Compiler Construction (CS-636)

Sadaf Manzoor
UIIT, Rawalpindi

Outline

1. Deterministic Finite Automata
2. Non-deterministic Finite Automata
3. Summary

Revision of Automata Theory Concepts

Lecture: 5 & 6

Formal Language Definitions

- Why need formal definitions of language
  - Define a precise, unambiguous and uniform interpretation
  - Communication with machines
- Formal Language notation/definition
  - Regular Expression
    - Tell how to generate words of that language
    - Tell which words belong to this language

Finite Automata

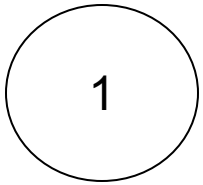
- Language Recognizers
- Machines embedded with grammatical rules that recognize a language
- Automated language recognition
- REs define a language and FAs accept (or reject) them
- FAs serve two purposes
  - Implicit language definition
  - Explicit Recognition

Finite Automaton

- A finite automaton is a collection of three things
  - A finite set of states
    - Exactly one initial state (start state)
    - One or more (may be none) final states that mark the acceptance of a word
    - Intermediate states that are neither start not final states
  - An alphabet $\sum$ of possible input letters
  - A finite set of transitions that tell for each state and for each letter of the input alphabet which state to go to next

Finite Automaton

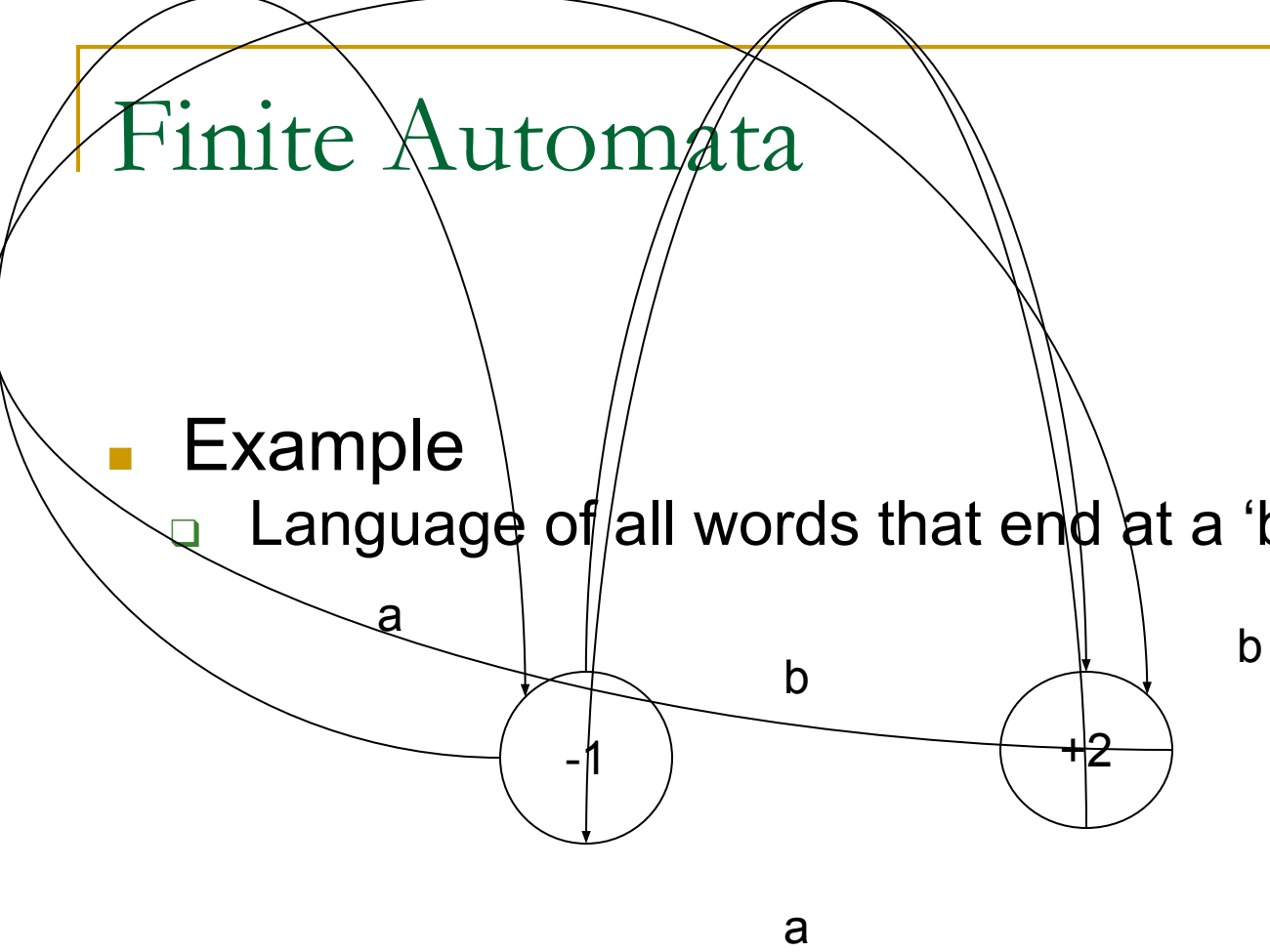
- Visual representations
  - States represented by circles labeled to identify each distinctly
    - Initial and Final states



