OpenFst: An Open-Source, Weighted Finite-State Transducer Library and its Applications to Speech and Language

Part I. *Theory and Algorithms*

Overview

1. Preliminaries
   - Semirings
   - Weighted Automata and Transducers
2. Algorithms
   - Rational Operations
   - Elementary Unary Operations
   - Fundamental Binary Operations
   - Optimization Algorithms
   - Search Operations
   - Fundamental String Algorithms

Weight Sets: Semirings

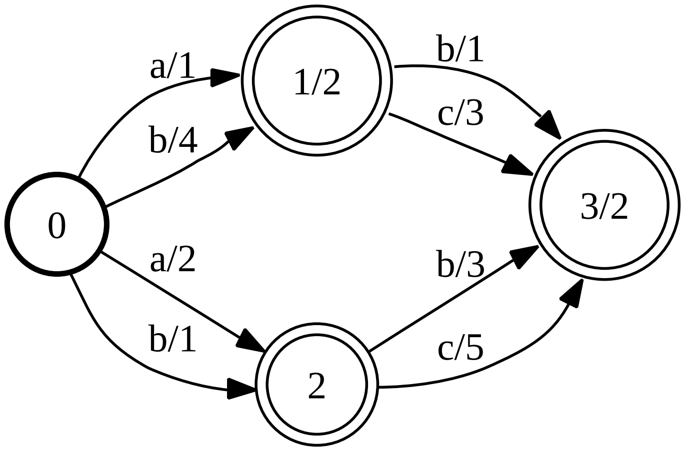
A *semiring* $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ = a ring that may lack negation.

- 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).

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
String	$\Sigma^* \cup \{\infty\}$	$\wedge$	$\cdot$	∞	ϵ

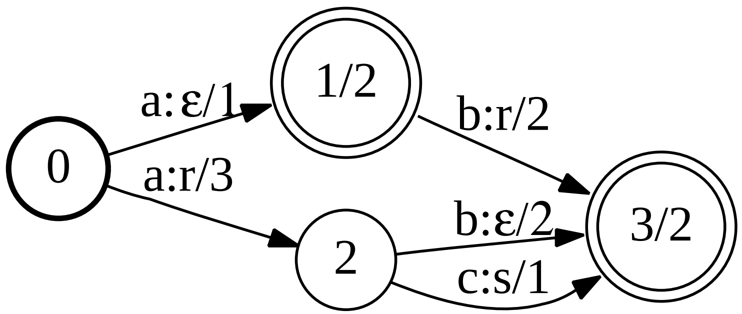
$\oplus_{\log}$ is defined by: $x \oplus_{\log} y = -\log(e^{-x} + e^{-y})$ and $\wedge$ is longest common prefix. The string semiring is a *left semiring*.

Weighted Automaton/Acceptor



Probability semiring $(\mathbb{R}_+, +, \times, 0, 1)$	Tropical semiring $(\mathbb{R}_+ \cup \{\infty\}, \min, +, \infty, 0)$
$[\![A]\!](ab) = 14$	$[\![A]\!](ab) = 4$
$(1 \times 1 \times 2 + 2 \times 3 \times 2 = 14)$	$(\min(1 + 1 + 2, 3 + 2 + 2) = 4)$

Weighted Transducer



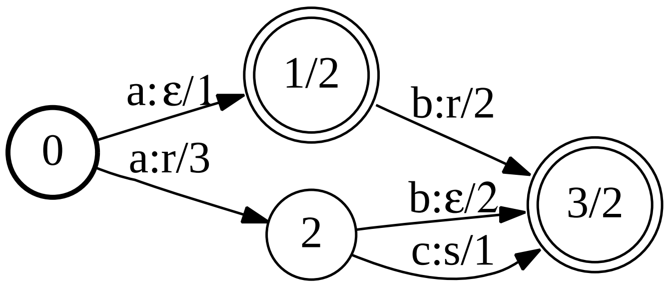
Probability semiring $(\mathbb{R}_+, +, \times, 0, 1)$	Tropical semiring $(\mathbb{R}_+ \cup \{\infty\}, \min, +, \infty, 0)$
$[\![T]\!](ab, r) = 16$	$[\![T]\!](ab, r) = 5$
$(1 \times 2 \times 2 + 3 \times 2 \times 2 = 16)$	$(\min(1 + 2 + 2, 3 + 2 + 2) = 5)$

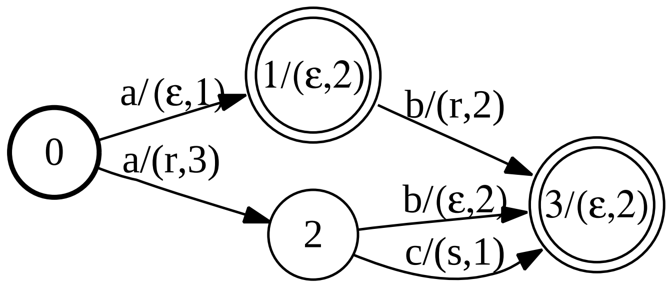
Transducers as Weighted Automata

A transducer T is *functional* iff for each x there exists at most one y such that $[\![T]\!](x, y) \neq \overline{0}$

- An unweighted functional transducer can be seen as as:
  $\rightarrow$ a weighted automata over the string semiring $(\Sigma^* \cup \{\infty\}, \wedge, \cdot, \infty, \epsilon)$

- A weighted functional transducer over the semiring $\mathbb{K}$ can be seen as:
  $\rightarrow$ a weighted automata over the cartesian product of the string semiring and $\mathbb{K}$



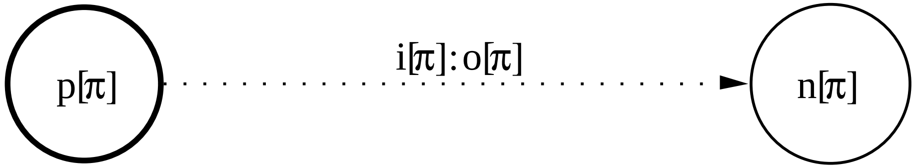


$[\![T]\!](ab, r) = 5$ $\qquad$ $[\![A]\!](ab) = (r, 5)$

[Tropical semiring $(\mathbb{R}_+ \cup \{\infty\}, \min, +, \infty, 0)$]

Definitions and Notation – Paths

- Path π
  - Origin or previous state: $p[\pi]$.
  - Destination or next state: $n[\pi]$.
  - Input label: $i[\pi]$.
  - Output label: $o[\pi]$.



- Sets of paths
  - $P(R_1, R_2)$: set of all 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.

Definitions and Notation – Automata and Transducers

1. General Definitions
   - Alphabets: input Σ, output Δ.
   - 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 weight function $\lambda : I \to \mathbb{K}$

     final weight function $\rho : F \to \mathbb{K}$.
2. Machines
   - Automaton $A = (\Sigma, Q, I, F, E, \lambda, \rho)$ with for all $x \in \Sigma^*$:

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

   - Transducer $T = (\Sigma, \Delta, Q, I, F, E, \lambda, \rho)$ with for all $x \in \Sigma^*, y \in \Delta^*$:

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

Rational Operations – Algorithms

- Definitions

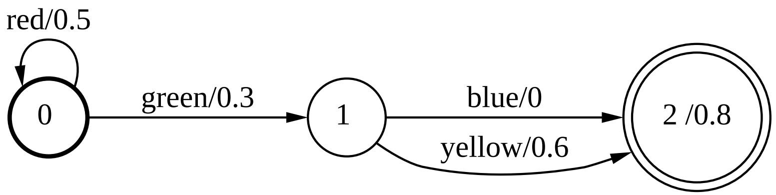
Operation	Definition and Notation
Sum	$[\![T_1 \oplus T_2]\!](x,y) = [\![T_1]\!](x,y) \oplus [\![T_2]\!](x,y)$
Product	$[\![T_1 \otimes T_2]\!](x,y) = \bigoplus_{x=x_1 x_2, y=y_1 y_2} [\![T_1]\!](x_1,y_1) \otimes [\![T_2]\!](x_2,y_2)$
Closure	$[\![T^*]\!](x,y) = \bigoplus_{n=0}^{\infty} [\![T^n]\!](x,y)$

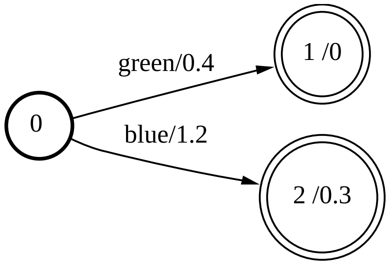
- Conditions on the closure operation: condition on T: e.g. weight of ϵ-cycles $= \overline{0}$ (*regulated transducers*), or semiring condition: e.g. $\overline{1} \oplus x = \overline{1}$ as with the tropical semiring (*locally closed semirings*).

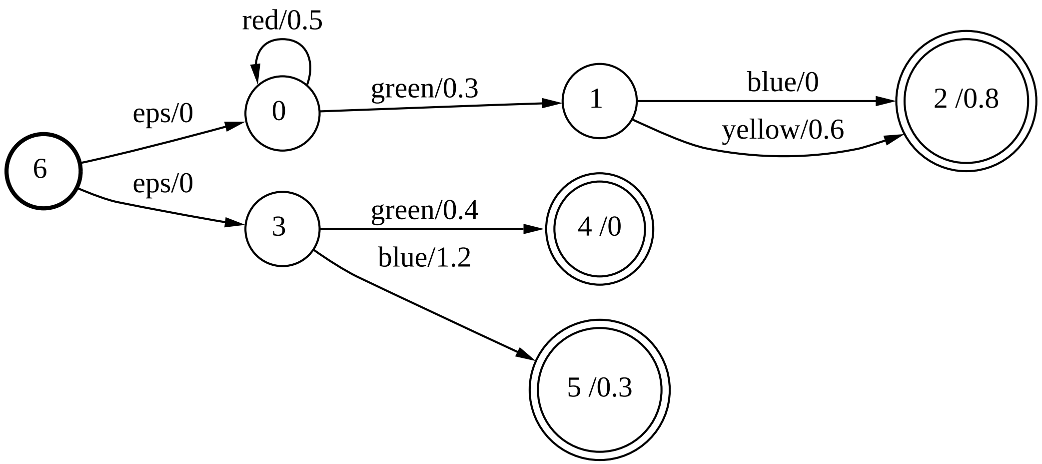
- Complexity and implementation
  - Complexity (linear): $O((|E_1| + |Q_1|) + (|E_2| + |Q_2|))$ or $O(|Q| + |E|)$.
  - Lazy implementation.

