NFA to DFA conversion and regular expressions
CSCI 3130 Formal Languages and Automata Theory

Siu On CHAN

Chinese University of Hong Kong

Fall 2015
DFAs and NFAS are equally powerful

NFA can do everything a DFA can do
How about the other way?

Every NFA can be converted into a DFA for the same language
NFA → DFA in two easy steps

1. Eliminate $\varepsilon$-transitions
2. Convert simplified NFA to DFA
   We will do this first
NFA $\rightarrow$ DFA: intuition
NFA $\rightarrow$ DFA: intuition
NFA → DFA: states

DFA has a state for every subset of NFA states
NFA $\rightarrow$ DFA: transitions

NFA

DFA

DFA has a state for every subset of NFA states
NFA → DFA: accepting states

DFA accepts if it contains an NFA accepting state
**NFA → DFA: eliminate unreachable states**

At the end, you may eliminate *unreachable* states
## General conversion

<table>
<thead>
<tr>
<th></th>
<th>NFA</th>
<th>DFA</th>
</tr>
</thead>
<tbody>
<tr>
<td>states</td>
<td>$q_0, q_1, \ldots, q_n$</td>
<td>$\emptyset, {q_0}, {q_1}, {q_0, q_1}, \ldots,$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>${q_0, \ldots, q_n}$</td>
</tr>
<tr>
<td>initial state</td>
<td>$q_0$</td>
<td>${q_0}$</td>
</tr>
<tr>
<td>transitions</td>
<td>$\delta$</td>
<td>$\delta'({q_{i_1}, \ldots, q_{i_k}}, a) =$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\delta(q_{i_1}, a) \cup \cdots \cup \delta(q_{i_k}, a)$</td>
</tr>
<tr>
<td>accepting states</td>
<td>$F \subseteq Q$</td>
<td>$F' = {S</td>
</tr>
</tbody>
</table>
NFA → DFA in two easy steps

1. Eliminate $\varepsilon$-transitions
2. Convert simplified NFA to DFA

✓
Eliminating $\varepsilon$-transitions

NFA:  

\[ \begin{array}{c}
q_0 \\
\varepsilon,1 \\
0 \\
\varepsilon \\
\end{array} \quad \begin{array}{c}
q_1 \\
0 \\
\varepsilon \\
\end{array} \quad \begin{array}{c}
q_2 \\
\end{array} \]

NFA without $\varepsilon$’s:

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_0$</td>
<td>${q_0, q_1, q_2}$</td>
<td>${q_1, q_2}$</td>
</tr>
<tr>
<td>$q_1$</td>
<td>${q_0, q_1, q_2}$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>$q_2$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
</tr>
</tbody>
</table>

Accepting states: $q_2$, $q_1$, $q_0$
Eliminating $\varepsilon$-transitions

NFA:

- $q_0$ (start state)
- $q_1$
- $q_2$ (accept state)

Transitions:
- $\varepsilon, 1$ from $q_0$ to $q_1$
- $\varepsilon$ from $q_2$
- $0$ from $q_0$ to $q_1$
- $0$ from $q_1$ to $q_2$

New NFA:

- States: $q_0, q_1, q_2$
- Transitions:
  - $0$: $\{q_0, q_1, q_2\}$ from $q_0$ to $\emptyset$ from $q_1$ to $q_2$
  - $1$: $\{q_1, q_2\}$ from $q_0$ to $\emptyset$ from $q_1$ to $\emptyset$
Eliminating $\varepsilon$-transitions

Paths with $\varepsilon$’s are replaced with a single transition

States that can reach accepting state by $\varepsilon$ are all accepting
Regular expressions
String concatenation

\[ s = \text{abb} \]
\[ t = \text{bab} \]
\[ st = \text{abbbab} \]
\[ ts = \text{bababb} \]
\[ ss = \text{abbabb} \]
\[ sst = \text{abbabbbab} \]

\[ s = x_1 \ldots x_n, \quad t = y_1 \ldots y_m \]
\[ \downarrow \]
\[ st = x_1 \ldots x_n y_1 \ldots y_m \]
Operations on languages

