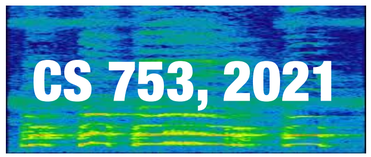
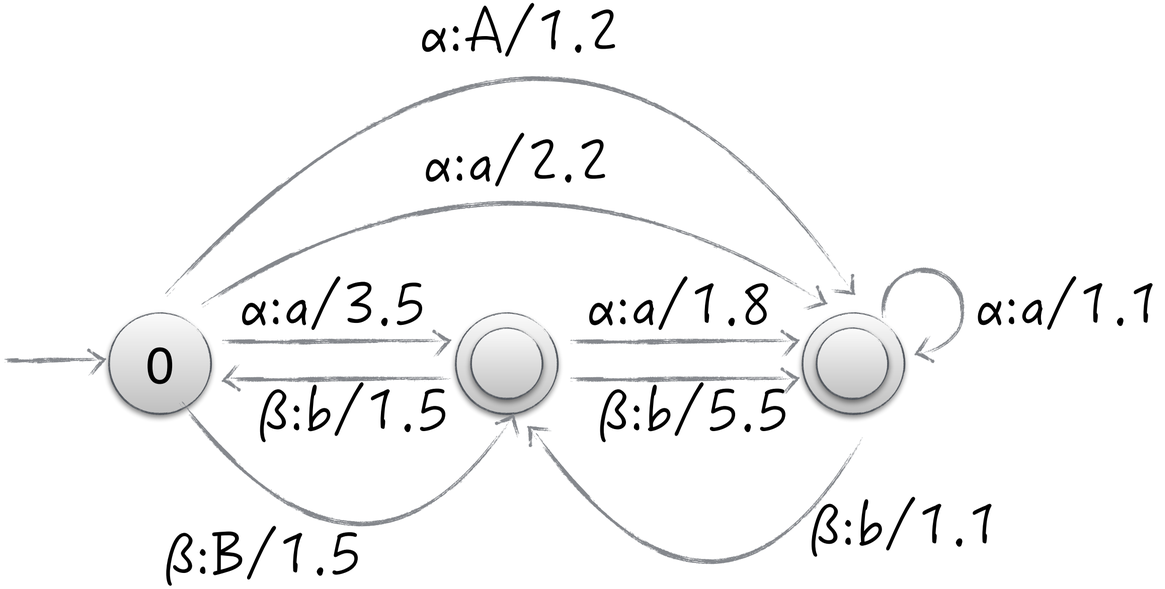
WFST Algorithms

Lecture 3a



Instructor: Preethi Jyothi, IITB

Recall weighted paths in WFSTs



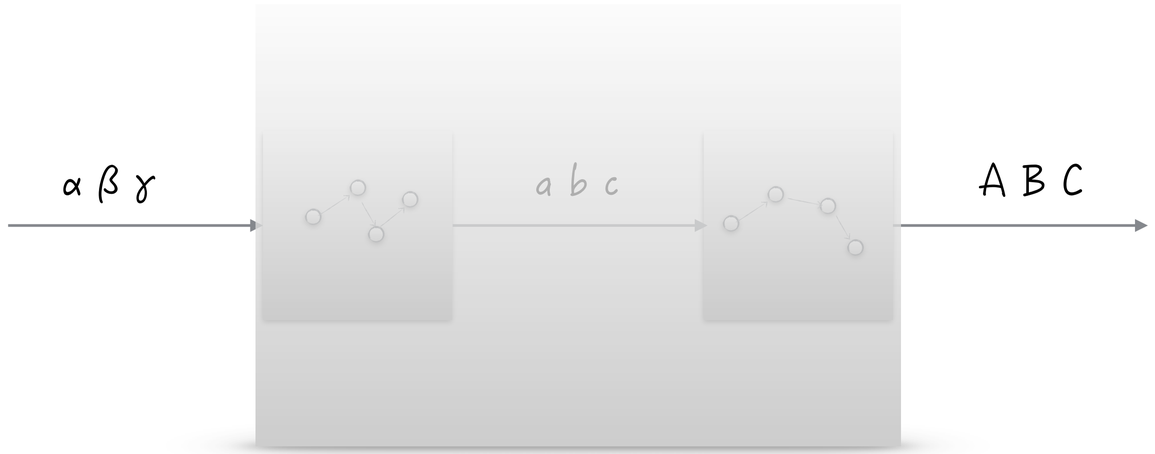
Recall: Path weight with x as input and y as output is $T(x,y) = \bigoplus_{\pi \in P(x,y)} w(\pi)$

where $P(x,y)$ = set of paths with input/output (x,y); $w(\pi) = \bigotimes_{e \in \pi} w(e)$

$\oplus$ and $\otimes$ are two operations associated with the semiring on the weights

- In the tropical semiring $\oplus$ is *min*.
  - $T(x,y)$ associated with a single path in $P(x,y)$: *Shortest Path*

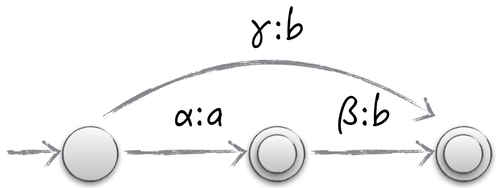
Composition of WFSTs



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

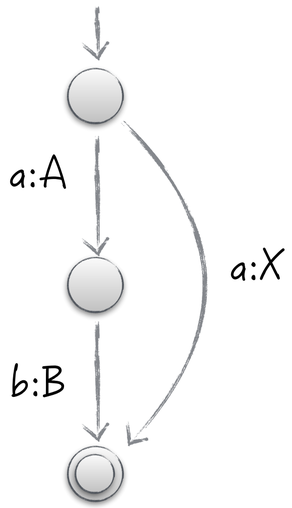
- $(T_1 \circ T_2)(x, y) = \bigoplus_z T_1(x, z) \otimes T_2(z, y)$

