MECHTRON 2MD3

Data Structures and Algorithms for Mechatronics

Winter 2022

32 Graphs Continued, Finite State Automata

Department of Computing and Software

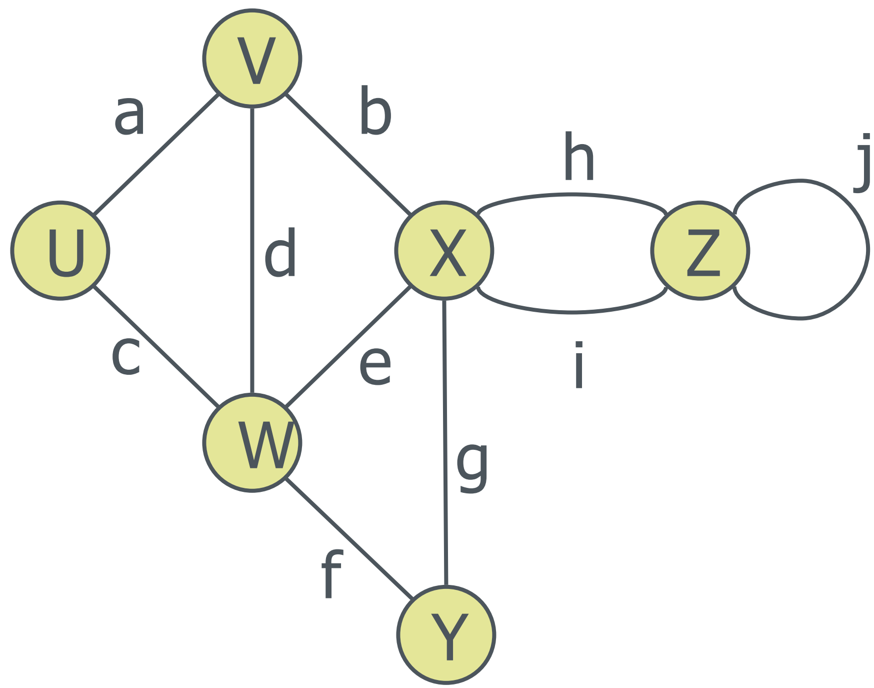
Instructor:

Omid Isfahanialamdari

April 11, 2022

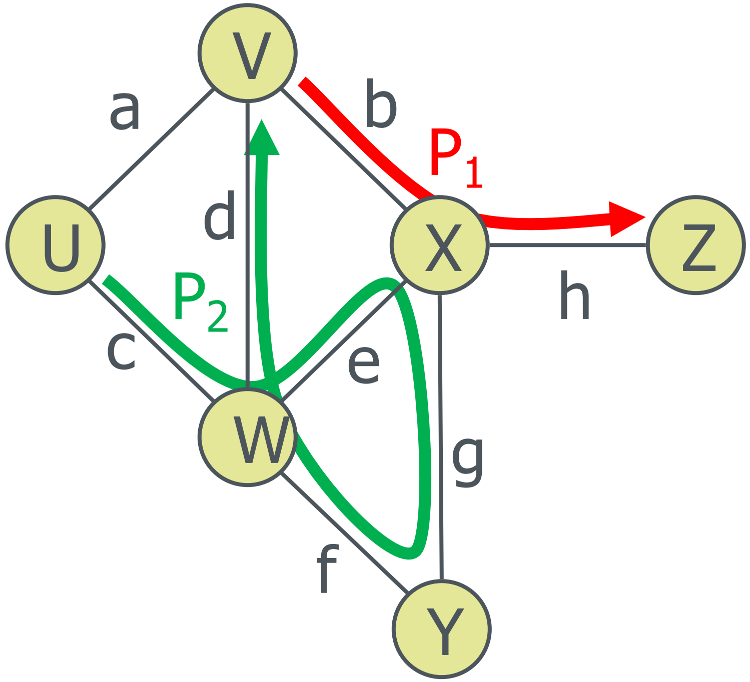
Terminology

- End vertices (or endpoints) of an edge
  - **U** and **V** are the endpoints of **a**
- Edges incident on a vertex
  - **a**, **d**, and **b** are incident on **V**
- Adjacent vertices
  - **U** and **V** are adjacent
- Degree of a vertex
  - **X** has degree **5**
- Parallel edges
  - **h** and **i** are parallel edges
- Self-loop
  - **j** is a self-loop



