Automatic Speech Recognition (CS753)

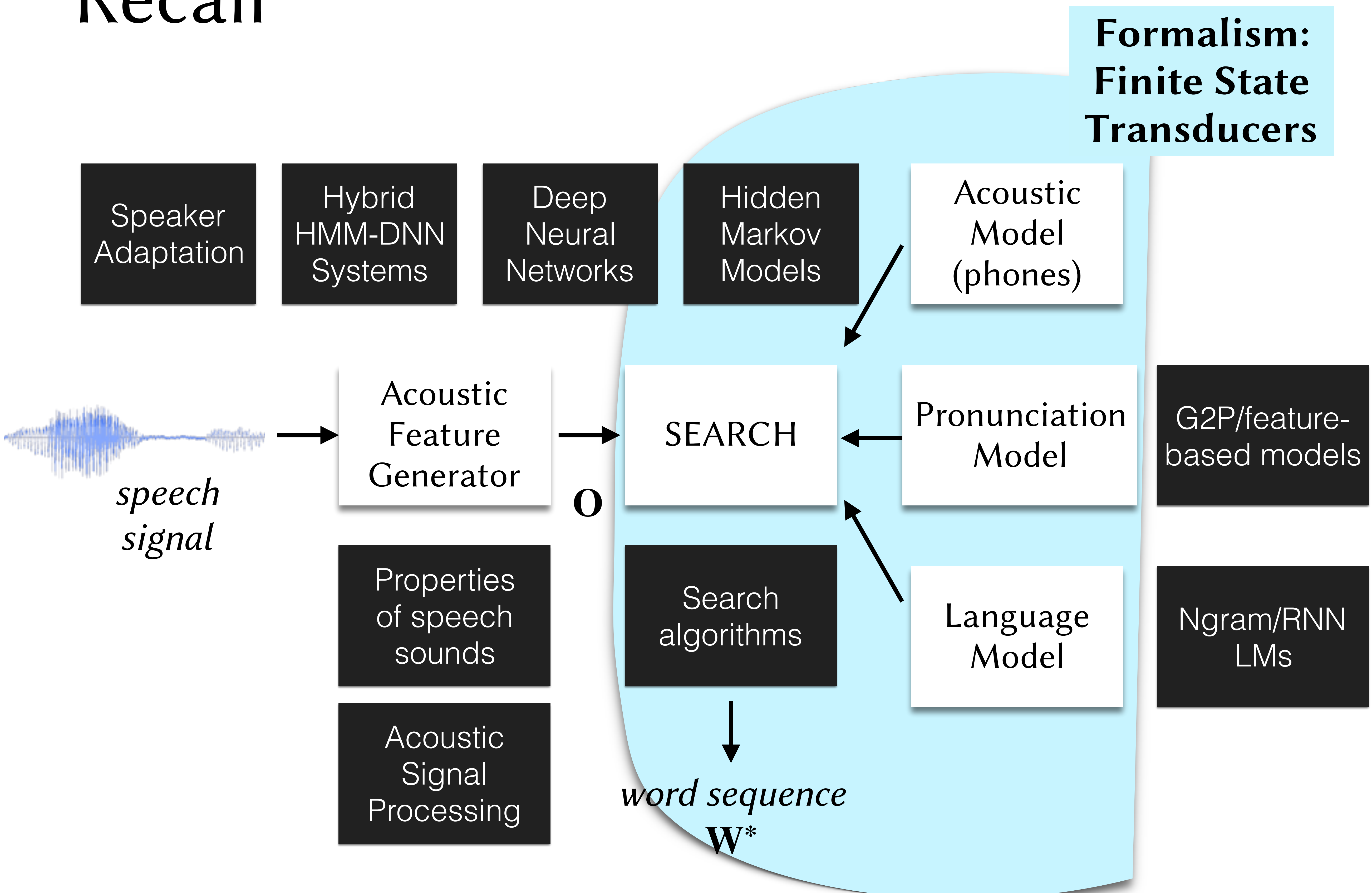
Lecture 2: Introducing WFSTs and WFST Algorithms

Instructor: Preethi Jyothi

Lecture 2

Includes content from a tutorial by the instructor at the
20th JSALT Workshop, Univ. of Washington, 2015

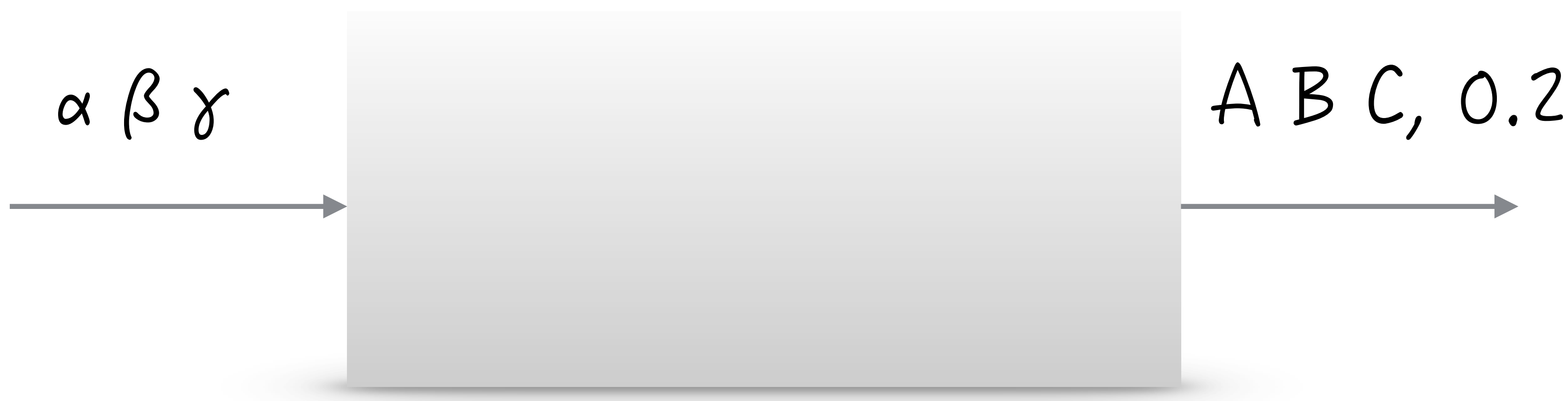
Recall


Formalism: Finite State Transducers
Speaker Adaptation
Hybrid HMM-DNN Systems
Deep Neural Networks
Hidden Markov Models
Acoustic Model (phones)
speech signal
Acoustic Feature Generator
O
SEARCH
Pronunciation Model
G2P/feature-based models
Properties of speech sounds
Search algorithms
Language Model
Ngram/RNN LMs
Acoustic Signal Processing
word sequence
W*

(Weighted) Automaton

- Accepts a subset of strings (over an alphabet), and rejects the rest
  - Mathematically, specified by $L \subseteq \Sigma^*$ or equivalently $f : \Sigma^* \longrightarrow \{0,1\}$
- Weighted: outputs a "weight" as well (e.g., probability)
  - $f : \Sigma^* \longrightarrow W$
- Transducer: outputs another string (over possibly another alphabet)
  - $f : \Sigma^* \times \Delta^* \longrightarrow W$

(Weighted) *Finite State* Automaton

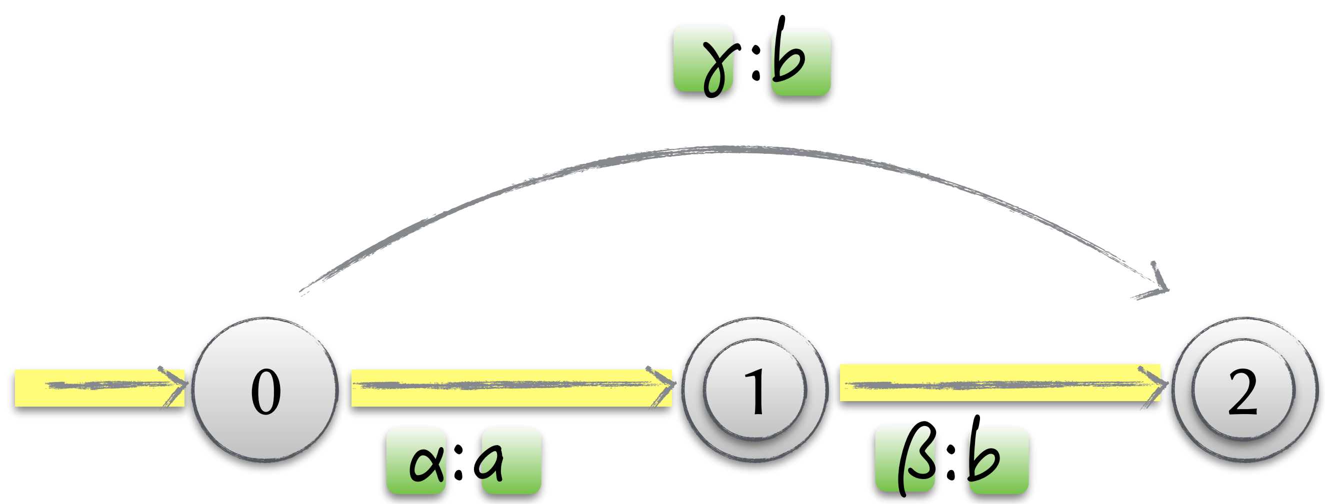


- Functions that can be implemented using a machine which:
  - reads the string one symbol at a time
  - has a fixed amount of memory: so, at any moment, the machine can be in only one of finitely many *states*, irrespective of the length of the input string
- Allows efficient algorithms to reason about the machine
  - e.g., output string with maximum weight for input $\alpha\beta\gamma$

Why WFSTs?

- Powerful enough to (reasonably) model processes in language, speech, computational biology and other machine learning applications
  - Simpler WFSTs can be combined to create complex WFSTs, e.g., speech recognition systems
- If using WFST models, efficient algorithms available to train the models and to make inferences
  - Toolkits that don't have domain specific dependencies

Structure: Finite State Transducer (FST)

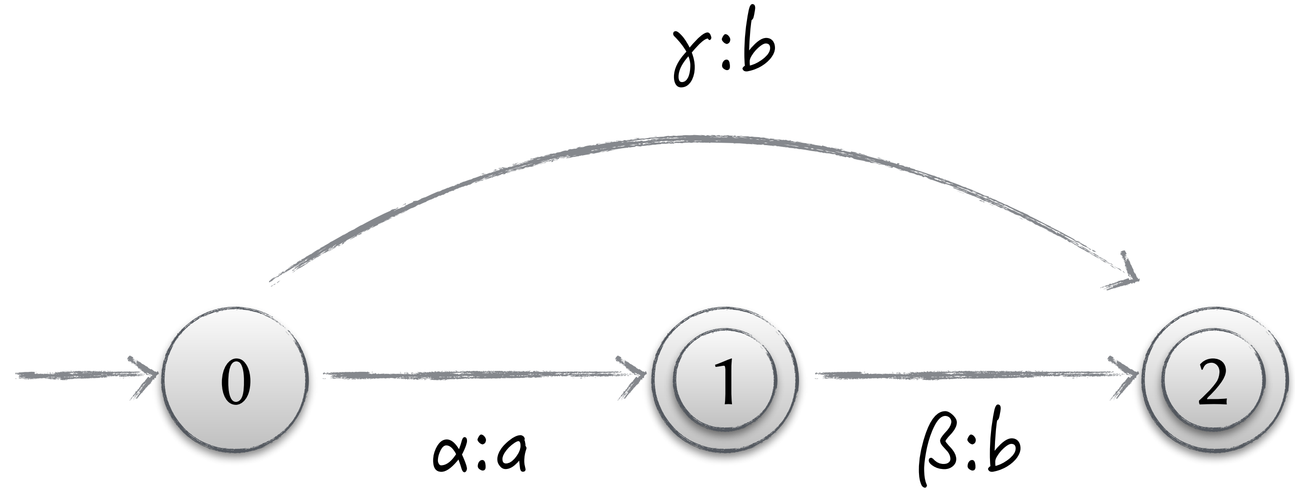


Elements of an FST

- States
- Start state (0)
- Final states (1 & 2)
- Arcs (transitions)
- Input symbols (from alphabet Σ)
- Output symbols (from alphabet Δ)

FST maps input strings to output strings

Path



- A successful "path" $\longrightarrow$ Sequence of transitions from the start state to any final state

- Input label of a path $\longrightarrow$ Concatenation of input labels on arcs. Similarly for output label of a path.

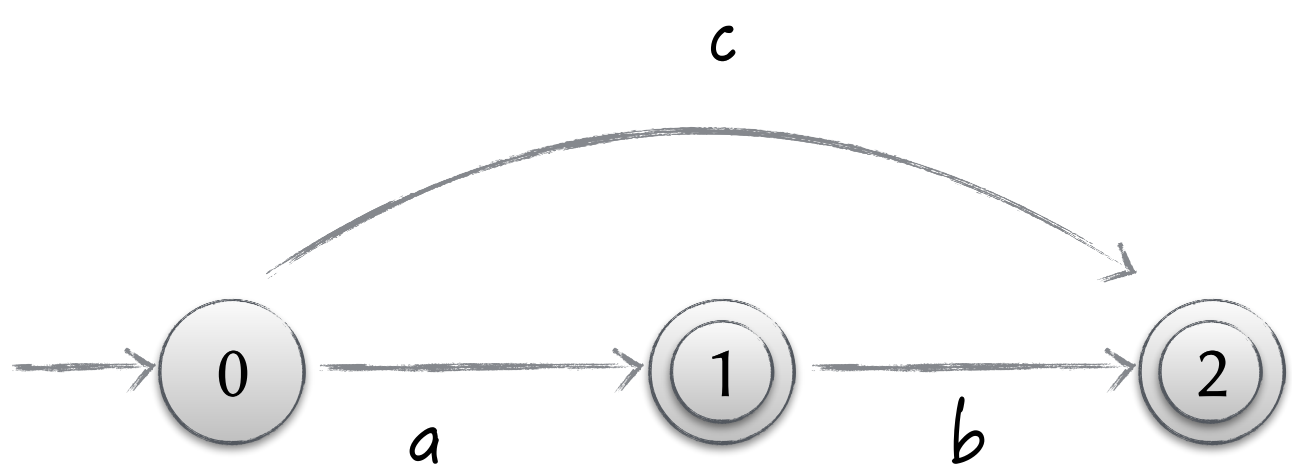
FSAs and FSTs

- Finite state acceptors (FSAs)
  - Each transition has a source & destination state, and **a label**
  - FSA accepts a set of strings, $L \subseteq \Sigma^*$

FSA can be thought of as a special kind of FST

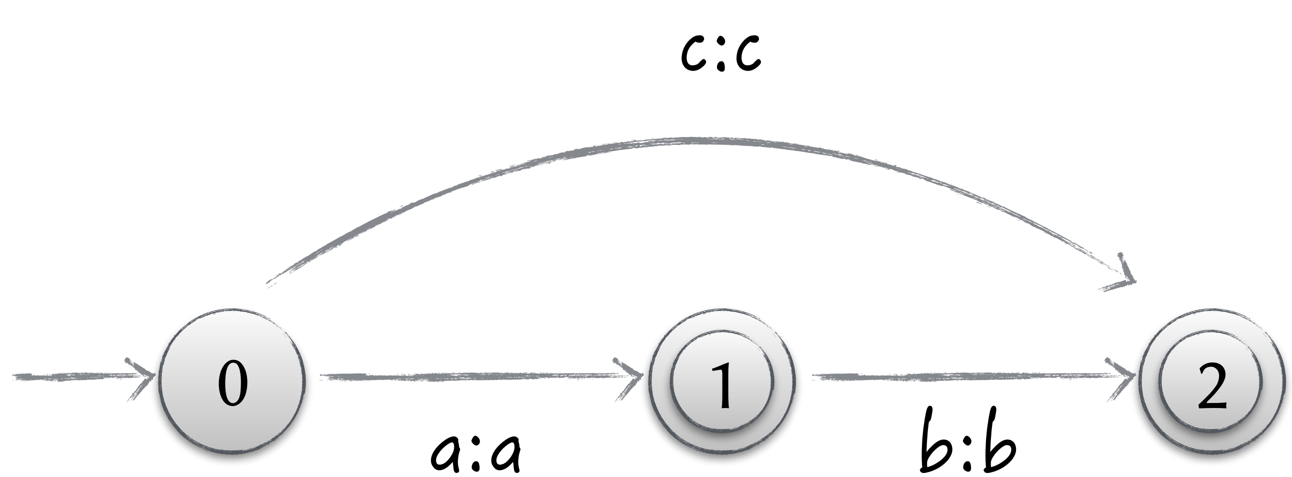
- Finite state transducers (FSTs)
  - Each transition has a source & destination state, **an input label and an output label**
  - FST represents a relationship, $R \subseteq \Sigma^* \times \Delta^*$

