Introduction to TOC

Hemant Kumar

July 31, 2020

Outline

Computation

Computation

Any task that can be performed by a calculator or computer.

In theory of compuatation, we are going to mathematically model of a computer or any machine in general & then going to study the theory about it, which means

- What are the capabilities of this machines?
- What are the problem could be solved by this machines?
- what are the limitations of such machines?

Sets

Definition: Set

A set is a collection of elements, without any structure other than membership.
The symbol $\in$ is used to represent membership.

Example:

- $S = \{1, 2, 3\}$
- $S = \{a, b, c, \ldots, z\}$
- $S = \{1, 3, 5, \ldots\}$ OR $S = \{i : i \geq 1, \text{i is odd}\}$
- $S = \{2, 4, 6, \ldots\}$ OR $S = \{i : i \geq 0, \text{i is even}\}$

Basic Terminolgy in set theory

Null Set

A set which contains no elements is called as empty set or null set.
For example, the set of months with 32 days.
It is represented by the symbol $\{\}$ or Φ.

Universal Set

A set which contains all the elements in the domainon which is given set defined, is called as universal set.
For example, the set of natural numbers.
$U = \{1, 2, 3, \ldots, \infty\}$

Cardinality of a Set

Cardinality of a given set id the number of element in the set.
For example, if set $A = \{1, 2, 3, 4\}$ then its cardinality is 4.
It is represented as $|A| = 4$.

Basic Terminolgy in set theory

Subset & proper subset

A set A which is said to be the subset of a set B if every element of set A is also an element of set B. This relationship is usually denoted by $A \subset B$, and mathematically this relationship is written as if $x \in A$ implies $x \in B$. The concept of a subset is also written in the from of $A \subseteq B$.

If A is a subset of B ($A \subseteq B$), but A is not equal to B, then we say A is a *proper subset* of B, written as $A \subset B$ *or* $A \subsetneq B$.
Example:
$A = \{1, 3, 5\}, B = \{1, 2, 3, 4, 5\}, C = \{1, 2, 3, 4, 5\}$
A is a subset of B, $A \subseteq B$. because every element in A is also in B.
A is also proper subset of B, $A \subset B$. because every element in A is also in B and $A \neq B$.
C is subset of B, $C \subseteq B$. but is not a proper subset of B because C = B.

Basic Terminolgy in set theory

Equivalent and Equal Sets

Two sets are said to be equivalent if they contain same number of elements. Two sets are said to be equal when they contain exactly the same elemets.

Example:
$A = \{1, 2, 3\}, B = \{4, 5, 6\}, C = \{1, 2, 3\}$
A is equivalent to B, and A is equal to C

Finite and Infinite Set

Finite sets are sets that have finite number of elements, otherwise infinite sets.
For example, If A is the set of positive integers less than 12 then
$A = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}$ and n(A) = 11 (Number of elements) then A is finite set.
if C is the set of numbers which are also multiples of 3 then
$C = \{3, 6, 9, \dots\}$ and C is an infinite set

Set Operations

Set Operations			
Operation	Symbol	Definition	Example
Union	$\cup$	$A \cup B = \{x : x \in A \text{ or } x \in B\}$, that is, Elements in either A or B	$\{a, b, c\} \cup \{c, d\} = \{a, b, c, d\}$
Intersection	$\cap$	$A \cap B = \{x : x \in A \text{ and } x \in B\}$, that is, Elements in both A and B	$\{a, b, c\} \cup \{c, d\} = \{c\}$
Difference	$-$	$A - B = \{x : x \in A \text{ and } x \notin B\}$, that is, Elements in A and not in B	$\{a, b, c\} - \{c, d\} = \{a, b\}$
Complements	$\complement$	$A^{\complement} = U - A = \{x : x \in U \text{ and } x \notin A\}$ that is, Elements in U and not in A	
Symmetric Difference	$\oplus$	$A \oplus B = (A - B) \cup (B - A) = (A \cup B) - (A \cap B)$, that is, Elements in either A or B but not in Both	$\{a, b, c\} \oplus \{c, d, e\} = \{a, b, d, e\}$
Cartesian Products	X	$A \ X \ B = \{\{a, b\} : a \in A \text{ and } b \in B\}$, that is, All possible ordered pairs whose first component is member of A and second component is member of B.	$\{a, b\} \ X \ \{0, 1, 2\} = \{(a, 0), (a, 1), (a, 2), (b, 0), (b, 1), (b, 2)\}$

Basic Concepts

The three basic fundamental ideas in study of theory of copuatation is

1. Formal Languages
2. Grammars
3. Automata or Model

Formal Language

For the study of formal languages, needto know the some terminology

Symbol

It is the basic building block, i.e.,

$$a, b, 0, 1, \ \textit{or picture}$$

Alphabet (Σ)

It is a finite collection of symbols, and denoted by Σ
$\Sigma = \{a, b\}$, here we ahve two symbols in alphabet
$\Sigma = \{0, 1, 2\}$

String

String is a finite sequence of symbols over the given alphabet, denoted by ω
$\Sigma = \{a, b\}$, then various string that can be formed { a, b, aa, bb, ab, ba, aaa, aba, abb, . . . }

Length of String

$|\omega|=$ Number of symbols in a string
$\omega = aaabba$ then $|\omega| = 6$

Number of String of length n

We have n symbols in alphabet Σ then
Number of strings of length n is possible over $\Sigma = |\Sigma|^n$

Formal Language

Formal Language

In TOC, the language is a collection of appropriate strings over the given input alphabet.

