

# Lecture 6

# Finite Automata

Slides by David Albrecht (2011), modified by Graham Farr (2013).

FIT2014 Theory of Computation

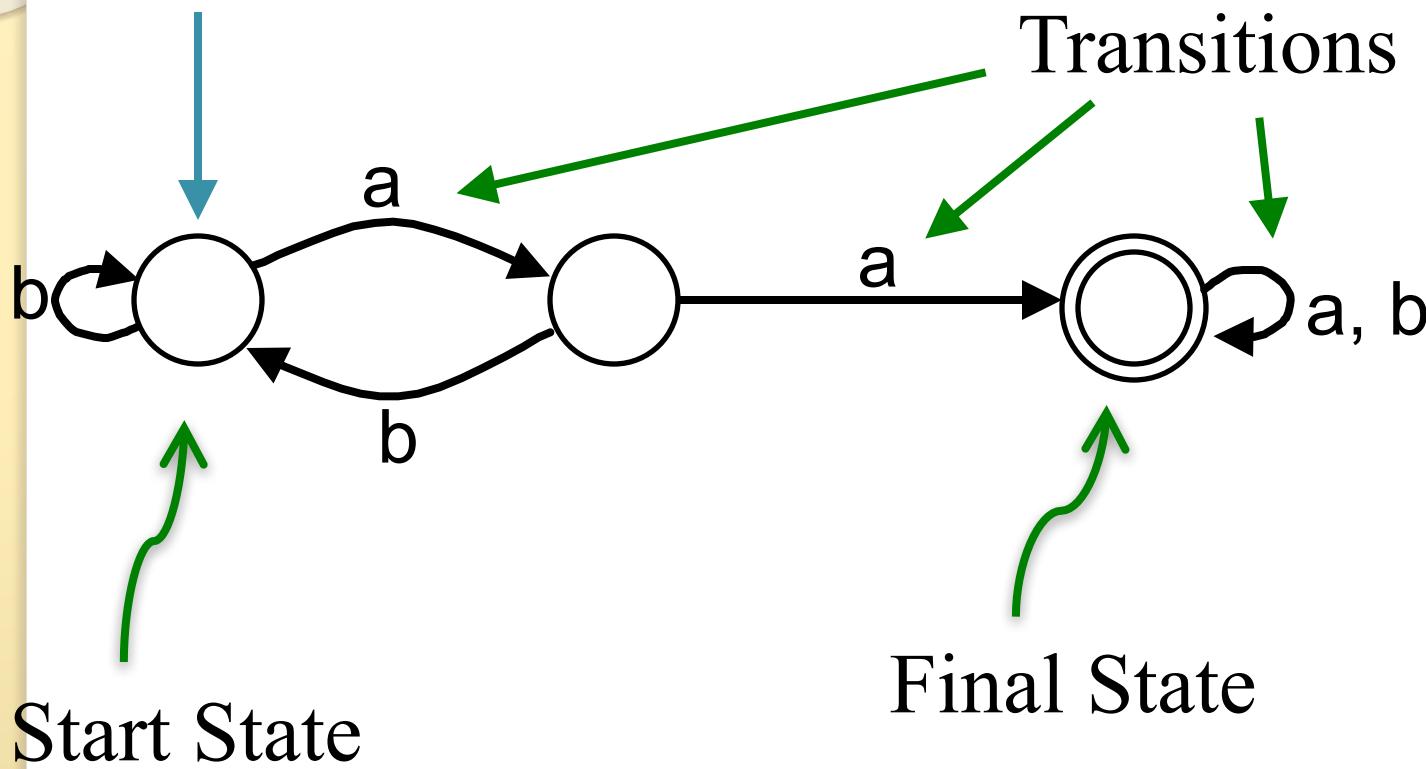
# Overview

- Definition
- How they are used to define languages
- Representations
- Complement Languages
- Comparison with Regular Expressions

