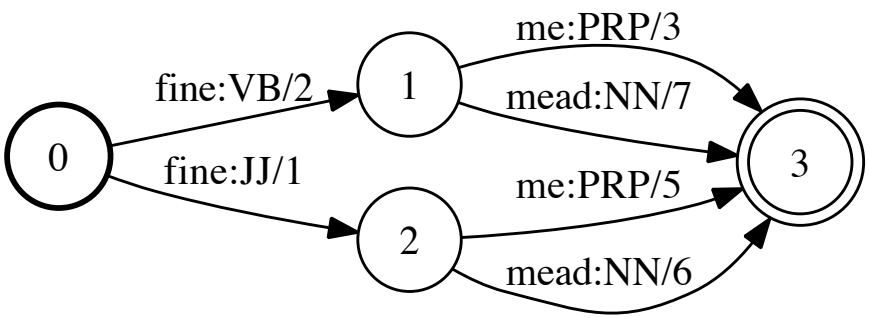
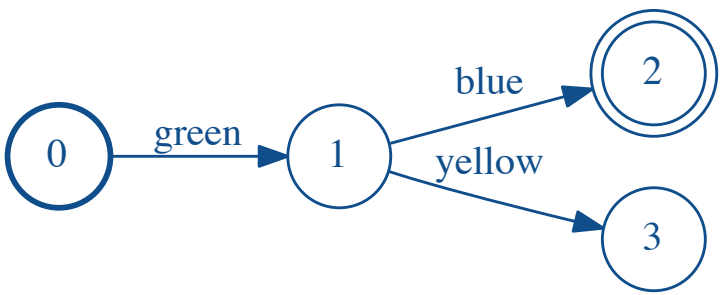


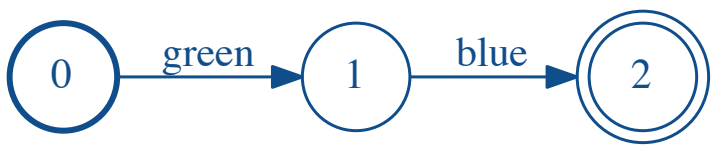
Fig. 1. In this simple illustrative POS-tagged WFST, 2-best scoring path discards all analyses of *fine mead*.

Disambiguation of an WFST is the task of creating a new WFST that encodes only the best-scoring path of each input string, while still maintaining the arc-level mapping between input and output symbols.

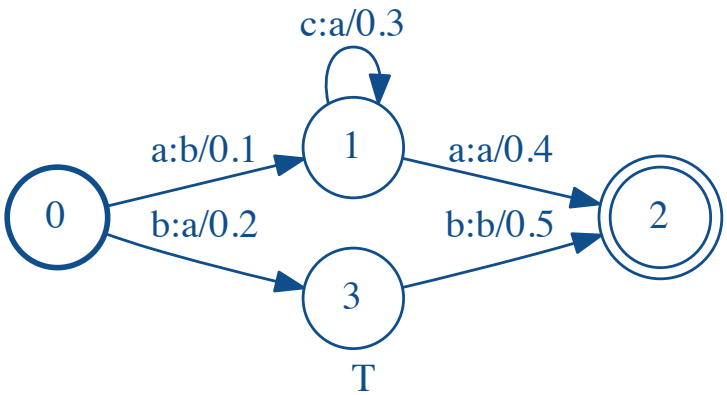
[3] Sproat et al. *Applications of Lexicographic Semirings to Problems in Speech and Language Processing*, Computational Linguistics 2014 `http://www.aclweb.org/anthology/J/J14/`

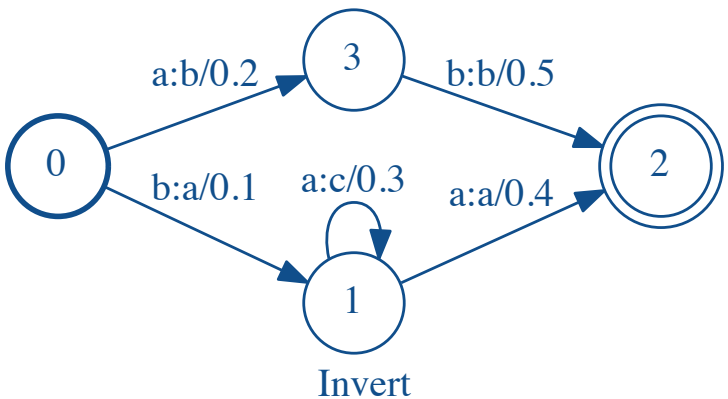
Connect – remove useless states and arcs

