Monash University
Faculty of Information Technology

FIT2014 Theory of Computation

Lecture 7

Finite Automata

slides by Graham Farr
based in part on previous slides by David Albrecht

Overview

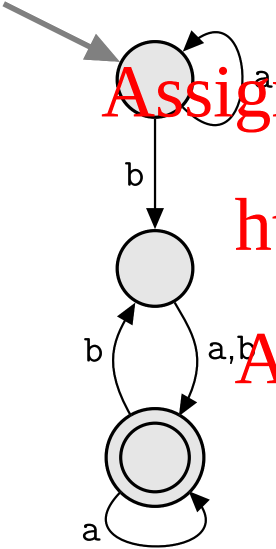
- Definition
- How they are used to define languages

- Representations
- Complement Languages
- Comparison with Regular Expressions

Finite Automaton (FA)

- Sometimes known as a **Deterministic Finite Automaton (DFA)**.
- Used for determining whether a word does or does not belong to a Regular Language.
- Used for defining a Regular Language.
- Used in Lexical Analysers.

Notation and terminology

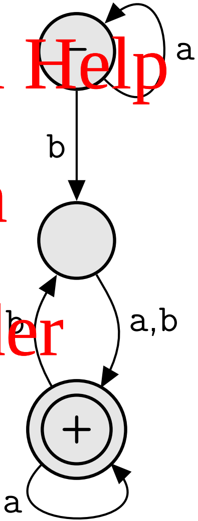


states,
represented as vertices.
Special states:

- Start State
- Final State

transitions,
represented as directed edges,
labelled by letters

. . . and Alternative notation

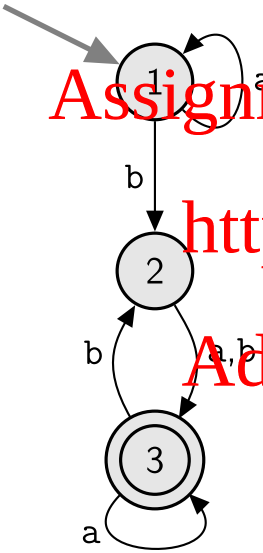


Finite automaton: definition

- A finite set of **states**
  - One called the Start State
  - Some (maybe none) called Final States

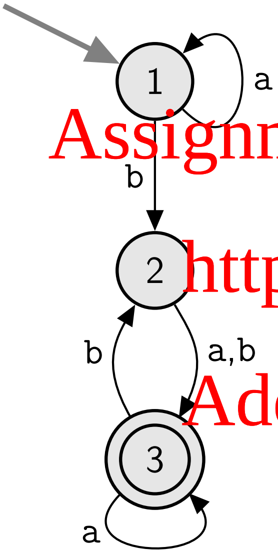
- An alphabet of possible input letters
- A finite set of **transitions**
  - that tell, for each state and each letter in the alphabet, which state to go to next.
  - There is a unique transition from any state for each letter in the alphabet.

Finite automaton: representations



		a	b
Start	1	1	2
	2	3	3
Final	3	3	2

Every string traces a unique path in the automaton, starting from the Start State and following the transitions, letter by letter.



		a	b
Start	1	1	2
	2	3	3
Final	3	3	2

Algorithm 1: Execution of a Finite Automaton

Input: a string

Begin at the Start State.

while *there is another input letter* **do**

  Read the next letter of the input string.

  Move along the directed edge labelled by this letter.

if *you're in a Final State* **then**

  Accept input string.

else

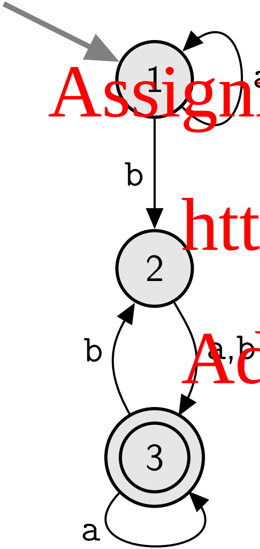
  Reject input string.