Composition: Construction

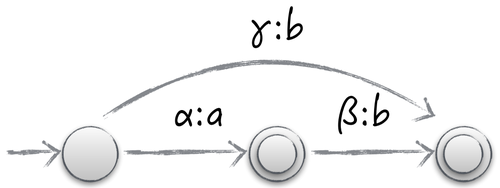


M_1

M_2

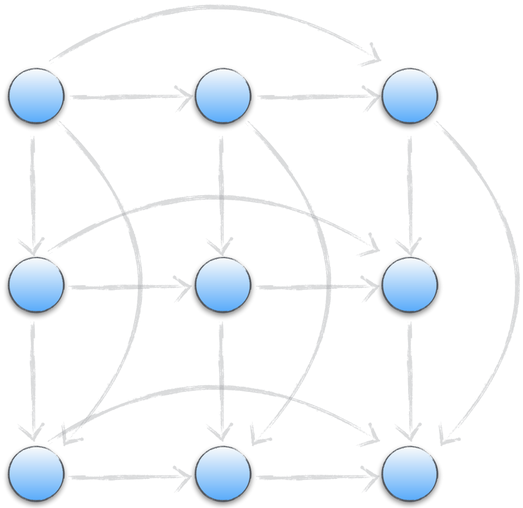


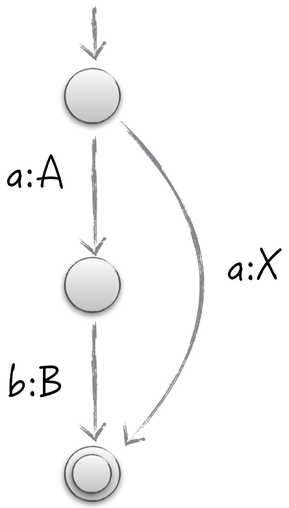
Composition: Construction



M_1

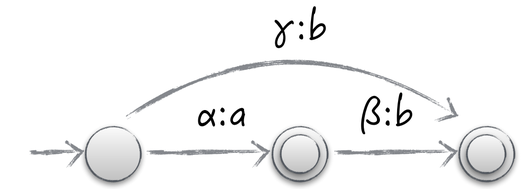
M_2





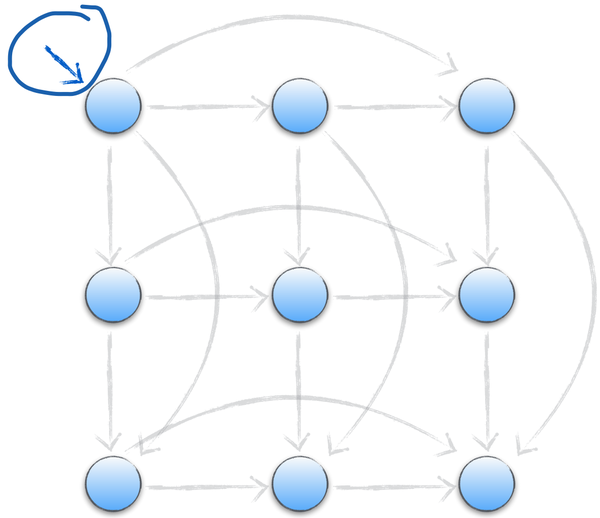
$M_1 \circ M_2$

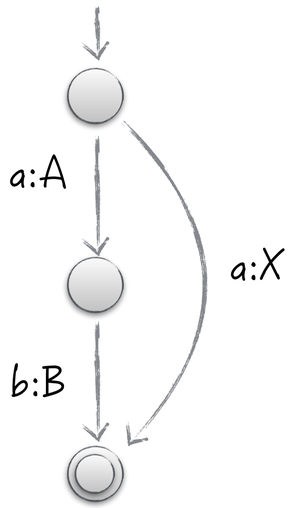
Composition: Construction



M_1

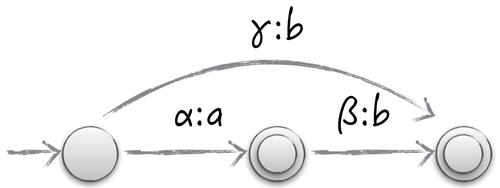
M_2





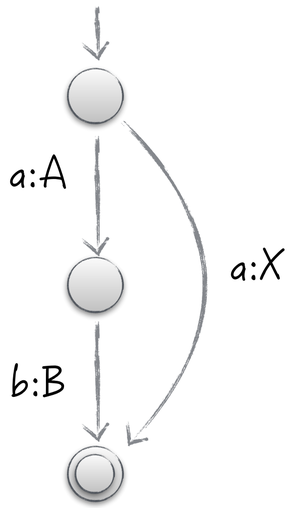
$M_1 \circ M_2$

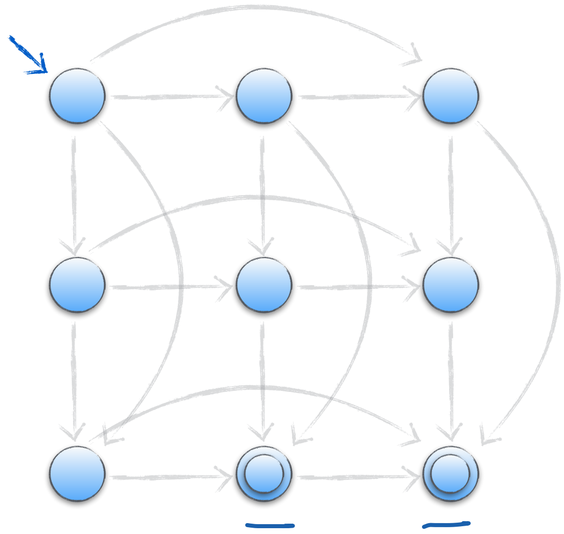
Composition: Construction



M_1

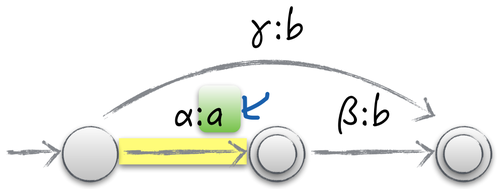
M_2





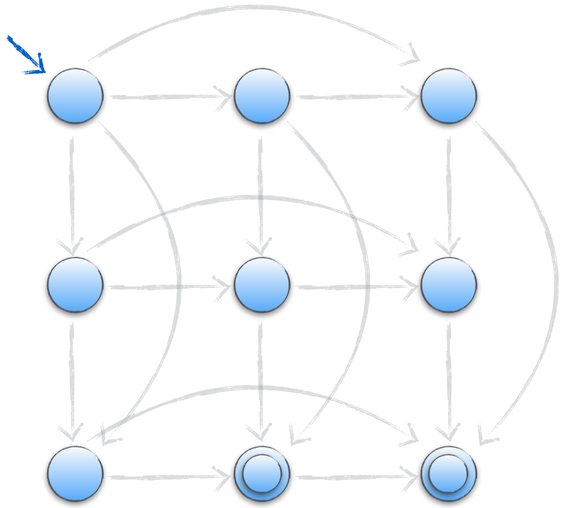
$M_1 \circ M_2$

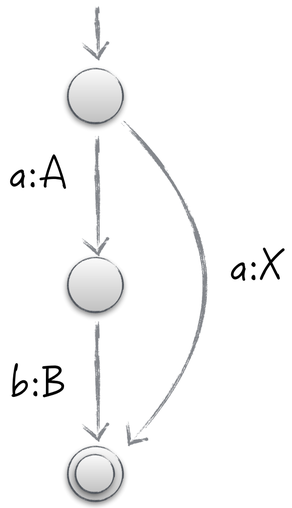
Composition: Construction



M_1

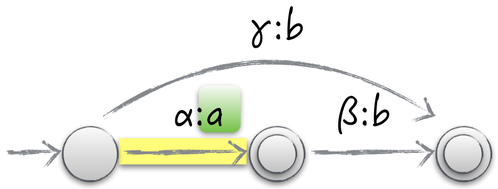
M_2





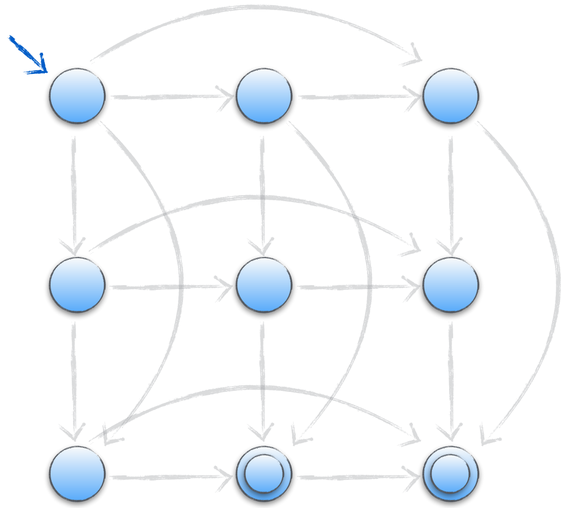
$M_1 \circ M_2$

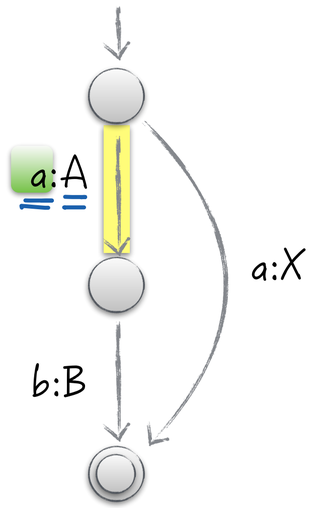
Composition: Construction



M_1

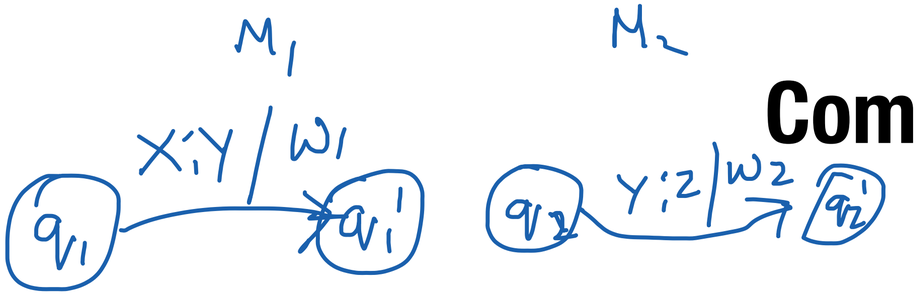
M_2

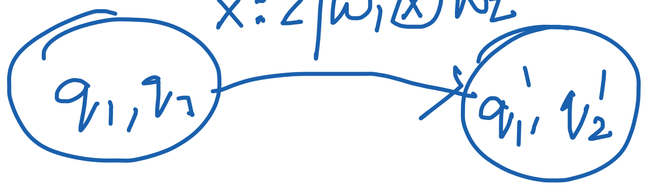


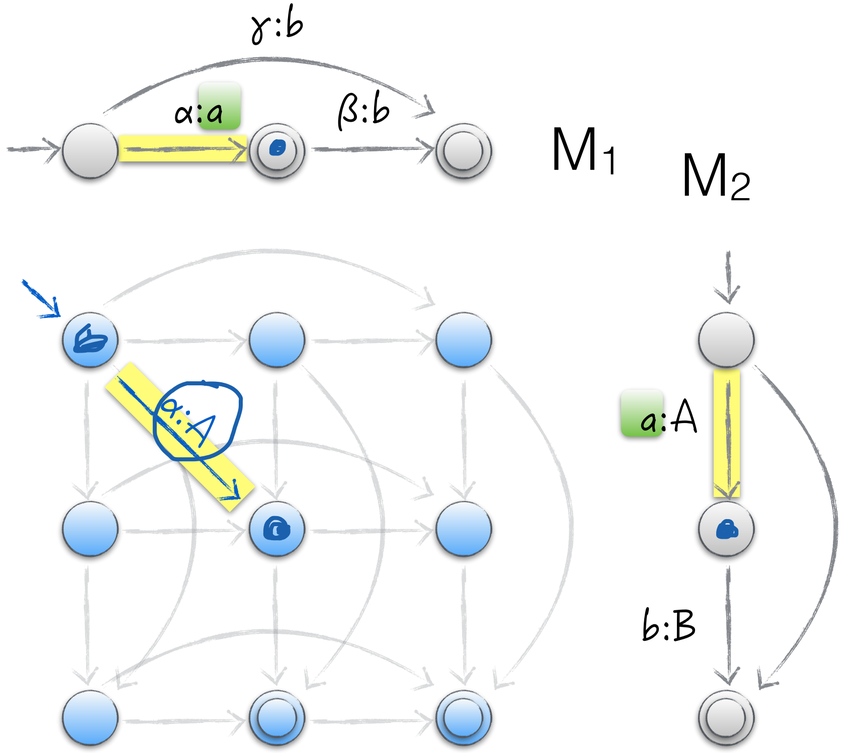


$M_1 \circ M_2$

Composition: Construction







$M_1 \circ M_2$

$e \in E = \overbrace{Q \times (\Sigma \cup \epsilon) \times (\Delta \cup \epsilon) \times \mathbb{R} \times Q}$

Compute a transition in $M_1 \circ M_2$ from transitions of M_1 and M_2:

M_1

(q_1, x, y, w_1, q_1')

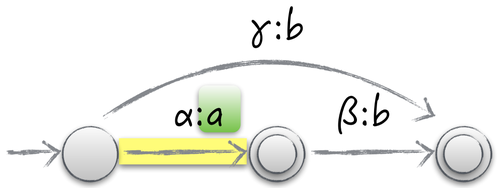
M_2

(q_2, y, z, w_2, q_2')

$M_1 \circ M_2$

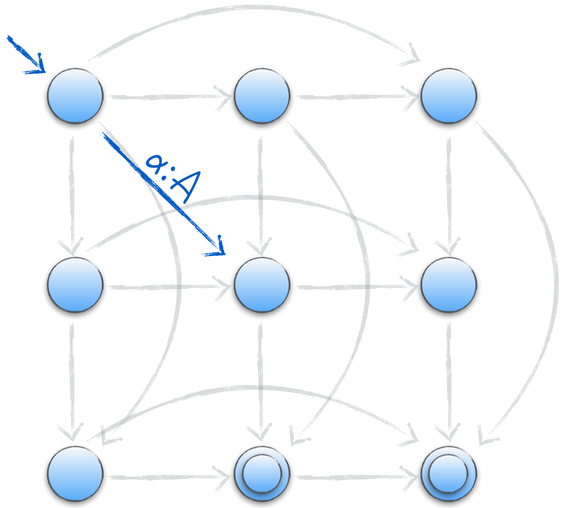
$((q_1, q_2), x, z, w_1 \otimes w_2, (q_1', q_2'))$