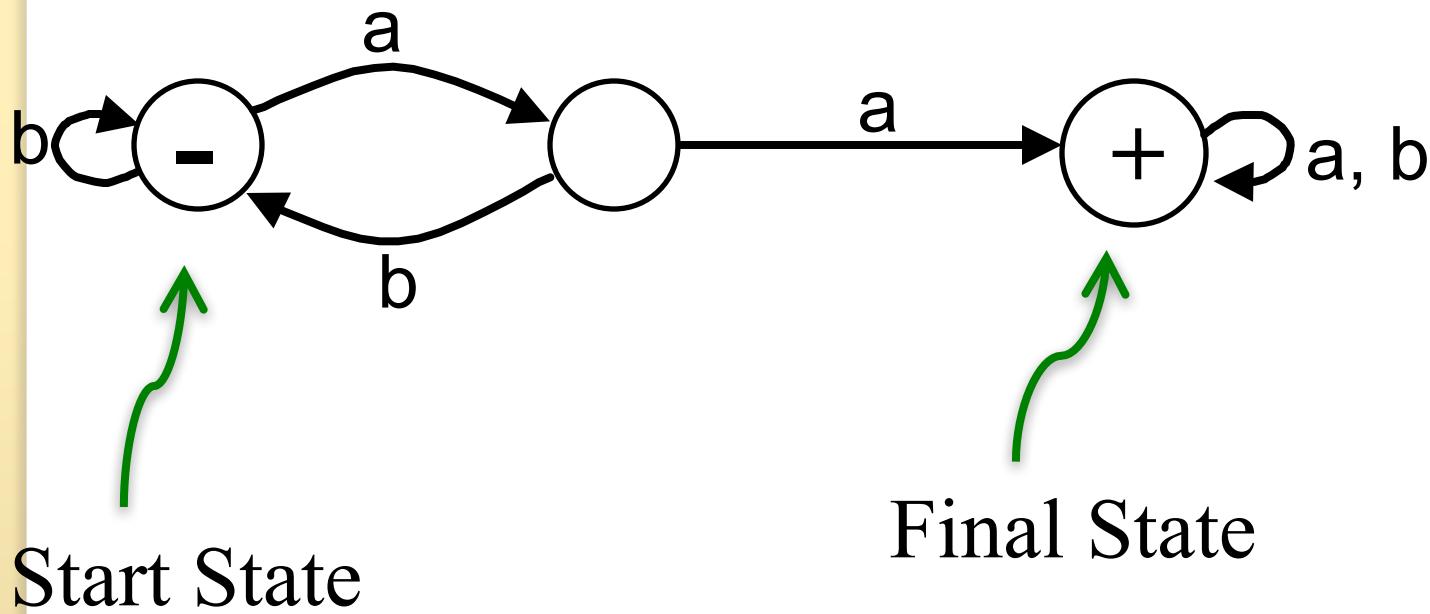
# Finite Automaton

- Sometimes known as a **Deterministic Finite Automaton**.
- 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



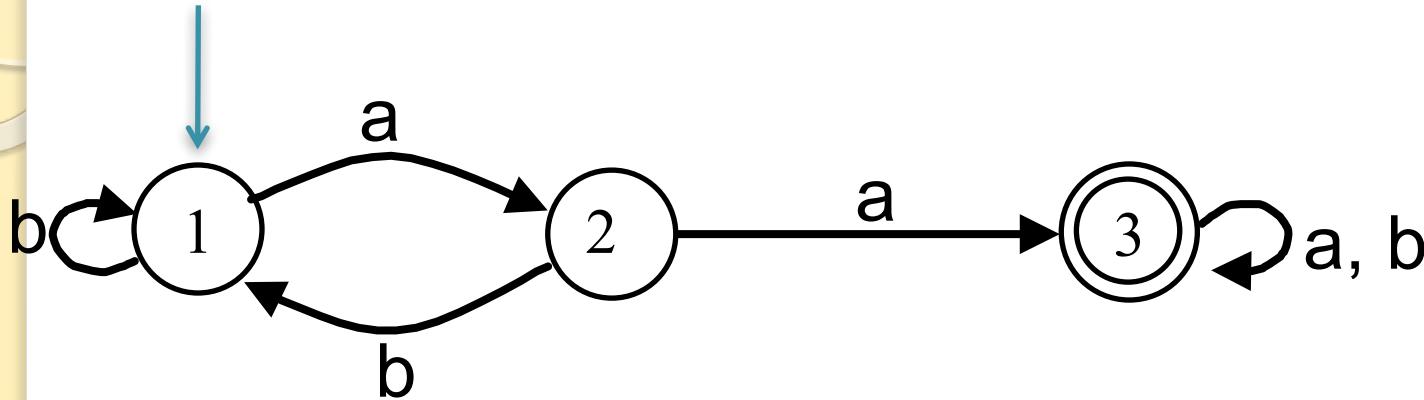
# Alternative Notation



# 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 an unique transition from any state for each letter in the alphabet.

