Software Libraries

  

- **OpenFst Library**: Open-source Finite-state transducer Library (Allauzen et al., 2007).
  
  http://www.openfst.org
Software Libraries


  http://www.research.att.com/~fsmtools/dcd
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 -i A.sym < A.txt > A.fsm`
  - `fsmcompile -s log -i A.sym -o A.sym -t < T.txt > T.fsm`

- **Printing**
  - `fsmprint -i A.sym < A.fsm > A.txt`
  - `fsmprint -i A.sym -o A.sym < T.fsm | dot -Tps > T.ps`

- **Drawing**
  - `fsmdraw -i A.sym < A.fsm | dot -Tps > A.ps`
  - `fsmdraw -i A.sym -o A.sym < T.fsm | dot -Tps > T.ps`
A **semiring** \((\mathbb{K}, \oplus, \otimes, \bar{0}, \bar{1})\) 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

<table>
<thead>
<tr>
<th>SEMIRING</th>
<th>SET</th>
<th>⊕</th>
<th>⊗</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>{0, 1}</td>
<td>∨</td>
<td>∧</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Probability</td>
<td>(\mathbb{R}_+)</td>
<td>+</td>
<td>×</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Log</td>
<td>(\mathbb{R} \cup {\infty, +\infty})</td>
<td>(\oplus_{\text{log}})</td>
<td>+</td>
<td>+∞</td>
<td>0</td>
</tr>
<tr>
<td>Tropical</td>
<td>(\mathbb{R} \cup {\infty, +\infty})</td>
<td>min</td>
<td>+</td>
<td>+∞</td>
<td>0</td>
</tr>
</tbody>
</table>

with \(\oplus_{\text{log}}\) defined by: \(x \oplus_{\text{log}} y = -\log(e^{-x} + e^{-y})\).
Automata/Acceptors

Graphical Representation \((A.ps)\)

![Graphical Representation Diagram]

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** \( (T.ps) \)
  \[
  \begin{array}{c}
  0 \quad \text{green:blue/0.3} \quad 1 \\
  & \quad 1 \quad \text{blue:green/0} \quad 2 /0.8
  \end{array}
  \]
  - red:yellow/0.5

- **Transducer file** \( (T.txt) \)
  
  \[
  \begin{array}{cccc}
  0 & 0 & \text{red} & \text{yellow} \quad 0.5 \\
  0 & 1 & \text{green} & \text{blue} \quad 0.3 \\
  1 & 2 & \text{blue} & \text{green} \\
  1 & 2 & \text{yellow} & \text{red} \quad 0.6 \\
  2 & & & 0.8
  \end{array}
  \]

- **Symbols file** \( (T.syms) \)
  
  \[
  \begin{array}{cccc}
  \text{red} & 1 \\
  \text{green} & 2 \\
  \text{blue} & 3 \\
  \text{yellow} & 4
  \end{array}
  \]
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 $y$. 

[Diagram showing a path $\pi$ with input label $i[\pi]$, output label $o[\pi]$, previous state or source state $p[\pi]$, and next state or destination state $n[\pi]$]
General Definitions

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

$$[[A]](x) = \text{Sum of the weights of all successful paths labeled with } x$$

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

\[[T]\](x, y) = \text{Sum of the weights of all successful paths with input } x \text{ 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**

\[
[T_1 \otimes T_2](x, y) = \bigoplus_{\begin{subarray}{c} x=x_1x_2 \\ y=y_1y_2 \end{subarray}} [T_1](x_1, y_1) \otimes [T_2](x_2, y_2).
\]

- **Closure**

\[
[T^*](x, y) = \bigoplus_{n=0}^{\infty} [T]^n(x, y)
\]
• **Conditions** (on the closure operation): condition on $T$: e.g., weight of $\varepsilon$-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:
    $$O((|E_1| + |Q_1|) + (|E_2| + |Q_2|)) \text{ or } O(|Q| + |E|)$$
  - lazy implementation.
Program: \texttt{fsmunion A.fsm B.fsm >C.fsm}

Graphical representation:
Program: \texttt{fsmconcat A.fsm B.fsm >C.fsm}

Graphical representation:
Closure - Illustration

- **Program:** `fsmclosure B.fsm > C.fsm`

- **Graphical representation:**

```
A.fsa
0
red/0.5
1 green/0.3
2 /0.8 blue/0
yellow/0.6
B.fsa
0
1 /0 green/0.4
2 /0.3 blue/1.2
C.fsa
0
red/0.5
1 green/0.3
2 /0.8 blue/0
yellow/0.6
3 4 /0 green/0.4
5 /0.3 blue/1.2
6 eps/0
eps/0
```
This Lecture

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

- **Reversal**
  \[
  \tilde{T}(x, y) = [T](\tilde{x}, \tilde{y})
  \]