  - Transitions
    - Directed edges labeled with the characters of ∑

Finite Automata

- Example
  - Language of all words that end at a ‘b’:



Characteristics of DFA

- Every FA must have exactly one start state
- There may be multiple or may be no final states
  - In the latter case the FA does not accept any language
- Only a single character is read on a state at a time

Characteristics of DFA

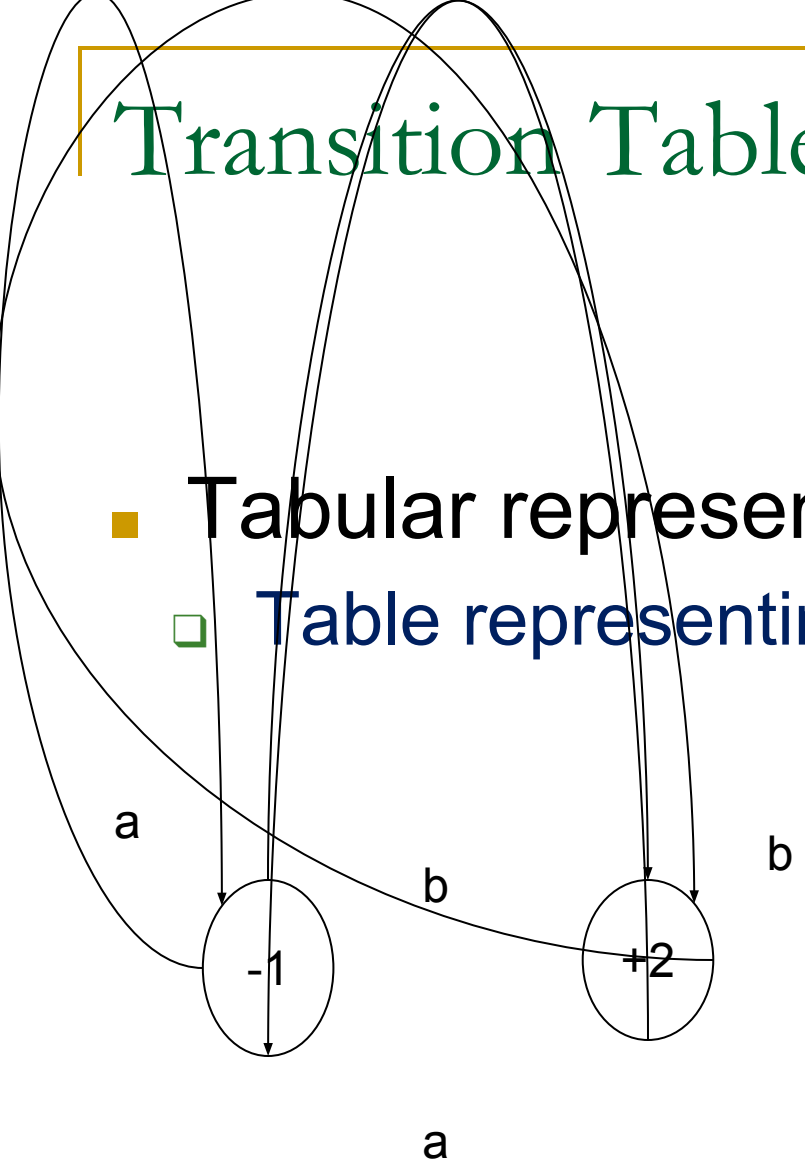
- Every state define a transition for every character in the alphabet set or alternatively every state has exactly as many outgoing transitions as the number of characters in ∑, each labeled with a distinct letter from ∑
  - No duplicate edges
  - No missing edges