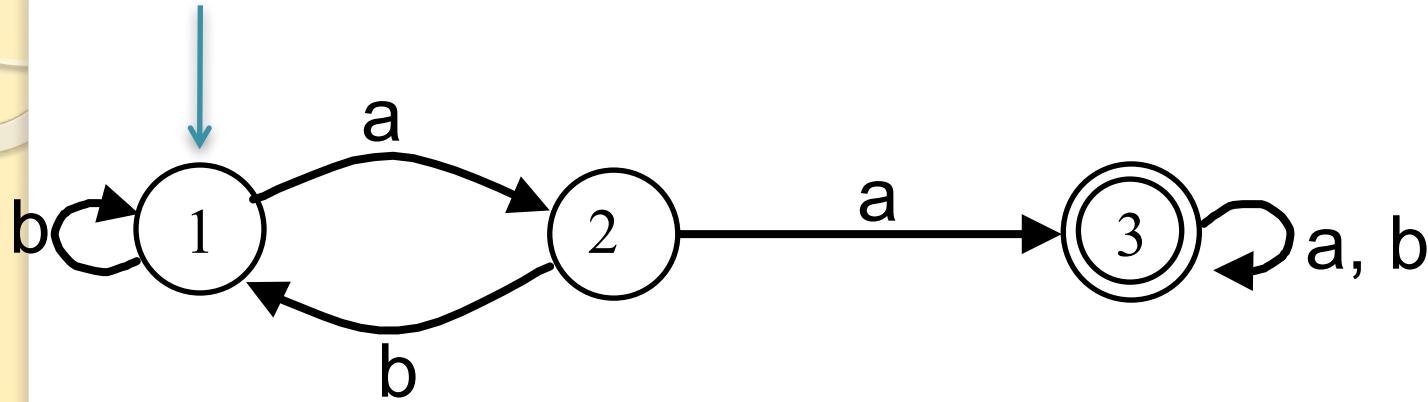
# Finite automata: representations



	a	b
Start	2	1
	2	3
Final	3	3

# Finite automata

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	2 1
	2	3 1
Final	3	3

Begin in the Start State

Is there INPUT?

Yes

Read next letter  
in the INPUT

No

Move along the edge with this label

Are you in a  
Final State?

Yes

Accept String

No

Reject String

# Finite automata

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

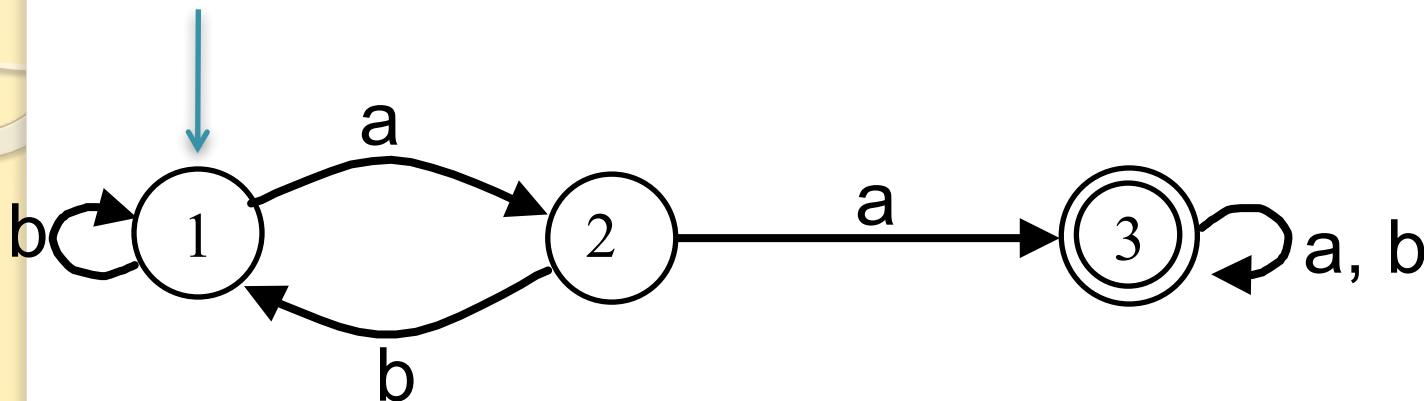
## Definitions

A string is **accepted** by a FA if its path ends on a Final State. Otherwise the string is **rejected**.

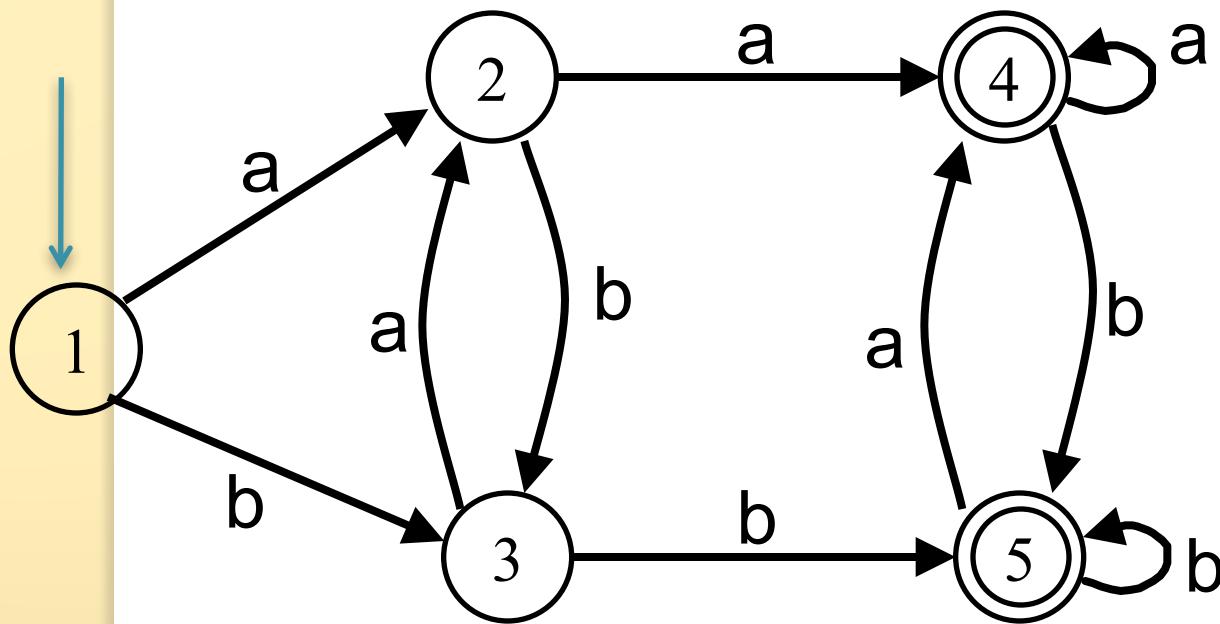
The **language recognised** by a FA is the set of all strings it accepts.

