DISCRETE MATHEMATICS

LECTURE 10

Plan:

Deterministic Finite Automata

(DFA)

Reference:

Section 1.1 of book:

Introduction to the Theory of Computation

(third edition)

by Michael Sipser

First: finishing the exercises on regular expressions done in the last lecture.

Example 3:

Write regular expressions for the following:

$\Sigma = \{a, b\}$

done

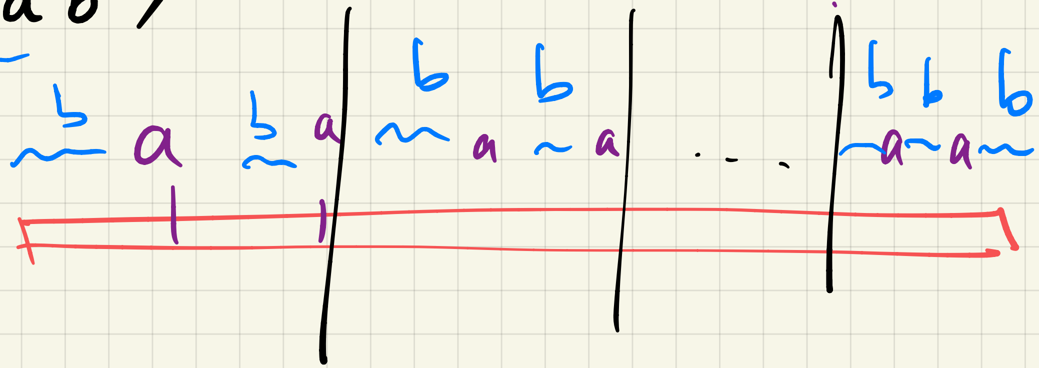
3.1. Words that do not contain 'ab'

3.2. Words with even length - $[(a+b).(a+b)]^*$

3.3. Words that contain even no. of 'a's.

Words containing even no. of 'a's

$$b^* + (b^* a b^* a b^*)^*$$



Example 4: Write the language corresponding to the foll. expr.

4.1. $ab + \epsilon$ $\quad \{ab, \epsilon\}$

4.2. $ab + \Phi$ $\quad \{ab\}$

4.3. $a \cdot \epsilon$ $\quad \{a\}$

4.4. $a \cdot \Phi$ $\quad \{\ \}$

4.5. $(a + \epsilon) \cdot b$ $\quad \{ab, b\}$

4.6. Φ^* $\quad \{\epsilon\}$

Coming next: Finite automata

Application: Finite automata can be used to construct an algorithm to search for a pattern matching a given regular expression in a string.

Fix an alphabet $\Sigma = \{a, b\}$

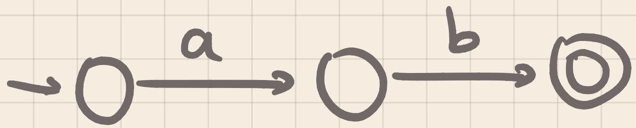
Language : is a subset of Σ^*

Regular expressions can be used to describe languages in a finite notation.

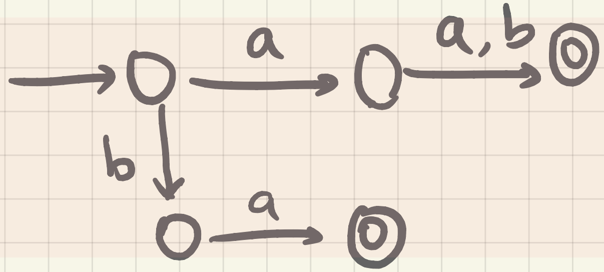
Finite automata: are yet another formalism to represent languages in a finite way.

Examples Of DFA: Guess why they represent these languages.

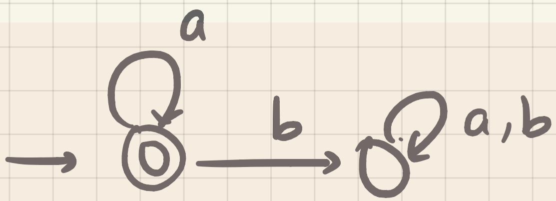
ab :



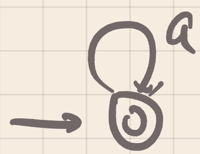
$ab + aa + ba$:



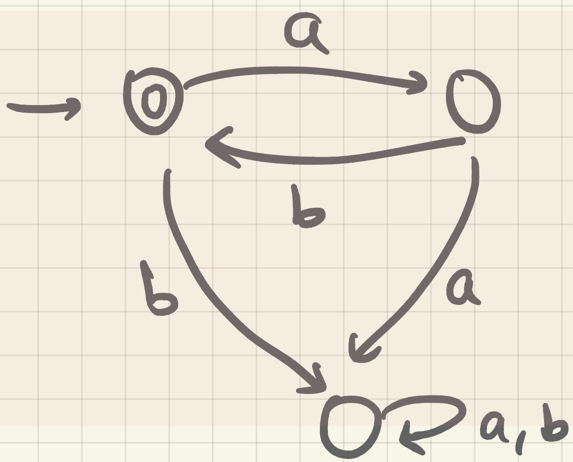
a^* :



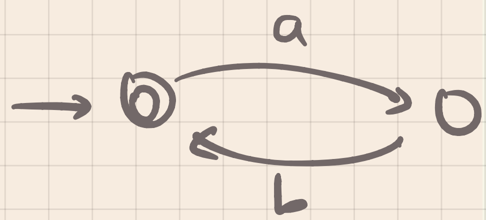
(or)



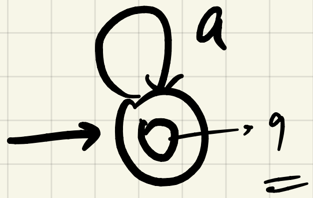
$(ab)^*$:



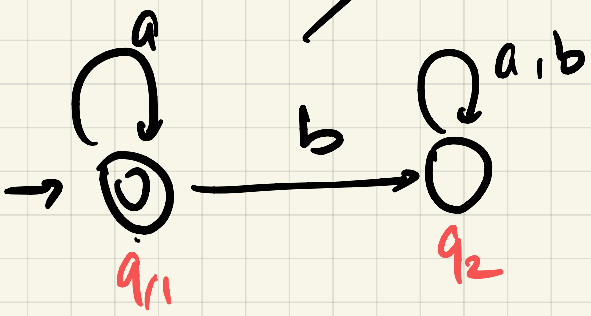
(or)



a^*



$\{\epsilon, a, aa, aaa, \ldots\}$



b a ✗

q_1 q_2 q_2

non-acc.

