Lecture Outline

NFA

NFA computation
Equivalence of DFAs and NFAs
$\varepsilon$-closure
Applications
Regular Languages and Operations
Clean NFA
Nondeterminism

In a DFA, the next state is completely determined by the current state and the symbol being scanned, according to the transition function $\delta$.

In a nondeterministic finite automaton (NFA), there may be a choice of several states to transition into at each step, and we may decide not to advance the read head.

The machine accepts the input iff there is some series of choices that can lead to an accepting state having read the entire input.

Transition diagram may have any number of edges with the same label leaving the same state.
There can also be $\epsilon$-edges (meaning: make the transition without reading the symbol or advancing the read head).
Consider example of an NFA accepting all strings that end in 01:
NFA example

Consider example of an NFA accepting all strings that end in 01:

![NFA diagram]

The states of this NFA during the processing of input sequence 00101 change as follows:

```
q₀ → q₀ → q₀ → q₀ → q₀ → q₀ → q₀
|     |     |     |     |     |     |
| q₁   q₁   q₁   q₁   q₁   q₁   q₁ |
|     |     |     |     |     |     |
| q₂ (stuck) q₂ (stuck) |
```

0 0 1 0 1
Given an alphabet $\Sigma$, we define

$$\Sigma_{\varepsilon} = \Sigma \cup \{\varepsilon\},$$

which is the set of all strings over $\Sigma$ of length 0 or 1.

**Definition**

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

- $Q$ is a finite set (the set of states),
- $\Sigma$ is a finite set (the alphabet),
- $\delta : Q \times \Sigma_{\varepsilon} \rightarrow 2^Q$ is the transition function,
- $q_0 \in Q$ is the start state, and
- $F \subseteq Q$ is the set of accepting states.
Example of an NFA that recognizes the language of strings over \( \{a, b\} \) that start with \( b \) and contain either \( aaa \) or \( abbaa \) as a substring:

Q: There is no outgoing edge labeled \( a \) from the node 0. What does that mean? What is the value of \( \delta(0, a) \)?
Example of an NFA that recognizes the language of decimal integers:
NFA computation

Notice that $\delta(q, a)$ is now a set of states instead of just a single state. It is the set of possible successor states of $q$ reading $a$.

In particular, $\delta(q, \varepsilon)$ is also defined to be a set of states. This is the set of possible successors to $q$ via an $\varepsilon$-transition. Such a transition, when taken, does not advance the read head.

Given an NFA $N$ and an input string $w$, there may now be many possible computations (or computation paths) of $N$ on $w$. We say that $N$ accepts $w$ iff at least one of these computation paths ends in a final state after reading all of $w$. Informally, $N$ accepts if there is a correct set of transition “guesses” that leads to an accepting state and reads the entire input.
NFA computation

Definition
Let $N = (Q, \Sigma, \delta, q_0, F)$ be an NFA and let $w \in \Sigma^*$ be a string. A complete computation of $N$ on input $w$ is a pair of tuples $((s_0, \ldots, s_m), (y_1, \ldots, y_m))$ where

- $s_i \in Q$ for all $0 \leq i \leq m$,
- $y_i \in \Sigma^\epsilon$ for all $1 \leq i \leq m$,
- $w = y_1 \cdots y_m$,
- $s_0 = q_0$, and
- $s_i \in \delta(s_{i-1}, y_i)$ for all $1 \leq i \leq m$.

The complete computation above is accepting iff $s_m \in F$; otherwise, it is rejecting. We say that $M$ accepts $w$ if there exists some accepting complete computation of $M$ on $w$.

A complete computation represents a legal path through the NFA while reading the entire input; $\epsilon$-transitions correspond to steps where $y_i = \epsilon$.

Q: What can you say about $|w|$ and $m$?
As with DFAs before, we define the language $L(N)$ recognized by NFA $N$ to be the set of all input strings accepted by $N$.

We say that two automata (DFAs, NFAs, or NFAs) are equivalent iff they recognize the same language.

