Lecture 2: Finite Automata and Nondeterminism
6.045

No Problem Set this week!
They’ll start next week

Recitations start tomorrow
The DFA accepts a string $x$ if the process on $x$ ends in a double circle.

The above DFA accepts exactly those strings with an odd number of 1s.
A DFA is a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$

$Q$ is the set of states (finite)

$\Sigma$ is the alphabet (finite)

$\delta : Q \times \Sigma \rightarrow Q$ is the transition function

$q_0 \in Q$ is the start state

$F \subseteq Q$ is the set of accept/final states

$L(M) =$ set of all strings that $M$ accepts

= “the language recognized by $M$”

= the function computed by $M$
Definition: A language $L'$ is regular if $L'$ is recognized by a DFA; that is, there is a DFA $M$ where $L' = L(M)$. 

$L(M) = \text{set of all strings that } M \text{ accepts} = \text{“the language recognized by } M\text{”}$ 

A DFA is a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$.
Theorem: The union of two regular languages is also a regular language

Proof: Let

\[ M_1 = (Q_1, \Sigma, \delta_1, q_0^1, F_1) \] be a finite automaton for \( L_1 \)

and

\[ M_2 = (Q_2, \Sigma, \delta_2, q_0^2, F_2) \] be a finite automaton for \( L_2 \)

We want to construct a finite automaton

\[ M = (Q, \Sigma, \delta, q_0, F) \] that recognizes \( L = L_1 \cup L_2 \)
Proof Idea: Run both $M_1$ and $M_2$ “in parallel”!

$$M_1 = (Q_1, \Sigma, \delta_1, q_0, F_1) \text{ recognizes } L_1 \text{ and }$$

$$M_2 = (Q_2, \Sigma, \delta_2, q_0, F_2) \text{ recognizes } L_2$$

$Q$ = pairs of states, one from $M_1$ and one from $M_2$

$$= \{ (q_1, q_2) \mid q_1 \in Q_1 \text{ and } q_2 \in Q_2 \}$$

$$= Q_1 \times Q_2$$

$q_0 = (q_0^1, q_0^2)$

$F = \{ (q_1, q_2) \mid q_1 \in F_1 \text{ OR } q_2 \in F_2 \}$

$\delta( (q_1, q_2), \sigma) = (\delta_1(q_1, \sigma), \delta_2(q_2, \sigma))$
Theorem: The union of two regular languages is also a regular language.
How about the INTERSECTION of two languages?

\[ F = \{ (q_1, q_2) \mid q_1 \in F_1 \text{ AND } q_2 \in F_2 \} \]
Intersection Theorem for Regular Languages

Given two languages, $L_1$ and $L_2$, define the intersection of $L_1$ and $L_2$ as

$L_1 \cap L_2 = \{ w \mid w \in L_1 \text{ and } w \in L_2 \}$

Theorem: The intersection of two regular languages is also a regular language
Proof Idea: Again, run “in parallel” $M_1$ and $M_2$

$Q = \text{pairs of states, one from } M_1 \text{ and one from } M_2$

$=$ \{ $(q_1, q_2)$ | $q_1 \in Q_1$ and $q_2 \in Q_2$ \}

$= Q_1 \times Q_2$

$q_0 = (q_0^1, q_0^2)$

$F = \{ (q_1, q_2) | q_1 \in F_1 \text{ AND } q_2 \in F_2 \}$

$\delta((q_1, q_2), \sigma) = (\delta_1(q_1, \sigma), \delta_2(q_2, \sigma))$
Union Theorem for Regular Languages

The union of two regular languages is also a regular language

“Regular Languages are closed under union”

Intersection Theorem for Regular Languages

The intersection of two regular languages is also a regular language
Complement Theorem for Regular Languages

The complement of a regular language is also a regular language.

In other words, if A is regular than so is \( \overline{A} \),

where \( \overline{A} = \{ w \in \Sigma^* \mid w \notin A \} \)

Proof Idea?
The Reverse of a Language

Reverse of A:
\[ A^R = \{ w_1 \ldots w_k \mid w_k \ldots w_1 \in A, w_i \in \Sigma \} \]

If A is recognized by the usual kind of DFA, Then \( A^R \) is recognized by a “backwards” DFA that reads its strings from right to left!

Question: If A is regular, then is \( A^R \) also regular?

*Can every “Right-to-Left” DFA be replaced by a normal “Left-to-Right” DFA?*
Suppose M read its input from right to left...

Then $L(M) = \{ w \mid w \text{ ends with a } 1 \}$. Is this regular?
Reverse Theorem for Regular Languages

The reverse of a regular language is also a regular language!

“Regular Languages Are Closed Under Reverse”

*If* a language can be recognized by a DFA that reads its input from *right to left*, *then* there is an “normal” left-to-right DFA that accepts the same language

Counterintuitive! DFAs have finite memory...
Reversing DFAs?