a a a ✓

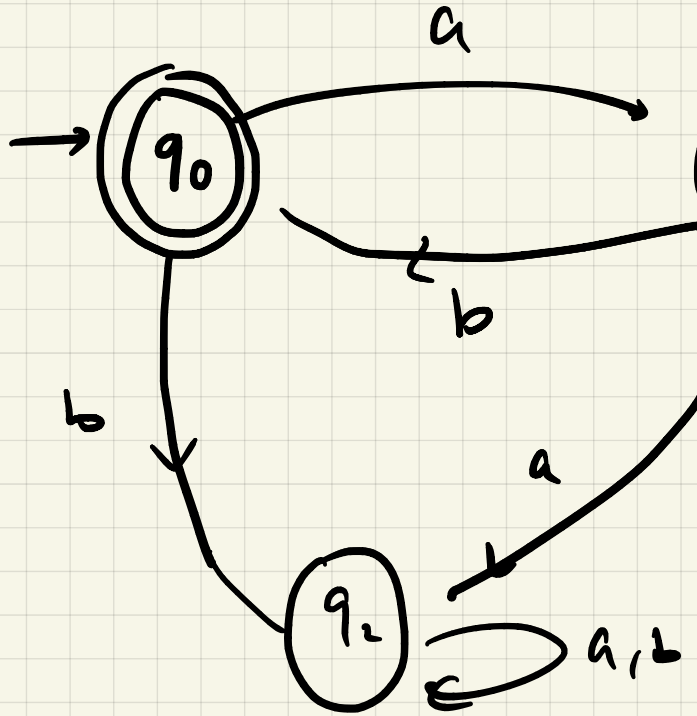
q q q q

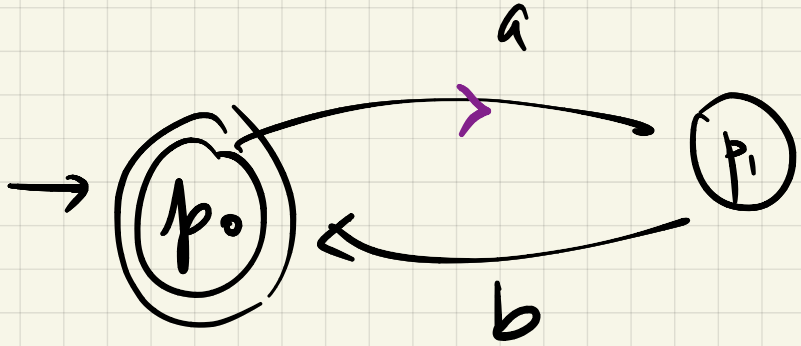
b a ✗

$(ab)^*$

$\{\epsilon, ab, abab, \dots\}$

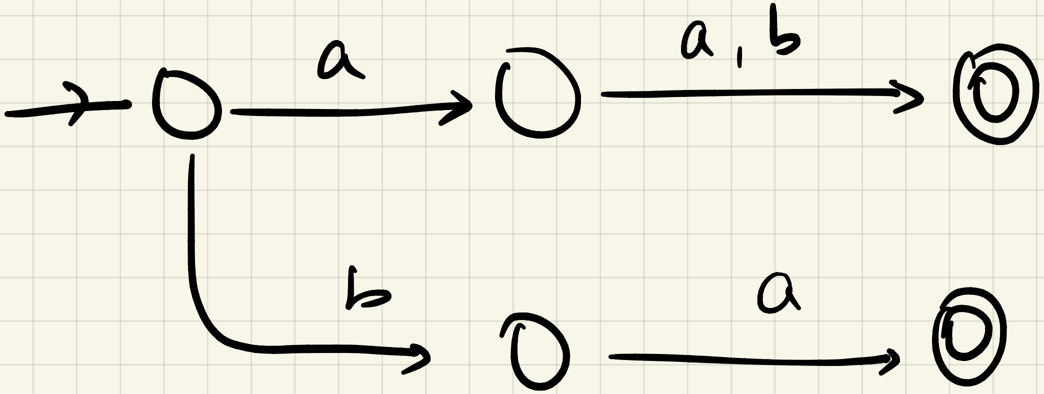
bab





$ab + aa + ba$

$\{ab, aa, ba\}$



Example 1: Draw DFA for the following languages:

1.1. $a\Sigma^*$

1.2. $a\Sigma^* b$

1.3. $\Sigma^* ab \Sigma^*$

1.4 Even no. of a's

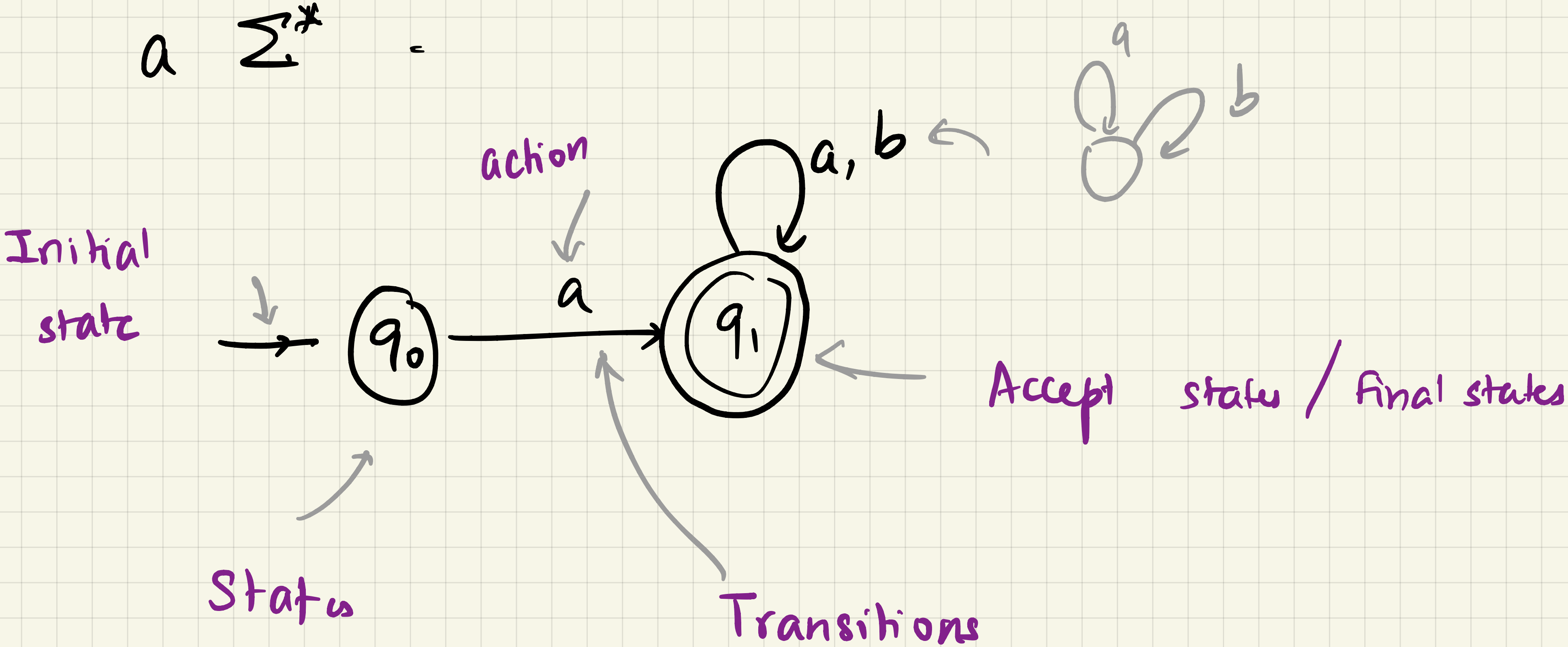
1.5 No. of a's is a multiple of 3

1.6. $b^* a^*$

1.7. $(a+b)^*$

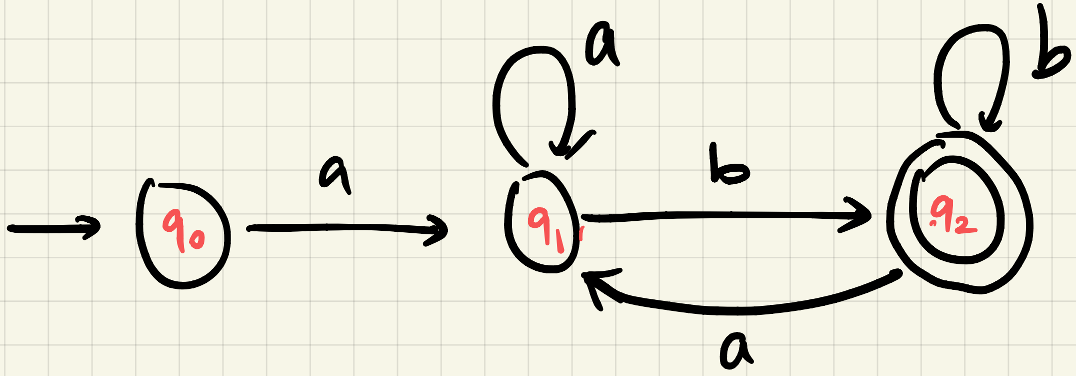
1.8. $b^* a + a^* b$

$a \quad \Sigma^*$



Initial state
action
a
q_0
q_1
a, b
a
b
Accept states / final states
States
Transitions

$a \Sigma^* b$



abab ✓

$q_0 \; q_1 \; q_2 \; q_1 \; q_2$

ba ✗

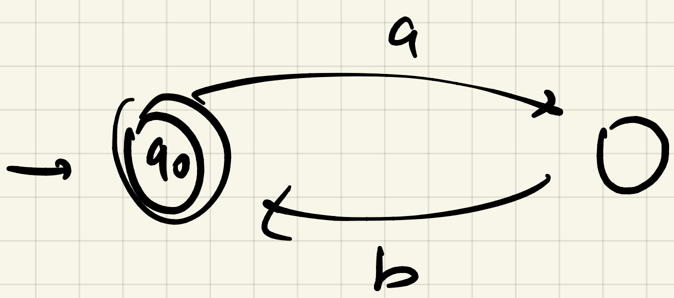
q_0

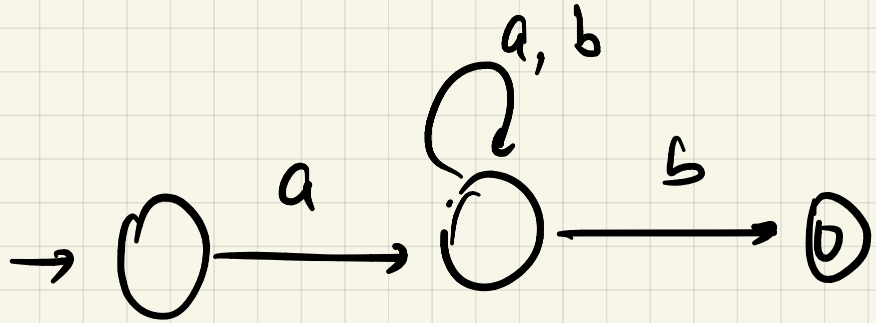
aaba ✗

$q_0 \; q_1 \; q_1 \; q_2 \; q_1$

Deterministic - Finite automaton:

From every state, on a letter, there is a deterministic choice.

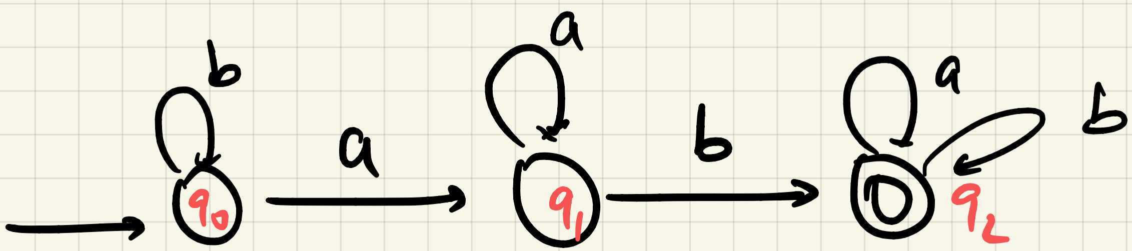




→ later

multiple choices at a state on a letter.

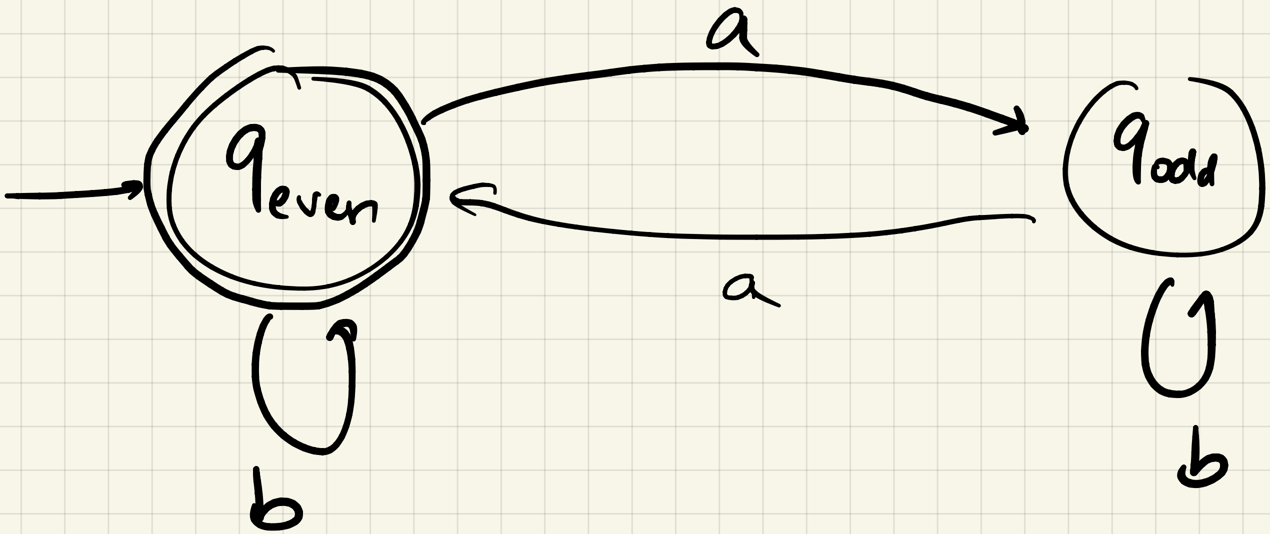
$\Sigma^* \, ab \, \Sigma^*$



b b a a a b a b

$q_0 \quad q_0 \quad q_1 \quad q_1 \quad q_1 \quad q_2 \quad q_2 \quad q_2$

Even no. of a's:



b a b a a a a

q_e q_e q_o q_o q_e q_o q_e q_o

$\delta(q_{even}, a) = q_{odd}$

$\delta(q_{odd}, b) = q_{odd}$

DETERMINISTIC FINITE AUTOMATA: - SYNTAX :