- **Concatenation** of languages $L_1$ and $L_2$

  $$L_1 L_2 = \{st : s \in L_1, t \in L_2\}$$

- **n-th power** of language $L$

  $$L^n = \{s_1 s_2 \ldots s_n \mid s_1, s_2, \ldots, s_n \in L\}$$

- **Union** of $L_1$ and $L_2$

  $$L_1 \cup L_2 = \{s \mid s \in L_1 \text{ or } s \in L_2\}$$
Example

\[ L_1 = \{0, 01\} \quad L_2 = \{\varepsilon, 1, 11, 111, \ldots\} \]

\[ L_1 L_2 = \{0, 01, 011, 0111, \ldots\} \cup \{01, 011, 0111, 01111, \ldots\} \]
\[ = \{0, 01, 011, 0111, \ldots\} \]

0 followed by any number of 1s

\[ L_1^2 = \{00, 001, 010, 0101\} \quad L_2^2 = L_2 \]
\[ L_2^n = L_2 \quad \text{for any } n \geq 1 \]

\[ L_1 \cup L_2 = \{0, 01, \varepsilon, 1, 11, 111, \ldots\} \]
Operations on languages

The star of $L$ contains strings made up of zero or more chunks from $L$

\[ L^* = L^0 \cup L^1 \cup L^2 \cup \ldots \]

Example: $L_1 = \{0, 01\}$ and $L_2 = \{\varepsilon, 1, 11, 111, \ldots\}$

What is $L_1^*$? $L_2^*$?
Example

$L_1 = \{0, 01\}$

$L_1^0 = \{\varepsilon\}$

$L_1^1 = \{0, 01\}$

$L_1^2 = \{00, 001, 010, 0101\}$

$L_1^3 = \{000, 0001, 0010, 00101, 0100, 01001, 01010, 010101\}$

Which of the following are in $L_1^*$?

00100001 00110001 10010001
Example

$L_1 = \{0, 01\}$

$L_1^0 = \{\varepsilon\}$
$L_1^1 = \{0, 01\}$
$L_1^2 = \{00, 001, 010, 0101\}$
$L_1^3 = \{000, 0001, 0010, 00101, 0100, 01001, 01010, 010101\}$

Which of the following are in $L_1^*$?

- 00100001: Yes
- 00110001: No
- 10010001: No
Example

$L_1 = \{0, 01\}$

$L_1^0 = \{\varepsilon\}$
$L_1^1 = \{0, 01\}$
$L_1^2 = \{00, 001, 010, 0101\}$
$L_1^3 = \{000, 0001, 0010, 00101, 0100, 01001, 01010, 010101\}$

Which of the following are in $L_1^*$?

<table>
<thead>
<tr>
<th>String</th>
<th>$L_1^*$?</th>
</tr>
</thead>
<tbody>
<tr>
<td>00100001</td>
<td>Yes</td>
</tr>
<tr>
<td>00110001</td>
<td>No</td>
</tr>
<tr>
<td>10010001</td>
<td>No</td>
</tr>
</tbody>
</table>

$L_1^*$ contains all strings such that any 1 is preceded by a 0
Example

\[ L_2 = \{ \varepsilon, 1, 11, 111, \ldots \} \]

any number of 1s

\[ L_2^0 = \{ \varepsilon \} \]
\[ L_2^1 = L_2 \]
\[ L_2^2 = L_2 \]
\[ L_2^n = L_2 \quad (n \geq 1) \]
Example

$L_2 = \{\varepsilon, 1, 11, 111, \ldots \}$

any number of 1s

$L_2^0 = \{\varepsilon\}$

$L_2^1 = L_2$

$L_2^2 = L_2$

$L_2^n = L_2 \quad (n \geq 1)$

$L_2^* = L_2^0 \cup L_2^1 \cup L_2^2 \cup \ldots$

$= \{\varepsilon\} \cup L_2 \cup L_2 \cup \ldots$

$= L_2$

$L_2^* = L_2$
Combining languages

We can construct languages by starting with simple ones, like \{0\} and \{1\}, and combining them

\[\{0\}(\{0\} \cup \{1\})^* \quad \Rightarrow \quad 0(0 + 1)^*\]

all strings that start with 0
Combining languages

We can construct languages by starting with simple ones, like \{0\} and \{1\}, and combining them

\[
\{0\}(\{0\} \cup \{1\})^* \quad \Rightarrow \quad 0(0 + 1)^*
\]

all strings that start with 0

\[
(\{0\}\{1\}^*) \cup (\{1\}\{0\}^*) \quad \Rightarrow \quad 01^* + 10^*
\]

