Introduction to Finite-State Automata

John McDonough

Language Technologies Institute,
Machine Learning for Signal Processing Group,
Carnegie Mellon University

March 26, 2012
In this lecture, we present the basic definitions associated with conventional *finite-state automata* (FSA).

We also investigate various aspects related to determinism, including $\epsilon$-transitions.

In the second part of the lecture, we discuss semirings, which will enable important generalizations of the definition of path labels.

This discussion will lead naturally to our discussion of shortest path algorithms in the next lecture.

**Coverage:** Hopcroft and Ullman (1979), Sections 2.3 and 2.4; Aho *et al.* (1974), Section 5.6.
Let us now define the *spherical harmonic* of order $n$ and degree $m$ as

$$Y_n^m(\theta, \phi) \triangleq \sqrt{\frac{(2n + 1) (n - m)!}{4\pi (n + m)!}} P_n^m(\cos \theta) e^{im\phi},$$

where $P_n^m$ is the *associated Legendre function*.

The *addition theorem for spherical harmonics* states

$$P_n(\cos \gamma) = \frac{4\pi}{2n + 1} \sum_{m=-n}^{n} Y_n^m(\theta_s, \phi_s) \bar{Y}_n^m(\theta, \phi),$$

where $\bar{Y}$ denotes the complex conjugate of $Y$. 
Orthonormality

**Figure:** The spherical harmonics $Y_0, Y_1, Y_2$ and $Y_3$.

The spherical harmonics possess the all important property of *orthonormality*, which implies

$$
\delta_{n,n'} \delta_{m,m'} = \int_{\Omega} Y_n^m(\theta, \phi) \bar{Y}_{n'}^{m'}(\theta, \phi) \, d\Omega
$$

where $\Omega$ denotes the surface of a sphere.
Three-Dimensional Beampatterns

Radially Symmetric MVDR  Asymmetric MVDR
The Man-Wolf-Goat-Cabbage Problem Revisited

- A solution to the man-wolf-goat-cabbage problem corresponds to a path through the transition diagram from the start state MWGC-; to the end state ;-MWGC.

- It is clear from the transition diagram that there are two equally short solutions to the problem.

- There is an infinitude of possible solutions, all but two of which involve useless cycles.

- As with all finite-state automata, there is a unique start state.

- This particular FSA also has a single valid end or accepting state, which is not generally the case.
Formally define a finite-state automaton (FSA) as the 5-tuple \((Q, \Sigma, \delta, i, F)\) where

- \(Q\) is a finite set of states,
- \(\Sigma\) is a finite alphabet,
- \(i \in Q\) is the initial state,
- \(F \subset Q\) is the set of final states,
- \(\delta\) is the transition function mapping \(Q \times \Sigma\) to \(Q\), which implies \(\delta(q, a)\) is a state for each state \(q\) and input \(a\) provided that \(a\) is accepted when in state \(q\).
Extending \( \delta \) to Strings

- To handle strings, we must extend \( \delta \) from a function mapping \( Q \times \Sigma \) to \( Q \), to a function mapping \( Q \times \Sigma^* \) to \( Q \), where \( \Sigma^* \) is the Kleene closure.
- Let \( \delta(q, w) \) be the state that the FSA is in after beginning from state \( q \) and reading the input string \( w \).
- Formally, we require:
  1. \( \hat{\delta}(q, \epsilon) = q \),
  2. for all strings \( w \) and symbols \( a \), \( \hat{\delta}(q, wa) = \delta(\hat{\delta}(q, w), a) \).
- Condition (1) implies that the FSA cannot change state without receiving an input.
- Condition (2) tells us how to find the current state after reading a nonempty input string \( wa \); find \( p = \hat{\delta}(q, w) \), then find \( \delta(p, a) \).
- As \( \hat{\delta}(q, a) = \delta(\hat{\delta}(q, \epsilon), a) = \delta(q, a) \) we shall use \( \delta \) to represent both \( \delta \) and \( \hat{\delta} \) henceforth.
A string $x$ is accepted by a FSA $M = (Q, \Sigma, \delta, i, F)$ if and only if $\delta(i, x) = p$ for some $p \in F$.

The language accepted by $M$, which is denoted as $L(M)$, is that set $\{x | \delta(i, x) \in F\}$.

A language is a regular set, or simply regular, if it is the set accepted by some automaton.

