Speech Recognition
Lecture 2: Finite Automata and Finite-State Transducers

Cyril Allauzen
Google, NYU Courant Institute
allauzen@cs.nyu.edu
Slide Credit: Mehryar Mohri
Preliminaries

- Finite alphabet $\Sigma$, empty string $\epsilon$.
- Set of all strings over $\Sigma$: $\Sigma^*$.
- Length of a string $x \in \Sigma^*$: $|x|$.
- Mirror image or reverse of a string $x = x_1 \cdots x_n$:
  \[ x^R = x_n \cdots x_1. \]
- A language $L$: subset of $\Sigma^*$. 
Rational Operations

Rational operations over languages:

- union: also denoted $L_1 + L_2$,
  \[ L_1 \cup L_2 = \{ x \in \Sigma^* : x \in L_1 \lor x \in L_2 \} \]
- concatenation:
  \[ L_1 \cdot L_2 = \{ x = uv \in \Sigma^* : u \in L_1 \lor v \in L_2 \} \]
- closure:
  \[ L^* = \bigcup_{n=0}^{\infty} L^n, \quad \text{where} \quad L^n = \underbrace{L \cdot \cdots \cdot L}_n \]
Regular or Rational Languages

**Definition**: the class of regular/rational languages over $\Sigma$ is the smallest set $\mathcal{L}$ containing the empty set and closed under the rational operations. i.e.,

- $\emptyset \in \mathcal{L}$
- $\forall x \in \Sigma^*, \{x\} \in \mathcal{L}$
- $\forall L_1, L_2 \in \mathcal{L}, L_1 \cup L_2 \in \mathcal{L}, L_1 \cdot L_2 \in \mathcal{L}, L_1^* \in \mathcal{L}$.

**Examples of regular languages over $\Sigma = \{a, b, c\}$**: 
- $\Sigma^*$, $(a + b)^*c$, $ab^n c$, $(a + (b + c)^*ba)^* cb$. 
Finite Automata

Definition: a finite automaton $A$ over the alphabet $\Sigma$ is 4-tuple $(Q, I, F, E)$ where $Q$ is a finite set of states, $I \subseteq Q$ a set of initial states, $F \subseteq Q$ a set of final states, and $E$ a multiset of transitions which are elements of $Q \times (\Sigma \cup \{\epsilon\}) \times Q$.

- a path $\pi$ in an automaton $A = (Q, I, F, E)$ is an element of $E^*$.

- a path from a state in $I$ to a state in $F$ is called an accepting path. Language $L(A)$ accepted by $A$: set of strings labeling accepting paths.
Finite Automata - Example

```
<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td></td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>b</td>
<td></td>
<td>b</td>
</tr>
</tbody>
</table>
```
Finite Automata - Some Properties

- Trim: any state lies on some accepting path.
- Unambiguous: no two accepting paths have the same label.
- Deterministic: unique initial state, transitions leaving the same state have different labels.
- Complete: at least one outgoing transition labeled with any alphabet element at any state.
- Acyclic: no path with a cycle.
Definition: a finite automaton is normalized if

- it has a unique initial state with no incoming transition.
- it has a unique final state with no outgoing transition.
**Elementary Normalized Automaton**

**Definition:** normalized automaton accepting an element $a \in \Sigma \cup \{\epsilon\}$ constructed as follows.

![Diagram of a normalized automaton with transition a from state 0 to state 1]
Normalized Automata: Union

Construction: the union of two normalized automata is a normalized automaton constructed as follows.
Normalized Automata: Concatenation

**Construction**: the concatenation of two normalized automata is a normalized automaton constructed as follows.

```
\begin{figure}
\centering
\begin{tikzpicture}
  \node (i1) [circle, draw] {$i_1$};
  \node (f1) [circle, draw] at (2,0) {$f_1$};
  \node (i2) [circle, draw] at (4,0) {$i_2$};
  \node (f2) [circle, draw] at (6,0) {$f_2$};
  \node (A1) [rectangle, draw] at (1,1) {$A_1$};
  \node (A2) [rectangle, draw] at (5,1) {$A_2$};

  \draw [->] (i1) -- (f1);
  \draw [->] (f1) -- (i2);
  \draw [->] (i2) -- (f2);
  \draw [->] (i1) -- (A1);
  \draw [->] (A1) -- (f1);
  \draw [->] (A2) -- (i2);
  \draw [->] (f2) -- (A2);
\end{tikzpicture}
\end{figure}
```
**Normalized Automata: Closure**

- **Construction**: the closure of a normalized automaton is a normalized automaton constructed as follows.
Normalized Automata - Properties

Construction properties:

• each rational operation require creating at most two states.
• each state has at most two outgoing transitions.
• the complexity of each operation is linear.
Thompson’s Construction

(Thompson, 1968)

let \( r \) be a regular expression over the alphabet \( \Sigma \). Then, there exists a normalized automaton \( A \) with at most \( 2 |r| \) states representing \( r \).