We say the FA **recognises** the language, or **accepts** the language.

# Representations



	a	b
Start	1	2
	2	3
Final	3	3



	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.
- Only the empty word accepted.
- A single word accepted.

# Complements

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

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

# EVEN-EVEN

**EVEN-EVEN** is the set of strings that contain an **even** number of **a**'s and an **even** number of **b**'s.

- E.g.

$\epsilon$  aa bb aaaa aabb abab abba baab

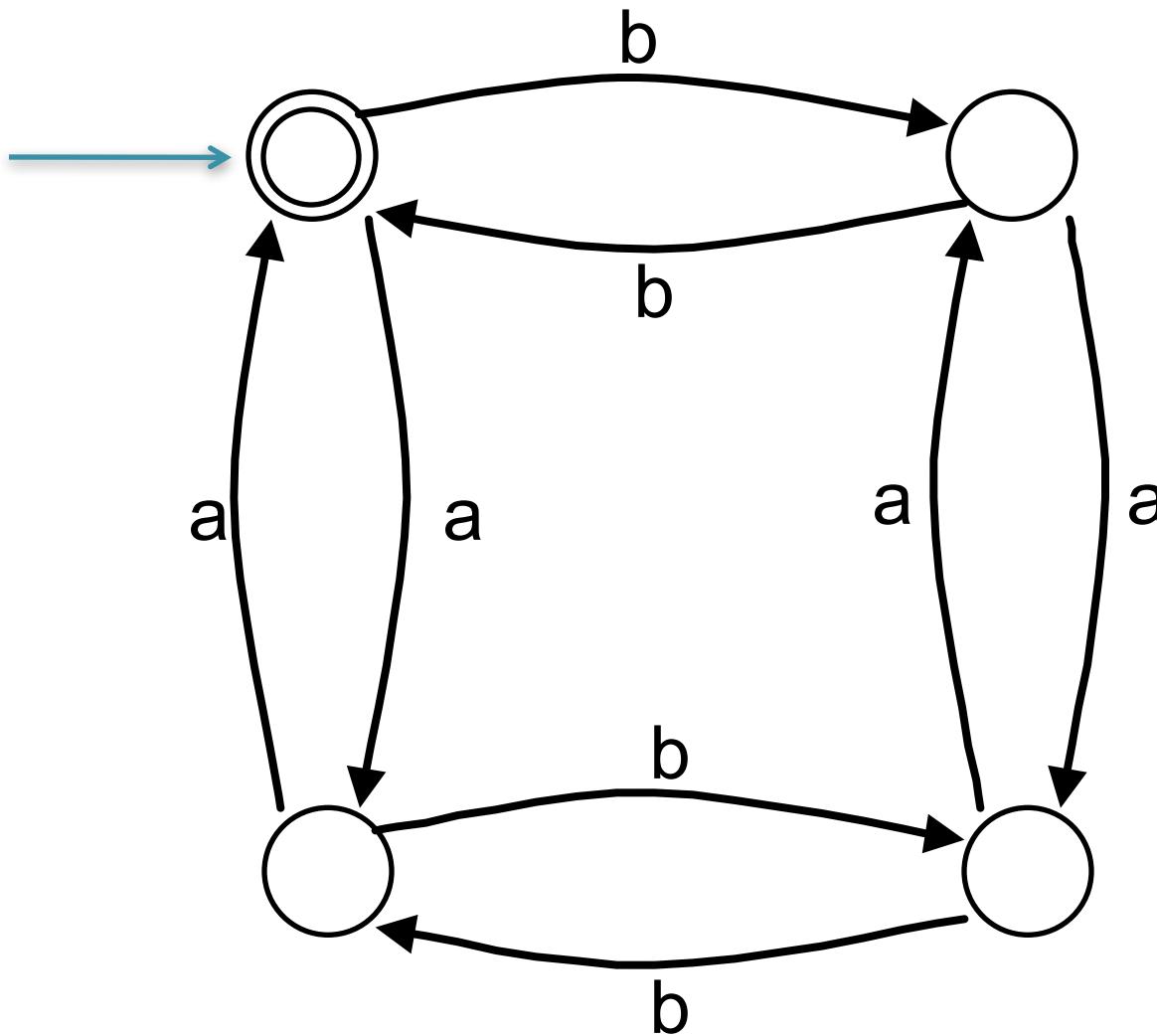
---

**EVEN-EVEN** is the set of strings which contain an **odd** number of **a**'s or an **odd** number of **b**'s.

- E.g.

a b ab ba aaa aab aba abb baa ...

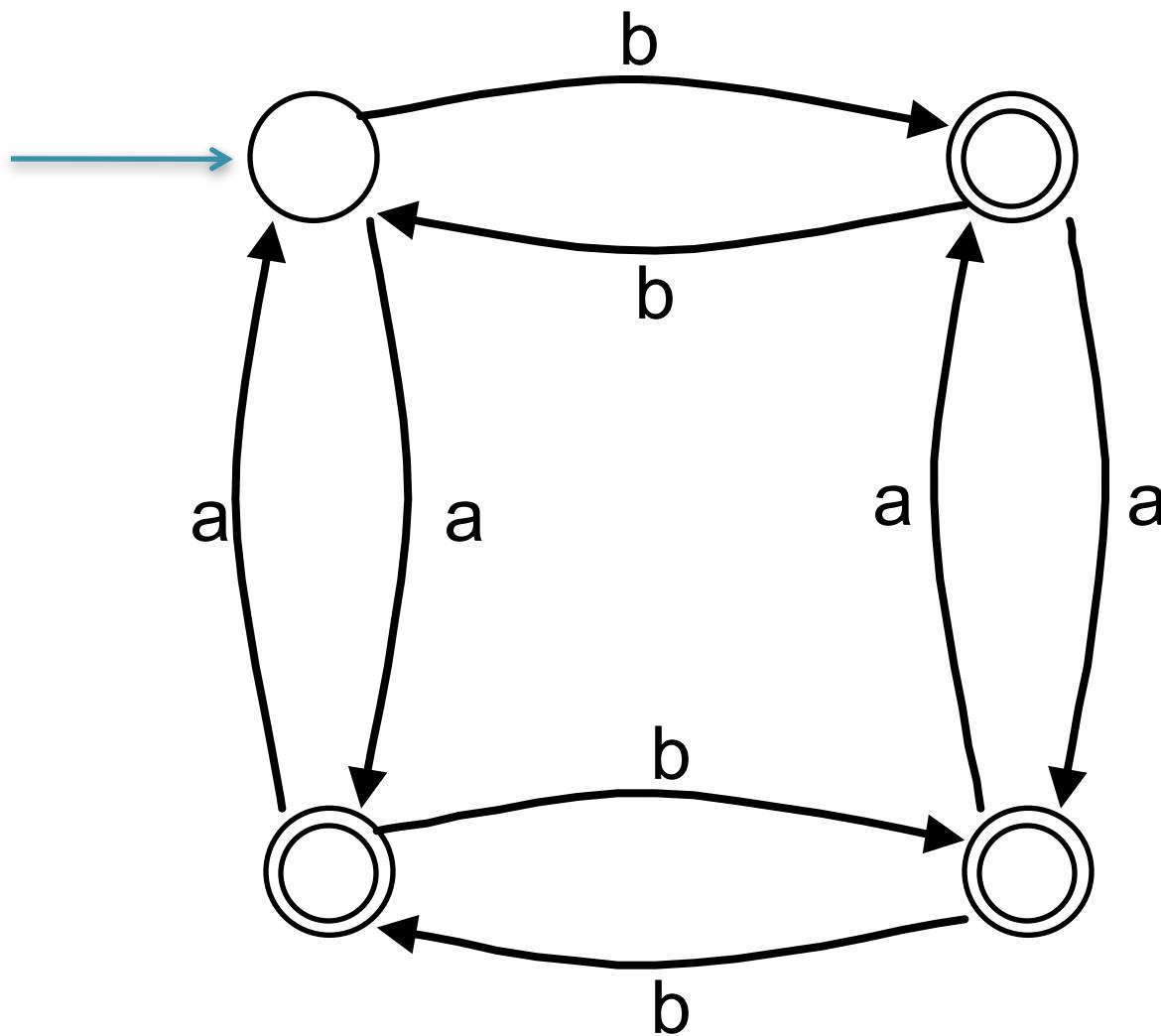
# EVEN-EVEN



# 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  $\bar{L}$ ), and rejects all the words that the original accepted (i.e., the words in  $L$ ).
- So the new FA accepts  $\bar{L}$  .

