Computational Linguistics
Lecture 2

Dr. Dina Khattab
dina.khattab@cis.asu.edu.eg

Models for getting Knowledge of Language

- We tackle the problem of defining languages by considering how we could recognise them.

Models for getting Knowledge of Language

1. State Machines
   - Finite State Automata (FSA) and Regular Expressions (Regex)
2. Formal Rule System
   - Context Free Grammar (CFG)
   - Backus-Naur Form (BNF)
3. Logic
   - Predicate Calculus
   - Semantic Networks
4. Probability Theory

Representations for languages

- We will discuss the two principal methods for defining languages: the generator and the recognizer

- In particular we will focus on a particular class of generators (grammars) and of recognizers (automata)

- Regular languages are the simplest formal languages:
  - Their generators are the regular expressions
  - Their recognizers are the finite state automata

Concepts and Notations

- **Set**: An unordered collection of unique elements

  $S_1 = \{ a, b, c \}$ $S_2 = \{ 0, 1, \ldots, 19 \}$ *empty set:* $\emptyset$

  membership: $x \in S$ *union:* $S_1 \cup S_2 = \{ a, b, c, 0, 1, \ldots, 19 \}$

  universe of discourse: U *subset:* $S_1 \subset U$

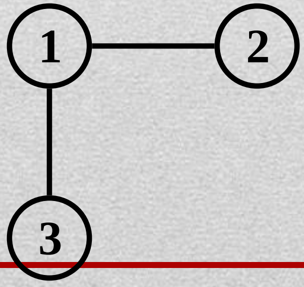
  complement: if $U = \{ a, b, \ldots, z \}$, then $S_1' = \{ d, e, \ldots, z \} = U - S_1$

- **Alphabet**: A finite set of symbols
  - Examples:
    - $\Sigma_1 = \{ a, b \}$ $\Sigma_2 = \{$ Spring, Summer, Autumn, Winter $\}$

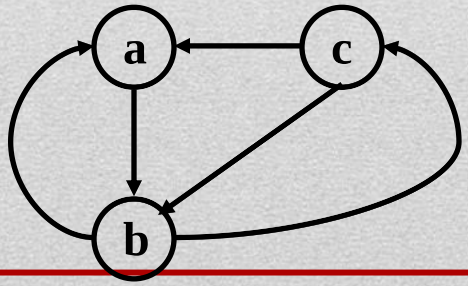
- **String/word**: A sequence of zero or more symbols from an alphabet
  - The empty string: ε

Concepts and Notations

- **Language**: A set of strings over an alphabet
  - Also known as a formal language; may not bear any resemblance to a natural language, but could model a subset of one.
  - The language comprising **all** strings over an alphabet Σ is written as: Σ^*
- **Graph**: A set of nodes (or vertices), some or all of which may be connected by edges.
  - An example:



  - A directed graph example:



Finite State Automata (FSA)

Finite State Automata

- Language Recognition Problem:

Whether a word belonging to language?

i.e. given a language description and a string, is there an algorithm which will answer yes or no correctly?

Finite State Automata

- A finite state automaton is an abstract model of a simple machine (or computer) i.e. a computational device to solve the language recognition problem

- The machine can be in a finite number of states. It receives symbols as input, and the result of receiving a particular input in a particular state moves the machine to a specified new state.

- Certain states are finishing states, and if the machine is in one of those states when the input ends, it has ended successfully (or has accepted the input).

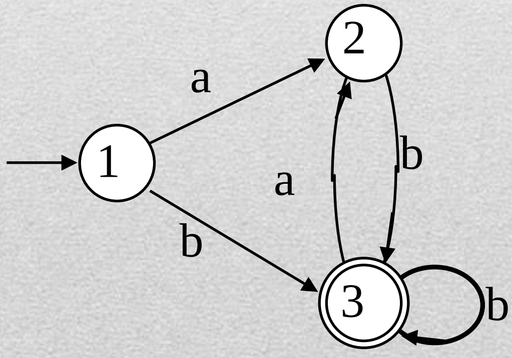
FSA: Formal Definition

A Finite State Automaton (FSA) is a 5-tuple (Q, I, F, T, E) where:

Q	= states	a finite set;
I	= initial states	a nonempty subset of Q;
F	= final states	a subset of Q;
T	= an alphabet;	
E	= edges	a subset of $Q \times T \rightarrow Q$.

FSA can be represented by a labelled, directed graph
= set of nodes (some final/initial) +
directed arcs (arrows) between nodes +
each arc has a label from the alphabet.

Example: formal definition of A_1



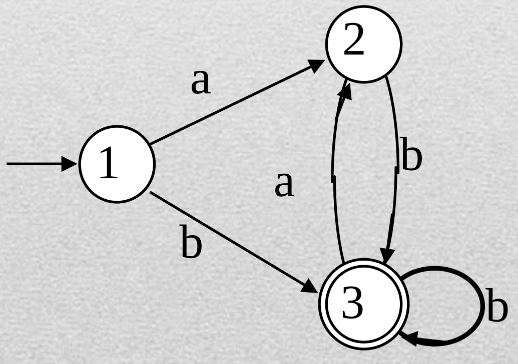
Q = {1, 2, 3}
I = {1}
F = {3}
T = {a, b}
E = { (1,a,2), (1,b,3), (2,b,3), (3,a,2), (3,b,3) }

What does it mean to accept a string/language?