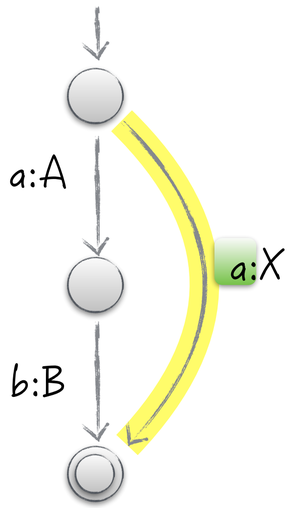
Composition: Construction



M_1

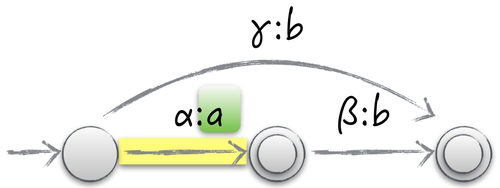
M_2





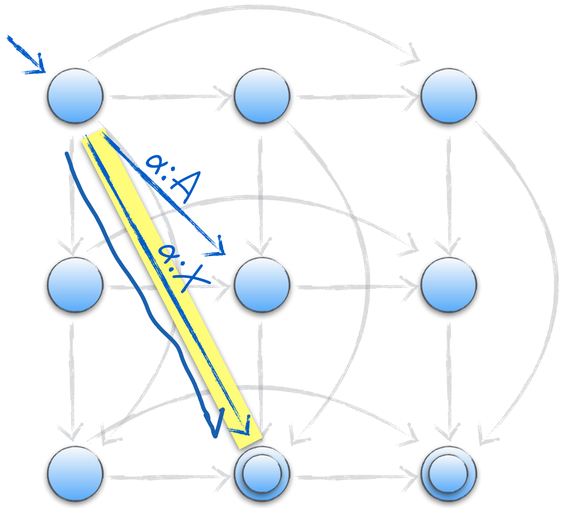
$M_1 \circ M_2$

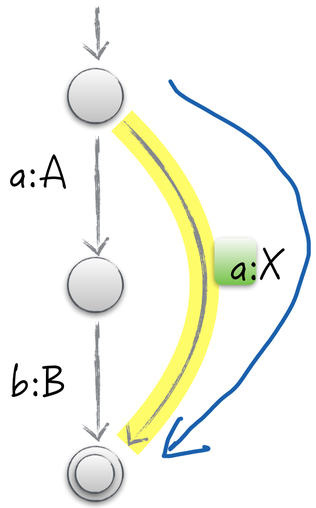
Composition: Construction



M_1

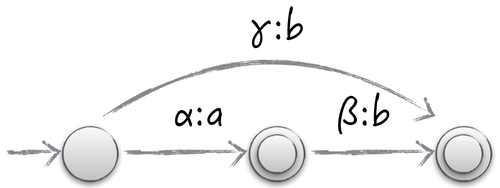
M_2





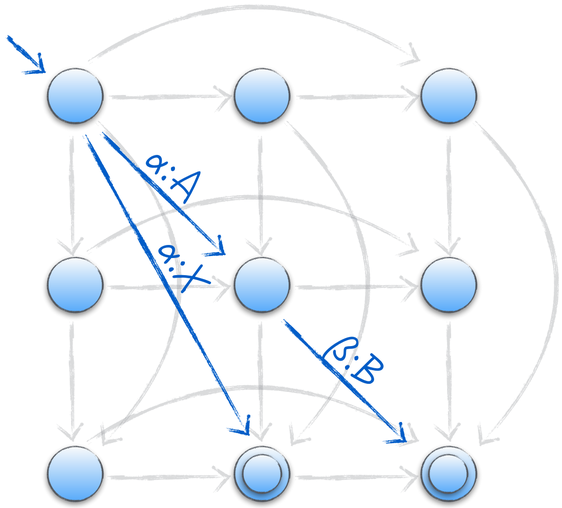
$M_1 \circ M_2$

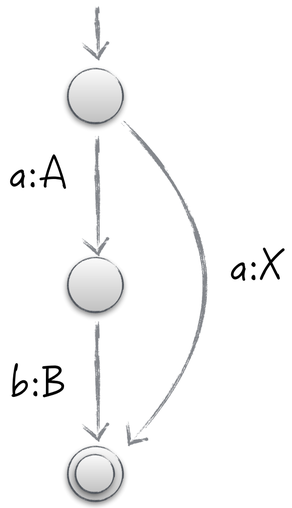
Composition: Construction



M_1

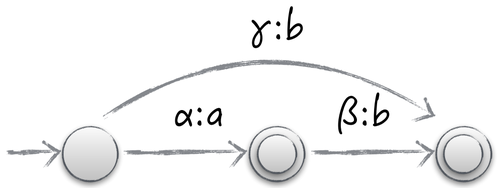
M_2





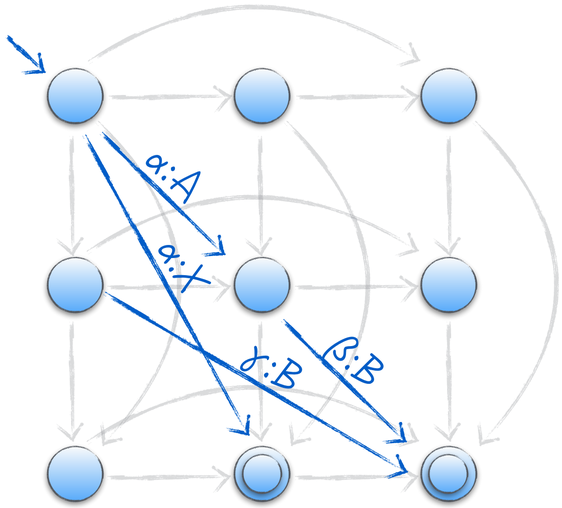
$M_1 \circ M_2$

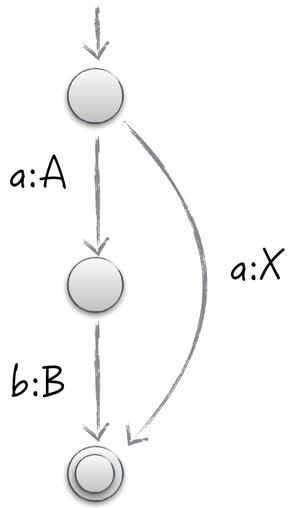
Composition: Construction



M_1

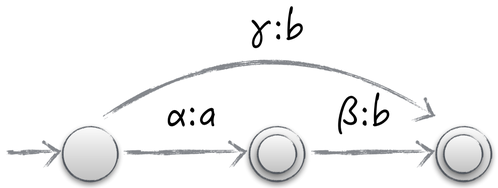
M_2





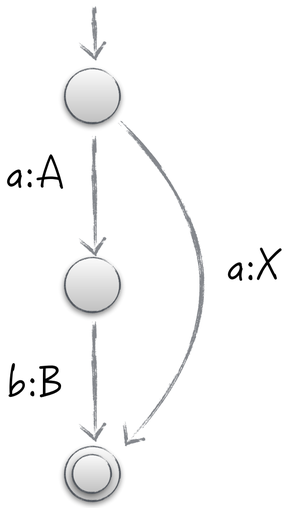
$M_1 \circ M_2$

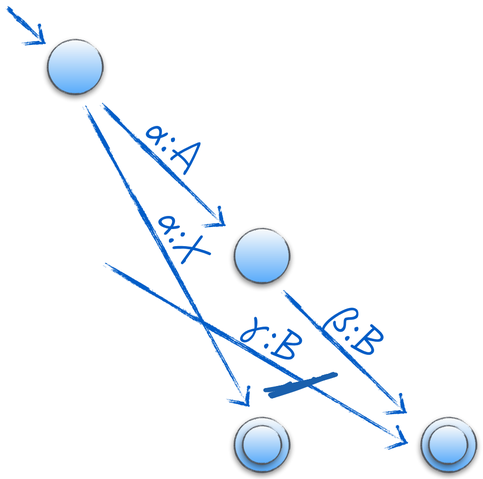
Composition



M_1

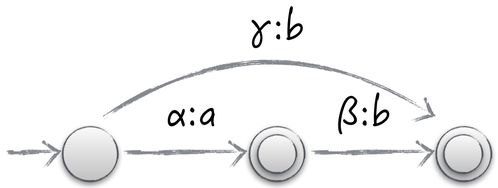
M_2





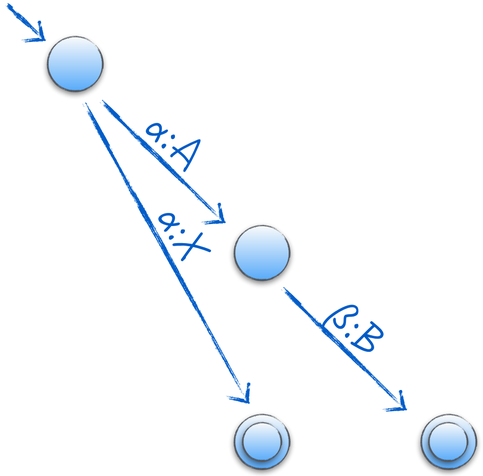
$M_1 \circ M_2$

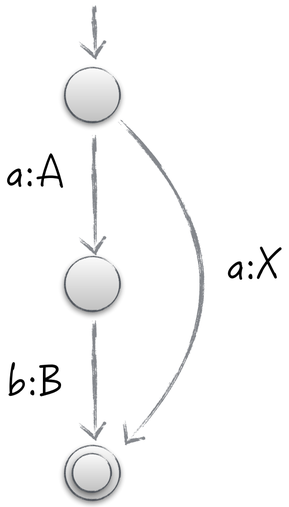
Composition



M_1

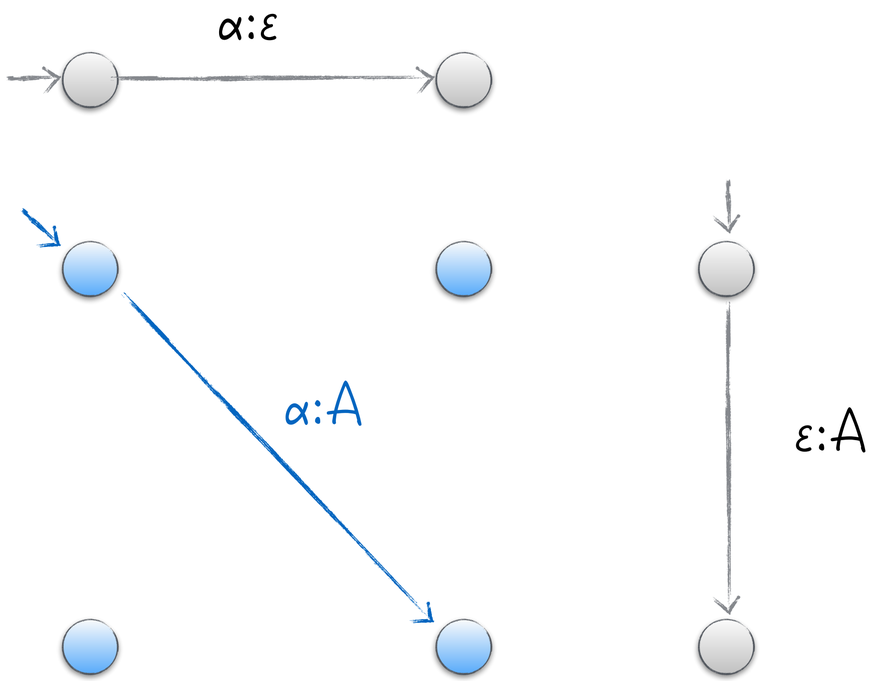
M_2



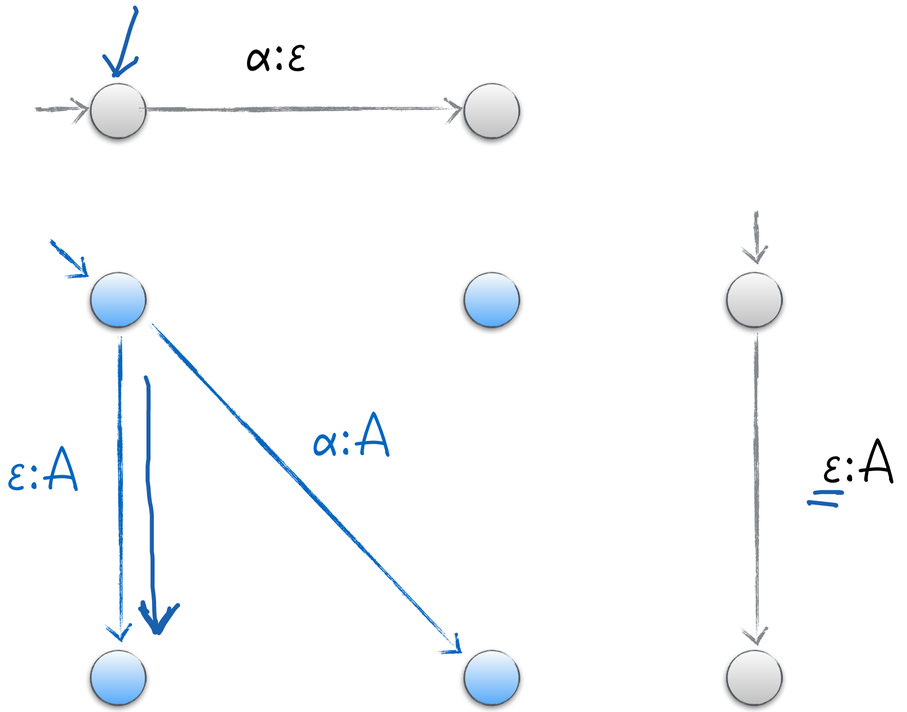


$M_1 \circ M_2$

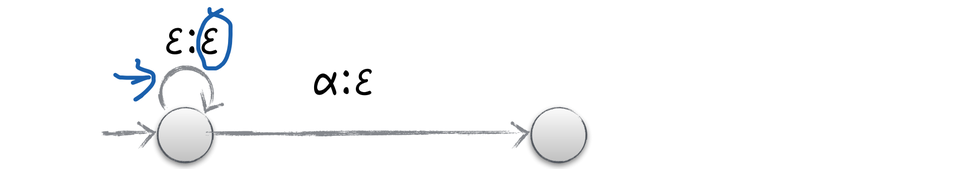
Composition: Handling epsilons

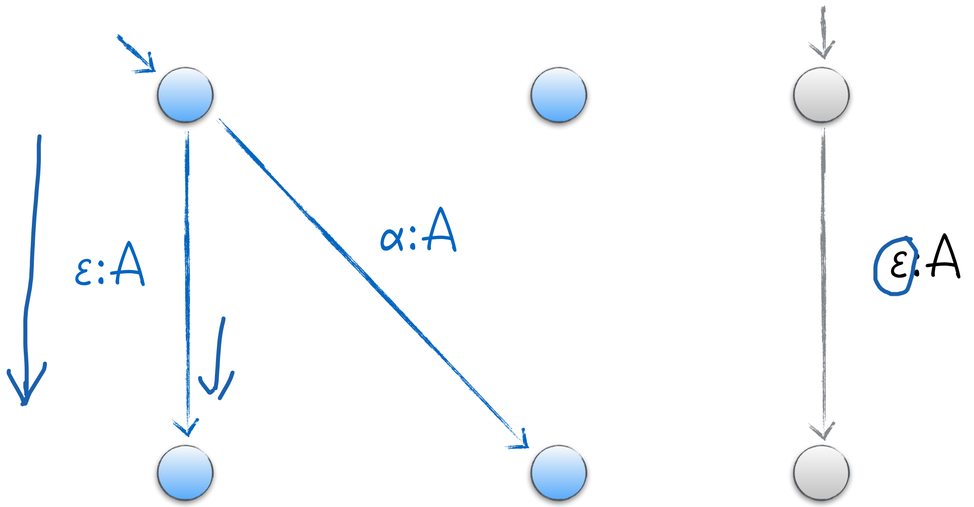


Composition: Handling epsilons



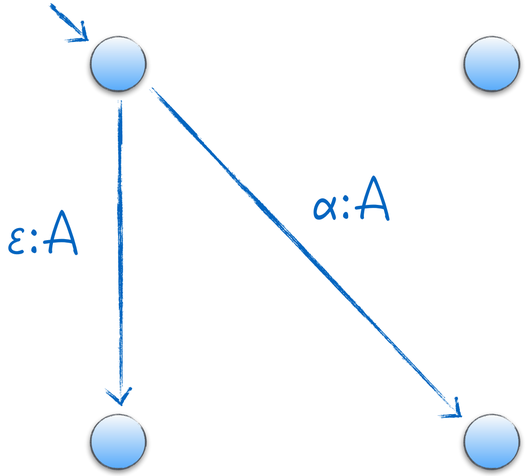
Composition: Handling epsilons

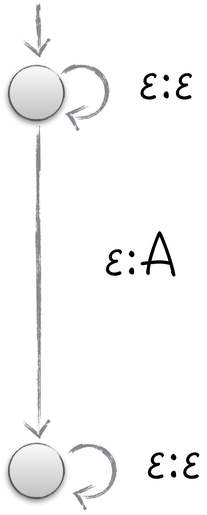




Composition: Handling epsilons

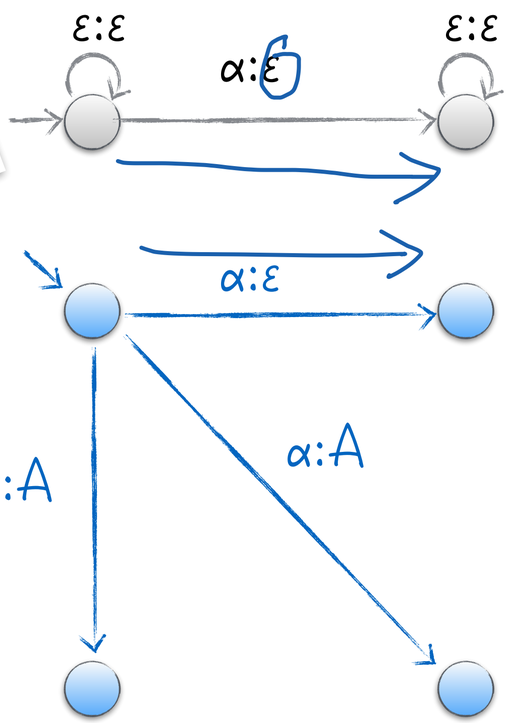
Implicit
ε:ε self-loops
on every state

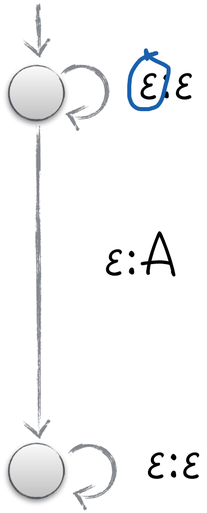




Composition: Handling epsilons

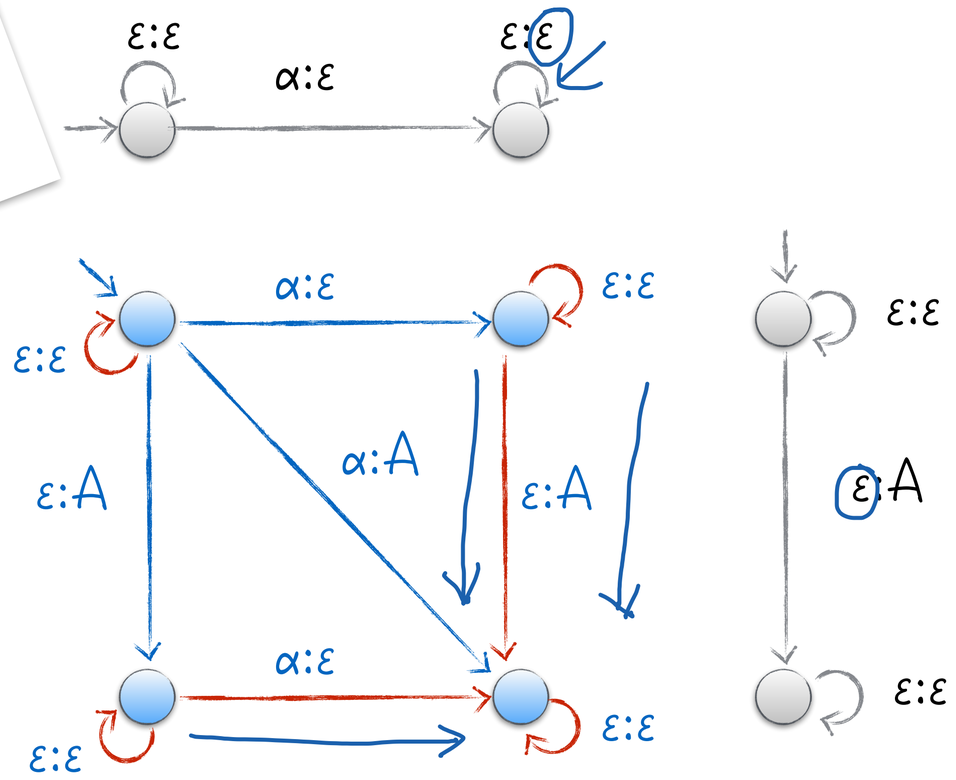
Implicit
ε:ε self-loops
on every state





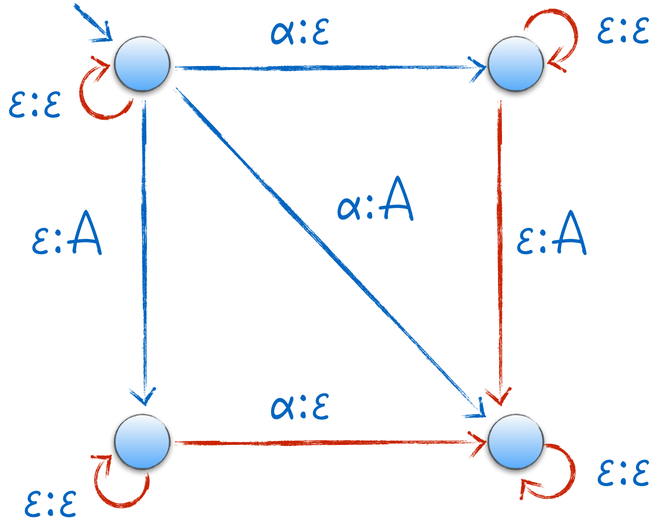
Composition: Handling epsilons

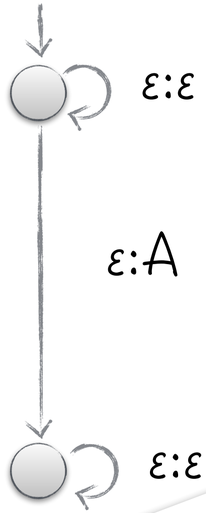
Implicit
ε:ε self-loops
on every state



Composition: Handling epsilons

Implicit
ε:ε self-loops
on every state





non-idempotent

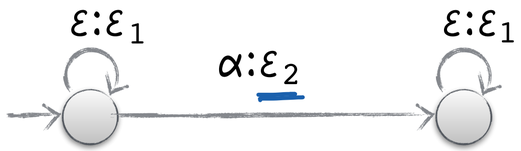
$a \oplus a = a \not\forall a$

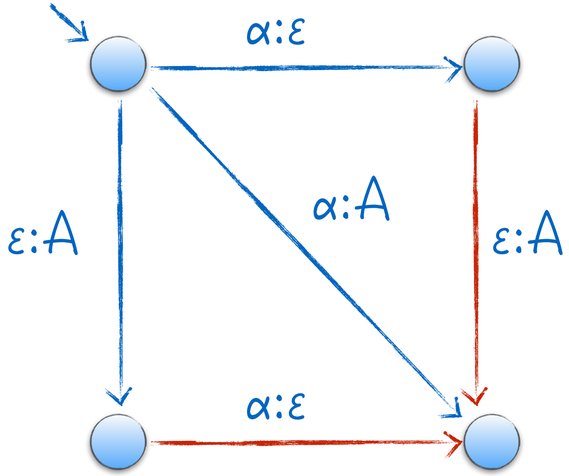
idempotent

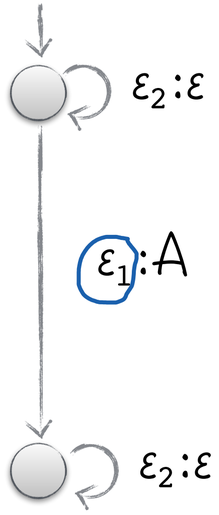
Duplicate paths!
Weights can be wrong
(in certain semirings)

