CSE 135: Introduction to Theory of Computation
Nondeterministic Finite Automata

Sungjin Im

University of California, Merced

01-27-2015
Nondeterminism
Michael Rabin and Dana Scott (1959)

Nondeterminism
Given a current state of the machine and input symbol to be read, the next state is not uniquely determined.
Comparison to DFAs

Nondeterministic Finite Automata (NFA)

NFAs have 3 features when compared with DFAs.
Comparison to DFAs

Nondeterministic Finite Automata (NFA)

NFAs have 3 features when compared with DFAs.

1. Ability to take a step without reading any input symbol
Comparison to DFAs

Nondeterministic Finite Automata (NFA)

NFAs have 3 features when compared with DFAs.

1. Ability to take a step without reading any input symbol
2. A state may have no transition on a particular symbol
Comparison to DFAs

Nondeterministic Finite Automata (NFA)

NFAs have 3 features when compared with DFAs.

1. Ability to take a step without reading any input symbol
2. A state may have no transition on a particular symbol
3. Ability to transition to more than one state on a given symbol
**ε-Transitions**

Transitions without reading input symbols

**Example**

The British spelling of “color” is “colour”. In a web search application, you may want to recognize both variants.

![NFA with ε-transitions](image-url)
No transitions

Example

In the above automaton, if the string starts with a 0 then the string has no computation (i.e., rejected).
Multiple Transitions

$q_{\epsilon}$ has two 0-transitions
Parallel Computation View

At each step, the machine “forks” a thread corresponding to one of the possible next states.

- If a state has an $\epsilon$-transition, then you fork a new process for each of the possible $\epsilon$-transitions, without reading any input symbol.
- If the state has multiple transitions on the current input symbol read, then fork a process for each possibility.
- If from current state of a thread, there is no transition on the current input symbol then the thread dies.
Parallel Computation View: An Example

Example NFA
Parallel Computation View: An Example

Example NFA

Computation on 0100
Nondeterministic Acceptance
Parallel Computation View

Input is accepted if after reading all the symbols, one of the live threads of the automaton is in a final/accepting state.
Nondeterministic Acceptance
Parallel Computation View

Input is accepted if after reading all the symbols, one of the live threads of the automaton is in a final/accepting state. If none of the live threads are in a final/accepting state, the input is rejected.
Input is **accepted** if after reading all the symbols, one of the live threads of the automaton is in a final/accepting state. If none of the live threads are in a final/accepting state, the input is **rejected**.

0100 is accepted because one thread of computation is

\[
q_\epsilon \xrightarrow{0} q_0 \xrightarrow{\epsilon} q_{00} \xrightarrow{1} q_p \xrightarrow{0} q_p \xrightarrow{0} q_p
\]
Computation: Guessing View

The machine magically guesses the choices that lead to acceptance
Computation: Guessing View

The machine magically guesses the choices that lead to acceptance.

Example

![NFA Diagram]

NFA $M_{color}$

After seeing “colo” the automaton guesses if it will see the british or the american spelling. If it guesses american then it moves without reading the next input symbol.
Observations: Guessing View

- If there is a sequence of choices that will lead to the automaton (not “dying” and) ending up in an accept state, then those choices will be magically guessed.
Observations: Guessing View

- If there is a sequence of choices that will lead to the automaton (not “dying” and) ending up in an accept state, then those choices will be magically guessed.
- On the other hand, if the input will not be accepted then no guess will lead the to automaton being in an accept state.
Observations: Guessing View

- If there is a sequence of choices that will lead to the automaton (not “dying” and) ending up in an accept state, then those choices will be magically guessed.
- On the other hand, if the input will not be accepted then no guess will lead the automaton to being in an accept state.
  - On the input “colobr”, whether automaton $M_{\text{color}}$ guesses British or American, it will not proceed when it reads ‘b’.
Example 1

The automaton "guesses" at some point that the 1 it is seeing is 2 positions from end of input.
Example 1

Automaton accepts strings having a 1 two positions from end of input.