Example of an FSA



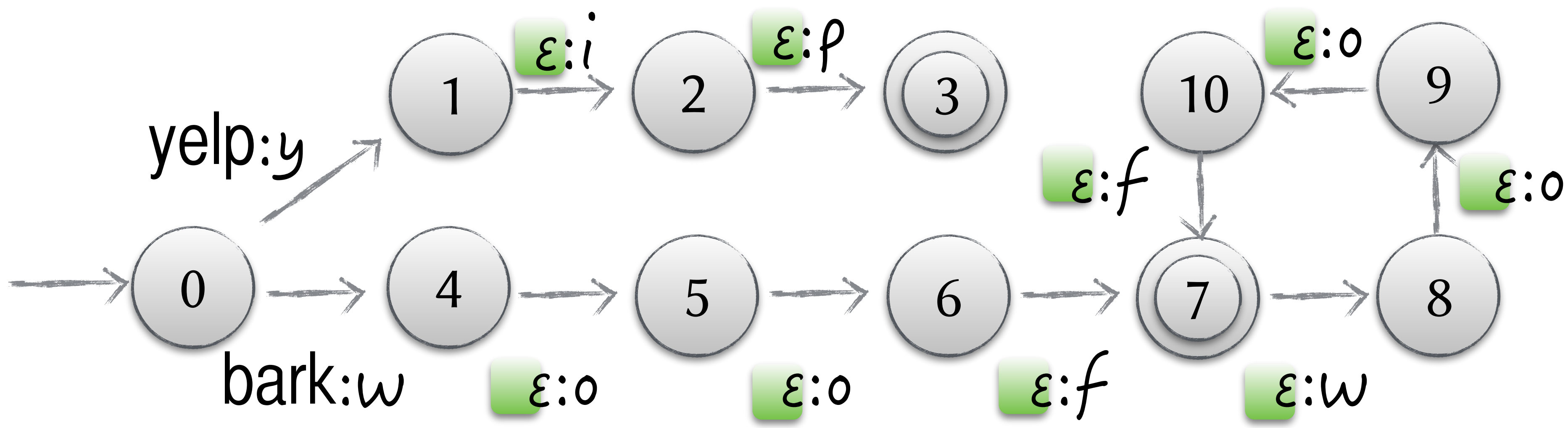
Accepts strings
{c, a, ab}

Equivalent FST



Accepts strings
{c, a, ab}
and outputs identical strings
{c, a, ab}

Barking dog FST



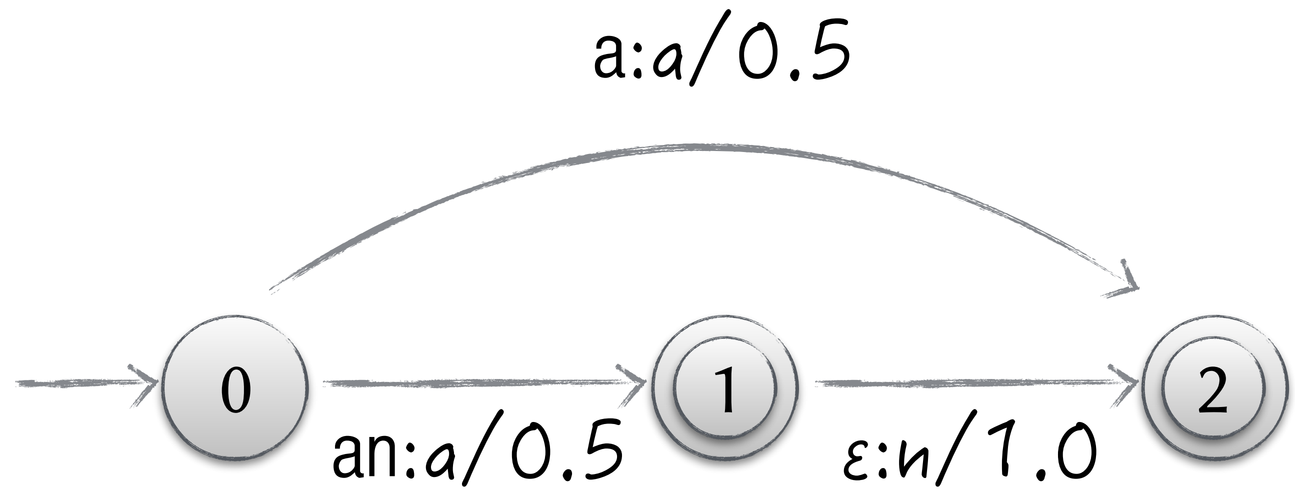
Σ = { yelp, bark }, Δ = { *a*, …, *z* }

yelp ⟶ *y i p*. bark ⟶ *w o o f* | *w o o f w o o f* |…

Special symbol, ε (epsilon) : allows to make a move without consuming an input symbol

or without producing an output symbol

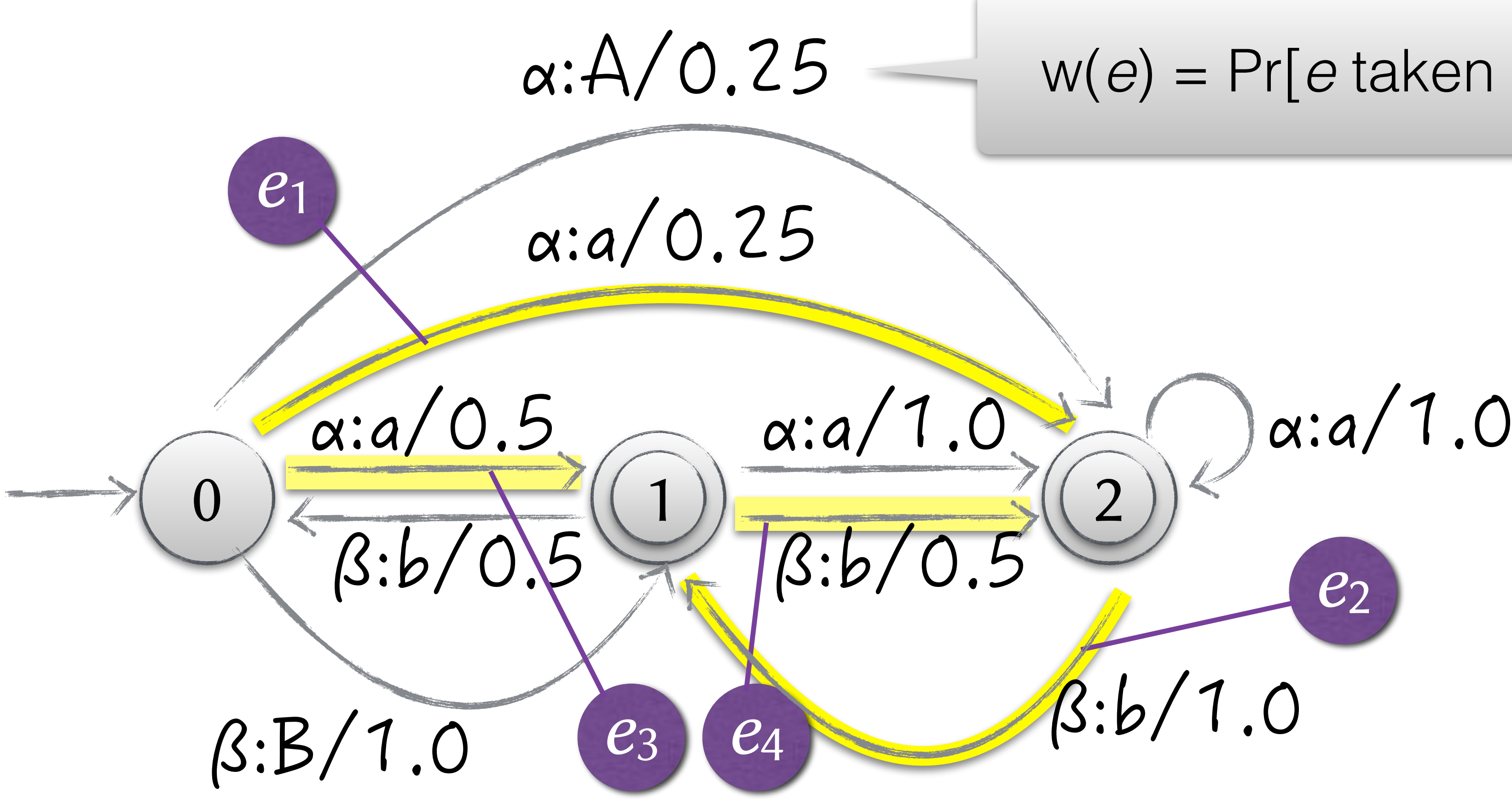
Weighted Path



- "Weights" can be probabilities, negative log-likelihoods, or any cost function representing the cost incurred in mapping an input sequence to an output sequence

- How are the weights accumulated along a path?

Weighted Path: Probabilistic FST



w(e) = Pr[e taken | state=0,in-symbol=α]

$\pi_1 = e_1 e_2$

$\pi_2 = e_3 e_4$

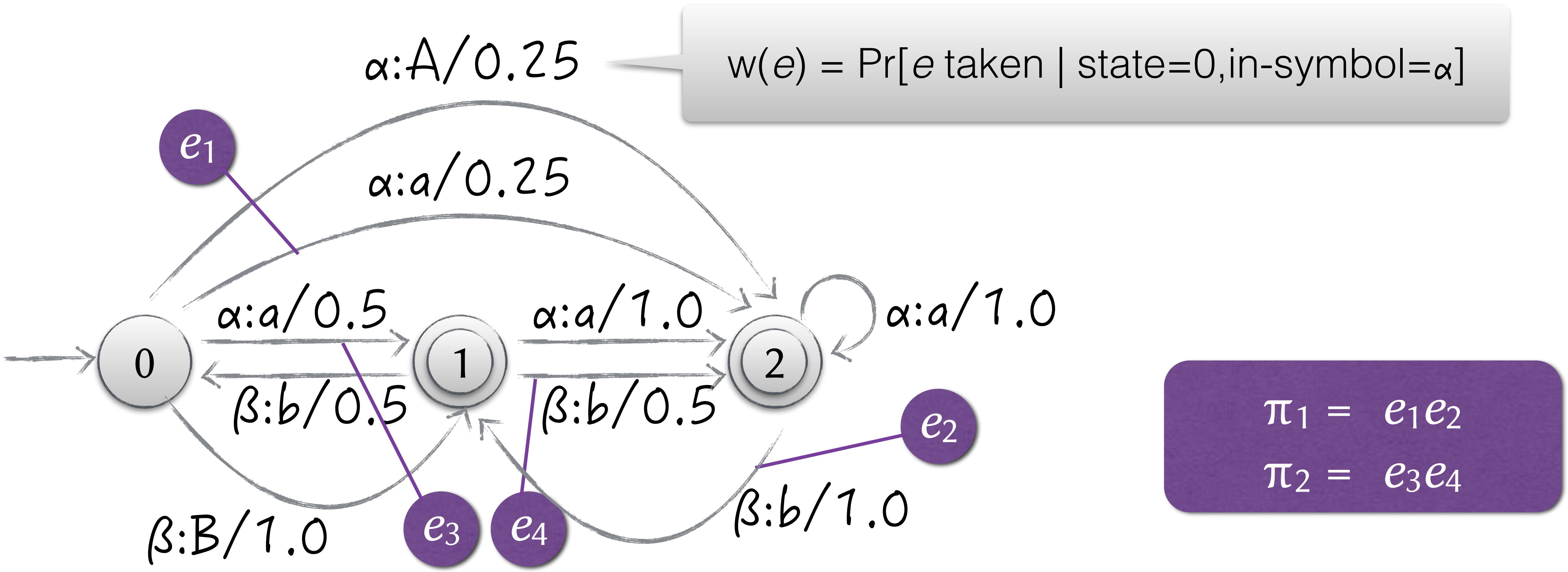
- T(αβ,ab) = Pr[output=ab, accepts | input=αβ, start]
  = Pr[π_1 | input=αβ, start] + Pr[π_2 | input=αβ, start]

= 0.25 + 0.25 = 0.5

= Pr[e_1 | input=αβ, start] × Pr[e_2 | input=αβ, start, e_1]
= Pr[e_1 | state=0, in-symb=α] × Pr[e_2 | state=2, in-symb=β]
= w(e_1) × w(e_2) = 0.25×1.0 = 0.25

= w(e3) × w(e4)
= 0.5×0.5 = 0.25

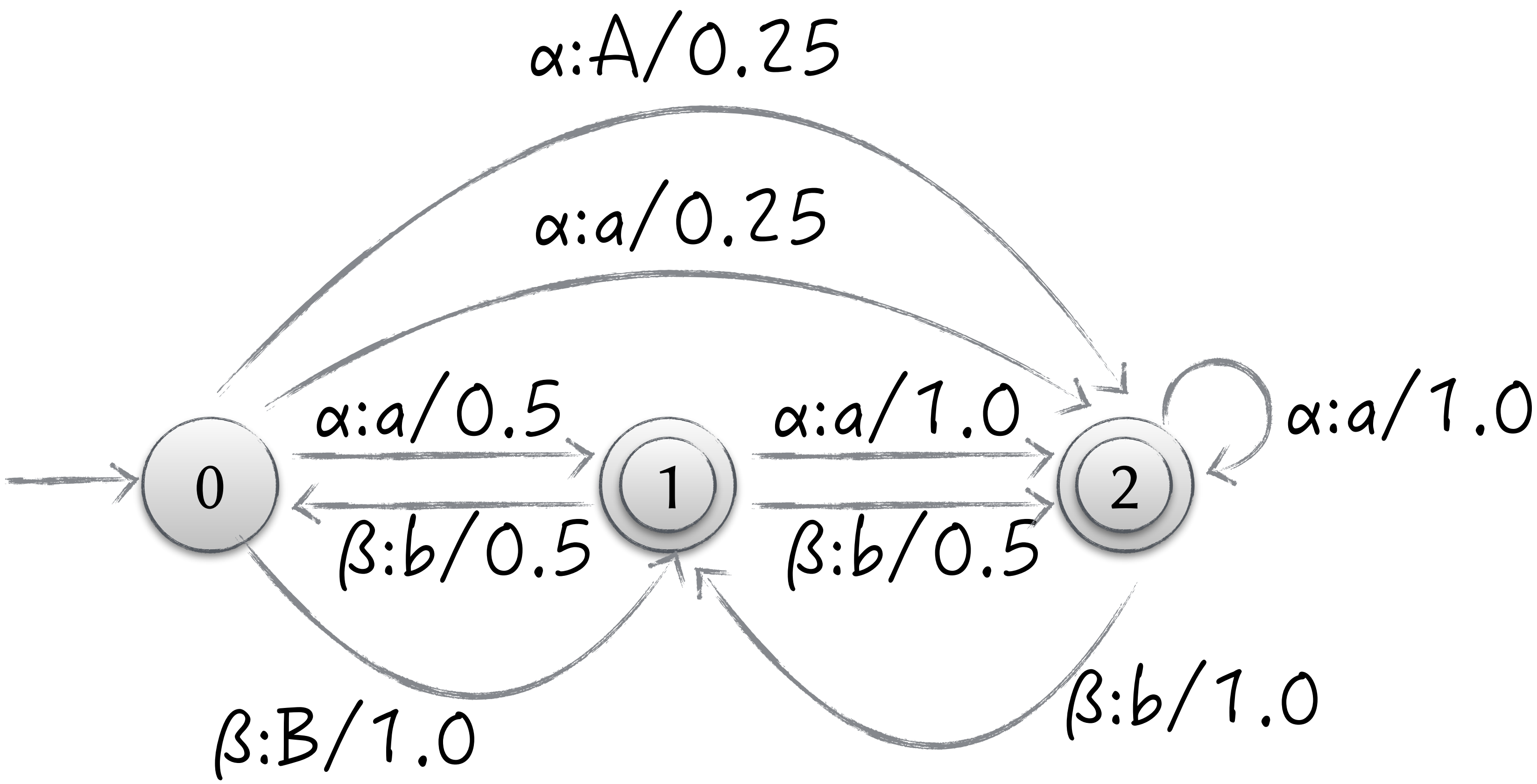
Weighted Path: Probabilistic FST



- $T(\alpha\beta, ab)$ = Pr[output=ab, accepts | input=$\alpha\beta$, start]
  = Pr[π_1 | input=$\alpha\beta$, start] + Pr[π_2 | input=$\alpha\beta$, start]

