# Speech Recognition
## Lecture 2: Weighted Finite-State Transducer Software Library

Mehryar Mohri
Courant Institute of Mathematical Sciences
mohri@cims.nyu.edu

# Software Libraries

- FSM Library: Finite-State Machine Library. General software utilities for building, combining, optimizing, and searching weighted automata and transducers (MM, Pereira, and Riley, 2000).

  http://www.research.att.com/projects/mohri/fsm

- OpenFst Library: Open-source Finite-state transducer Library (Allauzen et al., 2007).

  http://www.openfst.org

# Software Libraries

- GRM Library: Grammar Library. General software collection for constructing and modifying weighted automata and transducers representing grammars and statistical language models (Allauzen, MM, and Roark, 2005).

  http://www.research.att.com/projects/mohri/grm

- DCD Library: Decoder Library. General software collection for speech recognition decoding and related functions (MM and Riley, 2003).

  http://www.research.att.com/~fsmtools/dcd

# FSM Library

- The FSM utilities construct, combine, minimize, and search *weighted finite-states transducers*.
  - User Program Level: Programs that read from and write to files or pipelines, *fsm(1)*:

```
fsmintersect in1.fsm in2.fsm >out.fsm
```

  - C(++) Library Level: Library archive of C(++) functions that implements the user program level, *fsm(3)*:

```
Fsm in1 = FSMLoad("in1.fsm");
Fsm in2 = FSMLoad("in2.fsm");
Fsm out = FSMIntersect(fsm1, fsm2);
FSMDump("out.fsm", out);
```

- Definition Level: Specification of *labels*, of *costs*, and of types of FSM representations.

# This Lecture

- Weighted automata and transducers
- Rational operations
- Elementary unary operations
- Fundamental binary operations
- Optimization algorithms
- Search algorithms

# FSM File Types

- Textual format
  - automata/acceptor files,
  - transducer files,
  - symbols files.
- Binary format: *compiled* representation used by all FSM utilities.

# Compiling, Printing, and Drawing

- Compiling
  - fsmcompile -s tropical -iA.syms <A.txt >A.fsm
  - fsmcompile -s log -iA.syms -oA.syms -t <T.txt >T.fsm
- Printing
  - fsmprint -iA.syms <A.fsm >A.txt
  - fsmprint -iA.syms -oA.syms <T.fsm | dot -Tps >T.ps
- Drawing
  - fsmdraw -iA.syms <A.fsm | dot -Tps >A.ps
  - fsmdraw -iA.syms -oA.syms <T.fsm | dot -Tps >T.ps

# Weight Sets: Semirings

- A semiring $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ is a ring that may lack negation.
  - sum: to compute the weight of a sequence (sum of the weights of the paths labeled with that sequence).
  - product: to compute the weight of a path (product of the weights of constituent transitions).

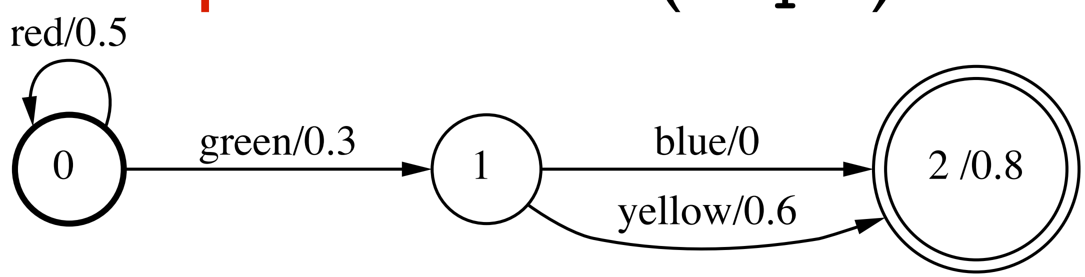
# Semirings - Examples

| Semiring | Set | $\oplus$ | $\otimes$ | $\overline{0}$ | $\overline{1}$ |
|---|---|---|---|---|---|
| Boolean | $\{0, 1\}$ | $\vee$ | $\wedge$ | $0$ | $1$ |
| Probability | $\mathbb{R}_+$ | $+$ | $\times$ | $0$ | $1$ |
| Log | $\mathbb{R} \cup \{-\infty, +\infty\}$ | $\oplus_{\log}$ | $+$ | $+\infty$ | $0$ |
| Tropical | $\mathbb{R} \cup \{-\infty, +\infty\}$ | $\min$ | $+$ | $+\infty$ | $0$ |

with $\oplus_{\log}$ defined by: $x \oplus_{\log} y = -\log(e^{-x} + e^{-y})$.

# Automata/Acceptors

- Graphical Representation (`A.ps`)




- Acceptor file (`A.txt`)

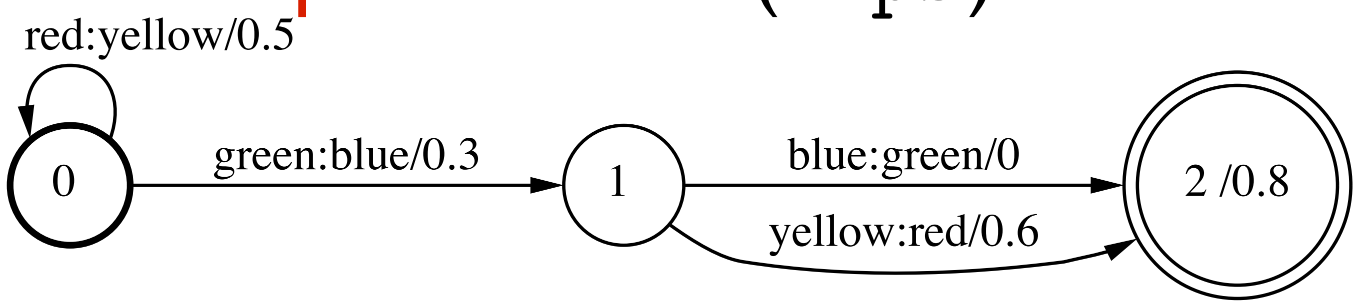
```
0   0   red      .5
0   1   green    .3
1   2   blue
1   2   yellow   .6
2   .8
```

- Symbols file (`A.syms`)

```
red      1
green    2
blue     3
yellow   4
```

# Transducers

- Graphical Representation (`T.ps`)




- Transducer file (`T.txt`)

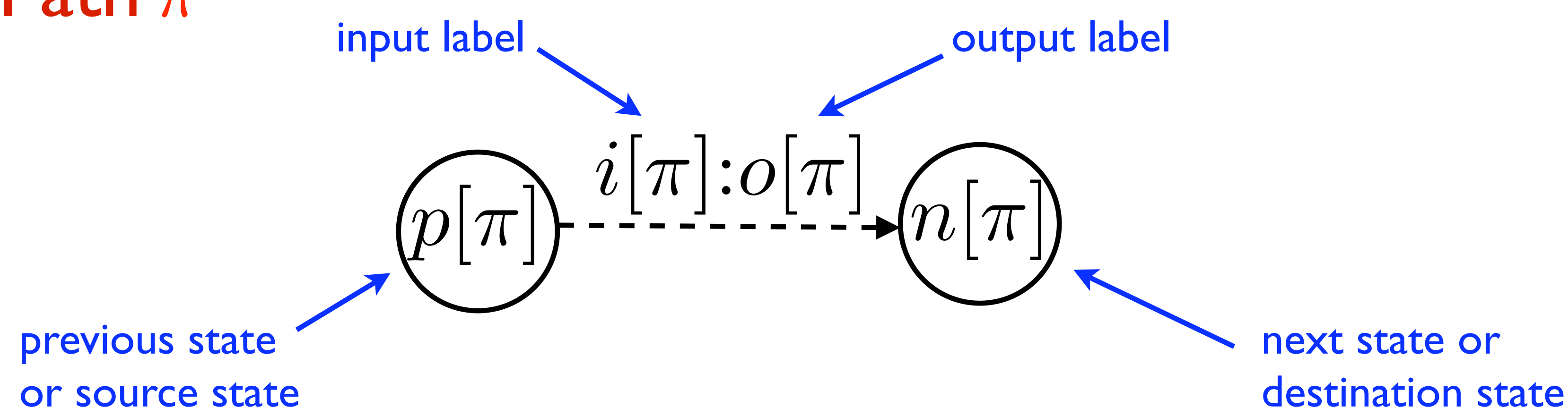
```
0   0   red      yellow   .5
0   1   green    blue     .3
1   2   blue     green
1   2   yellow   red      .6
2   .8
```

- Symbols file (`T.syms`)

```
red      1
green    2
blue     3
yellow   4
```

# Paths - Definitions and Notation

- Path $\pi$




- Sets of paths
  - $P(R_1, R_2)$ : paths from $R_1 \subseteq Q$ to $R_2 \subseteq Q$ .
  - $P(R_1, x, R_2)$ : paths in $P(R_1, R_2)$ with input label *x*.
  - $P(R_1, x, y, R_2)$ : paths in $P(R_1, x, R_2)$ with output label *y*.

# General Definitions

- **Alphabets**: input $\Sigma$, output $\Delta$.
- **States**: $Q$, initial states $I$, final states $F$.
- **Transitions**: $E \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times (\Delta \cup \{\epsilon\}) \times \mathbb{K} \times Q$.
- **Weight functions**:
  - initial: $\lambda : I \to \mathbb{K}$ .
  - final: $\rho : F \to \mathbb{K}$ .

# Automata and Transducers - Definitions

- **Automaton** $A = (\Sigma, Q, I, F, E, \lambda, \rho)$

$\forall x \in \Sigma^*,$

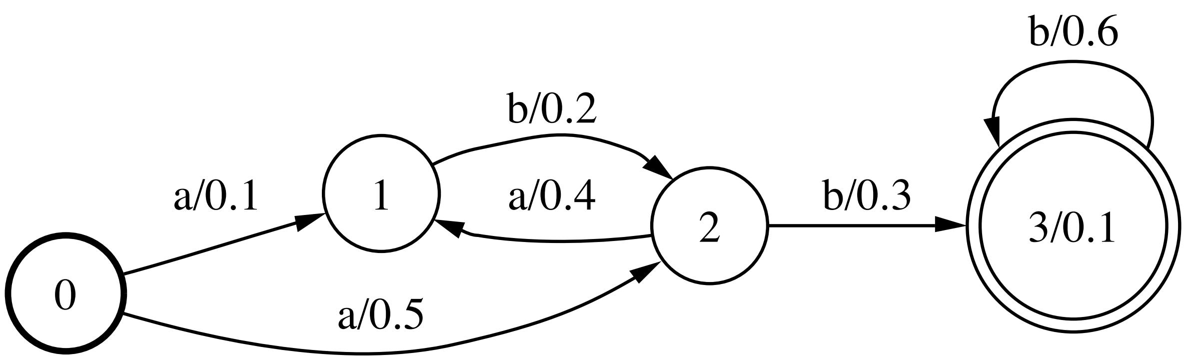
$$[\![A]\!](x) = \bigoplus_{\pi \in P(I,x,F)} \lambda(p[\pi]) \otimes w[\pi] \otimes \rho(n[\pi])$$