Mathematical Representations of FA

- FA = $(Q, \sum, q_0, F, \delta)$
  - $Q = \{q_0, q_1, q_2, ..q_n$ where n is finite$\}$
  - $\sum$ = set of input alphabets
  - q_0 is the start states
  - $F \subseteq Q$ is the set of final states F may be ϕ
  - δ is the transition function
    - $\delta (q_i, x_j) = q_k$
- Mathematical representation of FA for all words ending at b.

Transition Tables

- Tabular representation of an FA
  - Table representing states and transitions



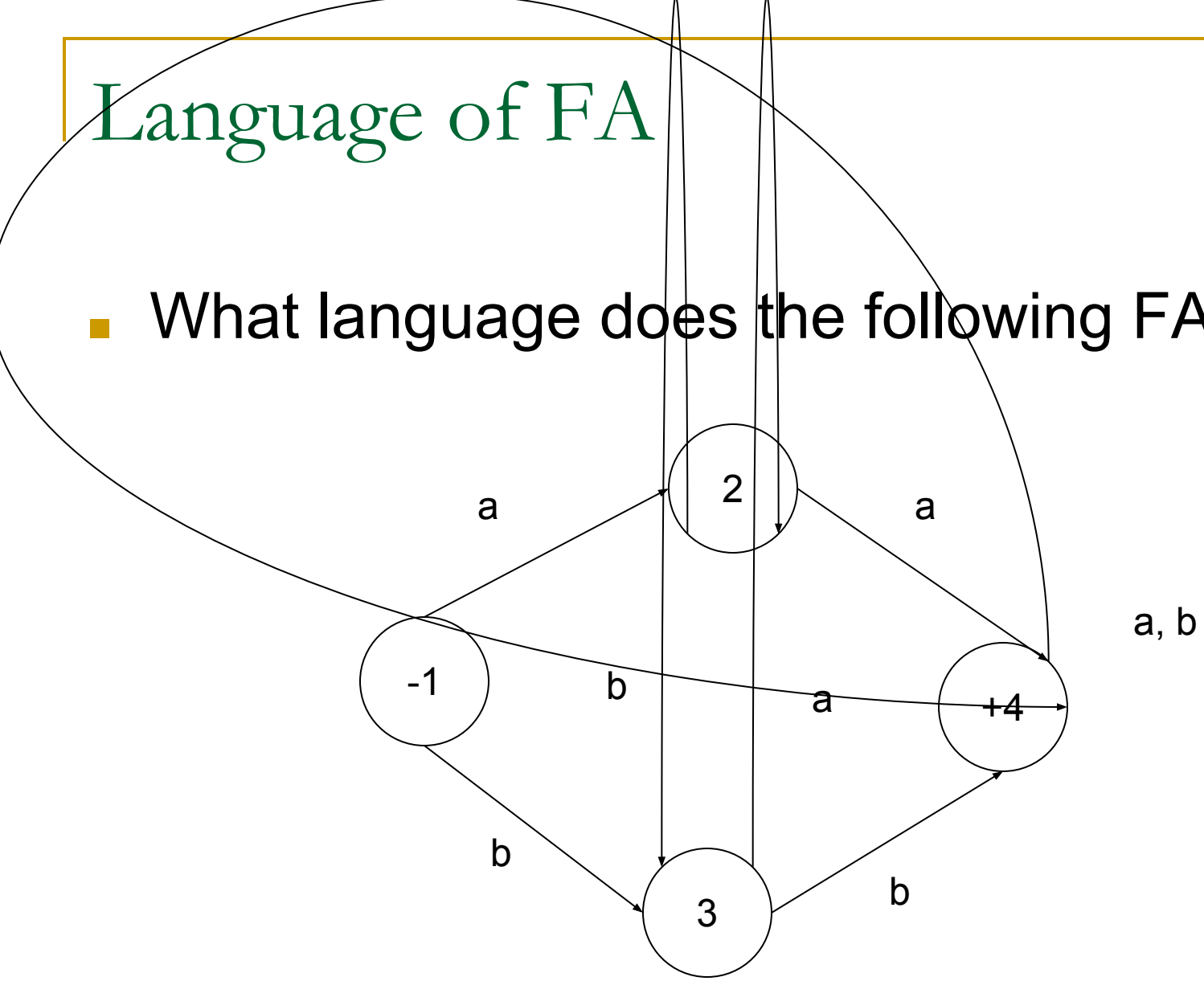
	a	b
-1	1	2
+2	1	2

Languages of FA

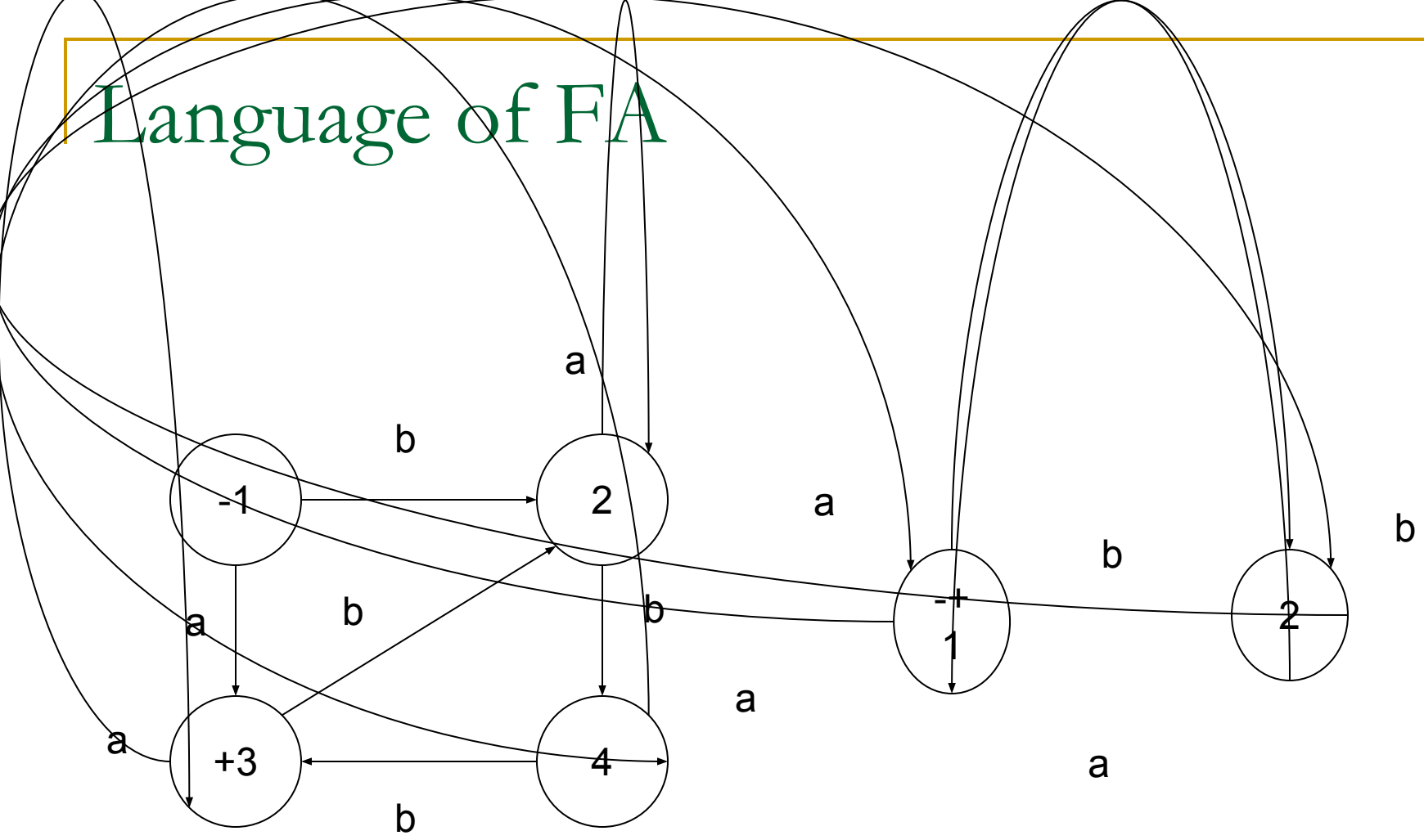
- FAs define Regular Language
- Any language that can be define by a regular expression can be recognized by an FA