Our goal is to show that DFAs and NFAs are equally powerful models of computation, that is, they recognize the same class of languages, namely, the regular languages. We show this in two steps:

1. For every DFA, there is an equivalent NFA.
2. For every NFA, there is an equivalent DFA.
From DFA to NFA

This direction is obvious:

Q: Why?
From DFA to NFA

This direction is obvious:

For any DFA $D$, we build an equivalent NFA $N$ with the same transition diagram, i.e., each $\delta_N(q, a)$ is a singleton for $a \in \Sigma$, and $\delta_N(q, \epsilon) = \emptyset$. More formally, if $D = (Q, \Sigma, \delta_D, q_0, F)$ is a DFA, we define the NFA $N = (Q, \Sigma, \delta_N, q_0, F)$, where

$$
\delta_N(q, a) = \begin{cases} 
\{ \delta_D(q, a) \} & \text{if } a \in \Sigma, \\
\emptyset & \text{if } a = \epsilon.
\end{cases}
$$

It is easy to see that $L(N) = L(D)$, i.e., that $N$ is equivalent to $D$. 
From NFA to DFA

Theorem
For every NFA, there is an equivalent DFA.

We give a sketch of a proof, which includes the construction of the equivalent DFA but does not verify that it is correct.

The idea of the proof is that, as we simulate a given NFA, we keep track of the set of all states we could be in after reading each successive symbol of the input.
For a given NFA \( N \), we will construct an equivalent DFA \( D \).

Let \( w = xa \) where \( a \) is the last symbol of \( w \). If \( S \) is the set of all states of \( N \) we could possibly get to by reading \( x \), then the set \( S' \) of states of \( N \) that we could possibly be in after reading \( a \) (i.e., reading whole \( w \)) only depends on \( S \) and \( a \), and can be computed from the transition diagram of \( N \).

We make each possible set \( S \) of states in \( N \) a single state of the DFA \( D \), and the transitions of \( D \) correspond to shifting the set of possible states of \( N \) upon reading each symbol. For example, the transition \( S \xrightarrow{a} S' \) would be a transition in \( D \).

There are only finitely many subsets of the states of \( N \), so \( D \) is really a deterministic finite automaton.
From NFA to DFA: eliminating $\varepsilon$-transitions

Notice that $N$ may have $\varepsilon$-transitions that need to be taken into account. So, for a state $S$ in $D$ and $a \in \Sigma$, to define the target state $S'$ in a transition $S \xrightarrow{a} S'$, we first follow any $a$-transitions from states in $S$, and then follow any $\varepsilon$-transitions thereafter (possibly several in a row).

Q: Should we also follow any $\varepsilon$-transitions before the $a$-transitions?
From NFA to DFA: eliminating $\varepsilon$-transitions

Notice that $N$ may have $\varepsilon$-transitions that need to be taken into account. So, for a state $S$ in $D$ and $a \in \Sigma$, to define the target state $S'$ in a transition $S \xrightarrow{a} S'$, we first follow any $a$-transitions from states in $S$, and then follow any $\varepsilon$-transitions thereafter (possibly several in a row).

- The start state of $D$ corresponds to the set of states of $N$ that one could possibly be in before reading any input symbols. This is exactly the set of states reachable from the start state of $N$ via $\varepsilon$-transitions only.

- A set $S$ of states of $N$ constitutes an accepting state of $D$ iff it includes at least one accepting state of $N$, i.e., the computation could possibly end up in a accepting state of $N$. 
Suppose that the given NFA $N = (Q, \Sigma, \delta, q_0, F)$. For any set of states $S \subseteq Q$, we define the $\varepsilon$-closure of $S$, denoted $\text{ECLOSE}(S)$, to be the set of states reachable from $S$ via zero or more $\varepsilon$-transitions in a row.

Formally, for $i \geq 0$ we define $E_i(S)$ inductively to be the set of states reachable from $S$ via exactly $i$ many $\varepsilon$-transitions, so that:

- $E_0(S) = S$, and
- for all $i \geq 0$,

$$E_{i+1}(S) = \{r \in Q \mid (\exists q \in E_i(S) : r \in \delta(q, \varepsilon))\} = \bigcup_{q \in E_i(S)} \delta(q, \varepsilon).$$

This lets us define

$$\text{ECLOSE}(S) = \bigcup_{i \geq 0} E_i(S),$$

i.e., the set of all states reachable from $S$ via zero or more $\varepsilon$-transitions.
Each state of this NFA is its own \( \varepsilon \)-closure, with two exceptions:

- \( \text{ECLOSE}(\{q_0\}) = \{q_0, q_1\} \)
- \( \text{ECLOSE}(\{q_3\}) = \{q_3, q_5\} \)
ε-closure properties

- State $z$ is in $\text{ECLOSE}(S)$ if and only if there exists a sequence of zero or more ε-transitions starting at some state $q_1 \in S$ and ending at $z$:

  $q_1 \xrightarrow{\varepsilon} q_2 \xrightarrow{\varepsilon} \cdots \xrightarrow{\varepsilon} q_k \xrightarrow{\varepsilon} z$.

- Any set $S$ is a subset of $\text{ECLOSE}(S)$, i.e., $S \subseteq \text{ECLOSE}(S)$.

- ε-closure is closed under ε-transitions, i.e., there are no ε-transitions from inside $\text{ECLOSE}(S)$ to outside $\text{ECLOSE}(S)$:

  $\text{ECLOSE}(\text{ECLOSE}(S)) = \text{ECLOSE}(S)$.

- ε-closure is distributive w.r.t. union:

  $\text{ECLOSE}(S_1 \cup S_2) = \text{ECLOSE}(S_1) \cup \text{ECLOSE}(S_2)$. 
From NFA to DFA: formal construction

Given an NFA \( N = (Q_N, \Sigma, \delta_N, q_0, F_N) \), an equivalent DFA can be constructed as \( D = (Q_D, \Sigma, \delta_D, S_0, F_D) \), where

- \( Q_D = 2^{Q_N} \) is the powerset of \( Q_N \),
- \( S_0 = \text{ECLOSE}(\{q_0\}) \), the set of states reachable from \( q_0 \) via \( \varepsilon \)-transitions,
- \( F_D = \{ S \subseteq Q_N \mid S \cap F_N \neq \emptyset \} \), the set of sets of states of \( N \) which contain at least one state from \( F_N \), and
- for every \( S \subseteq Q_N \) and \( a \in \Sigma \),

\[
\delta_D(S, a) = \text{ECLOSE} \left( \bigcup_{q \in S} \delta_N(q, a) \right).
\]

From the above definition, every state of \( D \) that is reachable from the start state \( S_0 \) is closed under \( \varepsilon \)-transitions, so there is no need to follow \( \varepsilon \)-transitions before \( a \)-transitions in the definition of \( \delta_D \).
When an NFA is converted to an equivalent DFA, often not all the possible sets of states of the NFA are used, so not all the DFA states are reachable from the start state, and they can be eliminated (e.g., any state sets which are not closed under $\varepsilon$-transitions).

As a practical matter, instead of including all possible subsets of states of the NFA, it is better to “grow” the DFA states from the start state by following transitions. We will usually wind up with a lot fewer DFA states this way.

To construct the equivalent DFA, we build its states one by one, starting with its start state, then following transitions. Whenever we encounter a new set of states of the NFA, we add it as a new state to the DFA.
From NFA to DFA: an Algorithm

Input: NFA $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$
Output: equivalent DFA $D = (Q_D, \Sigma, \delta_D, S_0, F_D)$

Set $\delta_D := \emptyset$ (no transitions yet)
Set $F_D := \emptyset$ (no final states yet)
Set $S_0 := \text{ECLOSE}(\{q_0\})$
Set $Q_D := \{S_0\}$

While there is an unmarked $S \in Q_D$, do
  For every $a \in \Sigma$, do
    Let $T$ be the set of states reachable from states in $S$ via
    an $a$-transition followed by any number of $\varepsilon$-transitions
    Add the transition $\delta_D(S, a) = T$ to $\delta_D$
    If $T \notin Q_D$ then
      $Q_D := Q_D \cup \{T\}$
      If $T \cap F_N \neq \emptyset$, then $F_D := F_D \cup \{T\}$
  End-for
  Mark $S$ as “processed”
End-while
Return $D = (Q_D, \Sigma, \delta_D, S_0, F_D)$
Let us apply this algorithm to the NFA that recognizes the language of strings over \( \{a, b\} \) that start with \( b \) and contain either \( aaa \) or \( abbaa \) as a substring:

The start state of the DFA is \( S_0 = \text{ECLOSE}(\{0\}) = \{0\} \).