- More generally, $T(x,y) = \Sigma_{\pi \in P(x,y)} \; \Pi_{e \in \pi} \; w(e)$
  where $P(x,y)$ is the set of all accepting paths with input x and output y

Weighted Path



- But not all WFSTs are probabilistic FSTs
- Weight is often a "score" and maybe accumulated differently
- But helpful to retain some basic algebraic properties of weights: abstracted as **semirings**

Semirings

A semiring is a set of values associated with two operations $\oplus$ and $\otimes$, along with their identity values $\bar{0}$ and $\bar{1}$

Weight assigned to an input/output pair

$$T(x,y) = \bigoplus_{\pi \in P(x,y)} \bigotimes_{e \in \pi} w(e)$$

where $P(x,y)$ is the set of all accepting paths with input x, output y

(generalizing the weight function for a probabilistic FST)

Semirings

Some popular semirings [M02]

SEMIRING	SET	$\oplus$	$\otimes$	$\bar{0}$	$\bar{1}$
Boolean	{F,T}	$\vee$	$\wedge$	F	T
Probability	$\mathbb{R}_+$	+	$\times$	0	1
Log	$\mathbb{R} \cup \{-\infty, +\infty\}$	$\oplus_{\log}$	+	$+\infty$	0
Tropical	$\mathbb{R} \cup \{-\infty, +\infty\}$	min	+	$+\infty$	0

Is there

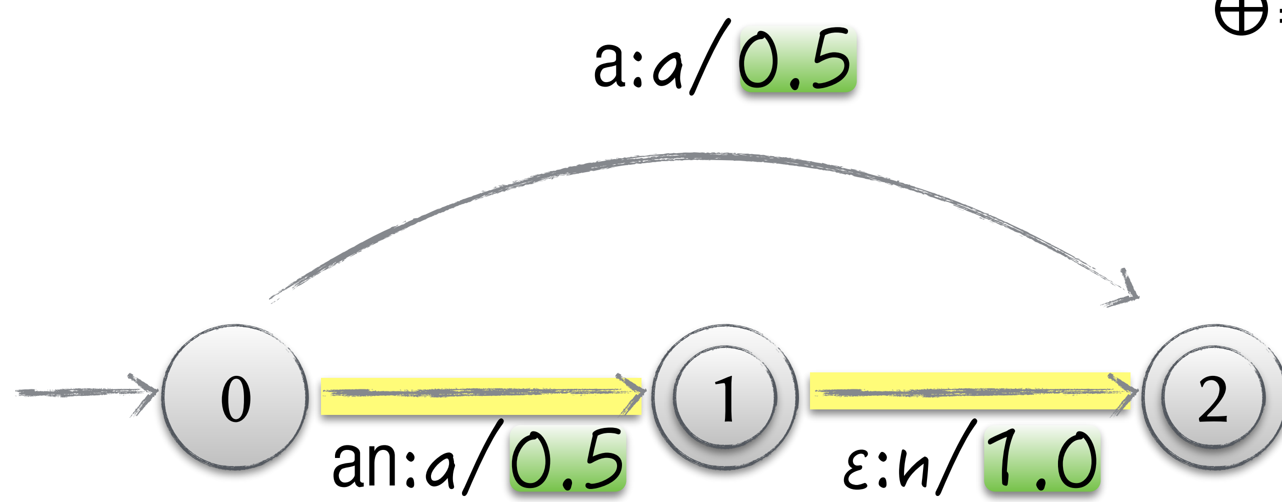
“Viterbi Approx.” of -log Pr[*y*|*x*]

Operator $\oplus_{\log}$ defined as: $x \oplus_{\log} y = -\log(e^{-x} + e^{-y})$

[M02] Mohri, M. Semiring frameworks and algorithms for shortest-distance problems, *Journal of Automata, Languages and Combinatorics*, 7(3):321—350, 2002.

Weighted Path: Tropical Semiring

$\oplus$=min | $\otimes$=+



- Weight of a path π is the $\otimes$-product of all the transitions in π

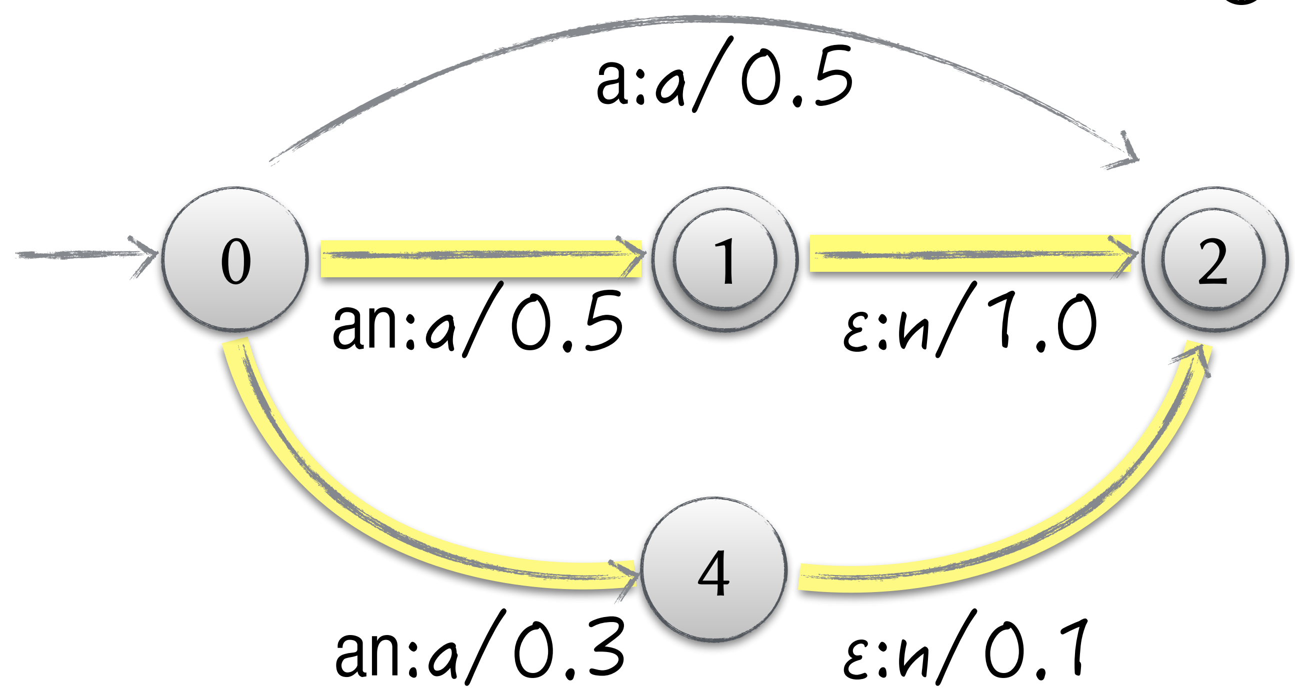
$w(\pi)$: $(0.5 \otimes 1.0) = 0.5 + 1.0 = 1.5$

- Weight of a sequence "$\underline{x}$,$\underline{y}$" is the $\oplus$-sum of all paths labeled with "$\underline{x}$,$\underline{y}$"

$w((\text{an}), (a\ n)) = (1.5 \oplus \bar{0}) = \min(1.5, \infty) = 1.5$

Weighted Path: Tropical Semiring

$\oplus$=min | $\otimes$=+



- Weight of a sequence "$\underline{x}$,$\underline{y}$" is the $\oplus$-sum of all paths labeled with "$\underline{x}$,$\underline{y}$"
  w((an), (a n)) = ?

  Path 1: $(0.5 \otimes 1.0) = 1.5$

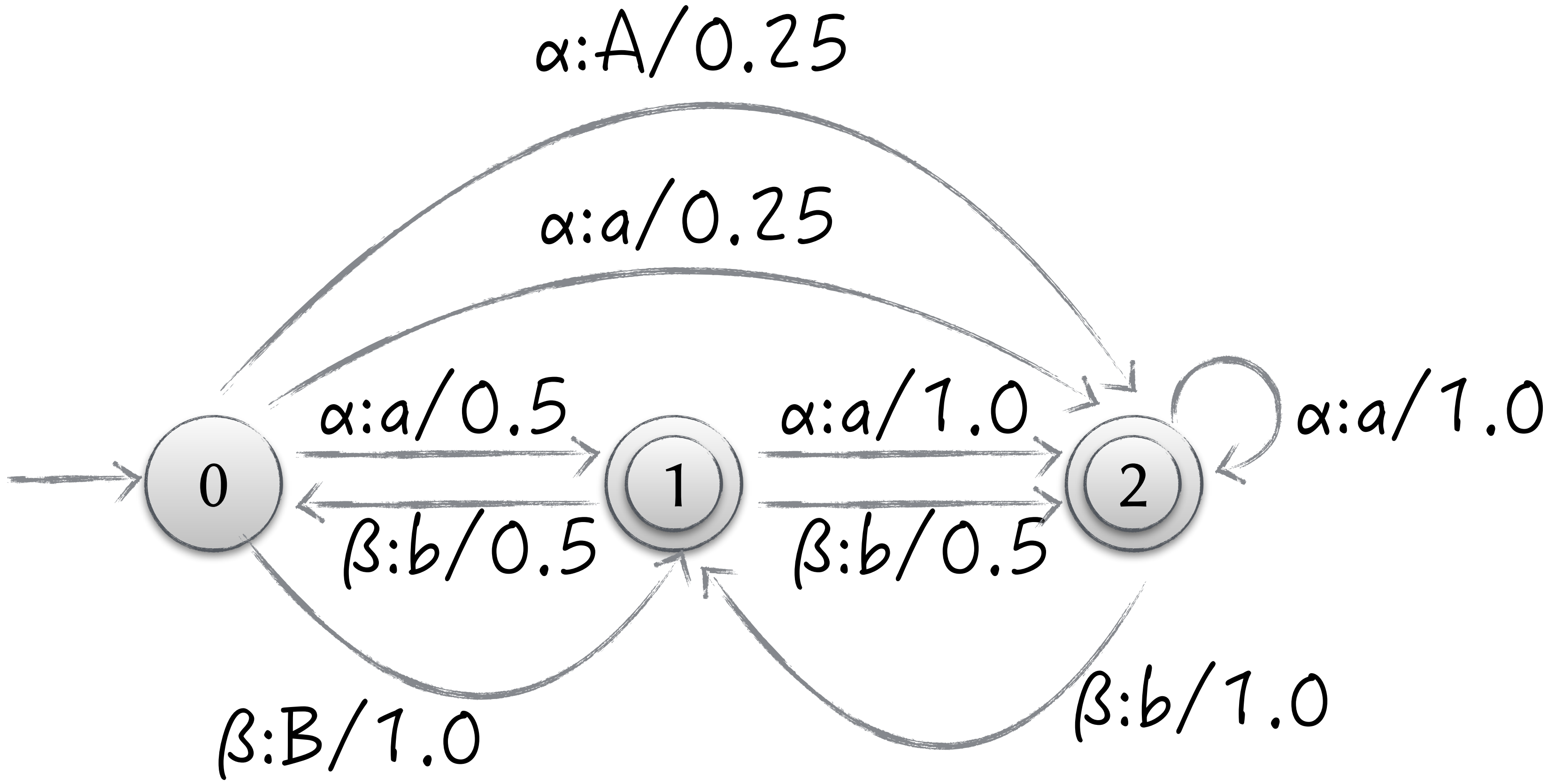
  Path 2: $(0.3 \otimes 0.1) = 0.4$

  Weight of "((an), (a n))" = $(1.5 \oplus 0.4) = 0.4$

Shortest Path

- Recall $T(x,y) = \bigoplus_{\pi \in P(x,y)} w(\pi)$
  where $P(x,y)$ = set of paths with input/output (x,y); $w(\pi) = \bigotimes_{e \in \pi} w(e)$
- In the probability semiring, a dynamic program to compute $T(x,y)$
  - $\Theta(|Q|^3)$ time : impractical for large FSTs
- In the tropical semiring $\oplus$ is min. $T(x,y)$ associated with a single path in $P(x,y)$: *Shortest Path*
  - Can be found using Dijkstra's algorithm : $\Theta(|E| + |Q| \cdot \log|Q|)$ time

Shortest Path

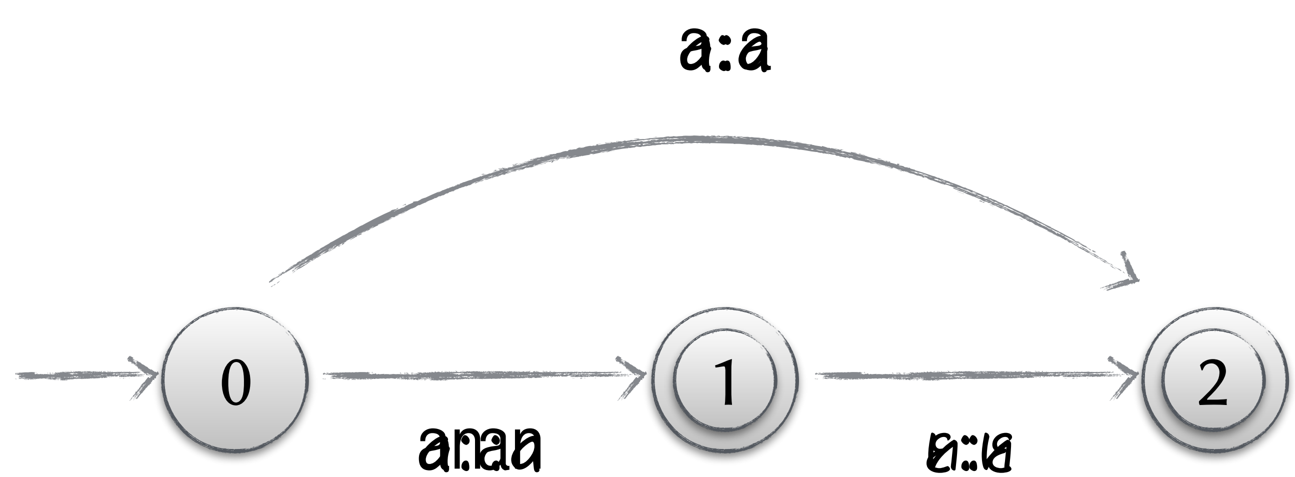


T("α", "a") = ?

T("αα", "aa") = ?

