Announcements

Assignment 1 out today, due two week from today.
The plan for the next few lectures

For the next few lectures, we’ll start with the simplest models of language.

- DFAs (last time)
- NFAs and Regular languages (today)
- Finite state transducers (FSTs) and applications
Definition: DFA

A **deterministic finite automaton** is a 5-tuple \( M = \langle Q, \Sigma, \delta, q_0, F \rangle \) where

- \( Q \) is a finite set of states
- \( \Sigma \) is a finite alphabet
- \( \delta : Q \times \Sigma \rightarrow Q \) is the transition function
- \( q_0 \in Q \) is the start (initial) state
- \( F \subseteq Q \) is the set of final (accept) states

\( L(M) \subseteq \Sigma^* \) is the **language of** \( M \), i.e. the set of strings \( M \) accepts
DFAs

(q₀, 0, 1, 0, 1, 0, 1, 0, 1)

Graph:
- States: q₀, 0, 1
- Edges: (q₀, 0, q₀), (q₀, 1, 0), (0, 0, 1), (0, 1, 0), (1, 0, 1), (1, 1, q₀)

Transitions:
- q₀ on 0 goes to 0
- q₀ on 1 goes to 1
- 0 on 0 goes to 1
- 0 on 1 goes to 0
- 1 on 0 goes to 1
- 1 on 1 goes back to q₀
Acceptance

- M is a DFA with alphabet $\Sigma$

- An input sentence (word / string) $w$ is a sequence $w_1w_2w_3 \ldots w_n$ where each $w_i \in \Sigma$

- M accepts $w$ if after starting in $q_0$ and reading $w$ it ends in an accept state, i.e., if there is a sequence of states $s_0, s_1, \ldots, s_n$ such that

  $s_0 = q_0$

  $s_i = \delta(s_{i-1}, w_i)$

  $s_n \in F$
Regular Languages

A language $L \subseteq \Sigma^*$ is **regular** if there exists an FSA $M$ such that $L(M) = L$.
Regular Languages

A language \( L \subseteq \Sigma^* \) is regular if there exists an FSA \( M \) such that \( L(M) = L \)

Example regular languages:

\[
\begin{align*}
L &= \Sigma^* \\
L &= \emptyset \\
L &= \{ \varepsilon \}
\end{align*}
\]
Regular Languages

A language \( L \subseteq \Sigma^* \) is \textbf{regular} if there exists an FSA \( M \) such that \( L(M) = L \).

Example regular languages:

\[ L = \Sigma^* \]
\[ L = \emptyset \]
\[ L = \{ \varepsilon \} \]

Example non-regular languages (more later):

\[ L = \{ a^n b^n \mid n \geq 0 \} \]
\[ L = \{ w \mid w \text{ is a grammatical sentence of English} \} \]
Definition: DFA

A **deterministic finite automaton** is a 5-tuple $M = \langle Q, \Sigma, \delta, q_0, F \rangle$ where

- $Q$ is a finite **set of states**
- $\Sigma$ is a finite **alphabet**
- $\delta : Q \times Q \rightarrow \Sigma$ is the **transition function**
- $q_0 \in Q$ is the **start (initial) state**
- $F \subseteq Q$ is the **set of final (accept) states**

$L(M) \subseteq \Sigma^*$ is **the language of** $M$, i.e. the set of strings $M$ accepts
Definition: NFA

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

- \( Q \) is a finite **set of states**
- \( \Sigma \) is a finite **alphabet**
- \( \delta : Q \times Q \to \Sigma \) is a **transition function**
- \( \delta : Q \times \Sigma \to 2^Q \) is the **transition relation**
- \( q_0 \in Q \) is the **start (initial) state**
- \( F \subseteq Q \) is the **set of final (accept) states**
- \( L(M) \subseteq \Sigma^* \) is the **language of** \( M \), i.e. the set of strings \( M \) accepts
Definition: NFA

A **nondeterministic finite automaton** is a 5-tuple $M = \langle Q, \Sigma, \delta, q_0, F \rangle$ where

- $Q$ is a finite set of states
- $\Sigma$ is a finite alphabet
- $\delta : Q \times \Sigma \rightarrow 2^Q$ is the transition relation
- $q_0 \in Q$ is the start (initial) state
- $F \subseteq Q$ is the set of final (accept) states
- $L(M) \subseteq \Sigma^*$ is the language of $M$, i.e. the set of strings $M$ accepts

An NFA accepts a word $w$ if there exists a computation path that ends in a final state.
Remarks

- A DFA is required to have a **complete** transition function, whereas an NFA can have an **incomplete** transition function.