- **Transducer** $T = (\Sigma, \Delta, Q, I, F, E, \lambda, \rho)$

$\forall x \in \Sigma^*, y \in \Delta^*,$

$$[\![T]\!](x, y) = \bigoplus_{\pi \in P(I,x,y,F)} \lambda(p[\pi]) \otimes w[\pi] \otimes \rho(n[\pi])$$

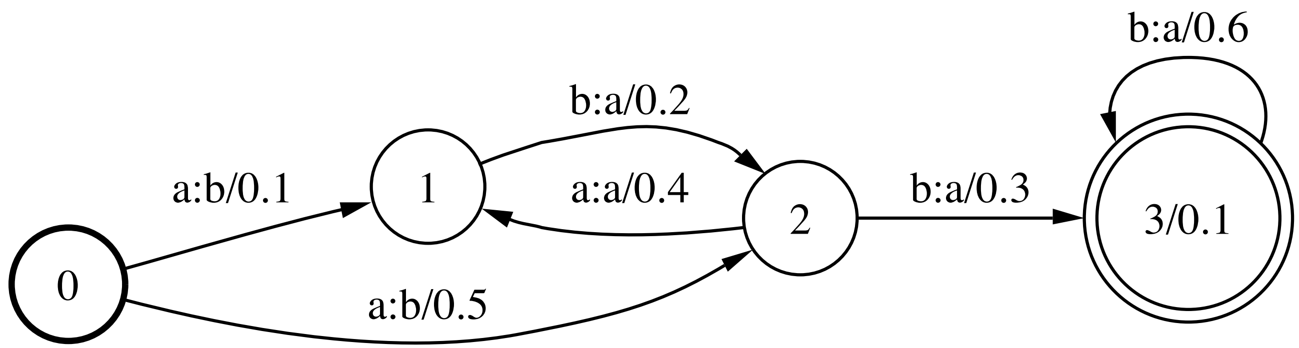
# Weighted Automata




$[[A]](x) =$ Sum of the weights of all successful paths labeled with *x*

$$[[A]](abb) = .1 \times .2 \times .3 \times .1 + .5 \times .3 \times .6 \times .1$$

# Weighted Transducers




$[[T]](x, y) =$ Sum of the weights of all successful paths with input *x* and output *y*.

$$[[T]](abb, baa) = .1 \times .2 \times .3 \times .1 + .5 \times .3 \times .6 \times .1$$

# This Lecture

- Weighted automata and transducers
- Rational operations
- Elementary unary operations
- Fundamental binary operations
- Optimization algorithms
- Search algorithms

# Rational Operations

- Sum

$$[\![T_1 \oplus T_2]\!](x,y) = [\![T_1]\!](x,y) \oplus [\![T_2]\!](x,y)$$

- Product

$$[\![T_1 \otimes T_2]\!](x,y) = \bigoplus_{\substack{x=x_1x_2 \\ y=y_1y_2}} [\![T_1]\!](x_1,y_1) \otimes [\![T_2]\!](x_2,y_2).$$

- Closure

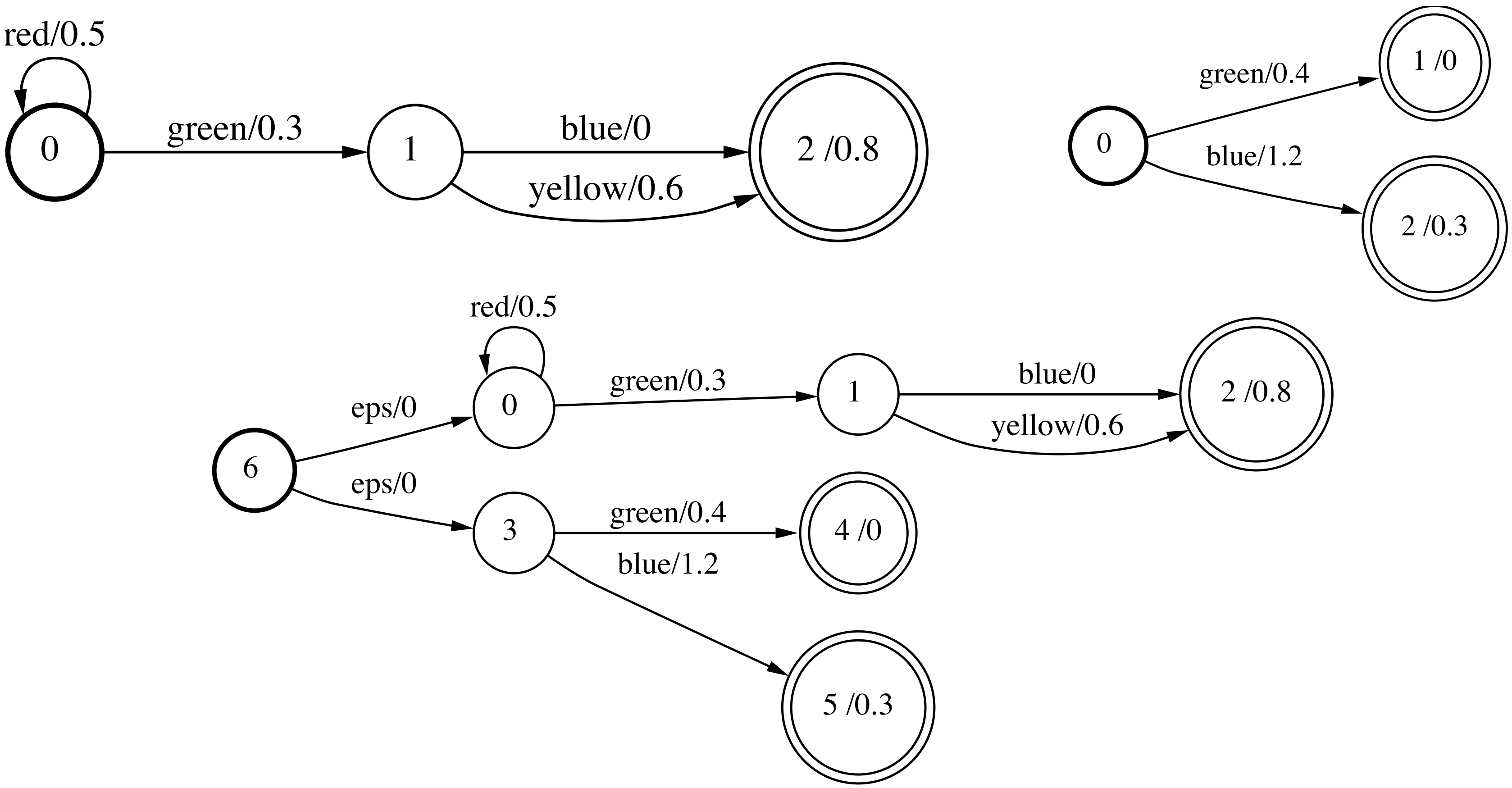
$$[\![T^*]\!](x,y) = \bigoplus_{n=0}^{\infty} [\![T]\!]^n(x,y)$$

- Conditions (on the closure operation): condition on *T*: e.g., weight of ϵ-cycles = $\overline{0}$ (*regulated transducers*), or semiring condition: e.g., $\overline{1} \oplus x = \overline{1}$ as with the tropical semiring (more generally *locally closed semirings*).

- Complexity and implementation:
  - linear-time complexity:
    $O((|E_1| + |Q_1|) + (|E_2| + |Q_2|))$ or
    $O(|Q| + |E|)$
  - lazy implementation.

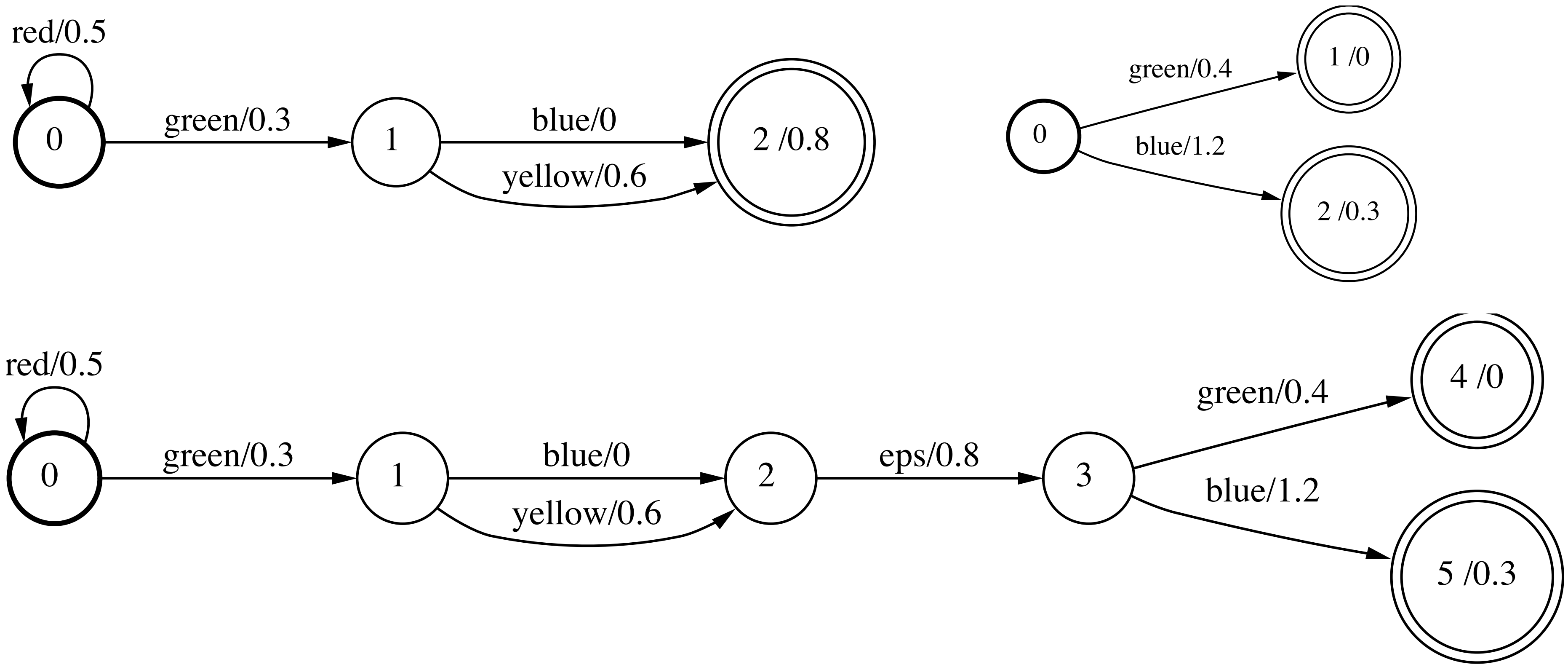
# Sum - Illustration

- **Program**: fsmunion A.fsm B.fsm >C.fsm
- **Graphical representation**:



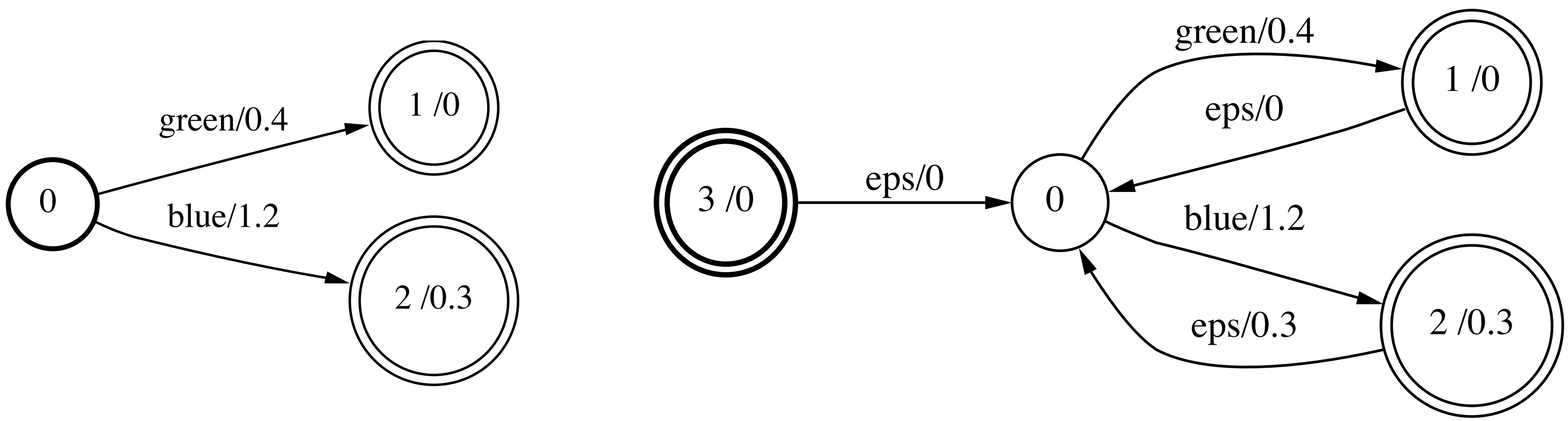
# Product - Illustration

- Program: fsmconcat A.fsm B.fsm >C.fsm
- Graphical representation:



# Closure - Illustration

- Program: fsmclosure B.fsm >C.fsm
- Graphical representation:



# This Lecture

- Weighted automata and transducers
- Rational operations
- Elementary unary operations
- Fundamental binary operations
- Optimization algorithms
- Search algorithms

# Elementary Unary Operations

- Reversal

$$[\![\widetilde{T}]\!](x, y) = [\![T]\!](\widetilde{x}, \widetilde{y})$$

- Inversion

$$[\![T^{-1}]\!](x, y) = [\![T]\!](y, x)$$

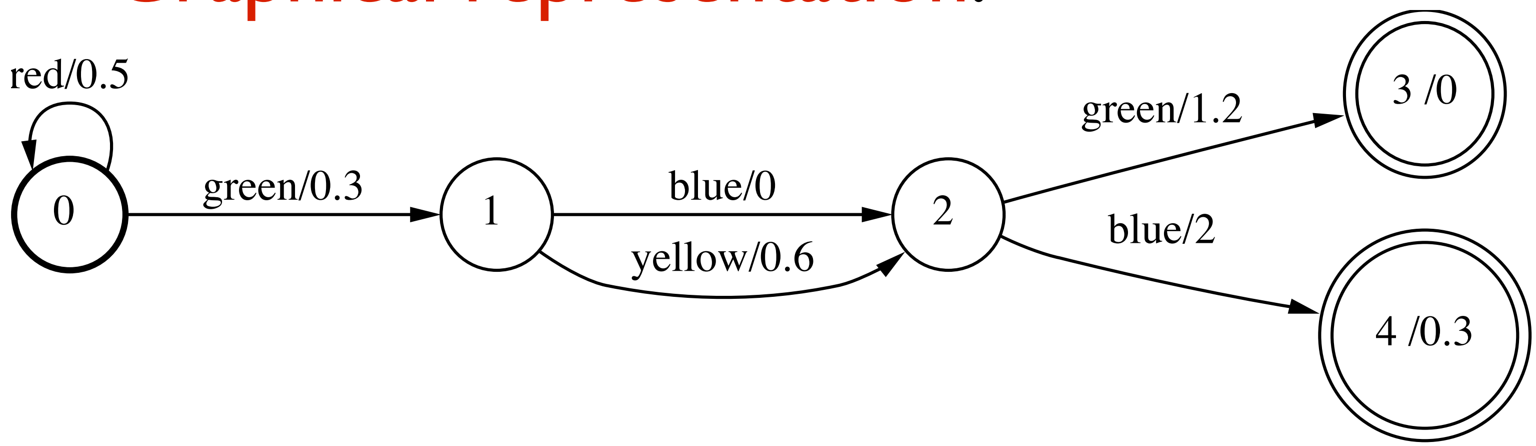
- Projection

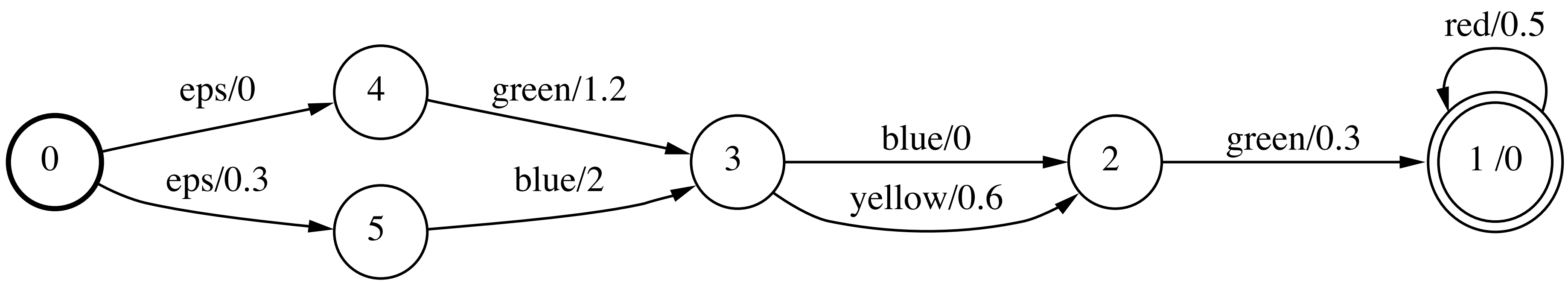
$$[\![A]\!](x) = \bigoplus_{y} [\![T]\!](x, y)$$

- Linear-time complexity, lazy implementation (not for reversal).

# Reversal - Illustration

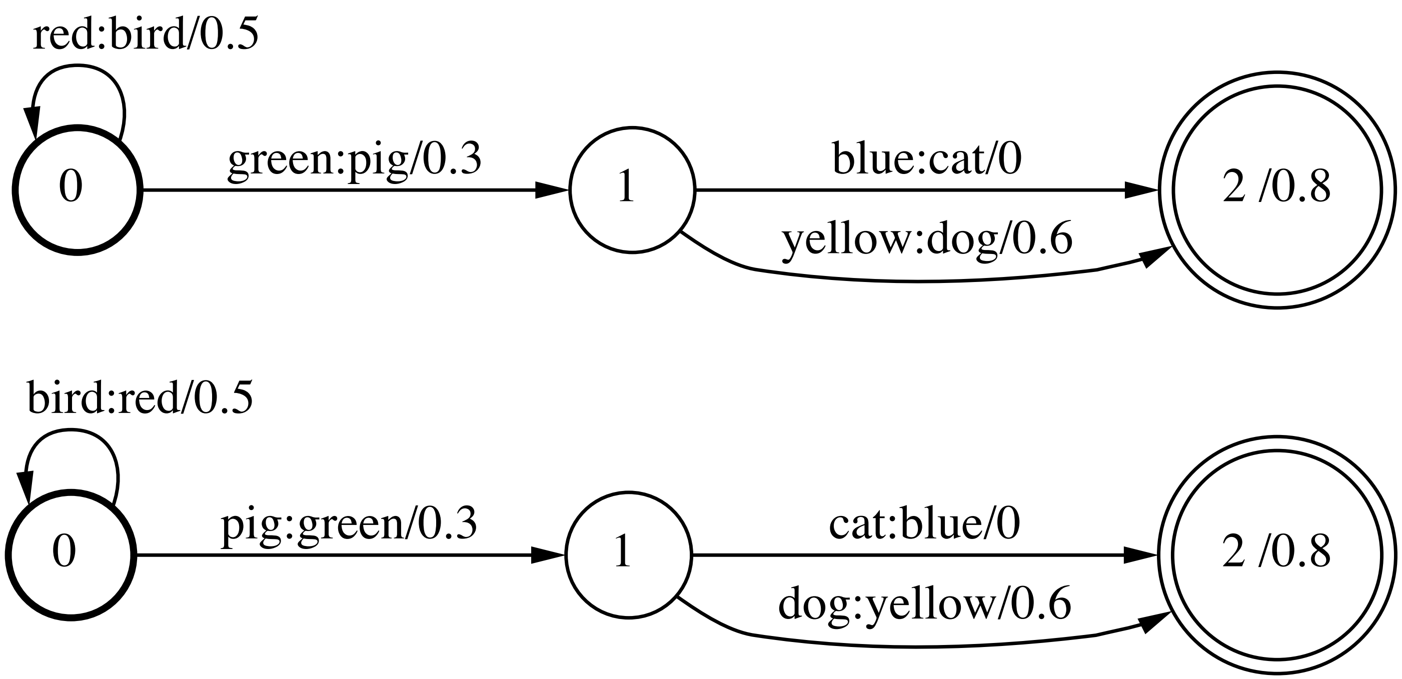
- Program: fsmreverse A.fsm >C.fsm
- Graphical representation:



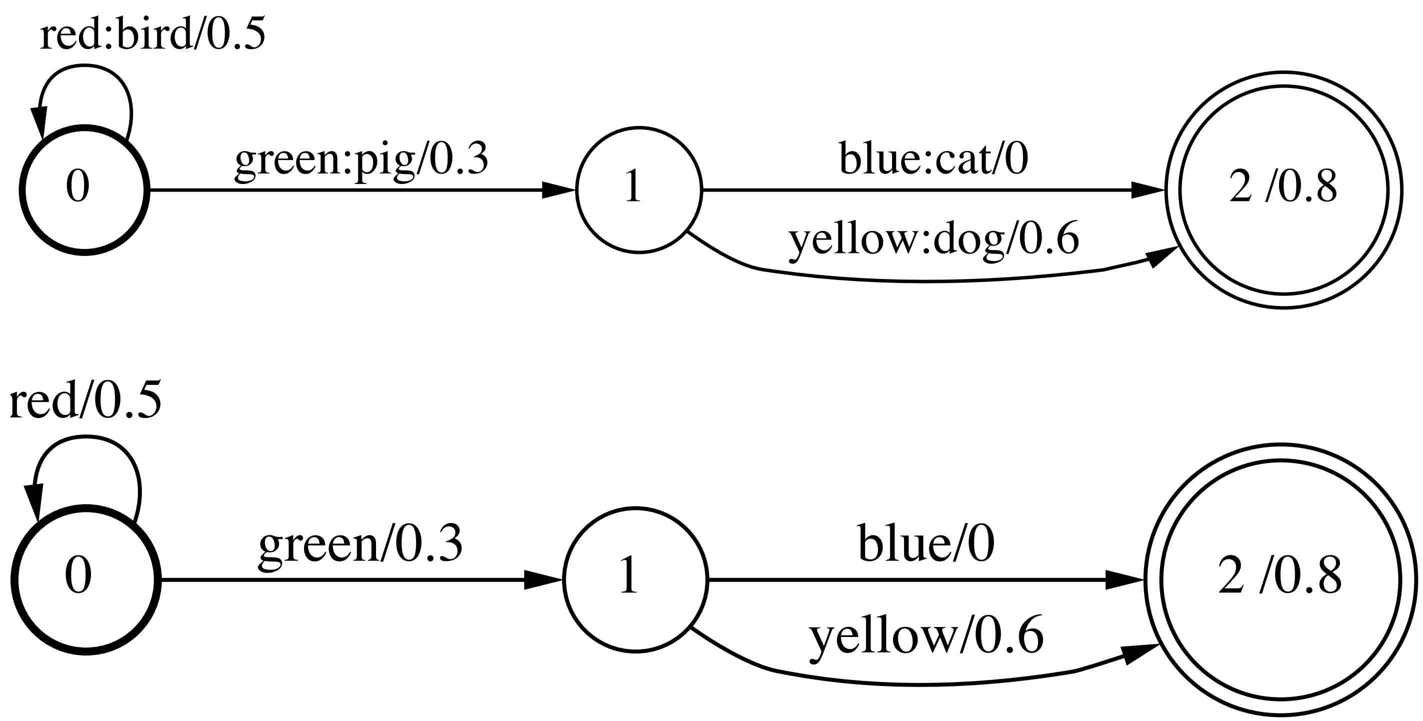
# Inversion - Illustration

- Program: fsminvert A.fsm >C.fsm
- Graphical representation:



# Projection - Illustration

- Program: fsmproject -1 T.fsm >A.fsm
- Graphical representation:



# This Lecture

- Weighted automata and transducers
- Rational operations
- Elementary unary operations
- Fundamental binary operations
- Optimization algorithms
- Search algorithms

# Some Fundamental Binary Operations

(Pereira and Riley, 1997; MM et al. 1996)

- Composition ($(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ commutative)

$$[\![T_1 \circ T_2]\!](x, y) = \bigoplus_z [\![T_1]\!](x, z) \otimes [\![T_2]\!](z, y)$$

- Intersection ($(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ commutative)

$$[\![A_1 \cap A_2]\!](x) = [\![A_1]\!](x) \otimes [\![A_2]\!](x)$$

- Difference ($A_2$ unweighted and deterministic)

$$[\![A_1 - A_2]\!](x) = [\![A_1 \cap \overline{A_2}]\!](x)$$

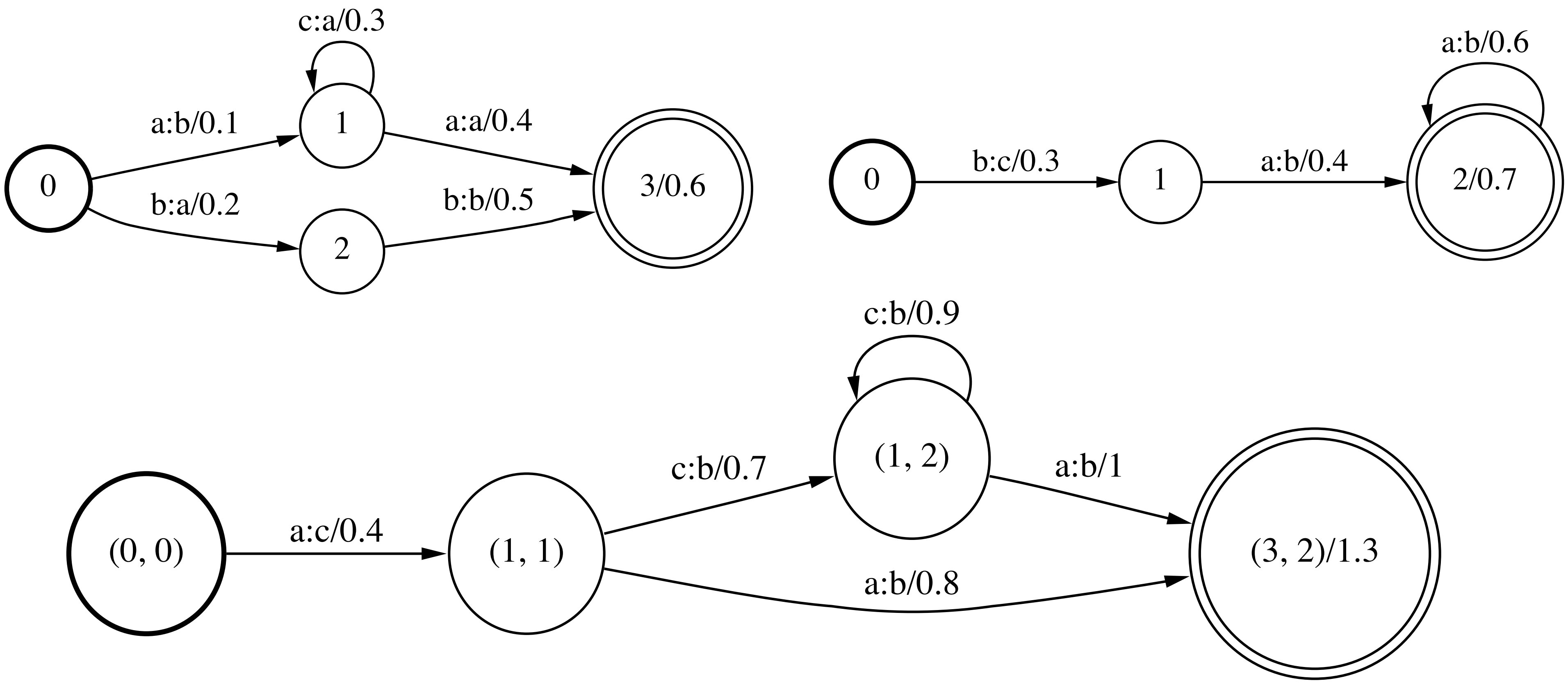
- Complexity and implementation:
  - quadratic complexity:

$$O((|E_1| + |Q_1|)\,(|E_2| + |Q_2|))$$

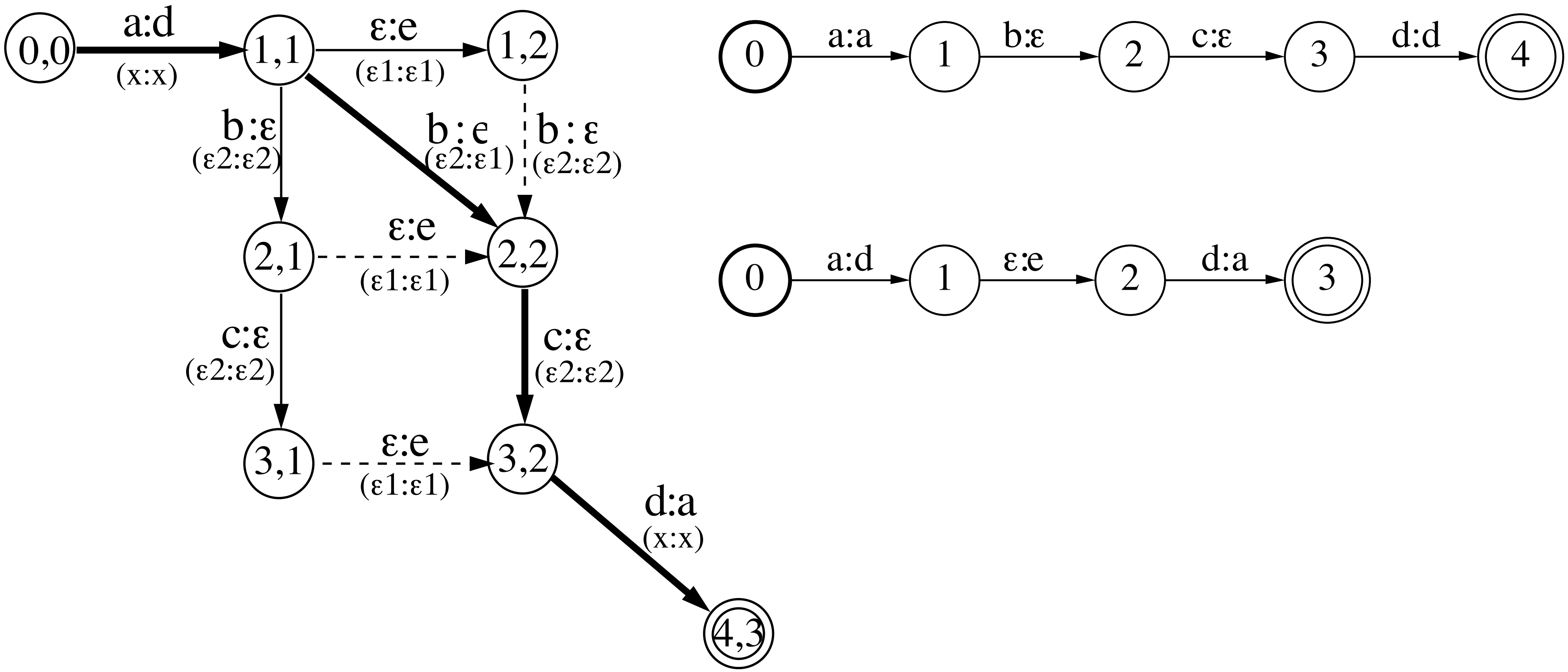
  - path multiplicity in presence of ϵ-transitions: ϵ-filter;
  - lazy implementation.

# Composition - Illustration

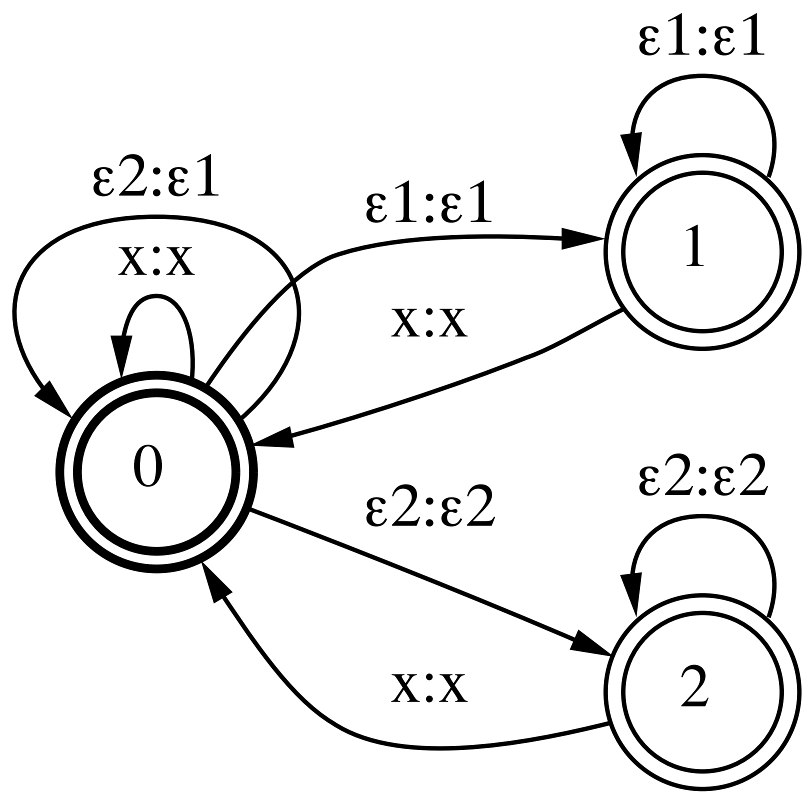
- **Program**: fsmcompose A.fsm B.fsm >C.fsm
- **Graphical representation**:



# Multiplicity and ϵ-Transitions - Problem



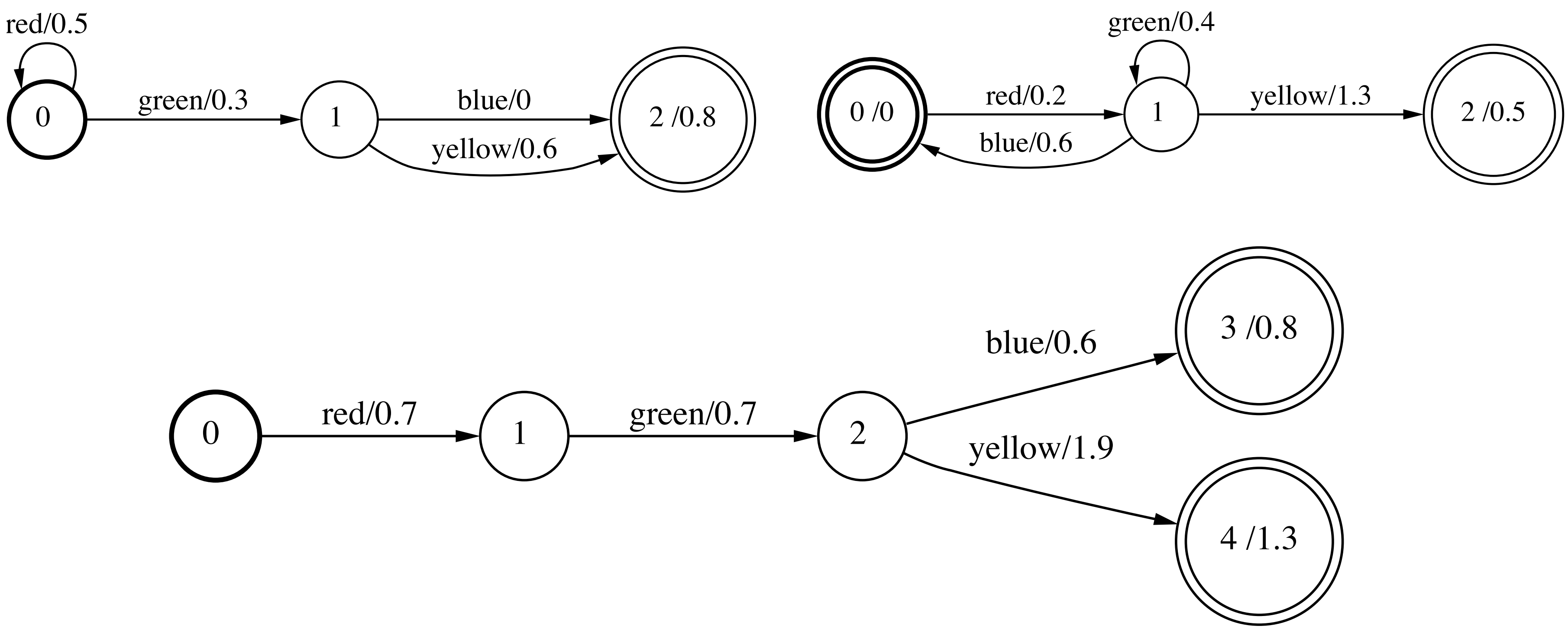
# Solution - Filter $F$ for Composition




$$\text{Replace } T_1 \circ T_2 \text{ with } \tilde{T}_1 \circ F \circ \tilde{T}_2.$$

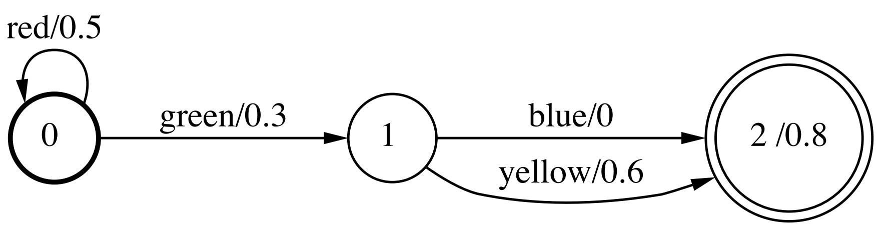
# Intersection - Illustration

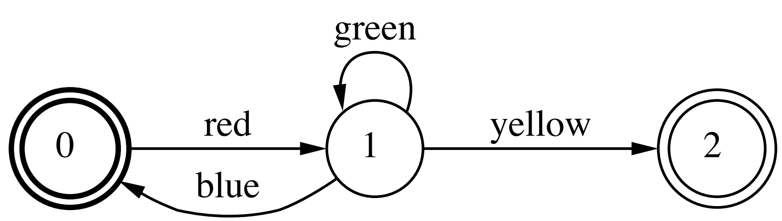
- **Program**: fsmintersect A.fsm B.fsm >C.fsm
- **Graphical representation**:

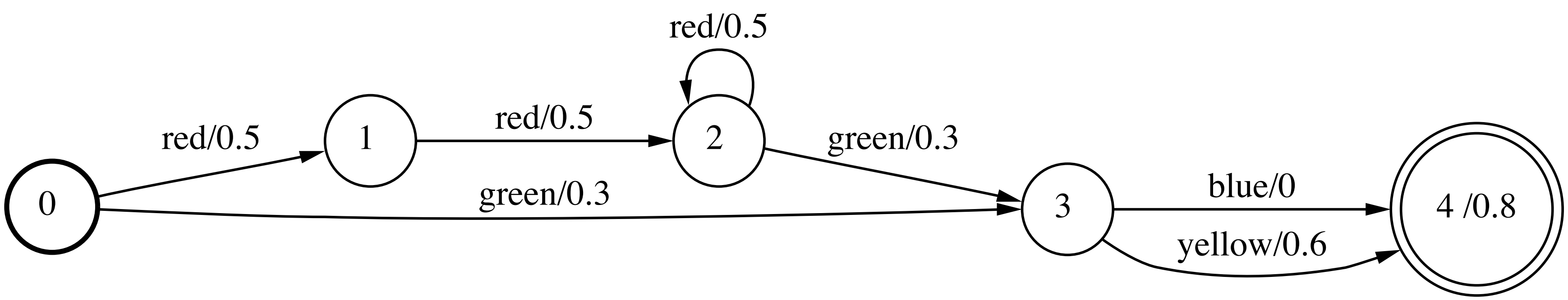


# Difference - Illustration

- **Program**: fsmdifference A.fsm B.fsm >C.fsm
- **Graphical representation**:



# This Lecture

- Weighted automata and transducers
- Rational operations
- Elementary unary operations
- Fundamental binary operations
- Optimization algorithms
- Search algorithms

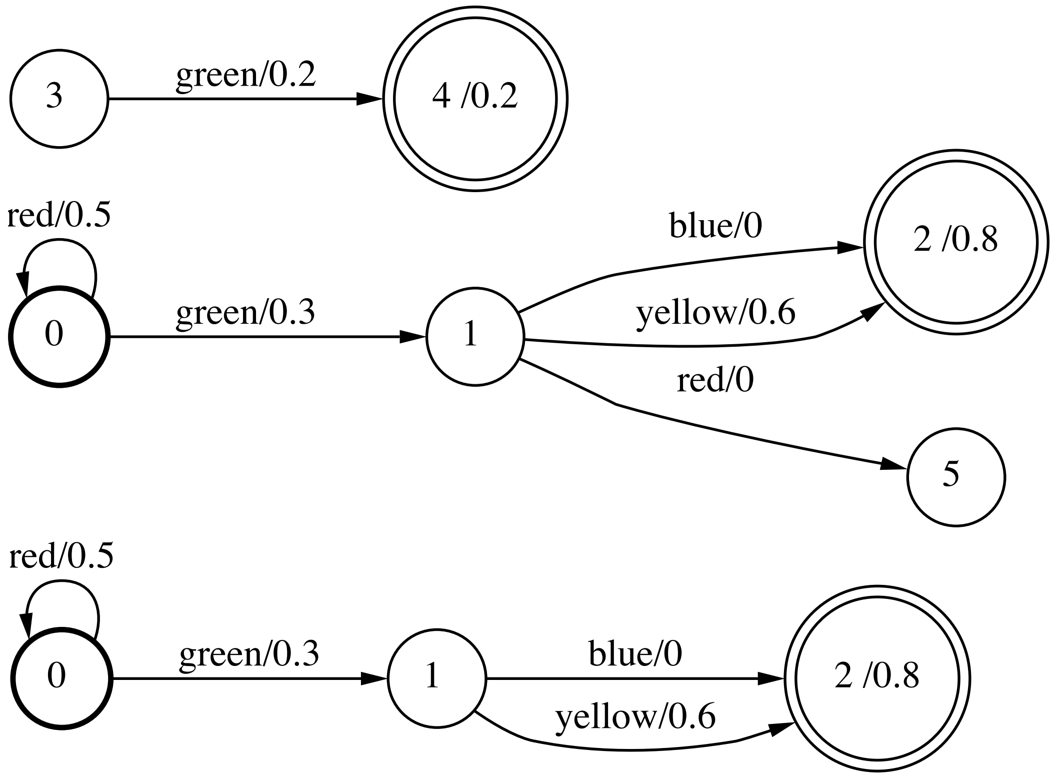
# Optimization Algorithms

- Connection: removes non-accessible/non-coaccessible states.
- $\epsilon$-Removal: removes $\epsilon$-transitions.
- Determinization: creates equivalent *deterministic* machine.
- Pushing: creates equivalent pushed/stochastic machine.
- Minimization: creates equivalent minimal deterministic machine.

- Conditions: there are specific semiring conditions for the use of these algorithms, e.g., not all weighted automata or transducers can be determinized using the determinization algorithm.