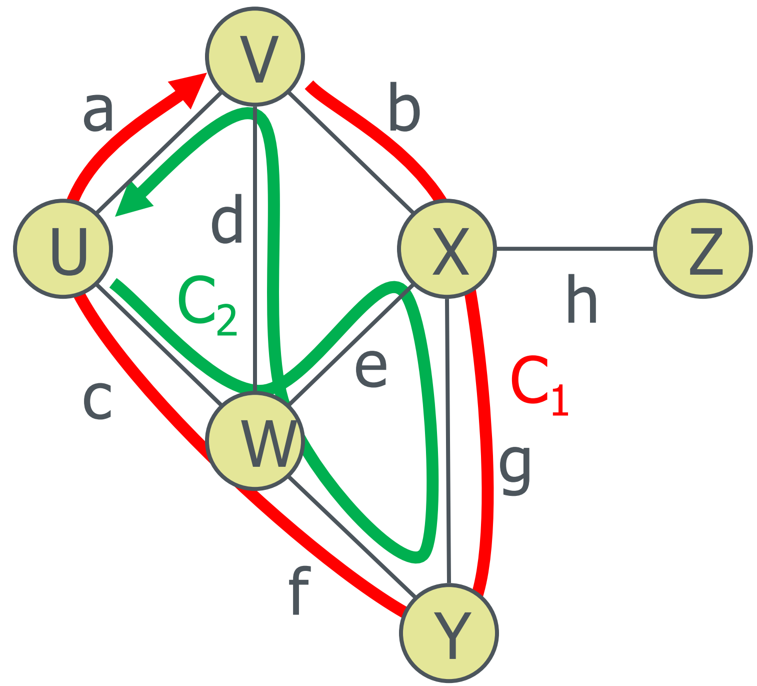
Terminology (cont.)

- Path
  - sequence of alternating vertices and edges
  - begins with a vertex
  - ends with a vertex
  - each edge is preceded and followed by its endpoints
- Simple path
  - path such that all its vertices and edges are distinct
- Examples
  - P_1=(V,b,X,h,Z) is a simple path
  - P_2=(U,c,W,e,X,g,Y,f,W,d,V) is a path that is not simple



Terminology (cont.)

- Cycle
  - circular sequence of alternating vertices and edges
  - each edge is preceded and followed by its endpoints
- Simple cycle
  - cycle such that all its vertices and edges are distinct
- Examples
  - C_1=(V,b,X,g,Y,f,W,c,U,a,↵) is a simple cycle
  - C_2=(U,c,W,e,X,g,Y,f,W,d,V,a,↵) is a cycle that is not simple



Properties

Property 1

$\sum_v \deg(v) = 2m$

Proof: each edge is counted twice

Property 2

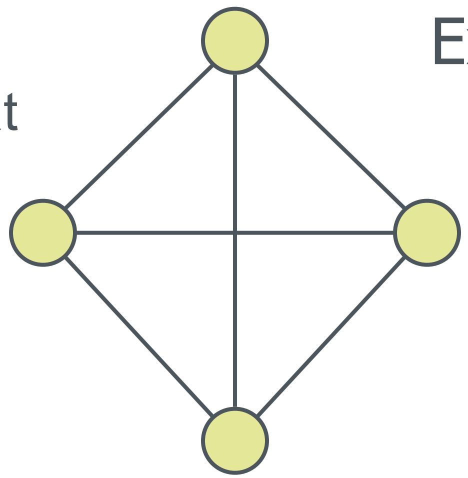
In an undirected graph with no self-loops and no multiple edges

