Announcements

• HW 2 out
  • Due Mon 2/19 12pm EST (noon)
  • Due Wed 2/21 12pm EST (noon)

• No class Mon (2/19)
Another operation: Concatenation

Example: Recognizing street addresses

212 Beacon Street

$M_3$: “CONCAT”

$M_1$: recognize numbers

$M_2$: recognize words
Concatenation of Languages

Let the alphabet $\Sigma$ be the standard 26 letters $\{a, b, \ldots, z\}$.

If $A = \{\text{fort, south}\}$, $B = \{\text{point, boston}\}$

$$A \circ B = \{\text{fortpoint, fortboston, southpoint, southboston}\}$$
The class of regular languages is closed under the concatenation operation. In other words, if $A_1$ and $A_2$ are regular languages then so is $A_1 \circ A_2$.

- **Construct:** *new machine* $M$ recognizing $A_1 \circ A_2$? (like union)
  - Using: DFA $M_1$ (which recognizes $A_1$),
  - and DFA $M_2$ (which recognizes $A_2$)
Let $M_1$ recognize $A_1$, and $M_2$ recognize $A_2$.

**Want:** Construction of $M$ to recognize $A_1 \circ A_2$.

**Problem:** Can only read input once! (can’t backtrack)

**Need to switch machines at some point, but when?**
Overlapping Concatenation Example

• Let $M_1$ recognize language $A = \{\text{jen, jens}\}$
• and $M_2$ recognize language $B = \{\text{smith}\}$
• Want: Construct $M$ to recognize $A \circ B = \{\text{jen smith, jens smith}\}$

• If $M$ sees jen...
• $M$ must decide to either:
Overlapping Concatenation Example

• Let $M_1$ recognize language $A = \{jen, jens\}$
• and $M_2$ recognize language $B = \{smith\}$
• Want: Construct $M$ to recognize $A \circ B = \{jensmith, jenssmith\}$

• If $M$ sees $jen$...
• $M$ must decide to either:
  • stay in $M_1$ (correct, if full input is $jenssmith$)
Overlapping Concatenation Example

- Let $M_1$ recognize language $A = \{ \text{jen, } \text{jens} \}$
- and $M_2$ recognize language $B = \{ \text{smith} \}$
- Want: Construct $M$ to recognize $A \circ B = \{ \text{jensmith, jenssmith} \}$

- If $M$ sees $\text{jen}$...
- $M$ must decide to either:
  - stay in $M_1$ (correct, if full input is $\text{jenssmith}$)
  - or switch to $M_2$ (correct, if full input is $\text{jensmith}$)

- To recognize $A \circ B$, it needs to handle both cases!!
  - (Without backtracking)
Is Concatenation Closed?

FALSE?

**Theorem**

The class of regular languages is closed under the concatenation operation.

In other words, if $A_1$ and $A_2$ are regular languages then so is $A_1 \circ A_2$.

- Cannot combine $A_1$ and $A_2$'s machine because:
  - Need to switch from $A_1$ to $A_2$ at some point ...
  - ... but we don’t know when! (we can only read input once)

- What if: we create a new kind of machine!

- But does this mean concatenation is not closed for regular langs?
Deterministic vs Nondeterministic

Deterministic computation

- start
- states
- ...
- accept or reject

DFAs
Deterministic vs Nondeterministic

Deterministic computation

Nondeterministic computation

- start
- states
- reject
- accept or reject
- accept

DFAs

New FA

Nondeterministic computation can be in multiple states at the same time.
DFAs: The Formal Definition

**Definition**

A finite automaton is a 5-tuple \((Q, \Sigma, \delta, q_0, F)\), where

1. \(Q\) is a finite set called the **states**,
2. \(\Sigma\) is a finite set called the **alphabet**,
3. \(\delta : Q \times \Sigma \rightarrow Q\) is the **transition function**,
4. \(q_0 \in Q\) is the **start state**, and
5. \(F \subseteq Q\) is the **set of accept states**.

**Deterministic Finite Automata (DFA)**
Nondeterministic Finite Automata (NFA)

**Definition**

A **nondeterministic finite automaton** is a 5-tuple \( (Q, \Sigma, \delta, q_0, F) \), where

1. \( Q \) is a finite set of states,
2. \( \Sigma \) is a finite alphabet,
3. \( \delta : Q \times \Sigma \varepsilon \rightarrow \mathcal{P}(Q) \) is the transition function,
4. \( q_0 \in Q \) is the start state, and
5. \( F \subseteq Q \) is the set of accept states.