- if Σ is alphabet, then Σ^* is called universal language.
- if L is any language defined over the alphabet Σ, then $L \subseteq \Sigma^*$

Empty Language and Non-empty Language

Language does not contain any string or empty string(ϵ or null string) called empty language.

$$L = \phi = \{\} \Leftrightarrow |L| = 0$$

Otherwise language contains some string is called non-empty string

Non-empty language is two type

1. Finite Language

Language contains finite number of string where the length of each and every string is finite, called finite language. Empty language is also finite language.

L = { 01, 10 }

2. Infinite Language

Language contains infinite number of strings where the length of each and every string is finite, called infinite language.

$L = \{0^n : n \geq 1\}$

$L = \{0, 00, 000, 0000, 00000, 000000, \ldots\}$

Infinite language are two type

1. Countable
2. Uncountable

Grammar

in TOC,

Grammar

A grammar G is defined as a quadruple

$$G = (V, T, P, S)$$

where, V is finite set of objects called variables,
T is finite set of objects called terminal symbols,
P is a finite set of production rules,
S $\in$ V is start symbol.

Automata or Model

in TOC,

Automata

An Automata is an abstract model of a digital computer.It has a mechanism for reading input, and can produce output and finally automaton has control unit which can have finite number of internal states.

An automata model can have

1. Language Accepter (Accepting the string by a model)
2. Language Generator (generating the string)

Formal Languages

Type	Formal Language	Grammar	Automata
Type 3	Regular Language	Regular Grammar	Finite Automata
Type 2	Context-free Language	Context-free Grammar	Pushdown Automata
Type 1	Context Sensitive Language	Context Sensitive Grammar	Linear Bounded Automata
Type 0	Recursive Enumerable Language	Unrestricted Grammar	Turing Machine

Deterministic Finite Automata

The subtitle

Hemant Kumar

September 2, 2020

Outline

Finite Automata

Automata

Automata is pictorial representations of machines. It is self operting machines which act according to a set of predefined actions and do a specific task or accept strings of a specific language.

Finite Automata

The FA contains five tuples in a

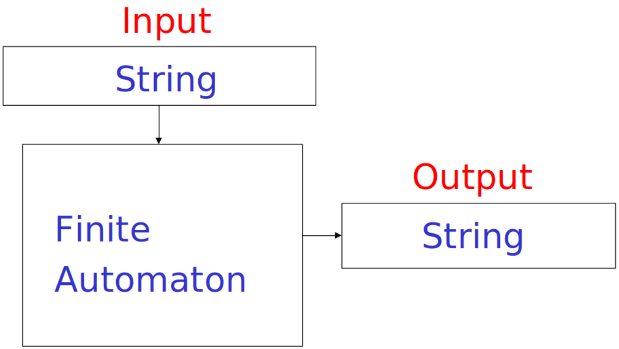
$$M = (Q, \Sigma, \delta, q_0, F)$$

where,
Q is finite set of states
Σ is input alphabet
q_0 is start state $q_0 \in Q$
F is set of final states $Q \supseteq F$ (Q is superset of F)
δ is transition function

Finite Automata

Finite Automata Model

Finite automata can be represented by input tape and finite control.
Input tape: It is a linear tape having some number of cells. Each input symbol is placed in each cell.