$m \leq n\,(n-1)/2$

Proof: each vertex has degree at most $(n-1)$

Notation

n	number of vertices
m	number of edges
$\deg(v)$	degree of vertex v



Example

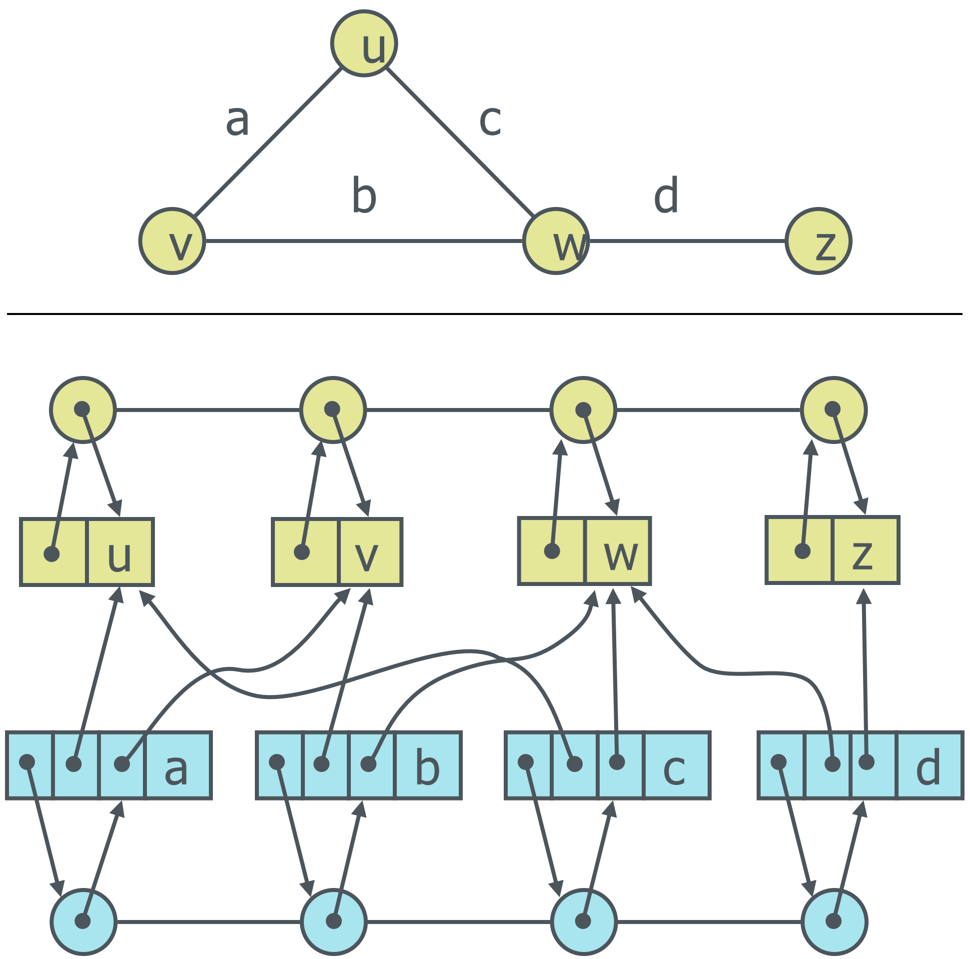
- $n = 4$
- $m = 6$
- $\deg(v) = 3$

Main Methods of the Graph ADT

- Vertices and edges
  - are positions
  - store elements
- Accessor methods
  - e.endVertices(): a list of the two endvertices of e
  - e.opposite(v): the vertex opposite of v on e
  - u.isAdjacentTo(v): true iff u and v are adjacent
  - *v: reference to element associated with vertex v
  - *e: reference to element associated with edge e
- Update methods
  - insertVertex(o): insert a vertex storing element o
  - insertEdge(v, w, o): insert an edge (v,w) storing element o
  - eraseVertex(v): remove vertex v (and its incident edges)
  - eraseEdge(e): remove edge e
- Iterable collection methods
  - incidentEdges(v): list of edges incident to v
  - vertices(): list of all vertices in the graph
  - edges(): list of all edges in the graph