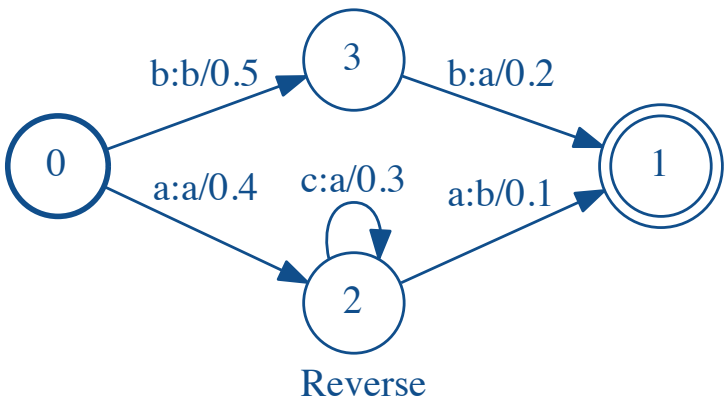




Invert – swaps input and output labels
Reverse – reverse input and output languages







Summary

- Weighted automata can defined probability distributions over the languages they define
  - Probabilistic context-free grammars and trees
  - Weighted finite state automata and regular languages
- Three semirings: probability, log, and tropical
- WFSAs as acceptors, and algorithms
- WFSAs as transducers, and algorithms
- Complexity

Operational Complexity

Suppose an automata has Q states and E edges:

- Operations on one automata
  - Reversal, Inversion, Projection, Connection have complexity $O(|Q| + |E|)$
  - Epsilon removal can be very expensive (cubic)
  - Determinization can be exponential
- Operations on two automata
  - Composition, Intersection, Difference have complexity $O((|Q_1| + |E_1|)(|Q_2| + |E_2|))$

Pushing - Algorithm

1. For each state q, compute its distance from the final states

$$d[q] = \bigoplus_{\pi \in P(q,F)} w[\pi]$$

2. Reweight every edge in the automata: $p \xrightarrow{i/w} n$

$$w \leftarrow (d[p])^{-1} w \otimes d[n]$$

- $d[p]$ is the distance from the start node of e to the final state(s)
- The multiplicative inverse has to be defined for the semiring [4].
  - For the tropical and log semirings, the multiplicative inverse in the semiring is simply subtraction.

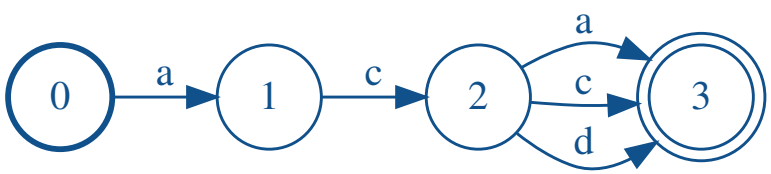
[4] M. Mohri, M. Riley. *A weight pushing algorithm for large vocabulary speech recognition.* Interspeech 2001

Failure Transitions – Special Symbols

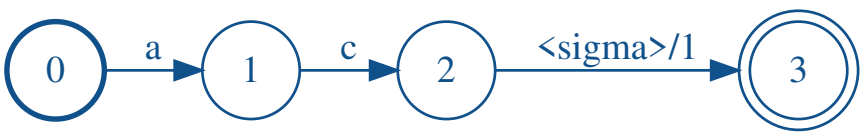
OpenFST provides *special symbols* used in matching symbols in composition and intersection.

