Finite Automaton and Regular sets

Machine Model Parlance

- **State of a system:** in one particular time step, description of the system. All relevant information necessary to determine where the system is at present and this helps to determine what action the system will take in the next time step.

Machine Model Parlance

- **State of a system:** in one particular time step, description of the system. All relevant information necessary to determine where the system is at present and this helps to determine what action the system will take in the next time step.
- **Transitions:** changes of state. Can happen spontaneously or in response to external inputs.

Machine Model Parlance

- **State of a system:** in one particular time step, description of the system. All relevant information necessary to determine where the system is at present and this helps to determine what action the system will take in the next time step.
- **Transitions:** changes of state. Can happen spontaneously or in response to external inputs.
- **Math assumption:** no time step, instantaneous transitions. Clock cycles in digital computers allow us to treat computers as digital instead of analog devices.

Machine Model Parlance

- **State of a system:** in one particular time step, description of the system. All relevant information necessary to determine where the system is at present and this helps to determine what action the system will take in the next time step.
- **Transitions:** changes of state. Can happen spontaneously or in response to external inputs.
- **Math assumption:** no time step, instantaneous transitions. Clock cycles in digital computers allow us to treat computers as digital instead of analog devices.
- **State transition system:** system modelled in terms of states and transitions. Eg. Digital watch, elevator, Rubic's cube

Machine Model Parlance

- **State of a system:** in one particular time step, description of the system. All relevant information necessary to determine where the system is at present and this helps to determine what action the system will take in the next time step.
- **Transitions:** changes of state. Can happen spontaneously or in response to external inputs.
- **Math assumption:** no time step, instantaneous transitions. Clock cycles in digital computers allow us to treat computers as digital instead of analog devices.
- **State transition system:** system modelled in terms of states and transitions. Eg. Digital watch, elevator, Rubic's cube
- System with *finite states and transitions* among them – finite state transition system. This leads to the math model called **finite automaton**.

Deterministic Finite Automaton: Math representation

DFA: Structure $M = (Q, \Sigma, \delta, s, F)$
Q – finite set of states
Σ – Finite alphabet
δ – $Q \times \Sigma \to Q$, state transition function. Which state to move to in response to input alphabet. Finite set of ordered pairs.
s – start state
F – subset of Q called accept/final states.
Note: A DFA structure can be represented as a finite string – pick favourite encoding. The structure definition involves finite items, so the structure can be represented as a finite string.

An Example DFA

$$M = (Q, \Sigma, \delta, s, F)$$

$$Q = \{0, 1, 2\} \quad , \quad \Sigma = \{a, b\}$$

$$\delta: \quad \delta(0, a) = 1 \;,\; \delta(0, b) = 0$$

$$\delta(1, a) = 2 \;,\; \delta(1, b) = 1$$

$$\delta(2, a) = \delta(2, b) = 2$$

$$s = 0 \quad , \quad F = \{2\}$$

Transition Table

$\delta: \delta(0,a)=1, \delta(0,b)=0, \delta(1,a)=2, \delta(1,b)=1, \delta(2,a)=\delta(2,b)=2$

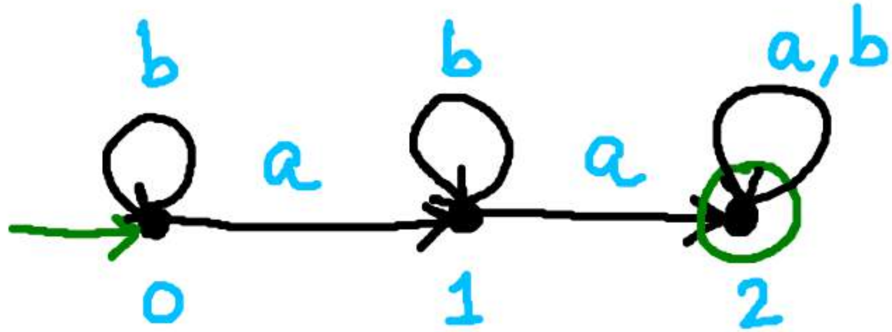
$s=0, \quad F=\{2\}$

	a	b
→0	1	0
1	2	1
2F	2	2

Transition Diagram

$$\delta(0,a)=1,\ \delta(0,b)=0,\ \delta(1,a)=2,\ \delta(1,b)=1,\ \delta(2,a)=\delta(2,b)=2$$