Composition: Filters

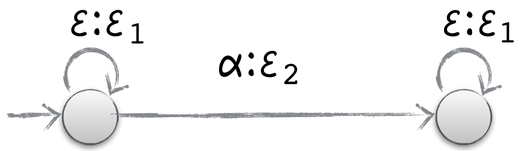
Distinguish epsilons

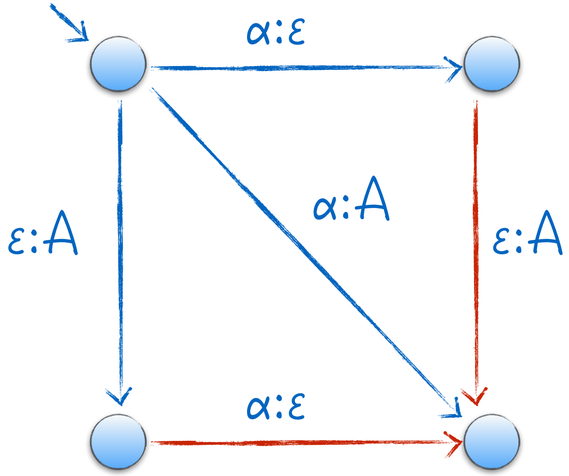


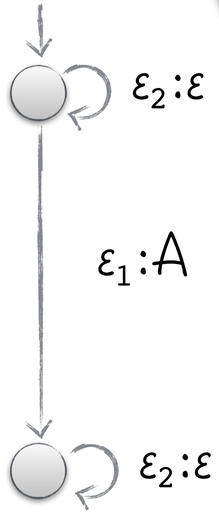




Composition: Filters



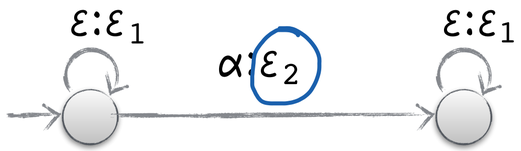




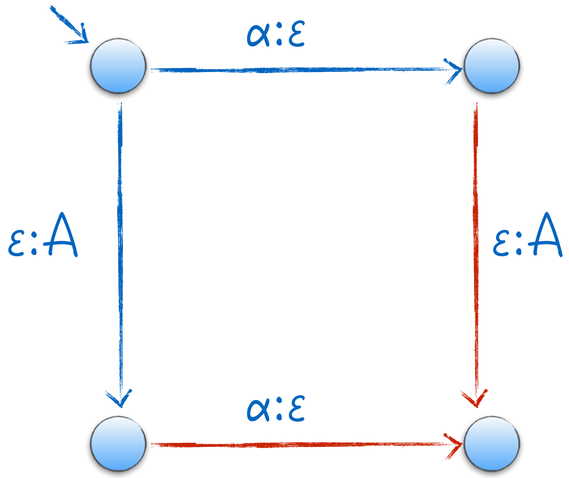
Gets rid of the ε:ε self-loops

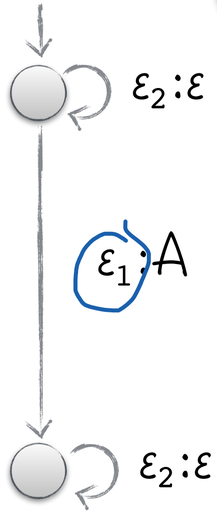
Composition: Filters

M_1'



Gets rid of the ε:ε self-loops

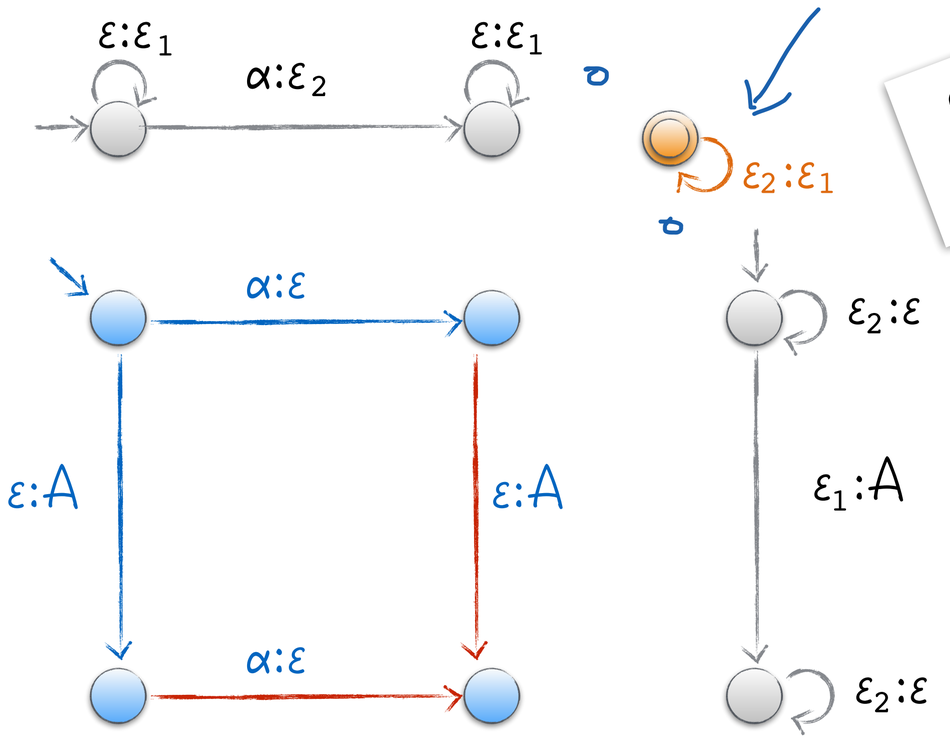




M_2'

