Relating DFAs and NFAs

6 February 2020
Assignment 1
Corrections due
Assignment 2
Out now
Academic intern (Eve) coaching hours:

- Tuesday: 3:00–5:30 p.m.
- Wednesday: 2:00–4:00 p.m.
- Thursday: 3:00–5:30 p.m.

Asprey Lab
Where are we?
A language is a *regular language* iff some deterministic finite automaton recognizes it.

What languages *aren’t* regular?

Languages that require you to remember more than a finite number of possibilities.

Finite automata can’t:

- count
- remember exactly what’s been read (in general)

$L = \{ww\}$ isn’t regular

$L = \{0^n1^n\}$ isn’t regular.
Imagine a string from here to the moon. You’re trying to recognize it, but you only have a fixed number of states, e.g., 100.
DFAs let us recognize languages where we only need to keep track of a fixed set of possibilities, e.g.,

- All even-length strings
- Any number of a's followed by any number of b's
- Alternating a's and b's
- Strings ending with er, or, or ist
**Deterministic finite automaton**: Computation proceeds according to the design of the transition function. There are no choices to make in the computation.

**Nondeterministic finite automaton**: Zero or more options to continue the computation. Accept if *any possible* sequence of them would succeed.
NFAs can have a special type of transition called an $\varepsilon$-transition.

An NFA may follow any number of $\varepsilon$-transitions at any time without consuming any input.
Thinking about NFAs
Nondeterministic machines are a serious departure from physical computers.

There are two helpful ways to think about nondeterministic computation:

- Perfect positive guessing
- Massive parallelism
Perfect positive guessing
Perfect positive guessing

\[ q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3 \]

\[ a, b \]

\[ a \ b \ a \ b \ a \ a \]
Perfect positive guessing
Perfect positive guessing

\[ q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3 \]

Input sequence: \[ a \ b \ a \ b \ a \]
Perfect positive guessing

\[
\begin{align*}
 q_0 & \xrightarrow{a, b} q_0 \\
 q_0 & \xrightarrow{a} q_1 \\
 q_1 & \xrightarrow{b} q_2 \\
 q_2 & \xrightarrow{a} q_3
\end{align*}
\]
Perfect positive guessing

\[
\begin{align*}
q_0 & \xrightarrow{a,b} q_0 \\
q_0 & \xrightarrow{a} q_1 \\
q_1 & \xrightarrow{b} q_2 \\
q_2 & \xrightarrow{a} q_3
\end{align*}
\]
Perfect positive guessing

\[ q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3 \]

Start

Input sequence: a b a b a a
Perfect positive guessing
Perfect positive guessing

- **Start state:** $q_0$
- **Transitions:**
  - $q_0$ to $q_1$: $a$
  - $q_1$ to $q_2$: $b$
  - $q_2$ to $q_3$: $a$
  - $q_0$ (self-loop): $\{a, b\}$

Input sequence: $a b a b a a$
Perfect positive guessing

\begin{align*}
&\text{\textit{start}} & \rightarrow & \text{q}_0 \\
& & \rightarrow & \text{q}_1 \\
& & \rightarrow & \text{q}_2 \\
& & \rightarrow & \text{q}_3
\end{align*}

\begin{align*}
a, b & \rightarrow \text{q}_0 \\
a & \rightarrow \text{q}_1 \\
b & \rightarrow \text{q}_2 \\
a & \rightarrow \text{q}_3
\end{align*}

\begin{align*}
\text{a b a b a a}
\end{align*}
Perfect positive guessing

$start\rightarrow q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3$

$a, b$
Perfect positive guessing

\[ \text{start} \rightarrow q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3 \]

\[ a, b \]

\[ a \ b \ a \ b \ a \ b \ a \]
Perfect positive guessing

Illustration by Gemma Correll
NFAs have a “Liquid Luck” potion
Perfect positive guessing

We can think of nondeterministic machines as having *magic powers* that enable them to guess the correct choice of moves to make.

If there is at least one choice leading to an accepting state for the input, the machine will guess it.

If there are no choices, the machine guesses any one of the wrong answers.