Every string traces a unique path in the automaton, starting from the Start State and following the transitions, letter by letter.

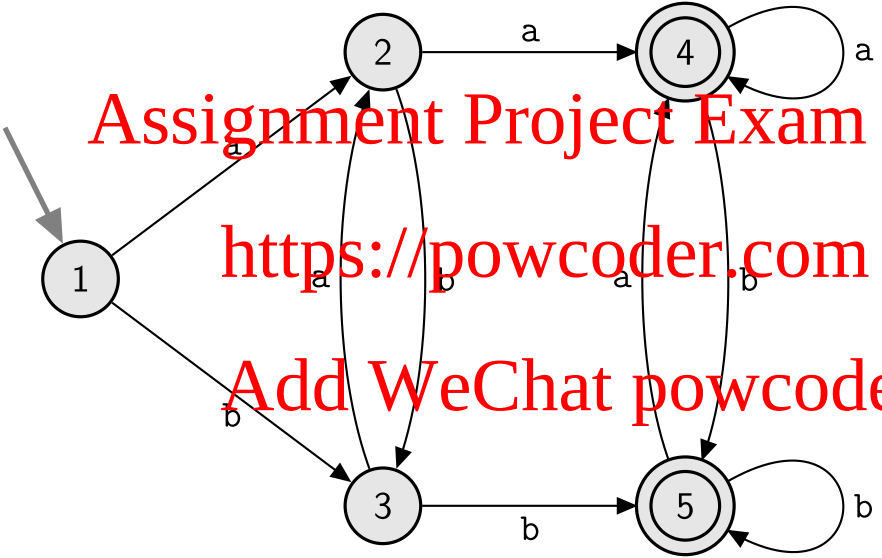
Definitions

A string is **accepted** by a FA if its path ends on a Final State.
Otherwise the string is **rejected**.
The **language recognised by** a FA is the set of all strings it accepts.
We say the FA **recognises** the language, or **accepts** the language.

Finite automata



		a	b
Start	1	1	2
	2	3	3
Final	3	3	2



		a	b
Start	1	2	3
	2	4	3
	3	2	5
Final	4	4	5
Final	5	4	5

Special Cases

- All words accepted.
- No words accepted.
- Only the empty word accepted.
- Only non-empty words accepted.
- A single word accepted.

Complements

If L is a language over an alphabet, then its **complement** $\overline{L}$ is the set of words over the alphabet that are not in L.

$$\overline{L} = \Sigma^* \setminus L$$

The complement of L is sometimes denoted by L' or L^c.

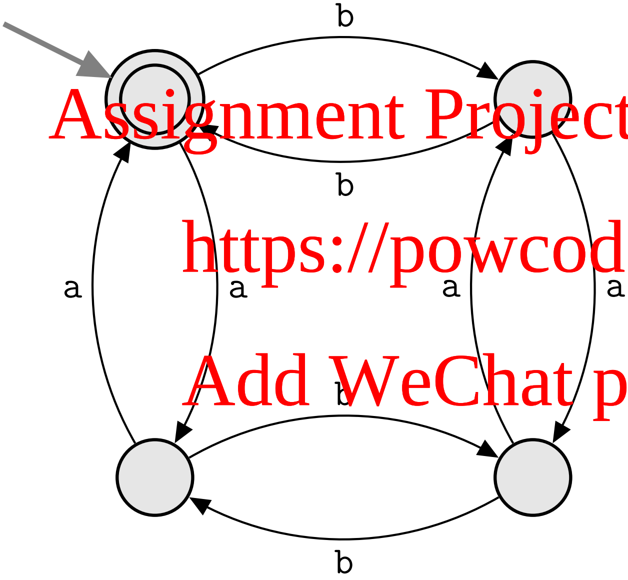
Examples

$$\overline{\emptyset} = \Sigma^* \qquad \overline{\Sigma^*} = \emptyset \qquad \overline{\{\text{words of} \leq 3 \text{ letters}\}} = \{\text{words of} \geq 4 \text{ letters}\}$$