$M_1' \circ M_2'$

Composition: Filters



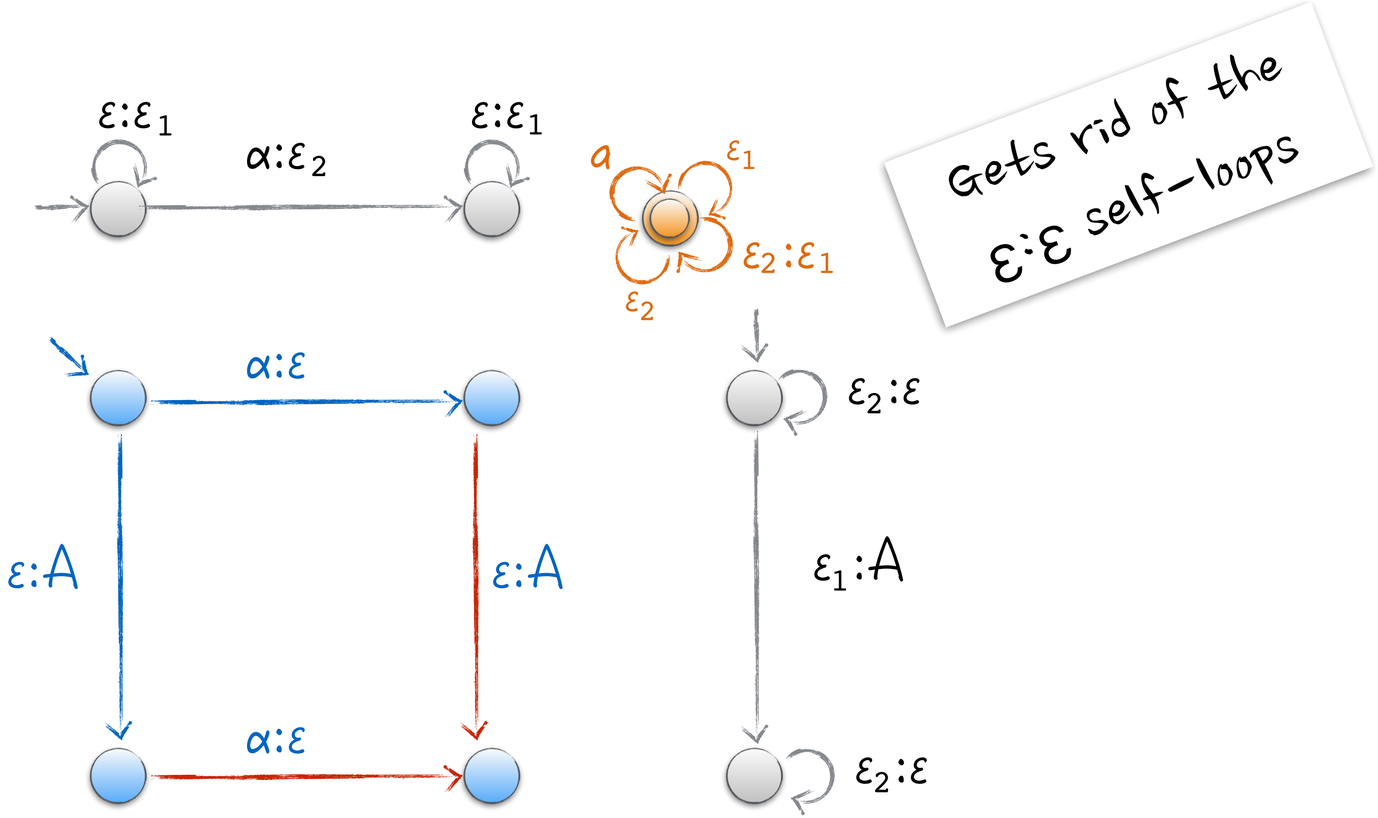
Gets rid of the ε:ε self-loops

$M_1 \circ M_2$

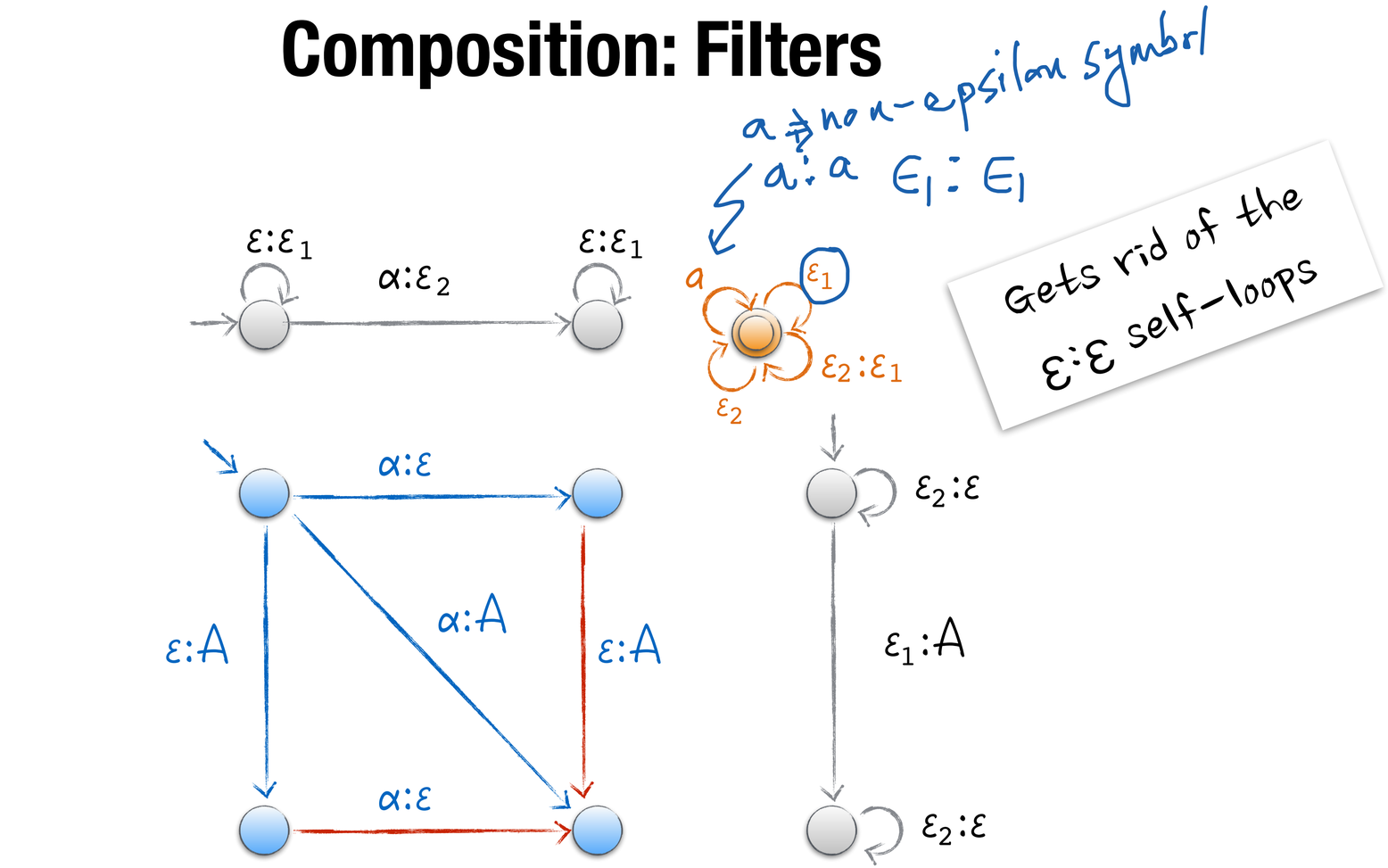
$\Downarrow$

$M_1^{\circ} \circ F \circ M_2^{\circ}$

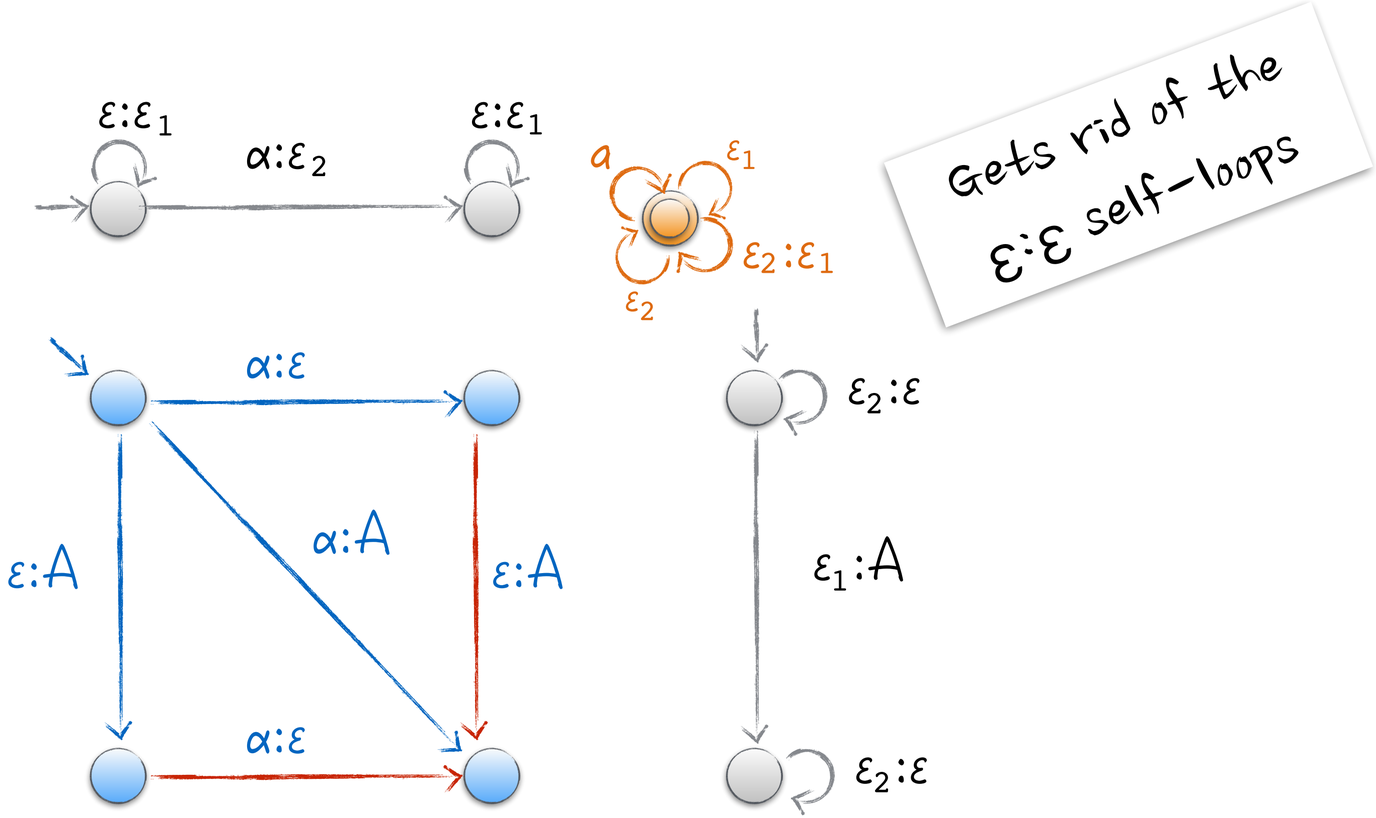
Composition: Filters



Composition: Filters

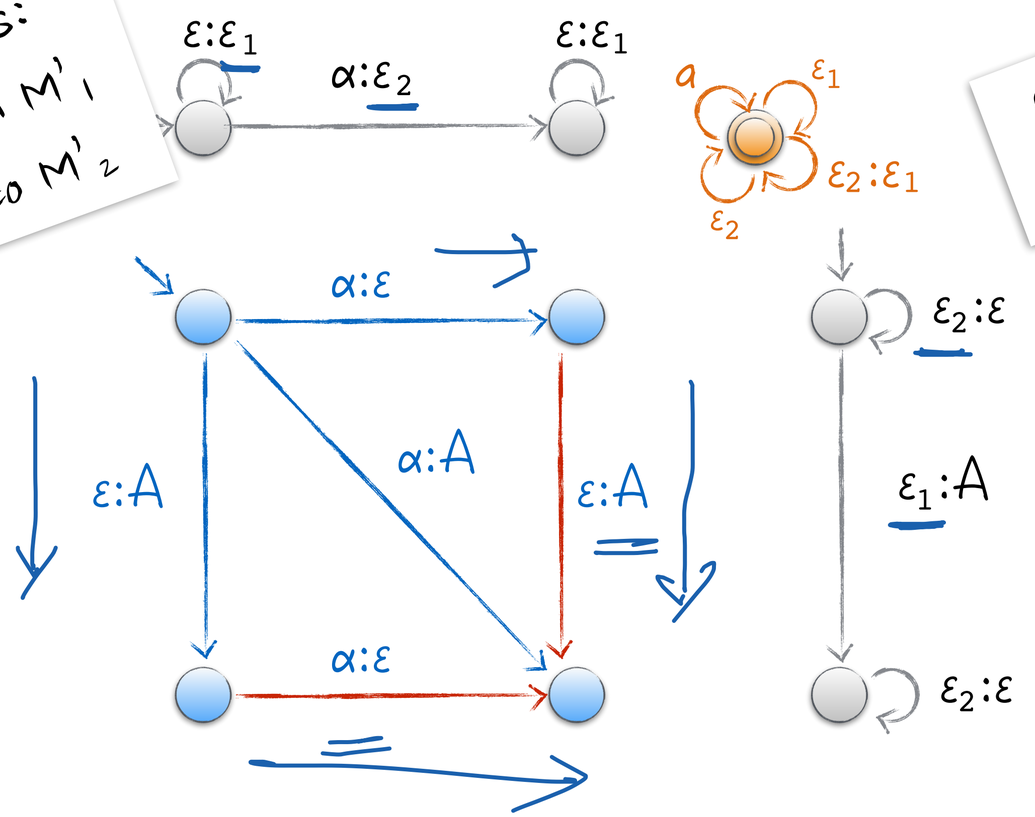


Composition: Filters



Composition: Filters

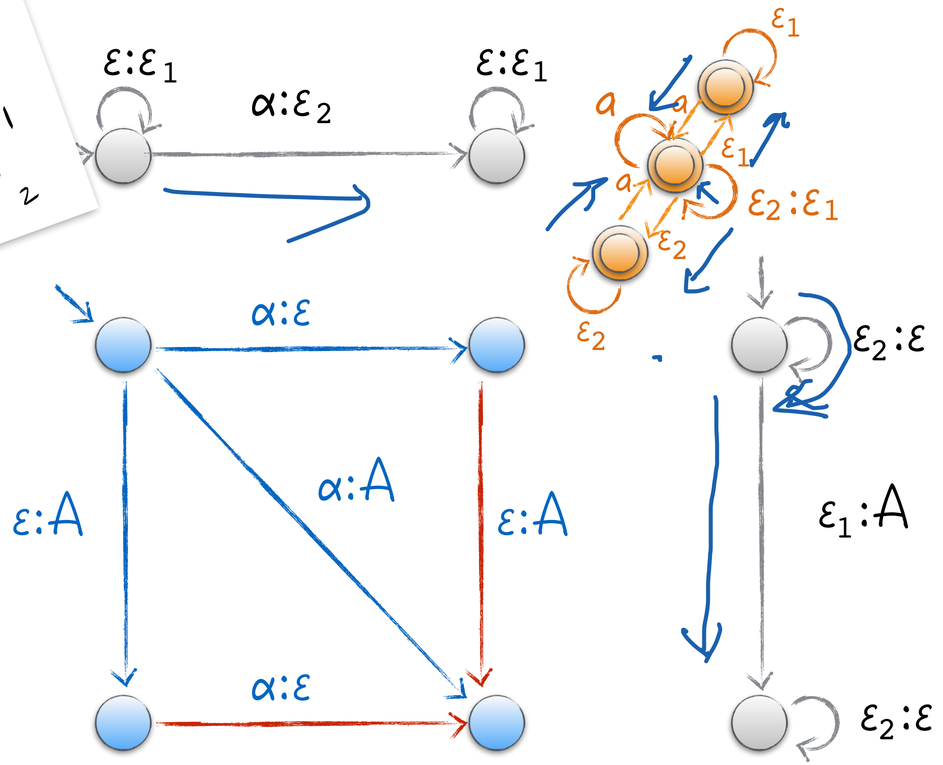
Problematic paths:
$\varepsilon_1\varepsilon_2$ output from M'$_1$
and $\varepsilon_2\varepsilon_1$ input to M'$_2$



Gets rid of the
ε:ε self-loops

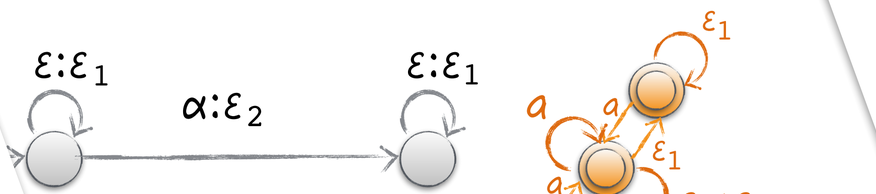
Composition: Filters

Problematic paths:
$\varepsilon_1\varepsilon_2$ output from M'$_1$
and $\varepsilon_2\varepsilon_1$ input to M'$_2$

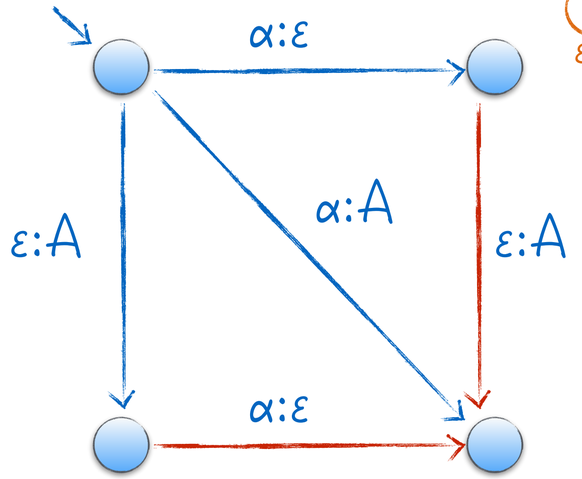


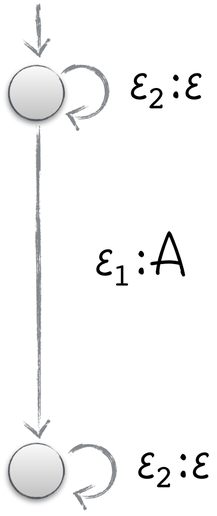
Composition: Filters

Problematic paths:
$\varepsilon_1\varepsilon_2$ output from M'$_1$
and $\varepsilon_2\varepsilon_1$ input to M'$_2$



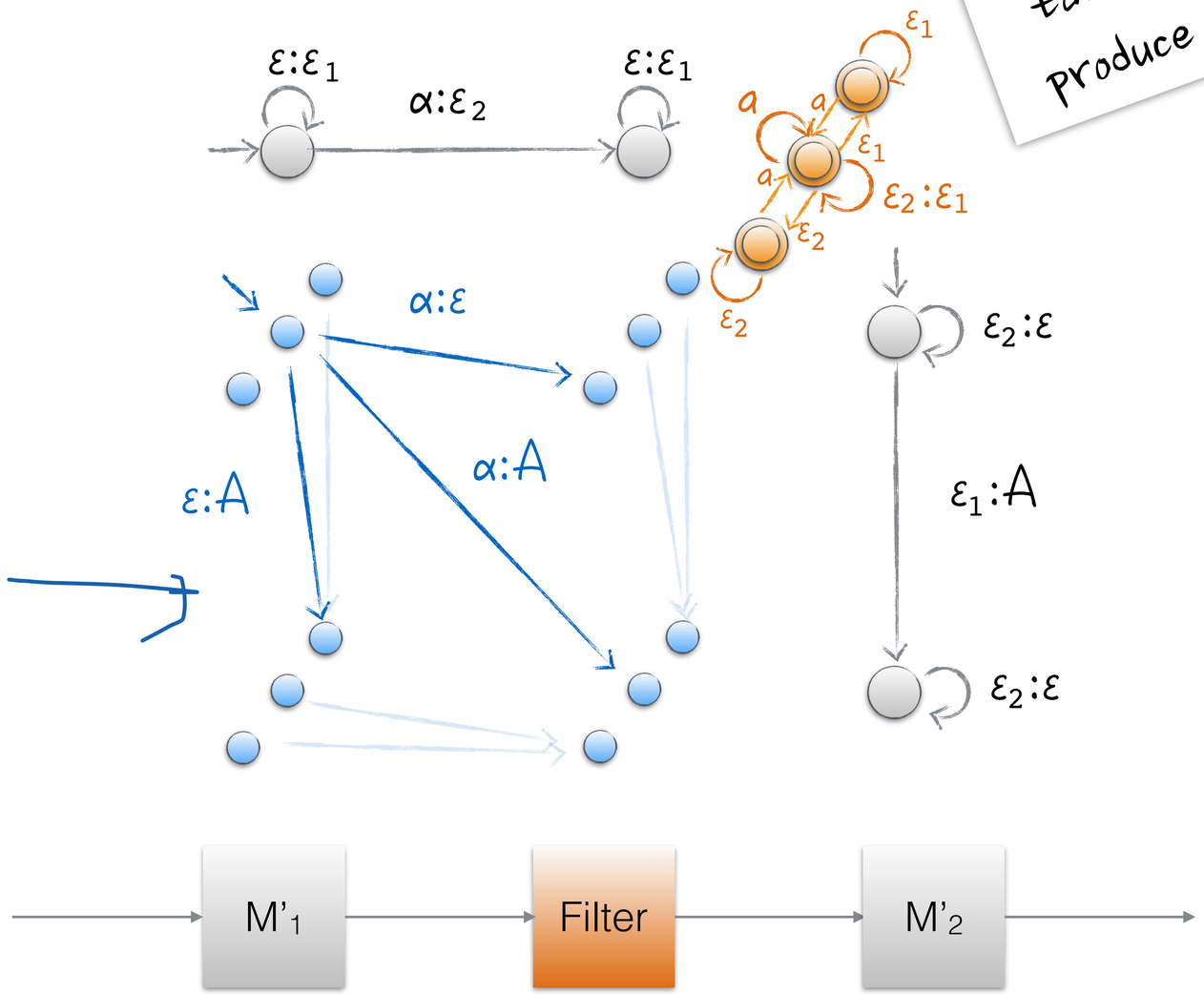
Filter that doesn't take input $\varepsilon_1\varepsilon_2$ or produce output $\varepsilon_2\varepsilon_1$