Language of FA

- What language does the following FA define



Language of FA



Nondeterministic Finite Automata (NFA)

- The FA where a state can have more than one transition for the same character. This puts the machine in an indecisive state for which transition to follow
  - Has duplicate transitions
  - Can miss transitions for some characters
- Thus called Non-deterministic Finite Automaton

NFA

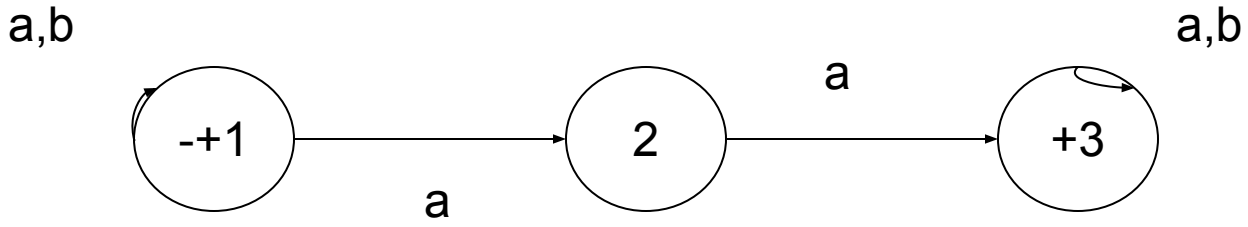
- Reduces number of states and transitions
- Costly execution
  - Needs concurrent processing to find a successful path
- An NFA can have a successful and an unsuccessful path for the same input
- If an NFA has at least one successful path for an input, it is considered to be valid.
- Machine crashes for an undefined transition thus causing implicit reject

NFA Language recognition

- Acceptance
  - If at least one successful path exists
- Rejection
  - Either machine crashes on input OR
  - No successful path exists

NFA Example

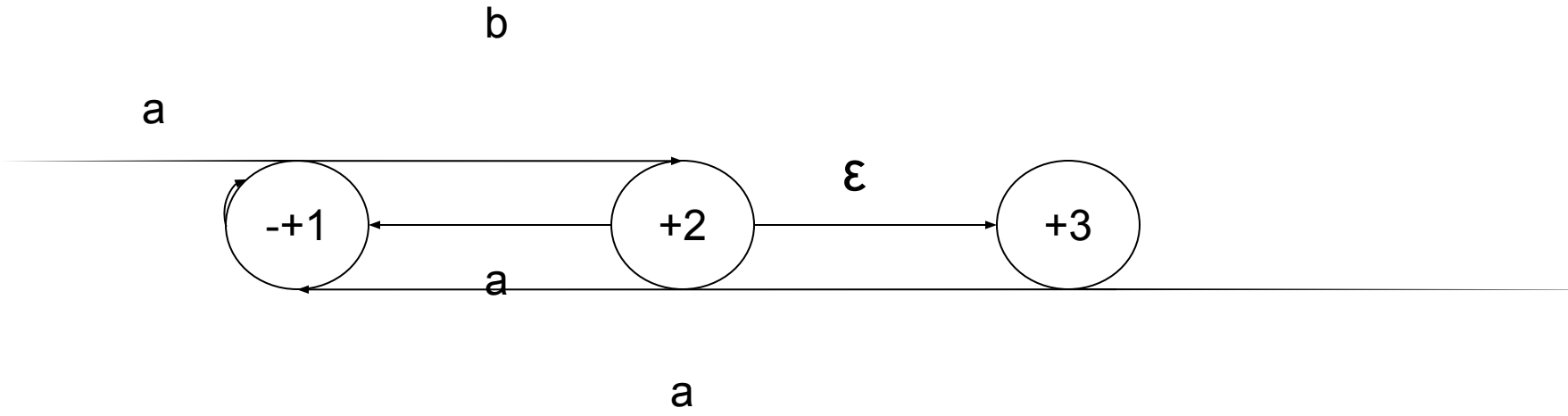
- All strings that contain at least one existence of aa



Epsilon Transitions

- ε- Transitions
- A null transition that changes state but doesn't consume any character
- Possible with NFAs and Transition Graphs

Epsilon Transitions Example



Why NFA?

- NFAs are sometimes easier to construct than DFAs and are less complicated than DFAs
- It is easier to prove properties of computational theory on NFAs rather than DFAs

Summary

Any Questions?