Compute the transitions from \( S_0 \). First the \( a \)-transition: there is nothing reachable from state 0 following \( a \), so the new state of the DFA is \( \emptyset \) and we have \( \delta_D(S_0, a) = \emptyset \), which is a new state, so we call it \( S_1 \) and add it to \( Q \). So, we now have \( Q = \{S_0, S_1\} \), where \( S_1 = \emptyset \), and the diagram looks like this:
Now consider $b$-transition from the state 0. The only state we can reach is 1 and $\text{ECLOSE}([1]) = \{1\}$, so we have a new state $S_2 = \{1\}$ and set $\delta_D(S_0, b) = S_2$.

How about transitions from $S_1$? $S_1$ has no states of the NFA, so we cannot get anywhere from $S_1$ following any symbol. Thus

$$\delta_D(S_1, a) = \delta_D(S_1, b) = \emptyset = S_1,$$

and we should add two self-loops from $S_1$ to itself. The diagram becomes:

$S_0$ and $S_1$ are both processed at this point.
Now we focus on $S_2 = \{1\}$. Following $a$ from state 1 of the NFA, we can get to states 1 and 2, then following the $\varepsilon$-transition from 2 we can get to 4. Thus we have a new state $S_3 = \{1, 2, 4\} = \delta_D(S_2, a)$. Following $b$ from state 1, we can only get back to state 1, so $\delta_D(S_2, b) = \{1\} = S_2$, and we put a self-loop $S_2 \xrightarrow{b} S_2$. Here is the new diagram:
Now we follow transitions from $S_3 = \{1, 2, 4\}$. For the $a$-transition, the NFA has transitions $1 \xrightarrow{a} 1$, $1 \xrightarrow{a} 2$, and $4 \xrightarrow{a} 5$. In addition, we can get from 2 to 4 again by following $\varepsilon$. So we have a new state $S_4 = \{1, 2, 4, 5\} = \delta_D(S_3, a)$. For the $b$-transition, the NFA has $1 \xrightarrow{b} 1$ and $2 \xrightarrow{b} 3$, with no additional $\varepsilon$-moves. So we have a new state $S_5 = \{1, 3\} = \delta_D(S_3, b)$, and the diagram for the DFA is now
Following $a$ from $S_4 = \{1, 2, 4, 5\}$, the NFA has $a$-transitions to states 1, 2, 5, and 6, with an $\varepsilon$-move to 4. So we get a new state $S_6 = \{1, 2, 4, 5, 6\} = \delta_D(S_4, a)$. Since $S_6 \cap F_N = \{6\} \neq \emptyset$, $S_6$ is our first final state. Following $b$ from states in $S_4$, we can reach states 1 and 3 and no others, and there are no additional $\varepsilon$-transitions possible. Since $S_5 = \{1, 3\}$, we have $\delta_D(S_4, b) = S_5$. 
We continue this way until each state of the DFA has a single outgoing edge for every alphabet symbol. The complete DFA looks like this:

Note that this is not an optimal solution. For example, all the accepting states can be coalesced into a single accepting state.
Applications of DFA/NFA for text search

Suppose we are given a set of words, call *keywords*, and we want to find occurrences of any of keywords in the text. We can employ a simple NFA that recognizes a set of keywords:

1. There is a start state with a transition to itself on every input symbol. Intuitively, the start state represents a “guess” that we have not yet begun to see any of the keywords, even if we have seen some of their letters.

2. For each keyword $a_1a_2\ldots a_k$, there are $k$ states, say, $q_1, q_2, \ldots, q_k$. There is a transition from start state to $q_1$ on symbol $a_1$, a transition from $q_1$ to $q_2$ on symbol $a_2$, and so on. The state $q_k$ is an accepting state and indicates that the keyword $a_1a_2\ldots a_k$ has been found.
Keyword search: example

The following NFA recognizes occurrences of the keywords WEB and EBAY:
There are two major choices for an implementation of NFA:

1. Write a program that simulates NFA by computing the set of states in reachable after reading each input symbol.

2. Convert the NFA into an equivalent DFA. Then simulate the DFA directly.
Regular Languages revisited

Recall the definition:

**Definition**
A language $A \subseteq \Sigma^*$ is regular iff some DFA recognizes it, i.e., $A = L(D)$ for some DFA $D$.