# Connection - Illustration

- **Program**: fsmconnect A.fsm >C.fsm
- **Graphical representation**:

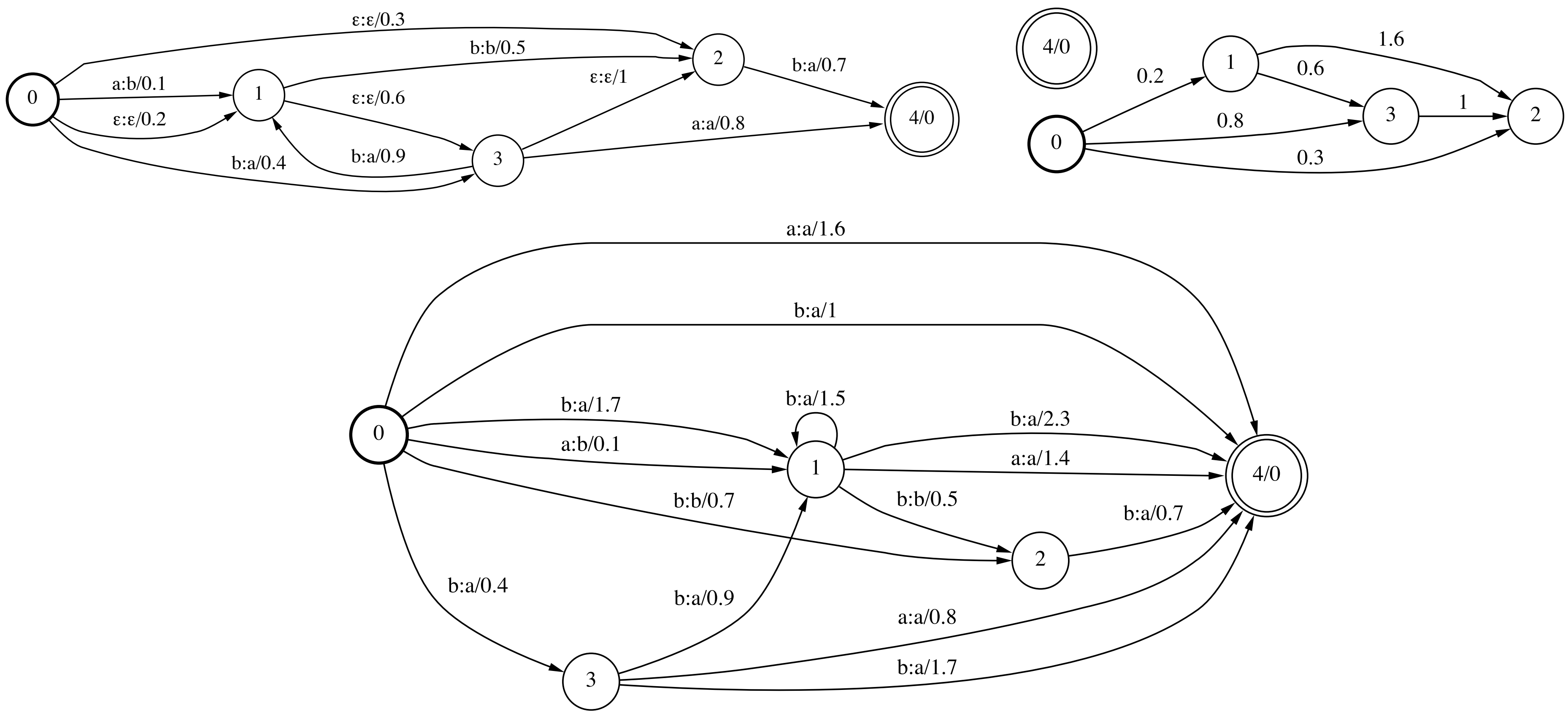


# Connection - Algorithm

- Definition:
  - Input: weighted transducer $T_1$ .
  - Output: equivalent weighted transducer $T_2$ with all states connected.
- Description:
  3. Depth-first search of $T_1$ from $I_1$ .
  4. Mark accessible and coaccessible states.
  5. Keep marked states and corresponding transitions.
- Complexity: linear $O(|Q_1| + |E_1|)$.

# ϵ-Removal - Illustration

- **Program**: fsmrmepsilon T.fsm >TP.fsm
- **Graphical representation**:



# ϵ-Removal - Algorithm

(MM, 2001)

- Definition:
  - Input: weighted transducer $T_1$ .
  - Output: equivalent WFST $T_2$ with no ϵ-transition.
- Description:
  - Computation of ϵ-closures.
  - Removal of ϵs.
- Complexity:
  - Acyclic $T_\epsilon$ : $O(|Q|^2 + |Q||E|(T_\oplus + T_\otimes))$.
  - General case (tropical semiring):

$$O(|Q||E| + |Q|^2 \log |Q|)$$

# Computation of ϵ-closures

- Definition: for $p$ in $Q$,

$$C[p] = \left\{(q, w) : q \in \epsilon[p],\ d[p, q] = w \neq \overline{0}\right\},$$

$$\text{where } d[p, q] = \bigoplus_{\pi \in P(p, \epsilon, q)} w[\pi].$$

- Problem formulation: all-pairs shortest-distance problem in $T_\epsilon$ ($T$ reduced to its ϵ-transitions).
  - closed semirings: generalization of Floyd-Warshall algorithm.
  - $k$-closed semirings: generic sparse shortest-distance algorithm.

# Determinization - Algorithm

(MM,1997)

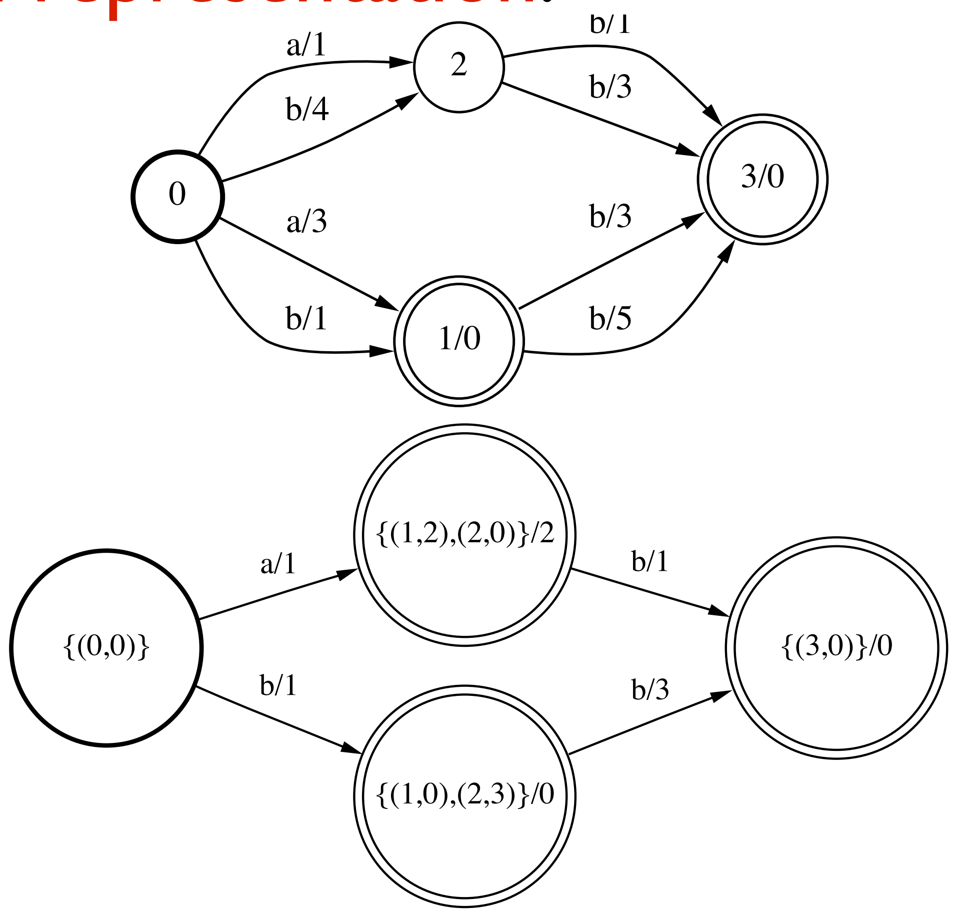
- Definition:
  - Input: weighted automaton or transducer $T_1$
  - Output: equivalent *subsequential* or *deterministic* machine $T_2$: has a unique initial state and no two transitions leaving the same state share the same input label.
- Description:
  3. Generalization of subset construction: *weighted subsets* $\{(q_1, w_1), \ldots, (q_n, w_n)\}$, where $w_i$s are remainder weights.
  4. Computation of the weight of resulting transitions.

# Determinization - Conditions

- Semiring: weakly left divisible semirings.
- Definition: *T* is *determinizable* when the determinization algorithm applies to *T*.
  - All unweighted automata are determinizable.
  - All acyclic machines are determinizable.
  - Not all weighted automata or transducers are determinizable.
  - Characterization based on the *twins property*.
- Complexity: exponential, but lazy implementation.

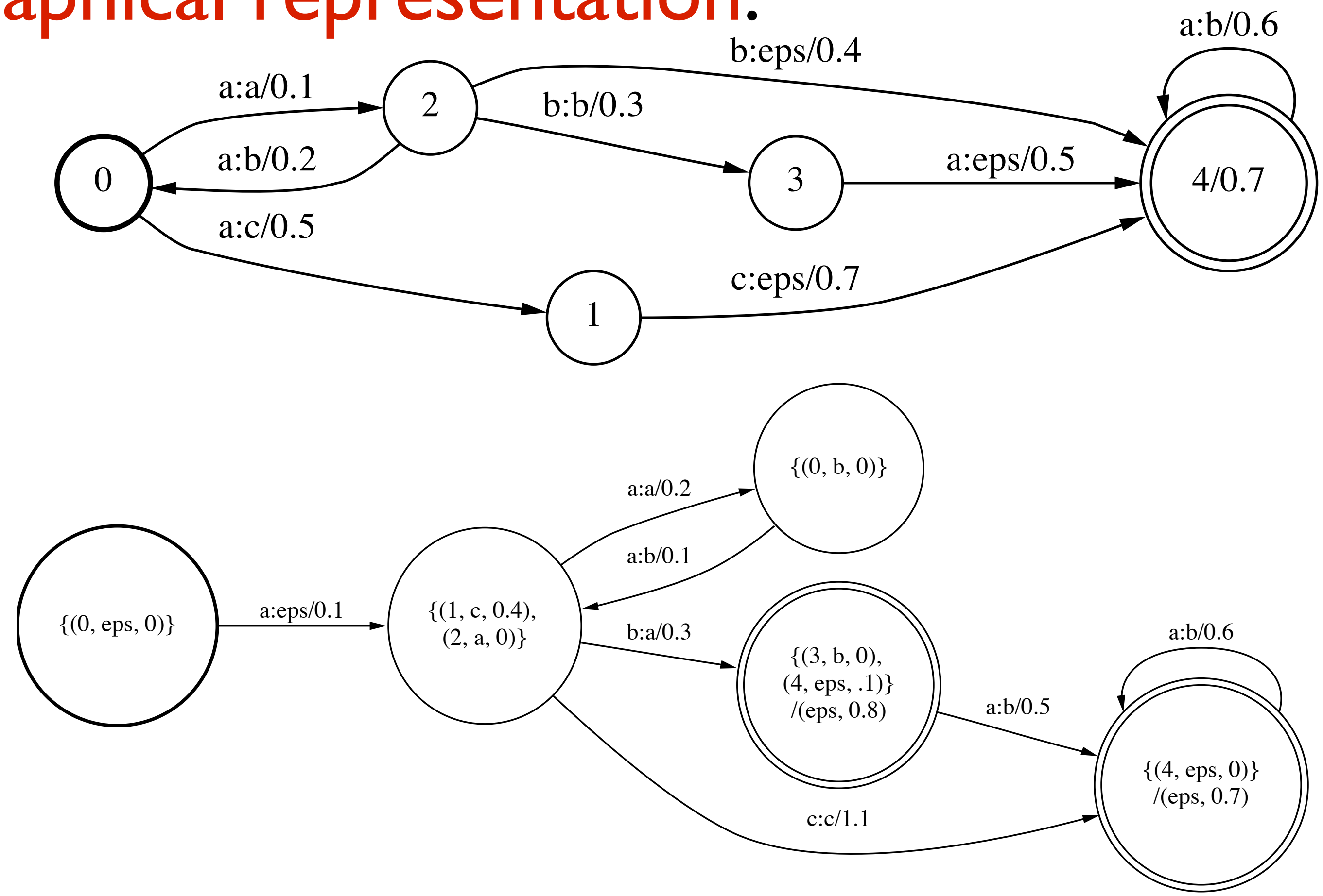
# Determinization of Weighted Automata - Illustration

- **Program**: fsmdeterminize A.fsm >D.fsm
- **Graphical representation**:



# Determinization of Weighted Transducers - Illustration

- **Program**: fsmdeterminize T.fsm >D.fsm
- **Graphical representation**:



# Pushing - Algorithm

(MM, 1997; 2004)

- Definition:
  - Input: weighted automaton or transducer $T_1$
  - Output: equivalent automaton or transducer $T_2$ such that the longest common prefix of all outgoing paths be $\epsilon$ or such that the sum of the weights of all outgoing transitions be $\overline{1}$ modulo the string or weight at the initial state.