- This is useful for specifying simple language generating automata, e.g. “linear chain” automata.

\[ L(M) = \{aba\} \]
Example: NFA
Example: NFA

Computation on $w = abaa$
Example: NFA

Computation on $w = abaa$

Accept!
DFA & NFA Equivalence

**Theorem.** For every NFA $A$ there exists a DFA $A'$ s.t.
$L(A) = L(A')$. 
DFA & NFA Equivalence

**Theorem.** For every NFA \( A \) there exists a DFA \( A' \) s.t. \( L(A) = L(A') \).

**Proof.** *This is a constructive proof.* That is, given an NFA \( A = \langle Q, \Sigma, \delta, q_0, F \rangle \) we construct a DFA \( A' = \langle Q', \Sigma', \delta', q'_0, F' \rangle \) s.t. \( L(A) = L(A') \)
DFA & NFA Equivalence

Theorem. For every NFA $A$ there exists a DFA $A'$ s.t. $L(A) = L(A')$.

Proof. This is a constructive proof. That is, given an NFA $A = \langle Q, \Sigma, \delta, q_0, F \rangle$ we construct a DFA $A' = \langle Q', \Sigma', \delta', q'_0, F' \rangle$ s.t. $L(A) = L(A')$

Powerset construction.

$\Sigma' = \Sigma$

$Q' = 2^Q$ Constructed DFA states are sets of NFA states

$q'_0 = \{q_0\}$

$F' = \{A \in Q' \mid A \cap F \neq \emptyset\}$
DFA & NFA Equivalence

Powerset construction.

\[ \Sigma' = \Sigma \]

\[ Q' = 2^Q \]

Constructed DFA states are sets of NFA states

\[ q_0' = \{ q_0 \} \]

\[ F' = \{ A \in Q' \mid A \cap F \neq \emptyset \} \]
DFA & NFA Equivalence

Powerset construction.

\[ \Sigma' = \Sigma \]

\[ Q' = 2^Q \quad \text{Constructed DFA states are sets of NFA states} \]

\[ q'_0 = \{q_0\} \]

\[ F' = \{ A \in Q' \mid A \cap F \neq \emptyset \} \]

Transition function.
DFA & NFA Equivalence

Powerset construction.

$\Sigma' = \Sigma$

$Q' = 2^Q$  Constructed DFA states are sets of NFA states

$q'_0 = \{q_0\}$

$F' = \{A \in Q' \mid A \cap F \neq \emptyset\}$

Transition function.

$\delta'(\emptyset, \sigma) = \emptyset \ \forall \sigma \in \Sigma'$  “Failure state”
DFA & NFA Equivalence

Powerset construction.

\[ \Sigma' = \Sigma \]
\[ Q' = 2^Q \text{ Constructed DFA states are sets of NFA states} \]
\[ q'_0 = \{q_0\} \]
\[ F' = \{A \in Q' | A \cap F \neq \emptyset\} \]

Transition function.

\[ \delta'(\emptyset, \sigma) = \emptyset \forall \sigma \in \Sigma' \text{ “Failure state”} \]
\[ \delta'(\{q_1, q_2, \ldots, q_i\}, \sigma) = \bigcup_{q \in \{q_1, q_2, \ldots, q_i\}} \delta(q, \sigma) \]
Example
Example
DFA & NFA Equivalence

It remains to show:

\[ A' \text{ is DFA} \quad L(A) = L(A') \]
DFA & NFA Equivalence

It remains to show:

\[ A' \text{ is DFA} \quad L(A) = L(A') \]

