Introduction to Finite Automata

Languages

Deterministic Finite Automata

Representations of Automata
Alphabets

◆ An *alphabet* is any finite set of symbols.

◆ *Examples*: ASCII, Unicode, \{0,1\} (binary alphabet), \{a, b, c\}.
Strings

- The set of *strings* over an alphabet $\Sigma$ is the set of lists, each element of which is a member of $\Sigma$.
  - Strings shown with no commas, e.g., abc.
- $\Sigma^*$ denotes this set of strings.
- $\epsilon$ stands for the *empty string* (string of length 0).
Example: Strings

\[ \{0,1\}^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, \ldots \} \]

Subtlety: 0 as a string, 0 as a symbol look the same.

Context determines the type.
Languages

◆ A *language* is a subset of $\Sigma^*$ for some alphabet $\Sigma$.

◆ *Example*: The set of strings of 0’s and 1’s with no two consecutive 1’s.

◆ $L = \{\varepsilon, 0, 1, 00, 01, 10, 000, 001, 010, 100, 101, 0000, 0001, 0010, 0100, 0101, 1000, 1001, 1010, \ldots \}$

Hmm… 1 of length 0, 2 of length 1, 3, of length 2, 5 of length 3, 8 of length 4. I wonder how many of length 5?
Deterministic Finite Automata

◆ A formalism for defining languages, consisting of:

1. A finite set of states \((Q, \text{ typically})\).
2. An input alphabet \((\Sigma, \text{ typically})\).
3. A transition function \((\delta, \text{ typically})\).
4. A start state \((q_0, \text{ in } Q, \text{ typically})\).
5. A set of final states \((F \subseteq Q, \text{ typically})\).

◆ “Final” and “accepting” are synonyms.
The Transition Function

- Takes two arguments: a state and an input symbol.
- $\delta(q, a) =$ the state that the DFA goes to when it is in state $q$ and input $a$ is received.
Graph Representation of DFA’s

- Nodes = states.
- Arcs represent transition function.
  - Arc from state $p$ to state $q$ labeled by all those input symbols that have transitions from $p$ to $q$.
- Arrow labeled “Start” to the start state.
- Final states indicated by double circles.
Example: Graph of a DFA

Accepts all strings without two consecutive 1’s.

Previous string OK, does not end in 1.
Previous String OK, ends in a single 1.
Consecutive 1’s have been seen.
### Alternative Representation: Transition Table

- **Rows = states**
- **Columns = input symbols**
- Final states starred
  - * A
  - * B
  - C
- Arrow for start state
  - * A
  - A
  - C

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>C</td>
<td></td>
</tr>
</tbody>
</table>
Extended Transition Function

- We describe the effect of a string of inputs on a DFA by extending \( \delta \) to a state and a string.
- Induction on length of string.
- **Basis:** \( \delta(q, \epsilon) = q \)
- **Induction:** \( \delta(q,wa) = \delta(\delta(q,w),a) \)
  - w is a string; a is an input symbol.
Extended $\delta$: Intuition

◆ **Convention:**
  - $\ldots w, x, y, x$ are strings.
  - $a, b, c, \ldots$ are single symbols.

◆ **Extended $\delta$ is computed for state $q$ and inputs $a_1 a_2 \ldots a_n$ by following a path in the transition graph, starting at $q$ and selecting the arcs with labels $a_1, a_2, \ldots, a_n$ in turn.**
Example: Extended Delta

$$\delta(B,011) = \delta(\delta(B,01),1) = \delta(\delta(\delta(B,0),1),1) =$$

$$\delta(\delta(A,1),1) = \delta(B,1) = C$$
In book, the extended $\delta$ has a “hat” to distinguish it from $\delta$ itself.

Not needed, because both agree when the string is a single symbol.

$\delta(q, a) = \delta(\delta(q, \varepsilon), a) = \delta(q, a)$
Language of a DFA

- Automata of all kinds define languages.
- If A is an automaton, $L(A)$ is its language.
- For a DFA $A$, $L(A)$ is the set of strings labeling paths from the start state to a final state.
- Formally: $L(A) = \{w \mid \delta(q_0, w) \in F\}$. 
Example: String in a Language

String 101 is in the language of the DFA below.
Start at A.
Example: String in a Language

String 101 is in the language of the DFA below.

Follow arc labeled 1.
Example: String in a Language

String 101 is in the language of the DFA below.

Then arc labeled 0 from current state B.
Example: String in a Language

String 101 is in the language of the DFA below.

Finally arc labeled 1 from current state A. Result is an accepting state, so 101 is in the language.
Example – Concluded

The language of our example DFA is:

\[ \{ w \mid w \text{ is in } \{0,1\}^* \text{ and } w \text{ does not have two consecutive 1’s}\} \]

Read a *set former* as

“The set of strings w…

Such that…

These conditions about w are true.
Proofs of Set Equivalence

- Often, we need to prove that two descriptions of sets are in fact the same set.
- Here, one set is “the language of this DFA,” and the other is “the set of strings of 0’s and 1’s with no consecutive 1’s.”
Proofs – (2)

♦ In general, to prove $S=\mathcal{T}$, we need to prove two parts: $S \subseteq \mathcal{T}$ and $\mathcal{T} \subseteq S$.

That is:

1. If $w$ is in $S$, then $w$ is in $\mathcal{T}$.
2. If $w$ is in $\mathcal{T}$, then $w$ is in $S$.

♦ As an example, let $S =$ the language of our running DFA, and $\mathcal{T} =$ “no consecutive 1’s.”
**Part 1:** \( S \subseteq T \)

- **To prove:** if \( w \) is accepted by
  
then \( w \) has no consecutive 1’s.