Finite Accepter

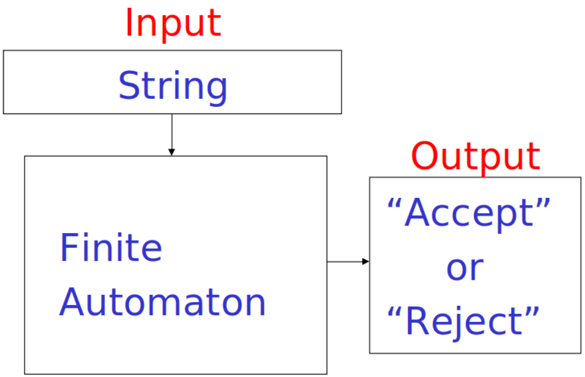


Figure: Finite Accepters

Representation of Finite Automata

Transition Diagram

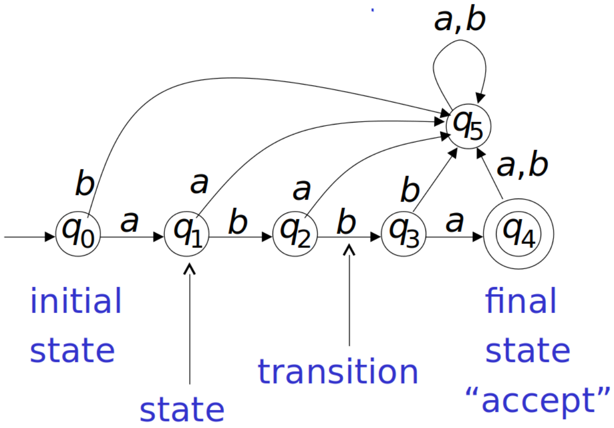


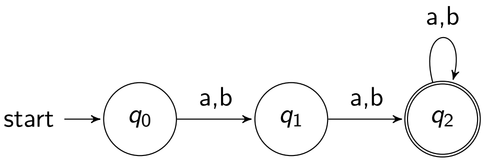
Figure: Transition Diagram

Representation of Finite Automata

Transition Table

The transition table is basically a tabular representation of the transition function. It takes two arguments (a state and a symbol) and returns a state (the "next state").

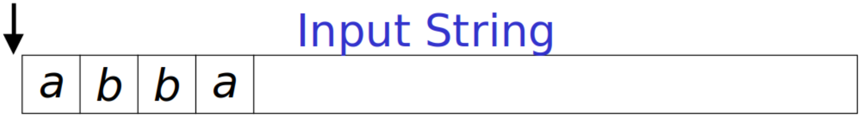
Transition Diagram

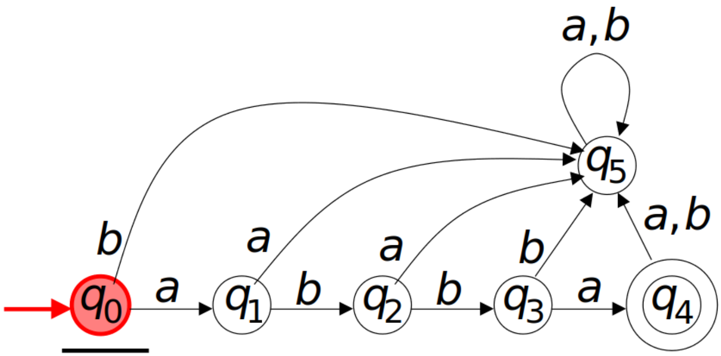


Transition Table

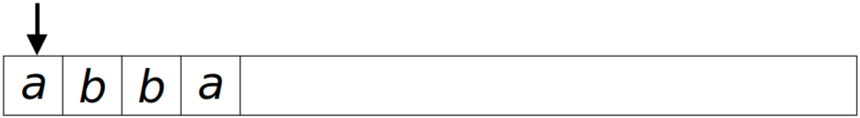
	input	
state	a	b
$\rightarrow q_0$	q_1	q_1
q_1	q_2	q_2
$*q_2$	q_2	q_2

