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?

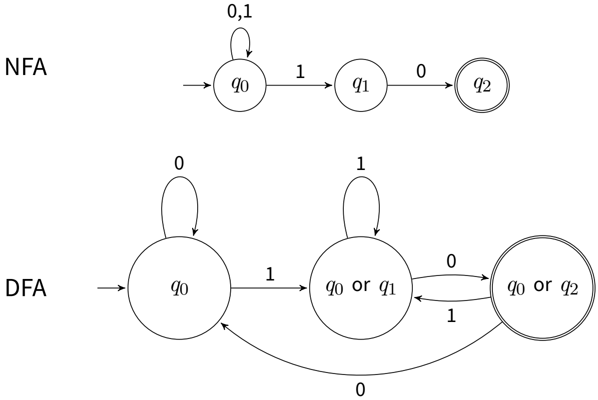
Every NFA can be converted into a DFA for the same language

NFA $\longrightarrow$ DFA in two easy steps

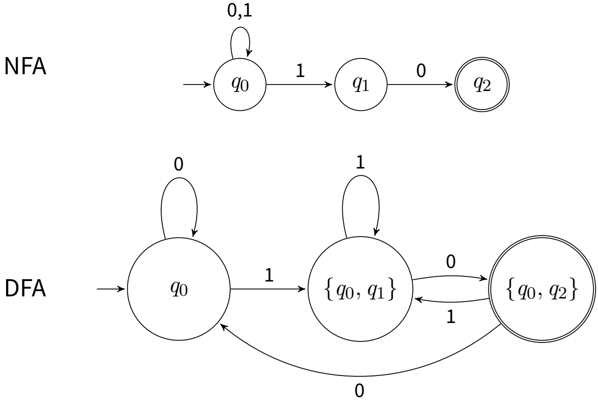
1. Eliminate ε-transitions
2. Convert simplified NFA to DFA

   We will do this first

NFA $\longrightarrow$ DFA: intuition



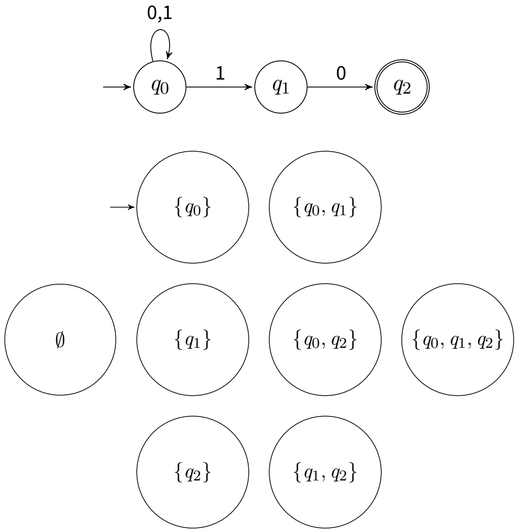
NFA $\longrightarrow$ DFA: intuition



NFA $\longrightarrow$ DFA: states

NFA

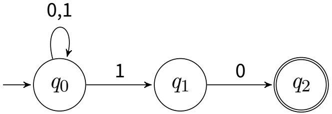
DFA



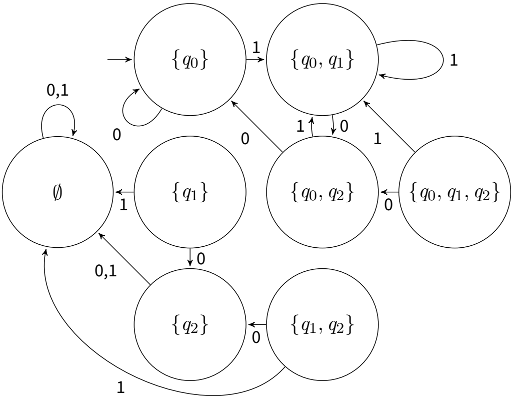
DFA has a state for every subset of NFA states

NFA $\longrightarrow$ DFA: transitions

NFA



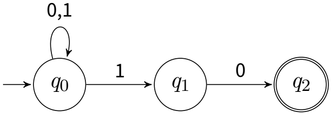
DFA



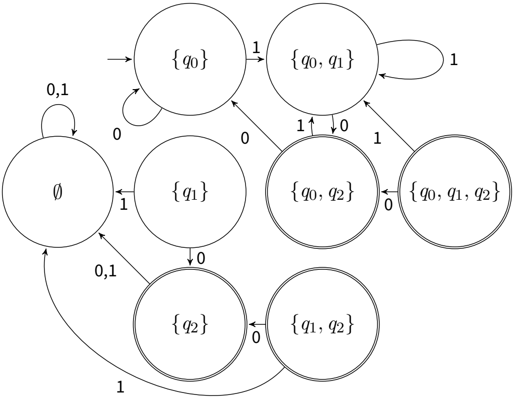
DFA has a state for every subset of NFA states