- **Proof is an induction on length of \( w \).**

- **Important trick:** Expand the inductive hypothesis to be more detailed than you need.
The Inductive Hypothesis

1. If $\delta(A, w) = A$, then $w$ has no consecutive 1’s and does not end in 1.
2. If $\delta(A, w) = B$, then $w$ has no consecutive 1’s and ends in a single 1.

- **Basis:** $|w| = 0$; i.e., $w = \varepsilon$.
  - (1) holds since $\varepsilon$ has no 1’s at all.
  - (2) holds *vacuously*, since $\delta(A, \varepsilon)$ is not B.

"length of"

**Important concept:**
If the "if" part of "if..then" is false, the statement is true.
Inductive Step

- Assume (1) and (2) are true for strings shorter than $w$, where $|w|$ is at least 1.
- Because $w$ is not empty, we can write $w = xa$, where $a$ is the last symbol of $w$, and $x$ is the string that precedes.
- IH is true for $x$. 
Inductive Step – (2)

- Need to prove (1) and (2) for \( w = xa \).
- (1) for \( w \) is: If \( \delta(A, w) = A \), then \( w \) has no consecutive 1’s and does not end in 1.
- Since \( \delta(A, w) = A \), \( \delta(A, x) \) must be A or B, and \( a \) must be 0 (look at the DFA).
- By the IH, \( x \) has no 11’s.
- Thus, \( w \) has no 11’s and does not end in 1.
Inductive Step – (3)

 Now, prove (2) for w = xa: If \( \delta(A, w) = B \), then w has no 11’s and ends in 1.

 Since \( \delta(A, w) = B \), \( \delta(A, x) \) must be A, and \( a \) must be 1 (look at the DFA).

 By the IH, x has no 11’s and does not end in 1.

 Thus, w has no 11’s and ends in 1.
Part 2: \( T \subseteq S \)

- Now, we must prove: if \( w \) has no 11’s, then \( w \) is accepted by \( \mathcal{V} \).

- Contrapositive: If \( w \) is not accepted by \( \mathcal{V} \), then \( w \) has 11.

Key idea: contrapositive of “if \( X \) then \( Y \)” is the equivalent statement “if not \( Y \) then not \( X \).”
Using the Contrapositive

◆ Every $w$ gets the DFA to exactly one state.
  ♦ Simple inductive proof based on:
    • Every state has exactly one transition on 1, one transition on 0.
◆ The only way $w$ is not accepted is if it gets to $C$. 

29
Using the Contrapositive – (2)

- The only way to get to C [formally: \( \delta(A,w) = C \)] is if \( w = x1y \), \( x \) gets to B, and \( y \) is the tail of \( w \) that follows what gets to C for the first time.

- If \( \delta(A,x) = B \) then surely \( x = z1 \) for some \( z \).

- Thus, \( w = z11y \) and has 11.
Regular Languages

◆ A language $L$ is *regular* if it is the language accepted by some DFA.
  ✤ **Note**: the DFA must accept only the strings in $L$, no others.

◆ Some languages are not regular.
  ✤ Intuitively, regular languages “cannot count” to arbitrarily high integers.
Example: A Nonregular Language

$L_1 = \{0^n1^n \mid n \geq 1\}$

- **Note**: $a^i$ is conventional for $i$ $a$'s.
  - Thus, $0^4 = 0000$, e.g.

- **Read**: “The set of strings consisting of $n$ 0’s followed by $n$ 1’s, such that $n$ is at least 1.

- Thus, $L_1 = \{01, 0011, 000111, \ldots\}$
Another Example

$L_2 = \{ w \mid w \text{ in } \{(, )\}^* \text{ and } w \text{ is balanced} \}$

♦ **Note:** alphabet consists of the parenthesis symbols ‘(’ and ‘)’.

♦ Balanced parens are those that can appear in an arithmetic expression.
  • E.g.: (), ()(), (), (), (),…
But Many Languages are Regular

- Regular Languages can be described in many ways, e.g., regular expressions.
- They appear in many contexts and have many useful properties.
- **Example**: the strings that represent floating point numbers in your favorite language is a regular language.
Example: A Regular Language

$L_3 = \{ w \mid w \text{ in } \{0,1\}^* \text{ and } w, \text{ viewed as a binary integer is divisible by } 23\}$

◆ The DFA:
  ◆ 23 states, named 0, 1, ..., 22.
  ◆ Correspond to the 23 remainders of an integer divided by 23.
  ◆ Start and only final state is 0.
Transitions of the DFA for $L_3$

- If string $w$ represents integer $i$, then assume $\delta(0, w) = i \mod 23$.
- Then $w0$ represents integer $2i$, so we want $\delta(i \mod 23, 0) = (2i) \mod 23$.
- Similarly: $w1$ represents $2i+1$, so we want $\delta(i \mod 23, 1) = (2i+1) \mod 23$.
- **Example:** $\delta(15, 0) = 30 \mod 23 = 7$; $\delta(11, 1) = 23 \mod 23 = 0$. **Key idea:** design a DFA by figuring out what each state needs to remember about the past.
Another Example

$L_4 = \{ w | w \text{ in } \{0,1\}^* \text{ and } w, \text{ viewed as } \text{the reverse of a binary integer is divisible by 23} \}$

- **Example:** 01110100 is in $L_4$, because its reverse, 00101110 is 46 in binary.
- **Hard to construct the DFA.**
- **But theorem says the reverse of a regular language is also regular.**