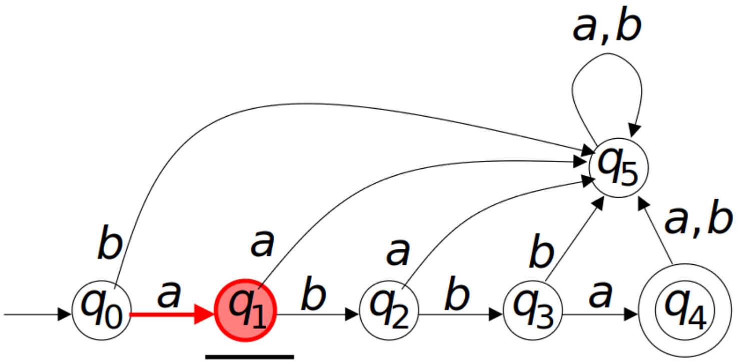
Initial Configuration

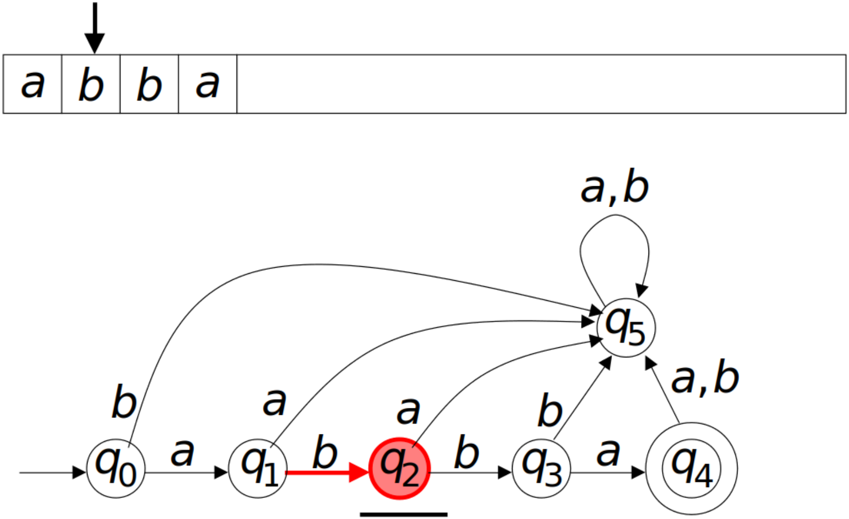


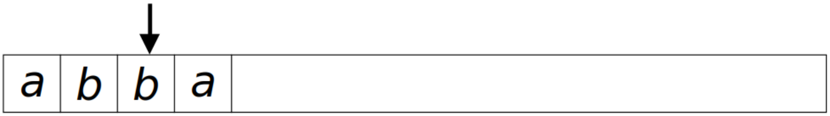


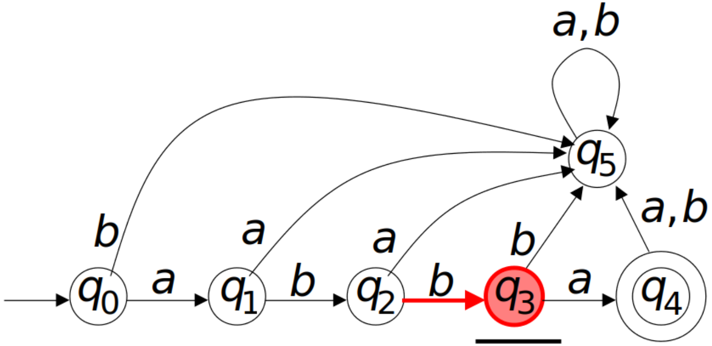
Reading the Input

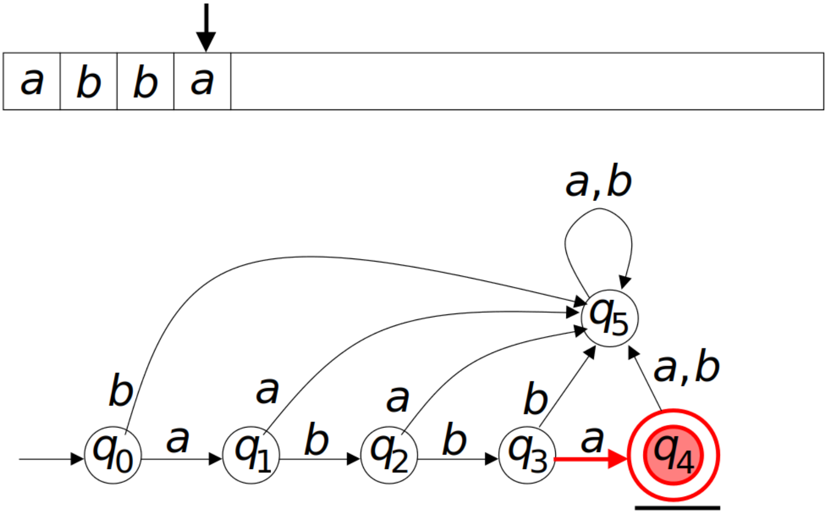


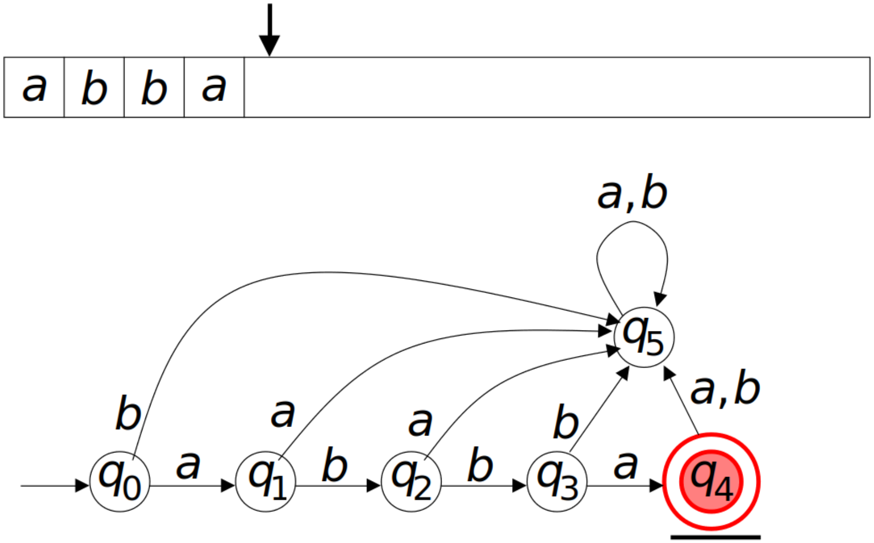



a
b
b
a
a,b
b
a
a
b
a
b