The automaton “guesses” at some point that the 1 it is seeing is 2 positions from end of input.
Example II

The NFA "guesses" at the beginning whether it will see a multiple of 2 or 3, and then confirms that the guess was correct.
Example II

NFA accepting strings of 0s where the length is either a multiple 2 or 3

The NFA “guesses” at the beginning whether it will see a multiple of 2 or 3, and then confirms that the guess was correct.
Example III

NFA accepting strings with 001 as substring

At some point the NFA "guesses" that the pattern 001 is starting and then checks to confirm the guess.
Example III

NFA accepting strings with 001 as substring

At some point the NFA “guesses” that the pattern 001 is starting and then checks to confirm the guess.
Definition
A nondeterministic finite automaton (NFA) is $M = (Q, \Sigma, \delta, q_0, F)$, where

- $Q$ is the finite set of states
- $\Sigma$ is the finite alphabet
- $\delta : Q \times (\Sigma \cup \{\epsilon\}) \to P(Q)$, where $P(Q)$ is the powerset of $Q$
- $q_0 \in Q$ initial state
- $F \subseteq Q$ final/accepting states
Nondeterministic Finite Automata (NFA)

Formal Definition

Definition
A nondeterministic finite automaton (NFA) is $M = \langle Q, \Sigma, \delta, q_0, F \rangle$, where

- $Q$ is the finite set of states
- $\Sigma$ is the finite alphabet
- $\delta : Q \times (\Sigma \cup \{\epsilon\}) \to \mathcal{P}(Q)$, where $\mathcal{P}(Q)$ is the powerset of $Q$
- $q_0 \in Q$ initial state
- $F \subseteq Q$ final/accepting states
Example of NFA

Transition Diagram of NFA

Formally, the NFA is $M_{001} = (\{q_\epsilon, q_0, q_{00}, q_p\}, \{0, 1\}, \delta, q_\epsilon, \{q_p\})$ where $
\delta$ is given by
$$
\delta(q_\epsilon, 0) = \{q_\epsilon, q_0\}
\delta(q_\epsilon, 1) = \{q_\epsilon\}
\delta(q_0, 0) = \{q_{00}\}
\delta(q_{00}, 1) = \{q_p\}
\delta(q_p, 0) = \{q_p\}
\delta(q_p, 1) = \{q_p\}
\delta$ is $\emptyset$ in all other cases.
Example of NFA

Transition Diagram of NFA

Formally, the NFA is $M_{001} = (\{q_\epsilon, q_0, q_{00}, q_p\}, \{0, 1\}, \delta, q_\epsilon, \{q_p\})$ where $\delta$ is given by

\[
\begin{align*}
\delta(q_\epsilon, 0) &= \{q_\epsilon, q_0\} & \delta(q_\epsilon, 1) &= \{q_\epsilon\} & \delta(q_0, 0) &= \{q_{00}\} \\
\delta(q_{00}, 1) &= \{q_p\} & \delta(q_p, 0) &= \{q_p\} & \delta(q_p, 1) &= \{q_p\}
\end{align*}
\]

$\delta$ is $\emptyset$ in all other cases.
Computation

Definition
For an NFA $M = (Q, \Sigma, \delta, q_0, F)$, string $w$, and states $q_1, q_2 \in Q$, we say $q_1 \xrightarrow{w} M q_2$ if there is one thread of computation on input $w$ from state $q_1$ that ends in $q_2$. 

Computation

Definition
For an NFA $M = (Q, \Sigma, \delta, q_0, F)$, string $w$, and states $q_1, q_2 \in Q$, we say $q_1 \xrightarrow{w}_M q_2$ if there is one thread of computation on input $w$ from state $q_1$ that ends in $q_2$. Formally, $q_1 \xrightarrow{w}_M q_2$ if there is a sequence of states $r_0, r_1, \ldots r_k$ and a sequence $x_1, x_2, \ldots x_k$, where for each $i$, $x_i \in \Sigma \cup \{\epsilon\}$, such that
Computation

Definition
For an NFA $M = (Q, \Sigma, \delta, q_0, F)$, string $w$, and states $q_1, q_2 \in Q$, we say $q_1 \xrightarrow{w} M q_2$ if there is one thread of computation on input $w$ from state $q_1$ that ends in $q_2$. Formally, $q_1 \xrightarrow{w} M q_2$ if there is a sequence of states $r_0, r_1, \ldots r_k$ and a sequence $x_1, x_2, \ldots x_k$, where for each $i$, $x_i \in \Sigma \cup \{\epsilon\}$, such that