Sum (Union) – Illustration

- Definition: $[\![T_1 \oplus T_2]\!](x, y) = [\![T_1]\!](x, y) \oplus [\![T_2]\!](x, y)$
- Example:

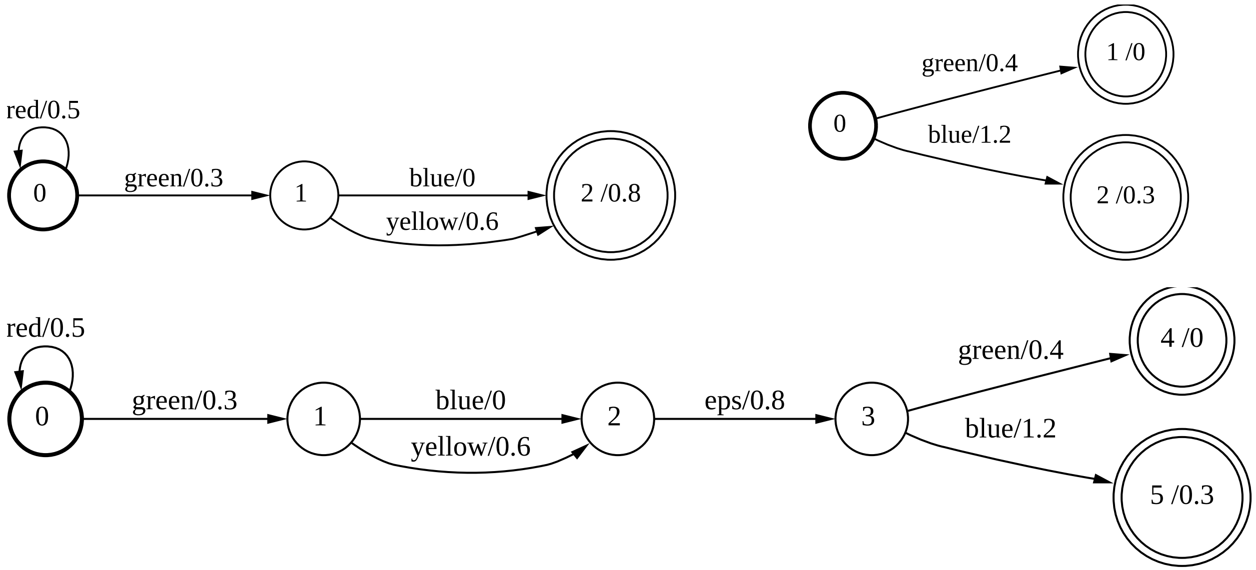






Product (Concatenation) – Illustration

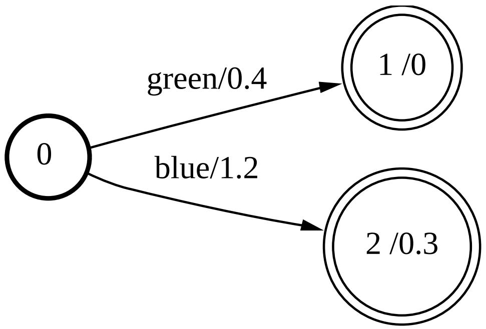
- Definition: $[\![T_1 \otimes T_2]\!](x, y) = \bigoplus_{x=x_1 x_2, y=y_1 y_2} [\![T_1]\!](x_1, y_1) \otimes [\![T_2]\!](x_2, y_2)$
- Example:

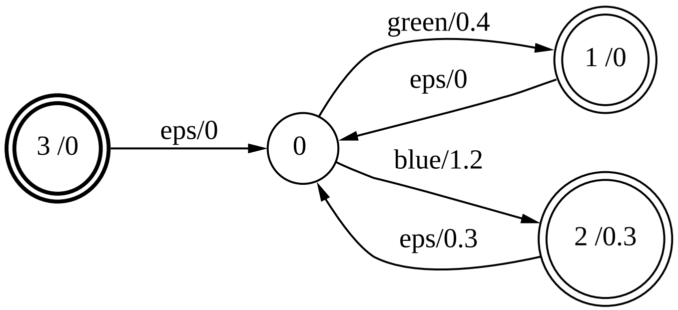


Closure – Illustration

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

- Example:





Some Elementary Unary Operations – Algorithms

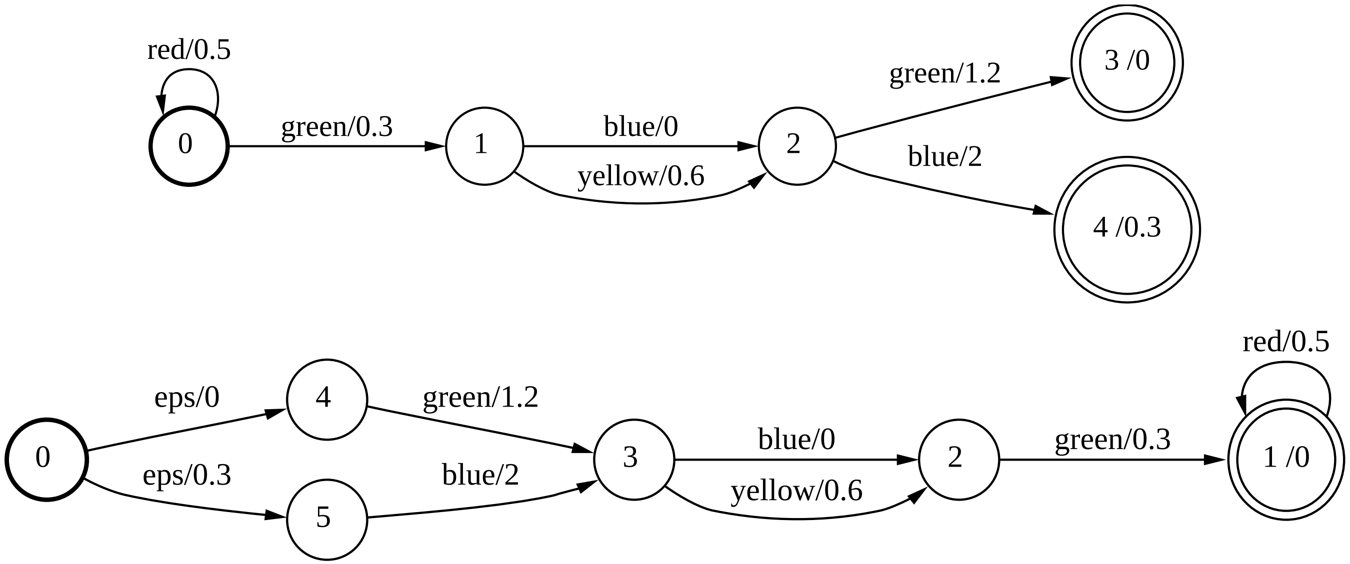
- Definitions

Operation	Definition and Notation	Lazy Implementation
Reversal	$[\![\widetilde{T}]\!](x,y) = [\![T]\!](\widetilde{x},\widetilde{y})$	No
Inversion	$[\![T^{-1}]\!](x,y) = [\![T]\!](y,x)$	Yes
Projection	$[\![\Pi_1(T)]\!](x) = \bigoplus_y [\![T]\!](x,y)$	Yes
	$[\![\Pi_2(T)]\!](x) = \bigoplus_y [\![T]\!](y,x)$	

- Complexity and implementation
  - Complexity (linear): $O(|Q| + |E|)$.
  - Lazy implementation (see table).

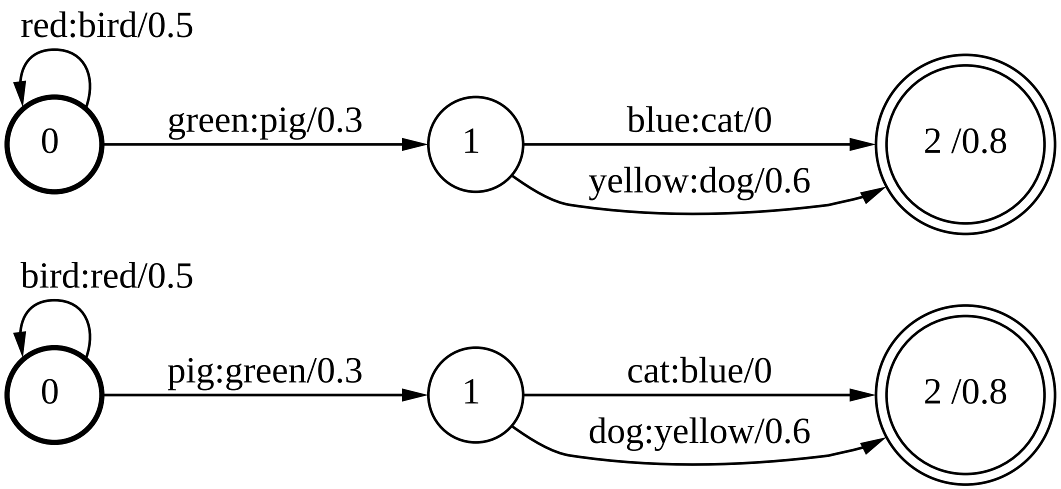
Reversal – Illustration

- Definition: $[\![\widetilde{T}]\!](x, y) = [\![T]\!](\widetilde{x}, \widetilde{y})$
- Example:



Inversion – Illustration

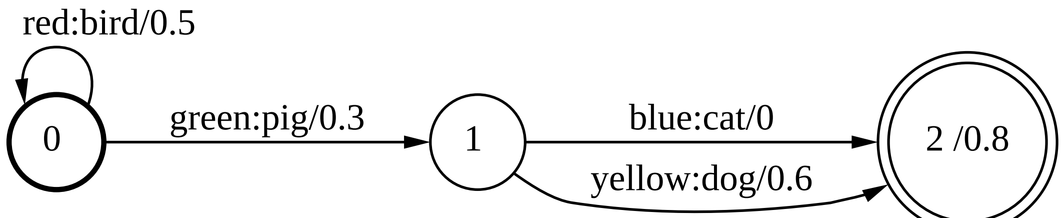
- Definition: $[\![T^{-1}]\!](x, y) = [\![T]\!](y, x)$
- Example:

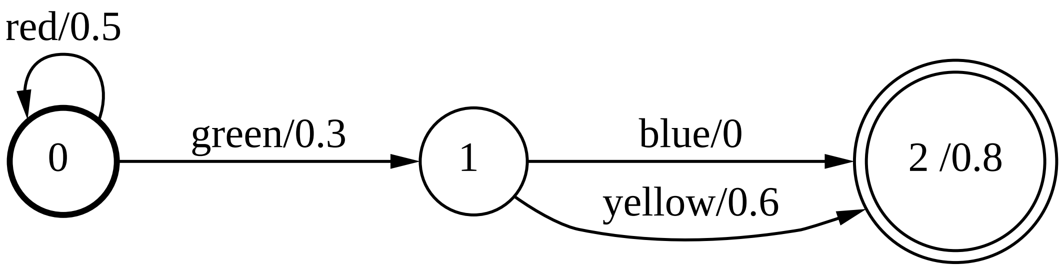


Projection – Illustration

- **Definition:** $[\![\Pi_1(T)]\!](x) = \bigoplus_{y} [\![T]\!](x, y)$

- **Example:**





Some Fundamental Binary Operations – Algorithms

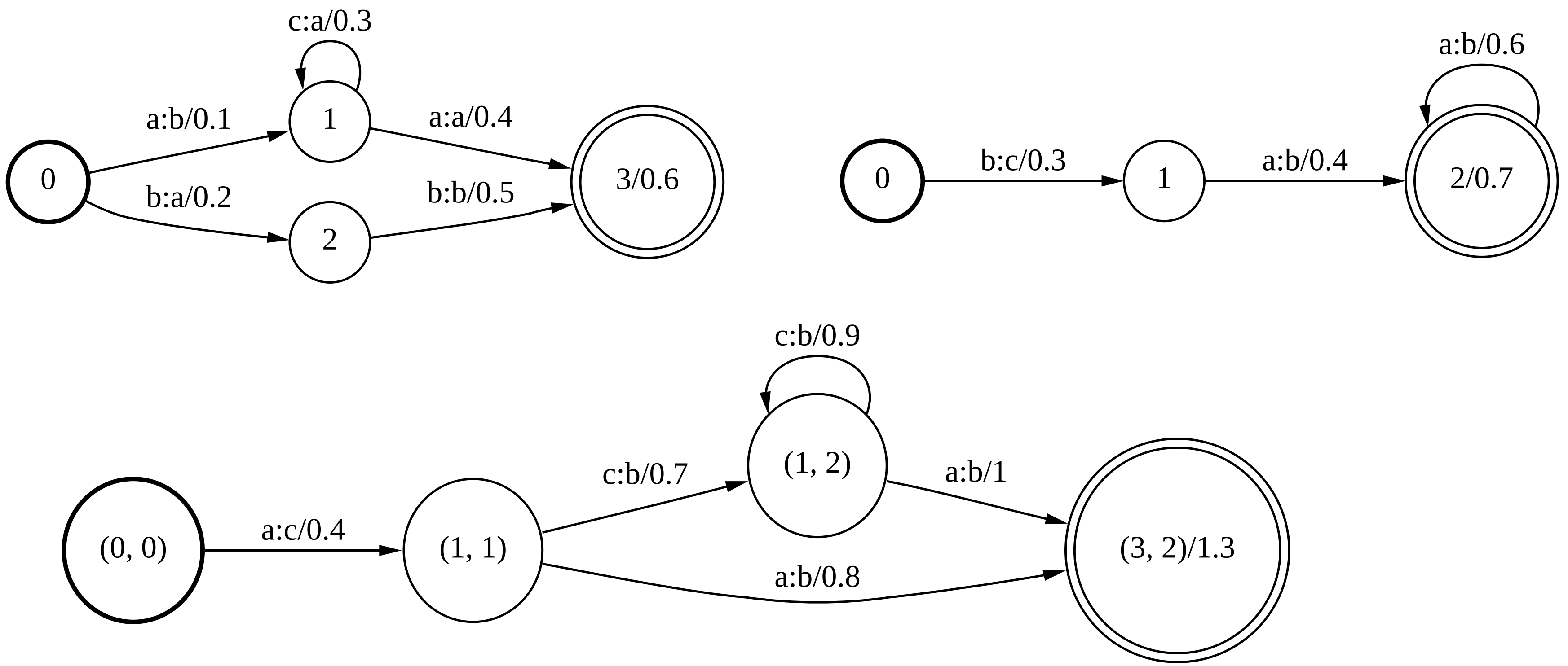
- Definitions

OPERATION	DEFINITION AND NOTATION	CONDITION
Composition	$[\![T_1 \circ T_2]\!](x, y) = \bigoplus_z [\![T_1]\!](x, z) \otimes [\![T_2]\!](z, y)$	$\mathbb{K}$ commutative
Intersection	$[\![A_1 \cap A_2]\!](x) = [\![A_1]\!](x) \otimes [\![A_2]\!](x)$	$\mathbb{K}$ commutative
Difference	$[\![A_1 - A_2]\!](x) = [\![A_1 \cap \overline{A_2}]\!](x)$	A_2 unweighted & deterministic

- Complexity and implementation
  - Complexity (quadratic): $O((|E_1| + |Q_1|)\,(|E_2| + |Q_2|))$.
  - Path multiplicity in presence of ϵ-transitions: ϵ-filter.
  - Lazy implementation.

Composition – Illustration

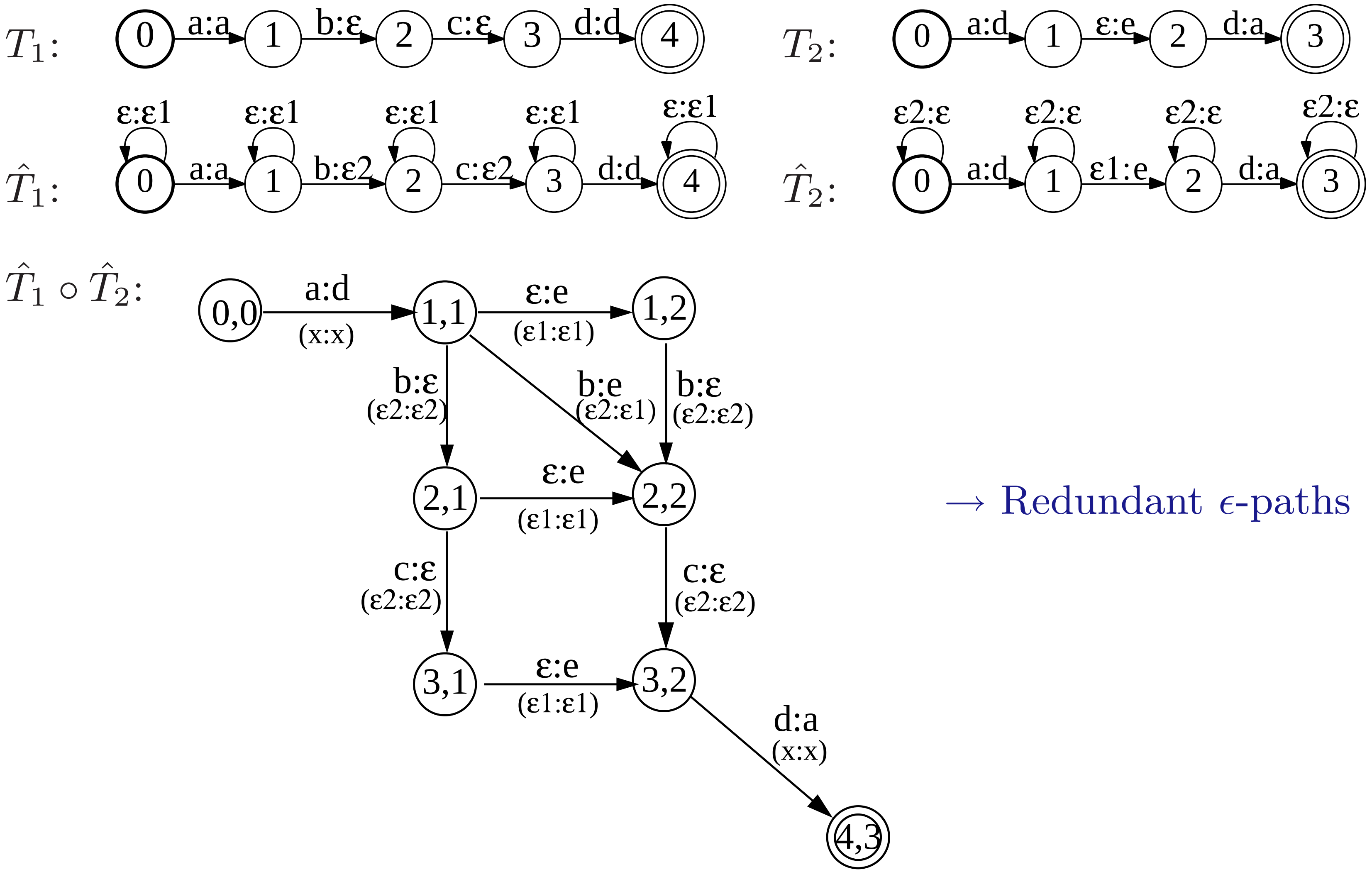
- Definition: $[\![T_1 \circ T_2]\!](x, y) = \bigoplus_z [\![T_1]\!](x, z) \otimes [\![T_2]\!](z, y)$
- Example:



Composition – Pseudocode

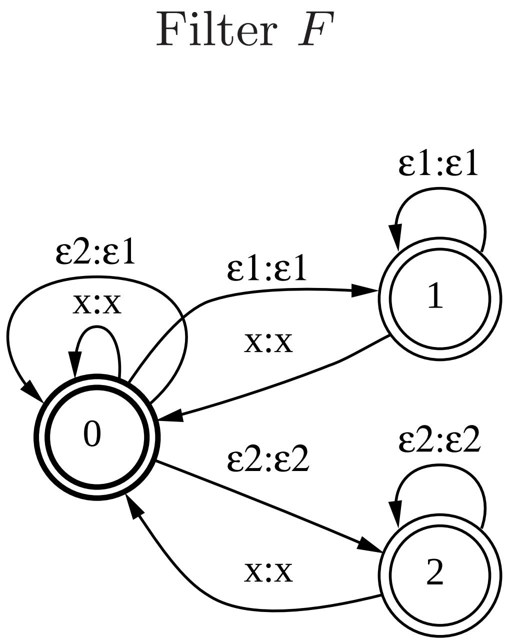
```
Composition(T_1, T_2)
S ← Q ← I_1 × I_2
while S ≠ ∅ do
        (q_1, q_2) ← Head(S)
        Dequeue(S)
        if (q_1, q_2) ∈ I_1 × I_2 then
                I ← I_1 × I_2
                λ(q_1, q_2) ← λ_1(q_1) ⊗ λ_2(q_2)
        if (q_1, q_2) ∈ F_1 × F_2 then
                F ← F ∪ {(q_1, q_2)}
                ρ(q_1, q_2) ← ρ_1(q_1) ⊗ ρ_2(q_2)
        for each(e_1, e_2) such that o[e_1] = i[e_2] do
                if (n[e_1], n[e_2]) ∉ Q then
                        Q ← Q ∪ {(n[e_1], n[e_2])}
                        Enqueue(S, (n[e_1], n[e_2]))
                E ← E ∪ {((q_1, q_2), i[e_1], o[e_2], w[e_1] ⊗ w[e_2], (n[e_1], n[e_2]))}
return T = (Σ, Δ, Q, I, F, E, λ, ρ)
```

