COL 352 Introduction to Automata and Theory of Computation

Nikhil Balaji

Bharti 420
Indian Institute of Technology, Delhi
nbalaji@cse.iitd.ac.in

January 5, 2023

Module 1: Finite Automata

Formal Languages and Problems

- An alphabet $\Sigma = \{a, b, c\}$ is a finite set of letters.
- A language is a set of strings over some alphabet.
- Σ^* is the set of all strings over Σ, e.g. $aabbaa \in \Sigma^*$.
- A language L over Σ is then a subset of Σ^*, e.g.,

Formal Languages and Problems

- An alphabet $\Sigma = \{a, b, c\}$ is a finite set of letters.
- A language is a set of strings over some alphabet.
- Σ^* is the set of all strings over Σ, e.g. $aabbaa \in \Sigma^*$.
- A language L over Σ is then a subset of Σ^*, e.g.,
  - $L_{even} = \{w \in \Sigma^* : w$ is of even length $\}$.

Formal Languages and Problems

- An alphabet $\Sigma = \{a, b, c\}$ is a finite set of letters.
- A language is a set of strings over some alphabet.
- Σ^* is the set of all strings over Σ, e.g. $aabbaa \in \Sigma^*$.
- A language L over Σ is then a subset of Σ^*, e.g.,
  - $L_{even} = \{w \in \Sigma^* : w$ is of even length $\}$.
  - $L_{a^n b^n} = \{w \in \Sigma^* : w$ is of the form $a^n b^n$ for $n \geq 0$ $\}$

Formal Languages and Problems

- An alphabet $\Sigma = \{a, b, c\}$ is a finite set of letters.
- A language is a set of strings over some alphabet.
- Σ^* is the set of all strings over Σ, e.g. $aabbaa \in \Sigma^*$.
- A language L over Σ is then a subset of Σ^*, e.g.,
  - $L_{even} = \{w \in \Sigma^* : w$ is of even length $\}$.
  - $L_{a^n b^n} = \{w \in \Sigma^* : w$ is of the form $a^n b^n$ for $n \geq 0$ $\}$
- Every decision problem $f : \Sigma^* \to \{0, 1\}$ corresponds to a language $L_f = \{w \in \Sigma^* | f(w) = 1\}$.

Formal Languages and Problems

- An alphabet $\Sigma = \{a, b, c\}$ is a finite set of letters.
- A language is a set of strings over some alphabet.
- Σ^* is the set of all strings over Σ, e.g. $aabbaa \in \Sigma^*$.
- A language L over Σ is then a subset of Σ^*, e.g.,
  - $L_{even} = \{w \in \Sigma^* : w$ is of even length $\}$.
  - $L_{a^n b^n} = \{w \in \Sigma^* : w$ is of the form $a^n b^n$ for $n \geq 0$ $\}$
- Every decision problem $f : \Sigma^* \to \{0, 1\}$ corresponds to a language $L_f = \{w \in \Sigma^* | f(w) = 1\}$.
- PRIMES $= \{w \in \{a\}^* || w|$ is prime $\}$

Formal Languages and Problems

- An alphabet $\Sigma = \{a, b, c\}$ is a finite set of letters.
- A language is a set of strings over some alphabet.
- Σ^* is the set of all strings over Σ, e.g. $aabbaa \in \Sigma^*$.
- A language L over Σ is then a subset of Σ^*, e.g.,
    - $L_{even} = \{w \in \Sigma^* : w$ is of even length $\}$.
    - $L_{a^n b^n} = \{w \in \Sigma^* : w$ is of the form $a^n b^n$ for $n \geq 0$ $\}$
- Every decision problem $f : \Sigma^* \to \{0, 1\}$ corresponds to a language $L_f = \{w \in \Sigma^* | f(w) = 1\}$.
- PRIMES = $\{w \in \{a\}^* || w|$ is prime $\}$
- CONNECTED = $\{w \in \{0, 1\}^* | G_w$ is connected $\}$

A Design Question

A Design Question

So, given a language : Can we define an automaton that accepts exactly that language?

A Design Question

So, given a language : Can we define an automaton that accepts exactly that language?