b
a
a,b
q_0
q_1
q_2
q_3
q_4
q_5


a
b
b
a

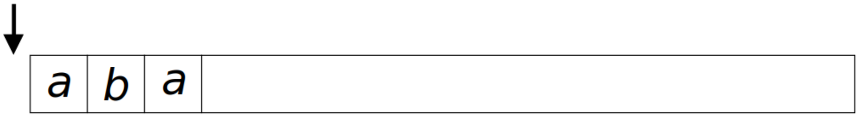

a,b
b
a
a
b
a,b
q0
a
q1
b
q2
b
q3
a
q4
q5

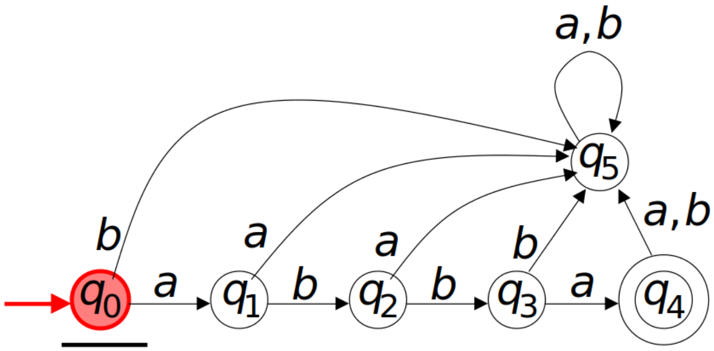

a
b
b
a
a,b
q5
a,b
b
a
a
b
q0
a
q1
b
q2
b
q3
a
q4

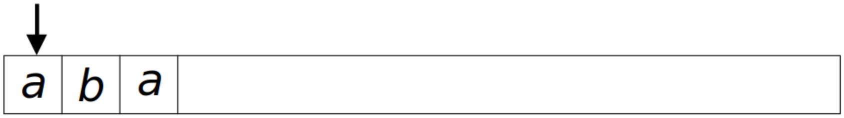


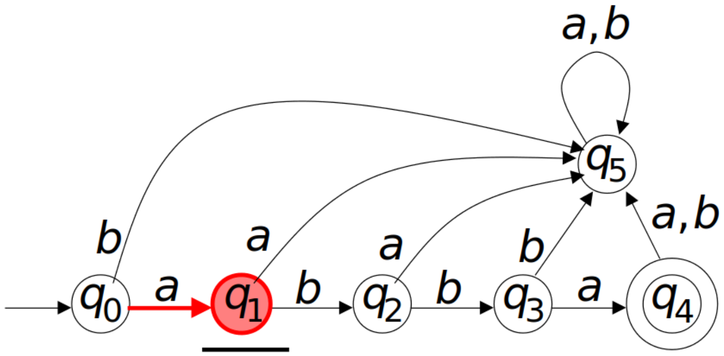
Output: “accept”