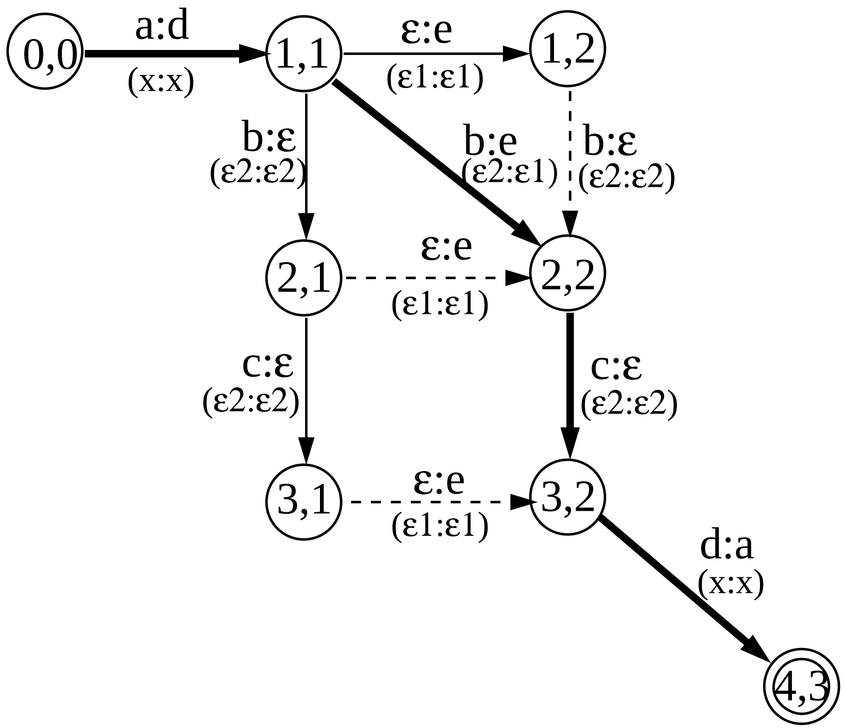
Multiplicity & ϵ-Transitions – Problem



$\rightarrow$ Redundant ϵ-paths

Solution – Filter F for Composition

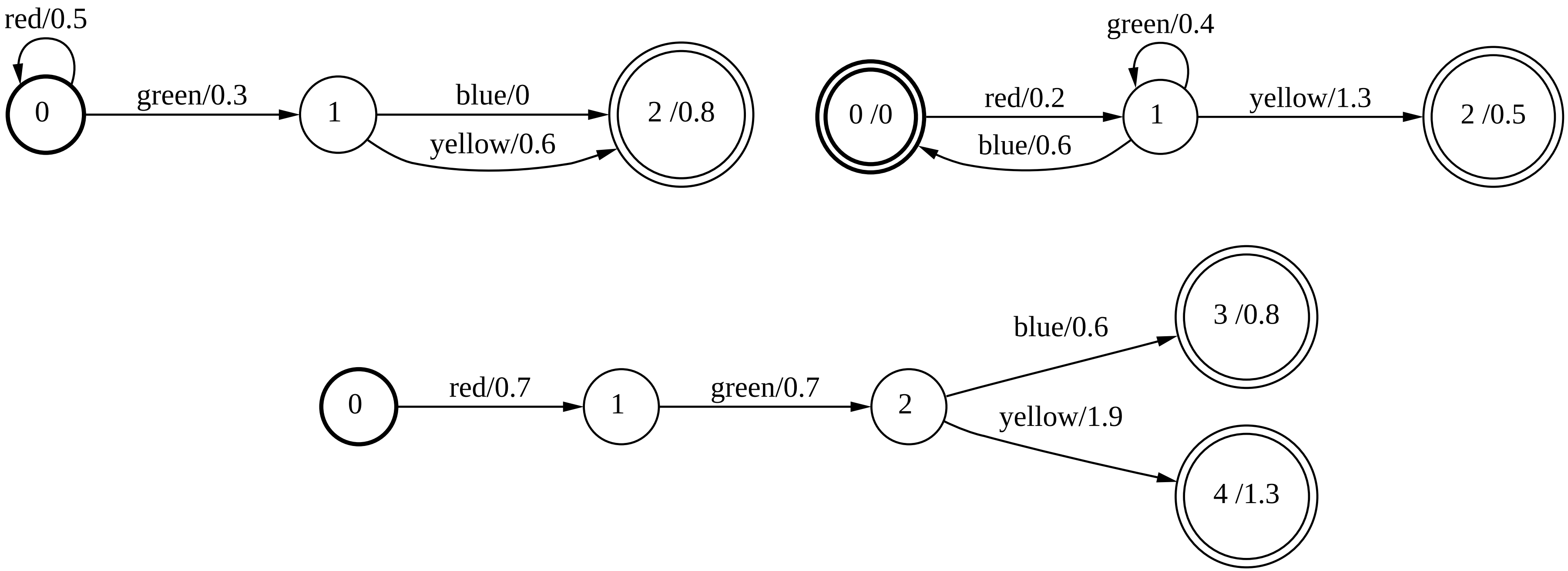




$\longrightarrow$ Replace $T_1 \circ T_2$ by $\hat{T}_1 \circ F \circ \hat{T}_2$.

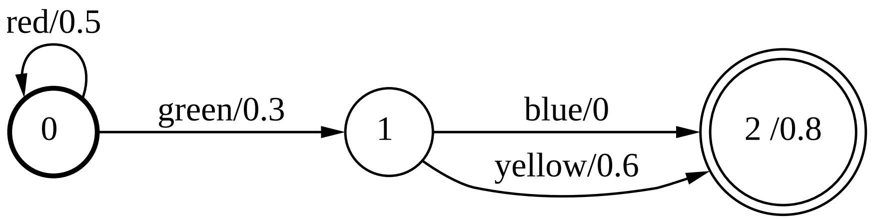
Intersection – Illustration

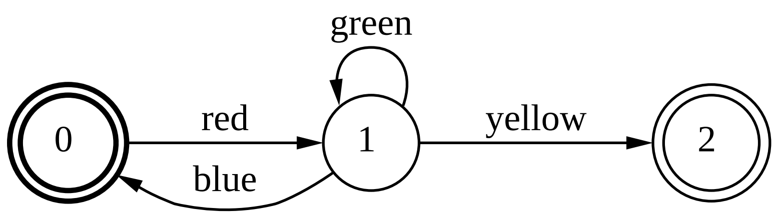
- Definition: $[\![A_1 \cap A_2]\!](x) = [\![A_1]\!](x) \otimes [\![A_2]\!](x)$
- Example:

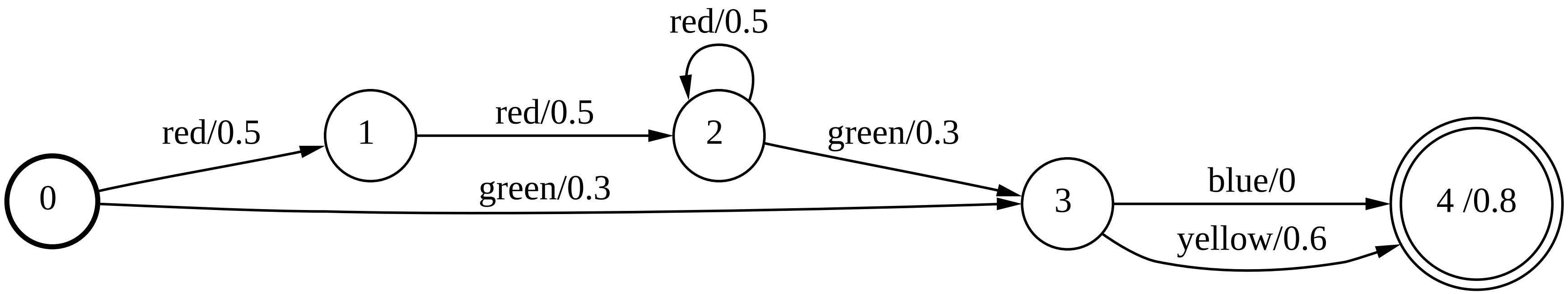


Difference – Illustration

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







Optimization Algorithms – Overview

- Definitions

Operation	Description
Connection	Removes non-accessible/non-coaccessible states
ϵ-Removal	Removes ϵ-transitions
Determinization	Creates equivalent deterministic machine
Pushing	Creates equivalent pushed/stochastic machine
Minimization	Creates equivalent minimal deterministic machine

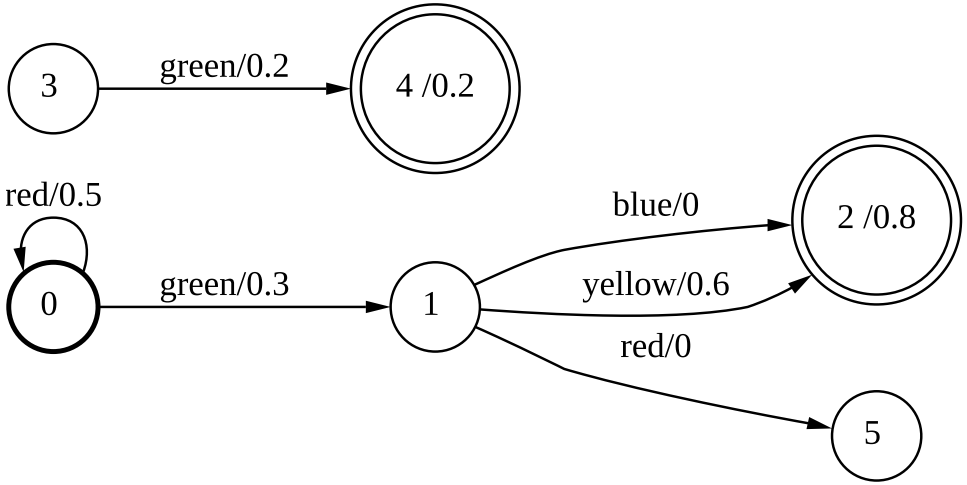
- Conditions: There are specific semiring conditions for the use of these algorithms. Not all weighted automata or transducers can be determinized using that algorithm.

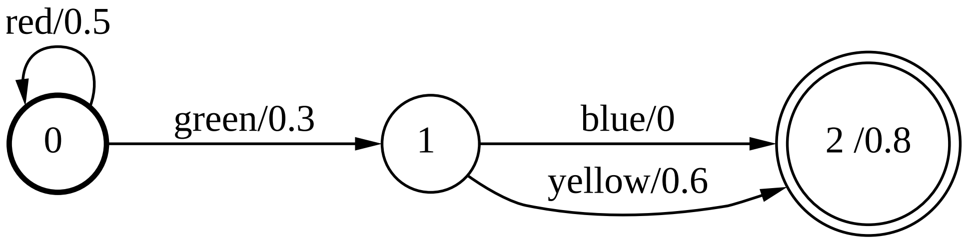
Connection – Algorithm

- Definition
  - Input: weighted transducer T_1.
  - Output: weighted transducer $T_2 \equiv T_1$ with all states connected.
- Description
  1. Depth-first search of T_1 from I_1.
  2. Mark accessible and coaccessible states.
  3. Keep marked states and corresponding transitions.
- Complexity and implementation
  - Complexity (linear): $O(|Q_1| + |E_1|)$.
  - No natural lazy implementation.

Connection – Illustration

- **Definition:** Removes non-accessible/non-coaccessible states
- **Example:**





ϵ-Removal – Algorithm

- Definition
  - Input: weighted transducer T_1 with ϵ-transitions.
  - Output: weighted transducer $T_2 \equiv T_1$ with no ϵ-transition.
- Description (two stages):
  1. Computation of ϵ-closures: for any state p, states q that can be reached from p via ϵ-paths and the total weight of the ϵ-paths from p to q.