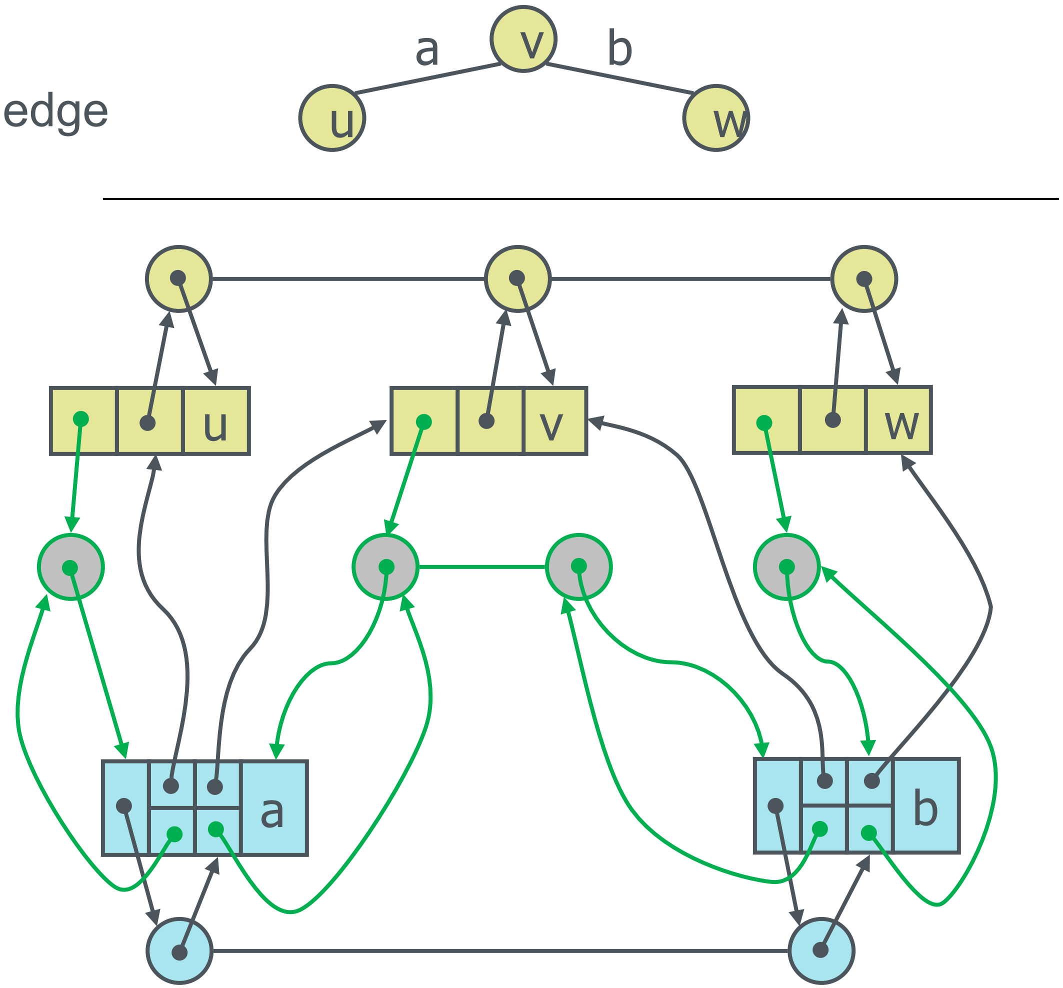
Edge List Structure

- Vertex object
  - element
  - reference to position in vertex sequence
- Edge object
  - element
  - origin vertex object
  - destination vertex object
  - reference to position in edge sequence
- Vertex sequence
  - sequence of vertex objects
- Edge sequence
  - sequence of edge objects