$\bullet \quad r_0 = q_1,$
Definition

For an NFA $M = (Q, \Sigma, \delta, q_0, F)$, string $w$, and states $q_1, q_2 \in Q$, we say $q_1 \xrightarrow{w} M q_2$ if there is one thread of computation on input $w$ from state $q_1$ that ends in $q_2$. Formally, $q_1 \xrightarrow{w} M q_2$ if there is a sequence of states $r_0, r_1, \ldots r_k$ and a sequence $x_1, x_2, \ldots x_k$, where for each $i$, $x_i \in \Sigma \cup \{\epsilon\}$, such that

- $r_0 = q_1$,
- for each $i$, $r_{i+1} \in \delta(r_i, x_{i+1})$. 


Computation

Definition
For an NFA $M = (Q, \Sigma, \delta, q_0, F)$, string $w$, and states $q_1, q_2 \in Q$, we say $q_1 \xrightarrow{w} M q_2$ if there is one thread of computation on input $w$ from state $q_1$ that ends in $q_2$. Formally, $q_1 \xrightarrow{w} M q_2$ if there is a sequence of states $r_0, r_1, \ldots, r_k$ and a sequence $x_1, x_2, \ldots, x_k$, where for each $i$, $x_i \in \Sigma \cup \{\epsilon\}$, such that

- $r_0 = q_1$,
- for each $i$, $r_{i+1} \in \delta(r_i, x_{i+1})$,
- $r_k = q_2$, and
Definition

For an NFA $M = (Q, \Sigma, \delta, q_0, F)$, string $w$, and states $q_1, q_2 \in Q$, we say $q_1 \xrightarrow{w} M q_2$ if there is one thread of computation on input $w$ from state $q_1$ that ends in $q_2$. Formally, $q_1 \xrightarrow{w} M q_2$ if there is a sequence of states $r_0, r_1, \ldots r_k$ and a sequence $x_1, x_2, \ldots x_k$, where for each $i$, $x_i \in \Sigma \cup \{\epsilon\}$, such that

- $r_0 = q_1$,
- for each $i$, $r_{i+1} \in \delta(r_i, x_{i+1})$,
- $r_k = q_2$, and
- $w = x_1x_2x_3 \cdots x_k$
Example Computation

$q_\epsilon \xrightarrow{0100} M q_p$ because taking $r_0 = q_\epsilon$, $r_1 = q_0$, $r_2 = q_{00}$, $r_3 = q_p$, $r_4 = q_p$, $r_5 = q_p$, and $x_1 = 0$, $x_2 = \epsilon$, $x_3 = 1$, $x_4 = 0$, $x_5 = 0$, we have

- $x_1 x_2 \cdots x_5 = 0\epsilon100 = 0100$
- $r_{i+1} \in \delta(r_i, x_{i+1})$
Defining $\hat{\Delta}$

**Definition**
For an NFA $M = (Q, \Sigma, \delta, q_0, F)$, string $w$, and state $q_1 \in Q$, we say $\hat{\Delta}(q_1, w)$ to denote states of all the active threads of computation on input $w$ from $q_1$. 
Defining \( \hat{\Delta} \)

Definition
For an NFA \( M = (Q, \Sigma, \delta, q_0, F) \), string \( w \), and state \( q_1 \in Q \), we say \( \hat{\Delta}(q_1, w) \) to denote states of all the active threads of computation on input \( w \) from \( q_1 \). Formally,

\[
\hat{\Delta}(q_1, w) = \{ q \in Q \mid q_1 \xrightarrow{w}^M q \}
\]
Example

Example NFA

\[ \hat{\Delta}(q_\epsilon, 0100) = \{ q_0, q_00, q_\epsilon \} \]
Example

Example NFA

Computation on 0100
Example

\[ \hat{\Delta}(q_\varepsilon, 0100) = \{q_0, q_p, q_00, q_\varepsilon\} \]
Definition
For an NFA $M = (Q, \Sigma, \delta, q_0, F)$ and string $w \in \Sigma^*$, we say $M$ accepts $w$ iff $\hat{\Delta}(q_0, w) \cap F \neq \emptyset$.
Acceptance/Recognition

Definition
For an NFA $M = (Q, \Sigma, \delta, q_0, F)$ and string $w \in \Sigma^*$, we say $M$ accepts $w$ iff $\hat{\Delta}(q_0, w) \cap F \neq \emptyset$.