Inversion

Swap the input and output labels in each transition

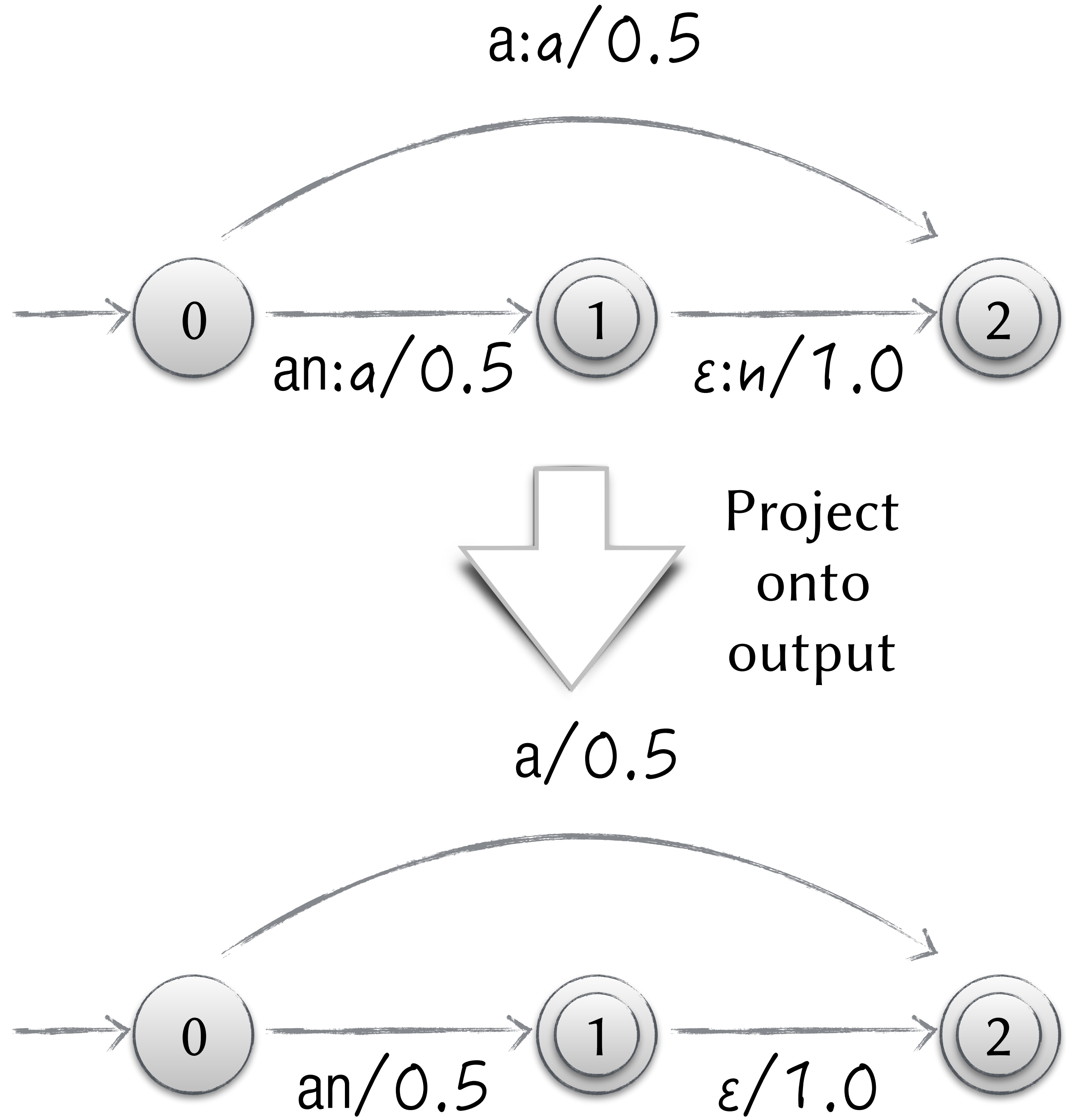


Weights (if they exist) are retained on the arcs

This operation comes in handy, especially during composition!

Projection

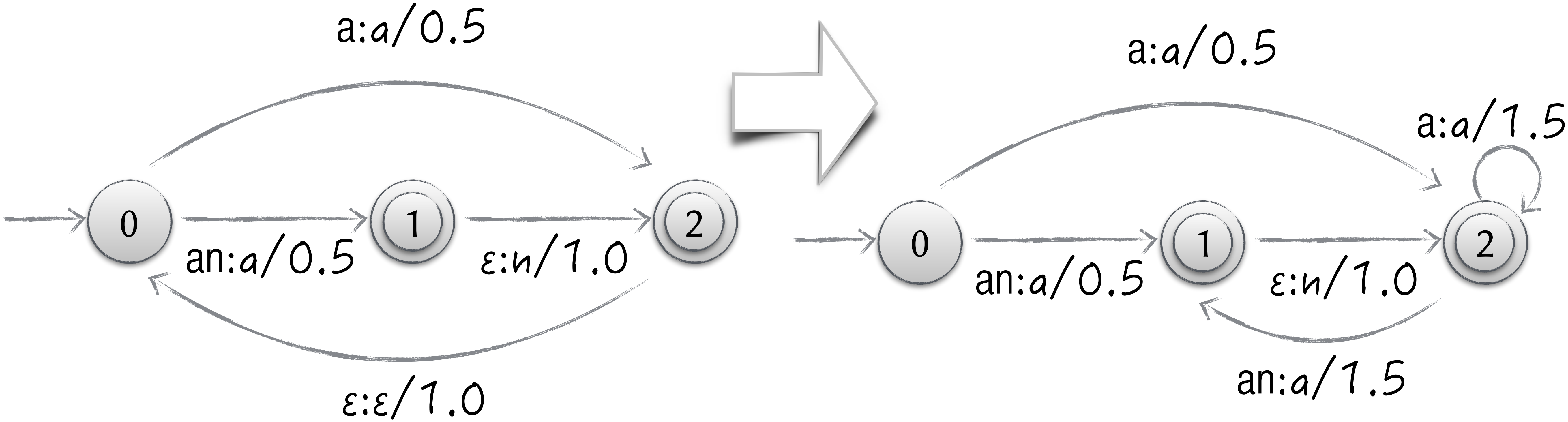
Project onto the input or output alphabet



Epsilon Removal

Attempts to remove epsilon arcs for more efficient use of WFSTs

Not all epsilons can be removed



Basic FST Operations (Rational Operations)

The set of weighted transducers are closed under the following operations [Mohri '02]:

1. Sum or **Union**: $(T_1 \oplus T_2)(\underline{x}, \underline{y}) = T_1(\underline{x}, \underline{y}) \oplus T_2(\underline{x}, \underline{y})$

2. Product or **Concatenation**: $(T_1 \otimes T_2)(x, y) = \bigoplus_{\substack{x=x_1x_2 \\ y=y_1y_2}} T_1(x_1, y_1) \otimes T_2(x_2, y_2)$

3. Kleene-**closure**: $T^*(x, y) = \bigoplus_{n=0}^{\infty} T^n(x, y)$

Basic FST Operations (Rational Operations)

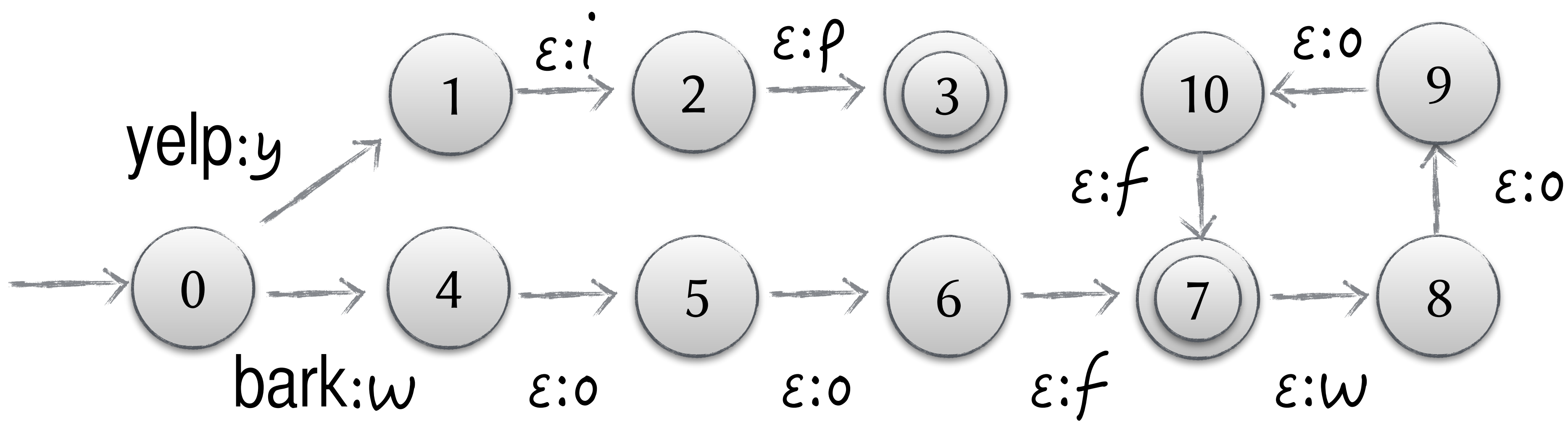
The set of weighted transducers are closed under the following operations [Mohri '02]:

1. Sum or **Union**: $(T_1 \oplus T_2)(\underline{x}, \underline{y}) = T_1(\underline{x}, \underline{y}) \oplus T_2(\underline{x}, \underline{y})$

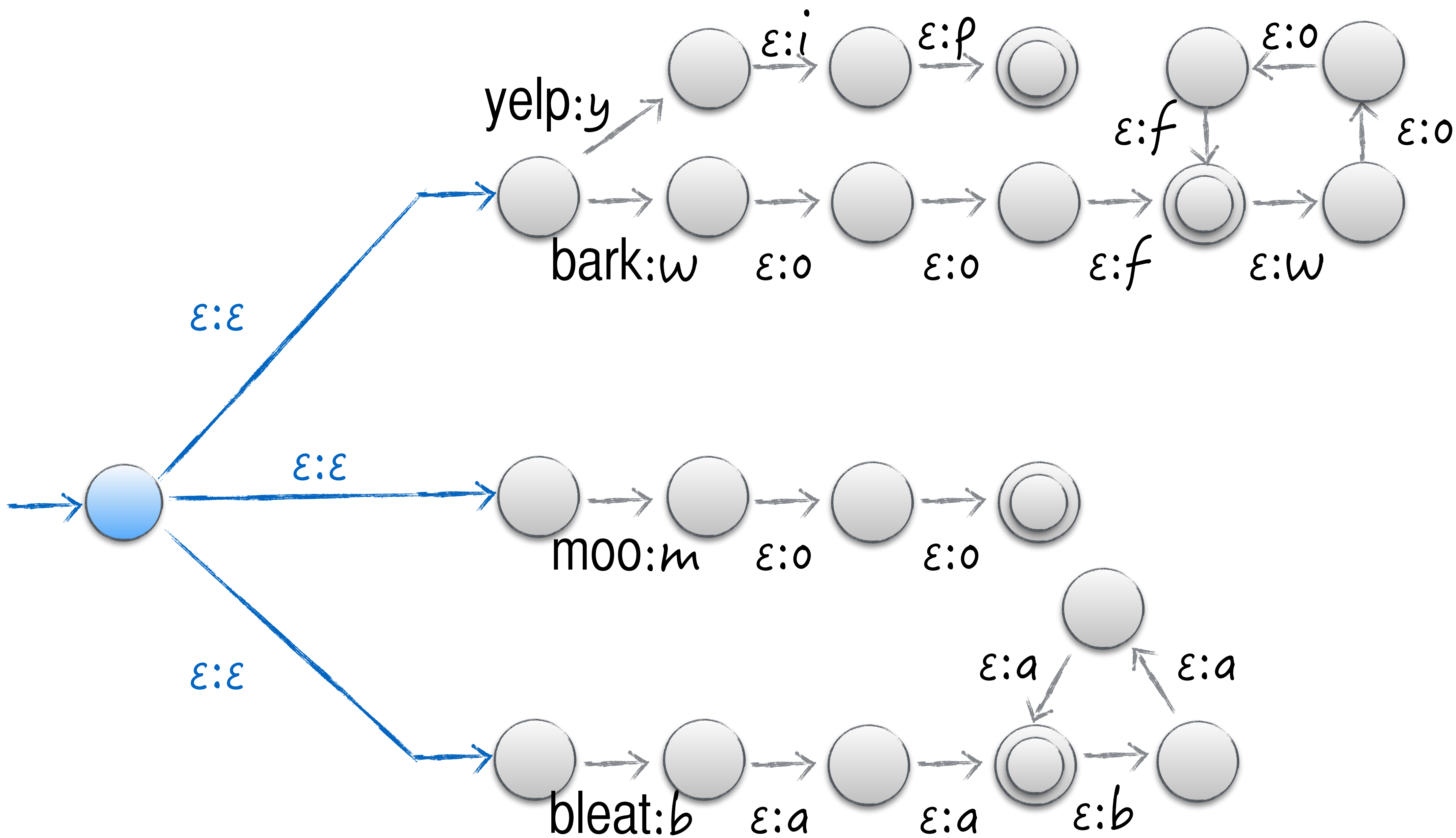
2. Product or **Concatenation**: $(T_1 \otimes T_2)(x, y) = \bigoplus_{\substack{x=x_1x_2 \\ y=y_1y_2}} T_1(x_1, y_1) \otimes T_2(x_2, y_2)$

3. Kleene-**closure**: $T^*(x, y) = \bigoplus_{n=0}^{\infty} T^n(x, y)$

Example: Recall Barking dog FST



Example: Union



Animal farm!

Basic FST Operations (Rational Operations)

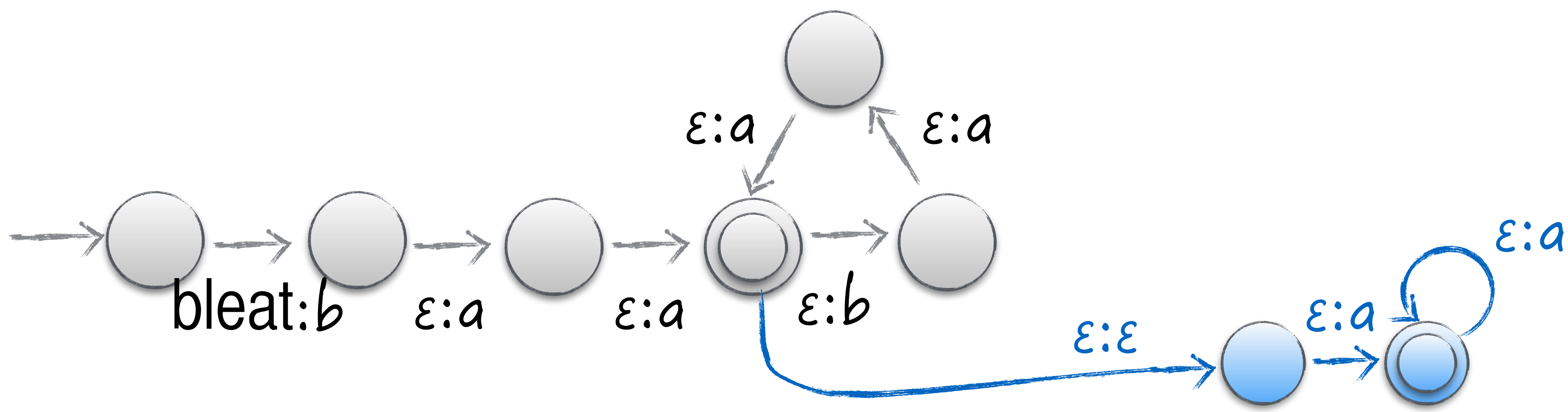
The set of weighted transducers are closed under the following operations [Mohri ‘02]:

1. Sum or **Union**: $(T_1 \oplus T_2)(\underline{x}, \underline{y}) = T_1(\underline{x}, \underline{y}) \oplus T_2(\underline{x}, \underline{y})$