$L(M)$ is the complete set of strings accepted by $M$. 
Consider a modification to the original definition of the FSA, whereby zero, one, or more transitions from a state with the same symbol are allowed.

This new model is known as the nondeterministic finite-state automaton (NFSA).

Observe that there are two edges labeled 0 out of state $i$, one each going back to state $i$ and to state $q_3$. 

Formally define a nondeterministic finite-state automaton (NFSA) as the 5-tuple \((Q, \Sigma, \delta, i, F)\) where

- \(Q\) is a finite set of states,
- \(\Sigma\) is a finite alphabet,
- \(i \in Q\) is the initial state,
- \(F \subseteq Q\) is the set of final states,
- \(\delta\) is the transition function mapping \(Q \times \Sigma\) to \(2^Q\), the power set of \(Q\).

This implies \(\delta(q, a)\) is the set of all states \(p\) such that there is a transition labeled \(a\) from \(q\) to \(p\).
Theorem (equivalence of DFSAs and NFSAs): Let $L$ be the language accepted by a nondeterministic finite-state automaton. Then there exists a deterministic finite-state automaton that accepts $L$. 
Let $M_1 = (Q_1, \Sigma, \delta_1, i_1, F_1)$ denote the NFSA accepting $L$.

Define a DFSA $M_2 = (Q_2, \Sigma, \delta_2, i_2, F_2)$ as follows:

- The states of $M_2$ are all subsets of the states of $M_1$, that is $Q_2 = 2^{Q_1}$.
- $M_2$ keeps track in its states the subset of states that $M_1$ could be in at any given time.
- $F_2$ is the subset of states in $Q_2$ which contain a state $f \in F_1$.
- An element $m \in Q_2$ will be denoted as $m = [m_1, m_2, \ldots, m_N]$, where each $m_n \in Q_1$.
- Finally, $i_2 = [i_1]$. 
Definition of $\delta_2([p_1, p_2, \ldots, p_N], a)$

- By definition,

$$\delta_2([m_1, m_2, \ldots, m_N], a) = [p_1, p_2, \ldots, p_N]$$

if and only if

$$\delta_1(\{m_1, m_2, \ldots, m_N\}, a) = \{p_1, p_2, \ldots, p_N\}.$$  

- In other words, $\delta_2([m_1, m_2, \ldots, m_N], a)$ is computed for $[m_1, m_2, \ldots, m_N] \in Q_2$ by applying $\delta$ to each $m_n \in Q_1$. 
Proof by Induction

- We wish to demonstrate through induction on the string length $|x|$ that
  \[ \delta_2(i_2, x) = [m_1, m_2, \ldots, m_N] \]
  if and only if
  \[ \delta_1(i_1, x) = \{m_1, m_2, \ldots, m_N\}. \]

- **Basis:** The result follows trivially for $|x| = 0$, as $i_2 = [i_1]$ and $x = \epsilon$.

- **Inductive Hypothesis:** Assume that the hypothesis is true for strings of length $N$ or less, and demonstrate it is then necessarily true for strings of length $N + 1$. 
Proof of Inductive Hypothesis

- Let $xa$ be a string of length $N + 1$, where $a \in \Sigma$.
- Then,
  $$\delta_2(i_2, xa) = \delta_2(\delta_2(i_2, x), a).$$
- By the inductive hypothesis,
  $$\delta_2(i_2, x) = [m_1, m_2, \ldots, m_N]$$
  if and only if
  $$\delta_1(i_1, x) = \{m_1, m_2, \ldots, m_N\}.$$
Proof (cont’d.)

- But by the definition of $\delta_2$,

$\delta_2([m_1, m_2, \ldots, m_N], a) = [p_1, p_2, \ldots, p_N]$

if and only if

$\delta_1(\{m_1, m_2, \ldots, m_N\}, a) = \{p_1, p_2, \ldots, p_N\}$.