As we proved earlier, NFAs has the same computational power as DFAs. We therefore can restate the definition:

**Definition**
A language $A \subseteq \Sigma^*$ is regular iff some NFA recognizes it, i.e., $A = L(N)$ for some NFA $N$. 
Let $A$ and $B$ be two languages. We define the following *regular operations*:

**Union:** $A \cup B = \{ x \mid x \in A \lor x \in B \}$

**Concatenation:** $AB = A.B = A \circ B = \{ xy \mid x \in A \land y \in B \}$

**Complement:** $\overline{A} = \Sigma^* \setminus A$, the *complement* of $A$ (in $\Sigma^*$). This is the set of all strings over $\Sigma$ that are not in $A$.

**Theorem**

*If $A$ is a regular language, then so is $\overline{A}$. If $A$ and $B$ are regular languages then so are $A \cup B$ and $A \circ B$.*

In other words, the class of regular languages is *closed under* the union, concatenation, and complement operations.
Complement is regular

Let us prove that if $A$ is regular, then so is $\overline{A}$.

Suppose that $A$ is recognized by a DFA $D = (Q, \Sigma, \delta, q_0, F)$. What would be a DFA recognizing $\overline{A}$.
Complement is regular

Let us prove that if $A$ is regular, then so is $\overline{A}$.

Suppose that $A$ is recognized by a DFA $D = (Q, \Sigma, \delta, q_0, F)$. What would be a DFA recognizing $\overline{A}$.

It is easy to see that the DFA $(Q, \Sigma, \delta, q_0, Q \setminus F)$ recognizes $\overline{A}$.
Union is regular

Let languages $A$ and $B$ be recognized by NFAs $N_1$ and $N_2$, respectively. Then an NFA $N$ recognizing $A \cup B$ can be constructed as follows:
Concatenation is regular

Let languages $A$ and $B$ be recognized by NFAs $N_1$ and $N_2$, respectively. Then an NFA $N$ recognizing $A \circ B$ can be constructed as follows:
Powers and Kleene closure

For a language $A$, we let $A^0 = \{\epsilon\}$ and for $k > 0$, $A^k = (A^{k-1}) \circ A$. In other words,

$$A^k = A \circ A \circ \ldots \circ A.$$  

Definition

Kleene star (or Kleene closure) of a language $A$ is the language

$$A^* = \{x_1x_2\ldots x_k \mid k \in \mathbb{N} \land \forall i = 1, 2, \ldots, k, \ x_i \in A\}.$$  

In other words,

$$A^* = A^0 \cup A^1 \cup \ldots = \bigcup_{k \geq 0} A^k.$$  

Powers and Kleene closure: example

If $A = \{0, 11\}$, then

- $A^0 = \{\epsilon\}$
- $A^1 = A = \{0, 11\}$
- $A^2 = AA = \{00, 011, 110, 1111\}$
- and so on

$$A^* = A^0 \cup A^1 \cup \ldots$$ can be described as consisting of the strings over $\{0, 1\}$ where each run of consecutive 1’s has even length.
The notion of a clean NFA pops up on several occasions.

**Definition**

Let $M$ be an NFA. We say that $M$ is *clean* iff

- $M$ has exactly one final state, which is distinct from the start state,
- $M$ has no transitions into its start state (even self-loops), and
- $M$ has no transitions out of its final state (even self-loops).
Existence of a Clean NFA

Proposition
For any NFA $M$, there is an equivalent clean NFA $N$.

Proof.
If $M$ is not clean, then we can “clean it up” by adding two additional states:

- a new start state with a single $\varepsilon$-transition to $M$’s original start state (which is no longer the start state), and
- a new final state with $\varepsilon$-transitions from all of $M$’s original final states (which are no longer final states) to the new final state.

The new NFA is obviously clean, and a simple, informal argument shows that it is equivalent to the original $M$.  

Q: Give clean NFAs for concatenation and union (series and parallel connections, respectively) as well as for the Kleene closure.