2. Product or **Concatenation**: $(T_1 \otimes T_2)(x, y) = \bigoplus_{\substack{x=x_1x_2 \\ y=y_1y_2}} T_1(x_1, y_1) \otimes T_2(x_2, y_2)$

3. Kleene-**closure**: $T^*(x, y) = \bigoplus_{n=0}^{\infty} T^n(x, y)$

Example: Concatenation



Suppose the last "*baa*" in a bleat should be followed by one or more a's

(e.g., "*baabaa*" is not OK, but "*baaa*" and "*baabaaaaa*" are)

Basic FST Operations (Rational Operations)

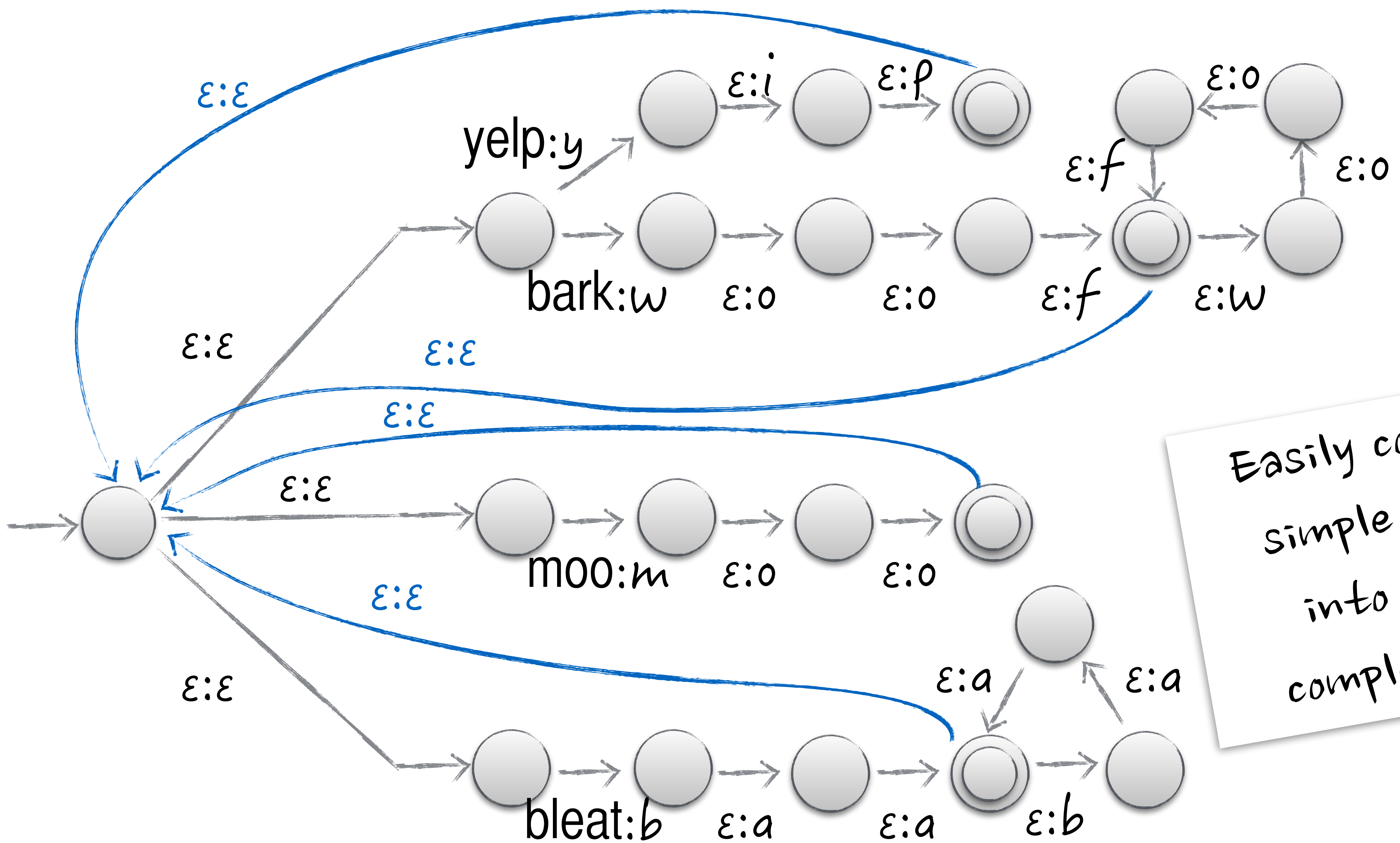
The set of weighted transducers are closed under the following operations [Mohri '02]:

1. Sum or **Union**: $(T_1 \oplus T_2)(\underline{x}, \underline{y}) = T_1(\underline{x}, \underline{y}) \oplus T_2(\underline{x}, \underline{y})$

2. Product or **Concatenation**: $(T_1 \otimes T_2)(x, y) = \bigoplus_{\substack{x=x_1x_2 \\ y=y_1y_2}} T_1(x_1, y_1) \otimes T_2(x_2, y_2)$

3. Kleene-**closure**: $T^*(x, y) = \bigoplus_{n=0}^{\infty} T^n(x, y)$

Example: Closure

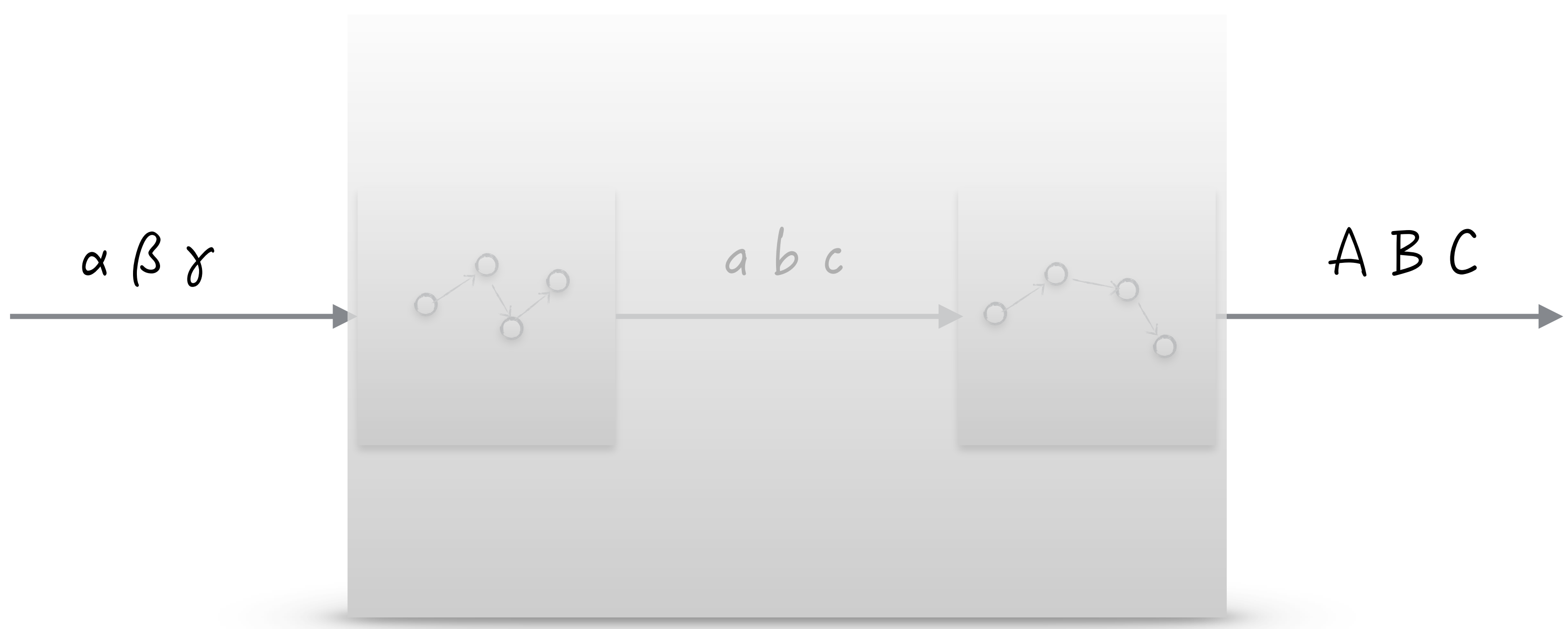


Easily combine simple FSTs into more complex ones

Animal farm: allow arbitrarily long sequence of sounds!

bark moo yelp bleat ⟶ w o o f w o o f m o o y i p b a a b a a

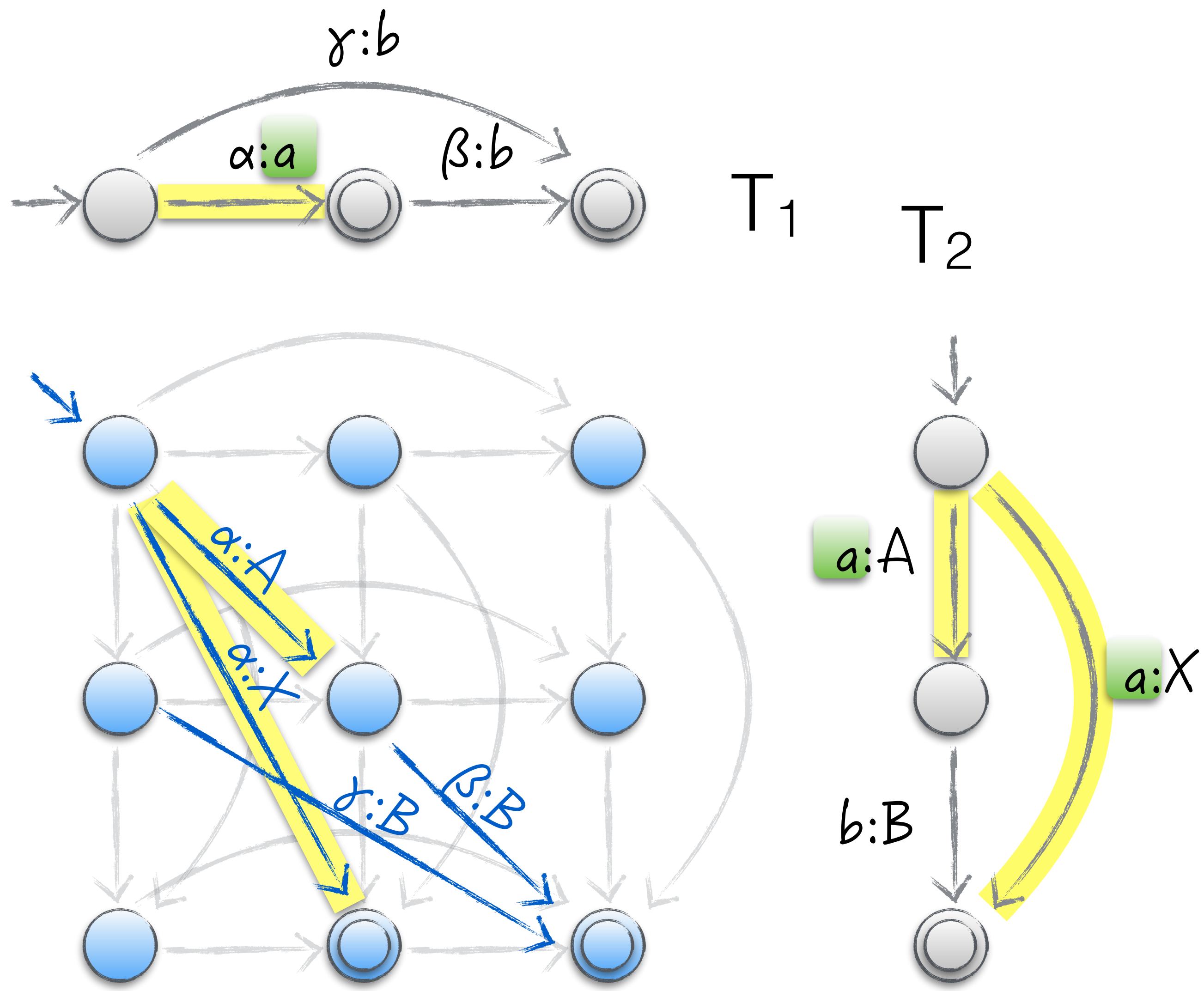
Composition



- If T_1 transduces x to z, and T_2 transduces z to y, then $T_1 \circ T_2$ transduces x to y

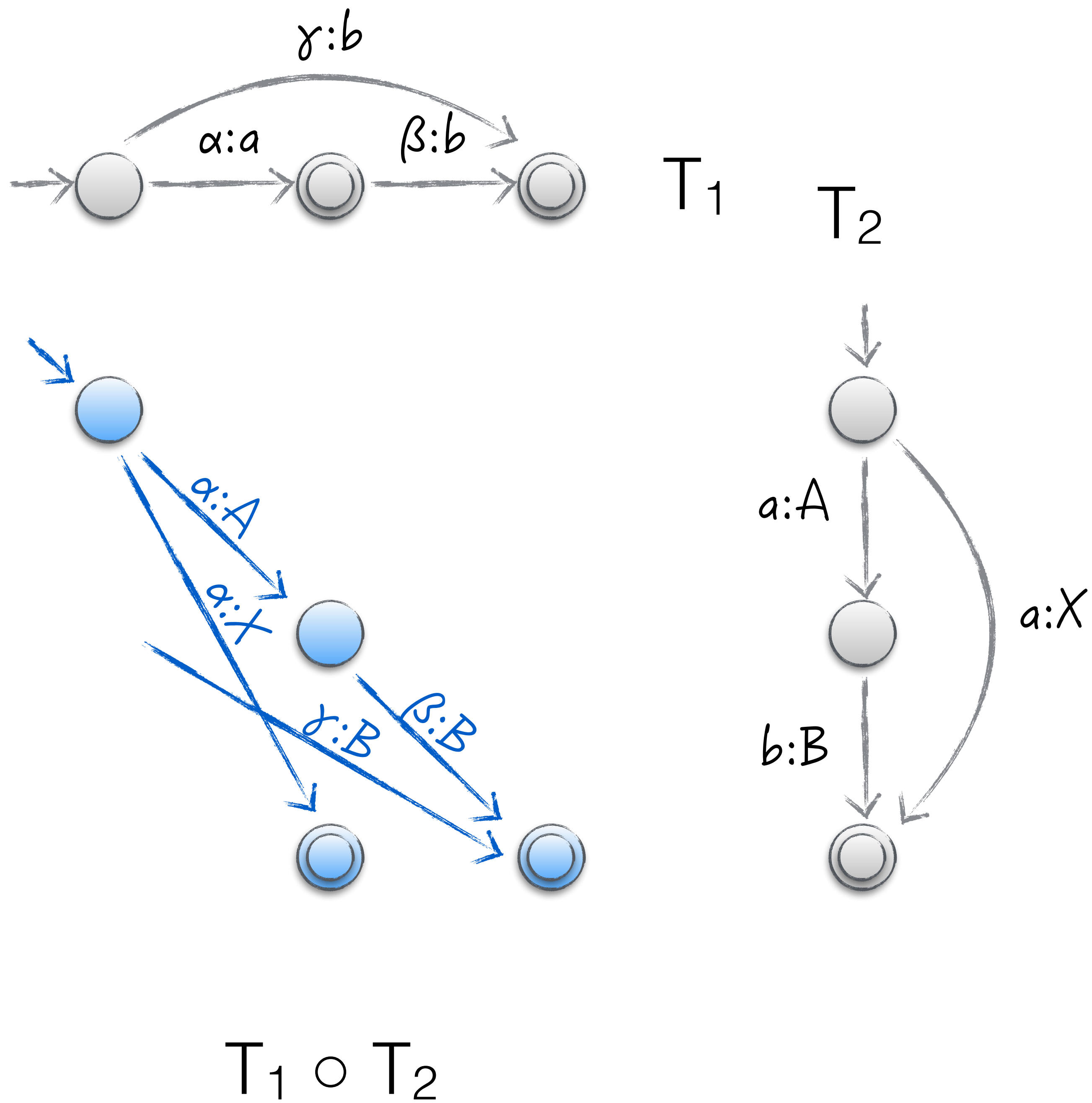
- $(T_1 \circ T_2)(x, y) = \bigoplus_{z} T_1(x, z) \otimes T_2(z, y)$