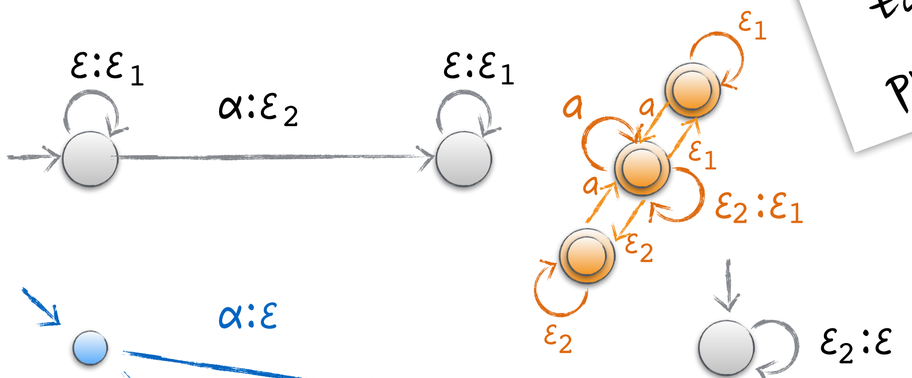
Composition: Filters

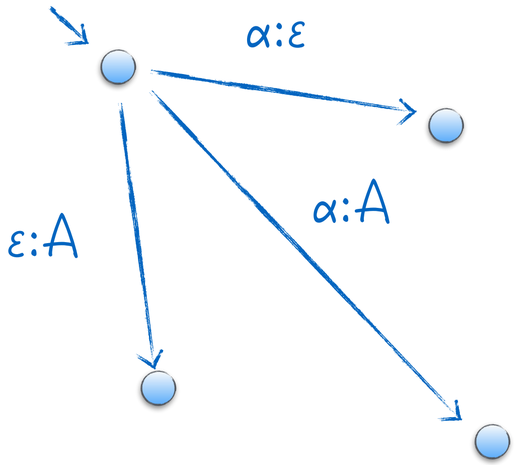
Filter that doesn't take input $\varepsilon_1\varepsilon_2$ or produce output $\varepsilon_2\varepsilon_1$

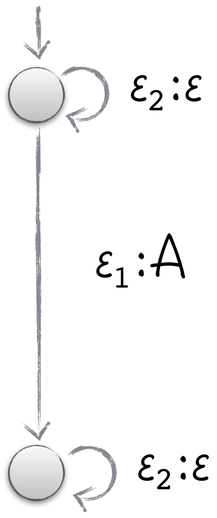


Composition: Filters

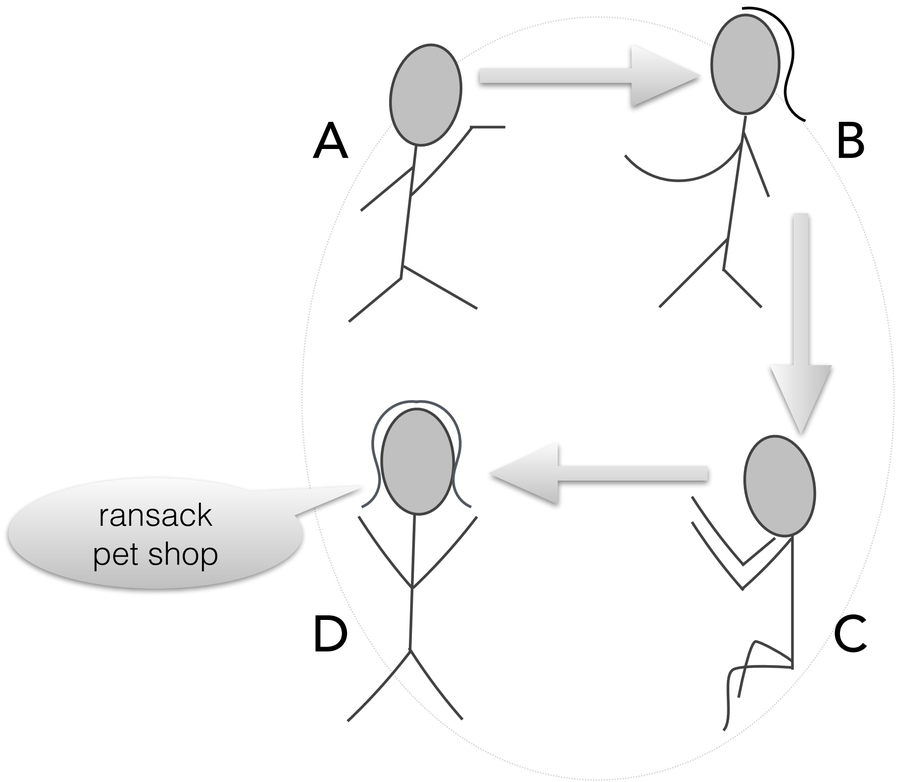
Filter that doesn't take input $\varepsilon_1\varepsilon_2$ or produce output $\varepsilon_2\varepsilon_1$





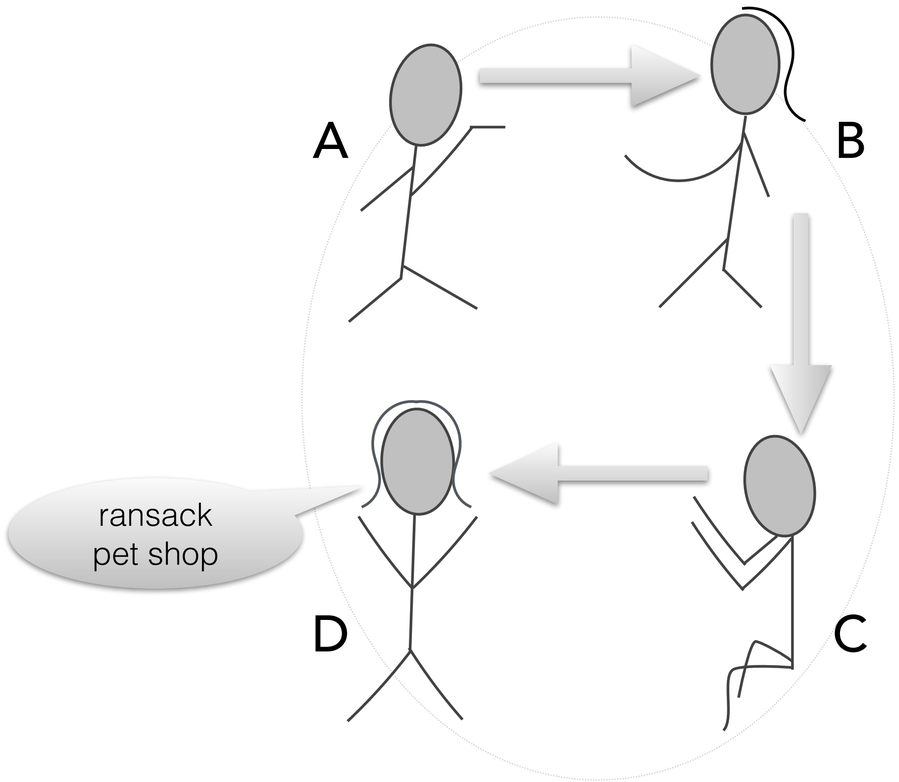


The Telephone Game



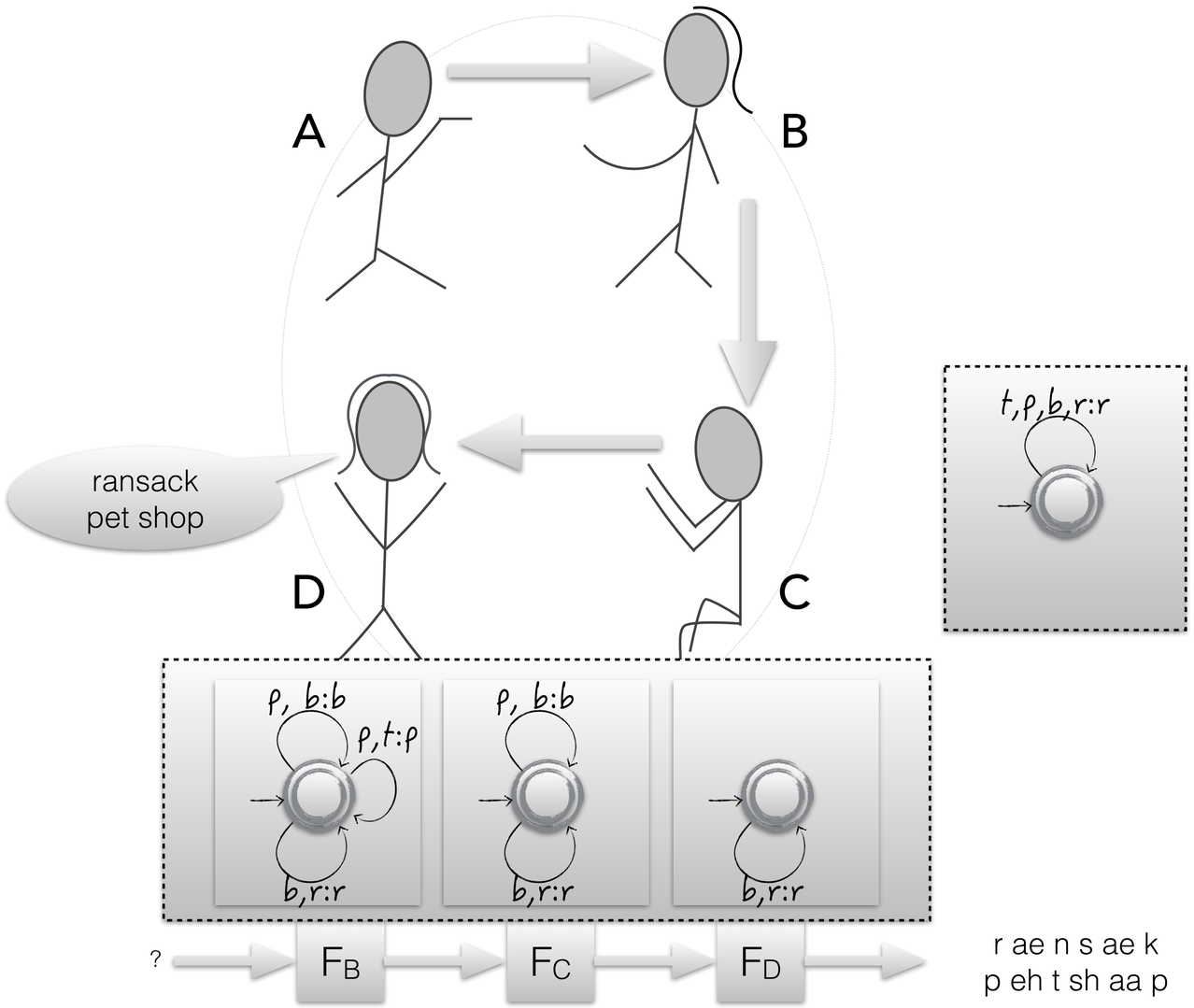
Given what D said,
can we infer the message A started with?