- Description:
  1. Single-source shortest distance computation: for each state *q*,

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

  2. Reweighting: for each transition e such that $d[p[e]] \neq \overline{0}$,

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

- Conditions (automata case): weakly divisible semiring, zero-sum free semiring or automaton.
- Complexity:
  - automata case
    - acyclic case: linear $O(|Q| + |E|(T_\oplus + T_\otimes))$.
    - general case (tropical semiring):

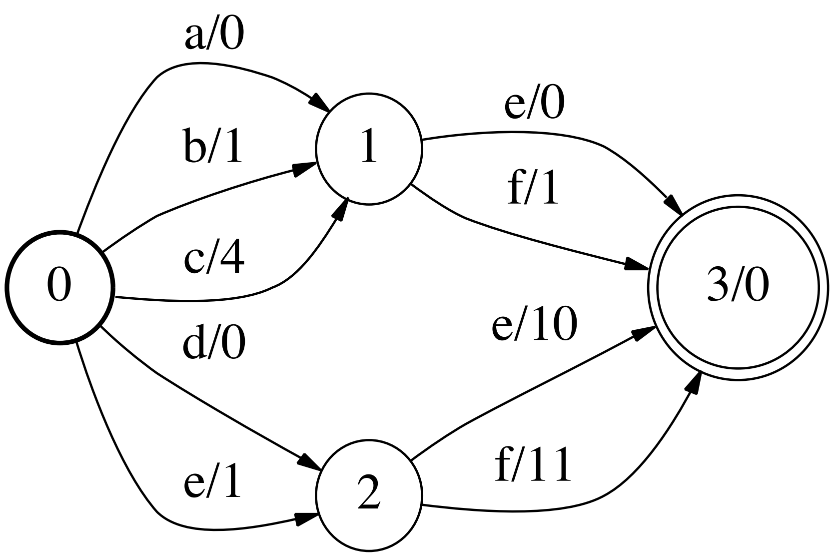
$$O(|Q| \log |Q| + |E|).$$

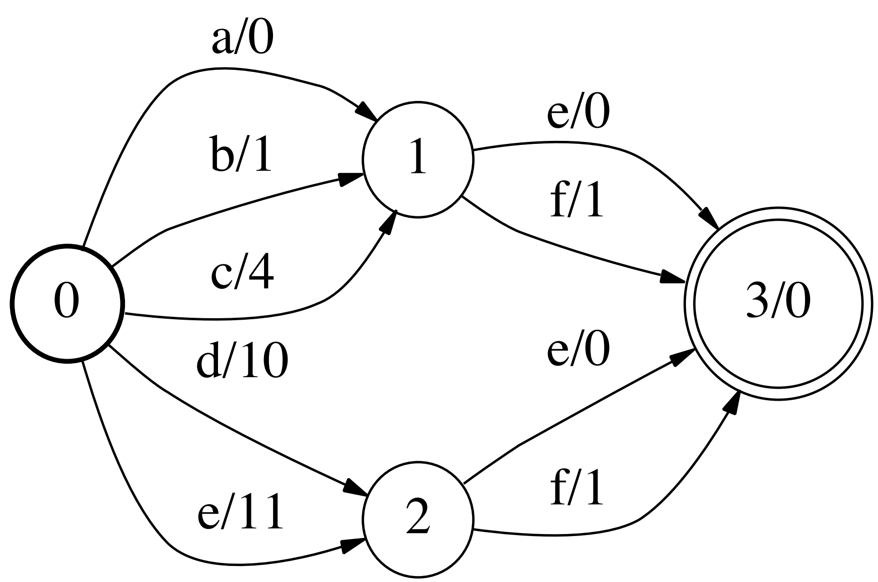
  - transducer case:

$$O((|P_{max}| + 1)\,|E|).$$

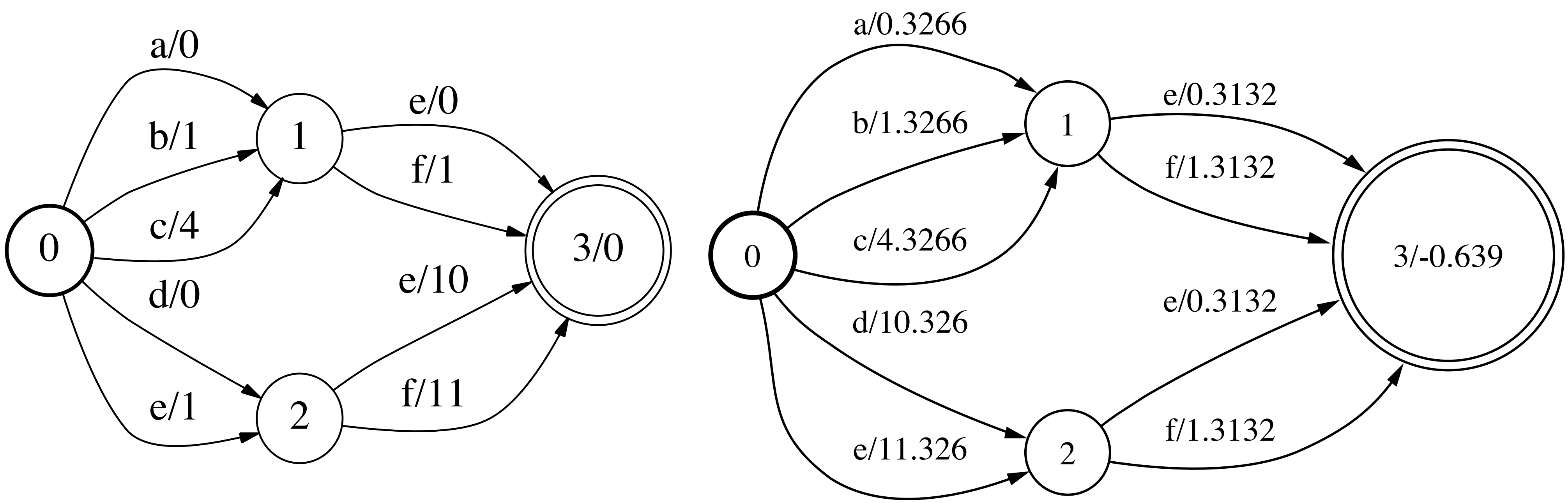
# Weight Pushing - Illustration

- Program: fsmpush -ic A.fsm >P.fsm
- Graphical representation:
  - Tropical semiring:



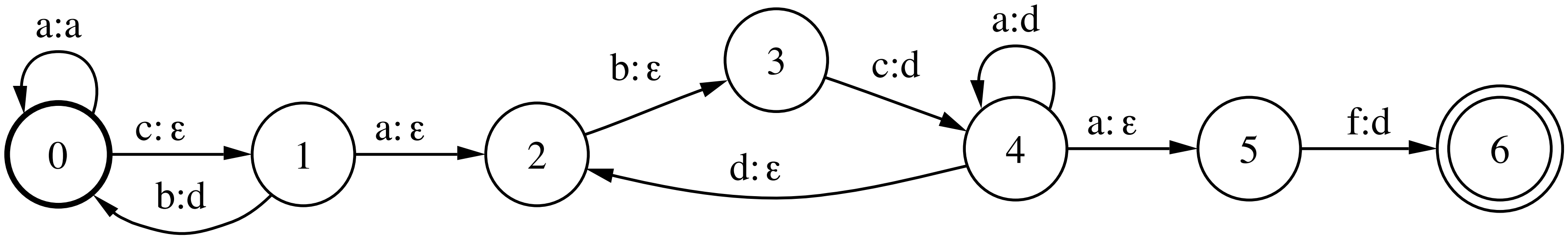



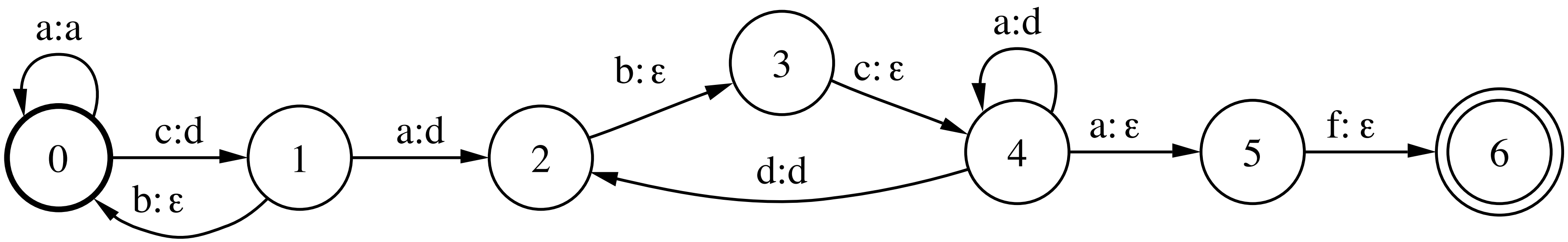
- Log semiring:



# Label Pushing - Illustration

- Program: fsmpush -il T.fsm >P.fsm
- Graphical representation:



# Minimization - Algorithm

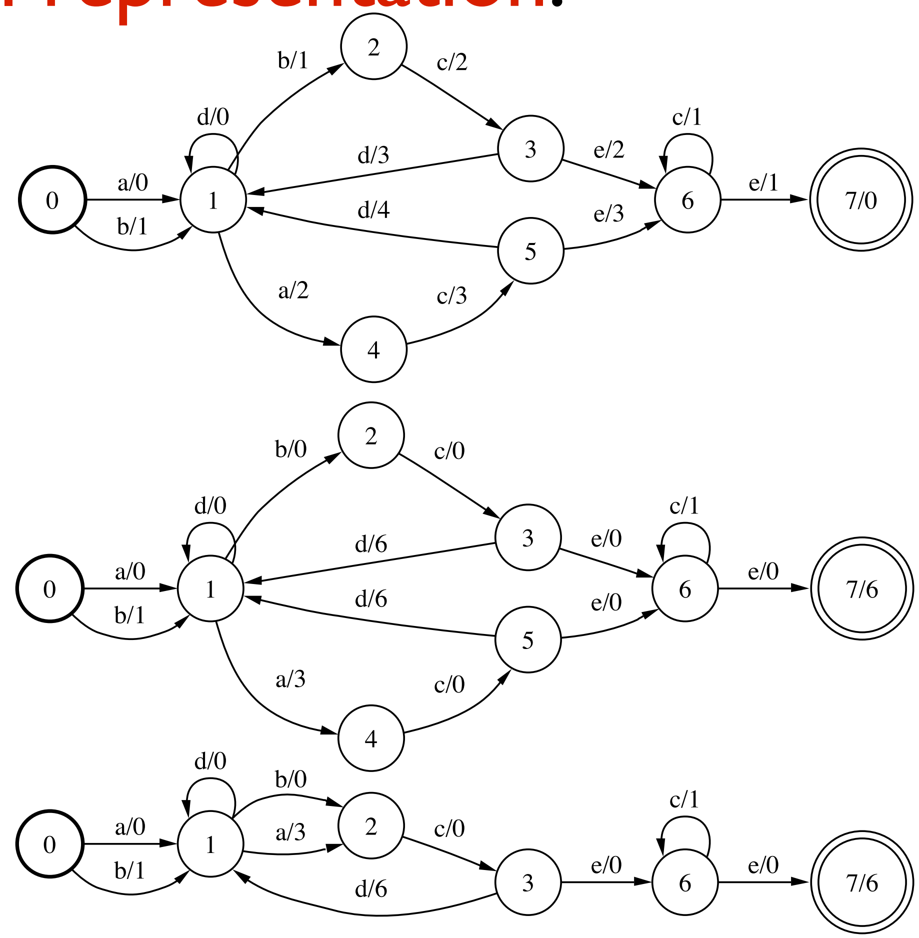
(MM,1997)

- Definition:
  - Input: deterministic weighted automaton or transducer $T_1$.
  - Output: equivalent deterministic automaton or transducer $T_2$ with the minimal number of states and transitions.
- Description:
  - Canonical representation: use pushing or other algorithm to standardize input automata.
  - Automata minimization: encode pairs (label, weight) as labels and use classical unweighted minimization algorithm.