Let $L$ be a regular language, let $M$ be a DFA that recognizes $L$

We want to build a machine $M^R$ that accepts $L^R$

$M$ accepts $w$ $\iff$ $w$ describes a directed path in $M$ from start to an accept state

Want: $M^R$ accepts $w^R$ $\iff$ $M$ accepts $w$

First Attempt:
Try to define $M^R$ as $M$ with the arrows reversed!
Turn start state into a final state,
turn final states into start states
Problem: $M^R$ IS NOT ALWAYS A DFA!

It could have many start states

Some states may have *more than one* transition for a given symbol, or it may have none at all!
What happens with 100?

We will say this new kind of machine accepts string $x$ if there is some path reading in $x$ that reaches some accept state from some start state.
Then, this machine recognizes: \( \{ w \mid w \text{ contains } 100 \} \)

We will say this new kind of machine accepts string \( x \) if there is some path reading in \( x \) that reaches some accept state from some start state.
Another Example of an NFA

At each state, we’ll allow any number (including zero) of out-arrows for letters $\sigma \in \Sigma$, including $\varepsilon$.

Set of strings accepted by this NFA = \{w | w contains a 0\}
Multiple Start States

We allow *multiple* start states for NFAs, and Sipser allows only one

Can easily convert NFA with many start states into one with a single start state:
A non-deterministic finite automaton (NFA) is a 5-tuple \( N = (Q, \Sigma, \delta, Q_0, F) \) where

\( Q \) is the set of states
\( \Sigma \) is the alphabet
\( \delta : Q \times \Sigma_{\varepsilon} \rightarrow 2^Q \) is the transition function
\( Q_0 \subseteq Q \) is the set of start states
\( F \subseteq Q \) is the set of accept states

\( 2^Q \) is the set of all possible subsets of \( Q \)
\( \Sigma_{\varepsilon} = \Sigma \cup \{\varepsilon\} \)
N = (Q, Σ, δ, Q₀, F)

Q = {q₁, q₂, q₃, q₄}
Σ = {0,1}
Q₀ = {q₁, q₂}
F = {q₄}

δ(q₂,1) = {q₄}  δ(q₄,1) = ∅
δ(q₃,1) = ∅
δ(q₁,0) = {q₃}

Set of strings accepted = {1,00,01}
Def. Let $w \in \Sigma^*$. Let $N$ be an NFA. $N$ accepts $w$ if there's a sequence of states $r_0, r_1, ..., r_k \in Q$ and $w$ can be written as $w_1 \cdots w_k$ with $w_i \in \Sigma \cup \{\varepsilon\}$ such that

1. $r_0 \in Q_0$
2. $r_i \in \delta(r_{i-1}, w_i)$ for all $i = 1, ..., k$, and
3. $r_k \in F$

$L(N)$ = the language recognized by $N$
= set of all strings that NFA $N$ accepts

A language $L'$ is recognized by an NFA $N$ if $L' = L(N)$. 
Are these equally powerful???
NFAs are generally simpler than DFAs

A DFA recognizing the language \{1\}

An NFA recognizing the language \{1\}
Theorem: For every NFA N, there is a DFA M such that \( L(M) = L(N) \)

Corollary: A language A is regular if and only if A is recognized by an NFA

Corollary: A is regular iff \( A^R \) is regular

left-to-right DFAs \( \equiv \) right-to-left DFAs
From NFAs to DFAs

Input: NFA $N = (Q, \Sigma, \delta, Q_0, F)$

Output: DFA $M = (Q', \Sigma, \delta', q_0', F')$

To learn if NFA $N$ accepts, we could do the computation of $N$ in parallel, maintaining the set of all possible states that can be reached.

Idea:

Set $Q' = 2^Q$
From NFAs to DFAs: Subset Construction

Input: NFA $N = (Q, \Sigma, \delta, Q_0, F)$

Output: DFA $M = (Q', \Sigma, \delta', q_0', F')$

$Q' = 2^Q$

$\delta' : Q' \times \Sigma \rightarrow Q'$

For $S \in Q'$, $\sigma \in \Sigma$:

$\delta'(S, \sigma) = \bigcup_{q \in S} \varepsilon(\delta(q, \sigma))$

$q_0' = \varepsilon(Q_0)$

$F' = \{ S \in Q' \mid f \in S \text{ for some } f \in F \}$

For $S \subseteq Q$, the $\varepsilon$-closure of $S$ is

$\varepsilon(S) = \{ r \in Q \text{ reachable from some } q \in S \text{ by taking zero or more } \varepsilon\text{-transitions} \}$
Example of the ε-closure

\[ \varepsilon(\{q_0\}) = \{q_0, q_1, q_2\} \]

\[ \varepsilon(\{q_1\}) = \{q_1, q_2\} \]

\[ \varepsilon(\{q_2\}) = \{q_2\} \]
Given: NFA $N = (\{1,2,3\}, \{a,b\}, \delta, \{1\}, \{1\})$

Construct: Equivalent DFA $M$

$M = (2^{\{1,2,3\}}, \{a,b\}, \delta', \{1,3\}, ...)$

$\varepsilon(\{1\}) = \{1,3\}$

$\varepsilon(\{1\}) = \{1,3\}$

$\{1\}, \{1,2\}?$

$\{1,2,3\}$