- **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:
Program: fsminvert A.fsm > C.fsm

Graphical representation:

```
A.fst

0 → 1 → 2
red:bird/0.5
green:pig/0.3
blue:cat/0
yellow:dog/0.6

C.fst

0 → 1 → 2
bird:red/0.5
pig:green/0.3
cat:blue/0
dog:yellow/0.6
```
**Projection - Illustration**

- **Program:** `fsmproject -l T.fsm > A.fsm`
- **Graphical representation:**

```plaintext
A.fst
0
red:bird/0.5
1
green:pig/0.3
2 /0.8
blue:cat/0
yellow:dog/0.6

C.fst
0
bird:red/0.5
1
pig:green/0.3
2 /0.8
cat:blue/0
dog:yellow/0.6
```

```plaintext
T.fst
0
red:bird/0.5
1
green:pig/0.3
2 /0.8
blue:cat/0
yellow:dog/0.6

A.fsa
0
red/0.5
1
green/0.3
2 /0.8
blue/0
yellow/0.6
```
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** \(((\mathbb{K}, \oplus, \otimes, \bar{0}, \bar{1}) \text{ commutative})\)
  \[
  [T_1 \circ T_2](x, y) = \bigoplus_z [T_1](x, z) \otimes [T_2](z, y)
  \]

- **Intersection** \(((\mathbb{K}, \oplus, \otimes, \bar{0}, \bar{1}) \text{ commutative})\)
  \[
  [A_1 \cap A_2](x) = [A_1](x) \otimes [A_2](x)
  \]

- **Difference** \((A_2 \text{ unweighted and deterministic})\)
  \[
  [A_1 - A_2](x) = [A_1 \cap \overline{A_2}](x)
  \]
• **Complexity and implementation:**

  • **quadratic complexity:**

    \[ O\left((|E_1| + |Q_1|) (|E_2| + |Q_2|)\right) \]

  • **path multiplicity in presence of \( \epsilon \)-transitions: \( \epsilon \)-filter;**

  • **lazy implementation.**
Program: fsmcompose A.fsm B.fsm >C.fsm

Graphical representation:

Composition - Illustration
Multiplicity and $\varepsilon$-Transitions - Problem

```
0,0  a:d  (x:x)  1,1  $\varepsilon$:e  (e1:e1)  1,2
     \b: e  (e2:e2)  \b: e  (e2:e1)  \b: e  (e2:e2)
     \c: e  (e2:e2)  \c: e  (e2:e2)  \c: e  (e2:e2)
  2,1  \varepsilon$:e  (e1:e1)  2,2
       \varepsilon$:e  (e1:e1)  \varepsilon$:e  (e1:e1)
    \varepsilon$:e  (e1:e1)  \varepsilon$:e  (e1:e1)
  3,1  \varepsilon$:e  (e1:e1)  3,2
       \varepsilon$:e  (e1:e1)  \varepsilon$:e  (e1:e1)
    d:a  (x:x)  \varepsilon$:e  (e1:e1)
  4,3
```

```
0  a:a  1  b: e  2  c: e  3  d:d  4
    \varepsilon$:e  (e1:e1)  \varepsilon$:e  (e1:e1)
2  a:d  1  \varepsilon$:e  2  d:a  3
```

Mehryar Mohri - Speech Recognition

page  Courant Institute, NYU
Solution - Filter $F$ for Composition

Replace $T_1 \circ T_2$ with $\tilde{T}_1 \circ F \circ \tilde{T}_2$. 
Intersection - Illustration

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

```
A.fsa
0  red/0.5
1  green/0.3
2  blue/0.8

B.fsa
0  red/0.2
1  blue/0.6
2  green/0.4

C.fsa
0  red/0.7
1  green/0.7
2  blue/0.6
3  yellow/0.8
4  yellow/1.3
```
**Program**: fsmdifference A.fsm B.fsm > C.fsm

**Graphical representation:**

A.fsa

- 0
- red/0.5
- green/0.3

- 1
- blue/0
- yellow/0.6

- 2
- red/0.8

B.fsa

- 0
- red
- blue
- green
- yellow

- 1
- red
- blue

- 2
- yellow

C.fsa

- 0
- red/0.5
- green/0.3

- 1
- red/0.5
- green/0.3

- 2
- red/0.8

- 3
- blue/0
- yellow/0.6

- 4
- red/0.8
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.
- **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, e.g., not all weighted automata or transducers can be determinized using the determinization algorithm.
Program: `fsmconnect A.fsm > C.fsm`

Graphical representation:

[Diagram of a state machine with transitions labeled as follows:
- State 0 to State 1: Green 0.3
- State 1 to State 4: Green 0.2
- State 1 to State 2: Yellow 0.6
- State 1 to State 5: Red 0
- State 2 to State 5: Blue 0
- State 3 to State 4: Red 0.5
- State 3 to State 1: Red 0.5
- State 0 to State 1: Green 0.3
- State 0 to State 2: Yellow 0.6
- State 0 to State 5: Red 0.5]
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**:
\( \epsilon\)-Removal - Algorithm

(\text{MM, 2001})

\begin{itemize}
  \item **Definition:**
    \begin{itemize}
      \item Input: weighted transducer \( T_1 \).
      \item Output: equivalent WFST \( T_2 \) with no \( \epsilon \)-transition.
    \end{itemize}
  \item **Description:**
    \begin{itemize}
      \item Computation of \( \epsilon \)-closures.
      \item Removal of \( \epsilon \)s.
    \end{itemize}
  \item **Complexity:**
    \begin{itemize}
      \item Acyclic \( T_\epsilon \): \( O(\|Q\|^2 + \|Q\|\|E\|(T_\oplus + T_\otimes)) \).
      \item General case (tropical semiring):
        \( O(\|Q\|\|E\| + \|Q\|^2 \log \|Q\|) \).
    \end{itemize}
\end{itemize}
Computation of $\epsilon$-closures

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

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

  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 $\epsilon$-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:**

```
        2
      /|\  
     /  |  
    a/1 b/1
   /    /    
 0 -> 2 -> 3/0
      |  
   b/4 b/3
   |    |
   b/1  b/1  b/3  b/3
  0 -> 1/0 -> 3/0
        |
   a/3  

{(0,0)}

{(1,2),(2,0)}/2

{(1,0),(2,3)}/0

{(3,0)}/0

{(1,0),(2,3)}/0

{(3,0)}/0
```

Mehryar Mohri - Speech Recognition

Courant Institute, NYU
Program: `fsmdeterminize T.fsm > D.fsm`

Graphical representation:

```
Program: fsmdeterminize T.fsm > D.fsm

Graphical representation:
```

```
Determinization of Weighted Transducers – Illustration
```

```
• Program: fsmdeterminize T.fsm > D.fsm

• Graphical representation:

```
```
Pushing - Algorithm

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 $\varepsilon$ 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 $d[p[e]] \neq 0$,

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

![Diagram of Weight Pushing Illustration]

```
Program: fsmpush -ic A.fsm > P.fsm

Graphical representation:

• Tropical semiring:

![Graphical Representation Diagram]
```
• Log semiring:
Program: fsmpush -il T.fsm >P.fsm

Graphical representation:

```
0
  a:a
  c: ε
  b: d

1
  a: ε

2
  a: ε
  b: ε
  d: ε

3
  b: ε
  c: d

4
  a: ε
  d: ε
  c: d

5
  a: ε

6
  f: d

0
  a:a
  c: d
  b: ε

1
  a: d

2
  a: d
  b: ε

3
  b: ε
  c: ε

4
  a: d
  d: d

5
  a: ε

6
  f: ε
```
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: \(O(S + |Q| + |E| (|P_{max}| + 1))\).
  
  • general case (tropical semiring):
    \[O(S + |Q| + |E| (\log |Q| + |P_{max}|)).\]
**Minimization - Illustration**

- **Program:** `fsminimize D.fsm > M.fsm`
- **Graphical representation:**

```
Program: fsminimize D.fsm > M.fsm

Graphical representation:
```

![Diagram of a finite-state machine with states and transitions labeled with inputs and outputs.](Diagram.png)
Equivalence - Algorithm

- **Definition:**
  - Input: deterministic weighted automata $A$ and $B$.
  - Output: $\text{TRUE}$ 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**:

```
D.fsa
0 1 red/0.3
2 /0.3 blue/0.7
3 /0.4 yellow/0.9

M.fsa
0 1 red/0
2 /1.3 blue/0
3 /0.3 yellow/0.3
```

---

Mehryar Mohri - Speech Recognition

Courant Institute, NYU
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:**

```
0 -----> 1: red/0.5
  
1 -----> 0: green/0.3

0 -----> 2: red/0.5

2 -----> 3: green/0.3
  
3 -----> 4: blue/0
  
4 -----> 3: yellow/0.6
```

```
0 -----> 1: green/0.3
  
1 -----> 2: blue/0
  
2 -----> 0: blue/0
```

```
0 -----> 1: red/0.5
  
1 -----> 2: green/0.3
  
2 -----> 0: blue/0
```

```
0 -----> 1: red/0.5
  
1 -----> 2: green/0.3
  
2 -----> 0: blue/0
```
Pruning - Illustration

Program: `fsmprune -c 1.0 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., ε-removal, determinization, minimization);
  - search algorithms (e.g., shortest-distance algorithms, n-best paths algorithms, pruning).
References


References


