Today’s agenda

• NFA Review and Design

• NFA’s Equivalence to DFA’s

• Another Closure Property proof for Regular Languages

Announcements:

Reminders:
• HW 2 Due Friday, April 8
• Reading Quiz 3 Due Wednesday, April 13
Nondeterministic Finite Automata

Guessing added to DFA’s!
Review: DFAs vs. NFAs

DFAs

• For each character in the alphabet, exactly one transition leaving every state
• Computation is determined by the input, i.e., only one choice of next state every time for a given input

NFAs

• There may be 0, 1, or many transitions leaving a state for the same input character.
• Transitions may be labeled with the “empty string” $\varepsilon$. You can take this choice without using up input!
  • “spontaneous action”
• There may be several different ways to reach a final state for a string
  • “nondeterministic”
Review: Formal Definition of an NFA

• An NFA $M$ is defined as a 5-tuple as follows:

• $M = (Q, \Sigma, \delta, q_0, F)$, where:
  • $Q$ is a finite set of states
  • $\Sigma$ is a finite set of characters, the alphabet
  • $\delta: Q \times (\Sigma \cup \{\varepsilon\}) \rightarrow P(Q)$, the transition function
  • $q_0$, a member of $Q$, the start state
  • $F$, a subset of $Q$, the accept state(s)
NFA State Diagram to Formal Description

Example:

Set of states $Q = ? \{q0, q1\}$

Alphabet $\Sigma = ? \{0, 1\}$

Start state $= ? q0$

Set of final states $F = ? \{q1\}$

Recall that $\delta: Q \times (\Sigma \cup \{\varepsilon\}) \rightarrow P(Q)$

<table>
<thead>
<tr>
<th>$\delta$</th>
<th>0</th>
<th>1</th>
<th>$\varepsilon$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q0$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q1$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
NFA diagram to $\delta$

What is the formal description of the transition function $\delta$: $Q \times (\Sigma \cup \{\varepsilon\}) \rightarrow P(Q)$ for this NFA diagram?

A. $\delta(q_0, 0) = \{q_0, q_1\}$, $\delta(q_0, 1) = \{q_0\}$, $\delta(q_1, 0) = \emptyset$, $\delta(q_1, 1) = \emptyset$

B. $\delta(q_0, 0) = q_0$, $\delta(q_0, 0) = q_1$, $\delta(q_0, 1) = q_0$

C. $\delta(q_0, 0) = \{q_0, q_1\}$, $\delta(q_0, 1) = \{q_0\}$, and for all other cases of $q$ in $Q$ and $a$ in $\Sigma \cup \{\varepsilon\}$, $\delta(q, a) = \emptyset$

D. $\delta(q_0, \{0, 1\}) = q_0$, $\delta(q_0, 0) = q_1$, $\delta(q_1, \emptyset) = q_1$

E. None of the above.
Review: NFA Acceptance

“100”

Is 100 accepted by the NFA?
A. Yes
B. No
C. I don’t know
Review: Tracing in an NFA

What are the two sequences of states on the input “100”?

A. \( (q_0,q_0,q_1,q_2[\text{accept}]), (q_0,q_1,q_2[\text{accept}]) \)
   Final: Accept

B. \( (q_0,q_0,q_1,q_2[\text{accept}]), (q_0,q_1,q_2[\text{reject}]) \)
   Final: Accept

C. \( (q_0,q_0,q_1,q_2[\text{accept}]), (q_0,q_1,q_2[\text{reject}]) \)
   Final: Reject

D. \( (q_0,q_0,q_1,q_2[\text{reject}]), (q_0,q_1,q_2[\text{reject}]) \)
   Final: Reject
Is 11 accepted by this NFA?
A. Yes
B. No
C. Sometimes
Review: Tracing in NFA

NFA with input 010110

Each row is a set of states that we are in at the “same time”

- \( \{q1\} \)
- \( \{q1, q2, q3\} \)
- \( \{q1, q3\} \)
- \( \{q1, q2, q3, q4\} \)
- \( \{q1, q3, q4\} \)
Nondeterminism

- Because NFAs are non-deterministic, the outcome (accept/reject) of the computation may be different from path to path in a trace on the same input.