The Telephone Game

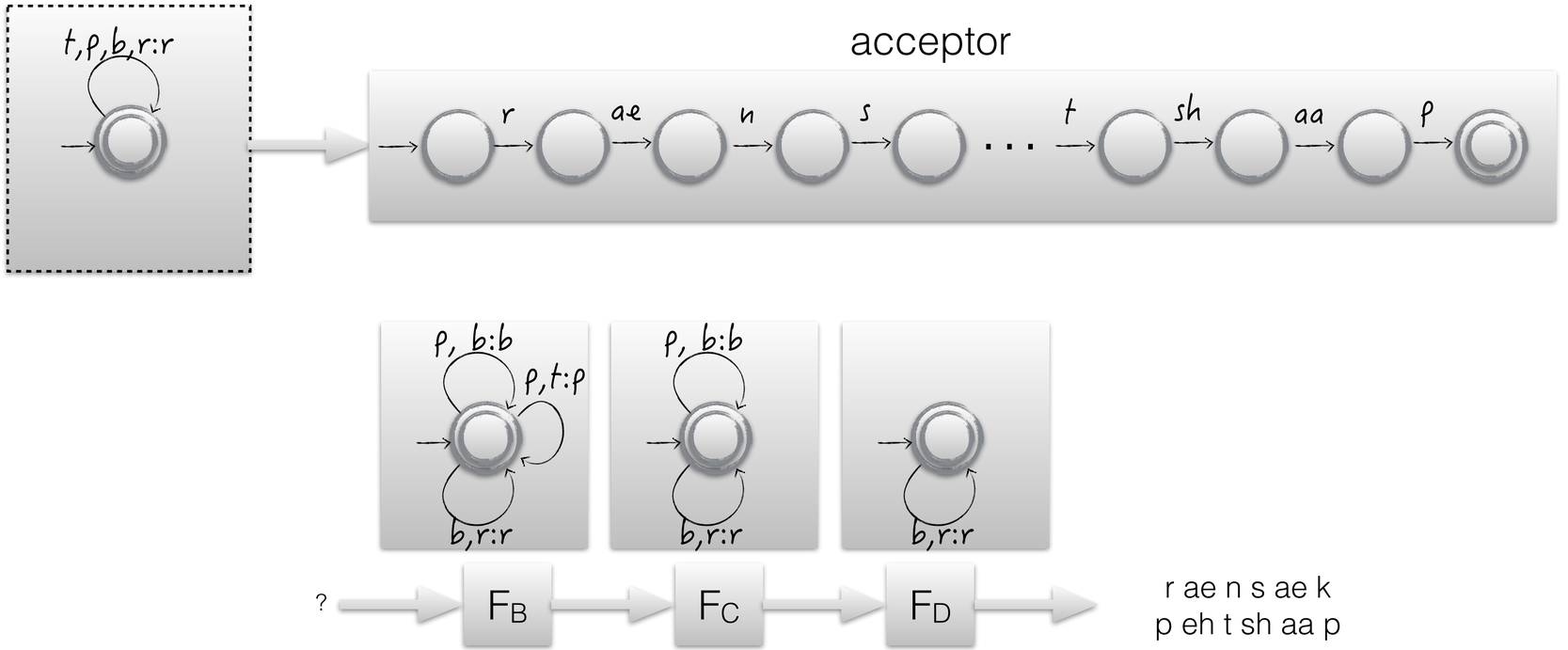


Model the errors made by each player using an FST

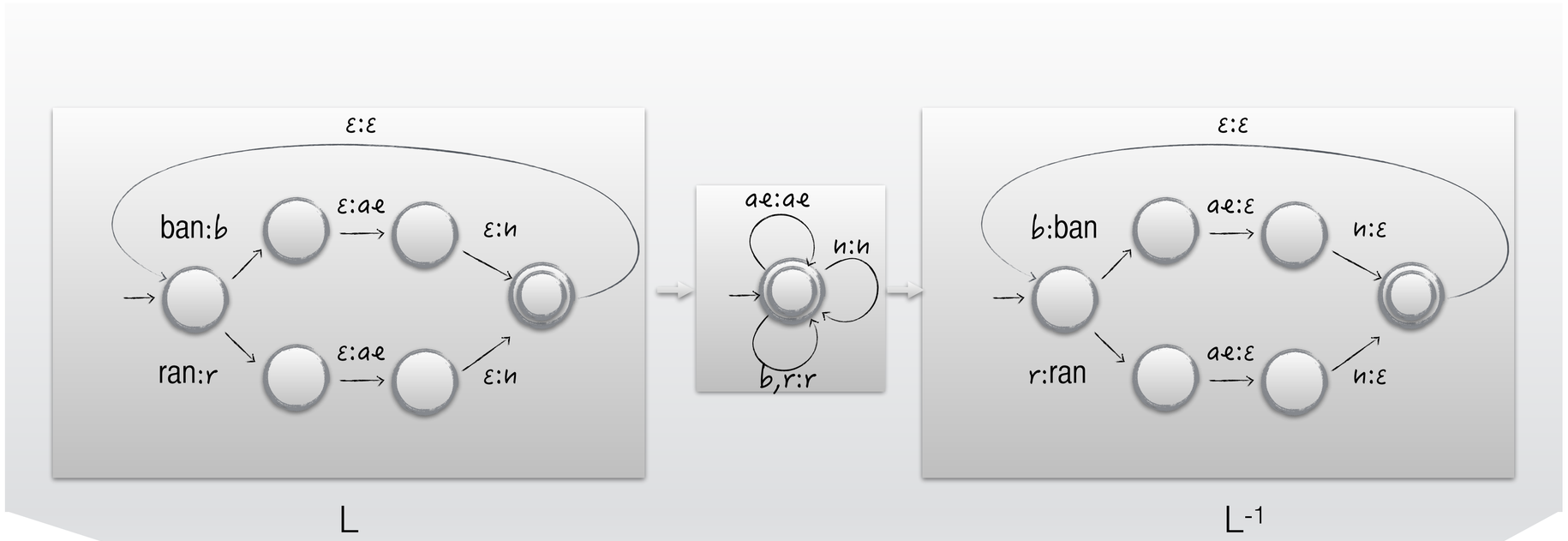
The Telephone Game



The Telephone Game

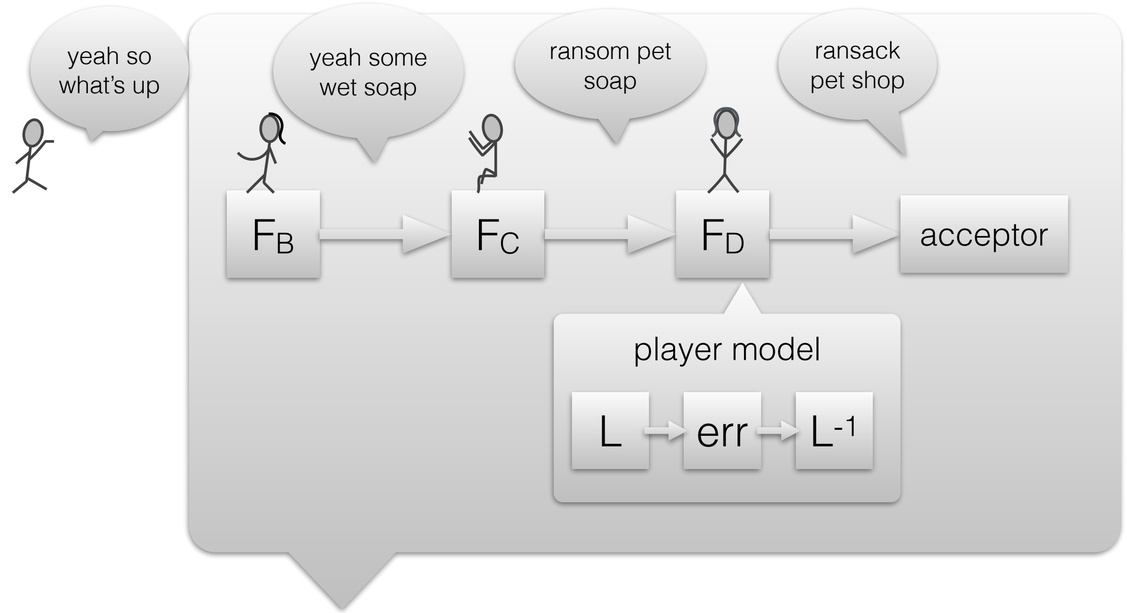


The Telephone Game



? → F_B → F_C → F_D → ransack pet shop

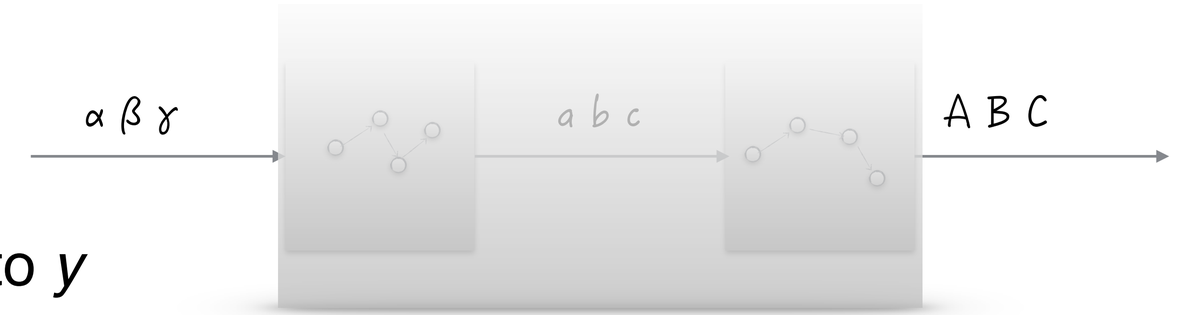
The Telephone Game



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

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*



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

- E.g. If T_1 removes punctuation symbols from a string, and T_2 changes uppercase letters to lowercase letters, then $T_1 \circ T_2$ brings about both changes

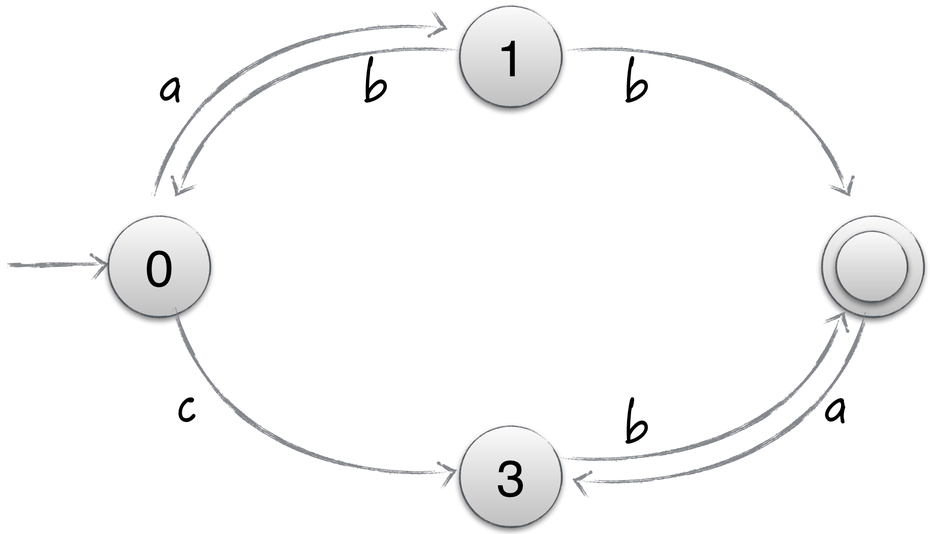
Determinization and Minimization

- WFSTs constructed using various operations (or designed by hand) may have several redundancies
  - Affects the efficiency of subsequent operations
- Determinization and minimization seek to remove redundancies
  - Determinization can expand a WFST, but makes it faster to process an input string
  - Minimization results in the smallest number of states
- Will discuss WFSAs here. Extends to WFSTs.

Deterministic FSAs

- An FSA is **deterministic** if:
  - Unique start state
  - No two transitions from a state share the same label
  - No epsilon labels

Any input sequence yields a unique path (if at all)

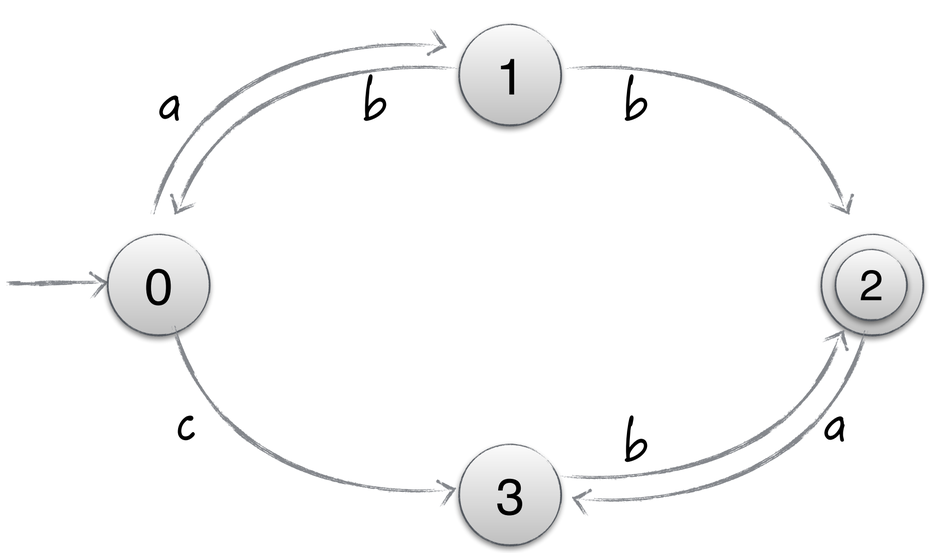


Deterministic or non-deterministic?

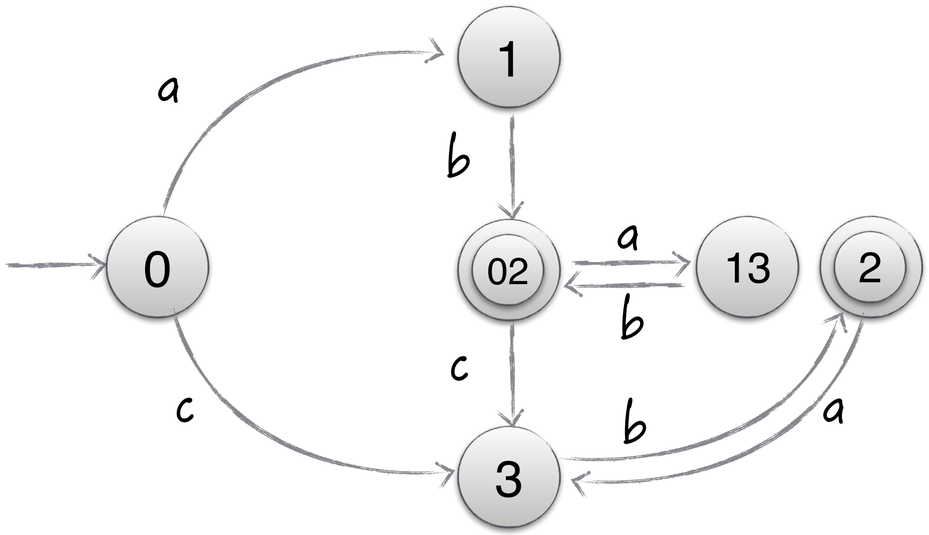
Determinization

States correspond to subsets of states in the original FSA

Construct an equivalent deterministic FSA



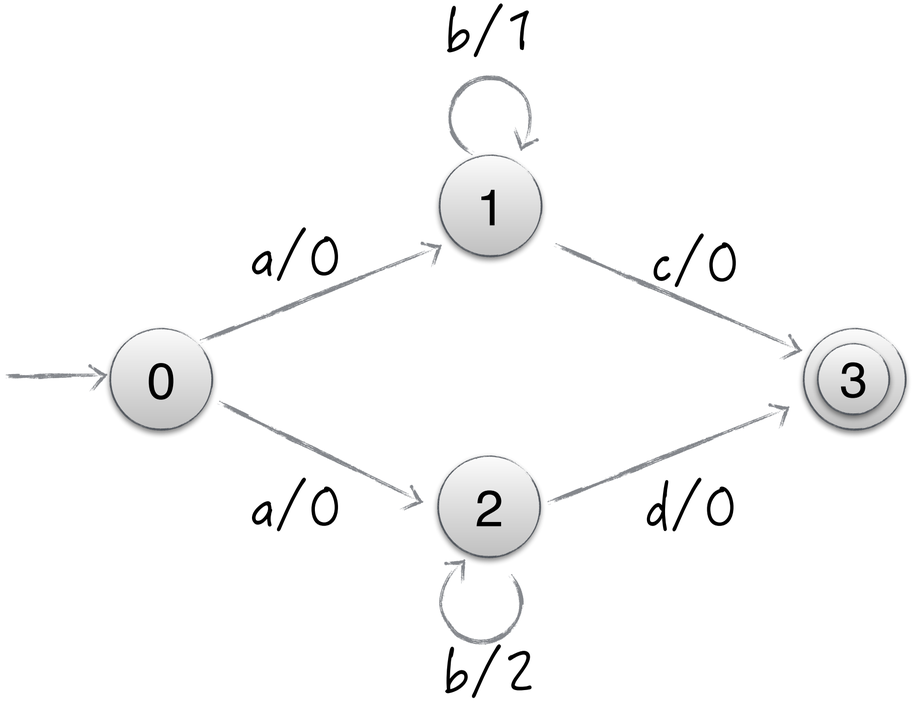
non-deterministic FSA



equivalent deterministic FSA

Determinization: Weighted FSA

Some *Weighted*-FSAs are not determinizable! [M97]