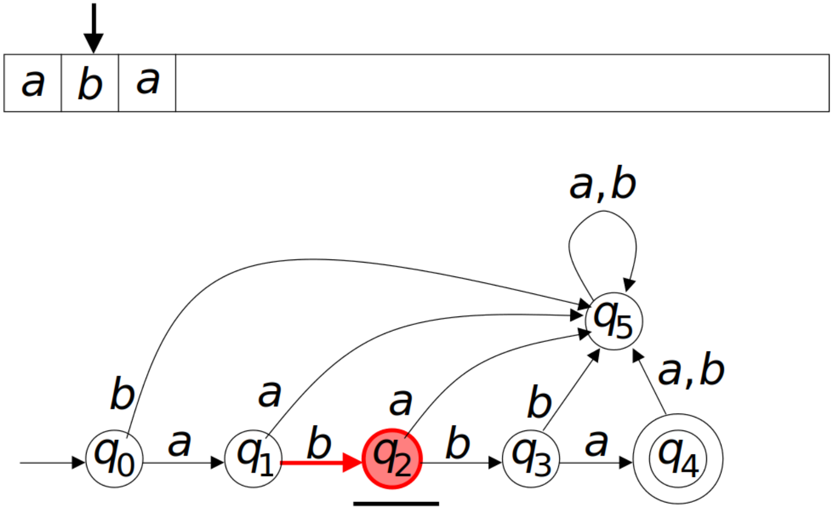
Rejection

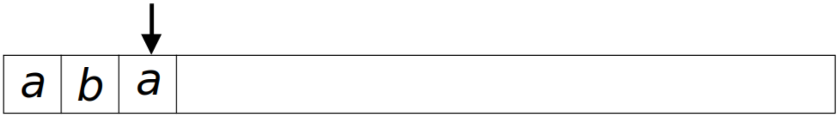


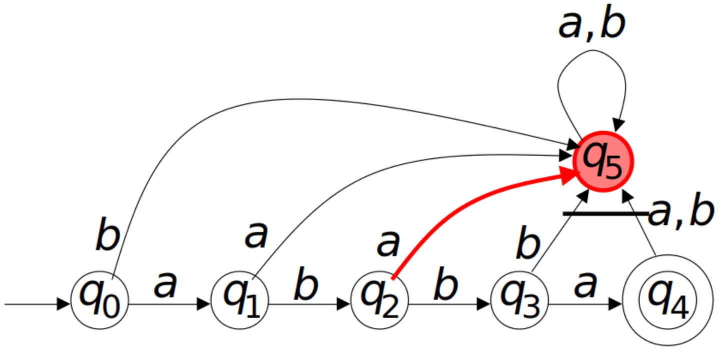


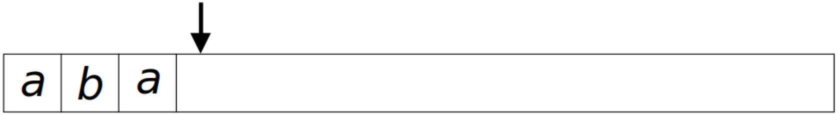

a
b
a

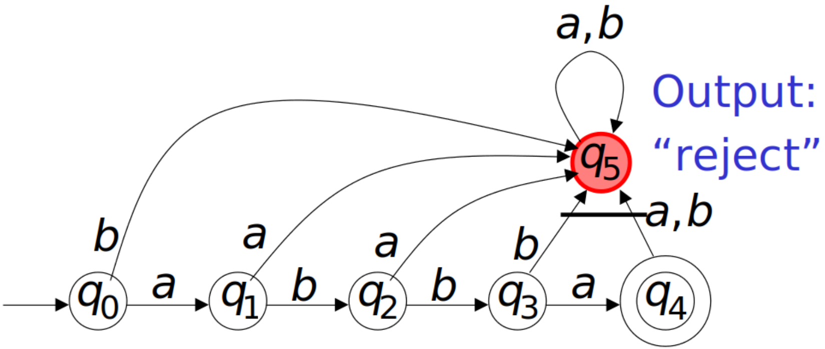

a,b
b
a
a
b
a,b
q0
a
q1
b
q2
b
q3
a
q4
q5


a
b
a
a,b
b
a
a
b
a,b
q_0
a
q_1
b
q_2
b
q_3
a
q_4
q_5


a
b
a


a,b
b
a
a
a,b
b
q0
a
q1
b
q2
b
q3
a
q4
q5


a
b
a


a,b
Output:
“reject”
a,b
b
a
a
b
q0
q1
q2
q3
q4
q5
a
b
b
a

Types of Finite Automata

Finite Automata is two type :

1. Finite Automata without output (Language Recognizers or Compilers)
    1. Deterministic Finite Automata (DFA)
    2. Non-deterministic Finite Automata(NFA)
2. Finite Automata with output (Digital Circuit Generators)
    1. Moore Machine
    2. Mealy Machine

Deterministic Finite Automata(DFA)

Definition: DFA

The DFA contains five tuples in a

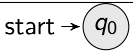
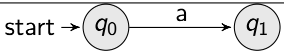
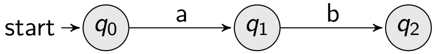
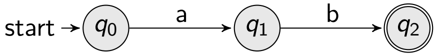
$$M = (Q, \Sigma, \delta, q_0, F)$$

where,
Q is finite Set of states
Σ is input alphabet
q_0 is start state $q_0 \in Q$
F is set of final states $Q \supseteq F$ (Q is superset of F)
δ is transition function $\delta : Q \times \Sigma \to Q$

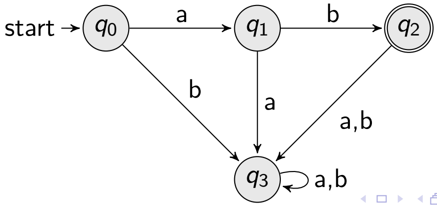
Properties :

- In DFA, every state for every input alphabet value, that is read, there is one and only one state.
- Missing trsnsition are not allowed in DFA. If there is no transition on a perticular input symbol from the state then introduced a new state (trap or dead state) which is sink the missing transition.

Draw the dfa tha recognizes the string "ab" on $\Sigma = \{a, b\}$

First we will make start state	
Now first symbol of string is a	
second symbol of string is b	
Since string is completed , after reading b, dfa is in state q_2, hence q_2 must be the final state	

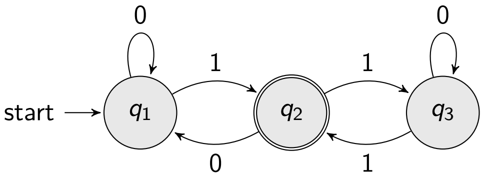
We know that missing transitions are not allowd in dfa so we add a new trap state which is sink all the missing transitions $\delta(q_0,\ b)$, $\delta(q_1,\ a)$ and $\delta(q_2,\ a)$, $\delta(q_2,\ b)$ are missing. so final dfa is



Language of DFA

String Acceptence

Scan the entire string, and reached the final state from start state.



Example: which of the following strings 0001, 01001, 0000110, are accepted bythe give dfa?

Solution: (a). string $\omega = 0001$

$\delta(q_1, 0001) \Rightarrow \delta(0q_1, 001) \Rightarrow \delta(00q_1, 01) \Rightarrow \delta(000q_1, 1) \Rightarrow \delta(0001q_2, \epsilon)$

(b). string $\omega = 01001$

$\delta(q_1, 01001) \Rightarrow \delta(0q_1, 1001) \Rightarrow \delta(01q_2, 001) \Rightarrow \delta(010q_1, 01) \Rightarrow$
$\delta(0100q_1, 1) \Rightarrow \delta(0100q_2, \epsilon)$

Language of DFA

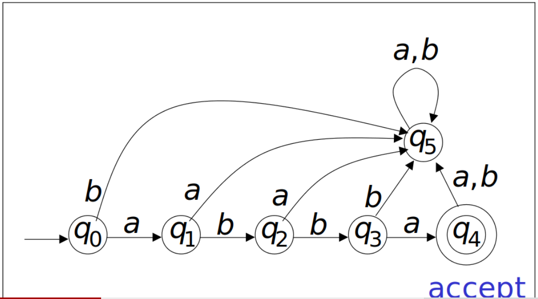
The language accepted by DFA $M = (Q, \Sigma, \delta, q_0, F)$ is the set of all strings on Σ accepted by M.

$$L(M) = \{\omega \in \Sigma^* \ : \ \delta^*(q_0, \omega) \in F\}$$

A DFA is said to be accept a language if all the string in the language are accepted & all/some of the string not in the language are rejected .

