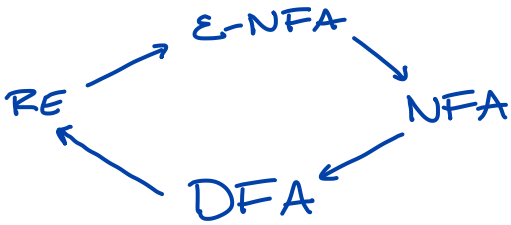
CSC 404 - Foundations of Computation

Section 1.4 – Regular Languages



Language Recognition

Definition 1.4.1.

A language is called a **regular language** if some finite automaton recognizes it.

Remark 1.4.2.

We have seen that finite-state automata can be used as language recognizers. What sets can be recognized by these machines? Although this seems like an extremely difficult problem, there is a simple characterization of the sets that can be recognized by finite state automata.

Language Recognition

Definition 1.4.1.

A language is called a **regular language** if some finite automaton recognizes it.

Remark 1.4.2.

We have seen that finite-state automata can be used as language recognizers. What sets can be recognized by these machines? Although this seems like an extremely difficult problem, there is a simple characterization of the sets that can be recognized by finite state automata.

Theorem 1.4.3 (Kleene's Theorem).

A language is regular if and only if some regular expression describes it.

Remark 1.4.4.

Via concatenation we can alternatively define Σ^* as

$$\Sigma^* = \{a_1 a_2 a_3 \dots a_k \mid k \geq 0 \text{ and each } a_i \in \Sigma\}.$$

Thus,

$$\begin{aligned}\Sigma^* &= \{\text{set of all strings over an alphabet } \Sigma\} \\ &= \{a_1 a_2 a_3 \dots a_k \mid k \geq 0 \text{ and each } a_i \in \Sigma\}.\end{aligned}$$

$0^* = \{\phi, 0, 00, 000, 0000, \dots\}$

$\{0,1\}^* = \{\phi, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100, 101, 110, 111, 0000, \dots\}$

Remark 1.4.4.

Via concatenation we can alternatively define Σ^* as

$$\Sigma^* = \{a_1 a_2 a_3 \dots a_k \mid k \geq 0 \text{ and each } a_i \in \Sigma\}.$$

Thus,

$$\begin{aligned}\Sigma^* &= \{\text{set of all strings over an alphabet } \Sigma\} \\ &= \{a_1 a_2 a_3 \dots a_k \mid k \geq 0 \text{ and each } a_i \in \Sigma\}.\end{aligned}$$

Remark 1.4.5.

Often we wish to exclude the empty string from consideration, we denote the set of all strings of length at least one by Σ^+. We see that

No empty string!

$$\Sigma^+ = \{a_1 a_2 a_3 \dots a_k \mid k \geq 1 \text{ and each } a_i \in \Sigma\}.$$

$\{0\}^+ = \{0, 00, 000, \dots\}$

$\{0,1\}^+ = \{0, 1, 00, 01, 10, 11, \dots\}$

Definition 1.4.6.

Let A and B be languages. We define the regular operations **union**, **concatenation**, and **star** as follows:

- **Union**: $A \cup B = \{x \mid x \in A \text{ or } x \in B\}$
- **Concatenation**: $AB = \{xy \mid x \in A \text{ and } y \in B\}$
- **Star**: $A^* = \{x_1 x_2 \dots x_k \mid k \geq 0 \text{ and each } x_i \in A\}$

IDEA: A = English Language = {yay, yes, cyber, ... }

B = Spanish Language = {si, cybernético, ... }

Definition 1.4.6.

Let A and B be languages. We define the regular operations **union**, **concatenation**, and **star** as follows:

- **Union**: $A \cup B = \{x \mid x \in A \text{ or } x \in B\}$
- **Concatenation**: $AB = \{xy \mid x \in A \text{ and } y \in B\}$
- **Star**: $A^* = \{x_1 x_2 \ldots x_k \mid k \geq 0 \text{ and each } x_i \in A\}$

IDEA: A = English Language = {yay, yes, cyber, ... }

B = Spanish Language = {sí, cybernético, ... }

yaysí yessí cybersí yaycybernético

all belong to AB

Definition 1.4.6.

Let A and B be languages. We define the regular operations **union**, **concatenation**, and **star** as follows:

- **Union**: $A \cup B = \{x \mid x \in A \text{ or } x \in B\}$
- **Concatenation**: $AB = \{xy \mid x \in A \text{ and } y \in B\}$
- **Star**: $A^* = \{x_1 x_2 \dots x_k \mid k \geq 0 \text{ and each } x_i \in A\}$

Example 1.4.7.