Composition: Construction



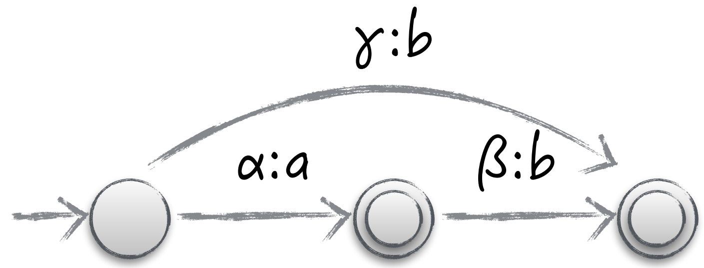
$T_1 \circ T_2$

Composition



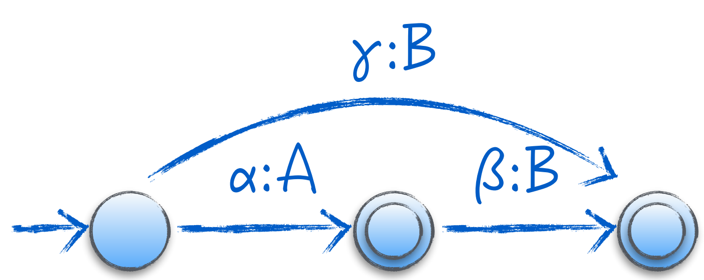
Composition: Example 1

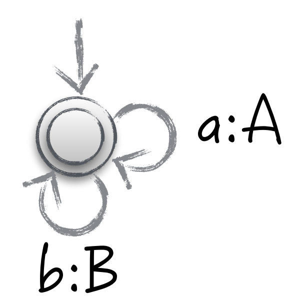
T_2 translates the output from one alphabet to another



T_1

T_2

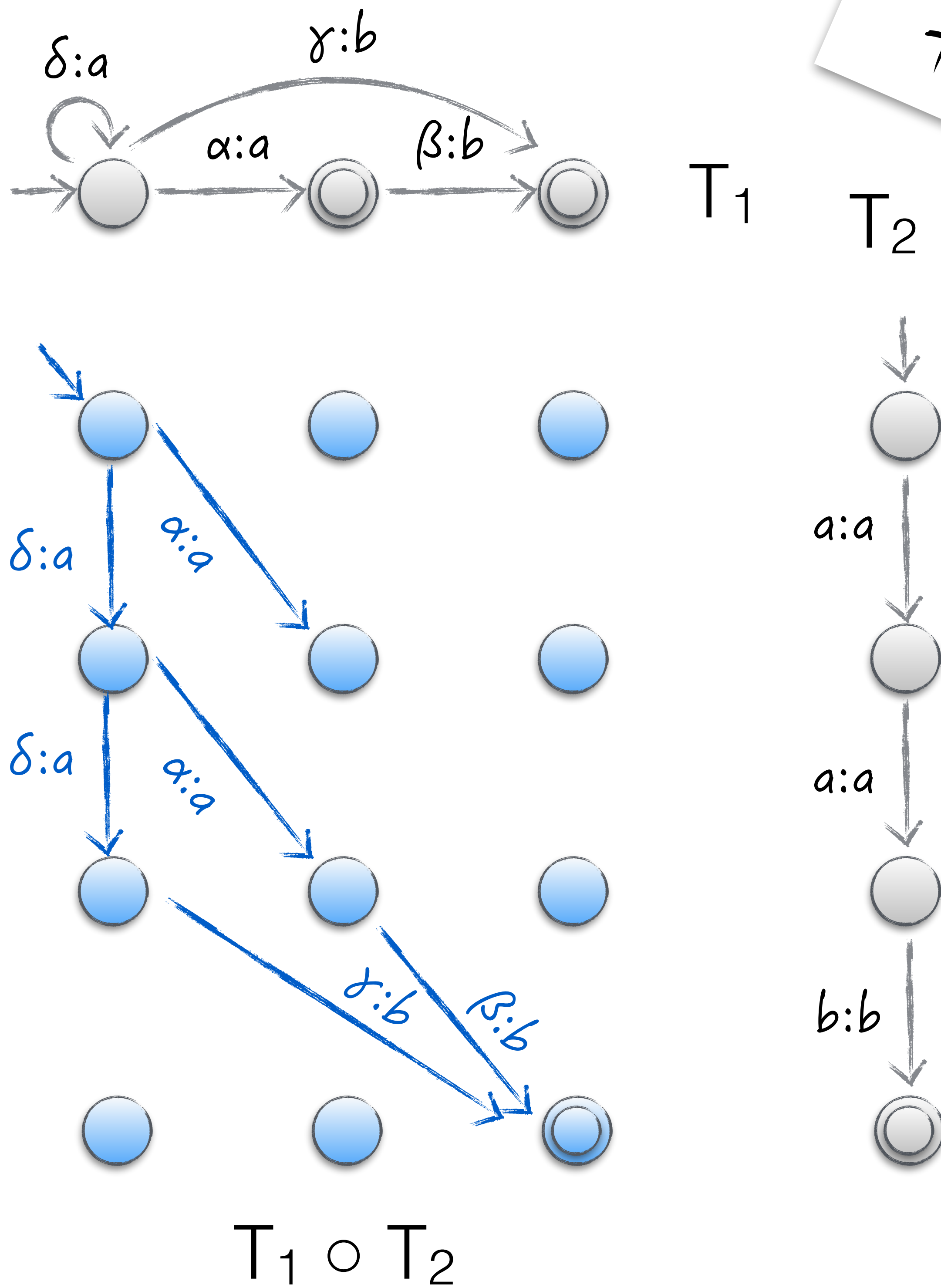




$T_1 \circ T_2$

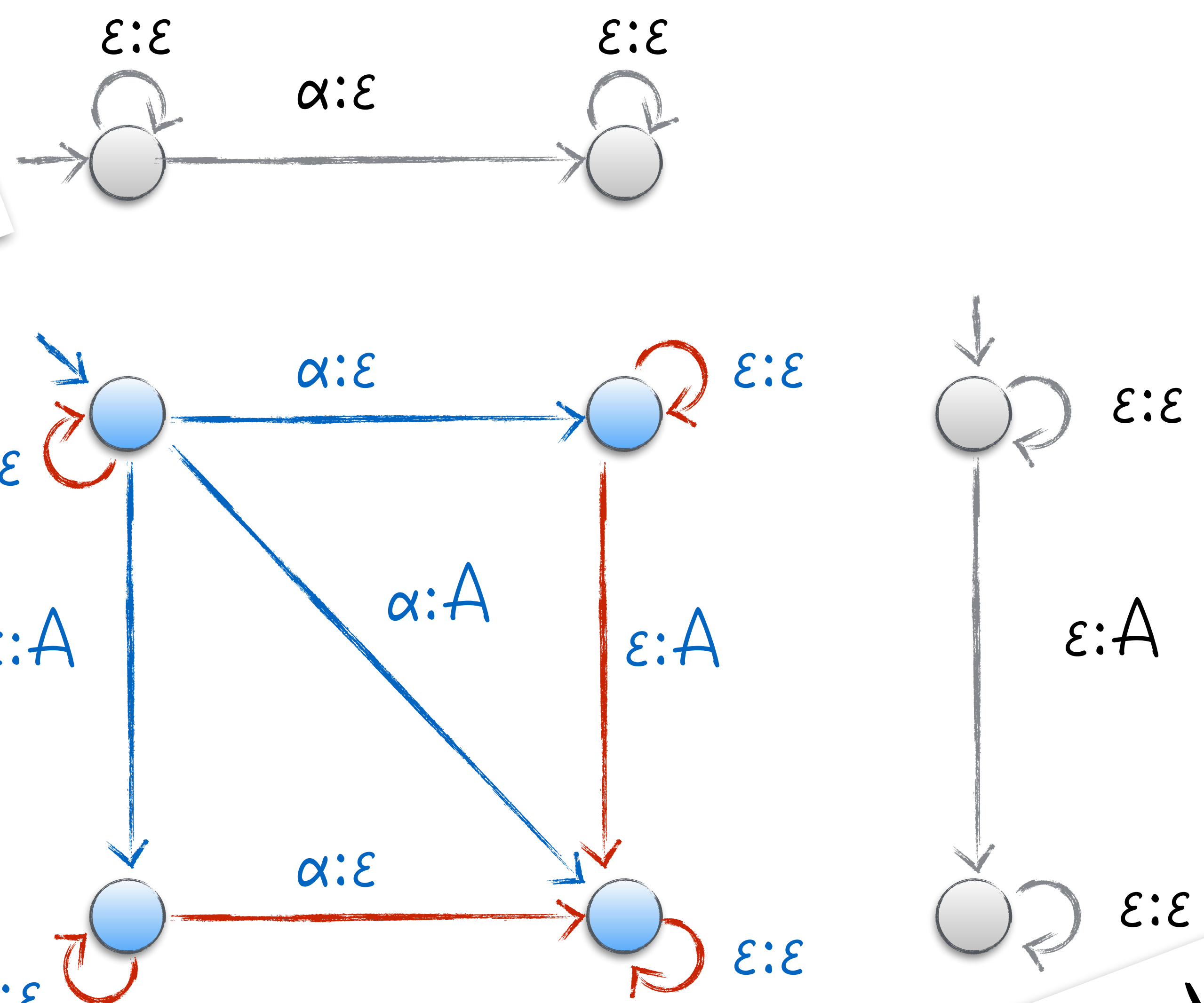
Composition: Example 2

T2 restricts the output string to aab.
T1 ○ T2 is a "lattice".



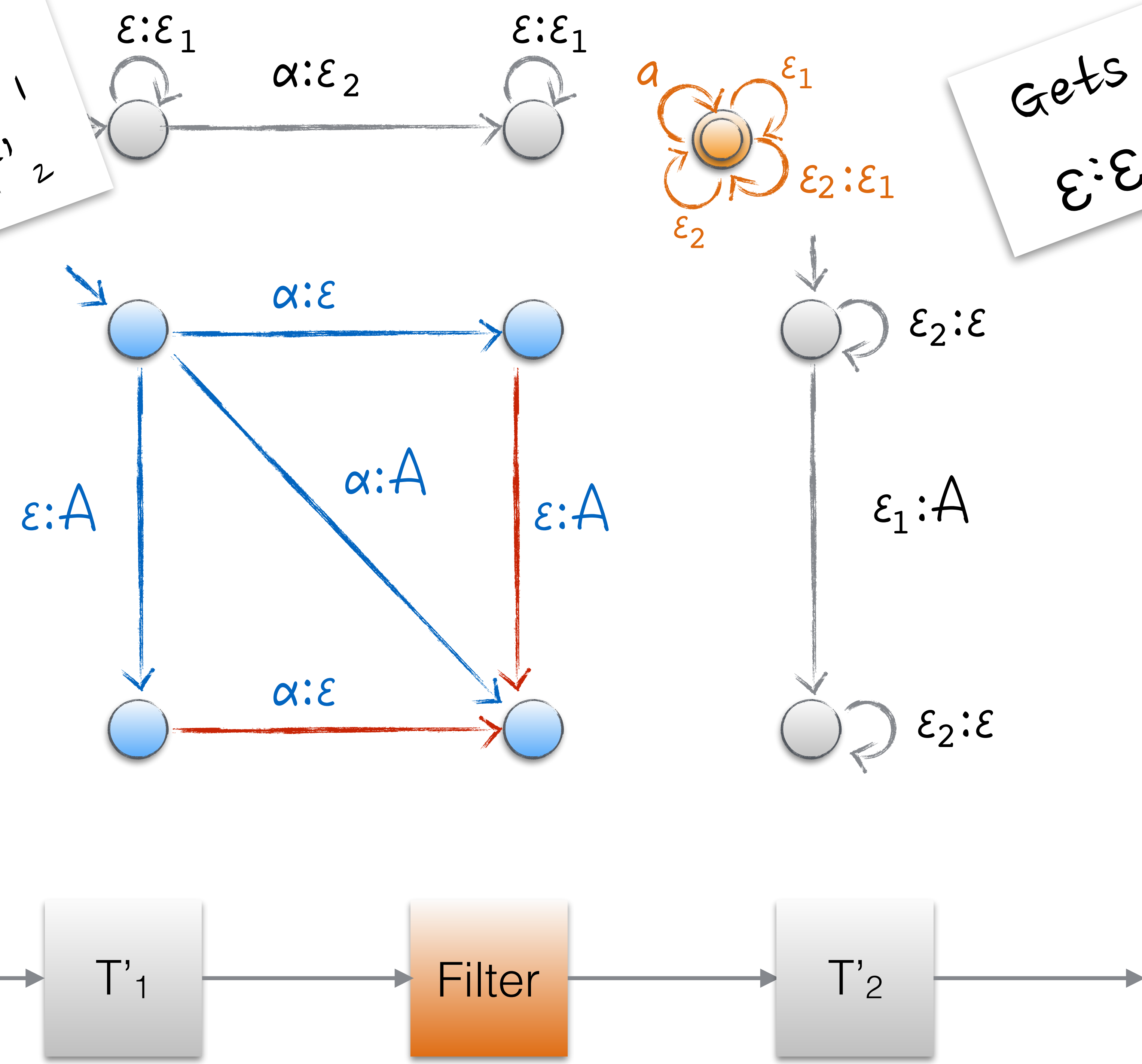
Composition: Handling epsilons

Implicit
ε:ε self-loops
on every state



Duplicate paths!
Weights can be wrong
(in certain semirings)

