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?

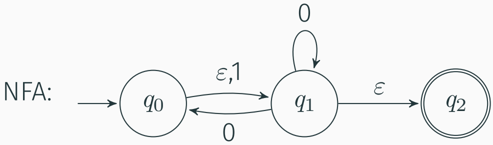
Every NFA is equivalent to some DFA for the same language

NFA $\to$ DFA algorithm

Given an NFA, figure out

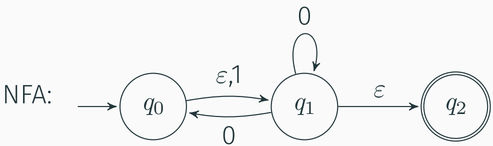
1. the initial active states
2. how the set of active states changes upon reading an input symbol

NFA → DFA example

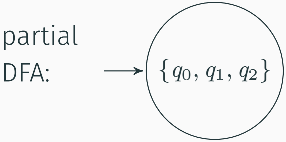


Initial active states (before reading any input)?

NFA $\to$ DFA example



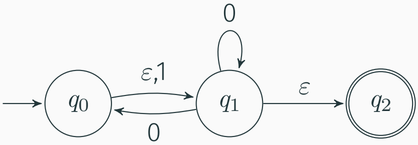
Initial active states (before reading any input)?



How does the set of active states change?

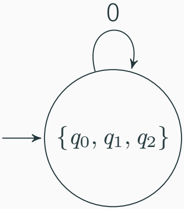
NFA → DFA example

NFA:



Initial active states (before reading any input)?

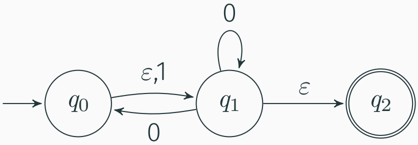
partial DFA:



How does the set of active states change?

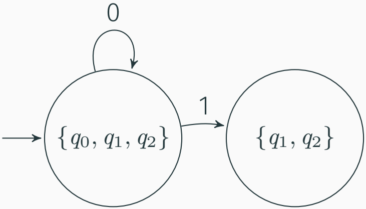
NFA $\rightarrow$ DFA example

NFA:



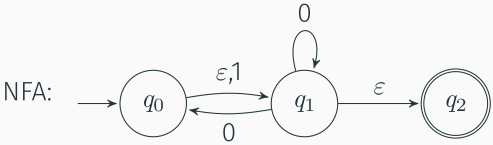
Initial active states (before reading any input)?

partial DFA:

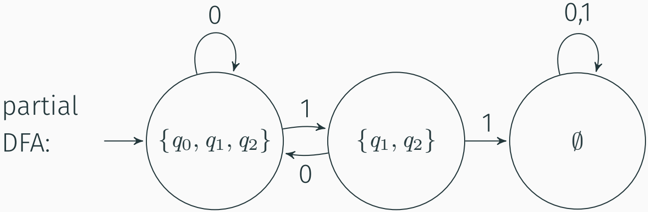


How does the set of active states change?

NFA $\rightarrow$ DFA example

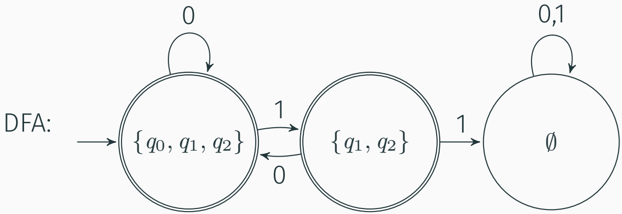


Initial active states (before reading any input)?



How does the set of active states change?

NFA $\rightarrow$ DFA summary



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

$$s = \text{abb}$$
$$t = \text{bab}$$

$$st = \text{abbbab}$$
$$ts = \text{bababb}$$
$$ss = \text{abbabb}$$
$$sst = \text{abbabbbab}$$

$$s = x_1 \dots x_n, \quad t = y_1 \dots y_m$$
$$\Downarrow$$
$$st = x_1 \dots x_n y_1 \dots y_m$$

Operations on languages

- Concantenation 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

$L_1 = \{0, 01\}$ $\quad$ $L_2 = \{\varepsilon, 1, 11, 111, \dots\}$

$$L_1 L_2 = \{0, 01, 011, 0111, \dots\} \cup \{01, 011, 0111, 01111, \dots\}$$
$$= \{0, 01, 011, 0111, \dots\}$$

0 followed by any number of 1s

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

$$L_2^2 = L_2$$
$$L_2^n = L_2 \quad \text{for any } n \geqslant 1$$

$$L_1 \cup L_2 = \{0, 01, \varepsilon, 1, 11, 111, \dots\}$$

Operations on languages

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

$$L^* = L^0 \cup L^1 \cup L^2 \cup \dots$$

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

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	00110001	10010001
Yes	No	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^*?