Let $\Sigma = \{0, 1\}$. Define the languages A, B, C, and D over Σ by

$$A = \{0\},\ B = \{1\},\ C = \{00, 11\}, \text{ and } D = \{01, 11\}.$$

$A \cup B = \{0\} \cup \{1\} = \{0, 1\}$

$A \cup C = \{0, 00, 11\}$

$C \cup D = \{00, 01, 11\}$

Definition 1.4.6.

Let A and B be languages. We define the regular operations **union**, **concatenation**, and **star** as follows:

- **Union**: $A \cup B = \{x \mid x \in A \text{ or } x \in B\}$
- **Concatenation**: $AB = \{xy \mid x \in A \text{ and } y \in B\}$
- **Star**: $A^* = \{x_1 x_2 \dots x_k \mid k \geq 0 \text{ and each } x_i \in A\}$

Example 1.4.7.

Let $\Sigma = \{0, 1\}$. Define the languages A, B, C, and D over Σ by

$$A = \{0\},\ B = \{1\},\ C = \{00, 11\}, \text{ and } D = \{01, 11\}.$$

$AB = \{0\}\{1\} = \{01\}$

$AC = \{000, 011\}$

$CD = \{0001, 0011, 1101, 1111\}$

Definition 1.4.6.

Let A and B be languages. We define the regular operations **union**, **concatenation**, and **star** as follows:

- **Union**: $A \cup B = \{x \mid x \in A \text{ or } x \in B\}$
- **Concatenation**: $AB = \{xy \mid x \in A \text{ and } y \in B\}$
- **Star**: $A^* = \{x_1 x_2 \dots x_k \mid k \geq 0 \text{ and each } x_i \in A\}$

Example 1.4.7.

Let $\Sigma = \{0, 1\}$. Define the languages A, B, C, and D over Σ by

$$A = \{0\},\ B = \{1\},\ C = \{00, 11\}, \text{ and } D = \{01, 11\}.$$

$A^* = \{0\}^* = \{\phi, 0, 00, 000, 0000, \dots\}$

$C^* = \{00, 11\}^* = \{\phi, 00, 11, 0000, 0011, 1100, 1111, 000000, 001111, \dots\}$

Definition 1.4.6.

Let A and B be languages. We define the regular operations **union**, **concatenation**, and **star** as follows:

- **Union**: $A \cup B = \{x \mid x \in A \text{ or } x \in B\}$
- **Concatenation**: $AB = \{xy \mid x \in A \text{ and } y \in B\}$
- **Star**: $A^* = \{x_1 x_2 \ldots x_k \mid k \geq 0 \text{ and each } x_i \in A\}$

Example 1.4.7.

Let $\Sigma = \{0, 1\}$. Define the languages A, B, C, and D over Σ by

$$A = \{0\}, B = \{1\}, C = \{00, 11\}, \text{and } D = \{01, 11\}.$$

Short hand: $= 0$ $= 1$ $= 00 \mid 11$ $= 01 \mid 11$

$A \cup B = A \mid B = \{0, 1\} = (0 \mid 1)$ OR

$AB = \{0\}\{1\} = \{01\} = 01$

Definition 1.4.8.

The regular expressions are defined recursively – we say R is a regular expression if R is

1. $\emptyset$ (the empty set)
2. The set $\{\emptyset\}$ (the set containing the empty string)
3. The set $\{a\}$ for some a in the alphabet Σ (The set containing a single string a).
4. $R_1 \cup R_2$, where R_1 and R_2 are regular expressions. (Union of two regular expressions.)
5. $R_1 R_2$, where R_1 and R_2 are regular expressions. (Concatenation of two regular expressions.)
6. R_1^*, where R_1 is a regular expression.

Example 1.4.9.

Determine whether the string 101010 is in each of these sets.

a. $\{0, 1\}^*$

b. $(01)^*$

c. $(10)^*$

d. 1^*0^*

e. 10^*

f. 01^*

Example 1.4.9.

Determine whether the string 101010 is in each of these sets.

a. $\{0,1\}^*$ YES! $\{0,1\}^* =$ All Bit strings!

b. $(01)^*$ No! $(01)^* = \{\phi, 01, 0101, 010101, \ldots\}$

c. $(10)^*$ YES! $(10)^* = \{\phi, 10, 1010, 101010, 10101010, \ldots\}$

d. 1^*0^* No! $1^*0^* = \{\phi, 1, 0, 10, 110, 100, 111, 000, \ldots\}$

e. 10^* No! $10^* = \{1, 10, 100, 1000, \ldots\}$

f. 01^* No! $01^* = \{0, 01, 011, 0111, \ldots\}$

Example 1.4.9.

