NFA to DFA conversion and regular expressions

CSCI 3130 Formal Languages and Automata Theory

Siu On CHAN
Fall 2018

Chinese University of Hong Kong
DFAs and NFAs are equally powerful

NFA can do everything a DFA can do

How about the other way?

Every NFA is equivalent to some DFA for the same language
Given an NFA, figure out

1. the initial active states
2. how the set of active states changes upon reading an input symbol
NFA $\rightarrow$ DFA example

Initial active states (before reading any input)?
NFA → DFA example

**NFA:**

- States: \( q_0, q_1, q_2 \)
- Initial active states: \( \{ q_0, q_1, q_2 \} \)
- Transitions:
  - \( q_0 \rightarrow q_1 \) on \( \epsilon, 1 \)
  - \( q_1 \rightarrow q_2 \) on \( \epsilon \)
  - \( q_1 \rightarrow q_1 \) on \( 0 \)

**Initial active states (before reading any input)?**

**Partial DFA:**

- States: \( \{ q_0, q_1, q_2 \} \)
- Initial active states: \( \{ q_0, q_1, q_2 \} \)

**How does the set of active states change?**
NFA $\rightarrow$ DFA example

Initial active states (before reading any input)?

Partial DFA:

How does the set of active states change?
NFA → DFA example

Initial active states (before reading any input)?

How does the set of active states change?
NFA → DFA example

Initial active states (before reading any input)?

How does the set of active states change?
Every DFA state corresponds to a subset of NFA states.
A DFA state is accepting if it contains an accepting NFA state.
Regular expressions
Regular expressions

Powerful string matching feature in advanced editors (e.g. Vim, Emacs) and modern programming languages (e.g. PERL, Python)

PERL regex examples:

- `colou?r` matches “color”/“colour”
- `[A-Za-z]*ing` matches any word ending in “ing”

We will learn to parse complicated regex recursively by building up from simpler ones.

Also construct the language matched by the expression recursively.

Will focus on regular expressions in formal language theory (notations differ from PERL/Python/POSIX regex)
String concatenation

\[
\begin{align*}
    s &= abb \\
    t &= bab \\
    st &= abbbab \\
    ts &= bababb \\
    ss &= abbabb \\
    sst &= abbabbbab
\end{align*}
\]

\[
\begin{align*}
    s &= x_1 \ldots x_n, \\
    t &= y_1 \ldots y_m \\
    \Downarrow \\
    st &= x_1 \ldots x_n y_1 \ldots y_m
\end{align*}
\]
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

\[ L_1 = \{0, 01\} \]

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

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

\[ = \{0, 01, 011, 0111, \ldots\} \]

0 followed by any number of 1s

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

\[ L_2^2 = L_2 \]

\[ L_2^n = L_2 \quad \text{for any } n \geq 1 \]

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

The star of $L$ are 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

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

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

<table>
<thead>
<tr>
<th>String</th>
<th>Result</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

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

$L_2^1 = L_2$

$L_2^2 = L_2$

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

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

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

$= L_2$

$L_2^* = L_2$
We can construct languages by starting with simple ones, like \{0\} and \{1\}, and combining them

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

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

\[
\{0\}(\{0\} \cup \{1\})^* \quad \Rightarrow \quad 0(0 + 1)^* \\
\text{all strings that start with 0}
\]

\[
(\{0\}\{1\}^*) \cup (\{1\}\{0\}^*) \quad \Rightarrow \quad 01^* + 10^* \\
0 \text{ followed by any number of 1s, or} \\
1 \text{ followed by any number of 0s}
\]

\(0(0 + 1)^*\) and \(01^* + 10^*\) are regular expressions

Blueprints for combining simpler languages into complex ones
A regular expression over \( \Sigma \) is an expression formed by the following rules:

- The symbols \( \emptyset \) and \( \varepsilon \) are regular expressions.
- Every symbol \( a \) 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

$(0 + 1)^* \text{ yields } \{\varepsilon, 0, 1, 00, 01, 10, 11, \ldots \}$

any string

$(0 + 1)^*010$

any string that ends in 010

$(0 + 1)^*01(0 + 1)^*$

any string containing 01
What language does the following represent?

\(((0 + 1)(0 + 1))^* + ((0 + 1)(0 + 1)(0 + 1))^*\)
Understanding regular expressions

What language does the following represent?

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

$$(((0 + 1)(0 + 1))^* + ((0 + 1)(0 + 1)(0 + 1))^*)$$
Understanding regular expressions

What language does the following represent?

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

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

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

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

$$(0 + 1)(0 + 1)(0 + 1)$$
What language does the following represent?

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

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

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

strings of length 2

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

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

strings of length 3
What language does the following represent?

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

- \(((0 + 1)(0 + 1))^*\)
  - 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 language does the following represent?

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

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
Understanding regular expressions

What language does the following represent?

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

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

\((0 + 1)(0 + 1) + (0 + 1)(0 + 1)(0 + 1)\)
What language does the following represent?

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

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

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

$$ (0 + 1)(0 + 1)(0 + 1)$$
What language does the following represent?

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

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

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

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

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

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

strings of length 2 or 3

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

strings of length 2

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

strings of length 3
What language does the following represent?

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

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

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

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 language does the following represent?

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

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

Which are in the language?

| ε | 1 | 01 | 011 | 00110 | 011010110 |
What language does the following represent?

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

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

Which are in the language?

<table>
<thead>
<tr>
<th>String</th>
</tr>
</thead>
<tbody>
<tr>
<td>ε</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>01</td>
</tr>
<tr>
<td>011</td>
</tr>
<tr>
<td>00110</td>
</tr>
<tr>
<td>011010110</td>
</tr>
</tbody>
</table>

The regular expression represents all strings except 0 and 1.
What language does the following represent?

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

What language does the following represent?

ends in at most two 0s

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

What language does the following represent?

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

\(\in\) at most two 0s

\((1 + 01 + 001)^*\)

\(\varepsilon\)

end in at most two 0s

\((\varepsilon + 0 + 00)\)

at most two 0s between two consecutive 1s

Never three consecutive 0s

The regular expression represents strings not containing 000

Examples:

\(\varepsilon\)

00

01\text{\text{\text{\text{\text{1}1}}}}00\text{\text{\text{\text{\text{1}1}}}}0\text{\text{\text{\text{\text{0}}}}}

001\text{\text{\text{\text{\text{1}0\text{\text{\text{\text{\text{1}1\text{\text{\text{\text{\text{0}}}}}}}}}}}}}

\text{20/22}
Write a regular expression for all strings with **two consecutive 0s**
Write a regular expression for all strings with **two consecutive 0s**

\[(\text{anything})00(\text{anything})\]