**Lemma.** For every \( w \in \Sigma^* \),

\[ \delta'(q'_0, w) = \{p_1, p_2, \ldots, p_k\} \iff \delta(q_0, w) = \{p_1, p_2, \ldots, p_k\} \]
DFA & NFA Equivalence

It remains to show:

\[ A' \text{ is DFA} \quad \quad L(A) = L(A') \]

**Lemma.** For every \( w \in \Sigma^* \),

\[ \delta'(q_0', w) = \{p_1, p_2, \ldots, p_k\} \quad \iff \quad \delta(q_0, w) = \{p_1, p_2, \ldots, p_k\} \]

**Proof of lemma.** By induction on \( |w| \)
DFA & NFA Equivalence

It remains to show:

\[ A' \text{ is DFA} \quad L(A) = L(A') \]

Lemma. For every \( w \in \Sigma^* \),

\[ \delta'(q'_0, w) = \{p_1, p_2, \ldots, p_k\} \iff \delta(q_0, w) = \{p_1, p_2, \ldots, p_k\} \]

Proof of lemma. By induction on \( |w| \)

(i) base: \( |w| = 0 \quad w = \varepsilon : \delta'(q'_0, \varepsilon) = \{q_0\} \)

\[ \delta(q_0, \varepsilon) = \{q_0\} \]
DFA & NFA Equivalence

It remains to show:

\[ A' \text{ is DFA} \quad \Rightarrow \quad L(A) = L(A') \]

Lemma. For every \( w \in \Sigma^* \),

\[ \delta'(q_0', w) = \{p_1, p_2, \ldots, p_k\} \quad \iff \]

\[ \delta(q_0, w) = \{p_1, p_2, \ldots, p_k\} \]

Proof of lemma. By induction on \( |w| \)

(i) base: \( |w| = 0 \quad \Rightarrow \quad w = \varepsilon : \quad \delta'(q_0', \varepsilon) = \{q_0\} \)

\[ \delta(q_0, \varepsilon) = \{q_0\} \]

(ii) hypothesis: assume true for all \( w \) s.t. \( |w| \leq n \).
DFA & NFA Equivalence

**Lemma.** For every $w \in \Sigma^*$,

$$
\delta'(q_0, w) = \{p_1, p_2, \ldots, p_k\} \iff
\delta(q_0, w) = \{p_1, p_2, \ldots, p_k\}
$$

(ii) hypothesis: assume true for all $w$ s.t. $|w| \leq n$. 

Lemma. For every $w \in \Sigma^*$,

$$\delta'(q_0, w) = \{p_1, p_2, \ldots, p_k\} \iff \delta(q_0, w) = \{p_1, p_2, \ldots, p_k\}$$

(ii) hypothesis: assume true for all $w$ s.t. $|w| \leq n$.

(iii) inductive step: let $w = x\sigma$ s.t. $|w| = n + 1$
Lemma. For every $w \in \Sigma^*$,

\[
\delta'(q'_0, w) = \{p_1, p_2, \ldots, p_k\} \iff \\
\delta(q_0, w) = \{p_1, p_2, \ldots, p_k\}
\]

(ii) hypothesis: assume true for all $w$ s.t. $|w| \leq n$.

(iii) inductive step: let $w = x\sigma$ s.t. $|w| = n + 1$

\[
\delta'(q'_0, w) = \delta'(\{q_0\}, x\sigma) = \delta'(\delta'(\{q_0\}, x), \sigma)
\]
Lemma. For every $w \in \Sigma^*$,
\[
\delta'(q'_0, w) = \{p_1, p_2, \ldots, p_k\} \iff \\
\delta(q_0, w) = \{p_1, p_2, \ldots, p_k\}
\]

(ii) hypothesis: assume true for all $w$ s.t. $|w| \leq n$.