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

     with:

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

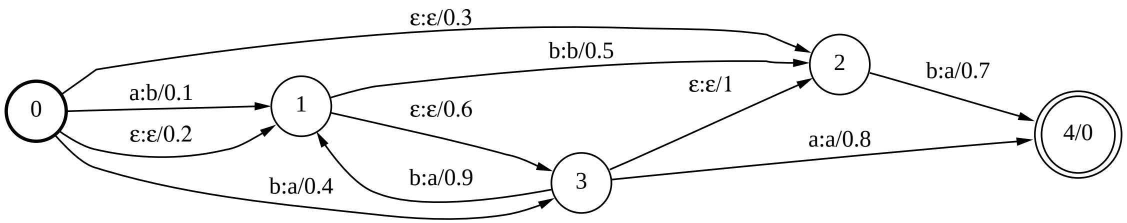
  2. Removal of ϵ's: actual removal of ϵ-transitions and addition of new transitions.

$\Longrightarrow$ All-pair $\mathbb{K}$-shortest-distance problem in T_ϵ (T reduced to its ϵ-transitions).

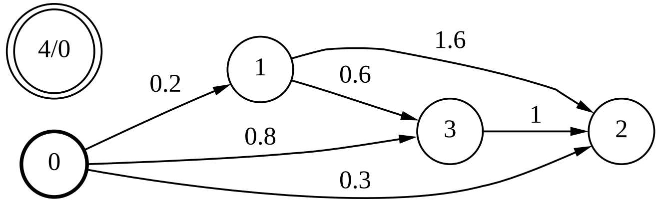
- Complexity and implementation
  - All-pair shortest-distance algorithm in T_ϵ.
    - k-Closed semirings (for T_ϵ) or approximation: generic sparse shortest-distance algorithm [See references].
    - Closed semirings: Floyd-Warshall or Gauss-Jordan elimination algorithm with decomposition of T_ϵ into strongly connected components [See references],
      space complexity (quadratic): $O(|Q|^2 + |E|)$.
      time complexity (cubic): $O(|Q|^3(T_\oplus + T_\otimes + T_*))$.
  - Complexity:
    - Acyclic T_ϵ: $O(|Q|^2 + |Q||E|(T_\oplus + T_\otimes))$.
    - General case (tropical semiring): $O(|Q||E| + |Q|^2 \log |Q|)$.
  - Lazy implementation: integration with on-the-fly weighted determinization.

ϵ-Removal – Illustration

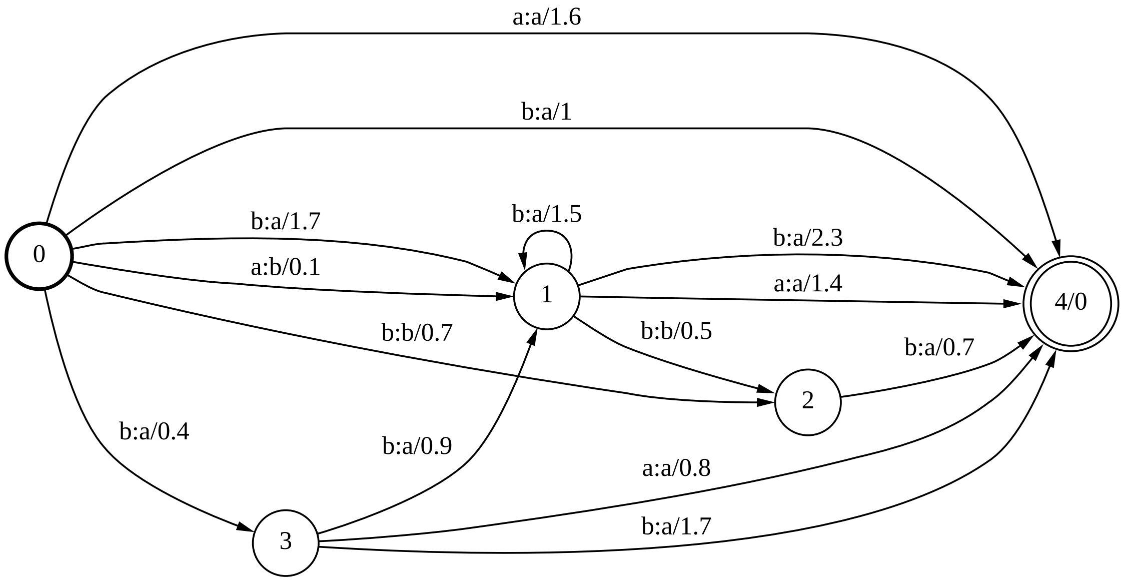
- **Definition:** Removes ϵ-transitions
- **Example:**



T



ϵ-Distances



rmeps(T)

Determinization – Algorithm

- Definition
  - Input: *determinizable* weighted automaton or transducer M_1.
  - Output: $M_2 \equiv M_1$ *subsequential* or *deterministic*: M_2 has a unique initial state and no two transitions leaving the same state share the same input label.
- Description
  1. Generalization of subset construction: weighted subsets $\{(q_1, w_1), \ldots, (q_n, w_n)\}$, w_i remainder weight at state q_i.
  2. Weight of a transition in the result: $\oplus$-sum of the original transitions pre-$\otimes$-multiplied by remainders.
- Conditions
  - Semiring: weakly left divisible semirings.
  - M is determinizable $\equiv$ the determinization algorithm applies to M.
  - All unweighted automata are determinizable.
  - All acyclic machines are determinizable.

– Not all weighted automata or transducers are determinizable.

– Characterization based on the *twins property*.

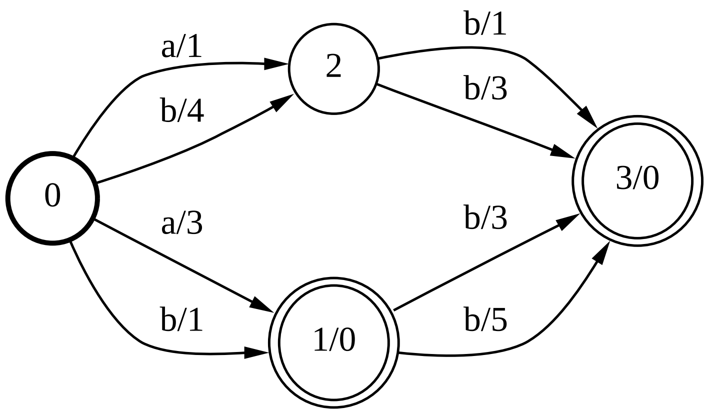
- Complexity and Implementation

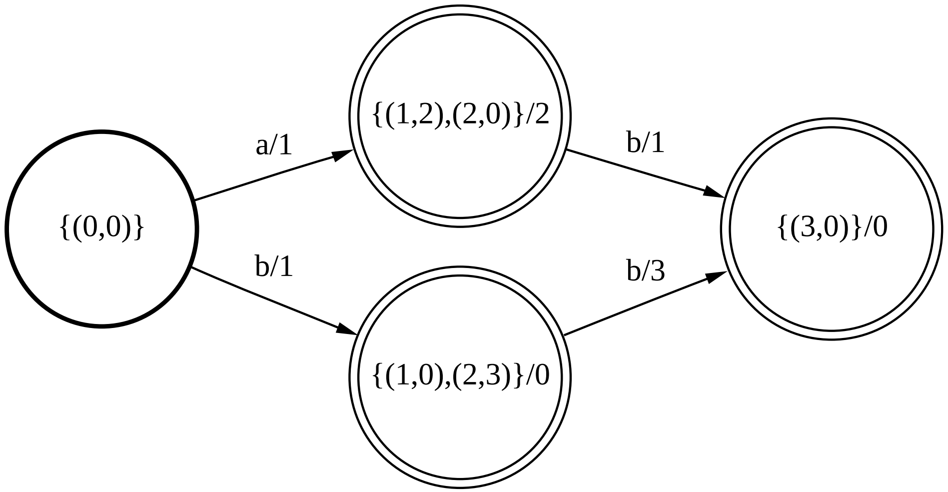
  – Complexity: exponential.

  – Lazy implementation.

Determinization of Weighted Automata – Illustration

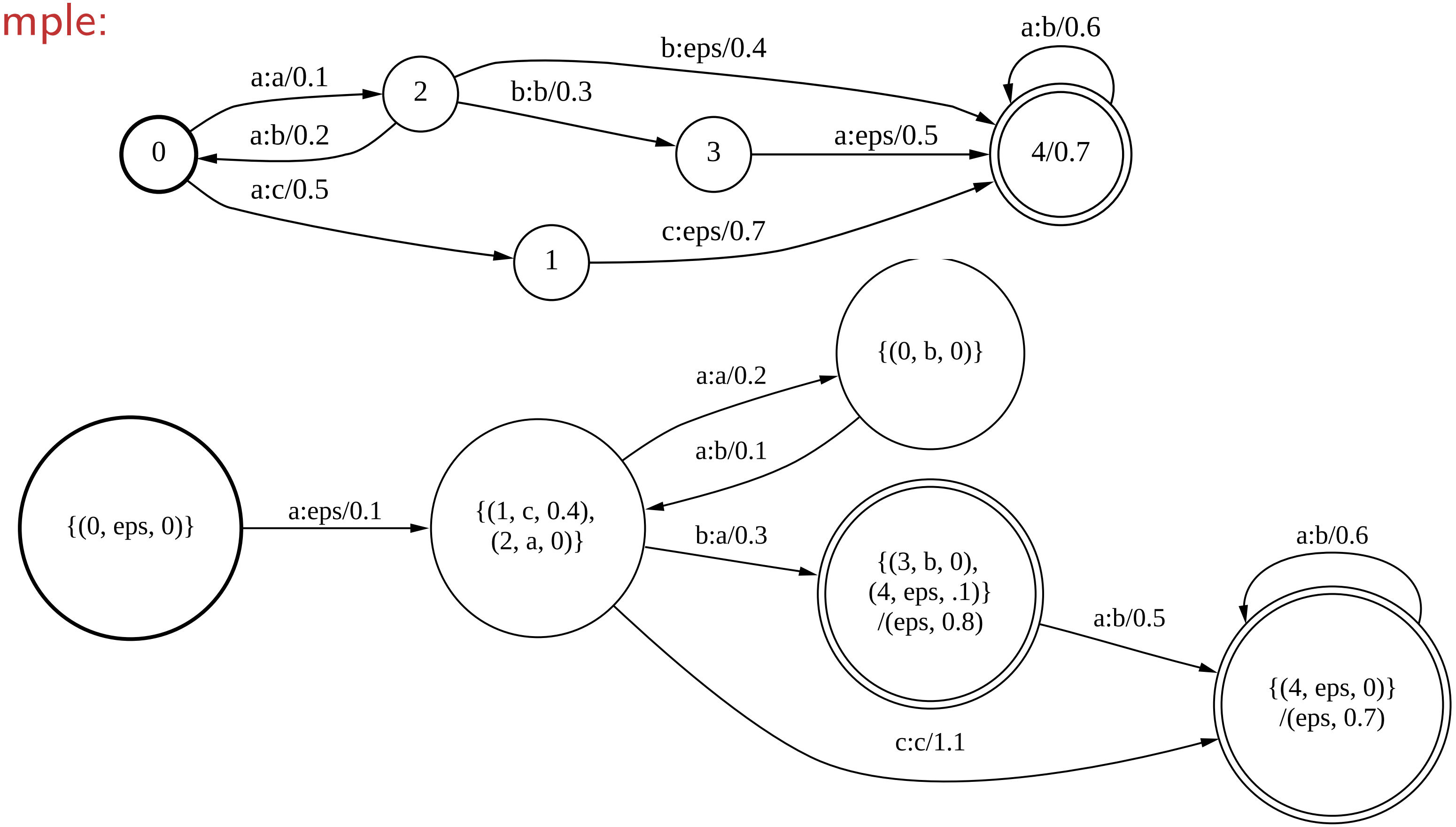
- Definition: Creates an equivalent deterministic weighted automaton
- Example:





Determinization of Weighted Transducers – Illustration

- Definition: Creates an equivalent deterministic weighted transducer
- Example:



Determinization of Weighted Automata – Pseudocode

DETERMINIZATION(A)

$i' \leftarrow \{(i, \lambda(i)) : i \in I\}$

$\lambda'(i') \leftarrow \overline{1}$

$S \leftarrow \{i'\}$

while $S \neq \emptyset$ **do**

    $p' \leftarrow$ HEAD(S)

    DEQUEUE(S)

    for each $x \in i[E[Q[p']]]$ **do**

        $w' \leftarrow \bigoplus \{v \otimes w : (p, v) \in p', (p, x, w, q) \in E\}$

        $q' \leftarrow \{(q, \bigoplus \{w'^{-1} \otimes (v \otimes w) : (p, v) \in p', (p, x, w, q) \in E\}) :$

            $q = n[e], i[e] = x, e \in E[Q[p']]\}$

        $E' \leftarrow E' \cup \{(p', x, w', q')\}$

        if $q' \notin Q'$ **then**

            $Q' \leftarrow Q' \cup \{q'\}$

            if $Q[q'] \cap F \neq \emptyset$ **then**

                $F' \leftarrow F' \cup \{q'\}$

                $\rho'(q') \leftarrow \bigoplus \{v \otimes \rho(q) : (q, v) \in q', q \in F\}$

            ENQUEUE(S, q')

return A'

Pushing – Algorithm

- Definition
  - Input: weighted automaton or transducer M_1.
  - Output: $M_2 \equiv M_1$ such that:
    - the longest common prefix of all outgoing paths is minimal, or
    - the $\oplus$-sum of the weights of all outgoing transitions $= \overline{1}$ modulo the string/weight at the initial state.

- Description (two stages):

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

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

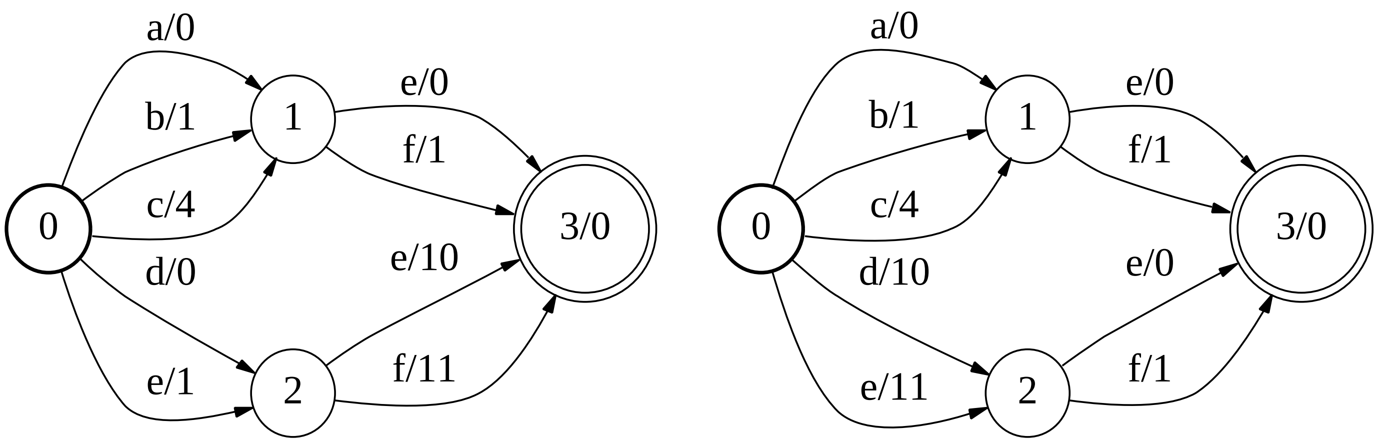
2. Reweighting: for each transition e such that $d[p[e]] \neq \overline{0}$,

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

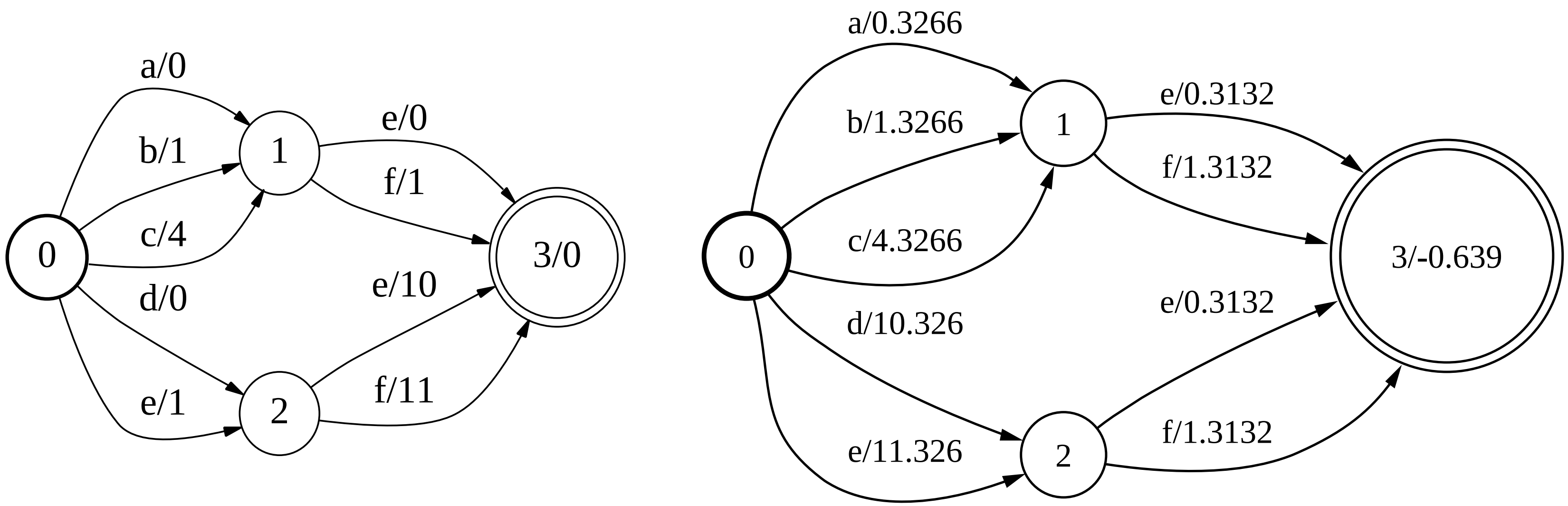
- Conditions (automata case)
  - Weakly divisible semiring.
  - Zero-sum free semiring or zero-sum free machine.
- Complexity
  - Automata case
    * Acyclic case (linear): $O(|Q| + |E|(T_\oplus + T_\otimes))$.
    * General case (tropical semiring): $O(|Q| \log |Q| + |E|)$.
  - Transducer case: $O((|P_{max}| + 1)\,|E|)$.

Weight Pushing – Illustration

- Definition: Creates an equivalent pushed/stochastic machine
- Example:
  - Tropical semiring

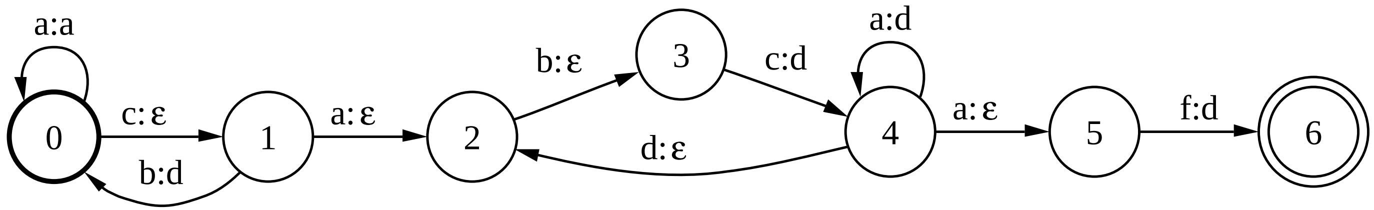


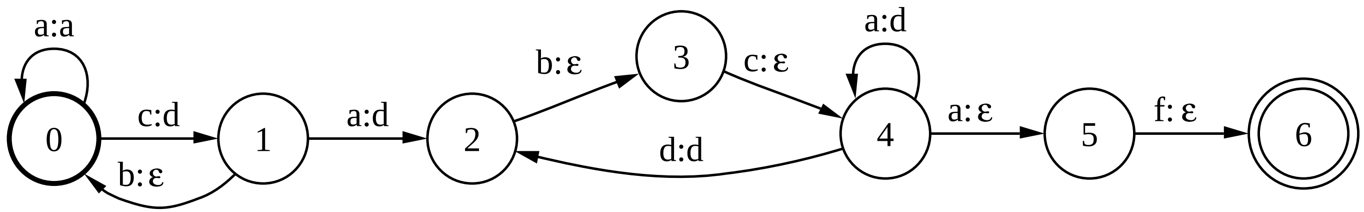
– Log semiring



Label Pushing – Illustration

- Definition: Minimizes at each state the length of the common prefix of all outgoing paths at that state.
- Example:



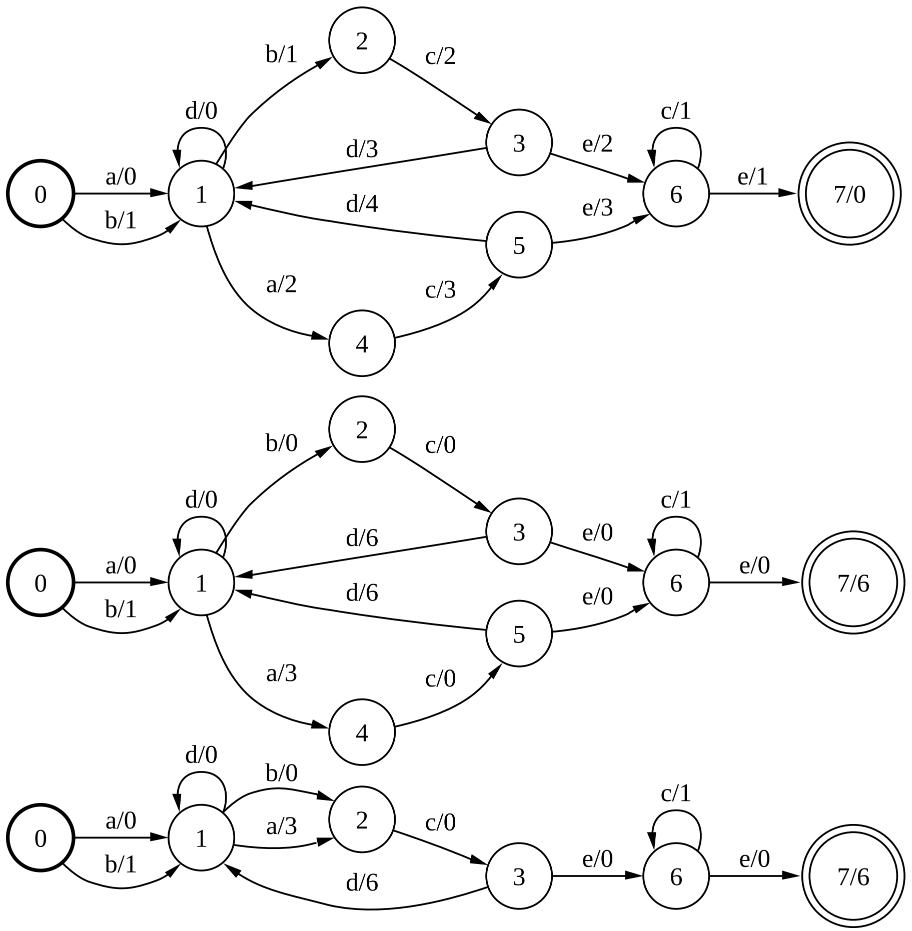


Minimization – Algorithm

- Definition
  - Input: deterministic weighted automaton or transducer M_1.
  - Output: deterministic $M_2 \equiv M_1$ with minimal number of states and transitions.
- Description: two stages
  1. Canonical representation: use pushing or other algorithm to standardize input automata.
  2. 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: $O(S + |Q| + |E| \, (|P_{max}| + 1))$.
    * General case: $O(S + |Q| + |E| \, (\log |Q| + |P_{max}|))$.

Minimization – Illustration

- Definition: Computes a minimal equivalent deterministic machine
- Example:

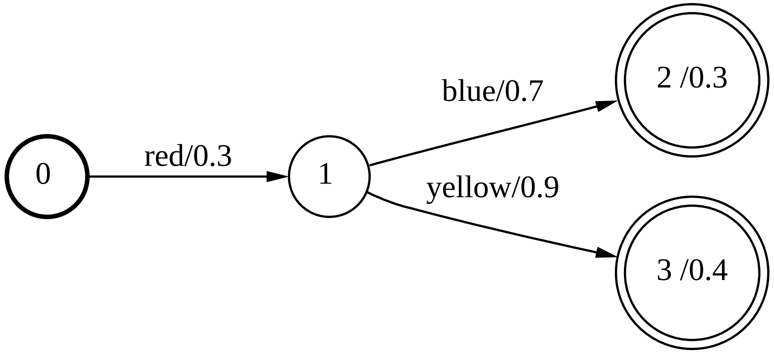


Equivalence – Algorithm

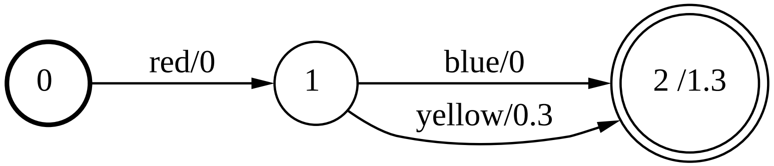
- Definition
  - Input: deterministic weighted automata A_1 and A_2.
  - Output: TRUE if $A_2 \equiv A_1$, FALSE otherwise.
- Description: two stages
  1. Canonical representation: use pushing or other algorithm to standardize input automata.
  2. Test: encode pairs (label, weight) as labels and use classical algorithm for testing the equivalence of unweighted automata.
- Complexity
  - First stage: $O((|E_1| + |E_2|) + (|Q_1| + |Q_2|) \log(|Q_1| + |Q_2|))$ if using pushing in the tropical semiring.
  - Second stage (quasi-linear): $O(m\, \alpha(m, n))$ where $m = |E_1| + |E_2|$ and $n = |Q_1| + |Q_2|$, and α is the *inverse of Ackermann's function*.

Equivalence – Illustration

- Definition: $A_1 \equiv A_2$ iff $[\![A_1]\!](x) = [\![A_2]\!](x)$ for all x
- Graphical Representation:



$$\equiv ?$$



Single-Source Shortest-Distance Algorithms – Algorithm

- Generic single-source shortest-distance algorithm
  - Definition: for each state q,

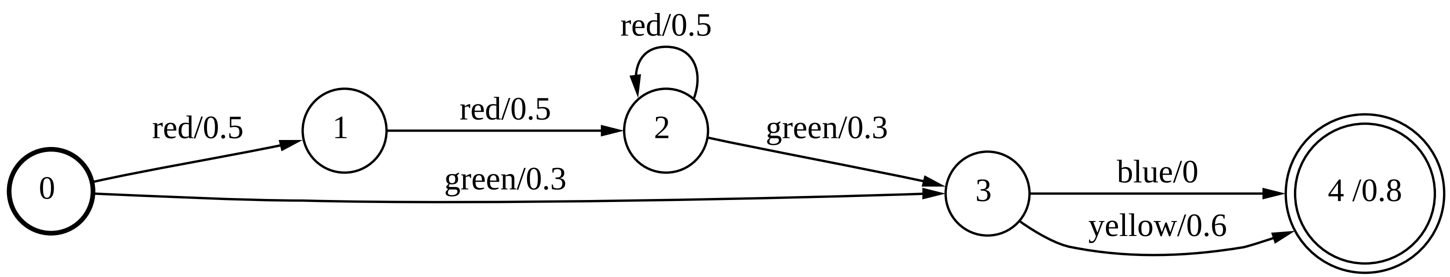
$$d[q] = \bigoplus_{\pi \in P(q,F)} w[\pi]$$

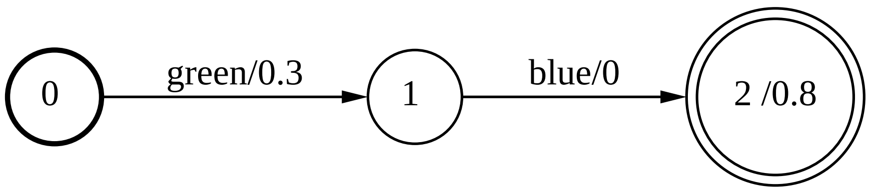
  - Works with any queue discipline and any semiring k-closed for the graph.
  - Coincides with classical algorithms in the specific case of the tropical semiring and the specific queue disciplines: shortest-first (Dijkstra), FIFO (Bellman-Ford), or topological sort order (Lawler).

- N-shortest paths algorithm
  - General N-shortest paths algorithm augmented with the computation of the potentials.
  - On-the-fly weighted determinization for n-shortest strings.

N-Shortest Paths – Illustration

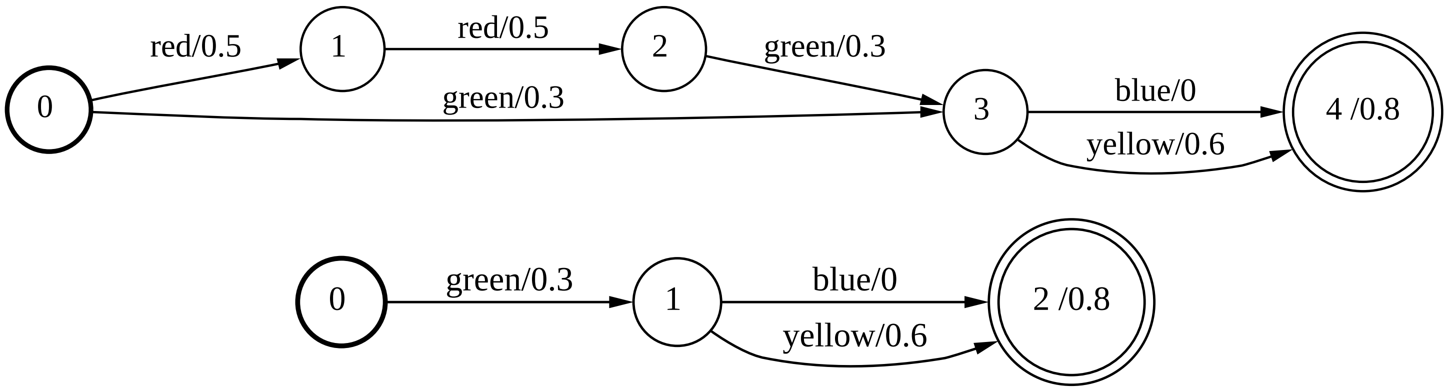
- Definition: Computes the N-shortest paths in the input machine
- Condition: Semiring needs to have the path property: $a \oplus b \in \{a, b\}$ (e.g. tropical semiring)
- Example:





Pruning – Illustration

- Definition: Removes any paths which weight is more than the shortest-distance $\otimes$-multiply by a specified threshold
- Condition: Semiring needs to be commutative and have the path property: $a \oplus b \in \{a, b\}$ (e.g. tropical semiring)
- Example:



String Algorithms – Overview

- How to implement some fundamental string algorithms using the operations previously described:
  - Counting patterns (e.g. n-grams) in automata
  - Pattern matching using automata
  - Compiling regular expression into automata
- Benefits: generality, efficiency and flexibility

Counting from weighted automata

- **Expected count of x in A:**
  Let A be a weighted automaton over the probability semiring,

$$c(x) = \sum_{u \in \Sigma^*} |u|_x \, [\![A]\!](u)$$

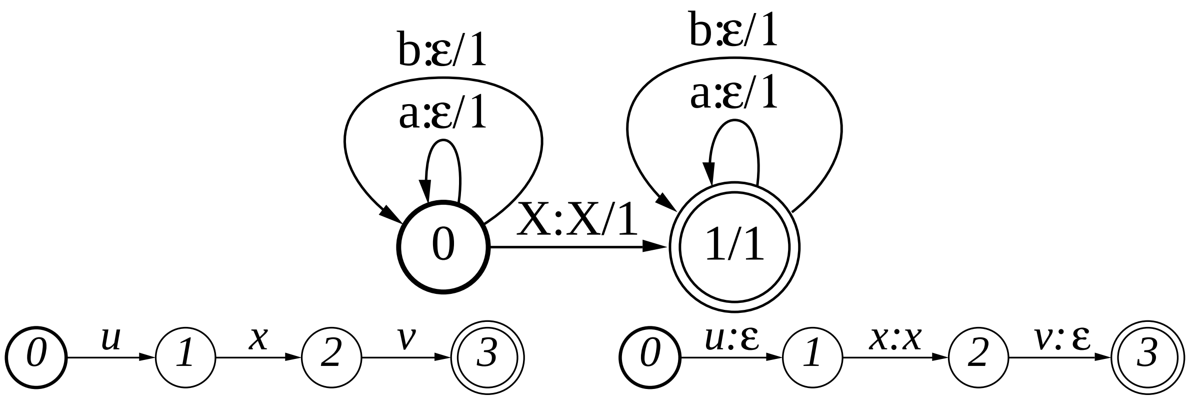
  where:

  - $|u|_x$: number of occurrences of x in u
  - $[\![A]\!](u)$: weight associated to u by A
    $\rightarrow$ $\Pr(u)$ if A is pushed

- **Condition:**
  The weight of any cycle in A should be less than 1.
  This is the case if A represents a probability distribution.

