A machine $M$ that consists of:

- an alphabet $\Sigma$
- a set of states, including:
  - start state
  - accepting state(s)
- transitions between states
  
  for every state, every letter in $\Sigma$ labels one and only one transition

Given a string $w \in \Sigma^*$, $M$ accepts $w$ if consuming $w$ causes $M$ to terminate in an accepting state.
Which language would you use to implement DFAs?

(A) Racket

(B) Prolog

(C) Hmmm

(D) Python

(E) Java
Draw these DFAs where $\Sigma = \{a,b,c\}$

Some examples taken from Rich’s “Automata, Computability, and Complexity”, Chapter 5

(1) $L = \{abc\}$

(2) $L = \{x\ abc\ y \mid x, y \in \Sigma^*\}$

(3) $L = \{w \mid w \text{ contains all but one of the letters of } \Sigma\}$
Non-deterministic Finite Automaton

Fewer restrictions than a DFA

A machine $M$ that consists of:

- an **alphabet** $\Sigma$
- a set of **states**, including:
  - start state
  - accepting state(s)
- **transitions** between states
  - for every state, every letter in $\Sigma$ labels one and only one transition
  - can be a $\lambda$ transition

Given a string $w \in \Sigma^*$, $M$ accepts $w$ if consuming $w$ causes $M$ to terminate in an accepting state.
$\Sigma = \{a, b\} \quad L = \{a?ab*\}$
$\Sigma = \{a, b\}$  \quad $L = \{aba \text{ or } w \text{ where } |w| \text{ is even}\}$
Which language would you use to implement NFAs?

(A) Racket
(B) Prolog
(C) Hmmm
(D) Python
(E) Java
Draw these NFAs where $\Sigma = \{a, b, c\}$

Some examples taken from Rich’s “Automata, Computability, and Complexity”, Chapter 5

(1) $L = \{abc\}$

(2) $L = \{x \ abcab\ b\ y \mid x, y \in \Sigma^*\}$

(3) $L = \{w \mid w \text{ contains all but one of the letters of } \Sigma\}$
Competing Models of Computation?

DFAs vs NFAs?
Equivalent Models of Computation

\[ \text{DFAs} \equiv \text{NFAs} \]

Rabin & Scott
Regular Languages

\[
\text{DFAs} \equiv \text{NFAs} \equiv \text{Regular Expressions}
\]

Rabin & Scott
Kleene
Regular Languages FTW!
The benefits of equivalent models?

(1) Express problem as regular expression (RE)

(2) Convert RE to NFA

(3) Convert NFA to DFA

(4) Minimize DFA

(5) Profit!

So...no more math homework?
Regular Languages FTW?

(1) $L = \{a^N b^N \mid N > 0\}$ // equality

(2) $L = \{a^N b^{2N} \mid N > 0\}$ // multiplication

(3) $L = \{a^N b^M c^{(N+M)} \mid N, M > 0\}$ // addition

Not Regular

$\not\exists$ a DFA that accepts $L$
Can you prove it?
Remember this?

**Distinguishability theorem**

If a set $S = \{w_1, w_2, \ldots, w_n\}$ is **pairwise distinguishable** for a language $L$ then a DFA that accepts $L$ must have $\geq n$ states.

![DFA Diagram]

<table>
<thead>
<tr>
<th>Pairwise Distinguishable</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda$</td>
</tr>
<tr>
<td>111</td>
</tr>
<tr>
<td>1111111</td>
</tr>
</tbody>
</table>

<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>11111111</td>
</tr>
</tbody>
</table>

<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
</tr>
<tr>
<td>111111111</td>
</tr>
</tbody>
</table>

<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>111</td>
</tr>
<tr>
<td>1111111111</td>
</tr>
</tbody>
</table>
Myhill–Nerode theorem

A language \( L \) is **regular** if and only if we can define a set \( S \) of pairwise distinguishable strings such that \( |S| \) is **finite**.

So...how can we prove that \( L \) is *not* regular?
Prove that $L = \{a^N b^N \mid N > 0\}$ is not regular.

Let $S = \{a^+\}$

<table>
<thead>
<tr>
<th>$w_i$</th>
<th>$w_j$</th>
<th>$z$</th>
<th>Accept $w_i z$</th>
<th>Reject $w_j z$</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>aa</td>
<td>b</td>
<td>ab</td>
<td>aab</td>
</tr>
<tr>
<td>a</td>
<td>aaa</td>
<td>b</td>
<td>ab</td>
<td>aaab</td>
</tr>
<tr>
<td>a</td>
<td>aaaa</td>
<td>b</td>
<td>ab</td>
<td>aaaaab</td>
</tr>
<tr>
<td>a</td>
<td>aaaaa</td>
<td>b</td>
<td>ab</td>
<td>aaaaab</td>
</tr>
<tr>
<td>a</td>
<td>aaaaaa</td>
<td>b</td>
<td>ab</td>
<td>aaaaab</td>
</tr>
</tbody>
</table>
Regular Languages FTW?

(1) \( L = \{ a^N b^N \mid N > 0 \} \)  // equality

(2) \( L = \{ a^N b^{2N} \mid N > 0 \} \)  // multiplication

(3) \( L = \{ a^N b^M c^{(N+M)} \mid N, M > 0 \} \)  // addition

Not Regular
No DFA “decides” these languages