There’s no physical analog for this style of computation.
Massive parallelism
Massive parallelism

\[ a, b \]

\[ q_0 \] \[ a \] \[ q_1 \] \[ b \] \[ q_2 \] \[ a \] \[ q_3 \]

\[ a b a b a b a \]
Massive parallelism

- Start state: $q_0$
- Transitions: a, b
- Path: $q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3$
- Input sequence: a b a b a a
Massive parallelism

\[ a, b \]

\[ q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3 \]

\[ a \ b \ a \ b \ a \]

\[ \text{start} \]
Massive parallelism
Massive parallelism

\[ a, b \]

\[ start \rightarrow q_0 \rightarrow q_1 \rightarrow q_2 \rightarrow q_3 \]

\[ a \ b \ a \ b \ a \]
Massive parallelism
Massive parallelism

\[ a, b \]

\[ q_0 \] \[ a \] \[ q_1 \] \[ b \] \[ q_2 \] \[ a \] \[ q_3 \]

\[ a \ b \ a \ b \ a \ b \ a \]
Massive parallelism

\[
\begin{align*}
&\text{start} \\
&q_0 \xrightarrow{a, b} q_0 \\
&q_0 \xrightarrow{a} q_1 \\
&q_1 \xrightarrow{b} q_2 \\
&q_2 \xrightarrow{a} q_3
\end{align*}
\]
Massive parallelism
Massive parallelism

a, b

q0

a

q1

b

q2

a

q3

a b a b a a
Massive parallelism

\[
\begin{align*}
q_0 &\xrightarrow{a} q_1 \\
q_1 &\xrightarrow{b} q_2 \\
q_2 &\xrightarrow{a} q_3
\end{align*}
\]
Massive parallelism

\( q_0 \xrightarrow{a,b} q_0 \)

\( q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3 \)

\( a \ b \ a \ b \ a \ a \)
Massive parallelism

START

q₀

a, b

q₁

a

q₂

b

q₃

a, b, a, b, a
Massive parallelism

Diagram:
- Initial state: $q_0$
- Transitions:
  - $a$: $q_0 \rightarrow q_1$
  - $b$: $q_1 \rightarrow q_2$
  - $a$: $q_2 \rightarrow q_3$
- Additional states: $q_1$, $q_2$, $q_3$
- Special state: $q_3$ (satisfies $a, b$)
Massive parallelism
Massive parallelism

\[ q_0, a, b, q_1, b, q_2, a, q_3 \]
Massive parallelism

\[ \begin{align*}
q_0 &\xrightarrow{a, b} q_0, \\
q_0 &\xrightarrow{a} q_1, \\
q_1 &\xrightarrow{b} q_2, \\
q_2 &\xrightarrow{a} q_3
\end{align*} \]
Massive parallelism

$q_0$ to $q_1$: $a$, $b$ to $q_2$: $b$, $a$ to $q_3$: $a$

Input sequence: $a \ b \ a \ b \ a \ a$
Massive parallelism
Massive parallelism

\[ q_0, a, b \rightarrow q_1, a, b, a \rightarrow q_2, b, a \rightarrow q_3 \]
Massive parallelism

\[ q_0, a, b, q_1, a, b, q_2, a, q_3 \]
Massive parallelism
Massive parallelism

START

\[ q_0 \xrightarrow{a,b} q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3 \]

Input: abcaba

State: q3
Massive parallelism
Massive parallelism
Massive parallelism

Start

q0 -> a -> q1 -> b -> q2 -> a -> q3

a, b

a b a b a
Massive parallelism

a, b

start

q0 -> q1 -> q2 -> q3

a b a a b a
Massive parallelism

- **Start state:** $q_0$
- Edges:
  - $q_0 \xrightarrow{a,b} q_0$
  - $q_0 \xrightarrow{a} q_1$
  - $q_1 \xrightarrow{b} q_2$
  - $q_2 \xrightarrow{a} q_3$

Input sequence: $abaaba$
Massive parallelism

\[ q_0, a, b, q_1, b, q_2, a, q_3 \]
Massive parallelism

\[ q_0, a, b, q_1, b, q_2, a, q_3 \]
Massive parallelism

\[ \begin{align*}
\text{start} & \rightarrow q_0 \quad \text{a, b} \\
q_0 & \rightarrow q_1 \quad \text{a} \\
q_1 & \rightarrow q_2 \quad \text{b} \\
q_2 & \rightarrow q_3 \quad \text{a}
\end{align*} \]
Massive parallelism

$a, b$

$start$

$q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3$

$\text{a b a b a}$
Massive parallelism

\[ \begin{align*}
q_0 & \xrightarrow{a} q_1 \\
q_1 & \xrightarrow{b} q_2 \\
q_2 & \xrightarrow{a} q_3
\end{align*} \]

start

\[ \begin{align*}
a, b
\end{align*} \]
Massive parallelism

\[ a, b \]

\[ a \rightarrow q_0 \rightarrow q_1 \rightarrow q_2 \rightarrow q_3 \]
Massive parallelism

One of the states we’re in is an accepting state, so there is a path where the NFA accepts the input string.
Massive parallelism

One of the states we’re in is an accepting state, so there is a path where the NFA accepts the input string.
The future was and is massive parallelism.
Massive parallelism

An NFA can also be thought of as a DFA that can be in many states at once.