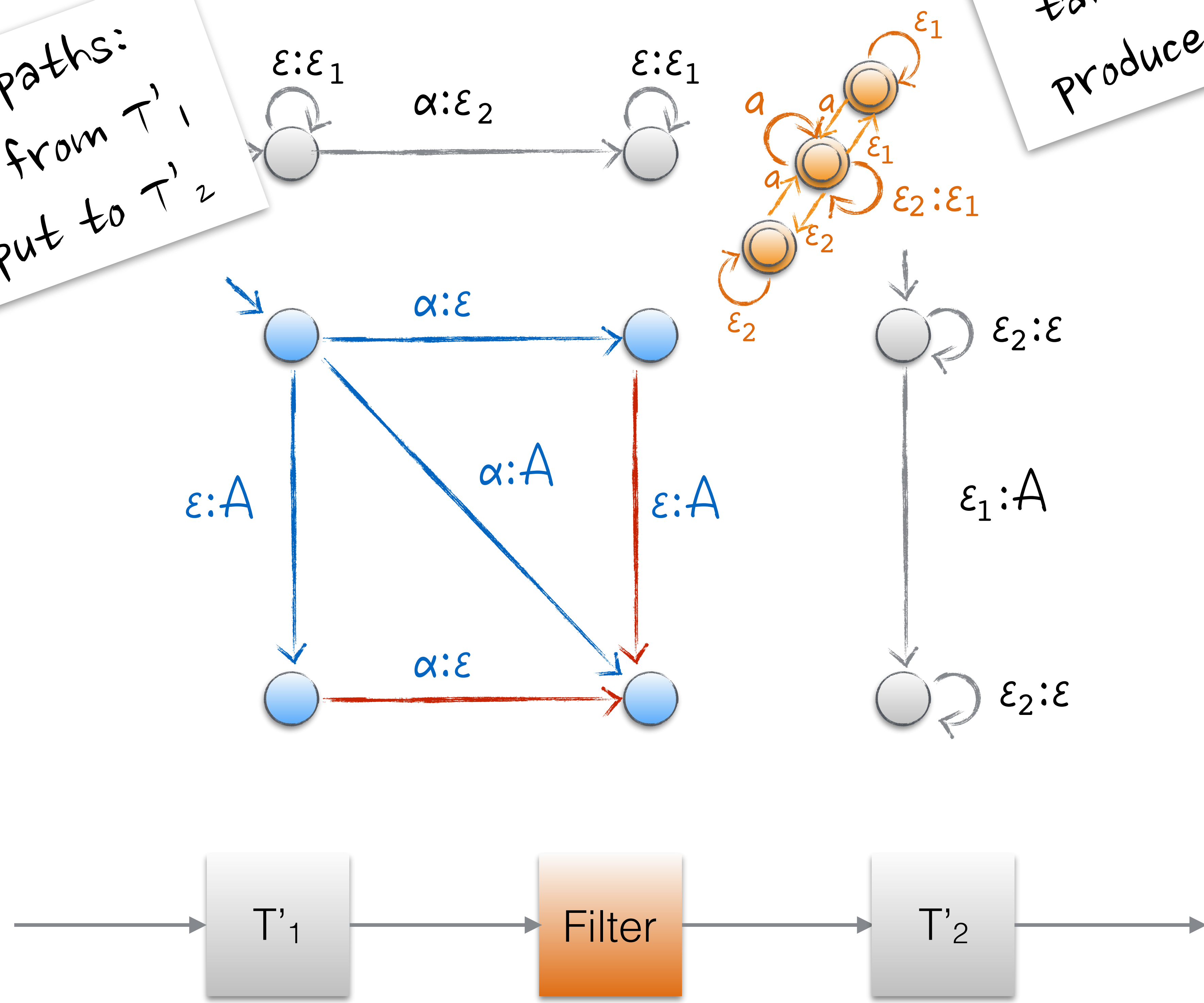
Composition: Filters



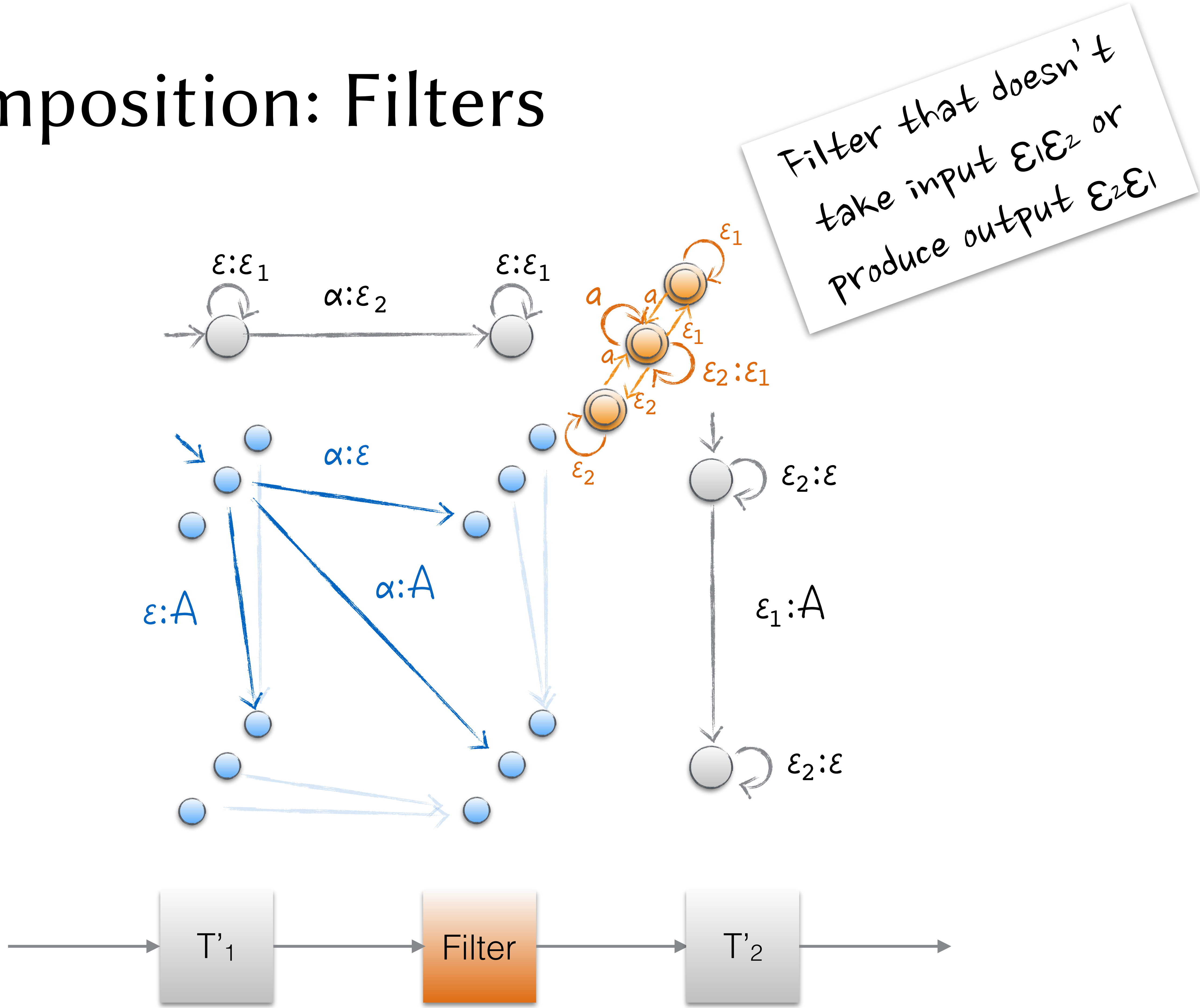
Composition: Filters

Problematic paths:
$\varepsilon_1\varepsilon_2$ output from T'_1
and $\varepsilon_2\varepsilon_1$ input to T'_2

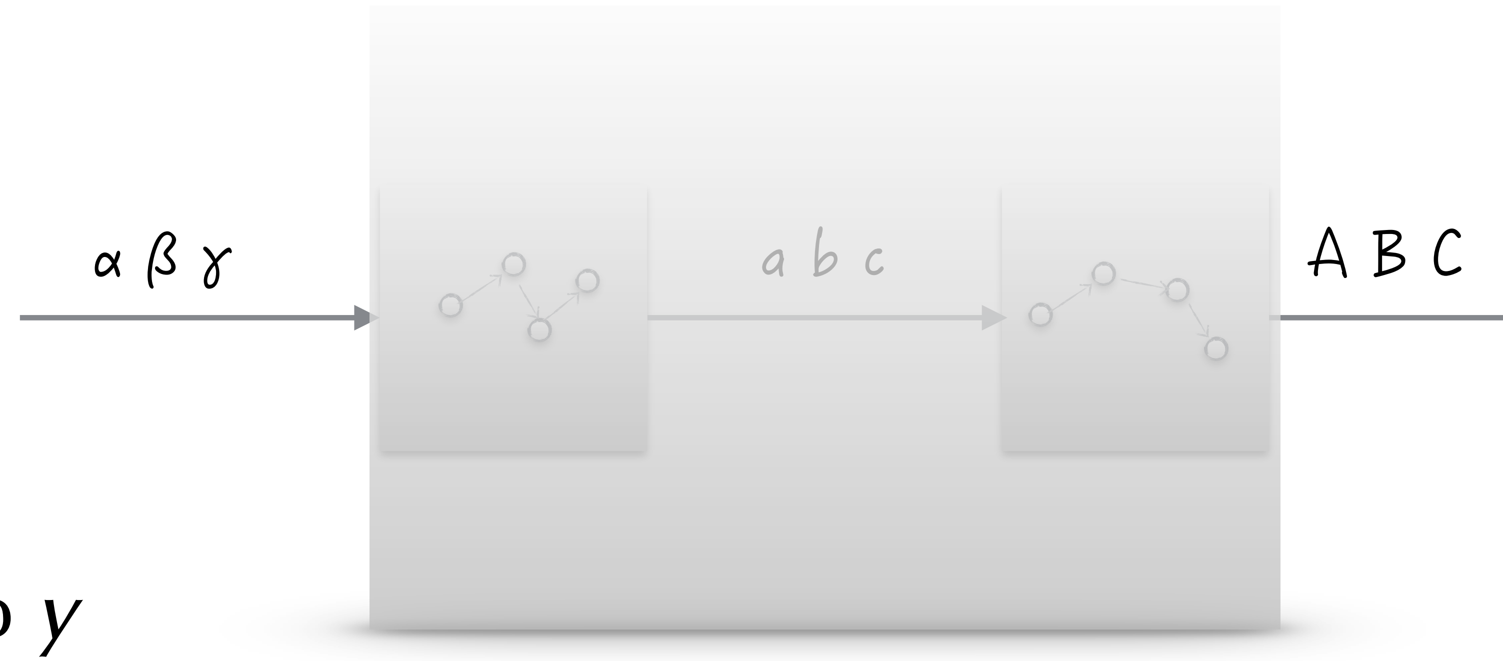
Filter that doesn't
take input $\varepsilon_1\varepsilon_2$ or
produce output $\varepsilon_2\varepsilon_1$



Composition: Filters



Composition: Recap



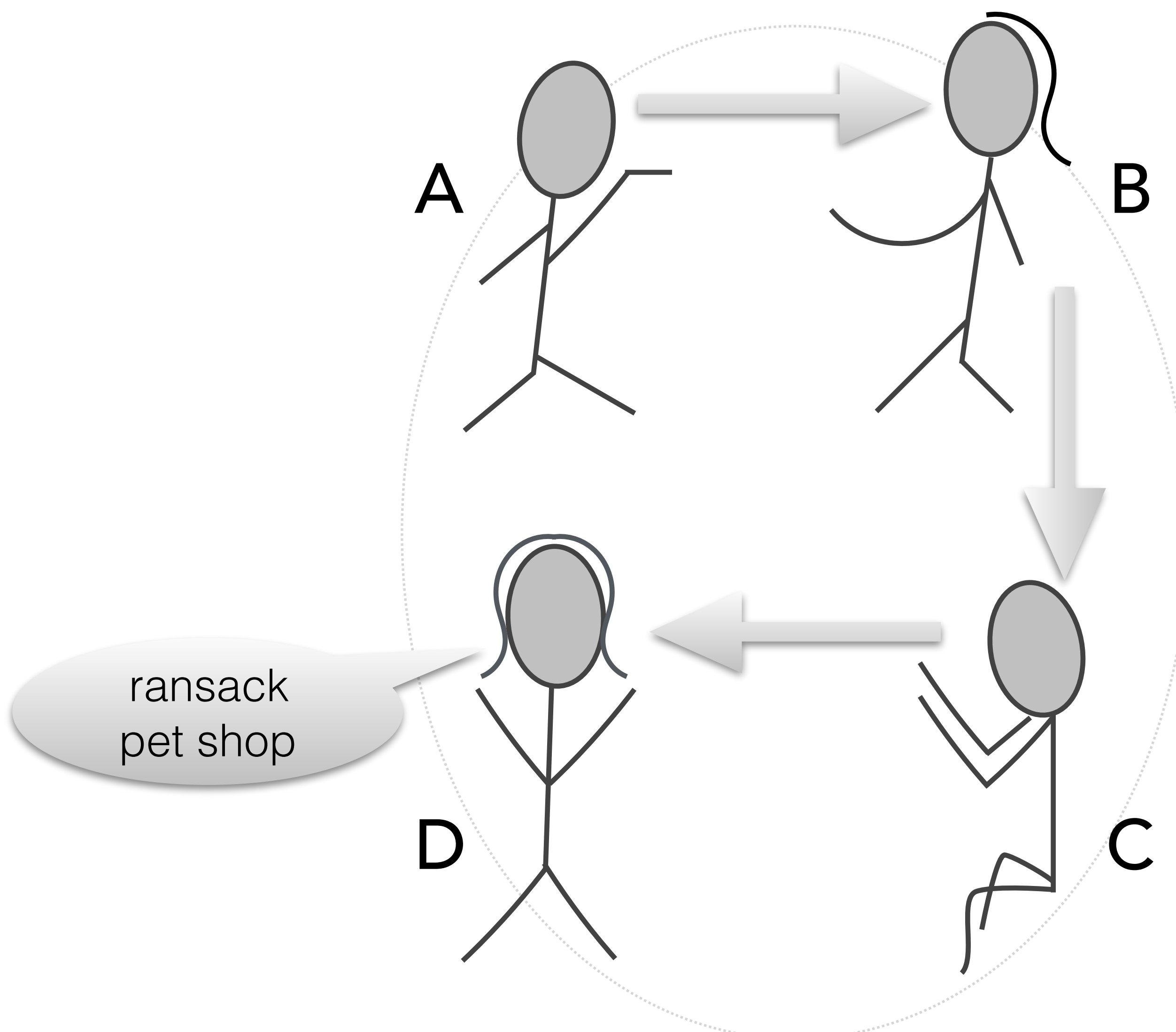
- If T_1 transduces x to z,
  and T_2 transduces z to y,
  then $T_1 \circ T_2$ transduces x to y

- $(T_1 \circ T_2)(x, y) = \bigoplus_{z} T_1(x, z) \otimes T_2(z, y)$

- Note: output alphabet of $T_1 \subseteq$ input alphabet of T_2

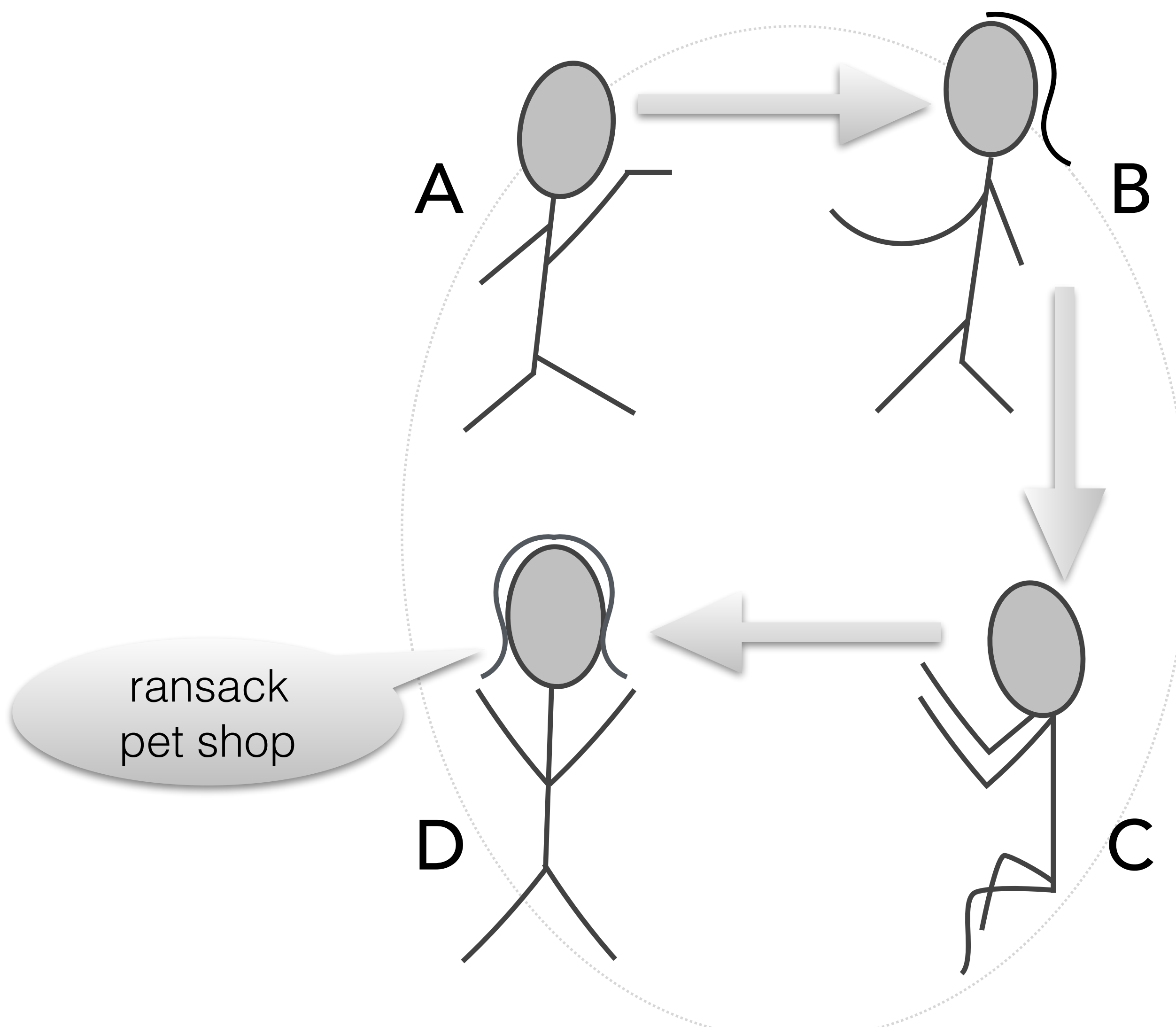
M. Mohri, F. Pereira, and M. Riley. The design principles of a weighted finite-state transducer library. Theoretical Computer Science, 231(1): 17–32, 2000.