(iii) inductive step: let $w = x\sigma$ s.t. $|w| = n + 1$
\[
\delta'(q'_0, w) = \delta'({q_0}, x\sigma) = \delta'(\delta'({q_0}, x), \sigma)
\]

use inductive hypothesis:
\[
\delta'({q_0}, x) = \{p_1, p_2, \ldots, p_k\} \iff \delta(q_0, x) = \{p_1, p_2, \ldots, p_k\}
\]
**DFA & NFA Equivalence**

**Lemma.** For every \( w \in \Sigma^* \),

\[
\delta'(q'_0, w) = \{p_1, p_2, \ldots, p_k\} \iff \\
\delta(q_0, w) = \{p_1, p_2, \ldots, p_k\}
\]

(ii) hypothesis: assume true for all \( w \) s.t. \(|w| \leq n\).

(iii) inductive step: let \( w = x\sigma \) s.t. \(|w| = n + 1\)

\[
\delta'(q'_0, w) = \delta'(\{q_0\}, x\sigma) = \delta'(\delta'(\{q_0\}, x), \sigma)
\]

use inductive hypothesis:

\[
\delta'(\{q_0\}, x) = \{p_1, p_2, \ldots, p_k\} \iff \delta(q_0, x) = \{p_1, p_2, \ldots, p_k\}
\]

\[
\delta'(\{p_1, p_2, \ldots, p_k\}, \sigma) = \{r_1, r_2, \ldots, r_j\} \iff \delta(\{p_1, p_2, \ldots, p_k\}, \sigma) = \{r_1, r_2, \ldots, r_j\}
\]
Lemma. For every $w \in \Sigma^*$,
\[
\delta'(q'_0, w) = \{p_1, p_2, \ldots, p_k\} \iff \\
\delta(q_0, w) = \{p_1, p_2, \ldots, p_k\}
\]
(ii) hypothesis: assume true for all $w$ s.t. $|w| \leq n$.
(iii) inductive step: let $w = x\sigma$ s.t. $|w| = n + 1$
\[
\delta'(q'_0, w) = \delta'(\{q_0\}, x\sigma) = \delta'(\delta'(\{q_0\}, x), \sigma)
\]
use inductive hypothesis:
\[
\delta'(\{q_0\}, x) = \{p_1, p_2, \ldots, p_k\} \iff \delta(q_0, x) = \{p_1, p_2, \ldots, p_k\}
\]
\[
\delta'\{p_1, p_2, \ldots, p_k\}, \sigma\} = \{r_1, r_2, \ldots, r_j\} \iff \delta\{p_1, p_2, \ldots, p_k\}, \sigma\} = \{r_1, r_2, \ldots, r_j\}
\]
\[
\delta'(q'_0, w) = \{r_1, r_2, \ldots, r_j\} = \delta(q_0, w) \blacksquare
DFA & NFA Equivalence

It still remains to show:

\[ A' \text{ is DFA} \quad L(A) = L(A') \]
DFA & NFA Equivalence

It still remains to show:

\[ A' \text{ is DFA} \quad \implies \quad L(A) = L(A') \]

**Proof of theorem.**

By definition, \( A' \) is a complete function, so it is a DFA
DFA & NFA Equivalence

It still remains to show:

\[ A' \text{ is DFA} \quad L(A) = L(A') \]

Proof of theorem.
By definition, \( A' \) is a complete function, so it is a DFA.
From the lemma,

\[ \delta'(q'_0, w) = \{ p_1, p_2, \ldots, p_j \} \iff \delta(q_0, w) = \{ p_1, p_2, \ldots, p_j \} \]
DFA & NFA Equivalence

It still remains to show:

\[ A' \text{ is DFA} \quad L(A) = L(A') \]

Proof of theorem.
By definition, \( A' \) is a complete function, so it is a DFA

From the lemma,
\[
\delta'(q'_0, w) = \{p_1, p_2, \ldots, p_j\} \iff \delta(q_0, w) = \{p_1, p_2, \ldots, p_j\}
\]
\[
\{p_1, p_2, \ldots, p_j\} \in F' \iff \{p_1, p_2, \ldots, p_j\} \cap F \neq \emptyset
\]
DFA & NFA Equivalence

It still remains to show:

\[ A' \text{ is DFA} \quad L(A) = L(A') \]

**Proof of theorem.**

By definition, \( A' \) is a complete function, so it is a DFA

From the lemma,

\[ \delta'(q_0', w) = \{p_1, p_2, \ldots, p_j\} \iff \delta(q_0, w) = \{p_1, p_2, \ldots, p_j\} \]

\[ \{p_1, p_2, \ldots, p_j\} \in F' \iff \{p_1, p_2, \ldots, p_j\} \cap F \neq \emptyset \]

thus,

\[ w \in L(A') \iff w \in L(A) \]
DFA & NFA Equivalence

It still remains to show:

\[ A' \text{ is DFA} \quad L(A) = L(A') \]

**Proof of theorem.**

By definition, \( A' \) is a complete function, so it is a DFA.