Determine whether the string 101010 is in each of these sets.

a. $\{0,1\}^*$

b. $(01)^*$

c. $(10)^*$

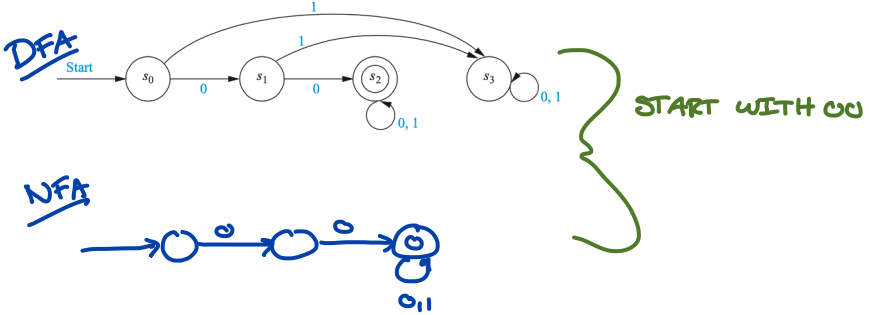
d. 1^*0^*

e. 10^*

f. 01^*

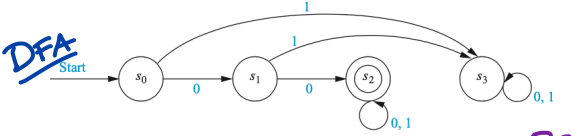
Example 1.4.10.

Find the language recognized by the given finite-state automaton.



Example 1.4.10.

Find the language recognized by the given finite-state automaton.



NFA

RE

$00\{0,1\}^*$

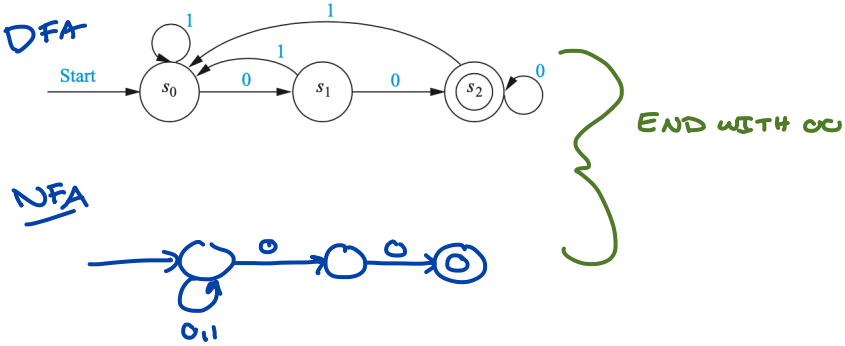
$00\Sigma^*$

$00(0|1)^*$

$00[0-1]^*$

Example 1.4.11.

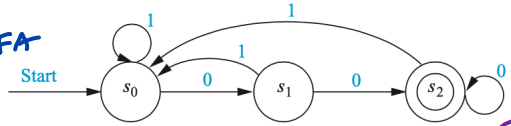
Find the language recognized by the given finite-state automaton.



Example 1.4.11.

Find the language recognized by the given finite-state automaton.

DFA



RE:

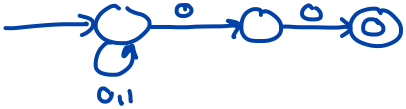
$\{0,1\}^*00$

Σ^*00

$(0|1)^*00$

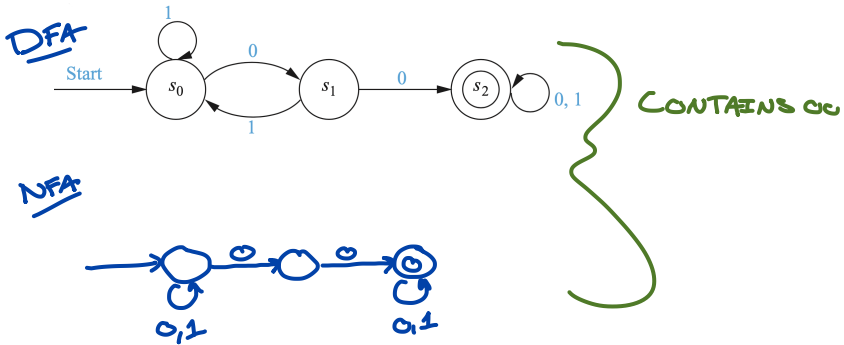
$[0-1]^*00$

NFA



Example 1.4.12.