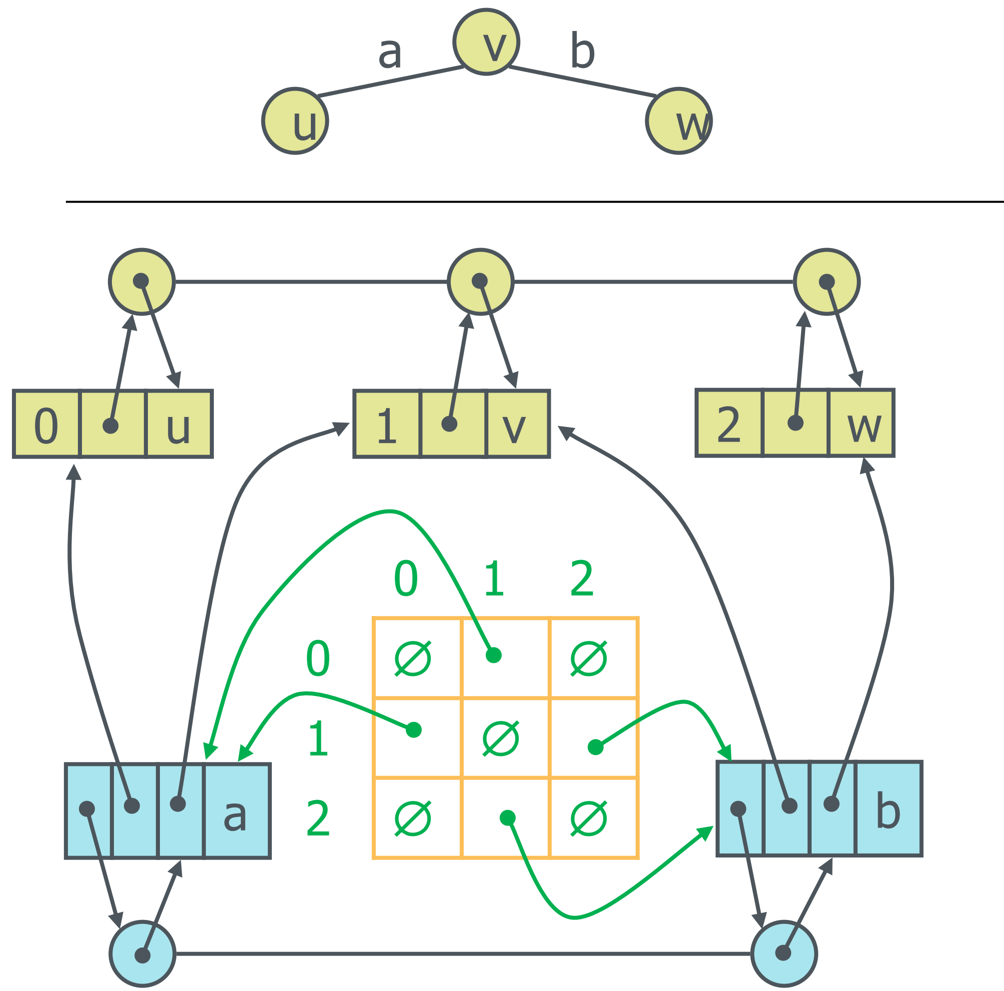
Adjacency List Structure

- Edge list structure
- Incidence sequence for each vertex
    - sequence of references to edge objects of incident edges
- Augmented edge objects
    - references to associated positions in incidence sequences of end vertices



Adjacency Matrix Structure

- Edge list structure
- Augmented vertex objects
  - Integer key (index) associated with vertex
- 2D-array adjacency array
  - Reference to edge object for adjacent vertices
  - Null for non nonadjacent vertices
- The "old fashioned" version just has 0 for no edge and 1 for edge



Finite State Automata and Language Concepts

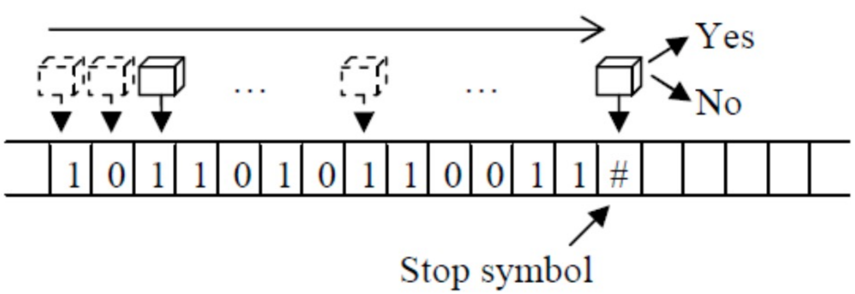
Discrete Systems

- **Discrete System:** A discrete system operates in a sequence of **discrete** steps or has signals taking discrete values.
- Example: Parking Counting System
  - every time a car enters
    - counter++
  - every time a car leaves
    - counter--

  - Each entry departure is a **discrete event**
    - Occurs at **some instant in time**, not continuously over time
    - after every event, the system is in a new **state**

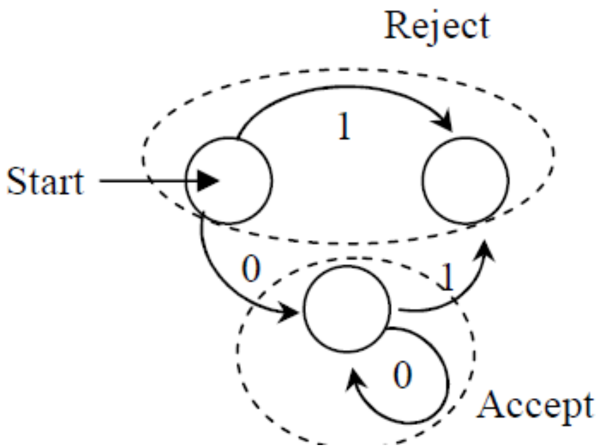
Finite State Automaton

- We use a Finite State Machine (or automaton) (FSM) to model a discrete system
  - Notion of **State**: System's condition at some point in time
  - The **input** to automaton is on a tape or comes from an input stream
  - The machine reads the input **one value at a time** while moving from left to right.
  - Machine's state changes depending on the **input** and **current state** of the machine
  - End of input is logically marked with a # sign
  - After reading the final input, the machine reaches a **final state** that is either:
    - accept
    - reject