A. TRUE
B. FALSE
What is the language of this example NFA?
A. \{w \mid \text{sum of } w\text{'s digits is a multiple of 2 or 3}\}
B. \{w \mid \text{sum of } w\text{'s digits is a multiple of 2 or 3}\} \cup \{\epsilon\}
C. None of the above
Designing NFA’s

Similar to designing DFA’s in that
• States are the only mechanism to “remember”
• Helps to associate a “meaning” with each state

Different than designing DFA’s in that
• More freedom on transitions
  • Multiple edges with same label from a node, $\varepsilon$ edges
  • Need not have edge for every symbol in $\Sigma$
• Parallelism
  • Launch threads to fork to all next possible states
  • If no possible next step, thread becomes dead
  • Accept if after reading all input, there is some live thread that is in a final state; otherwise reject.
Given a DFA D recognizing language L, can we always write a formal definition of an NFA whose language is L?

A. Yes
B. No
C. Sometimes but not always

N is equivalent to D (since they accept same language L)
Every DFA has an equivalent NFA!
Example 1

Construct an NFA that recognizes the following language with alphabet $\Sigma = \{0, 1\}$

$L = \{ w \mid w \text{ has a 1 in the third position from the right} \}$
Example 2

Let $\Sigma = \{0, 1, 2\}$, we define 
$L = \{w#c \mid c \in \Sigma, w \in \Sigma^*, \text{and } c \text{ occurs in } w\}$ 
(over alphabet $\Sigma \cup \{\#\}$)

Design an NFA that recognizes $L$. 
Another construction proof

Equivalence of Finite Automata  
NFA & DFA
Tracing in NFA with $\varepsilon$ Edges

Run input 010110 on this NFA:

- Each row is a set of states that we are in at the “same time”
  - $\{q_1\}$
  - $\{q_1,q_2,q_3\}$
  - $\{q_1,q_3\}$
  - $\{q_1,q_2,q_3,q_4\}$
  - $\{q_1,q_3,q_4\}$

- Recall that when we did the union closure proof with DFAs, we were always in a pair of states at the “same time”—similar concept
Thm 1.39: Every NFA has an equivalent DFA.

- **Given:** NFA \( N = (Q, \Sigma, \delta, q_0, F) \), where
  \[ \delta: Q \times (\Sigma \cup \{ \epsilon \}) \rightarrow P(Q), \]
- **Want:** DFA \( D = (Q', \Sigma, \delta', q_0', F') \) s.t. \( L(D) = L(N) \).

**Construction:** //need to make a DFA that simulates Nondeterminism by keeping track of the set of states of NFA!!
- **Case 1:** NFA \( N \) has no \( \epsilon \) edges. Define \( D \) as follows:
  \[ Q' = P(Q) \]
  \[ \Sigma \text{ is the same} \]
  \[ \delta'(R, a) = \{ q \text{ in } Q \mid q \text{ in } \delta(r, a) \text{ for some } r \text{ in } R \} \]
  \[ q_0' = \{ q_0 \} \]
  \[ F' = \{ R \mid R \subseteq Q \text{ and } R \cap F \text{ is nonempty} \} \]
- **Case 2:** NFA \( N \) has \( \epsilon \) edges (Proof in textbook)
- **A DFA recognizes** \( L(N) \), therefore every NFA has an equivalent DFA. Q.E.D.
ε edges

Example 1.38 (Sipser)
Convert this NFA to
An NFA without ε edges

To construct the DFA:
Keep track of the set of all states that NFA could be in, including ε edges
Thm. 1.39: Every NFA has an equivalent DFA.