- Q : a finite set of states
- q_0 : an initial state.
- Σ : a finite alphabet
- δ : transition function

$$\delta : Q \times \Sigma \rightarrow Q$$

(complete)

(partial)

- F : a set of accepting states

Semantics of DFA:

What is the language accepted by a given DFA?

$$(Q, q_0, \Sigma, \delta, F)$$

Run of a DFA $\mathcal{A}$ on a word w:

$$w = \quad w_1 \quad w_2 \quad w_3 \quad \cdots \quad w_n$$

$$q_0 \quad q_1 \quad q_2 \quad \cdots \quad q_n$$

$q_i \xrightarrow{w_{i+1}} q_{i+1}$ is a transition.

Accepting run: A run is accepting if the final state is in F

Complete DFA: Transition function is complete.

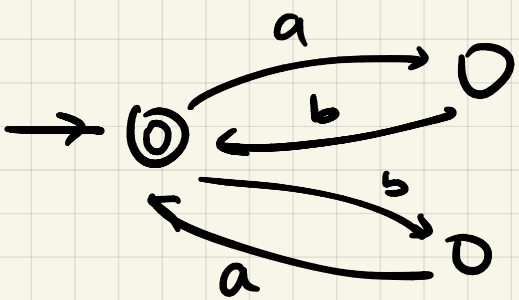
Given a word w: a DFA either has a run or not (in case the transition function is partial)

Language of a DFA: is the set of words that are accepted by the DFA.

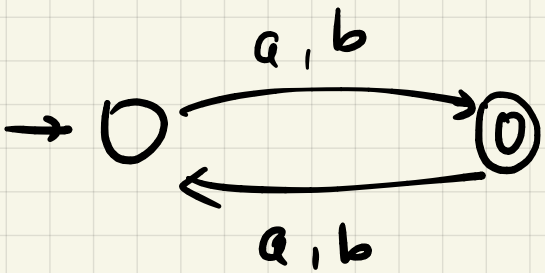
- A word w is accepted by DFA $\mathcal{A}$, if $\mathcal{A}$ has an accepting run on w.

Example 2: What is the language of the following DFA? (Exercise)

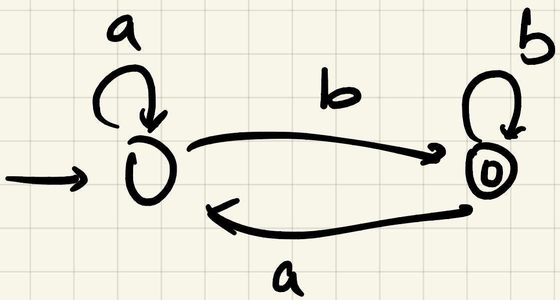
2.1.



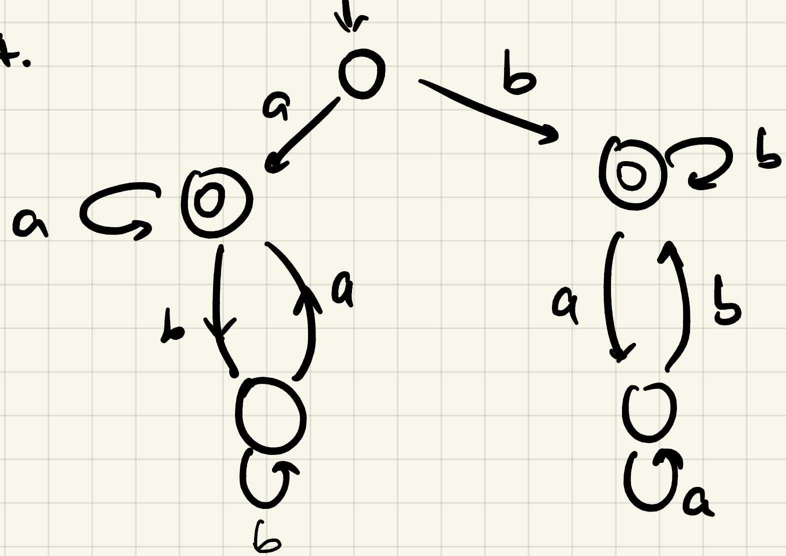
2.2.



2.3.



2.4.



Summary:

- DFA as a representation for languages
- Syntax, Semantics, examples