- Thus,

$\delta_2(i_2, xa) = [p_1, p_2, \ldots, p_N]$

if and only if

$\delta_1(i_1, xa) = \{p_1, p_2, \ldots, p_N\}$,

which establishes the inductive hypothesis.
Implementing the Power Set Construction

- The power set $2^Q$ of $Q$ contains $2^{|Q|}$ subsets.
- This implies that the power set construction requires exponential running time in the worst case; i.e., it is intractable.
- Fortunately, for the FSAs used for speech recognition and many other applications, the vast majority of subsets in the power set are never constructed.
- The key to successfully implementing the power set construction is to not construct a priori all subsets in the power set.
- Rather, only those subsets are constructed which are actually required.
- This subset is comprised of those subsets which are *accessible* from the initial node.
The pseudocode for the power set construction is given below.

```python
def powerSetConstruction(τ₁, τ₂):
    F₂ ← ∅
    i₂ ← i₁
    Q ← {i₂}
    while |Q| > 0:
        pop q₂ from Q
        if ∃ q ∈ q₂ such that q ∈ F₁:
            F₂ ← F₂ ∪ {q₂}
        for a such that δ(q₂, a) ≠ ∅:
            if δ₂(q₂, a) ∉ Q₂:
                Q₂ ← Q₂ ∪ {δ₂(q₂, a)}
        push δ₂(q₂, a) on Q
```
Finite-State Automata with $\epsilon$-Transitions

- We can further extend the definition of finite-state automata to allow $\epsilon$-transitions, which by definition consume no input symbol.

- Formally, define a nondeterministic finite-state automaton with $\epsilon$-transitions as the quintuple $M = (Q, \Sigma, \delta, i, F)$.

- All elements of $M$ have the same meaning as before except that $\delta$ maps $Q \times (\Sigma \cup \{\epsilon\})$ to $2^Q$.

- This implies that $\delta(q, a)$ will consist of all states $m \in Q$ such that there is a transition labeled $a$ from $q$ to $p$, where either $a = \epsilon$ or $a \in \Sigma$.

- As before, we let $L(M)$ denote the language accepted by $M = (Q, \Sigma, \delta, i, F)$ such that $L(M) = \{w | \hat{\delta}(i, w) \text{ contains a state } p \in F\}$. 
We now extend the definition of $\delta$ to $\hat{\delta}$ that maps $Q \times (\Sigma \cup \{\epsilon\})^*$ to $2^Q$.

In the end, $\hat{\delta}(q, w)$ will include all states $p$ such that there is a path from $q$ to $p$ labeled with $w$, perhaps including edges labeled with $\epsilon$.

In computing $\hat{\delta}$, it will be necessary to determine the set of states accessible from a given state $q$ using only $\epsilon$-transitions.
Computing the $\epsilon$-closure($q$)

- We use $\epsilon$–closure($q$) to denote the set of states $p \in Q$ such that there is a path from $q$ to $p$ consisting solely of $\epsilon$-transitions.
- This definition can be extended naturally to a set $P \subseteq Q$ according to

$$\epsilon$–closure(P) = \bigcup_{q \in P} \epsilon$–closure(q).$$
Theorem: If \( L \) is accepted by a NFSA with \( \epsilon \)-transitions, then \( L \) is accepted by a DFSA without \( \epsilon \)-transitions.

- Let \( M_1 = (Q_1, \Sigma, \delta_1, i_1, F_1) \) denote a NFSA with \( \epsilon \)-transitions. Let us construct \( M_2 = (Q_2, \Sigma, \delta_2, i_2, F_2) \) where

\[
F_2 = \begin{cases} 
F_1 \cup \{i_1\}, & \text{if } \epsilon\text{-closure}(i_1) \text{ contains a state } p \in F_1, \\
F_1, & \text{otherwise},
\end{cases}
\]

and \( \delta_2(q, a) = \hat{\delta}_1(q, a) \) for \( q \in Q_1 \) and \( a \in \Sigma \).

- We wish to show by induction on \(|x|\) that \( \delta_2(i_2, x) = \hat{\delta}_1(i_1, x) \).
Inductive Hypothesis

- This may be untrue for $x = \epsilon$, however, as $\delta'(i, \epsilon) = \{i\}$, while $\hat{\delta}(i, \epsilon) = \epsilon$–closure($i$).
- Hence, we begin the induction with $|x| = 1$:
  - **Basis:** For $|x| = 1$, let $x = a$, and $\delta'(i, a) = \hat{\delta}(i, a)$ by the definition of $\delta'$.
  - **Induction:** For $|x| > 1$, let $x = wa$ for $w \in \Sigma^*$ and $a \in \Sigma$. Then
    $$\delta'(i, wa) = \delta'(\delta'(i, w), a).$$
Introduction to Finite-State Automata