$$s=0,\quad F=\{2\}$$



Input String to a DFA M

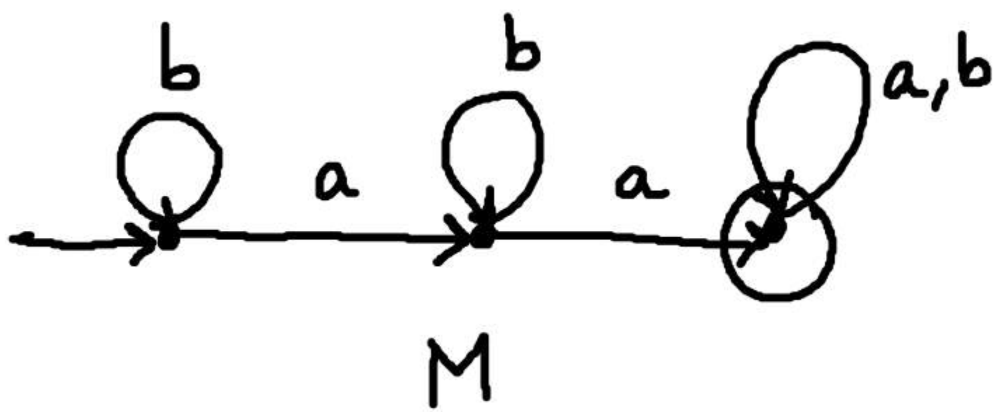
- Following a path along the transition diagram to see in what state M ends up when the input string ends.

Input String to a DFA M

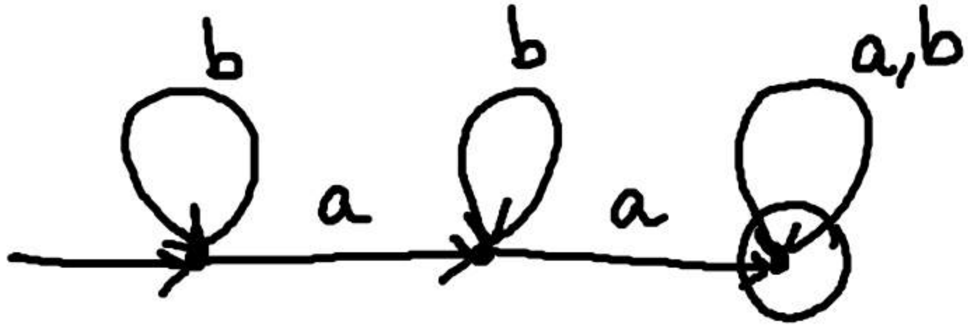
- Following a path along the transition diagram to see in what state M ends up when the input string ends.
- String accepted if the end is in F, rejected otherwise.

Input String to a DFA M

- Following a path along the transition diagram to see in what state M ends up when the input string ends.
- String accepted if the end is in F, rejected otherwise.
- **Ex**: What are the strings accepted in the example above?



Strings accepted by Example DFA



Accepted strings: Must have at least 2 a's

babbb ✗

baab ✓

abbbbab ✓

Formalisation: Function for processing an input string

- Processing an input alphabet: Transition function
  $\delta : Q \times \Sigma \rightarrow Q$.

Formalisation: Function for processing an input string

- Processing an input alphabet: Transition function $\delta : Q \times \Sigma \rightarrow Q$.
- Processing an input string: Extend δ to a multistep transition function, $\hat{\delta} : Q \times \Sigma^* \rightarrow Q$
  $\hat{\delta}(q, \epsilon) = q$
  $\hat{\delta}(q, xa) = \delta(\hat{\delta}(q, x), a)$ – Inductive definition

Formalisation: Function for processing an input string

- Processing an input alphabet: Transition function $\delta : Q \times \Sigma \to Q$.
- Processing an input string: Extend δ to a multistep transition function, $\hat{\delta} : Q \times \Sigma^* \to Q$
  $\hat{\delta}(q, \epsilon) = q$
  $\hat{\delta}(q, xa) = \delta(\hat{\delta}(q, x), a)$ – Inductive definition
- Each step is determined by δ.