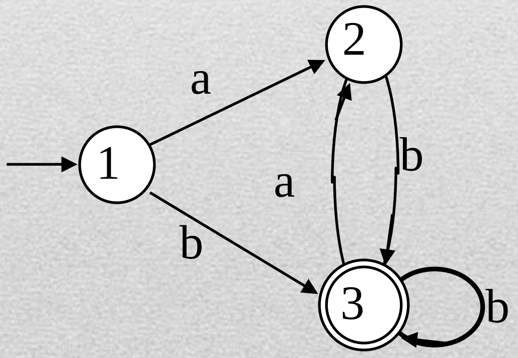
If the FSA is in a final (or accepting) state after all input symbols have been consumed, then the string is accepted (or recognized), otherwise it is rejected

e.x. String: abb

Give other Examples!

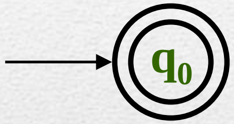


The language accepted by A_1 is the set of strings of *a*'s and *b*'s which end in *b*, and in which no two *a*'s are adjacent



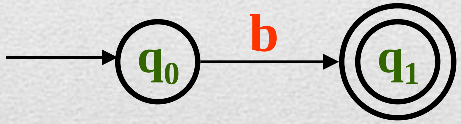
Finite-state Automata

- An FSA defines a regular language over an alphabet Σ:
  - $\varnothing$ is a regular language:

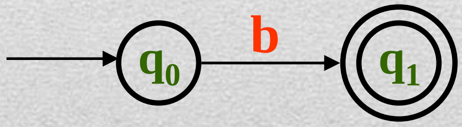


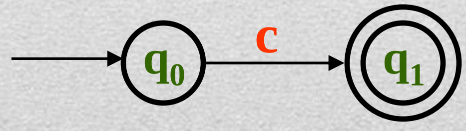
  - Any symbol from Σ is a regular language:

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

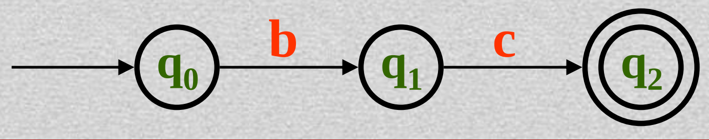


  - Two concatenated regular languages is a regular language:





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



FSA Example

- **Consider the following FSA**

T: {0, 1}

Q: {s1, s2}

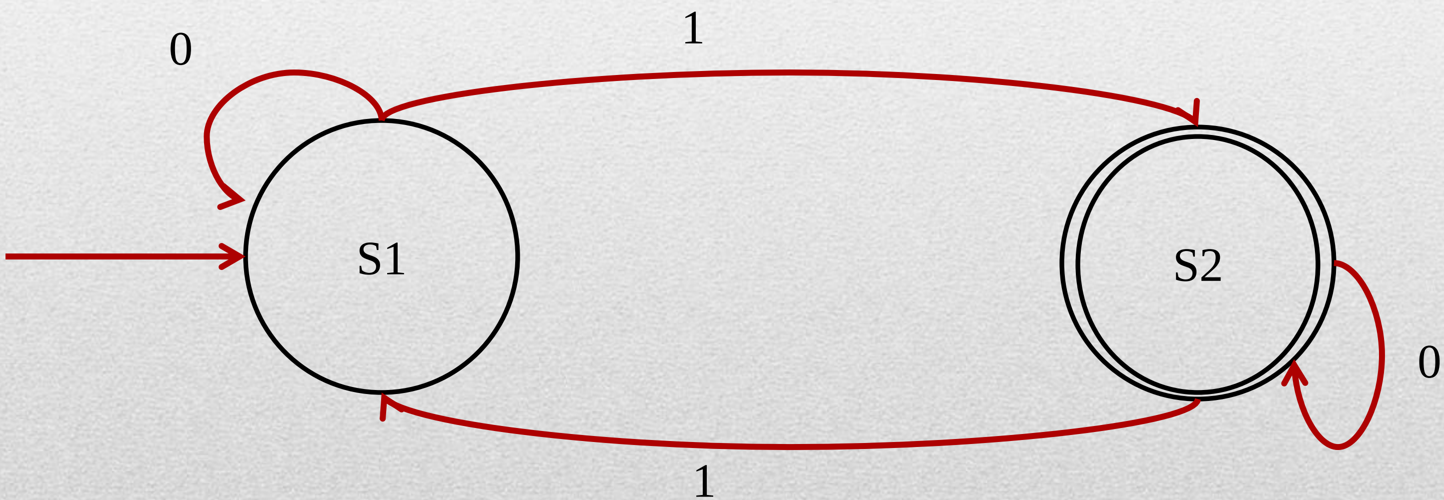
I: s1

F: s2

E:

	0	1
S1	S1	S2
S2	S2	S1

FSA Example



FSA Example

Determine which string is accepted and which is rejected:

- **01101**
- **011011**
- **00000**
- **11111**
- **10101010**

FSA Exercise

- **Consider the following FSA**

T: {a, b, c}

Q: {s1, s2, s3}

I: s1

F: s2, s3

E:

	a	**b**	**c**
S1	S1	S2	S2
S2	S1	S2	S3
S3	S3	S1	S2

FSA Exercise

- abb
- abba
- bcbccc
- caaabbc

FSA Exercise