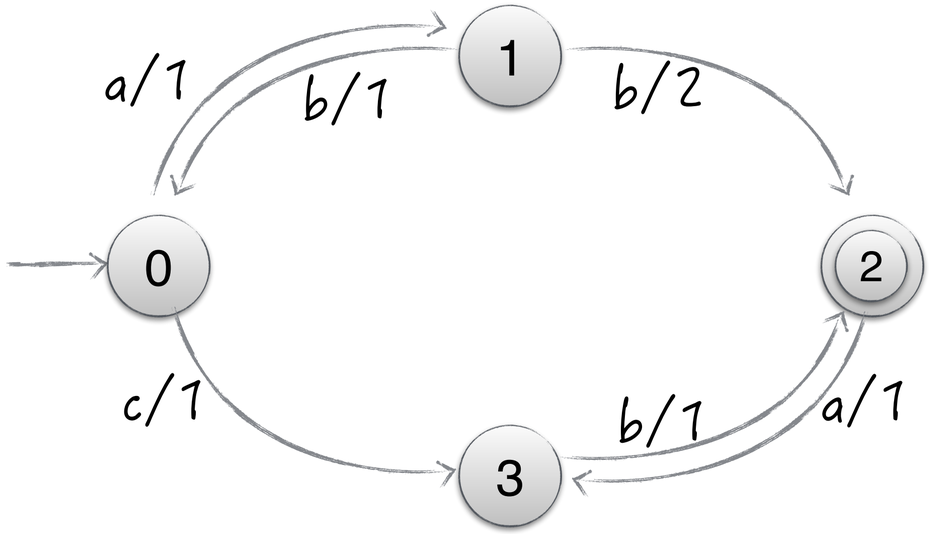
Weight of string $ab^nc = n$ and weight of $ab^nd = 2n$

After seeing ab^n an FSA can't remember n

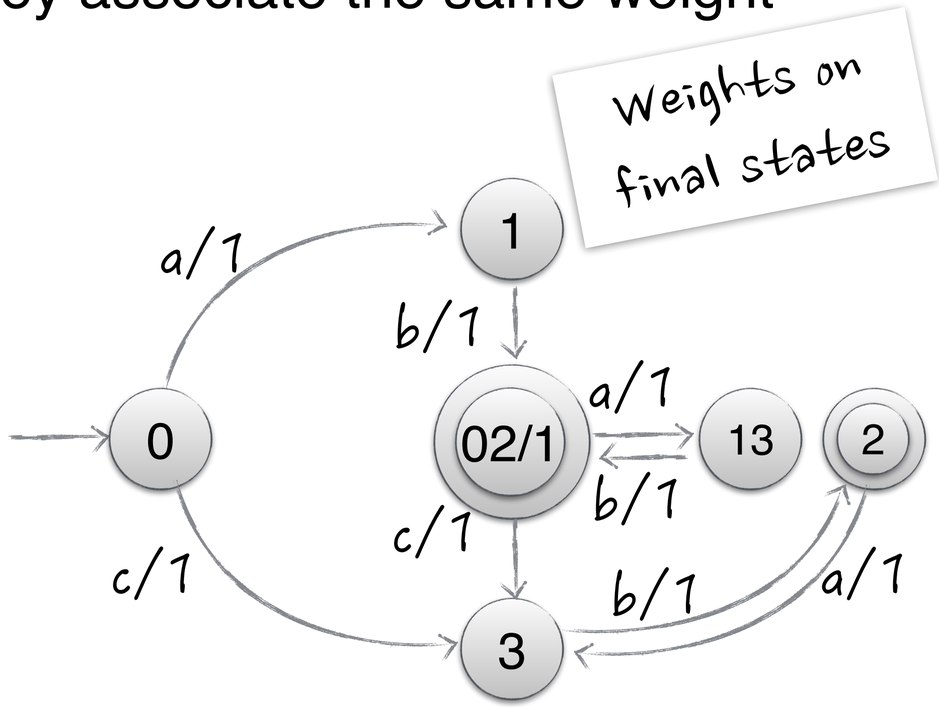
[M97] M. Mohri. Finite-State Transducers in Language and Speech Processing. Computational Linguistics, 23(2), 1997

Determinization: Weighted FSA

Two WFSAs are equivalent if they associate the same weight to each input string



non-deterministic WFSA

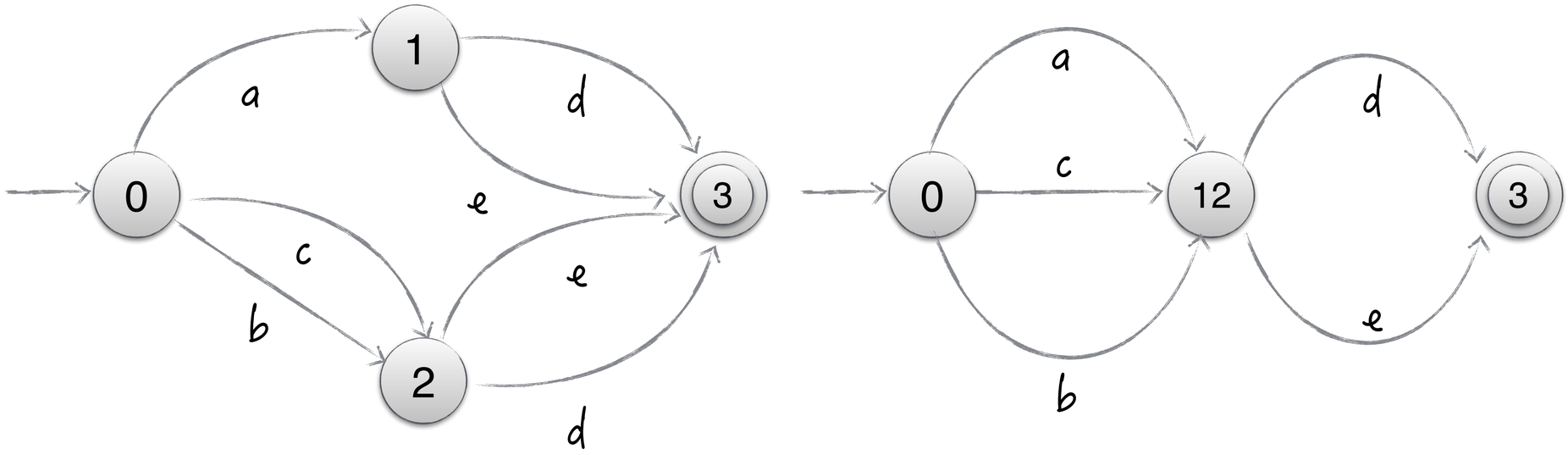


equivalent deterministic WFSA

Minimization

Minimization: find an equivalent deterministic FSA with the least number of states (and transitions)

Unweighted FSAs have a unique minimal FSA [Aho74]

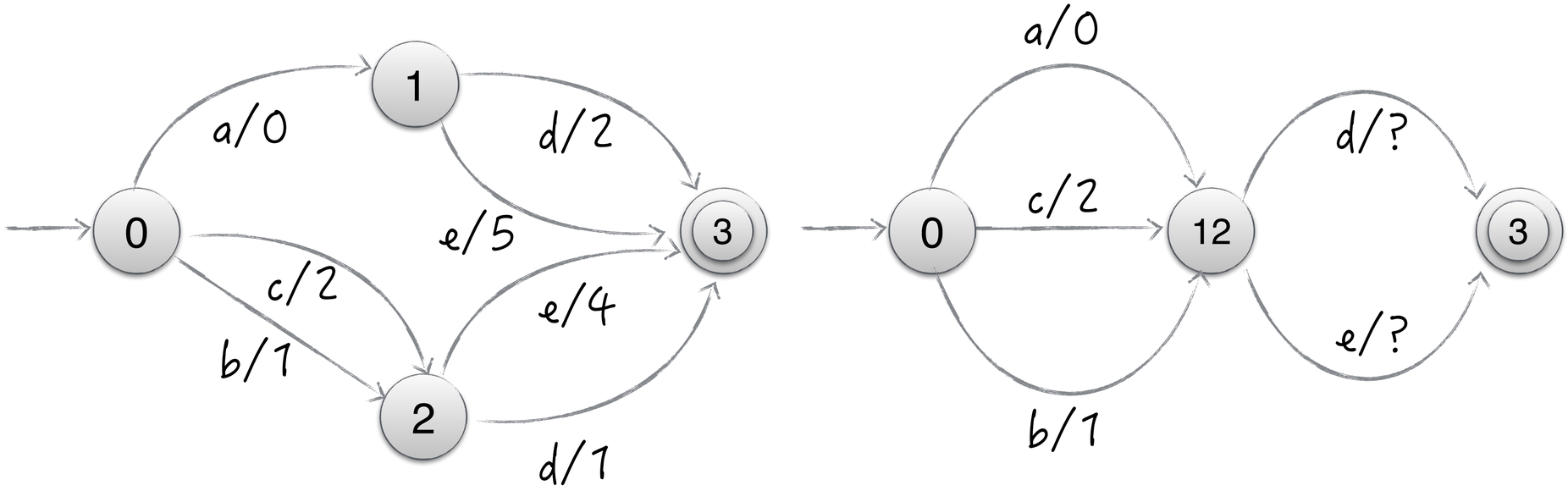


Obtained by identifying and merging *equivalent states*

Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. The design and analysis of computer algorithms. Addison Wesley, 1974.

Minimization: Weighted FSA

Two states are equivalent only if for every input string, the outcome — weight assigned to the string, if accepted — starting from the two states are the same

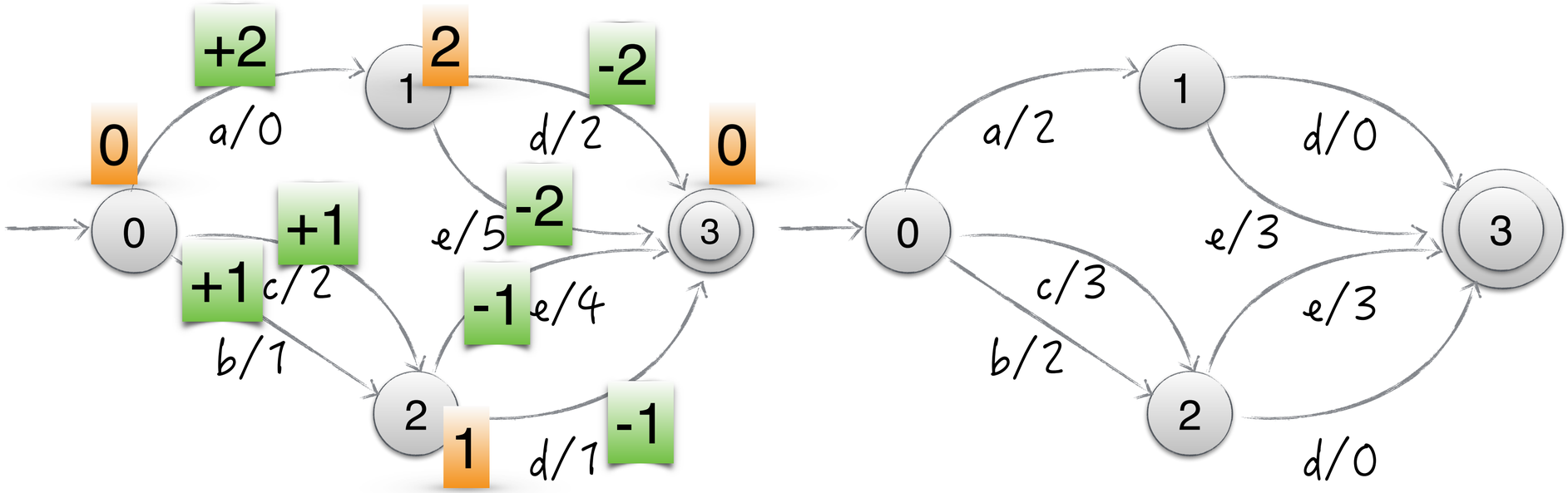


Redistribute weights before identifying equivalent states

Minimization: Weighted FSA

Reweighting OK as long as resulting WFSA is equivalent

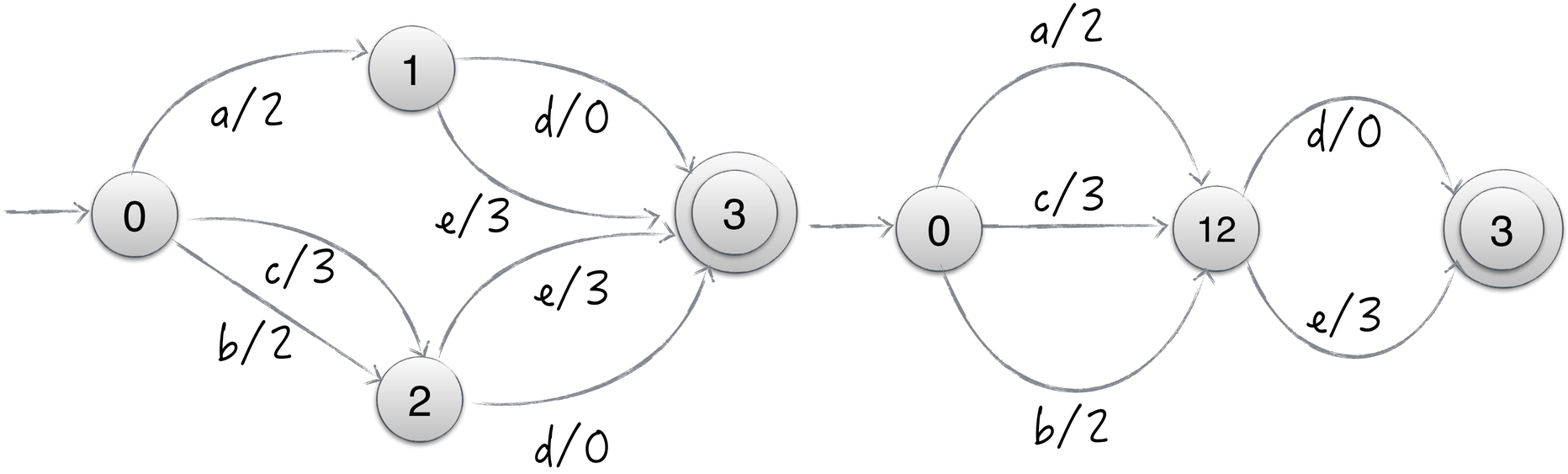
Can reweight using a “potential function” on states



“Weight pushing”: Reweighting using a potential function that optimally moves weights towards the start state

Minimization: Weighted FSA

After weight-pushing, can simply apply unweighted FSA minimization (treating label/weight as label)



Guaranteed to yield a minimal WFSA (under some technical conditions required for weight-pushing)