NFA $\longrightarrow$ DFA: accepting states

NFA



DFA

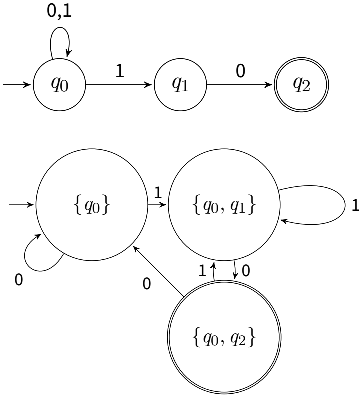


DFA accepts if it contains an NFA accepting state

NFA $\longrightarrow$ DFA: eliminate unreachable states

NFA

DFA



At the end, you may eliminate unreachable states

General conversion

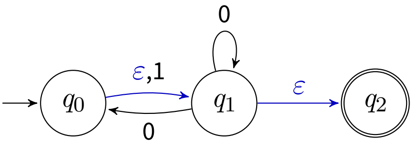
	NFA	DFA
states	$q_0, q_1, \ldots, q_n$	$\emptyset, \{q_0\}, \{q_1\}, \{q_0, q_1\}, \ldots,$ $\{q_0, \ldots, q_n\}$ one for each subset of states
initial state	q_0	$\{q_0\}$
transitions	δ	$\delta'(\{q_{i_1}, \ldots, q_{i_k}\}, a) =$ $\delta(q_{i_1}, a) \cup \cdots \cup \delta(q_{i_k}, a)$
accepting states	$F \subseteq Q$	$F' = \{S \mid S$ contains some state in $F\}$

NFA $\longrightarrow$ DFA in two easy steps

1. Eliminate ε-transitions
2. Convert simplified NFA to DFA ✓

Eliminating ε-transitions

NFA:



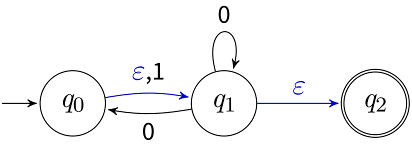
NFA without ε's:

	0	1
q_0	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$
q_1	$\{q_0, q_1, q_2\}$	$\emptyset$
q_2	$\emptyset$	$\emptyset$

Accepting states: q_2, q_1, q_0

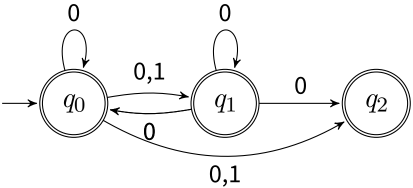
Eliminating ε-transitions

NFA:



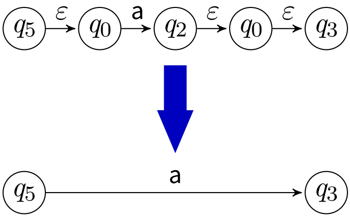
new NFA:

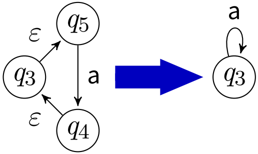
	0	1
q_0	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$
q_1	$\{q_0, q_1, q_2\}$	$\emptyset$
q_2	$\emptyset$	$\emptyset$



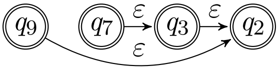
Eliminating ε-transitions

Paths with ε's are replaced with a single transition





States that can reach accepting state by ε are all accepting



Regular expressions

String concatenation

$$s = \mathsf{abb} \qquad t = \mathsf{bab}$$

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

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

Operations on languages

- 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

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

- 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

$L_1 = \{0, 01\}$ $\qquad$ $L_2 = \{\varepsilon, 1, 11, 111, \ldots\}$

$$\begin{aligned} L_1 L_2 &= \{0, 01, 011, 0111, \ldots\} \cup \{01, 011, 0111, 01111, \ldots\} \\ &= \{0, 01, 011, 0111, \ldots\} \end{aligned}$$

0 followed by any number of 1s

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

$L_2^2 = L_2$

$L_2^n = L_2$ for any $n \geqslant 1$

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

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	00110001	10010001
Yes	No	No

Example

$$L_1 = \{\mathsf{0}, \mathsf{01}\}$$

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

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

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

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

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

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

$$L_2^* = L_2$$

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

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

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

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

$0(0+1)^*$ 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 $\emptyset$ and ε are regular expressions
- Every a in Σ is a regular expression
- If R asd S are regular expressions, so are $R + S$, RS and R^*

Examples:

$$\emptyset \qquad \varepsilon$$

$$0(0+1)^* \qquad 1^*(\varepsilon+0)$$

$$01^* + 10^* \qquad (0+1)^*01(0+1)^*$$

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

Understanding regular expressions

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

$01^* = 0(1)^*$ represents $\{0, 01, 011, 0111, \ldots\}$
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 $\{\varepsilon, 0, 1, 00, 01, 10, 11, \ldots\}$ — 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?

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

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?

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

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

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

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

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

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

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

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 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 $0, 2, 3, 4, 6, 8, 9, 10, 12, \ldots$

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

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 is the following language?

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

Understanding regular expressions

What is the following language?

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

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

Understanding regular expressions

What is the following language?

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

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

$$(0+1)(0+1) \qquad\qquad (0+1)(0+1)(0+1)$$

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?

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

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

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?

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

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 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?

ε	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?

ε	1	01	011	00110	011010110
✓	✗	✓	✓	✓	✓

The regular expression represents all strings except 0 and 1

Understanding regular expressions

What is the following language?

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

Understanding regular expressions

What is the following language?

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

Understanding regular expressions

What is the following language?

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

Never three consecutive 0s

The regular expression represents strings not containing 000

Examples:

ε 00 0110010110 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)

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