$$\begin{aligned}\text{EVEN-EVEN} &:= \{\,\text{strings that contain an } \textbf{even} \text{ number of a's } \textbf{and} \text{ an } \textbf{even} \text{ number of b's}\,\} \\ &= \{\,\varepsilon, \mathrm{aa}, \mathrm{bb}, \mathrm{aaaa}, \mathrm{aabb}, \mathrm{abab}, \mathrm{abba}, \mathrm{baab}, \ldots\,\}.\end{aligned}$$

$$\begin{aligned}\overline{\text{EVEN-EVEN}} &:= \{\,\text{strings which contain an } \textbf{odd} \text{ number of a's } \textbf{or} \text{ an } \textbf{odd} \text{ number of b's}\,\} \\ &= \{\,\mathrm{a}, \mathrm{b}, \mathrm{ab}, \mathrm{ba}, \mathrm{aaa}, \mathrm{aab}, \mathrm{aba}, \mathrm{abb}, \mathrm{baa}, \ldots\,\}\end{aligned}$$

EVEN-EVEN



Complement Finite Automaton (FA)

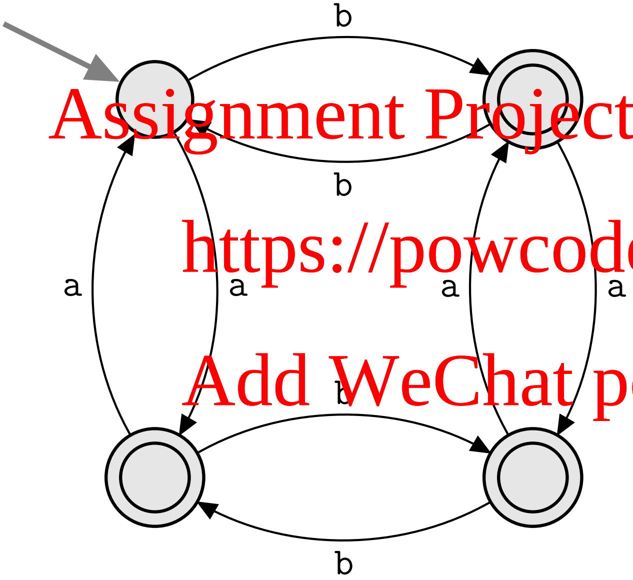
Suppose some FA accepts the language L.

Change all the final states in this FA to non-final states,
and all the non-final states to final states.

This new FA now accepts all the strings not accepted by the original FA
(i.e., all the words in $\overline{L}$),
and rejects all the words that the original accepted
(i.e., the words in L).

So the new FA accepts $\overline{L}$.

EVEN-EVEN



Comparison with Regular Expressions

It is (usually) easier

to write down a **regular expression** that defines a language

than

to design a **FA** to accept this language.

It is easier

to check whether a given string is accepted by a **FA**

than

to see whether it matches a **regular expression**.

It is easier

to find complements using a **FA**

than

by using a **regular expression**.

Some Generalizations of Finite Automata

This week:

- It is *not* required that, for every state and letter, there is a *unique* transition.
- It can change state *without* reading a letter.
- It can read *more than one letter* at a time.

Next week:

- It can read strings which match *regular expressions*, not just single letters.

Later:

- Each transition can produce *output letters* as well as changing state. *(transducer)*
- Transitions can *read and write* letters from some kind of *memory*.
- For a given state and letter, the next state is chosen *probabilistically*.

Nondeterministic Finite Automaton (NFA)

NFA are defined like a Finite Automaton (FA) except for transitions.

Transitions

- For some states and letters there is a transition.
- The labels may include the empty word ε.