Proof of Inductive Hypothesis

- By the inductive hypothesis, $\delta'(i, w) = \hat{\delta}(i, w)$.
- Let $\hat{\delta}(i, w) = P$. We must demonstrate that $\delta'(P, a) = \hat{\delta}(i, wa)$.
- But
  \[
  \delta'(P, a) = \bigcup_{q \in P} \delta'(q, a) = \bigcup_{q \in P} \hat{\delta}(q, a).
  \]

Then as $P = \hat{\delta}(i, w)$ we have
\[
\bigcup_{q \in P} \hat{\delta}(q, a) = \hat{\delta}(i, wa)
\]
by the definition of $\hat{\delta}$.
- Therefore,
  \[
  \delta'(i, wa) = \hat{\delta}(i, wa).
  \]
Completing the Proof

Completing the proof requires demonstrating that $\delta'(i, x)$ contains a state $q' \in F'$ if and only if $\hat{\delta}(i, x)$ contains a state $q \in F$. 
Pseudocode for $\epsilon$–Removal

- In Line 02, all edges not labeled with $\epsilon$ are added to $p$.
- In the for loop

```python
00 def epsilonRemoval(\tau):
01     for \( p \in Q_1 \):
02         Edges[\( p \)] \leftarrow \{ e \in Edges[\( p \)] : Symbol[\( e \)] \neq \epsilon \}
03         for \( q \in \epsilon\text{-closure}[\( p \)] : \)
04             Edges[\( p \)] \leftarrow Edges[\( p \)] \cup \{ (p, a, w \otimes w_1, r) : (q, a, w_1, r) \in Edges[\( q \)], a \neq \epsilon \}
05             if \( q \in F \) and \( p \notin F : \)
06                 F \leftarrow F \cup \{p\}
07             \rho[p] \leftarrow \rho[p] \oplus (w \otimes \rho[q])
```
Definition: Closed Semi-Ring

A closed semiring is a system $S \triangleq (\Sigma, \oplus, \otimes, \bar{0}, \bar{1})$ where $\Sigma$ is a set of elements, $\oplus$ and $\otimes$ are binary operations on elements of $\Sigma$, satisfying the following properties:

1. $(\Sigma, \oplus, \bar{0})$ is a monoid, which implies it is closed under $\oplus$, and $\oplus$ is associative, and $\bar{0}$ is the identity. Likewise, $(\Sigma, \otimes, \bar{1})$ is a monoid. Moreover, we will assume $\bar{0}$ is an annihilator on $\otimes$; i.e., $a \otimes \bar{0} = \bar{0} \otimes a = \bar{0}$.

2. $\oplus$ is commutative; it may also be idempotent such that $a \oplus a = a$.

3. $\otimes$ distributes over $\oplus$, such that $a \otimes (b \oplus c) = a \otimes b \oplus a \otimes c$, and $(b \oplus c) \otimes a = b \otimes a \oplus c \otimes a$.
Definition (cont’d.)

1. If $a_1, a_2, \ldots, a_n, \ldots$ is a countable sequence where $a_n \in S$, then $a_1 \oplus a_2 \oplus \cdots \oplus a_n \oplus \cdots$ exists and is unique. Moreover, associativity and commutativity apply to infinite as well as finite sums.

2.⊗ must distribute over countably infinite as well as finite sums.

Properties 4 and 5 together imply

$$\left( \bigoplus_{n} a_n \right) \otimes \left( \bigoplus_{m} b_m \right) = \bigoplus_{n,m} a_n \otimes b_n = \bigoplus_{n} \left( \bigoplus_{m} (a_n \otimes b_m) \right)$$
Semiring Example 1

- Let $S_1 \triangleq (\{0, 1\}, \oplus, \otimes, 0, 1)$ with $\oplus$ and $\otimes$ defined as follows:

  \[
  \begin{bmatrix}
  \oplus & 0 & 1 \\
  0 & 0 & 1 \\
  1 & 1 & 1 \\
  \end{bmatrix}; \quad \begin{bmatrix}
  \otimes & 0 & 1 \\
  0 & 0 & 0 \\
  1 & 0 & 1 \\
  \end{bmatrix}.
  \]

- Properties 1–3 are easily verified.
- For Properties 4 and 5 note that a countable $\oplus$–sum is 0 iff all terms are 0.
Example 2: Tropical Semiring

- Let $S_2 \triangleq (R, \min, +, \infty, 0)$, where $R$ is the set of nonnegative real numbers including $\infty$.
- It is easy to verify that $\infty$ is the identity under $\min$.
- Similarly, $0$ is the identity under $+$. 
Example 3: String Semiring