Each symbol read causes a transition on every active state into each potential state that could be visited.

Nondeterministic machines can be thought of as machines that can try any number of options in parallel.
Two roads diverged in a wood, and I –

both of them, at the same time, like a boss

I took the one less traveled by,

And that has made all the difference.

Robert Frost
Perfect guessing is a helpful way to think about how to design a machine to recognize a language.

Massive parallelism is a great way to test machines, and it has nice theoretical implications.
Designing NFAs
Embrace the nondeterminism.

A good approach is guess-and-check:

Is there some information you’d like to have?
Have the machine nondeterministically guess that information.
Then have it deterministically check that the choice was right, i.e., filter out the bad guesses.

The guess phase corresponds to trying lots of different options.

The check phase corresponds to filtering out bad guesses or wrong options.
L = \{w \in \{0, 1\}^* | w \text{ ends in 010 or 101}\}
\[ L = \{w \in \{0, 1\}^* \mid w \text{ ends in } 010 \text{ or } 101\} \]
\[ L = \{ w \in \{0, 1\}^* \mid \text{w ends in 010 or 101}\} \]
\[ L = \{ w \in \{0, 1\}^* \mid w \text{ ends in } 010 \text{ or } 101 \} \]
\[ L = \{ w \in \{0, 1\}^* \mid w \text{ ends in } 010 \text{ or } 101 \} \]

Nondeterministically guess when the end of the string is coming up. Deterministically check whether you were correct.
\[ L = \{ w \in \{0, 1\}^* \mid w \text{ ends in 010 or 101}\} \]
$$L = \{ w \in \{0, 1\}^* \mid w \text{ ends in } 010 \text{ or } 101 \}$$
\[ L = \{ w \in \{0, 1\}^* \mid w \text{ ends in } 010 \text{ or } 101 \} \]
\[ L = \{ w \in \{0, 1\}^* \mid w \text{ ends in 010 or 101} \} \]
$L = \{ w \in \{0, 1\}^* \mid w \text{ ends in } 010 \text{ or } 101 \}$
\[ L = \{ w \in \{0, 1\}^* \mid w \text{ ends in } 010 \text{ or } 101 \} \]
$L = \{ w \in \{0, 1\}^* \mid w \text{ ends in } 010 \text{ or } 101 \}$
\[ L = \{ w \in \{0, 1\}^* \mid w \text{ ends in } 010 \text{ or } 101 \} \]
\[ L = \{ w \in \{0, 1\}^* \mid w \text{ ends in } 010 \text{ or } 101 \} \]
\[ L = \{ w \in \{0, 1\}^* \mid w \text{ ends in } 010 \text{ or } 101 \} \]
\[ L = \{ w \in \{0, 1\}^* \mid \text{w ends in 010 or 101} \} \]
\[ L = \{ w \in \{0, 1\}^* \mid w \text{ ends in } 010 \text{ or } 101 \} \]
\[ L = \{w \in \{0, 1\}^* \mid w \text{ ends in } 010 \text{ or } 101\} \]
\[ L = \{ w \in \{a, b, c\}^* \mid \text{at least one of } a, b, \text{ or } c \text{ is not in } w \} \]
\( L = \{ w \in \{a, b, c\}^* \mid \text{at least one of } a, b, \text{ or } c \text{ is not in } w \} \)
$L = \{ w \in \{a, b, c\}^* \mid \text{at least one of } a, b, \text{ or } c \text{ is not in } w \}$
\[ L = \{ w \in \{a, b, c\}^* \mid \text{at least one of } a, b, \text{ or } c \text{ is not in } w \} \]
\[ L = \{w \in \{a, b, c\}^* \mid \text{at least one of } a, b, \text{ or } c \text{ is not in } w\} \]
\[ L = \{ w \in \{a, b, c\}^* \mid \text{at least one of } a, b, \text{ or } c \text{ is not in } w \} \]

Nondeterministically guess which character is missing.
Deterministically check whether that character is indeed missing.
Formal NFAs
Formally, an NFA is defined like a DFA:

\[ N = (Q, \Sigma, \delta, q_0, F) \]

Except now \( \delta(q, a) \) is a \textit{set of states}, rather than a single state.
We need to extend our definition of $\hat{\delta}$:

**Basis:**

$$\hat{\delta}(q, \varepsilon) = \{q\}$$

**Induction:**

$$\hat{\delta}(q, w) = \{p_1, p_2, \ldots, p_k\}$$

$$\delta(p_i, \alpha) = S_i \text{ for } i = 1, 2, \ldots, k$$

Then

$$\hat{\delta}(q, w\alpha) = S_1 \cup S_2 \cup \cdots \cup S_k$$
Language of an NFA:

An NFA accepts $w$ if any path from the start state to an accepting state is labeled $w$.