0 followed by any number of 1s, or 1 followed by any number of 0s
Combining languages

We can construct languages by starting with simple ones, like \{0\} and \{1\}, and combining them

\[
\{0\}(\{0\} \cup \{1\})^* \quad \Rightarrow \quad 0(0 + 1)^*
\]

all strings that start with 0

\[
(\{0\}\{1\}^*) \cup (\{1\}\{0\}^*) \quad \Rightarrow \quad 01^* + 10^*
\]

0 followed by any number of 1s, or 1 followed by any number of 0s

\[
0(0 + 1)^* \text{ and } 01^* + 10^* \text{ are regular expressions}
\]

Blueprints for combining simpler languages into complex ones
Syntax of regular expressions

A regular expression over $\Sigma$ is an expression formed by the following rules

- The symbols $\emptyset$ and $\varepsilon$ are regular expressions
- Every $a$ in $\Sigma$ is a regular expression
- If $R$ and $S$ are regular expressions, so are $R + S$, $RS$ and $R^*$

Examples:

$$
\emptyset \\
0(0 + 1)^* \\
01^* + 10^* \\
\varepsilon \\
1^*(\varepsilon + 0) \\
(0 + 1)^*01(0 + 1)^*
$$

A language is **regular** if it is represented by a regular expression
Understanding regular expressions

\[ \Sigma = \{0, 1\} \]

\[ 01^* = 0(1)^* \text{ represents } \{0, 01, 011, 0111, \ldots\} \]

0 followed by any number of 1s

\[ 01^* \text{ is not } (01)^* \]
Understanding regular expressions

- $0 + 1$ yields $\{0, 1\}$ (strings of length 1)
- $(0 + 1)^*$ yields $\{\varepsilon, 0, 1, 00, 01, 10, 11, \ldots \}$ (any string)
- $(0 + 1)^*010$ (any string that ends in 010)
- $(0 + 1)^*01(0 + 1)^*$ (any string containing 01)
Understanding regular expressions

What is the following language?

$$((0 + 1)(0 + 1))^* + ((0 + 1)(0 + 1)(0 + 1))^*$$

strings whose length is even or a multiple of 3

strings of even length

strings of length 2

strings whose length is a multiple of 3
Understanding regular expressions

What is the following language?

\(((0 + 1)(0 + 1))^* + ((0 + 1)(0 + 1)(0 + 1))^*\)

\(((0 + 1)(0 + 1))^*\) \hspace{1cm} \(((0 + 1)(0 + 1)(0 + 1))^*\)
Understanding regular expressions

What is the following language?

\[ (((0 + 1)(0 + 1))^* + ((0 + 1)(0 + 1)(0 + 1))^*) \]

\[ ((0 + 1)(0 + 1))^* \]

\[ ((0 + 1)(0 + 1)(0 + 1))^* \]

\[ (0 + 1)(0 + 1) \]

\[ (0 + 1)(0 + 1)(0 + 1) \]
Understanding regular expressions

What is the following language?

$$((0 + 1)(0 + 1))^* + ((0 + 1)(0 + 1)(0 + 1))^*$$

$$((0 + 1)(0 + 1))^*$$

$$((0 + 1)(0 + 1)(0 + 1))^*$$

$$(0 + 1)(0 + 1)$$

strings of length 2

$$(0 + 1)(0 + 1)(0 + 1)$$

strings of length 3
Understanding regular expressions

What is the following language?
\(((0 + 1)(0 + 1))^* + ((0 + 1)(0 + 1)(0 + 1))^*\)

\(((0 + 1)(0 + 1))^*\)
strings of even length

\((0 + 1)(0 + 1)\)
strings of length 2

\(((0 + 1)(0 + 1)(0 + 1))^*\)
strings whose length is a multiple of 3

\((0 + 1)(0 + 1)(0 + 1)\)
strings of length 3
Understanding regular expressions

What is the following language?