- We also know every DFA is equivalent to an NFA.
- Corollary of these two facts:
  - The class of languages recognized by DFAs and the class of languages recognized by NFAs are the same class:

The Class of Regular Languages

- Surprising that adding something as powerful as guessing/parallelism to the DFA model could turn out to not increase the power of the model!
- You can use either model in proofs
Why NFA’s are so Useful for Proofs

CLOSURE OF REGULAR LANGUAGES UNDER CONCATENATION
Our working example

Basic idea: Put side by side, and add “spontaneous” transition from every final state of $M_1$ to start state of $M_2$
Thm 1.47. The class of regular languages over fixed $\Sigma$ is closed under concatenation

• Proof:
• Given: Two regular languages $L_1$, $L_2$.
• Want to show: $L_1 \circ L_2$ is regular.
• Because $L_1$ and $L_2$ are regular, we know there exist NFAs $M_1 = (Q_1, \Sigma, \delta_1, q_{01}, F_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, q_{02}, F_2)$ that recognize $L_1$ and $L_2$.
• We construct a new NFA $M = (Q, \Sigma, \delta, q_0, F)$, s.t.: $Q = Q_1 \cup Q_2$ $\Sigma$ is the same $q_0 = q_{01}$ $F = F_2$
$\delta(q,a) = ?$
Thm 1.47. The class of regular languages over fixed $\Sigma$ is closed under concatenation.

Which is an incorrect case for $\delta(q,a) =$ ?

A. $\delta_1(q,a)$ if $q$ in $Q_1$

B. $\delta_1(q,a)$ if $q$ in $Q_1$ and $q$ not in $F_1$

C. $\delta_1(q,a) \cup \{q_02\}$ if $q$ in $F_1$ and $a = \varepsilon$

D. $\delta_2(q,a)$ if $q$ in $Q_2$

E. More than one of the above
Proof:

Given: Two regular languages $L_1$, $L_2$.

Want to show: $L_1 \circ L_2$ is regular.

Because $L_1$ and $L_2$ are regular, we know there exist NFAs $M_1 = (Q_1, \Sigma, \delta_1, q_{01}, F_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, q_{02}, F_2)$ that recognize $L_1$ and $L_2$. We construct an NFA $M = (Q, \Sigma, \delta, q_0, F)$, s.t.:

- $Q = Q_1 \cup Q_2$
- $\Sigma$ is the same
- $q_0 = q_{01}$
- $F = F_2$
- $\delta(q, a) = \begin{cases} 
\delta_1(q, a) & \text{if } q \text{ in } Q_1 \text{ and } q \text{ not in } F_1 \\
\delta_1(q, a) & \text{if } q \text{ in } F_1 \text{ and } a \neq \varepsilon \\
\delta_1(q, a) \cup \{q_{02}\} & \text{if } q \text{ in } F_1 \text{ and } a = \varepsilon \\
\delta_2(q, a) & \text{if } q \text{ in } Q_2
\end{cases}$

Claim to prove: $L(M) = L_1 \circ L_2$
Thm 1.49. The class of regular languages over fixed $\Sigma$ is closed under * (star).

- Proof also uses NFA’s
- Basic Idea, given NFA $N = (Q, \Sigma, \delta, q_0, F)$, define $N'$:
  - For every final state in $N$, add $\varepsilon$ edge back to $q_0$, $N$’s start state
  - Create a new start state $q'$, add to $F$, and add $\varepsilon$ edge to $q_0$
- See Proof in Sipser
Qu: In your opinion, would it be easier for you to write a program to simulate a DFA or an NFA?

A. NFA, because they seem to be more powerful and can compute more
B. DFA, because they do one step at a time
C. Neither of the above
Summary of Closure Properties of Regular Languages

The class of regular languages is closed under:

- Union
  - Sipser, Th. 1.25
- Intersection
  - Lec. 3
- Complement
  - HW 2
- Symmetric Difference
  - Lec. 3, Ex. 4
- Concatenation
  - Sipser, Th. 1.47
- Star (*)
  - Sipser, Th 1.49
- Flipping Bits
  - Lec. 3, Ex. 1

Other examples:
- Symbol-by-symbol translation
  - Lec. 3, Ex. 5
- Deleting a symbol
  - Lec 3, Ex. 2