Counting by composition with a transducer

- Counting transducer T for set of sequences X with $\Sigma = \{a, b\}$:



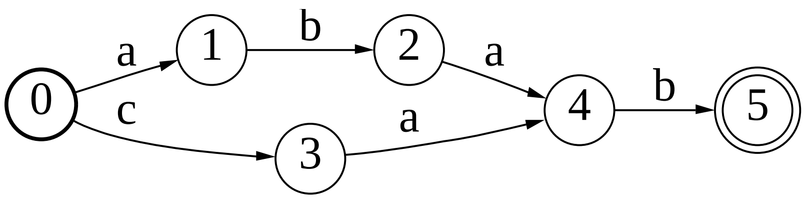
in A in $A \circ T$

To each successful path π in A and each occurence of x along π
$\rightarrow$ corresponds a successful path with output x in $A \circ T$
$\rightarrow$ $c(x)$ is the sum of the weight of all the successful path with output x in $A \circ T$

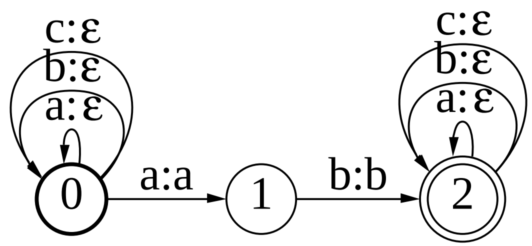
- Theorem:
Let Π_2 denote projection onto output labels. For all $x \in X$,

$$c(x) = [\![\Pi_2(A \circ T)]\!](x)$$

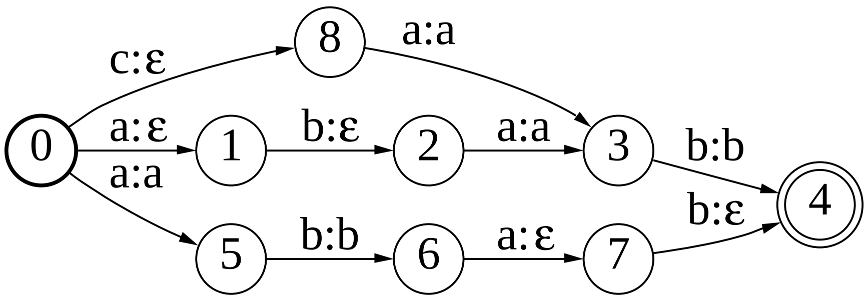
Counting with transducers – Example



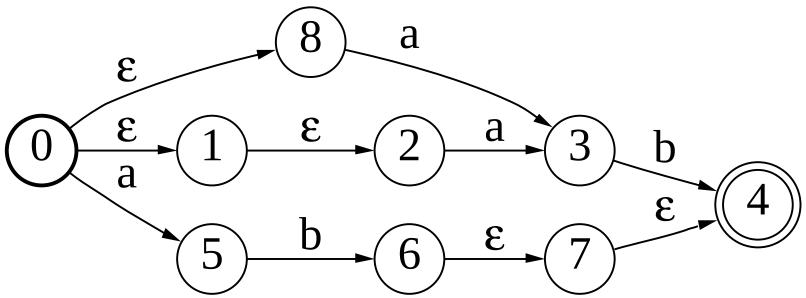
Automaton A



Counting transducer T



$A \circ T$



$\Pi_2(A \circ T)$

$$[\![\Pi_2(A \circ T)]\!](ab) = 3 = c(ab)$$

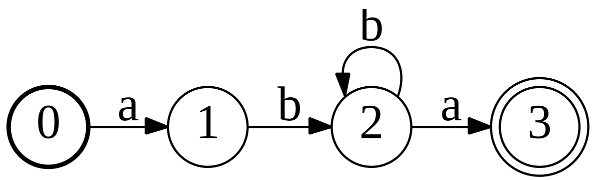
Local Grammar – Algorithm

[Mohri, 94]

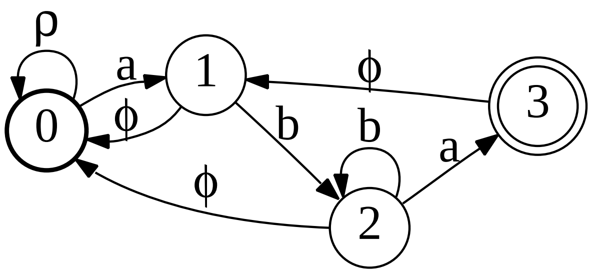
- Definition
  - Input: a deterministic finite automaton A
  - Output: a compact representation of $\det(\Sigma^* A)$
- Description
  - A generalization of [Aho-Corasick, 75]
  - Failure transitions: labeled by ϕ, non-consuming, traversed when no transition with required label is present
  - Default transitions: labeled by ρ, consuming, traversed when no transition with required label is present, only present at the initial state

Local Grammar – Illustration

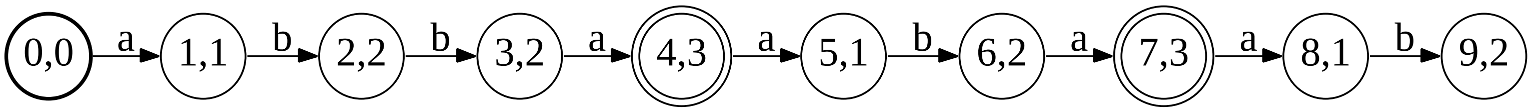
- Pattern matching: find all occurences of pattern A in text T
  $T = abbaabaab$, $A = ab^+a$



A



$\det(\Sigma^* A)$



$T \cap \det(\Sigma^* A)$

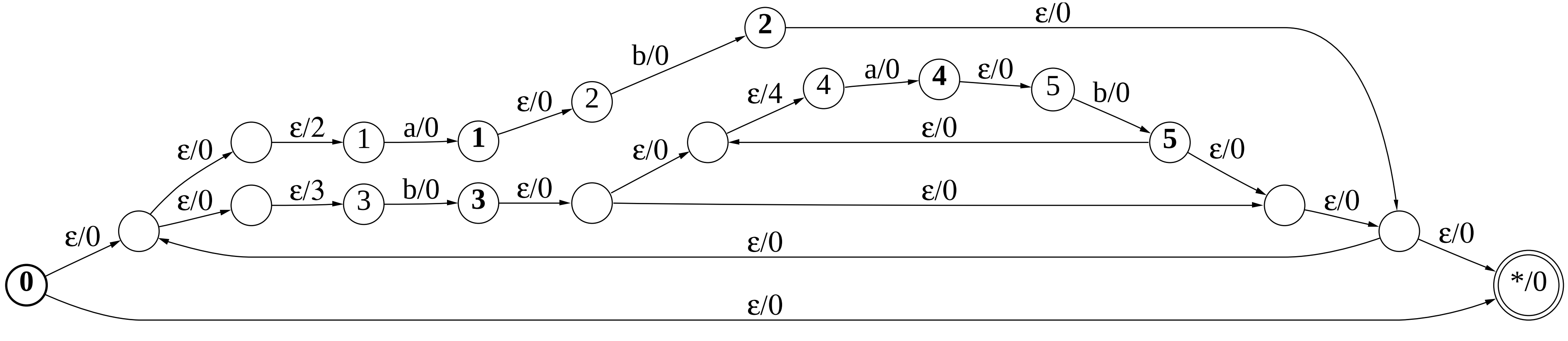
- Complexity: search time linear in $|T|$

Regular Expression Compilation – Algorithms

- Definition
  - Input: a (weighted) regular expression α
  - Output: a (weighted) automata representing α
- Description: Thompson construction
  1. Build a sparse tree for α
  2. Walk the tree bottom-up and apply the relevant rational operation at each node
- Complexity and implementation
  - Linear in the length of α
  - Admits lazy implementation
- Other constructions (Glushkov, Antimirov, Follow) can be obtained from Thompson using epsilon-removal and minimization

Regular Expression Compilation – Thompson

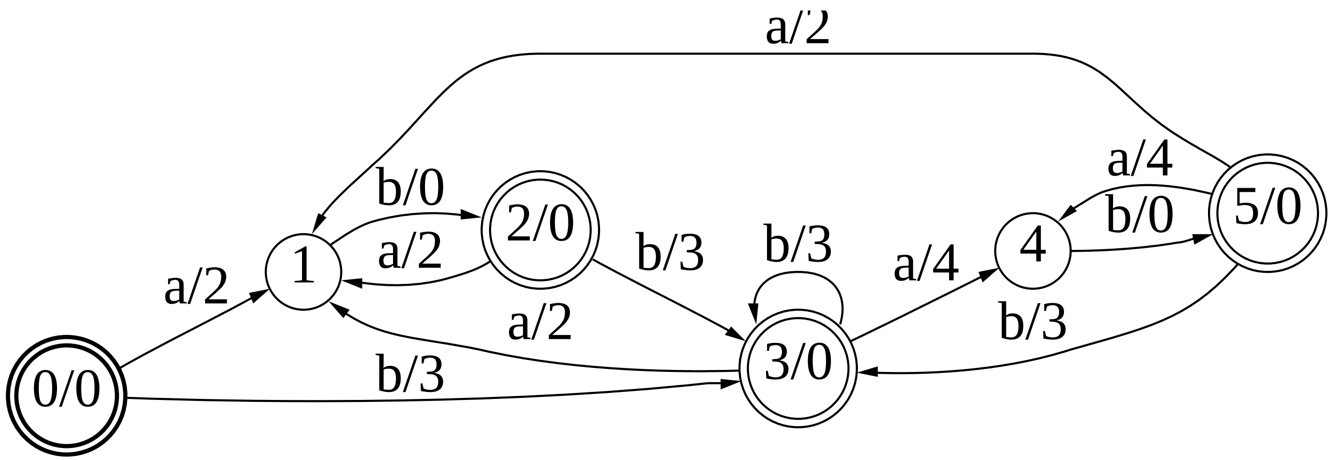
- Regular expression: $\alpha = (2ab + 3b(4ab)^*)^*$
- Thompson automaton:



$A_T(\alpha)$

Regular Expression Compilation – Glushkov

- Regular expression: $\alpha = (2ab + 3b(4ab)^*)^*$
- Glushkov automaton:



$$A_G(\alpha) = \text{rmeps}(A_T(\alpha))$$