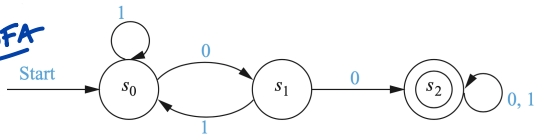
Find the language recognized by the given finite-state automaton.



Example 1.4.12.

Find the language recognized by the given finite-state automaton.

DFA



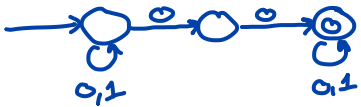
RE: $\{0,1\}^*00\{0,1\}^*$

$\Sigma^*00\Sigma^*$

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

$[0-1]^*00[0-1]^*$

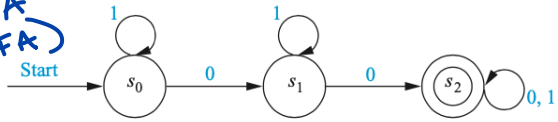
NFA



Example 1.4.13.

Find the language recognized by the given finite-state automaton.

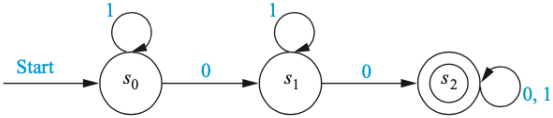
DFA (NFA)



CONTAINS TWO 0s

Example 1.4.13.

Find the language recognized by the given finite-state automaton.



RE: $1^*01^*0\{0,1\}^*$

$1^*01^*0\Sigma^*$

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

$1^*01^*0[0-1]^*$

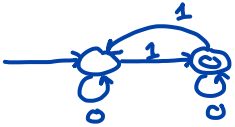
Example 1.4.14 (Bit String with Odd Parity).

Consider the language of all bit strings with an odd number of 1s. (e.g., `111`, `010`, `0001`, `101010`)

Example 1.4.14 (Bit String with Odd Parity).

Consider the language of all bit strings with an odd number of 1s. (e.g., `111`, `010`, `0001`, `101010`)

DFA:



ACCEPT:
111
100
010
10101

REJECT:
110
011
1100
1111

$$L(M) = 0^*10^*(10^*10^*)^*$$

$$= 0^*10^*((10^*)\{2^3\})^*$$

Example 1.4.14 (Bit String with Odd Parity).

Consider the language of all bit strings with an odd number of 1s. (e.g., `111`, `010`, `0001`, `101010`)

Example 1.4.15 (Bit String with Odd Parity).

Consider the language of all bit strings with an even number of 0s and at most one 1. (e.g., `010`, `00001`, `00`, `10000`)

ACCEPT

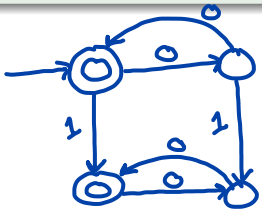
010
001
100
1
00100
01000

REJECT:

11
101
0100
000

Example 1.4.15 (Bit String with Odd Parity).

Consider the language of all bit strings with an even number of 0s and at most one 1. (e.g., `010`, `00001`, `00`, `10000`)



ACCEPT

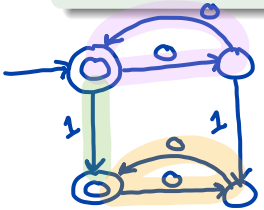
010
001
100
1
00100
01000

REJECT:

11
101
0100
000

Example 1.4.15 (Bit String with Odd Parity).

Consider the language of all bit strings with an even number of 0s and at most one 1. (e.g., `010`, `00001`, `00`, `10000`)



(00)*

(00)*1(00)*

(00)*010(00)*

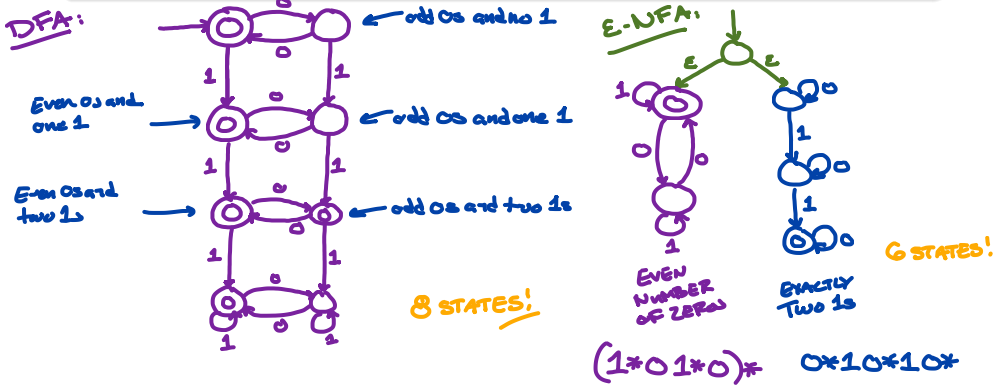
RE: (00)*010(00)* | (00)*1(00)* | (00)*

(00)*(010|1|)(00)*

(00)*(010|1)?(00)*

Example 1.4.16 (Even number of `0`s or exactly two `1`s).