Formalisation: Function for processing an input string

- Processing an input alphabet: Transition function $\delta : Q \times \Sigma \to Q$.
- Processing an input string: Extend δ to a multistep transition function, $\hat{\delta} : Q \times \Sigma^* \to Q$
  $\hat{\delta}(q, \epsilon) = q$
  $\hat{\delta}(q, xa) = \delta(\hat{\delta}(q, x), a)$ – Inductive definition
- Each step is determined by δ.
- Induction step says $q \to^{xa} q'$ same as $q \to^{x} p \to^{a} q'$

Formalisation: Function for processing an input string

- Processing an input alphabet: Transition function $\delta : Q \times \Sigma \to Q$.
- Processing an input string: Extend δ to a multistep transition function, $\hat{\delta} : Q \times \Sigma^* \to Q$
  $\hat{\delta}(q, \epsilon) = q$
  $\hat{\delta}(q, xa) = \delta(\hat{\delta}(q, x), a)$ – Inductive definition
- Each step is determined by δ.
- Induction step says $q \to^{xa} q'$ same as $q \to^{x} p \to^{a} q'$
- $\hat{\delta}$ and δ agree on length 1 strings:
  $\hat{\delta}(q, a) = \hat{\delta}(q, \epsilon.a) = \delta(\hat{\delta}(q, \epsilon), a)$ [Inductive definition]
  $= \delta(q, a)$ [Base case of definition]
  x accepted if $\hat{\delta}(s, x) \in F$, otherwise not.

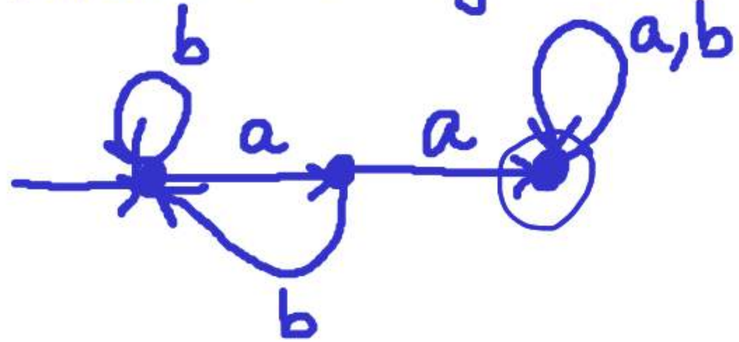
Language accepted by DFA M

$L(M) = \{x | \hat{\delta}(s, x) \in F\}$
Such a set is called a *regular set*. A is a regular set if $A = L(M)$ for a DFA M.
Eg: Is the set $\{x \in \{a, b\}^* | x \text{ contains two consecutive a's}\}$ a regular set?

Language accepted by DFA M

$L(M) = \{x | \hat{\delta}(s, x) \in F\}$
Such a set is called a *regular set*. A is a regular set if $A = L(M)$ for a DFA M.
Eg: Is the set $\{x \in \{a, b\}^* | x$ contains two consecutive a's$\}$ a regular set?

Transition diagram:



$$\hat{\delta}(q, xy) = \hat{\delta}(\hat{\delta}(q, x), y)$$

- Proof: Induction on length of y.

$$\hat{\delta}(q, xy) = \hat{\delta}(\hat{\delta}(q, x), y)$$

- Proof: Induction on length of y.
- Base Case 1: If $y = \epsilon$, for any x LHS $\hat{\delta}(q, x.\epsilon) = \hat{\delta}(q, x)$,

$\hat{\delta}(q, xy) = \hat{\delta}(\hat{\delta}(q, x), y)$

- Proof: Induction on length of y.
- Base Case 1: If $y = \epsilon$, for any x LHS $\hat{\delta}(q, x.\epsilon) = \hat{\delta}(q, x)$,
- Similarly, by definition of $\hat{\delta}$, RHS $\hat{\delta}(\hat{\delta}(q, x), \epsilon) = \hat{\delta}(q, x)$
  So $LHS = RHS$.

$\hat{\delta}(q, xy) = \hat{\delta}(\hat{\delta}(q, x), y)$

- Proof: Induction on length of y.
- Base Case 1: If $y = \epsilon$, for any x LHS $\hat{\delta}(q, x.\epsilon) = \hat{\delta}(q, x)$,
- Similarly, by definition of $\hat{\delta}$, RHS $\hat{\delta}(\hat{\delta}(q, x), \epsilon) = \hat{\delta}(q, x)$
  So $LHS = RHS$.
- Base Case 2: $y = a, a \in \Sigma$: For any x
  $\hat{\delta}(q, xa) = \delta(\hat{\delta}(q, x), a)$ [Inductive definition]
  $= \hat{\delta}(\hat{\delta}(q, x), a)$ [By equality on length 1 strings]

$\hat{\delta}(q, xy) = \hat{\delta}(\hat{\delta}(q, x), y)$ (contd.)