Definition
The language accepted or recognized by NFA $M$ over alphabet $\Sigma$ is $L(M) = \{ w \in \Sigma^* \mid M \text{ accepts } w \}$. 
Acceptance/Recognition

Definition
For an NFA $M = (Q, \Sigma, \delta, q_0, F)$ and string $w \in \Sigma^*$, we say $M$ accepts $w$ iff $\hat{\Delta}(q_0, w) \cap F \neq \emptyset$

Definition
The language accepted or recognized by NFA $M$ over alphabet $\Sigma$ is $L(M) = \{ w \in \Sigma^* \mid M$ accepts $w \}$. A language $L$ is said to be accepted/recognized by $M$ if $L = L(M)$.
Observations about NFAs

Observation 1
For NFA $M$, string $w$ and state $q_1$ it could be that
Observations about NFAs

**Observation 1**
For NFA $M$, string $w$ and state $q_1$ it could be that
- $\hat{\Delta}(q_1, w) = \emptyset$
Observations about NFAs

Observation 1
For NFA $M$, string $w$ and state $q_1$ it could be that
- $\hat{\Delta}(q_1, w) = \emptyset$
- $\hat{\Delta}(q_1, w)$ has more than one element
Observations about NFAs

Observation 1
For NFA $M$, string $w$ and state $q_1$ it could be that
- $\hat{\Delta}(q_1, w) = \emptyset$
- $\hat{\Delta}(q_1, w)$ has more than one element

Observation 2
However, the following proposition about DFAs continues to hold for NFAs
Observations about NFAs

Observation 1
For NFA $M$, string $w$ and state $q_1$ it could be that

- $\hat{\Delta}(q_1, w) = \emptyset$
- $\hat{\Delta}(q_1, w)$ has more than one element

Observation 2
However, the following proposition about DFAs continues to hold for NFAs

- For NFA $M$, strings $u$ and $v$, and state $q$,

\[
\hat{\Delta}(q, uv) = \bigcup_{q' \in \hat{\Delta}(q, u)} \hat{\Delta}(q', v)
\]
Using Nondeterminism

When designing an NFA for a language
Using Nondeterminism

When designing an NFA for a language
  - You follow the same methodology as for DFAs, like identifying what needs to be remembered
Using Nondeterminism

When designing an NFA for a language

- You follow the same methodology as for DFAs, like identifying what needs to be remembered
- But now, the machine can “guess” at certain steps
Problem
For $\Sigma = \{0, 1, 2, \#\}$, let

$$L = \{w\#c \mid w \in \Sigma^*, \ c \in \Sigma, \ \text{and} \ c \ \text{occurs in} \ w\}$$

So $1011\#0 \in L$ but $1011\#2 \notin L$. Design an NFA recognizing $L$. 
Problem
For $\Sigma = \{0, 1, 2, #\}$, let

$$ L = \{ w#c \mid w \in \Sigma^*, c \in \Sigma, \text{ and } c \text{ occurs in } w \} $$

So $1011#0 \in L$ but $1011#2 \notin L$. Design an NFA recognizing $L$.

Solution

- Read symbols of $w$, i.e., portion of input before $#$ is seen
- Guess at some point that current symbol in $w$ is going to be the same as ‘$c$’; store this symbol in the state
- Read the rest of $w$
- On reading $#$, check that the symbol immediately after is the one stored, and that the input ends immediately after that.
$L(M) = \{ w\#c \mid c \text{ occurs in } w \}$
Halving a Language

Definition
For a language $L$, define $\frac{1}{2}L$ as follows.

$$\frac{1}{2}L = \{x \mid \exists y. |x| = |y| \text{ and } xy \in L\}$$

In other words, $\frac{1}{2}L$ consists of the first halves of strings in $L$. 
Halving a Language

Definition
For a language $L$, define $\frac{1}{2}L$ as follows.

$$\frac{1}{2}L = \{x \mid \exists y. |x| = |y| \text{ and } xy \in L\}$$

In other words, $\frac{1}{2}L$ consists of the first halves of strings in $L$.