# EVEN-EVEN



# Comparison with Regular Expressions

- It is 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 it is 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

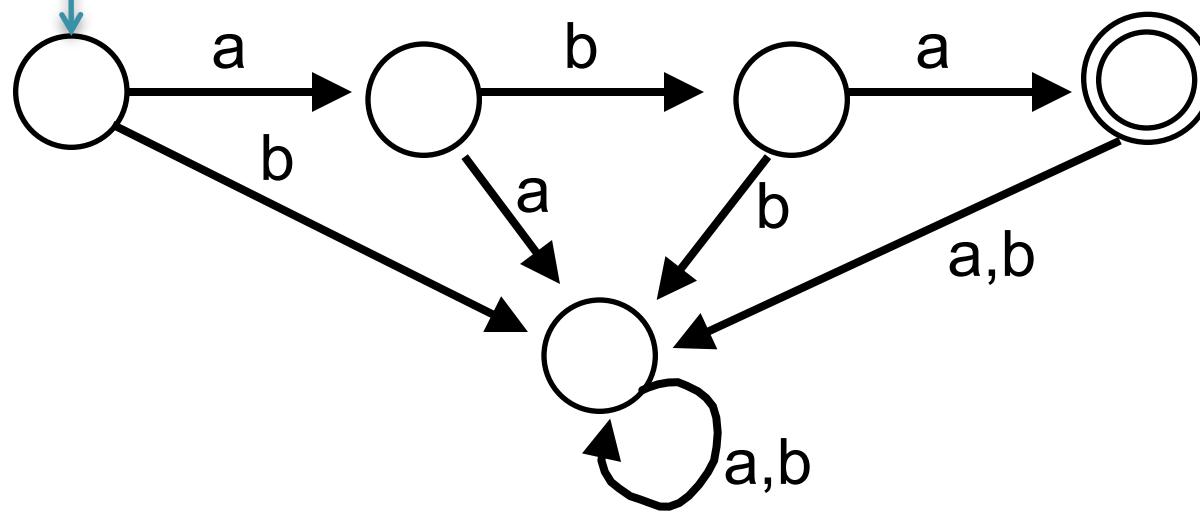
- For every state and letter, there is **not a unique** transition.
- **Change state without** reading any **letter**.
- Read **more than one** letter at a time.
- Read **strings** which **match regular expressions**.

# Nondeterministic Finite Automaton (NFA) Definition

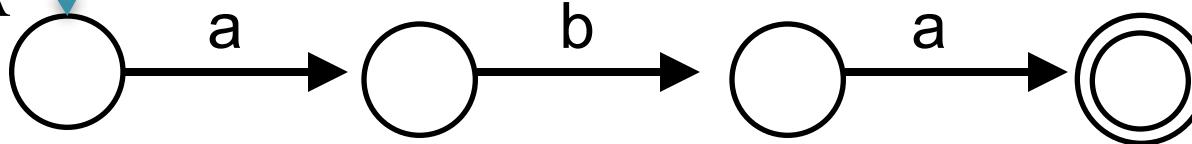
- Like a Finite Automaton (FA) except for transitions
- Transitions
  - For **some** states and letters there is a transition.
  - **The labels may include the empty word  $\epsilon$ .**
- So for a given letter and state there may be:
  - **No transition**
  - **More than one transition**
- For a given string, the path it takes ...
  - might **not exist**
  - might **not be unique**