Example

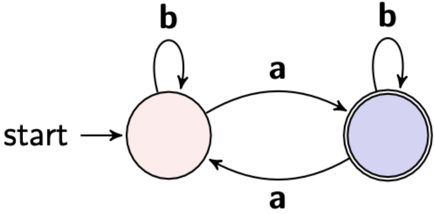
- Fix $\Sigma = \{a, b\}$
- Let L be all words over Σ that have odd number of a's.
- What automaton would accept L?

A Design Question

So, given a language : Can we define an automaton that accepts exactly that language?

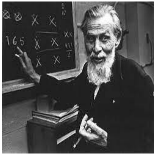
Example

- Fix $\Sigma = \{a, b\}$
- Let L be all words over Σ that have odd number of a's.
- What automaton would accept L?



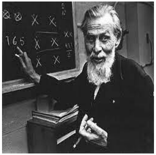
Finite Automata

Finite Automata



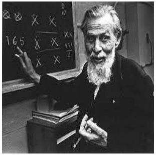
- Introduced first by two neuro-psychologists Warren S. McCullough and Walter Pitts in 1943 as a model for human brain.

Finite Automata



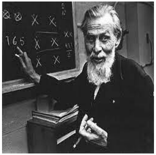
- Introduced first by two neuro-psychologists Warren S. McCullough and Walter Pitts in 1943 as a model for human brain.
- Computation with finite memory!

Finite Automata



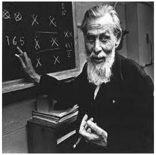
- Introduced first by two neuro-psychologists Warren S. McCullough and Walter Pitts in 1943 as a model for human brain.
- Computation with finite memory!
- Finite automata can naturally model microprocessors and even software programs working on variables with bounded domain.

Finite Automata



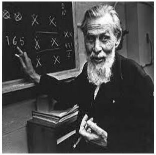
- Introduced first by two neuro-psychologists Warren S. McCullough and Walter Pitts in 1943 as a model for human brain.
- Computation with finite memory!
- Finite automata can naturally model microprocessors and even software programs working on variables with bounded domain.
- Capture so-called regular sets of sequences that occur in many different fields (logic, algebra, regEx)

Finite Automata



- Introduced first by two neuro-psychologists Warren S. McCullough and Walter Pitts in 1943 as a model for human brain.
- Computation with finite memory!
- Finite automata can naturally model microprocessors and even software programs working on variables with bounded domain.
- Capture so-called regular sets of sequences that occur in many different fields (logic, algebra, regEx)
- Nice theoretical properties

Finite Automata



- Introduced first by two neuro-psychologists Warren S. McCullough and Walter Pitts in 1943 as a model for human brain.
- Computation with finite memory!
- Finite automata can naturally model microprocessors and even software programs working on variables with bounded domain.
- Capture so-called regular sets of sequences that occur in many different fields (logic, algebra, regEx)
- Nice theoretical properties
- Applications in digital circuit/protocol verification, compilers, pattern recognition, etc

Finite Automata you use daily

Exercise: Design Automata for these!

Finite state automata

Example 1

Finite state automata

Example 1

Input: Text file over the alphabet $\{a, b\}$

Finite state automata

Example 1

Input: Text file over the alphabet $\{a, b\}$

Check: does the file end with the string 'aa'

Finite state automata

Example 1

Input: Text file over the alphabet $\{a, b\}$

Check: does the file end with the string ‘aa’

Finite state automata

Example 1

Input: Text file over the alphabet $\{a, b\}$

Check: does the file end with the string 'aa'

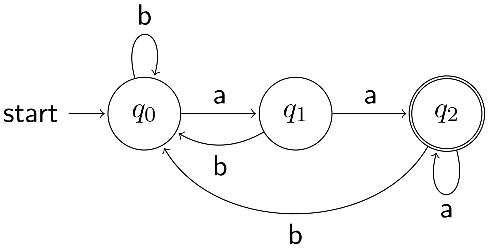
Idea: Start scanning from left, if you see an 'a' check if the next character is also 'a'. If yes, accept, else reset. If you reach end of string, reject.