Example
If $L = \{001, 0000, 01, 110010\}$ then $\frac{1}{2}L = \ldots$.
Halving a Language

Definition
For a language $L$, define $\frac{1}{2}L$ as follows.

$$\frac{1}{2}L = \{x \mid \exists y. |x| = |y| \text{ and } xy \in L\}$$

In other words, $\frac{1}{2}L$ consists of the first halves of strings in $L$

Example
If $L = \{001, 0000, 01, 110010\}$ then $\frac{1}{2}L = \{00, 0, 110\}$. 
Proposition

If $L$ is recognized by a DFA $M$ then there is a NFA $N$ such that $L(N) = \frac{1}{2}L$. 

Proof Idea

On input $x$, need to check if $x$ is the first half of some string $w = xy$ that is accepted by $M$.

▶ "Run" $M$ on input $x$; let $M$ be in state $q_i$ after reading all of $x$.

▶ Guess a string $y$ such that $|y| = |x|$.

▶ Check if $M$ reaches a final state on reading $y$ from $q_i$.
Recognizing Halves of Regular Languages

Proposition

If $L$ is recognized by a DFA $M$ then there is a NFA $N$ such that $L(N) = \frac{1}{2} L$.

Proof Idea

On input $x$, need to check if $x$ is the first half of some string $w = xy$ that is accepted by $M$.

- "Run" $M$ on input $x$; let $M$ be in state $q_i$ after reading all of $x$
- Guess a string $y$ such that $|y| = |x|$
- Check if $M$ reaches a final state on reading $y$ from $q_i$
Proposition

If $L$ is recognized by a DFA $M$ then there is a NFA $N$ such that $L(N) = \frac{1}{2}L$.

Proof Idea

On input $x$, need to check if $x$ is the first half of some string $w = xy$ that is accepted by $M$.

- “Run” $M$ on input $x$; let $M$ be in state $q_i$ after reading all of $x$
- Guess a string $y$ such that $|y| = |x|$
- Check if $M$ reaches a final state on reading $y$ from $q_i$

How do you guess a string $y$ of equal length to $x$ using finite memory? Seems to require remembering the length of $x$!
Fixing the Idea

Problem

- How do you guess a string $y$ of equal length to $x$ using finite memory?

- How do you “run” $M$ on $y$ from $q_i$, if you cannot store all the symbols of $y$?

- Run $M$ on $y$ as you guess each symbol, without waiting to finish the execution on $x$!

- If we don’t first execute $M$ on $x$, how do we know the state $q_i$ from which we have to execute $y$ from?

Guess it! And then check that running $M$ on $x$ does indeed end in $q_i$, your guessed state.
Fixing the Idea

Problem and Fix(?)

- How do you guess a string $y$ of equal length to $x$ using finite memory? Guess one symbol of $y$ as you read one symbol of $x$!
Fixing the Idea

Problem and Fix(?)

- How do you guess a string $y$ of equal length to $x$ using finite memory? Guess one symbol of $y$ as you read one symbol of $x$!
- How do you “run” $M$ on $y$ from $q_i$, if you cannot store all the symbols of $y$?
Problem and Fix(?)

- How do you guess a string $y$ of equal length to $x$ using finite memory? Guess one symbol of $y$ as you read one symbol of $x$!
- How do you “run” $M$ on $y$ from $q_i$, if you cannot store all the symbols of $y$? Run $M$ on $y$ as you guess each symbol, without waiting to finish the execution on $x$!
Fixing the Idea

Problem and Fix(?)

- How do you guess a string $y$ of equal length to $x$ using finite memory? Guess one symbol of $y$ as you read one symbol of $x$!
- How do you “run” $M$ on $y$ from $q_i$, if you cannot store all the symbols of $y$? Run $M$ on $y$ as you guess each symbol, without waiting to finish the execution on $x$!
- If we don’t first execute $M$ on $x$, how do we know the state $q_i$ from which we have to execute $y$ from?
Fixing the Idea

Problem and Fix(?)