\[(0 + 1)(0 + 1))^* + ((0 + 1)(0 + 1)(0 + 1))^*\]

strings whose length is **even or a multiple of 3**

= strings of length 0, 2, 3, 4, 6, 8, 9, 10, 12, . . .

\[((0 + 1)(0 + 1))^*\]

strings of **even** length

\[(0 + 1)(0 + 1)\]

strings of length 2

\[((0 + 1)(0 + 1)(0 + 1))^*\]

strings whose length is a **multiple of 3**

\[(0 + 1)(0 + 1)(0 + 1)\]

strings of length 3
What is the following language?

$$(((0 + 1)(0 + 1) + (0 + 1)(0 + 1)(0 + 1))^*$$

stringsthatcanbe brokeninto blocks, where each block has length 2 or 3

strings of length 2

strings of length 3
Understanding regular expressions

What is the following language?

\[ ((0 + 1)(0 + 1) + (0 + 1)(0 + 1)(0 + 1))^* \]

\[ (0 + 1)(0 + 1) + (0 + 1)(0 + 1)(0 + 1) \]
Understanding regular expressions

What is the following language?

\[ ((0 + 1)(0 + 1) + (0 + 1)(0 + 1)(0 + 1))^* \]

\[ (0 + 1)(0 + 1) + (0 + 1)(0 + 1)(0 + 1) \]

\[ (0 + 1)(0 + 1) \quad (0 + 1)(0 + 1)(0 + 1) \]
Understanding regular expressions

What is the following language?

\[((0 + 1)(0 + 1) + (0 + 1)(0 + 1)(0 + 1))\]^*\n
\((0 + 1)(0 + 1) + (0 + 1)(0 + 1)(0 + 1)\)

\((0 + 1)(0 + 1)\) strings of length 2

\((0 + 1)(0 + 1)(0 + 1)\) strings of length 3
What is the following language?

$$((0 + 1)(0 + 1) + (0 + 1)(0 + 1)(0 + 1))^*$$

Strings of length 2 or 3

$$(0 + 1)(0 + 1)$$
strings of length 2

$$(0 + 1)(0 + 1)(0 + 1)$$
strings of length 3
Understanding regular expressions

What is the following language?

\(((0 + 1)(0 + 1) + (0 + 1)(0 + 1)(0 + 1))\)^{\star}

strings that can be \textit{broken into blocks}, where each block has \textit{length 2 or 3}

\((0 + 1)(0 + 1) + (0 + 1)(0 + 1)(0 + 1)\)

strings of \textit{length 2 or 3}

\((0 + 1)(0 + 1)\)

strings of length 2

\((0 + 1)(0 + 1)(0 + 1)\)

strings of length 3
Understanding regular expressions

What is the following language?

\[(0 + 1)(0 + 1) + (0 + 1)(0 + 1)(0 + 1)\]^*\]

strings that can be broken into blocks, where each block has length 2 or 3

Which are in the language?

\[\varepsilon, 1, 01, 011, 00110, 011010110\]
Understanding regular expressions

What is the following language?

\(((0 + 1)(0 + 1) + (0 + 1)(0 + 1)(0 + 1))\)^*

strings that can be broken into blocks, where each block has length 2 or 3

Which are in the language?

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(\varepsilon)</td>
<td>1</td>
<td>01</td>
<td>011</td>
<td>00110</td>
<td>011010110</td>
</tr>
<tr>
<td>✔</td>
<td>✗</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>

The regular expression represents all strings except 0 and 1
What is the following language?

$$(1 + 01 + 001)^* (\varepsilon + 0 + 00)$$
Understanding regular expressions

What is the following language?

ends in at most two 0s

\[(1 + 01 + 001)^* (\varepsilon + 0 + 00)\]
Understanding regular expressions

What is the following language?

\[ (1 + 01 + 001)^* \left( \varepsilon + 0 + 00 \right) \]

- at most two 0s between two consecutive 1s
- ends in at most two 0s
- Never three consecutive 0s

The regular expression represents strings not containing 000

Examples:

- \( \varepsilon \)
- 00
- 0110010110
- 0010010
Writing regular expressions

Write a regular expression for all strings with two consecutive 0s
Write a regular expression for all strings with two consecutive 0s

(anything)00(anything)

(0 + 1)*00(0 + 1)*