From the lemma,

\[ \delta'(q_0', w) = \{p_1, p_2, \ldots, p_j\} \iff \delta(q_0, w) = \{p_1, p_2, \ldots, p_j\} \]
\[ \{p_1, p_2, \ldots, p_j\} \in F' \iff \{p_1, p_2, \ldots, p_j\} \cap F \neq \emptyset \]

thus, \( w \in L(A') \iff w \in L(A) \)

so, \( L(A') = L(A) \) \( \blacksquare \)
Remarks

• NFAs therefore only accept regular languages

• NFAs and DFAs can be used interchangeably
(3 min break)
A NFA with \( \varepsilon \)-transitions is an NFA that may change states without reading an input symbol.
A **nondeterministic finite automaton** is a 5-tuple $M = \langle Q, \Sigma, \delta, q_0, F \rangle$ where

- $Q$ is a finite **set of states**
- $\Sigma$ is a finite **alphabet**
- $\delta : Q \times \Sigma \rightarrow 2^Q$ is the **transition relation**
- $\delta : Q \times \Sigma \cup \{\varepsilon\} \rightarrow 2^Q$ is the **transition relation**
- $q_0 \in Q$ is the **start (initial) state**
- $F \subseteq Q$ is the **set of final (accept) states**
- $L(M) \subseteq \Sigma^*$ is the **language of** $M$, i.e. the set of strings $M$ accepts
NFA& \( \varepsilon \)-NFA Equivalence

**Theorem.** For every NFA \( A \) with epsilon moves there is an equivalent NFA \( A' \) without, s.t. \( L(A) = L(A') \)
Regular Expression

A regular expression is a way of describing the languages accepted by FSAs.

Defined recursively:
1. $\emptyset$ is an RE denoting the empty set
2. $\varepsilon$ is an RE denoting the set $\{\varepsilon\}$
3. for each $a \in \Sigma$, $a$ is a RE denoting $\{a\}$
4. If $r$ and $s$ are REs denoting the languages $R$ and $S$
   - $(r|s)$ denotes $R \cup S$
   - $(rs)$ denotes $R.S$
   - $r^*$ denotes $R^*$

Precedence means parentheses can sometimes be omitted: $\ast\ast\mid$
Examples

\((0\mid 1)^*\) denotes all finite words over \(\Sigma = \{0, 1\}\)

\(0^*\mid 1^*\) denotes all finite words containing only 0’s and 1’s
REs and $\varepsilon$-NFAs

**Theorem.** For every RE $r$ there is an $\varepsilon$-NFA $A$ s.t. $L(r) = L(A)$
REs and $\varepsilon$-NFAs

**Theorem.** For every RE $r$ there is an $\varepsilon$-NFA $A$ s.t. $L(r) = L(A)$

**Proof.** We will construct $A$ compositionally using induction on the number of operators in $r$. 
REs and $\varepsilon$-NFAs