- IH: True for all x and all y' with length strictly less than y.

$\hat{\delta}(q, xy) = \hat{\delta}(\hat{\delta}(q, x), y)$ (contd.)

- IH: True for all x and all y' with length strictly less than y.
- Consider the string $xy = xy'a$, $a \in \Sigma$.

$\hat{\delta}(q, xy) = \hat{\delta}(\hat{\delta}(q, x), y)$ (contd.)

- IH: True for all x and all y' with length strictly less than y.
- Consider the string $xy = xy'a$, $a \in \Sigma$.
- $\hat{\delta}(q, xy'a) = \delta(\hat{\delta}(q, xy'), a) = \hat{\delta}(\hat{\delta}(q, xy'), a)$ [equality on length 1 strings]

$\hat{\delta}(q, xy) = \hat{\delta}(\hat{\delta}(q, x), y)$ (contd.)

- IH: True for all x and all y' with length strictly less than y.
- Consider the string $xy = xy'a$, $a \in \Sigma$.
- $\hat{\delta}(q, xy'a) = \delta(\hat{\delta}(q, xy'), a) = \hat{\delta}(\hat{\delta}(q, xy'), a)$ [equality on length 1 strings]
- $= \hat{\delta}(\hat{\delta}(\hat{\delta}(q, x), y'), a)$ [IH]

$\hat{\delta}(q, xy) = \hat{\delta}(\hat{\delta}(q, x), y)$ (contd.)

- IH: True for all x and all y' with length strictly less than y.
- Consider the string $xy = xy'a$, $a \in \Sigma$.
- $\hat{\delta}(q, xy'a) = \delta(\hat{\delta}(q, xy'), a) = \hat{\delta}(\hat{\delta}(q, xy'), a)$ [equality on length 1 strings]
- $= \hat{\delta}(\hat{\delta}(\hat{\delta}(q, x), y'), a)$ [IH]
- (Let $q' = \hat{\delta}(q, x)$)

$\hat{\delta}(q, xy) = \hat{\delta}(\hat{\delta}(q, x), y)$ (contd.)

- IH: True for all x and all y' with length strictly less than y.
- Consider the string $xy = xy'a$, $a \in \Sigma$.
- $\hat{\delta}(q, xy'a) = \delta(\hat{\delta}(q, xy'), a) = \hat{\delta}(\hat{\delta}(q, xy'), a)$ [equality on length 1 strings]
- $= \hat{\delta}(\hat{\delta}(\hat{\delta}(q, x), y'), a)$ [IH]
- (Let $q' = \hat{\delta}(q, x)$)
- $= \hat{\delta}(\hat{\delta}(q', y'), a)$

$\hat{\delta}(q, xy) = \hat{\delta}(\hat{\delta}(q, x), y)$ (contd.)

- IH: True for all x and all y' with length strictly less than y.
- Consider the string $xy = xy'a$, $a \in \Sigma$.
- $\hat{\delta}(q, xy'a) = \delta(\hat{\delta}(q, xy'), a) = \hat{\delta}(\hat{\delta}(q, xy'), a)$ [equality on length 1 strings]
- $= \hat{\delta}(\hat{\delta}(\hat{\delta}(q, x), y'), a)$ [IH]
- (Let $q' = \hat{\delta}(q, x)$)
- $= \hat{\delta}(\hat{\delta}(q', y'), a)$
- $= \hat{\delta}(q', y'a)$ [Base Case 2: length 1 string]

$\hat{\delta}(q, xy) = \hat{\delta}(\hat{\delta}(q, x), y)$ (contd.)

- IH: True for all x and all y' with length strictly less than y.
- Consider the string $xy = xy'a$, $a \in \Sigma$.
- $\hat{\delta}(q, xy'a) = \delta(\hat{\delta}(q, xy'), a) = \hat{\delta}(\hat{\delta}(q, xy'), a)$ [equality on length 1 strings]
- $= \hat{\delta}(\hat{\delta}(\hat{\delta}(q, x), y'), a)$ [IH]
- (Let $q' = \hat{\delta}(q, x)$)
- $= \hat{\delta}(\hat{\delta}(q', y'), a)$
- $= \hat{\delta}(q', y'a)$ [Base Case 2: length 1 string]
- $= \hat{\delta}(\hat{\delta}(q, x), y'a) = \hat{\delta}(\hat{\delta}(q, x), y)$