Finite State Automaton

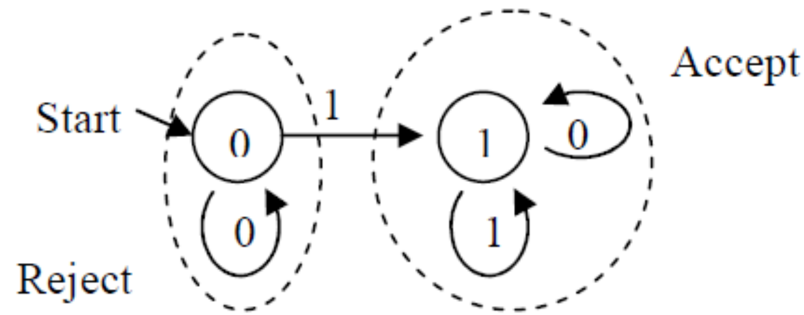
- We use a Finite State Machine (or automaton) (FSM) to model a discrete system
- An FSM is shown with a graph:
  - nodes are **states**
  - edges are **transitions**
    - from one state to another
    - **label** of edges indicate **input**
  - A transition from state **A** to state **B** after reading input **u** is represented by an edge from **A** to **B** labeled with **u.**
  - There is always a unique **initial state**
    - If the machine's final state is an **accept state**, this means that the machines accepts the input, otherwise rejects



McMaster University

Finite State Automaton - Example

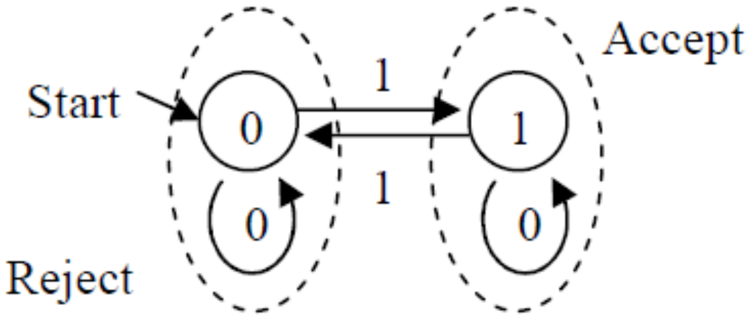
- We use a Finite State Machine (or automaton) (FSM) to model a discrete system



  - This machine determines if a 1 exists in an input
  - It accepts all inputs that have at least a 1
    - 00000100
    - 010101011
    - 00000000 (this will be rejected by the machine)

Finite State Automaton - Example

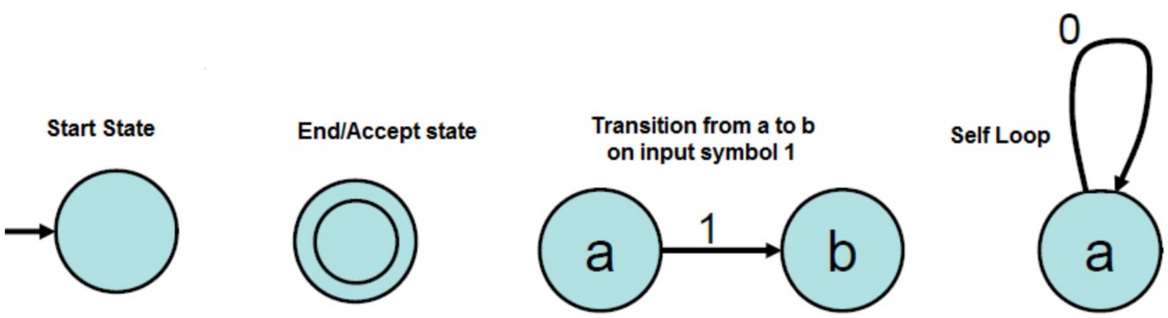
- We use a Finite State Machine (or automaton) (FSM) to model a discrete system



  - This machine determines if there are off number of 1s in an input
  - It accepts all inputs that have at least a 1
    - 00000100101
    - 01110101110
  - 00000011 (this will be rejected by the machine)