So for a given letter and state there may be:

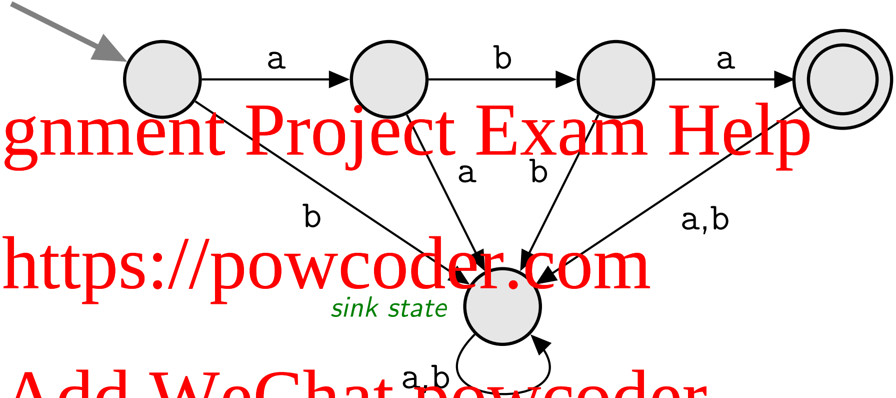
- No transition
- More than one transition

For a given string, the path it takes . . .