Formally:

$$L(N) = \{w \mid \hat{\delta}(q_0, w) \cap F \neq \emptyset\}$$
Relating DFAs and NFAs
Just how powerful are NFAs?
**NFAs must be at least as powerful as DFAs.**

Any language that can be recognized by a DFA can be recognized by an NFA.

*Why?* Essentially, every DFA already is an NFA, just one that doesn’t exploit nondeterminism.
Can every language recognized by an NFA also be recognized by a DFA?

NFAs seem more powerful, but, surprisingly, the answer is yes!
Consider: How could you simulate an NFA in software?
The diagram shows a finite automaton with states $q_0$, $q_1$, $q_2$, and $q_3$. The start state is $q_0$, and the transitions are:

- From $q_0$ to $q_1$ on input $a$.
- From $q_1$ to $q_2$ on input $b$.
- From $q_2$ back to $q_1$ on input $a$.
- From $q_1$ to $q_3$ on input $a$.

The input string $a b a b a a$ is shown, indicating the path taken by the automaton from the start state to the final state $q_3$. The automaton accepts the input string.
The diagram represents a finite automaton with the following transitions:

- Start state: $q_0$
- Transition: $a \rightarrow q_1$
- Transition: $b \rightarrow q_2$
- Transition: $a \rightarrow q_3$

The accepting state is $q_3$. The input sequence $ababa$ should be processed by the automaton.
The diagram shows a finite automaton with states $q_0$, $q_1$, $q_2$, and $q_3$. The automaton starts at state $q_0$ and moves to $q_1$ on input 'a', then to $q_2$ on input 'b', and finally to $q_3$ on input 'a'. The input sequence 'a b a b a b a' is accepted by the automaton, as indicated by the arrow pointing to $q_3$. The tape contains the sequence 'a b a b a b a', which corresponds to the input sequence.
A deterministic finite automaton (DFA) with transitions:

1. Start in state $q_0$.
2. On input $a$, move to state $q_1$.
3. On input $b$, move to state $q_2$.
4. On input $a$, move to state $q_3$.

The transition function is:

- $a(q_0) = q_1$
- $b(q_1) = q_2$
- $a(q_2) = q_3$

The language accepted by this DFA is $\{q_0\}$. The transition diagram is shown with states $q_0$, $q_1$, $q_2$, and $q_3$. Transitions are labeled with input symbols $a$ or $b$. The initial state is $q_0$ and the final state is $q_3$. The loop arrow indicates an additional transition on any input symbol from $q_3$. The language $\{q_0\}$ includes the empty string, indicating that the automaton accepts the empty language. The input alphabet includes $a$ and $b$. The diagram is a simple linear sequence with a loop back to $q_0$.
A transition diagram is shown with states labeled $q_0$, $q_1$, $q_2$, and $q_3$. The transitions are:

- From $q_0$ to $q_1$ on input $a$.
- From $q_1$ to $q_2$ on input $b$.
- From $q_2$ to $q_3$ on input $a$.
- A loop from $q_0$ back to itself on input $a$, $b$.

The set of final states is denoted as $\{q_0\}$.
\[ \rightarrow \{q_0\} \]
\[
\begin{array}{cc}
\rightarrow \{q_0\} & \{q_0, q_1\}
\end{array}
\]
The DFA accepts strings that end in \( q_3 \).

- \( \{q_0\} \) is accepted by \( q_0 \).
- \( \{q_0, q_1\} \) is accepted by \( q_2 \).

The transition diagram is as follows:

- Start at \( q_0 \).
- Accept \( a, b \) to stay in \( q_0 \).
- Move to \( q_1 \) on \( a \).
- Move to \( q_2 \) on \( b \).
- Move to \( q_3 \) on \( a \).
A, B

\( q_0 \quad q_1 \quad q_2 \quad q_3 \)

\( \rightarrow \{q_0\} \quad \{q_0, q_1\} \quad \{q_0\} \)
\begin{itemize}
  \item a, b
  \item start
  \item $q_0$
  \item $a ightarrow \{q_0\}$
  \item $b ightarrow \{q_0, q_1\}$
  \item $a ightarrow \{q_0, q_1\}$
  \item $b ightarrow \{q_0\}$
  \item $q_2$
  \item $a ightarrow \{q_0\}$
  \item $q_3$
\end{itemize}
\[
\begin{align*}
\text{start} & \quad a, b \\
q_0 & \quad a \quad q_1 \\
b & \quad q_1 \quad q_2 \\
a & \quad q_2 \quad q_3 \\
\end{align*}
\]
\[
\begin{align*}
\rightarrow \quad &\{q_0\} &\{q_0, q_1\} &\{q_0\} \\
&\{q_0, q_1\}
\end{align*}
\]
\[
\begin{align*}
\{q_0\} & \quad \{q_0, q_1\} & \quad \{q_0\} \\
\{q_0, q_1\} & \quad \{q_0\}
\end{align*}
\]
The given automaton has the following states and transitions:

- **States:** q₀, q₁, q₂, q₃
- **Initial State:** q₀
- **Final State:** q₃

**Transitions:**
- **a**:
  - From q₀ to q₁: \( \{q₀\} \rightarrow \{q₀, q₁\} \)
  - From q₁ to q₂: \( \{q₀, q₁\} \rightarrow \{q₀, q₁\} \)
  - From q₂ to q₃: \( \{q₀\} \rightarrow \{q₀\} \)
- **b**:
  - From q₀ to q₁: \( \{q₀\} \rightarrow \{q₀, q₁\} \)
  - From q₁ to q₂: \( \{q₀, q₁\} \rightarrow \{q₀, q₁\} \)
  - From q₂ to q₃: \( \{q₀\} \rightarrow \{q₀\} \)

**Start State:** q₀
\[
\begin{align*}
\delta(q_0, a) &= \{q_0\} \\
\delta(q_0, b) &= \{q_0, q_1\} \\
\delta(q_1, a) &= \{q_0, q_1\} \\
\delta(q_1, b) &= \{q_0\} \\
\delta(q_2, a) &= \{q_0, q_1\} \\
\delta(q_2, b) &= \{q_0\} \\
\end{align*}
\]
The given automaton has the following states and transitions:

- **States:** $q_0, q_1, q_2, q_3$
- **Transition Labels:** $a, b$

**Transitions:**
- From $q_0$: $a \rightarrow \{q_0\}, b \rightarrow \{q_0, q_1\}$
- From $q_1$: $b \rightarrow \{q_0\}, a \rightarrow \{q_0, q_1\}$
- From $q_2$: $a \rightarrow \{q_3\}$
- From $q_3$: $a \rightarrow \{q_3\}$

**Initial State:** $q_0$

**Accepting States:** $q_3$
\[ \begin{align*}
  q_0 & \rightarrow \{q_0\} \\
  q_0, q_1 & \rightarrow \{q_0, q_1\} \\
  q_0, q_1 & \rightarrow \{q_0, q_1\} \\
  q_0 & \rightarrow \{q_0\} \\
  q_1 & \rightarrow \{q_0, q_1\} \\
  q_2 & \rightarrow \{q_0\} \\
  q_3 & \rightarrow \{q_3\}
\end{align*} \]
\[
\begin{align*}
\text{start} & \rightarrow \{q_0\} \\
q_0 & \xrightarrow{a, b} \{q_0, q_1\} \\
q_1 & \xrightarrow{a} \{q_0, q_1\} \\
q_2 & \xrightarrow{b} \{q_0\} \\
q_3 & \xrightarrow{a} \{q_0, q_1\}
\end{align*}
\]
\[
\begin{array}{c c c}
\text{a} & \text{b} \\
\{q_0\} & \{q_0, q_1\} & \{q_0\} \\
\{q_0, q_1\} & \{q_0, q_1\} & \{q_0, q_2\}
\end{array}
\]
Transition Table:

\[
\begin{array}{ccc}
\text{Input} & \text{Transition} & \text{Next State(s)} \\
\text{a} & \{q_0\} & \{q_0, q_1\} \\
\text{b} & \{q_0\} & \{q_0\} \\
\text{a} & \{q_0, q_1\} & \{q_0, q_1\} \\
\text{b} & \{q_0, q_1\} & \{q_0, q_2\} \\
\text{b} & \{q_0, q_2\} & \{q_0, q_2\} \\
\end{array}
\]
\begin{align*}
&\rightarrow \quad \{q_0\} \quad \{q_0, q_1\} \quad \{q_0\} \\
&\quad \{q_0, q_1\} \quad \{q_0, q_1\} \quad \{q_0, q_2\} \\
&\quad \{q_0, q_2\}
\end{align*}
A DFA with states $q_0$, $q_1$, $q_2$, and $q_3$. The diagram shows transitions on input symbols $a$ and $b$.

- From $q_0$, on input $a$, go to $q_0$, and on input $b$, go to $q_1$.
- From $q_1$, on input $a$, go to $q_2$, and on input $b$, go to $q_2$.
- From $q_2$, on input $a$, go back to $q_3$.