$L(M) = \{abba\}$

M



Deterministic Finite Automata

Questions

Hemant Kumar

July 21, 2020

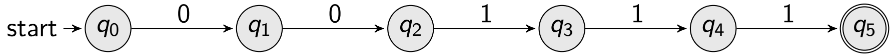
Outline

1. Construction of Minimal DFA
    1. Based of full length string(contains both a and b)
    2. Based on the number of a's or number of b's in a string
    3. Based on the remainder on length of string $|\omega| \mod n = k$
    4. Based on the no. of a's or b's in a string which is divisible by N (O
    5. Starts with some symbol(like a) or string(ab)
    6. String contains some symbol(like a) or string(ab)
    7. Ends with some symbol(like a) or string(ab)
    8. Starts with some symbol(like a) or string(ab)

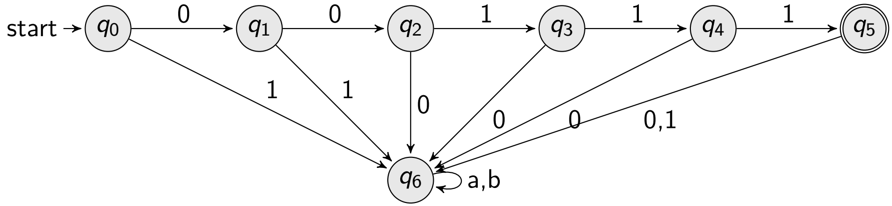
Example on DFA

Draw the dfa that recgnizes string '00111' on $\Sigma = \{0, 1\}$.

first draw the generalized dfa



Now complete missing transactions



1. Based of full length string(contains both a and b)

1. Minimal DFA of all strings of length atleast n over the $\Sigma = \{a, b\}$ OR $(|\omega| \geq n)$
2. Minimal DFA of all strings of length atmost n over the $\Sigma = \{a, b\}$ OR $(|\omega| \leq n)$
3. Minimal DFA of all strings of length equal to n over the $\Sigma = \{a, b\}$ OR $(|\omega| = n)$

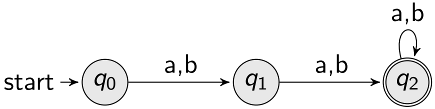
Steps for constructing the dfa

1. Construct a language using given condition.
2. Language is finite
   - take smallest length string from language set and construct DFA and enhance the string length.
3. Language is infinite
   - take largest length string from language set and construct DFA and reduce the string length.
4. If a language contain ϵ , them we make initial state as finial state.

A. Minimal DFA of all strings of length atleast n over the $\Sigma = \{a, b\}$ OR $(|\omega| \geq n)$

Minimal DFA of all strings of length atleast 2 over the $\Sigma = \{a, b\}$

Language $L = \{string\ of\ length\ 2,\ length\ 3,\ length\ 4, \ldots\}$
$L = \{aa, ab, ba, bb, aaa, aba, aab, abb, \ldots\}$ so that language is infinite so use all smallest length string for DFA construction



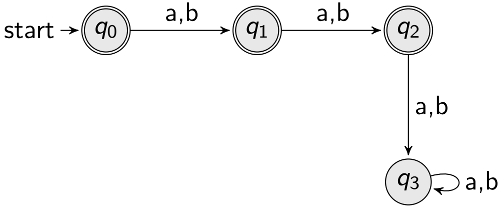
Number of state

Number of state = (length of string +1) = (n+1)

B. Minimal DFA of all strings of length atmost n over the $\Sigma = \{a, b\}$ OR $(|\omega| \le n)$

Minimal DFA of all strings of length atmost 2 over the $\Sigma = \{a, b\}$

Language $L = \{string\ of\ length\ 0,\ length\ 1,\ length\ 2\}$
$L = \{\epsilon, a, b, aa, ab, ba, bb\}$ so that language is finite so use all largest length string for DFA construction. If a language contain ϵ then we make initial state as final state.
In that case, we introduced dead state or trap state q_3



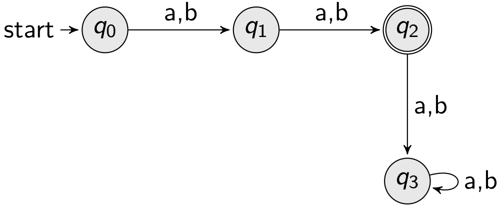
Number of state

Number of state = (length of string +2) = (n+2)