**Theorem.** For every RE $r$ there is an $\varepsilon$-NFA $A$ s.t.
$L(r) = L(A)$

**Proof.** We will construct $A$ compositionally using induction on the number of operators in $r$.

**Base cases.** $r$ has 0 operators
REs and $\varepsilon$-NFAs

**Theorem.** For every RE $r$ there is an $\varepsilon$-NFA s.t.

$L(r) = L(A)$

**Proof.** We will construct $A$ compositionally using induction on the number of operators in $r$.

**Base cases.** $r$ has 0 operators

$r = \emptyset \quad q_0 \quad q_f$
REs and $\varepsilon$-NFAs

**Theorem.** For every RE $r$ there is an $\varepsilon$-NFA $A$ s.t. $L(r) = L(A)$

**Proof.** We will construct $A$ compositionally using induction on the number of operators in $r$.

**Base cases.** $r$ has 0 operators

$r = \emptyset$

$r = \varepsilon$

![Diagram](image.png)
**REs and $\varepsilon$-NFAs**

**Theorem.** For every RE $r$ there is an $\varepsilon$-NFA $A$ s.t. $L(r) = L(A)$

**Proof.** We will construct $A$ compositionally using induction on the number of operators in $r$.

**Base cases.** $r$ has 0 operators

$r = \emptyset$

$r = \varepsilon$

$r = a$
REs and $\varepsilon$-NFAs

**Theorem.** For every RE $r$ there is an $\varepsilon$-NFA $A$ s.t. $L(r) = L(A)$

**Proof.** We will construct $A$ compositionally using induction on the number of operators in $r$.

**Base cases.** $r$ has 0 operators

- $r = \emptyset$
  - $q_0 \quad q_f$
  - Note: we assume there is exactly one final state.

- $r = \varepsilon$
  - $q_0 \xrightarrow{\varepsilon} q_f$

- $r = a$
  - $q_0 \xrightarrow{a} q_f$
REs and $\varepsilon$-NFAs

**Inductive step.** We assume hypothesis is true for all REs with $\leq n$ operations, and then prove is true for $n+1$ operations.
REs and $\varepsilon$-NFAs

**Inductive step.** We assume hypothesis is true for all REs with $\leq n$ operations, and then prove is true for $n+1$ operations.

There are three cases to be dealt with:

1. $r = r_1 | r_2$
2. $r = r_1 r_2$
3. $r = r_1^*$
Case 1: \( r = r_1 \mid r_2 \)

By the inductive hypothesis, there are two epsilon NFAs \( A_1 \) and \( A_2 \).

\[ A_1 \rightarrow q_{01} \rightarrow A_1 \]

\[ q_{f1} \rightarrow \]

\[ A_2 \rightarrow q_{01} \rightarrow A_2 \]
Case 1:  \( r = r_1 \mid r_2 \)

By the inductive hypothesis, there are two epsilon NFAs \( A_1 \) and \( A_2 \).

\[ \begin{align*}
A_1 & \quad q_{01} \quad q_{f1} \\
A_2 & \quad q_{02} \quad q_{f2}
\end{align*} \]
Case 1: $r = r_1 \mid r_2$

By the inductive hypothesis, there are two epsilon NFAs $A_1$ and $A_2$. **Construct the following $A$.**
Case 1: \( r = r_1 \mid r_2 \)

Formally, if

\[
A_1 = \langle Q_1, \Sigma, \delta_1, q_{01}, \{q_{f1}\} \rangle \\
A_2 = \langle Q_2, \Sigma, \delta_2, q_{02}, \{q_{f2}\} \rangle
\]

then,

\[
A = \langle Q_1 \cup Q_2 \cup \{q_0\} \cup \{q_f\}, \Sigma, \delta, q_0, \{q_f\} \rangle
\]

\[
\delta(q_0, \varepsilon) = \{q_{01}, q_{02}\}
\]

\[
\delta(q_{f1}, \varepsilon) = \{q_f\}
\]

\[
\delta(q_{f2}, \varepsilon) = \{q_f\}
\]

\[
\delta(q, \sigma) = \delta_1 q, \sigma \quad \forall q \in Q_1 - \{q_{f1}\}, \sigma \in \Sigma \cup \{\varepsilon\}
\]

\[
\delta(q, \sigma) = \delta_2 q, \sigma \quad \forall q \in Q_2 - \{q_{f2}\}, \sigma \in \Sigma \cup \{\varepsilon\}
\]
Case 1: \( r = r_1 \mid r_2 \)

It remains to show that \( L(A) = L(A_1) \cup L(A_2) \)

How to do this? Set containment.
Cases 2 & 3

- Strategy for showing this proceeds as with Case 1
- Refer to textbook for details.