Finite State Automaton - Drawing conventions

- We use a Finite State Machine (or automaton) (FSM) to model a discrete system



Language Concepts

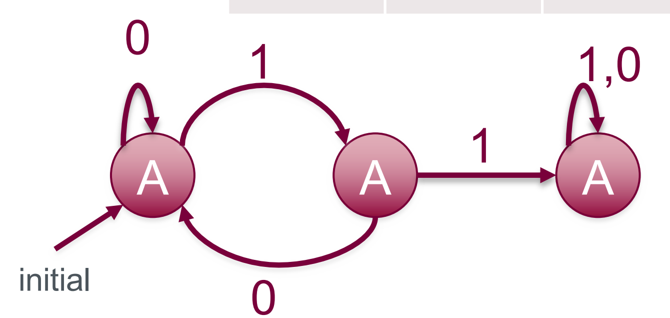
- **Alphabet**: A finite set of symbols
  - Notation: **Σ**
  - Example: $\Sigma = \{a, b\}$
- A **string** (word) over Σ : A finite sequence of symbols from Σ, e.g. {a, b, ab, bb, ...}
- Σ^* (Kleene star): The set of **all finite strings** over an alphabet Σ is the set of lists, each element of which is a member of Σ
  - $\Sigma^* = \{\varepsilon, a, b, aa, ab, ba, bb, \ldots\}$ for $\Sigma = \{a,b\}$
  - ε is a specific symbol representing the Null string
  - $\{0,1\}^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, \ldots\}$
- A language is a subset of Σ^* for some alphabet Σ

Formal Definition of Finite Automata

- We saw that an FSM can accept/reject strings
  - We use this property to define a language
- **A formalism for defining languages**
- **We define a finite Automata to be a 5-tuple:**
  - $(Q, \Sigma, \delta, q_0, F)$
  - Q: A finite set of states
  - Σ: An input alphabet
  - δ: A transition function
    - $\delta : Q \times \Sigma \to Q$
  - q_0 in Q: An initial state
  - F: A set of final states ($F \subseteq Q$, typically)
    - "Final" or "accepting" states.

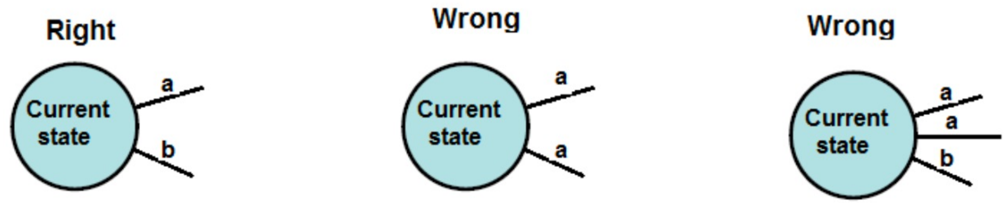
transition table

state	Input applied	
	0	1
A	A	B
B	A	C
C	C	C



Two Types of FA

- Deterministic Finite Automata (DFA)
  - For each input there is one and only one state to which the automaton can transition from its current state.



  - A state from which there is no transition out is called a **trap** or a **dead state**.
  - Every state in a DFA must have outgoing arrows equal to the number of symbols in alphabet.
  - If an arrow is omitted for a state corresponding to a symbol, it is assumed that it leads to a trap state.
- Non-deterministic Finite Automata (NFA)
  - On each input there is a set of states to which the automaton can transition from its current state

Regular Languages

- L(M) is a language recognized by an automaton M.
  - {S | S is accepted by M}
    - set of strings that are accepted by M
- A language that is recognized by a finite automaton is called a Regular Language.
- If we can construct a DFA for a given language then the language is called Regular.
- Regular Expression is another way to represent a Regular Language.
  - Example: a(ba)*
    - means “a” followed by zero or more “ba”s
      - abababa
      - a
- Regular Languages, Finite Automatons, and Regular Expressios are mutually convertible.

Final Words

- **Thank you for attending this class**
- **Hope that I see you in some future classes**
- **I tried my best, but clearly it was not perfect**
  - **This was my first experience in McMaster with > 100 students**
    - **midterm2 location! :(**
- **I don't remember all your names, but I remember your faces**
- **Feel free to contact me, if you want to know about my research works**
- **If you want, you can connect me in LinkedIn:**
  - https://www.linkedin.com/in/omid-alamdari
- **you can contact me:**
  - alamdari@di.unipi.it
  - isfahano@mcmaster.ca

Questions?

Please evaluate this course!
https://evals.mcmaster.ca/
Deadline is tomorrow!
Thank you