00100001	00110001	10010001
Yes	No	No

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

Example

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

any number of 1s

$$L_2^0 = \{\varepsilon\}$$
$$L_2^1 = L_2$$
$$L_2^2 = L_2$$
$$L_2^n = L_2 \quad (n \geqslant 1)$$

Example

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

any number of 1s

$$L_2^0 = \{\varepsilon\}$$
$$L_2^1 = L_2$$
$$L_2^2 = L_2$$
$$L_2^n = L_2 \quad (n \geqslant 1)$$

$$\begin{aligned} L_2^* &= L_2^0 \cup L_2^1 \cup L_2^2 \cup \dots \\ &= \{\varepsilon\} \cup L_2 \cup L_2 \cup \dots \\ &= L_2 \end{aligned}$$

$$L_2^* = L_2$$

Combining languages

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

$\{0\}(\{0\} \cup \{1\})^*$ $\Rightarrow$ $0(0+1)^*$
all strings that start with 0

Combining languages

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

$\{0\}(\{0\} \cup \{1\})^*$ $\Rightarrow$ $0(0+1)^*$
all strings that start with 0

$(\{0\}\{1\}^*) \cup (\{1\}\{0\}^*)$ $\Rightarrow$ $01^* + 10^*$
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

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

Blueprints for combining simpler languages into complex ones

Syntax of regular expressions

A regular expression over Σ is an expression formed by the following rules

- The symbols $\varnothing$ and ε are regular expressions
- Every symbol a in Σ is a regular expression
- If R asd S are regular expressions, so are $R + S$, RS and R^*

Examples:

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

A language is regular if it is represented by a regular expression

Understanding regular expressions

$$\Sigma = \{0, 1\}$$

$01^* = 0(1)^*$ represents $\{0, 01, 011, 0111, \dots\}$

0 followed by any number of 1s

01^* is not $(01)^*$

Understanding regular expressions

$0 + 1$ yields $\{0, 1\}$ strings of length 1

$(0 + 1)^*$ yields $\{\varepsilon, 0, 1, 00, 01, 10, 11, \dots\}$ any string

$(0 + 1)^*010$ any string that ends in 010

$(0 + 1)^*01(0 + 1)^*$ any string containing 01

Understanding regular expressions

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?

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

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

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

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

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

Understanding regular expressions

What language does the following represent?

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

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

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

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

$(0+1)(0+1)(0+1)$
strings of length 3

Understanding regular expressions

What language does the following represent?

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

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

$((0+1)(0+1)(0+1))^*$
strings whose length is a multiple of 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 whose length is even or a multiple of 3

= strings of length $0, 2, 3, 4, 6, 8, 9, 10, 12, \ldots$

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

$((0+1)(0+1)(0+1))^*$
strings whose length is a multiple of 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?

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

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?

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

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

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

Understanding regular expressions

What language does the following represent?

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

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

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

$(0+1)(0+1)(0+1)$
strings of length 3

Understanding regular expressions

What language does the following represent?

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

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

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

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

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

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	00|110	011|010|110
✓	✗	✓	✓	✓	✓

The regular expression represents all strings except 0 and 1

Understanding regular expressions

What language does the following represent?

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

Understanding regular expressions

What language does the following represent?

$$(1+01+001)^* \overbrace{(\varepsilon+0+00)}^{\text{ends in at most two 0s}}$$

Understanding regular expressions

What language does the following represent?

$$\underbrace{(1+01+001)^*}_{\text{at most two 0s between two consecutive 1s}}\ \overbrace{(\varepsilon+0+00)}^{\text{ends in at most two 0s}}$$

Never three consecutive 0s

The regular expression represents strings not containing 000

Examples:

ε 00 011001 0110 0010010

Writing regular expressions

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

Writing regular expressions

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

(anything)00(anything)

$$(0+1)^*00(0+1)^*$$