Consider the language of all bit strings that contain an even number of `0`s or contain exactly two `1`s.



Example 1.4.16 (Even number of `0`s or exactly two `1`s).

Consider the language of all bit strings that contain an even number of `0`s or contain exactly two `1`s.

RE: $(1^*01^*0)^* \mid 0^*10^*10^*$

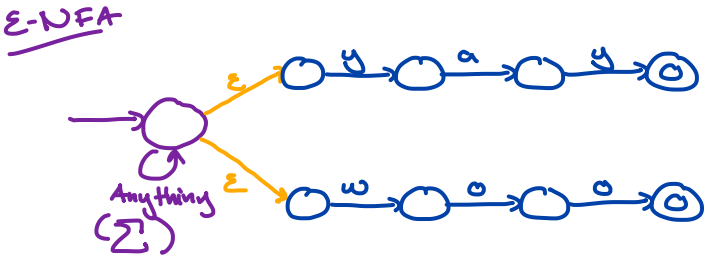
$((1^*0)^{\{2\}})^* \mid 0^*(10^*)^{\{2\}}$

Example 1.4.16 (Even number of `0`s or exactly two `1`s).

Consider the language of all bit strings that contain an even number of `0`s or contain exactly two `1`s.

Example 1.4.17 (Things that end with `yay` or `woo`).

Consider the language of all strings that end with `yay` or `woo`.



Example 1.4.17 (Things that end with `yay` or `woo`).

Consider the language of all strings that end with `yay` or `woo`.

RE $\{a,b,\ldots,z\}^*(yay|woo)$ $\Sigma = \{a,b,\ldots,z\}$

$[a-z]^*(yay|woo)$

If Σ is any character (except new line)

$.^*(yay|woo)$

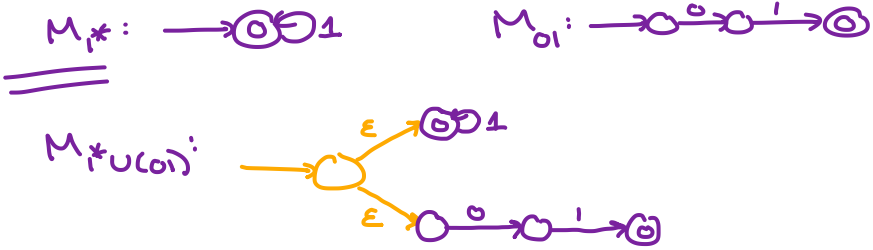
If Σ is any word character $\backslash w^*(yay|woo)$

Example 1.4.18.

Convert the regular expression $1^* \cup (01)$ to an NFA.

Example 1.4.18.

Convert the regular expression $1^* \cup (01)$ to an NFA.



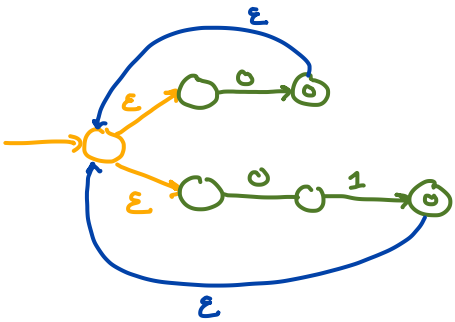
Example 1.4.19.

Convert the regular expression $(01 \cup 0)^*$ to an NFA.

Example 1.4.19.

Convert the regular expression $(01 \cup 0)^*$ to an NFA.

$(01 \cup 0)^* = (01|0)^*$



Kleene's Theorem

Theorem 1.4.22.

A language is regular if and only if some regular expression describes it.

Definition 1.4.23.

The regular expressions are defined recursively – we say R is a regular expression if R is

1. $\emptyset$ (the empty set)
2. The set $\{\emptyset\}$ (the set containing the empty string)
3. The set $\{a\}$ for some a in the alphabet Σ (The set containing a single string a).
4. $R_1 \cup R_2$, where R_1 and R_2 are regular expressions. (Union of two regular expressions.)
5. $R_1 R_2$, where R_1 and R_2 are regular expressions. (Concatenation of two regular expressions.)
6. R_1^*, where R_1 is a regular expression.

Theorem 1.4.22.

A language is regular if and only if some regular expression describes it.