*Compare with DFA:*

A finite automaton is a 5-tuple \( (Q, \Sigma, \delta, q_0, F) \), where

1. \( Q \) is a finite set called the **states**,  
2. \( \Sigma \) is a finite set called the **alphabet**,  
3. \( \delta : Q \times \Sigma \rightarrow \mathcal{P}(Q) \) is the **transition function**,  
4. \( q_0 \in Q \) is the **start state**, and  
5. \( F \subseteq Q \) is the **set of accept states**.

*Difference*

- Power set, i.e. a transition results in set of states
Power Sets

• **A power set** is the set of all subsets of a set

• **Example**: $S = \{a, b, c\}$

• Power set of $S =$
  • $\{\{\}, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}, \{a, b, c\}\}$
  • **Note**: includes the empty set!
A nondeterministic finite automaton is a 5-tuple \((Q, \Sigma, \delta, q_0, F)\), where

1. \(Q\) is a finite set of states,
2. \(\Sigma\) is a finite alphabet,
3. \(\delta : Q \times \Sigma_\varepsilon \rightarrow \mathcal{P}(Q)\) is the transition function,
4. \(q_0 \in Q\) is the start state, and
5. \(F \subseteq Q\) is the set of accept states.

\[\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}\]

**CAREFUL:**
- \(\varepsilon\) symbol is reused here, as a transition label.
- It’s not the empty string!
- And it’s (still) not a character in the alphabet \(\Sigma\)!
NFA Example

• Come up with a formal description of the following NFA:

**Definition**

A nondeterministic finite automaton is a 5-tuple \((Q, \Sigma, \delta, q_0, F)\), where

1. \(Q\) is a finite set of states,
2. \(\Sigma\) is a finite alphabet,
3. \(\delta: Q \times \Sigma_e \rightarrow \mathcal{P}(Q)\) is the transition function,
4. \(q_0 \in Q\) is the start state, and
5. \(F \subseteq Q\) is the set of accept states.
The formal description of $N_1$ is $(Q, \Sigma, \delta, q_1, F)$, where

1. $Q = \{q_1, q_2, q_3, q_4\}$,
2. $\Sigma = \{0, 1\}$,
3. $\delta$ is given as

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>$\varepsilon$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_1$</td>
<td>${q_1}$</td>
<td>${q_1, q_2}$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>$q_2$</td>
<td>${q_3}$</td>
<td>$\emptyset$</td>
<td>${q_3}$</td>
</tr>
<tr>
<td>$q_3$</td>
<td>$\emptyset$</td>
<td>${q_4}$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>$q_4$</td>
<td>${q_4}$</td>
<td>${q_4}$</td>
<td>$\emptyset$</td>
</tr>
</tbody>
</table>

$\delta: Q \times \Sigma \varepsilon \rightarrow \mathcal{P}(Q)$

4. $q_1$ is the start state, and
5. $F = \{q_4\}$. 

\[ Schema: q_1 \rightarrow 1 \rightarrow 1 \rightarrow 1 \rightarrow q_4 \\] 

\[ Diagram: q_1 \rightarrow 1 \rightarrow q_2 \rightarrow 0 \rightarrow q_3 \rightarrow \varepsilon \rightarrow 0 \rightarrow q_4 \\]
In-class Exercise

• Come up with a formal description for the following NFA
  • $\Sigma = \{ a, b \}$

**DEFINITION**

A *nondeterministic finite automaton* is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

1. $Q$ is a finite set of states,
2. $\Sigma$ is a finite alphabet,
3. $\delta: Q \times \Sigma_e \rightarrow \mathcal{P}(Q)$ is the transition function,
4. $q_0 \in Q$ is the start state, and
5. $F \subseteq Q$ is the set of accept states.
In-class Exercise Solution

Let \( N = (Q, \Sigma, \delta, q_0, F) \)

- \( Q = \{ q_1, q_2, q_3 \} \)
- \( \Sigma = \{ a, b \} \)
- \( \delta \) ...
- \( q_0 = q_1 \)
- \( F = \{ q_1 \} \)

\[
\delta(q_1, a) = \{ \}
\delta(q_1, b) = \{ q_2 \}
\delta(q_1, \varepsilon) = \{ q_3 \}
\delta(q_2, a) = \{ q_2, q_3 \}
\delta(q_2, b) = \{ q_3 \}
\delta(q_2, \varepsilon) = \{ \}
\delta(q_3, a) = \{ q_1 \}
\delta(q_3, b) = \{ \}
\delta(q_3, \varepsilon) = \{ \}
\]
NFA Computation (JFLAP demo): 010110
NFA Computation Sequence

Symbol read

0

1

0

1

0

NFA accepts input if: at least one path ends in accept state

Each step can branch into multiple states at the same time!

So this is an accepting computation
Submit in-class work 2/16

On gradescope