- Let $\Sigma$ denote a finite alphabet, and let $S_3 \triangleq (F_\Sigma, \cup, \cdot, \emptyset, \{\epsilon\})$, where $F_\Sigma$ is the family of sets of finite-length strings of symbols from $\Sigma$, including $\epsilon$.
- $\oplus = \cup$ is the set union operator, and $\cdot$ denotes set concatenation.
- The concatenation of sets $A$ and $B$, denoted as $A \cdot B$, is the set $\{x|x = yz, y \in A$ and $z \in B\}$.
- As an exercise, verify properties 1–3.
- For properties 4 and 5, observe that countable unions behave as they should if we define $x \in (A_1 \cup A_2 \cup \cdots)$ iff $x \in A_n$ for some $n$. 
Example 4: Cartesian Product of Semirings

Let $S_4 \triangleq S_2 \times S_3$ where $\times$ denotes the Cartesian product of two semirings.

Prove that $S_4$ is a semiring.
Idempotence

Consider the semiring $S \triangleq (\Sigma, \oplus, \otimes, \bar{0}, \bar{1})$.

For $a \in S$, if $a \oplus a = a$, then $\oplus$ is said to be *idempotent*. 
Let $\ast$ denote the closure operator.

If $(S, \oplus, \otimes, \bar{0}, \bar{1})$ is a closed semiring, and $a \in S$, then define

$$a^\ast \triangleq \bigoplus_{n=0}^{\infty} a^n,$$

where $a^0 \equiv 1$, and $a^n \triangleq a \otimes a^{n-1}$.

This is to say $a^\ast \equiv 1 \oplus a \oplus a \otimes a \oplus a \otimes a \otimes a \cdots$.

Property 4 ensures $a^\ast \in S$.

Properties 4 and 5 together imply $a^\ast = 1 \oplus a \otimes a^\ast$.

Note that $0^\ast = 1^\ast = 1$. 
Example 5

- Consider the semirings $S_1$, $S_2$, and $S_3$ defined in the previous examples.
- For $S_1$, $a^* = 1$, for $a = 0, 1$.
- For $S_2$, $a^* = 0$ for all $a \in R$.
- For $S_3$, $A^* = \{ \epsilon \} \cup \{ x_1 x_2 \cdots x_n | n \geq 1 \text{ and } x_k \in A \text{ for } 1 \leq k \leq n \}$ for all $A \in F_\Sigma$.
- That is $\{a, b\}^* = \{ \epsilon, a, b, aa, ab, ba, bb, aaa, \ldots \}$; i.e., all strings of $a$’s and $b$’s including the empty string.
- In fact $F_\Sigma = 2^{\Sigma^*}$, the power set of $\Sigma^*$. 
Consider a directed graph $G \triangleq (V, E)$ in which each edge $e \in E$ is labeled by an element from some semiring $(S, \oplus, \otimes, \bar{0}, \bar{1})$.

The *label of a path* is the $\otimes$-product of the edge labels in the path taken in the order in which they occur.

For each pair of vertices $(v, w)$, we define $c(v, w)$ to be the $\oplus$-sum of the labels of all paths between $v$ and $w$; we refer to $c(v, w)$ as the *cost* of going from $v$ to $w$.

If $G$ is cyclic, there may be an infinitude of paths from $v$ to $w$; our axiomatic definition of the semiring, however, will ensure that $c(v, w)$ is well defined.
Example 6

Consider the graph in the figure in which each edge is labeled with an element from semiring $S_1$.

- The label of path $v, w, x$ is $1 \cdot 1 = 1$.
- The cycle from $w$ to $w$ has label $1 \cdot 0 = 0$.
- In fact, every path of length greater than zero from $w$ to $w$ has label 0.
- The path of zero length from $w$ to $w$, however, has cost 1; hence $c(w, w) = 1$. 
Summary

- In this lecture, we have defined conventional finite-state automaton.
- We have considered both deterministic and nondeterministic finite-state automata.
- We have also considered the power set construction, whereby a deterministic automaton can be constructed from a nondeterministic automaton.
- In addition, we have generalized the definition of automata to include $\epsilon$-transitions.
- We have seen how an automaton without $\epsilon$-transitions can be constructed from an automaton with $\epsilon$-transitions.
- Finally, we have investigated the use of semirings to generalize the concept of path labels.
Item ...