Finite state automata

Example 1

Input: Text file over the alphabet $\{a, b\}$

Check: does the file end with the string 'aa'



Finite state automata

Example 2

Input: Text file over the alphabet $\{a, b\}$

Finite state automata

Example 2

Input: Text file over the alphabet $\{a, b\}$

Check: does the file contain the string 'aa'

Finite state automata

Example 2

Input: Text file over the alphabet $\{a, b\}$

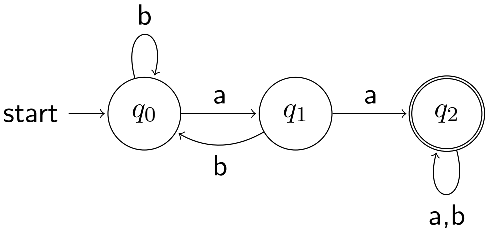
Check: does the file contain the string 'aa'

Finite state automata

Example 2

Input: Text file over the alphabet $\{a, b\}$

Check: does the file contain the string ‘aa’



Finite state automata

Example 3

Input: $w \in \{a, b\}^*$

Check: does w have odd number of a's?

Finite state automata

Example 3

Input: $w \in \{a, b\}^*$

Check: does w have odd number of a's? i.e. is $\#_a(w) \equiv 1 \pmod 2$?

Finite state automata

Example 3

Input: $w \in \{a, b\}^*$

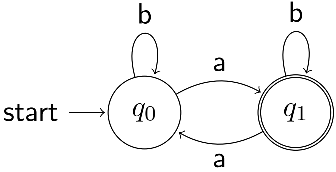
Check: does w have odd number of a's? i.e. is $\#_a(w) \equiv 1 \pmod 2$?

Finite state automata

Example 3

Input: $w \in \{a, b\}^*$

Check: does w have odd number of a's? i.e. is $\#_a(w) \equiv 1 \pmod 2$?



Exercise: Design an automaton to check if w has an even number of a's in every block of length 4 in w.

Deterministic Finite State Automata

An automaton has

- (Finite set of) States
- (Finite) Alphabet
- Initial state
- Accepting/final state
- (Finite) Set of transitions

More formally . . .

More formally ...

Definition (DFA)

A deterministic finite state automaton (DFA) $A = (Q, \Sigma, q_0, F, \delta)$, where

More formally ...

Definition (DFA)

A deterministic finite state automaton (DFA) $A = (Q, \Sigma, q_0, F, \delta)$, where

- Q is a set of states,
- Σ is the input alphabet,
- $q_0 \in Q$ is the initial state,
- $F \subseteq Q$ is the set of final states,
- δ is a set of transitions, i.e. $\delta : Q \times \Sigma \to Q$

More formally . . .

Definition (DFA)

A deterministic finite state automaton (DFA) $A = (Q, \Sigma, q_0, F, \delta)$, where

Q is a set of states,

Σ is the input alphabet,

$q_0 \in Q$ is the initial state,

$F \subseteq Q$ is the set of final states,

δ is a set of transitions, i.e. $\delta : Q \times \Sigma \to Q$ or

$\delta \subseteq Q \times \Sigma \times Q$ such that

More formally ...

Definition (DFA)

A deterministic finite state automaton (DFA) $A = (Q, \Sigma, q_0, F, \delta)$, where

- Q is a set of states,
- Σ is the input alphabet,
- $q_0 \in Q$ is the initial state,
- $F \subseteq Q$ is the set of final states,
- δ is a set of transitions, i.e. $\delta : Q \times \Sigma \to Q$ or
  $\delta \subseteq Q \times \Sigma \times Q$ such that
  $\forall q \in Q, \forall a \in \Sigma,\ |\delta(q, a)| \leq 1$.

Run of an automaton

Definition (Run of a DFA)

Run of an automaton

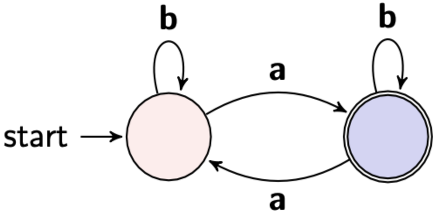
Definition (Run of a DFA)

Let $A = (Q, \Sigma, \delta, q_0, F)$ be a DFA. A run of A on word $w = a_1 \dots a_n$ is a sequence of states $q_0, \dots, q_n$ such that $q_i = \delta(q_{i-1}, a_i)$ for all $1 \leq i \leq n$.