Proof:

• first, parse regular expression.

• construction of normalized automaton starting from elementary expressions and following operations of the tree.
Normalized automaton for regular expression $ab^* + c$. 
Regular Languages and Finite Automata

(Theorem): A language is regular iff it can be accepted by a finite automaton.

Proof: Let \( A = (Q, I, F, E) \) be a finite automaton.

- for \((i, j, k) \in [1, |Q|] \times [1, |Q|] \times [0, |Q|]\) define
  \[
  X_{ij}^k = \{i \to q_1 \to q_2 \to \ldots \to q_n \to j : n \geq 0, q_i \leq k\}.
  \]
- \(X_{ij}^0\) is regular for all \((i, j)\) since \(E\) is finite.
- by induction \(X_{ij}^k\) for all \((i, j, k)\) since
  \[
  X_{ij}^{k+1} = X_{ij}^k + X_{i,k+1}^k(X_{k+1,j}^k)^* X_{k+1,j}^k.
  \]
- \(L(A) = \bigcup_{i \in I, f \in F} X_{if}^{\left|Q\right|}\) is thus regular.
Regular Languages and Finite Automata

**Proof**: the converse holds by Thompson’s construction.

**Notes**:

- a more general theorem (Schützenberger, 1961) holds for weighted automata.
- not all languages are regular, e.g., \( L = \{a^n b^n : n \in \mathbb{N}\} \) is not regular. Let \( A \) be an automaton. If \( L \subseteq L(A) \), then for large enough \( n \), \( a^n b^n \) corresponds to a path with a cycle: \( a^n b^n = a^p ubq, a^p u^* b^q \subseteq L(A) \), which implies \( L(A) \neq L \).
**Left Syntactic Congruence**

- **Definition:** for any language $L \subseteq \Sigma^*$, the left syntactic congruence is the equivalence relation defined by

$$u \equiv_L v \iff u^{-1}L = v^{-1}L,$$

where for any $u \in \Sigma^*$, $u^{-1}L$ is defined by

$$u^{-1}L = \{w \colon uw \in L\}.$$

- $u^{-1}L$ is sometimes called the partial derivative of $L$ with respect to $u$ and denoted $\frac{\partial L}{\partial u}$. 
Regular Languages - Characterization

- **Theorem**: a language $L$ is regular iff the set of $u^{-1}L$ is finite ($\equiv_L$ has a finite index).

- **Proof**: let $A = (Q, I, F, E)$ be a trim deterministic automaton accepting $L$ (existence seen later).
  
  - let $\delta$ the partial transition function. Then,
    
    $$u \sim_R v \iff \delta(i, u) = \delta(i, v).$$
    
    also defines an eq. relation with index $|Q|$.

  - since $\delta(i, u) = \delta(i, v) \Rightarrow u^{-1}L = v^{-1}L$, the index of $\equiv_L$ is at most $|Q|$, thus finite.
Regular Languages - Characterization

Proof: conversely, if the set of $u^{-1}L$ is finite, then the automaton $A = (Q, I, F, E)$ defined by

- $Q = \{u^{-1}L: u \in \Sigma^*\}$;
- $i = \epsilon^{-1}L = L$, $I = \{i\}$;
- $F = \{u^{-1}L: u \in L\}$;
- $E = \{(u^{-1}L, a, (ua)^{-1}L): u \in \Sigma^*\}$; is well defined since $u^{-1}L = v^{-1}L \Rightarrow (ua)^{-1}L = (va)^{-1}L$ and accepts exactly $L$. 
Illustration

Minimal deterministic automaton for \((a + b)^* ab\):
ε-Removal

Any finite automaton has an equivalent automaton with no ε-transitions.

For any state $q \in Q$, let $\epsilon[q]$ denote the set of states reached from $q$ by paths labeled with $\epsilon$.

Define $A' = (Q', I', F', E')$ as

- $Q' = \{ \epsilon[q] : q \in Q \}$,
- $I' = \bigcup_{q \in I} \epsilon[q]$,
- $F' = \{ \epsilon[q] : \epsilon[q] \cap F \neq \emptyset \}$,
- $E' = \{ (\epsilon[p], a, \epsilon[q]) : \exists (p', a, q) \in E, p' \in \epsilon[p] \}$.
\( \epsilon \)-Removal - Illustration

\[
\begin{array}{cccc}
\epsilon & a & b & \epsilon \\
0 & 1 & 2 & 3 \\
\{0\} & \{0, 1\} & \{0, 2\} & \{0, 1, 3\}
\end{array}
\]
Determinization

- Any automaton $A = (Q, I, F, E)$ without epsilon transitions has an equivalent deterministic automaton.