Transition tables:

<table>
<thead>
<tr>
<th>Input</th>
<th>$q_0$</th>
<th>$q_1$</th>
<th>$q_2$</th>
<th>$q_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>$q_0$</td>
<td>$q_2$</td>
<td>$q_2$</td>
<td></td>
</tr>
<tr>
<td>$b$</td>
<td>$q_1$</td>
<td>$q_2$</td>
<td>$q_0$</td>
<td>$q_3$</td>
</tr>
</tbody>
</table>

Symbols $a$ and $b$.

States:

- $q_0$ (start state)
- $q_1$
- $q_2$
- $q_3$ (accepting state)
\[
\begin{align*}
q_0 & \rightarrow \{q_0\} \\
\{q_0, q_1\} & \rightarrow \{q_0, q_1\} \\
\{q_0, q_1\} & \rightarrow \{q_0, q_2\} \\
\{q_0, q_2\} & \rightarrow \{q_0, q_2\}
\end{align*}
\]
\begin{align*}
\rightarrow \quad & \{q_0\} & \{q_0, q_1\} & \{q_0\} \\
& \{q_0, q_1\} & \{q_0, q_1\} & \{q_0, q_2\} \\
& \{q_0, q_2\} & & \\
\end{align*}
\[
\begin{align*}
&\rightarrow \{q_0\} \quad \{q_0, q_1\} \quad \{q_0\} \\
&\{q_0, q_1\} \quad \{q_0, q_1\} \quad \{q_0, q_2\} \\
&\{q_0, q_2\}
\end{align*}
\]
$q_0 \xrightarrow{a, b} q_0$

$$
\begin{array}{c c c c c}
\text{a} & \text{b} \\
\{q_0\} & \{q_0, q_1\} & \{q_0\} \\
\{q_0, q_1\} & \{q_0, q_1\} & \{q_0, q_2\} \\
\{q_0, q_2\} & \{q_0, q_1, q_3\} & \\
\end{array}
$$
a, b

start

\[ q_0 \rightarrow \{q_0\} \]
\[ \{q_0, q_1\} \rightarrow \{q_0\} \]
\[ \{q_0, q_1\} \rightarrow \{q_0, q_1\} \]
\[ \{q_0, q_2\} \rightarrow \{q_0, q_1, q_3\} \]
\[
\begin{array}{c|c|c}
& a & b \\
\hline
\{q_0\} & \{q_0, q_1\} & \{q_0\} \\
\{q_0, q_1\} & \{q_0, q_1\} & \{q_0, q_2\} \\
\{q_0, q_2\} & \{q_0, q_1, q_3\} & \end{array}
\]
\[
\begin{array}{cccc}
a, b & \rightarrow & \{q_0\} & \{q_0\} \\
& & \{q_0, q_1\} & \{q_0\} \\
& & \{q_0, q_1\} & \{q_0, q_2\} \\
& & \{q_0, q_2\} & \{q_0, q_1, q_3\}
\end{array}
\]
\[ \begin{array}{c}
\text{start} \\
\rightarrow \\
\{q_0\} \\
\{q_0, q_1\} \\
\{q_0, q_1\} \\
\{q_0, q_2\} \\
\{q_0, q_2\} \\
\{q_0, q_1, q_3\} \\
\end{array} \]
\[ \begin{align*}
\text{start} & \rightarrow \{q_0\} \\
\{q_0, q_1\} & \rightarrow \{q_0\} \\
\{q_0, q_1\} & \rightarrow \{q_0, q_1\} \\
\{q_0, q_2\} & \rightarrow \{q_0, q_1, q_3\} \\
\end{align*} \]
\[
\begin{align*}
\text{a, b} & \quad \{q_0\} & \{q_{0, q_1}\} & \{q_0\} \\
\text{a} & \quad \{q_{0, q_1}\} & \{q_{0, q_1}\} & \{q_{0, q_2}\} \\
\text{b} & \quad \{q_{0, q_2}\} & \{q_{0, q_1, q_3}\}
\end{align*}
\]
\[
\begin{align*}
{q_0} & \rightarrow \{q_0\} \quad \{q_0, q_1\} \quad \{q_0\} \\
\{q_0, q_1\} & \rightarrow \{q_0, q_1\} \quad \{q_0, q_2\} \\
\{q_0, q_2\} & \rightarrow \{q_0, q_1, q_3\} \quad \{q_0\}
\end{align*}
\]
A, B

\[
\begin{align*}
{q_0} & \rightarrow {q_0} \\
{q_0, q_1} & \rightarrow {q_0, q_1} \\
{q_0, q_1} & \rightarrow {q_0, q_1} \\
{q_0, q_2} & \rightarrow {q_0, q_1, q_3} \\
{q_0, q_1, q_3} & \rightarrow {q_0}
\end{align*}
\]
The given automaton is a deterministic finite automaton (DFA) with states $q_0$, $q_1$, $q_2$, and $q_3$. The alphabet includes $a$ and $b$. The transitions are as follows:

- From $q_0$, on input $a$, go to $q_1$, and on input $b$, go back to $q_0$.
- From $q_1$, on input $a$, go to $q_0$, and on input $b$, go to $q_2$.
- From $q_2$, on input $a$, go to $q_3$.