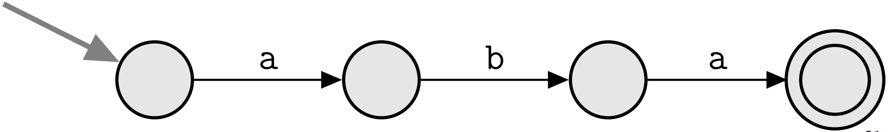
- might not exist
- might not be unique

aba

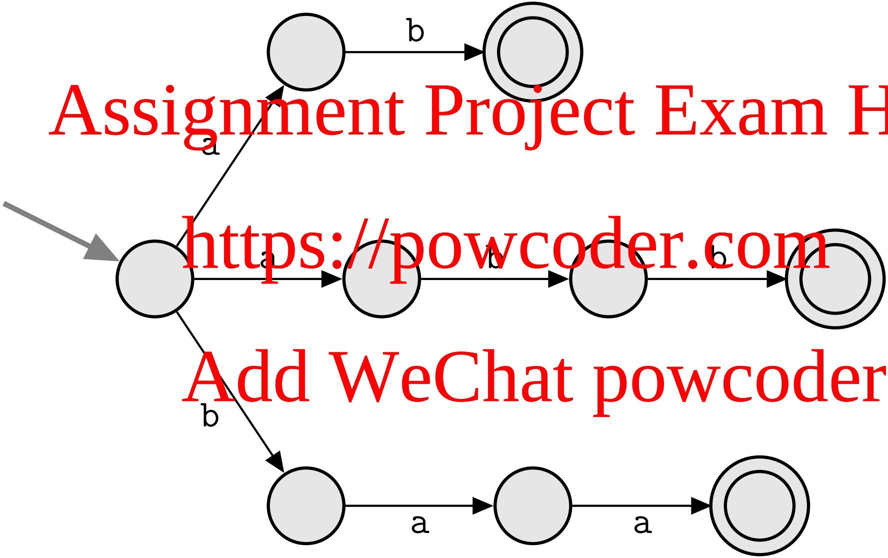
FA



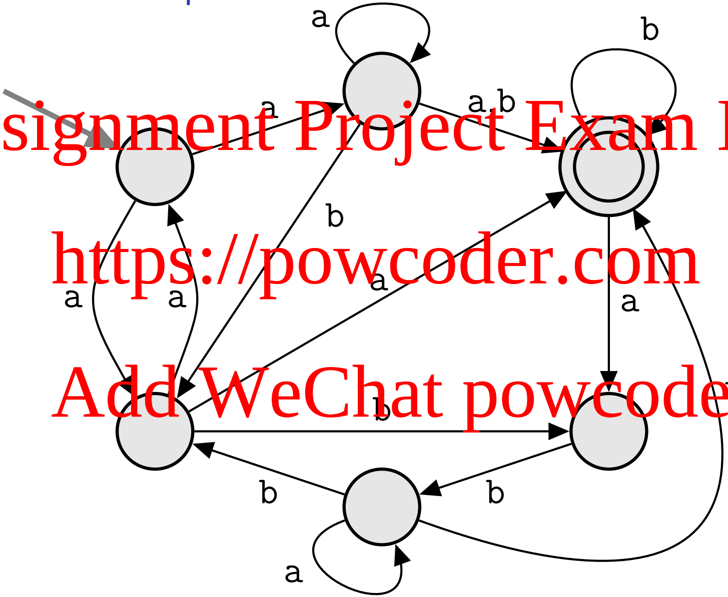
NFA



ab ∪ abb ∪ baa



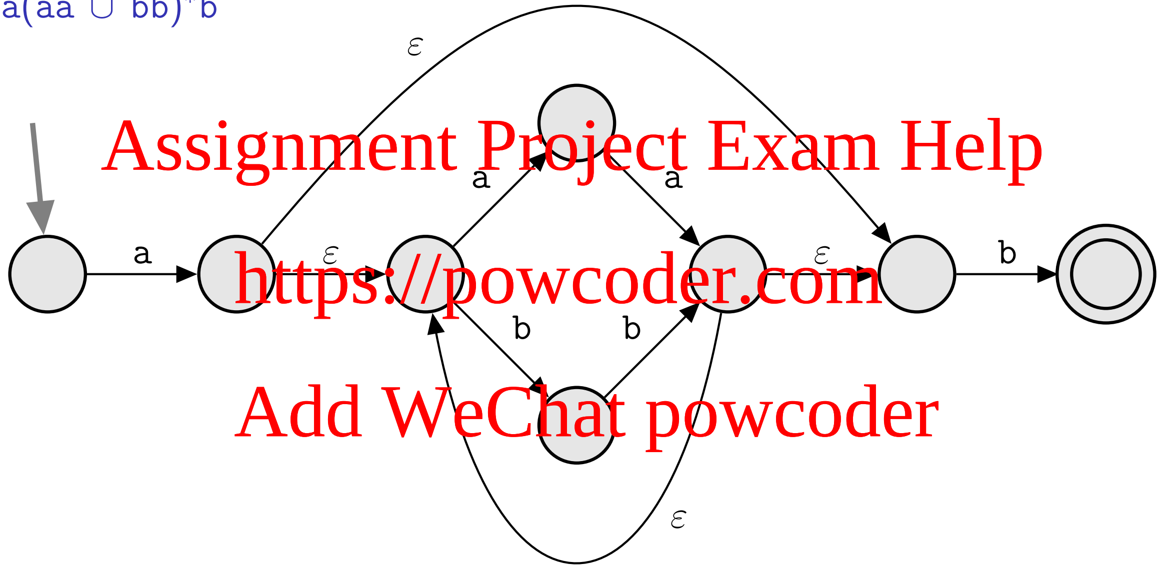
Is abbbabbbabba accepted?



Properties

- If there is no transition for the current letter and state the machine crashes.
- Paths from the Start State to a Final State for a given input:
  - One
  - None
  - Several (Nondeterministic)
- Accept a string if there is **at least one** path from the Start State to a Final State.
- Reject a string if there are **no** paths from the Start State to a Final State.

$a(aa \cup bb)^*b$



Revision

Finite Automata (FA)

- Definition. How to use them.
- How to construct a Finite Automaton to accept a language.

Complement Languages

- What they are. Designing FA to accept them.

Nondeterministic Finite Automata (NFA)

- Definition. How to use them
- How to construct a Nondeterministic Finite Automaton to accept a language.

Reading: Sipser Ch 1.