**aba**

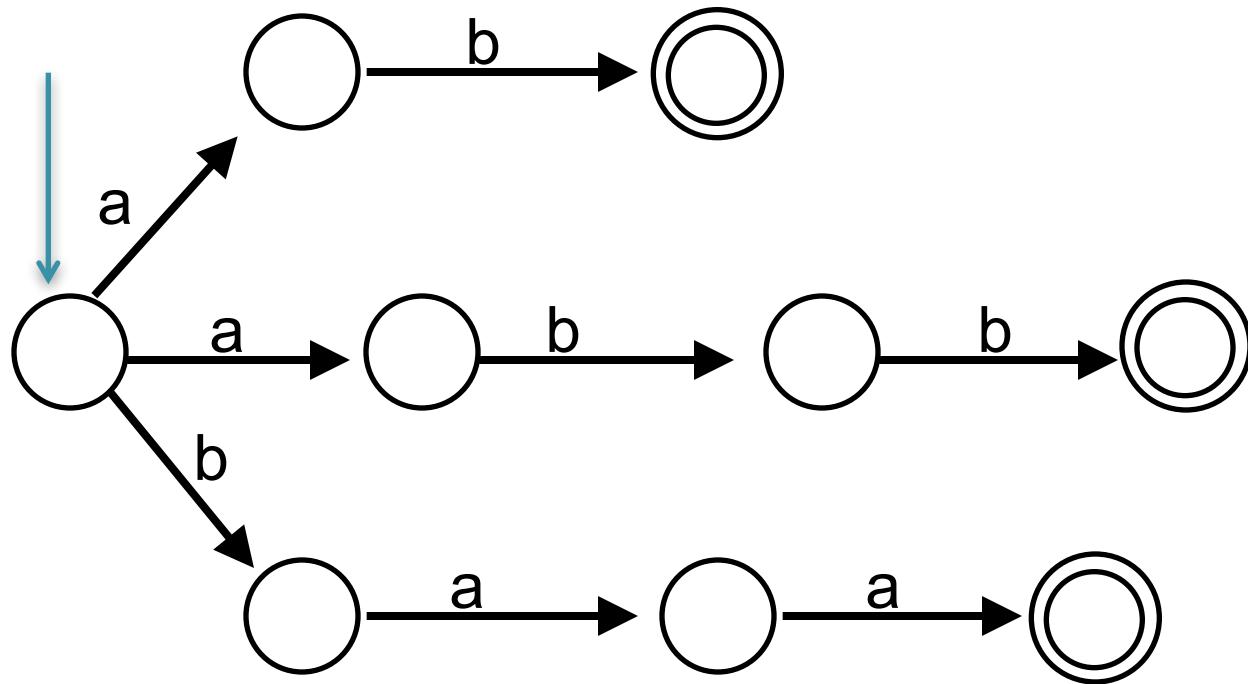
FA



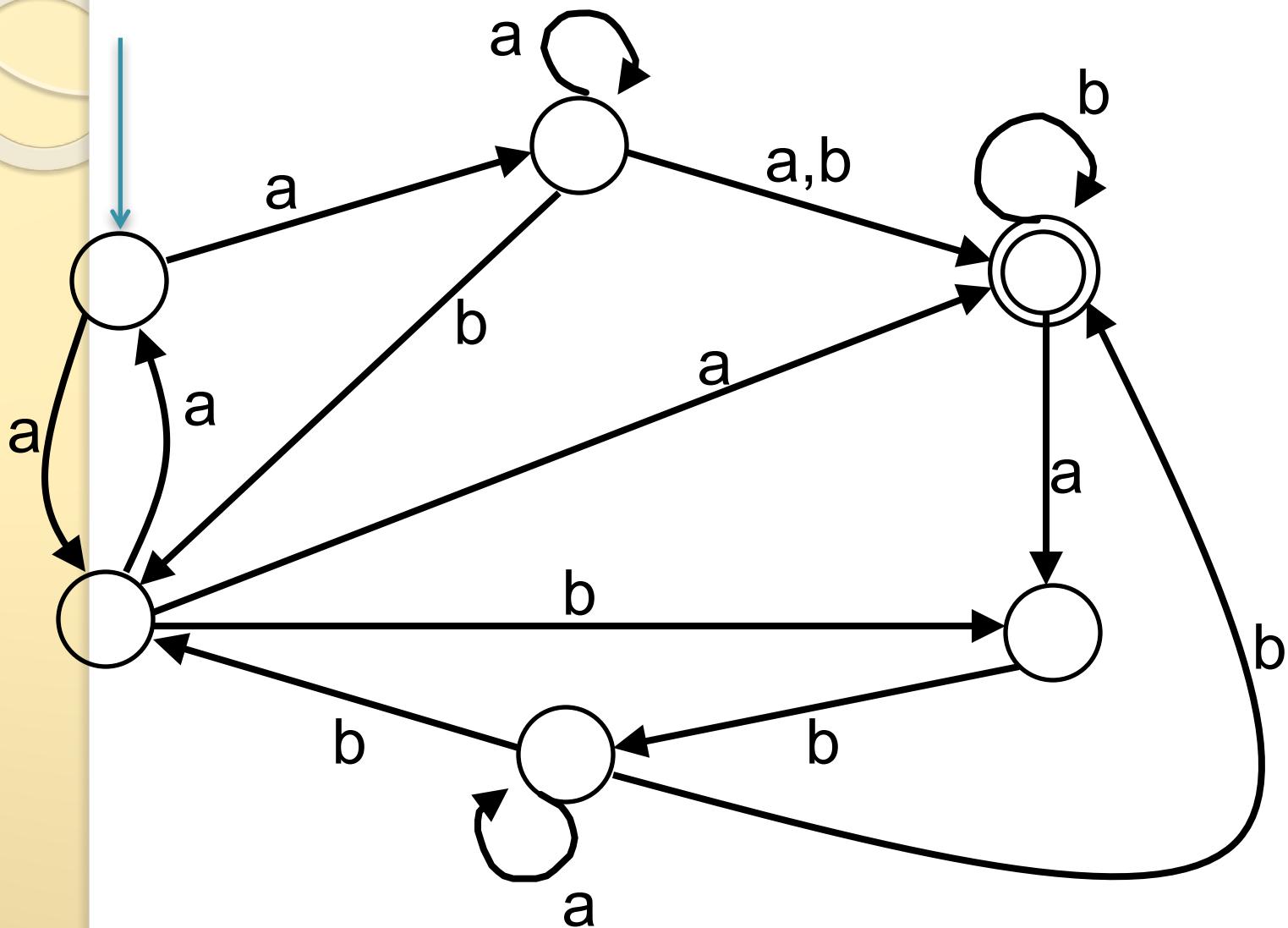
NFA



$ab \cup abb \cup baa$



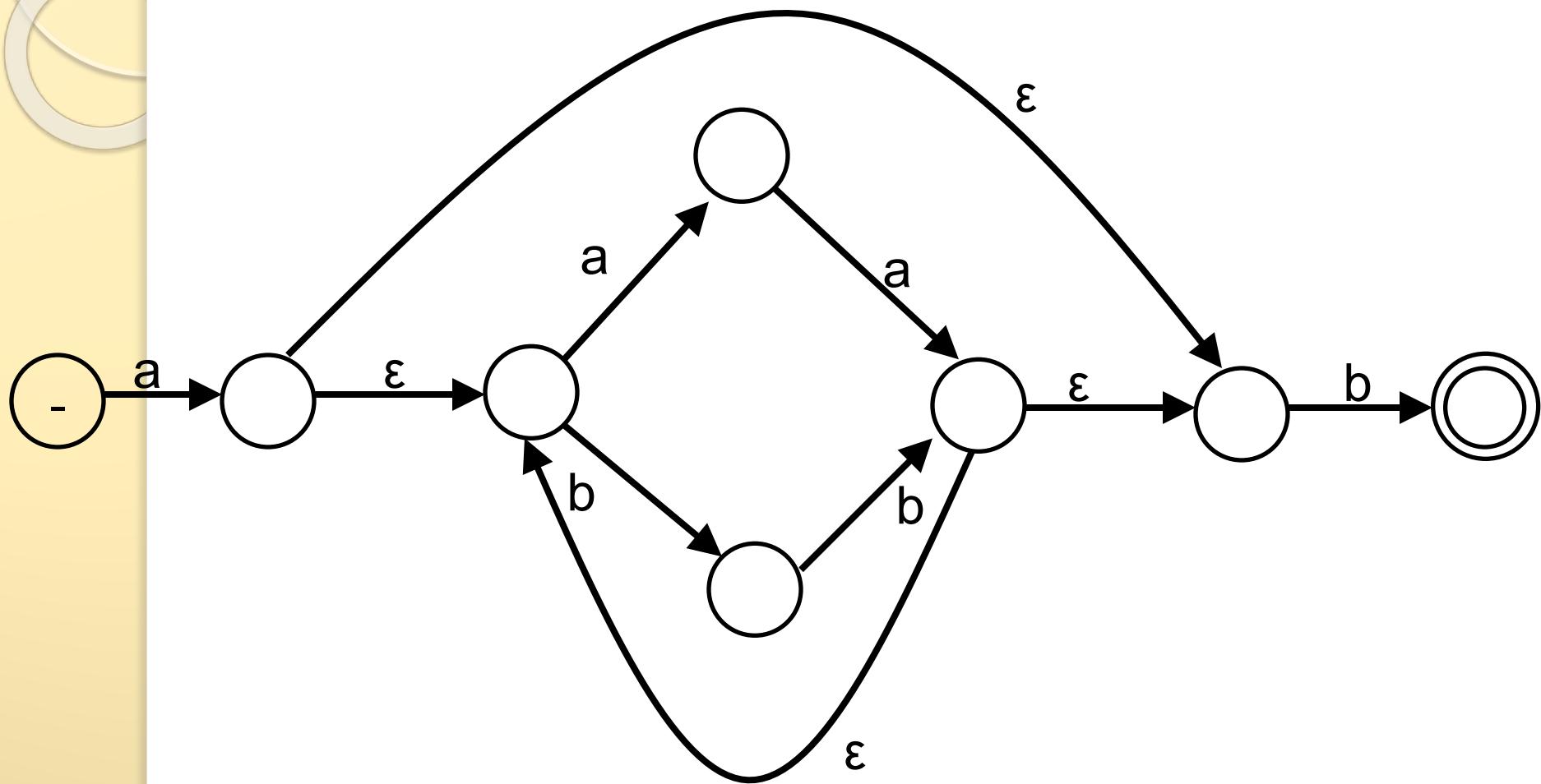
# Is abbbabbbbabba accepted?



# 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 Automata to accept a language.
- Reading: Sipser Ch I.