	Consumes no symbol	Consumes symbol
Matches all	ϵ	σ
Matches 'rest'	ϕ	ρ

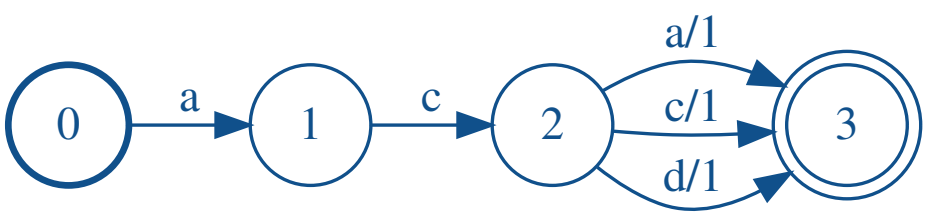
σ arcs



A



B



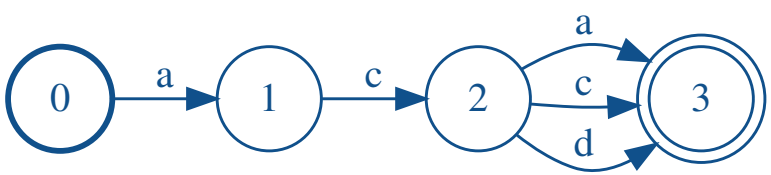
$A \circ B$

Failure Transitions – Special Symbols

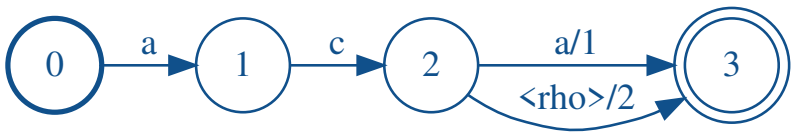
OpenFST provides *special symbols* used in matching symbols in composition and intersection.

	Consumes no symbol	Consumes symbol
Matches all	ϵ	σ
Matches 'rest'	ϕ	ρ

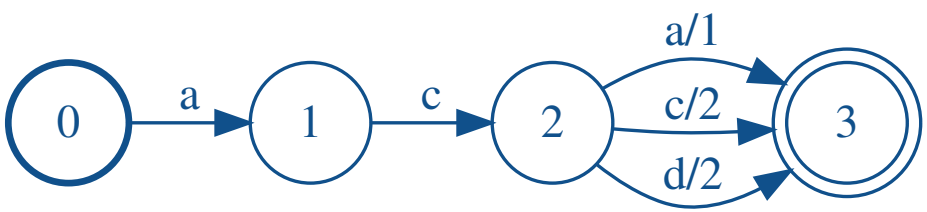
ρ arcs



A



B



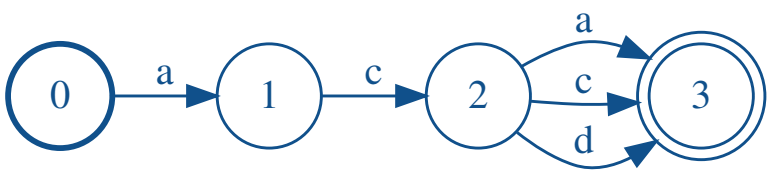
$A \circ B$

Failure Transitions – Special Symbols

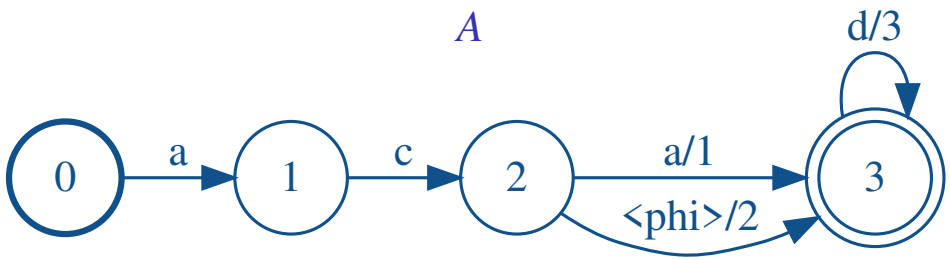
OpenFST provides *special symbols* used in matching symbols in composition and intersection.

	Consumes no symbol	Consumes symbol
Matches all	ϵ	σ
Matches 'rest'	ϕ	ρ

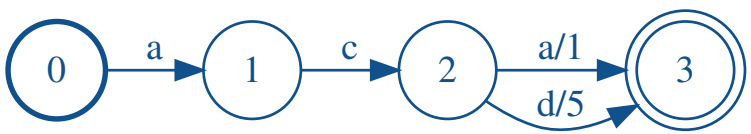
ϕ arcs



A



B



$A \circ B$