- Complexity:
  - Automata case
    - acyclic case: linear, $O(|Q| + |E|(T_\oplus + T_\otimes))$.
    - general case (tropical semiring): $O(|E| \log |Q|)$.
  - Transducer case
    - acyclic case: $O(S + |Q| + |E| \left(|P_{max}| + 1\right))$.
    - general case (tropical semiring):
      $$O(S + |Q| + |E| \left(\log |Q| + |P_{max}|\right)).$$

# Minimization - Illustration

- **Program**: fsmminimize D.fsm >M.fsm
- **Graphical representation**:



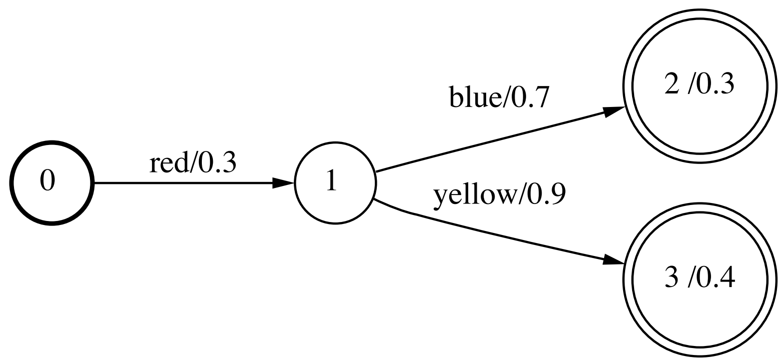
# Equivalence - Algorithm

- **Definition**:
  - Input: deterministic weighted automata $A$ and $B$.
  - Output: $\texttt{TRUE}$ iff $A$ and $B$ equivalent.
- **Description** (MM,1997):
  - Canonical representation: use pushing or other algorithm to standardize input automata.
  - Automata minimization: encode pairs (label, weight) as labels and use classical algorithm for testing the equivalence of unweighted automata.
- **Complexity**: (second stage is quasi-linear)

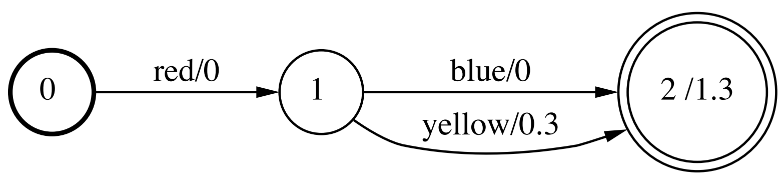
$$O(|E_1| + |E_2| + |Q_1| \log |Q_1| + |Q_2| \log |Q_2|).$$

# Equivalence - Illustration

- **Program**: fsmequiv [-v] D.fsm M.fsm
- **Graphical representation**:




$$=?$$

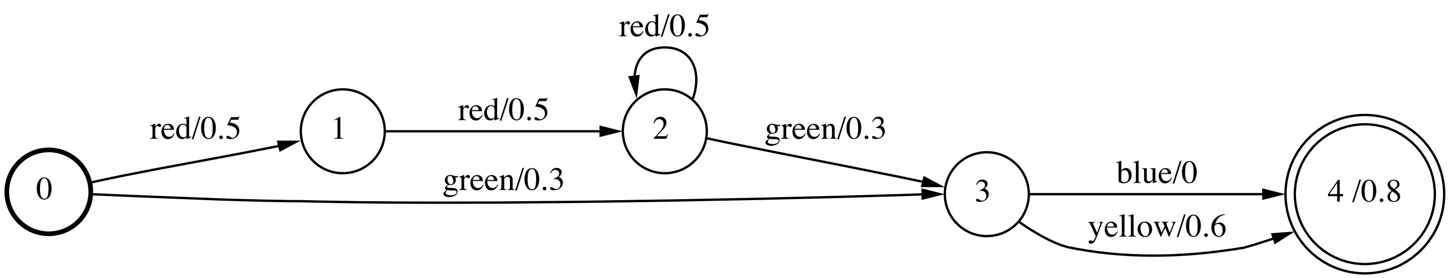


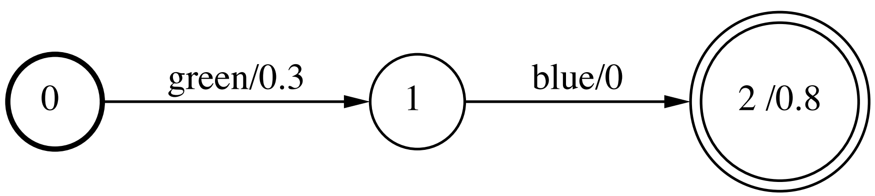
# This Lecture

- Weighted automata and transducers
- Rational operations
- Elementary unary operations
- Fundamental binary operations
- Optimization algorithms
- Search algorithms

# Single-Source Shortest-Distance Algorithms - Illustration

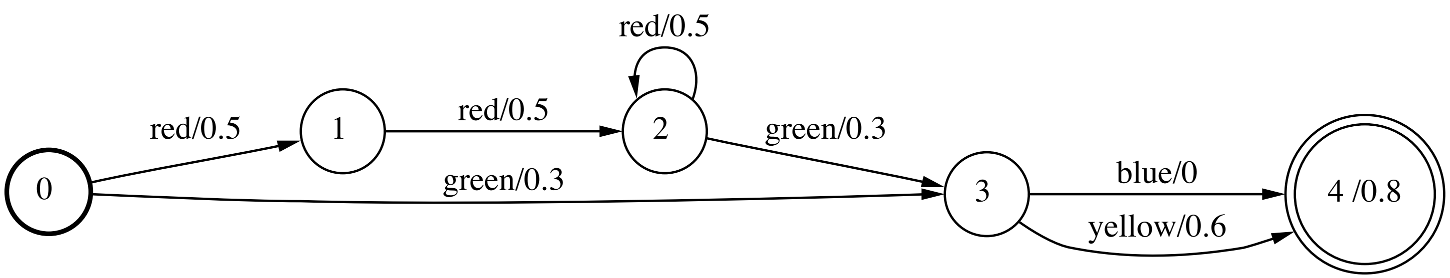
- Program: fsmbestpath [-n N] A.fsm >C.fsm
- Graphical representation:

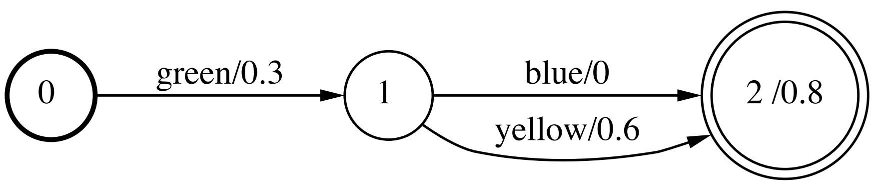


# Pruning - Illustration

- Program: fsmprune -c1.0 A.fsm >C.fsm
- Graphical representation:



# Summary

- FSM Library:
  - weighted finite-state transducers (semirings);
  - elementary unary operations (e.g., reversal);
  - rational operations (sum, product, closure);
  - fundamental binary operations (e.g., composition);
  - optimization algorithms (e.g., $\epsilon$-removal, determinization, minimization);
  - search algorithms (e.g., shortest-distance algorithms, *n*-best paths algorithms, pruning).

# References


- Cyril Allauzen and Mehryar Mohri. Efficient Algorithms for Testing the Twins Property. *Journal of Automata, Languages and Combinatorics*, 8(2):117-144, 2003.
- Cyril Allauzen, Mehryar Mohri, and Brian Roark. The Design Principles and Algorithms of a Weighted Grammar Library. *International Journal of Foundations of Computer Science*, 16(3): 403-421, 2005.
- Cyril Allauzen, Michael Riley, Johan Schalkwyk, Wojciech Skut, and Mehryar Mohri. OpenFst: a general and efficient weighted finite-state transducer library. In *Proceedings of the Ninth International Conference on Implementation and Application of Automata, (CIAA 2007), volume 4783 of Lecture Notes in Computer Science*, pages 11-23. Springer, Berlin, 2007.
- Mehryar Mohri. Finite-State Transducers in Language and Speech Processing. *Computational Linguistics*, 23:2, 1997.
- Mehryar Mohri. Weighted Grammar Tools: the GRM Library. In *Robustness in Language and Speech Technology*. pages 165-186. Kluwer Academic Publishers, The Netherlands, 2001.
- Mehryar Mohri. Statistical Natural Language Processing. In M. Lothaire, editor, *Applied Combinatorics on Words*. Cambridge University Press, 2005.

# References


- Mehryar Mohri. Generic Epsilon-Removal and Input Epsilon-Normalization Algorithms for Weighted Transducers. *International Journal of Foundations of Computer Science*, 13(1): 129-143, 2002.

- Mehryar Mohri, Fernando C. N. Pereira, and Michael Riley. The Design Principles of a Weighted Finite-State Transducer Library. *Theoretical Computer Science*, 231:17-32, January 2000.

- Mehryar Mohri, Fernando C. N. Pereira, and Michael Riley. Weighted Automata in Text and Speech Processing. In *Proceedings of the 12th biennial European Conference on Artificial Intelligence (ECAI-96), Workshop on Extended finite state models of language*. Budapest, Hungary, 1996. John Wiley and Sons, Chichester.

- Mehryar Mohri and Michael Riley. A Weight Pushing Algorithm for Large Vocabulary Speech Recognition. In *Proceedings of the 7th European Conference on Speech Communication and Technology (Eurospeech'01)*. Aalborg, Denmark, September 2001.

- Fernando Pereira and Michael Riley. *Finite State Language Processing*, chapter Speech Recognition by Composition of Weighted Finite Automata. The MIT Press, 1997.