The transitions are summarized as:

- $a$: $q_0 \rightarrow \{q_0\}$, $q_0 \rightarrow \{q_0, q_1\}$, $q_1 \rightarrow \{q_0, q_1\}$, $q_2 \rightarrow \{q_0\}$, $q_3 \rightarrow \{q_0\}$
- $b$: $q_0 \rightarrow \{q_0\}$, $q_0 \rightarrow \{q_0, q_1\}$, $q_1 \rightarrow \{q_0, q_1\}$, $q_2 \rightarrow \{q_0, q_2\}$, $q_3 \rightarrow \{q_0, q_1, q_3\}$, $q_3 \rightarrow \{q_0\}$
\[
\begin{align*}
\rightarrow & \quad \{q_0\} \quad \{q_0, q_1\} \quad \{q_0\} \\
& \quad \{q_0, q_1\} \quad \{q_0, q_1\} \quad \{q_0, q_2\} \\
& \quad \{q_0, q_2\} \quad \{q_0, q_1, q_3\} \quad \{q_0\} \\
& \quad \{q_0, q_1, q_3\}
\end{align*}
\]
\[
\begin{align*}
\rightarrow \quad & \{q_0\} \quad \{q_0, q_1\} \quad \{q_0\} \\
& \{q_0, q_1\} \quad \{q_0, q_1\} \quad \{q_0, q_2\} \\
& \{q_0, q_2\} \quad \{q_0, q_1, q_3\} \quad \{q_0\} \\
& \{q_0, q_1, q_3\}
\end{align*}
\]
\[ \begin{align*}
\text{start} & \rightarrow \{q_0\} \\
\{q_0, q_1\} & \rightarrow \{q_0, q_1\} \\
\{q_0, q_2\} & \rightarrow \{q_0, q_1, q_3\} \\
\{q_0, q_1, q_3\} & \rightarrow \{q_0\}
\end{align*} \]
Start

- Transition on a: States {q0}, {q0, q1}, {q0, q2}, {q0, q1, q3}
- Transition on b: States {q0, q1}, {q0, q3}
a, b

\[
\begin{array}{ccc}
q_0 & \xrightarrow{a} & q_1 \\
& \xrightarrow{b} & q_2 \\
& \xrightarrow{a} & q_3
\end{array}
\]

\[
\begin{array}{ccc}
a & \rightarrow & \{q_0\} \\
\{q_0, q_1\} & \rightarrow & \{q_0\} \\
\{q_0, q_1\} & \rightarrow & \{q_0, q_1\} \\
\{q_0, q_2\} & \rightarrow & \{q_0, q_2\} \\
\{q_0, q_2\} & \rightarrow & \{q_0, q_1, q_3\} \\
\{q_0, q_1, q_3\} & \rightarrow & \{q_0\}
\end{array}
\]
\[
\begin{array}{ccc}
q_0 & \xrightarrow{a} & q_1 \\
q_1 & \xrightarrow{b} & q_2 \\
q_2 & \xrightarrow{a} & q_3 \\
\end{array}
\]