The Telephone Game



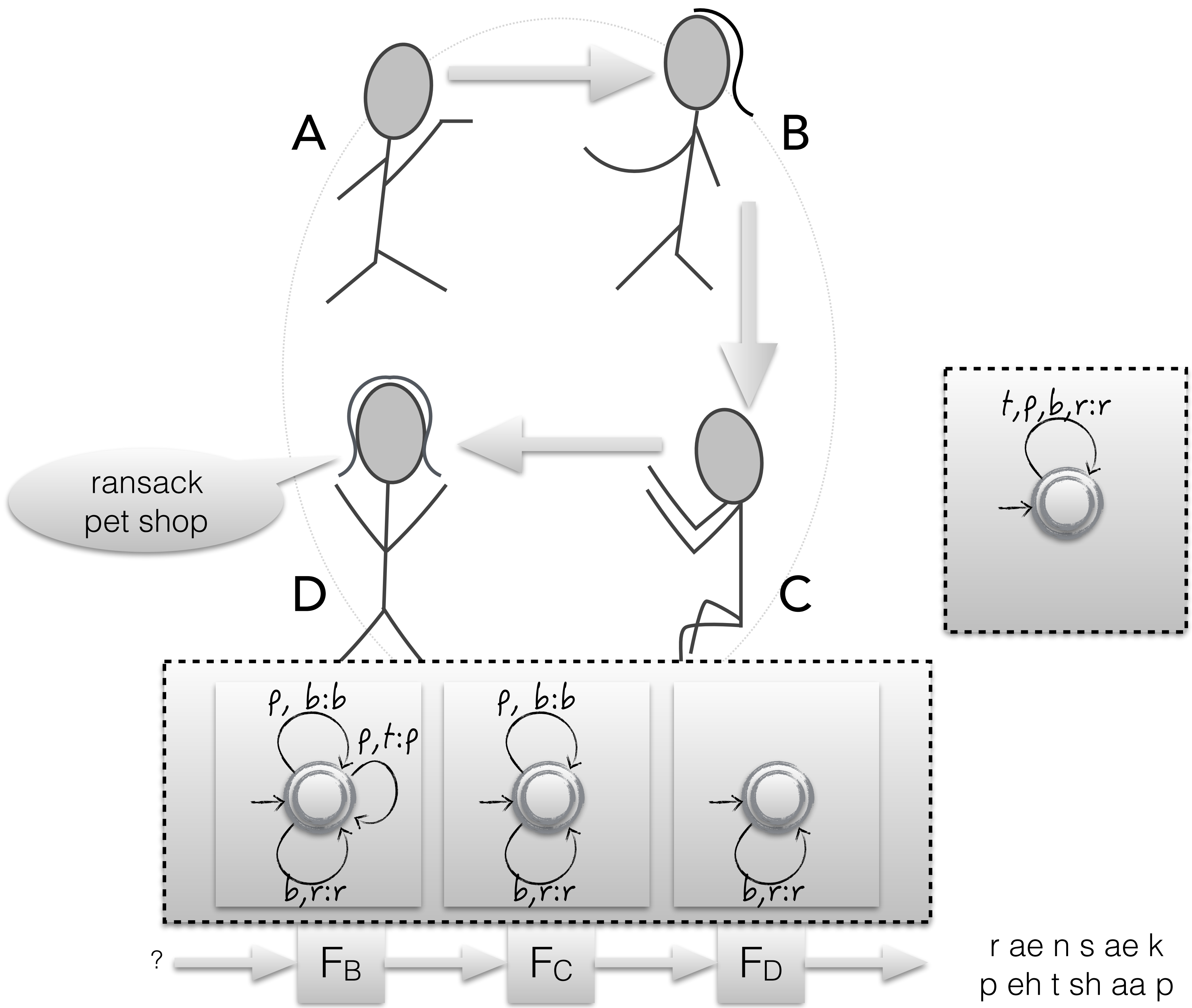
Given what D said,
can we infer the message A started with?

The Telephone Game

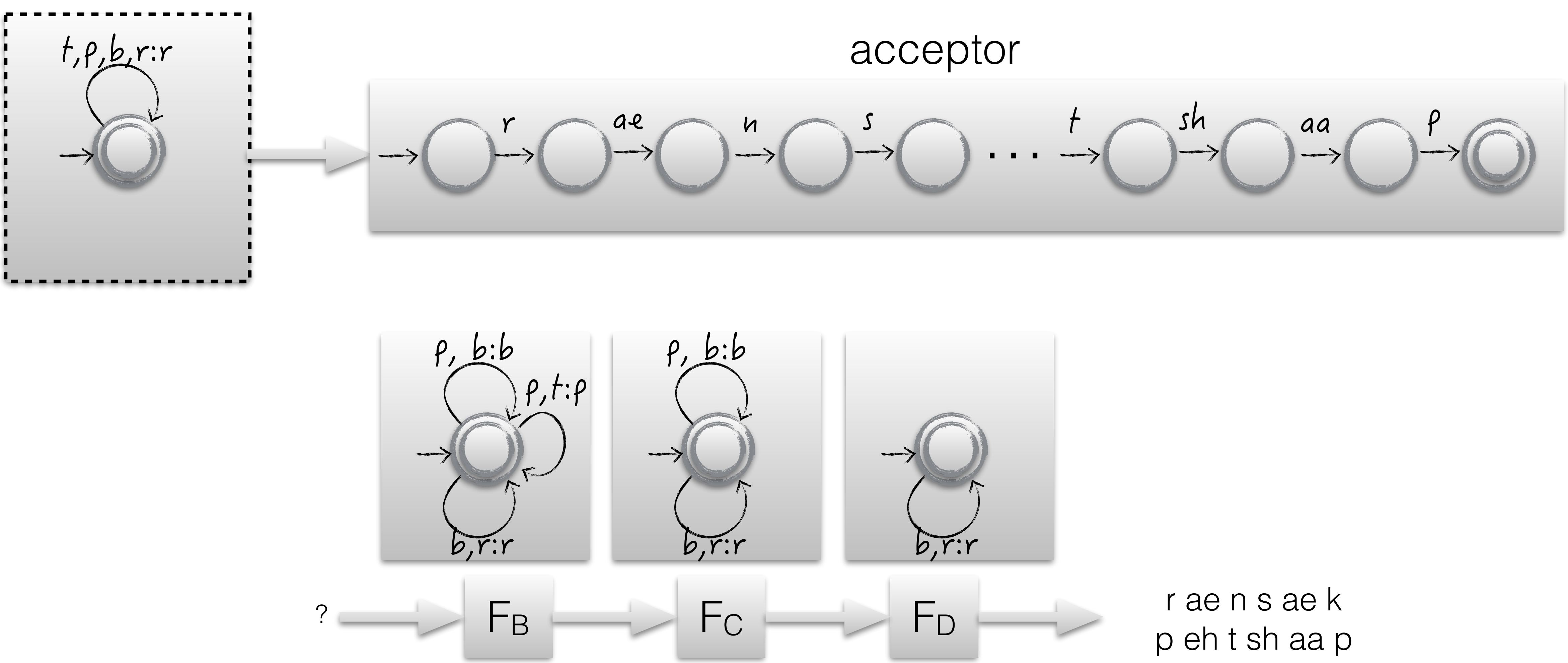


Model the errors made by each player using an FST

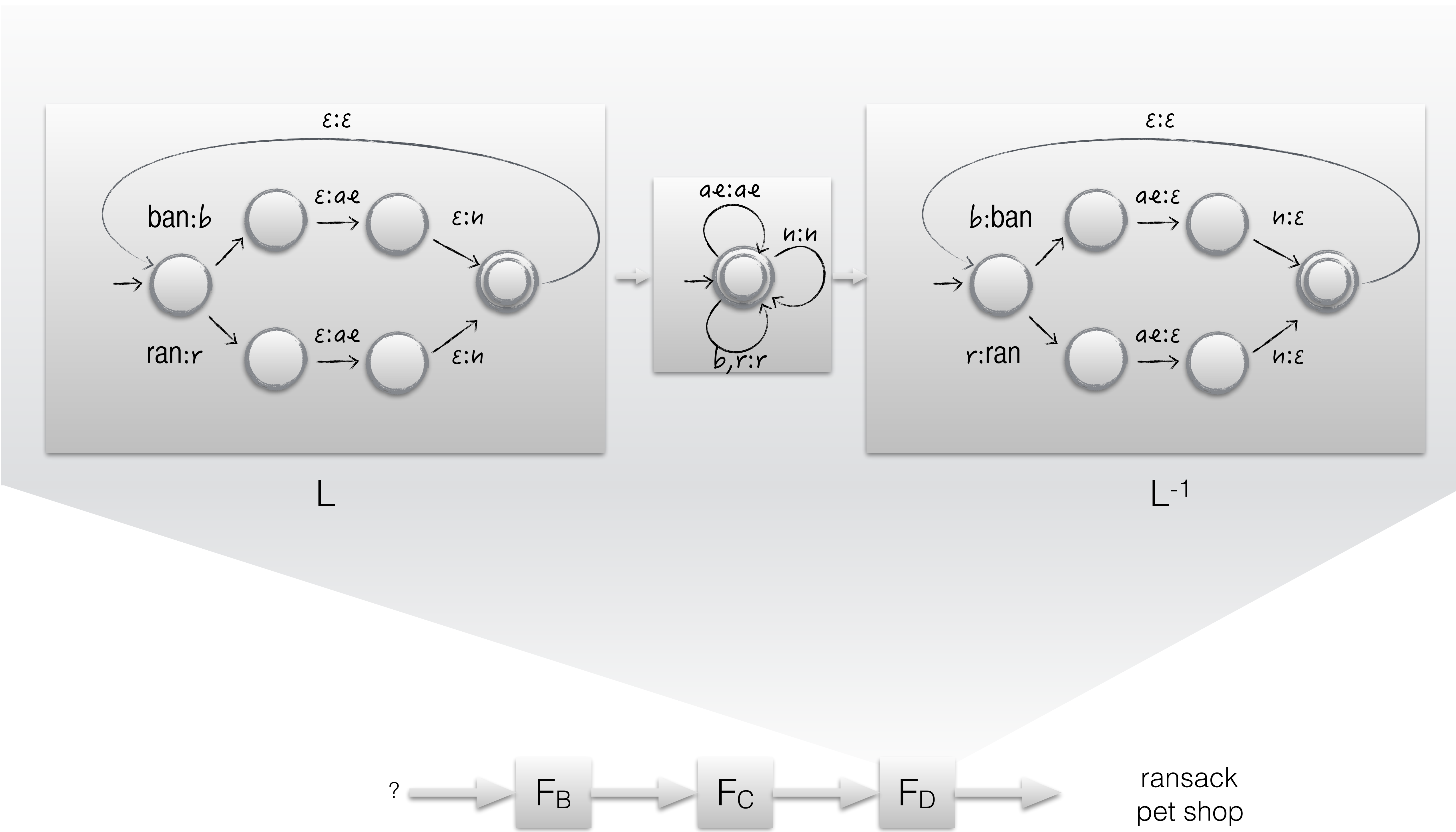
The Telephone Game



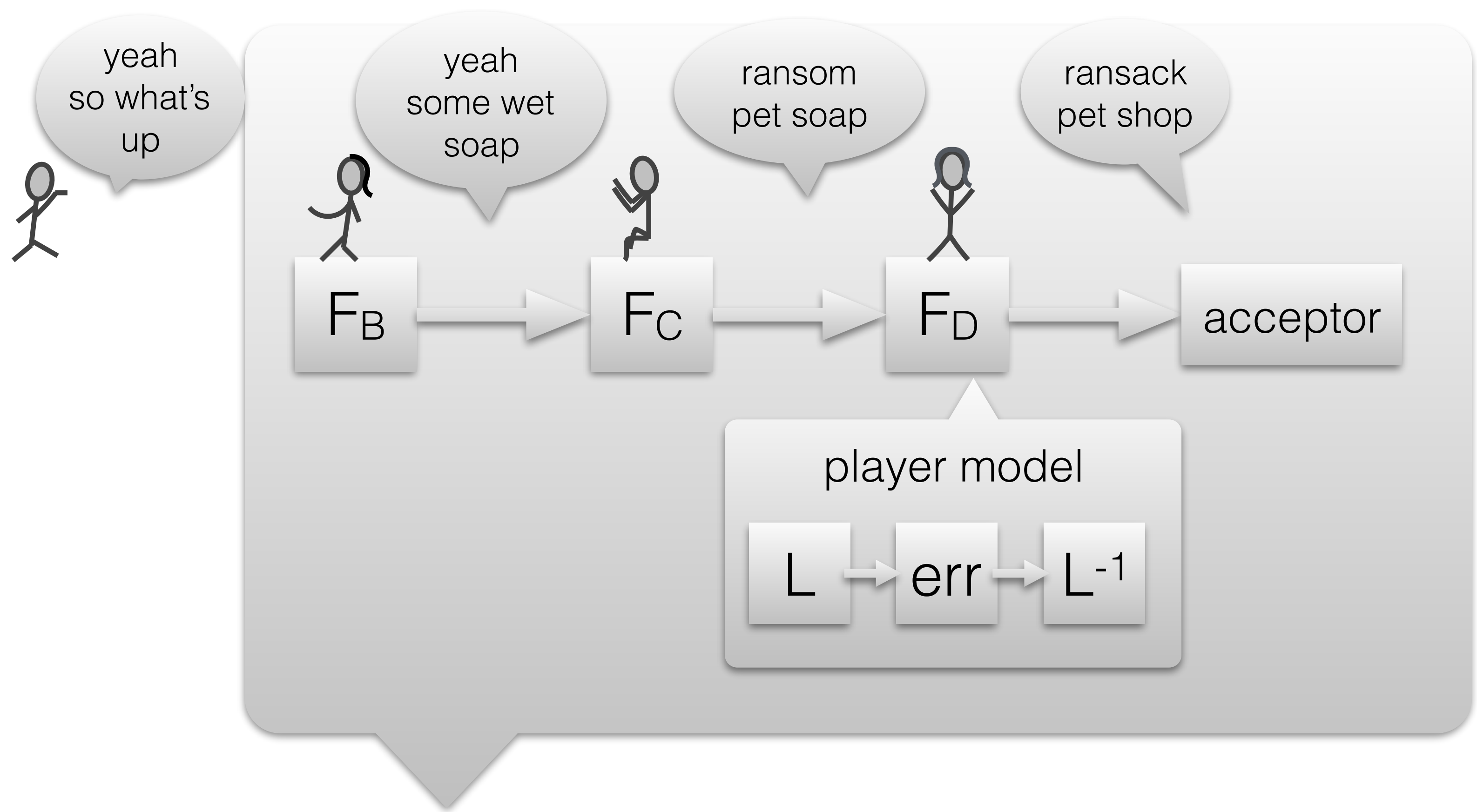
The Telephone Game



The Telephone Game



The Telephone Game



Find the best path in this FST
Read off the input words on the arcs
Can also find the best combination of paths in each player FST