- How do you guess a string $y$ of equal length to $x$ using finite memory? Guess one symbol of $y$ as you read one symbol of $x$!
- How do you “run” $M$ on $y$ from $q_i$, if you cannot store all the symbols of $y$? Run $M$ on $y$ as you guess each symbol, without waiting to finish the execution on $x$!
- If we don’t first execute $M$ on $x$, how do we know the state $q_i$ from which we have to execute $y$ from? Guess it! And then check that running $M$ on $x$ does indeed end in $q_i$, your guessed state.
New Algorithm

On input $x$, NFA $N$

1. Guess state $q_i$ and place “left finger” on (initial state of $M$) $q_0$ and “right finger” on $q_i$

2. As characters of $x$ are read, $N$ moves the left finger along transitions dictated by $x$ and simultaneously moves the right finger along nondeterministically chosen transitions labelled by some symbol
New Algorithm

On input \( x \), NFA \( N \)

1. Guess state \( q_i \) and place “left finger” on (initial state of \( M \)) \( q_0 \) and “right finger” on \( q_i \)

2. As characters of \( x \) are read, \( N \) moves the left finger along transitions dictated by \( x \) and simultaneously moves the right finger along nondeterministically chosen transitions labelled by some symbol

3. Accept if after reading \( x \), left finger is at \( q_i \) (state initially guessed for right finger) and right finger is at an accepting state
New Algorithm

On input $x$, NFA $N$

1. Guess state $q_i$ and place “left finger” on (initial state of $M$) $q_0$ and “right finger” on $q_i$

2. As characters of $x$ are read, $N$ moves the left finger along transitions dictated by $x$ and simultaneously moves the right finger along nondeterministically chosen transitions labelled by some symbol

3. Accept if after reading $x$, left finger is at $q_i$ (state initially guessed for right finger) and right finger is at an accepting state

Things to remember: initial guess for right finger, and positions of left and right finger.
Algorithm on Example

100010 ∈ \( L \) and so \( x = 100 ∈ \frac{1}{2} L \)

DFA \( M \)
Algorithm on Example

100010 ∈ L and so x = 100 ∈ 1/2 L
NFA N execution on x = 100 is

String Read     Left Finger     Right Finger
ε               q₀              q₂
1               q₁              q₂
10              q₃              q₁
100             q₂              q₃

DFA M
Algorithm on Example

100010 ∈ L and so x = 100 ∈ $\frac{1}{2}L$

NFA N execution on x = 100 is

<table>
<thead>
<tr>
<th>String</th>
<th>Read</th>
<th>Left Finger</th>
<th>Right Finger</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\epsilon$</td>
<td>$q_0$</td>
<td>$q_2$</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>$q_1$</td>
<td>$q_2$</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>$q_3$</td>
<td>$q_1$</td>
</tr>
<tr>
<td>100</td>
<td></td>
<td>$q_2$</td>
<td>accept?</td>
</tr>
</tbody>
</table>

DFA M
Formal Construction of NFA $N$
States and Initial State

Given $M = (Q, \Sigma, \delta, q_0, F)$ recognizing $L$ define $N = (Q', \Sigma, \delta', q'_0, F')$ that recognizes $\frac{1}{2}L$

- $Q' = Q \times Q \times Q \cup \{s\}$, where $s \not\in Q$
  - $s$ is a new start state
  - Other states are of the form $\langle$left finger, initial guess, right finger$\rangle$; “initial guess” records the initial guess for the right finger
- $q'_0 = s$
Formal Construction of NFA $N$

Transitions and Final States

- **Transitions**

  \[
  \delta'(s, \epsilon) = \{ \langle q_0, q_i, q_i \rangle \mid q_i \in Q \} 
  \]

  “Guess” the state $q_i$ that the input will lead to

  \[
  \delta'(\langle q_i, q_j, q_k \rangle, a) = \{ \langle q_l, q_j, q_m \rangle \mid \delta(q_i, a) = q_l, \exists b \in \Sigma. \delta(q_k, b) = q_m \} 
  \]

  $b$ is the guess for the next symbol of $y$ and initial guess does not change

- **$F'$**

  \[
  F' = \{ \langle q_i, q_i, q_j \rangle \mid q_i \in Q, q_j \in F \} 
  \]