Run of an automaton

Definition (Run of a DFA)

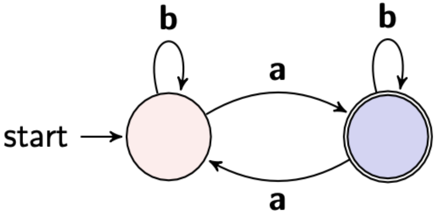
Let $A = (Q, \Sigma, \delta, q_0, F)$ be a DFA. A run of A on word $w = a_1 \ldots a_n$ is a sequence of states $q_0, \ldots, q_n$ such that $q_i = \delta(q_{i-1}, a_i)$ for all $1 \leq i \leq n$.



Run of an automaton

Definition (Run of a DFA)

Let $A = (Q, \Sigma, \delta, q_0, F)$ be a DFA. A run of A on word $w = a_1 \ldots a_n$ is a sequence of states $q_0, \ldots, q_n$ such that $q_i = \delta(q_{i-1}, a_i)$ for all $1 \leq i \leq n$.

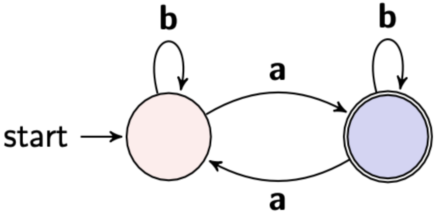


Consider the word $a\ b\ a\ b\ a$

Run of an automaton

Definition (Run of a DFA)

Let $A = (Q, \Sigma, \delta, q_0, F)$ be a DFA. A run of A on word $w = a_1 \ldots a_n$ is a sequence of states $q_0, \ldots, q_n$ such that $q_i = \delta(q_{i-1}, a_i)$ for all $1 \le i \le n$.



Consider the word $a\ b\ a\ b\ a$

Run gives the sequence of states: $q_0\ q_1\ q_0\ q_1\ q_0$.

Acceptance by DFA

Acceptance by DFA

Definition (Acceptance)

A word w is accepted by DFA A if there is a run of A on word w that reaches (ends in) an accepting state.

Acceptance by DFA

Definition (Acceptance)

A word w is accepted by DFA A if there is a run of A on word w that reaches (ends in) an accepting state.

Extended Transition Function

Let $\hat{\delta} : Q \times \Sigma^* \to Q$ be defined as:

$$\hat{\delta}(q, \varepsilon) = q$$

Acceptance by DFA

Definition (Acceptance)

A word w is accepted by DFA A if there is a run of A on word w that reaches (ends in) an accepting state.

Extended Transition Function

Let $\hat{\delta} : Q \times \Sigma^* \to Q$ be defined as:

$$\begin{aligned} \hat{\delta}(q, \varepsilon) &= q \\ \hat{\delta}(q, wa) &= \delta(\hat{\delta}(q, w), a) \end{aligned}$$

Acceptance by DFA

Definition (Acceptance)

A word w is accepted by DFA A if there is a run of A on word w that reaches (ends in) an accepting state.

Extended Transition Function

Let $\hat{\delta} : Q \times \Sigma^* \to Q$ be defined as:

$$\begin{aligned} \hat{\delta}(q, \varepsilon) &= q \\ \hat{\delta}(q, wa) &= \delta(\hat{\delta}(q, w), a) \end{aligned}$$

So A accepts w iff $\hat{\delta}(q_0, w) \in F$

Acceptance by DFA

Definition (Acceptance)

A word w is accepted by DFA A if there is a run of A on word w that reaches (ends in) an accepting state.

Extended Transition Function

Let $\hat{\delta} : Q \times \Sigma^* \to Q$ be defined as:

$$\begin{array}{rcl} \hat{\delta}(q, \varepsilon) & = & q \\ \hat{\delta}(q, wa) & = & \delta(\hat{\delta}(q, w), a) \end{array}$$

So A accepts w iff $\hat{\delta}(q_0, w) \in F$ else it is rejected (i.e, when $\hat{\delta}(q_0, w) \notin F$)

Acceptance by DFA

Definition (Acceptance)

A word w is accepted by DFA A if there is a run of A on word w that reaches (ends in) an accepting state.