- Subset construction: $A' = (Q', I', F', E')$ with
  - $Q' \subseteq 2^Q$.
  - $I' = \{I\}$.
  - $F' = \{s \in Q': s \cap F \neq \emptyset\}$.
  - $E' = \{(s, a, s'): s' = \{q': (q, a, q') \in E, q \in s\}\}$.
Determinization - Illustration

Left:
- States: 0, 1, 2
- Transitions: a from 0 to 1, b from 0 to 2, b from 1 to 2, a from 2 to 1, b from 2 to 0

Right:
- States: 0, 1, 2
- Transitions: a from 0 to 1, b from 0 to 2, a from 1 to 2, b from 1 to 0, a from 2 to 1, b from 2 to 0
Completion

- Any deterministic automaton has an equivalent complete deterministic automaton.

- Algorithm illustration:
Complementation

Let $A = (Q, I, F, E)$ be a deterministic automaton, then there exists a deterministic automaton accepting $\overline{L(A)}$.

By previous property, we can assume $A$ complete. The automaton $B = (\Sigma, Q, I, Q - F, E)$ obtained from $A$ by making non-final states final and final states non-final exactly accepts $\overline{L(A)}$. 
Complementation - Illustration

```
0  a  1  a  3
b   b   a

4  b  2  a  3
b   a

0  a  1  a  3
b   b   a

4  b  2  a  3
b   a
```
Regular Languages - Properties

**Theorem**: regular languages are closed under rational operations, intersection, complementation, reversal, morphism, inverse morphism, and quotient with any set.

**Proof**: closure under rational operations holds by definition.

- intersection: use De Morgan’s law.
- complementation: use algorithm.
- others: algorithms and equivalence relation.
Rational Relations

**Definition**: closure under rational operations of the monoid $\Sigma^* \times \Delta^*$, where $\Sigma$ and $\Delta$ are finite alphabets, denoted by $\text{Rat}(\Sigma^* \times \Delta^*)$.

- **Examples**: $(a, b)^*$, $(a, b)^*(bb, a) + (b, a)$.
Rational Relations - Characterization

(Chen et al., 1968)

**Theorem:** \( R \in \text{Rat}(\Sigma^* \times \Delta^*) \) is a rational relation iff there exists a regular language \( L \subseteq (\Sigma \cup \Delta)^* \) such that

\[
R = \{(\pi_\Sigma(x), \pi_\Delta(x)) : x \in L\}
\]

where \( \pi_\Sigma \) is the projection of \( (\Sigma \cup \Delta)^* \) over \( \Sigma^* \)
and \( \pi_\Delta \) the projection over \( \Delta^* \).

**Proof:** use surjective morphism

\[
\pi : (\Sigma \cup \Delta)^* \rightarrow (\Sigma^* \times \Delta^*)
\]

\[
x \rightarrow (\pi_\Sigma(x), \pi_\Delta(x)).
\]
Transductions

Definition: a function $\tau : \Sigma^* \rightarrow 2^{\Delta^*}$ is called a transduction from $\Sigma^*$ to $\Delta^*$.

- relation associate to $\tau$:
  \[
  R(\tau) = \{(x, y) \in \Sigma^* \times \Delta^* : y \in \tau(x)\}.
  \]

- transduction associated to a relation:
  \[
  \forall x \in \Sigma^*, \tau(x) = \{y : (x, y) \in R\}.
  \]

- rational transductions: transductions with rational relations.
Finite-State Transducers

Definition: a finite-state transducer $T$ over the alphabets $\Sigma$ and $\Delta$ is 4-tuple where $Q$ is a finite set of states, $I \subseteq Q$ a set of initial states, $F \subseteq Q$ a set of final states, and $E$ a multiset of transitions which are elements of $Q \times (\Sigma \cup \{\epsilon\}) \times (\Delta \cup \{\epsilon\}) \times Q$.

- $T$ defines a relation via the pair of input and output labels of its accepting paths,

$$R(T) = \{ (x, y) \in \Sigma^* \times \Delta^*: I \xrightarrow{x:y} F \}.$$
Rational Relations and Transducers

- **Theorem**: a transduction is rational iff it can be realized by a finite-state transducer.

- **Proof**: Nivat’s theorem combined with Kleene’s theorem, and construction of a normalized transducer from a finite-state transducer.
References


• Nivat, Maurice. 1968. Transductions des langages de Chomsky. Annales 18, Institut Fourier.

• Schützenberger, Marcel~Paul. 1961. On the definition of a family of automata. *Information and Control*, 4