\[
\begin{array}{ccc}
\{q_0\} & \{q_0, q_1\} & \{q_0\} \\
\{q_0, q_1\} & \{q_0, q_1\} & \{q_0, q_2\} \\
\{q_0, q_2\} & \{q_0, q_1, q_3\} & \{q_0\} \\
\{q_0, q_1, q_3\} & \{q_0, q_1\} & \{q_0\} \\
\end{array}
\]
\[
\begin{align*}
\text{\textbf{a}} & \rightarrow \{q_0\} & \{q_0, q_1\} & \{q_0\} \\
\{q_0, q_1\} & \{q_0, q_1\} & \{q_0, q_2\} \\
\{q_0, q_2\} & \{q_0, q_1, q_3\} & \{q_0\} \\
\{q_0, q_1, q_3\} & \{q_0, q_1\}
\end{align*}
\]
\begin{align*}
\rightarrow & \quad \{q_0\} \quad \{q_0, q_1\} \quad \{q_0\} \\
\& \quad \{q_0, q_1\} \quad \{q_0, q_1\} \quad \{q_0, q_2\} \\
\& \quad \{q_0, q_2\} \quad \{q_0, q_1, q_3\} \quad \{q_0\} \\
\& \quad \{q_0, q_1, q_3\} \quad \{q_0, q_1\}
\end{align*}
\[
\begin{align*}
\{q_0\} & \quad \{q_0, q_1\} & \quad \{q_0\} \\
\{q_0, q_1\} & \quad \{q_0, q_1\} & \quad \{q_0, q_2\} \\
\{q_0, q_2\} & \quad \{q_0, q_1, q_3\} & \quad \{q_0\} \\
\{q_0, q_1, q_3\} & \quad \{q_0, q_1\}
\end{align*}
\]
\[
\begin{align*}
\text{start} & \rightarrow q_0 \\
q_0 & \xrightarrow{a, b} q_0 \\
q_0 & \xrightarrow{a} q_1 \\
q_1 & \xrightarrow{b} q_2 \\
q_2 & \xrightarrow{a} q_3 \\
q_3 & \xrightarrow{a, b} q_3
\end{align*}
\]
\[
\begin{array}{c}
\{q_0\} \quad \{q_0, q_1\} \quad \{q_0\} \\
\{q_0, q_1\} \quad \{q_0, q_1\} \quad \{q_0, q_2\} \\
\{q_0, q_2\} \quad \{q_0, q_1, q_3\} \quad \{q_0\} \\
\{q_0, q_1, q_3\} \quad \{q_0, q_1\} \quad \{q_0, q_2\}
\end{array}
\]
The diagram represents a deterministic finite automaton (DFA) with transitions:

- Start state: $q_0$
- Transitions:
  - $q_0 \xrightarrow{a} q_1$
  - $q_0 \xrightarrow{b} q_2$
  - $q_1 \xrightarrow{a} q_2$
  - $q_2 \xrightarrow{a} q_3$

The input string $abaaba$ is processed as follows:

- Start at $q_0$.
- Move to $q_1$ on $a$.
- Stay at $q_1$ on $b$.
- Move to $q_2$ on $a$.
- Move to $q_3$ on $a$.

The accepting state is $q_3$. The automaton accepts the string $abaaba$. 

The QED automaton has states:

- $q_0$
- $q_1$
- $q_2$
- $q_3$

Transitions include:

- $q_0 \xrightarrow{a} q_1$
- $q_0 \xrightarrow{b} q_2$
- $q_1 \xrightarrow{a} q_2$
- $q_2 \xrightarrow{a} q_3$
- $q_0 \xrightarrow{b} q_2$
The given DFA accepts the string "a b a a b a a".
This method of transforming an NFA into a DFA is called the *subset construction* (or power set construction).

Each state in the DFA is associated with a set of states in the NFA. The start state in the DFA corresponds to the start state of the NFA.

If a state $q$ in the DFA corresponds to a set of states $S$ in the NFA, then the transition from state $q$ on a character $\alpha$ is found as follows:

Let $S'$ be the set of states in the NFA that can be reached by following a transition labeled $\alpha$ from any of the states in $S$.

The state $q$ in the DFA transitions on $\alpha$ to a DFA state corresponding to the set of states $S'$. 

In converting an NFA to a DFA, the DFA’s states correspond to sets of NFA states.

Useful fact: $|\mathcal{P}(S)| = 2^{|S|}$ for any finite set $S$.

In the worst case, the construction can result in a DFA that is exponentially larger than the original NFA.
THEOREM. Every nondeterministic finite automaton has an equivalent deterministic finite automaton.

PROOF. Let \( N = (Q, \Sigma, \delta, q_0, F) \) be an NFA recognizing some language \( A \).

We can construct a DFA \( M = (Q', \Sigma, \delta', q'_0, F') \) that recognizes \( A \):

\[
Q' = \wp(Q) \quad \text{As in Sipser, we use } R \text{ both as a state of } M \text{ and as a set of states of } N.
\]
\[
q'_0 = \{q_0\}
\]
\[
F' = \{R \in Q' \mid R \text{ contains an accept state of } N\}
\]

For \( R \in Q' \) and \( \alpha \in \Sigma \), we define \( \delta'(R, \alpha) = \bigcup_{r \in R} \delta(r, \alpha) \)

Every state \( R \) of \( M \) is a set of states of \( N \).

When \( M \) is in state \( R \) and reads a symbol \( \alpha \), it tracks where \( N \) would go on \( \alpha \) each state in \( R \).
Acknowledgments

This lecture incorporates material from:

Nancy Ide
Keith Schwarz
Michael Sipser
Jeffrey Ullman