Extended Transition Function

Let $\hat{\delta} : Q \times \Sigma^* \to Q$ be defined as:

$$\begin{aligned}\hat{\delta}(q, \varepsilon) &= q \\ \hat{\delta}(q, wa) &= \delta(\hat{\delta}(q, w), a)\end{aligned}$$

So A accepts w iff $\hat{\delta}(q_0, w) \in F$ else it is rejected (i.e, when $\hat{\delta}(q_0, w) \notin F$)

$$L(A) = \{w \in \Sigma^* \mid \hat{\delta}(q_0, w) \in F\}$$

Regular languages

REG

A language is said to be a **regular** if it is accepted by some DFA.
L is a regular language if there exists some DFA A such that $L(A) = L$

Examples

$$
\begin{aligned}
L &= \{w \in \{a,b\}^* \mid w \text{ ends with aa}\} \\
L' &= \{w \in \{a,b\}^* \mid w \text{ contains aa}\} \\
L_{odd} &= \{w \in \{a,b\}^* \mid w \text{ contains odd number of a}\}
\end{aligned}
$$

Regular languages

REG

A language is said to be a **regular** if it is accepted by some DFA.
L is a regular language if there exists some DFA A such that $L(A) = L$

Examples

$$
\begin{aligned}
L &= \{w \in \{a,b\}^* \mid w \text{ ends with aa}\} \\
L' &= \{w \in \{a,b\}^* \mid w \text{ contains aa}\} \\
L_{odd} &= \{w \in \{a,b\}^* \mid w \text{ contains odd number of a}\} \\
L_3 &= \{w \in \{0,1\}^* \mid w \text{ encodes a number in binary divisible by } 3\}
\end{aligned}
$$

Can we solve all problems using computers?

Theorem (Turing (1936))

There are some problems for which it is impossible to write a program solving it correctly on all inputs.

An impossibility result

Theorem

There exists a language for which there is no finite automata accepting it.

Intuitive proof

-

An impossibility result

Theorem

There exists a language for which there is no finite automata accepting it.

Intuitive proof

- The number of finite automata are countably infinite (why?)

An impossibility result

Theorem

There exists a language for which there is no finite automata accepting it.

Intuitive proof

- The number of finite automata are countably infinite (why?)
- Consider the set of languages over alphabet $\{0, 1\}$.
- $\{0, 1\}^*$ is countably infinite.

An impossibility result

Theorem

There exists a language for which there is no finite automata accepting it.

Intuitive proof

- The number of finite automata are countably infinite (why?)
- Consider the set of languages over alphabet $\{0, 1\}$.
- $\{0, 1\}^*$ is countably infinite.
- Hence the set of all languages over $\{0, 1\}$ is the power-set of the set of all strings.

An impossibility result

Theorem

There exists a language for which there is no finite automata accepting it.

Intuitive proof

- The number of finite automata are countably infinite (why?)
- Consider the set of languages over alphabet $\{0, 1\}$.
- $\{0, 1\}^*$ is countably infinite.
- Hence the set of all languages over $\{0, 1\}$ is the power-set of the set of all strings.
- By Cantor's theorem (for any set $|A| < |2^A|$), it must be the case that for some languages there is no recognizing program.

Finite state automata

Example 4

Input: $w \in \{0,1\}^*$

Finite state automata

Example 4

Input: $w \in \{0,1\}^*$

Check: is the number represented by w in binary a multiple of 3?

Finite state automata

Example 4

Input: $w \in \{0,1\}^*$

Check: is the number represented by w in binary a multiple of 3?

Idea 1: Possible remainders are $\{0,1,2\}$.

Finite state automata

Example 4

Input: $w \in \{0,1\}^*$

Check: is the number represented by w in binary a multiple of 3?

Idea 1: Possible remainders are $\{0,1,2\}$.

Idea 2: If you read a 0 at a state q then go to state $2q \pmod 3$, else go to state $2q+1 \pmod 3$

Finite state automata

Example 4

Input: $w \in \{0,1\}^*$

Check: is the number represented by w in binary a multiple of 3?