C. Minimal DFA of all strings of length is equal to n over the $\Sigma = \{a, b\}$ OR $(|\omega| = n)$

Minimal DFA of all strings of length is equal to 2 over the $\Sigma = \{a, b\}$

Language $L = \{string\ of\ length\ 2\}$
$L = \{aa, ab, ba, bb\}$ so that language is finite so use all largest length string for DFA construction.
In that case, we introduced dead state or trap state q_3


$$\begin{tikzcd}
\text{start} \arrow[r] & q_0 \arrow[r, "{a,b}"] & q_1 \arrow[r, "{a,b}"] & (q_2) \arrow[d, "{a,b}"] \\
 & & & q_3 \arrow[loop right, "{a,b}"]
\end{tikzcd}$$

Number of state

Number of state = (length of string +2) = (n+2)

2. Based on the number of a's or number of b's in a string

1. Minimal DFA of all strings contains atleast n number of a's and any number of b over the $\Sigma = \{a, b\}$ OR $(n_a(\omega) \geq n)$.
2. Minimal DFA of all strings contains atmost n number of a's and any number of b over the $\Sigma = \{a, b\}$ OR $(n_a(\omega) \leq n)$.
3. Minimal DFA of all strings contains exactly n number of a's and any number of b over the $\Sigma = \{a, b\}$ OR $(n_a(\omega) = n)$.

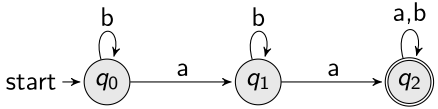
Steps for constructing the dfa

1. Construct a language using given condition.
2. Language is finite
    - take smallest length string from language set and construct DFA and enhance the string length.
3. Language is infinite
    - take largest length string from language set and construct DFA and reduce the string length.
4. If a language contain ϵ, then we make initial state as final state

A. Minimal DFA of all strings contains atleast n number of a's and any number of b over the $\Sigma = \{a, b\}$ OR $(n_a(\omega) \geq n)$

Minimal DFA of all strings contains atleast two a's and any number of b over the $\Sigma = \{a, b\}$ OR $(n_a(\omega) \geq 2)$

Language
$L = \{$string which conatins atleast two a's and any number of b's $\}$
$L = \{aa, aab, aba, baa, aabb, abab, abba, bbaba, \ldots\}$ so that language is infinite so use all smallest length string for DFA construction



$$\begin{tikzcd} \text{start} \arrow[r] & q_0 \arrow[loop above, "b"] \arrow[r, "a"] & q_1 \arrow[loop above, "b"] \arrow[r, "a"] & q_2 \arrow[loop above, "a,b"] \end{tikzcd}$$

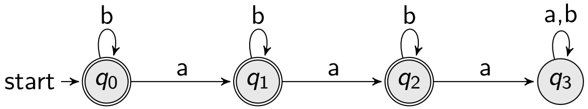
Number of state

Number of state = (n+1)

B. Minimal DFA of all strings contains atmost n number of a's and any number of b over the $\Sigma = \{a, b\}$ OR $(n_a(\omega) \leq n)$

Minimal DFA of all strings contains atmost two a's and any number of b over the $\Sigma = \{a, b\}$ OR $(n_a(\omega) \leq 2)$

Language $L = \{\epsilon, a, aa, bababbbb, aabbbbb, bbbbb, \ldots\}$ so that language is finite so use all largest length string for DFA construction. If a language contain ϵ then we make initial state as final state.
In that case, we introduced dead state or trap state q_3


$$\begin{tikzcd}
\text{start} \arrow[r] & q_0 \arrow[loop above, "b"] \arrow[r, "a"] & q_1 \arrow[loop above, "b"] \arrow[r, "a"] & q_2 \arrow[loop above, "b"] \arrow[r, "a"] & q_3 \arrow[loop above, "a,b"]
\end{tikzcd}$$

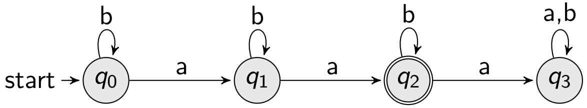
Number of state

Number of state = (length of string +2) = (n+2)

C. Minimal DFA of all strings contains exactly equal to n number of a's and any number of b over the $\Sigma = \{a, b\}$ OR $(n_a(\omega) = n)$

Minimal DFA of all strings contains exactly to two a's over the $\Sigma = \{a, b\}$ OR $(n_a(\omega) = 2)$

Language $L = \{\text{exactly 2 a's and any number of b's}\}$
$L = \{aa, \}$ so that language is finite so use all largest length string for DFA construction.
In that case, we introduced dead state or trap state q_3


$$\begin{tikzcd}
\text{start} \arrow[r] & q_0 \arrow[loop above, "b"] \arrow[r, "a"] & q_1 \arrow[loop above, "b"] \arrow[r, "a"] & \boxed{q_2} \arrow[loop above, "b"] \arrow[r, "a"] & q_3 \arrow[loop above, "a,b"]
\end{tikzcd}$$

Number of state

Number of state = (length of string +2) = (n+2)

3. Based on the remainder on length of string OR ($|\omega|$ *mod* $n = k$)

$|\omega|$ *mod* $n = k$ where k is denotes remainder, i.e., (0,1,2,3,.......)

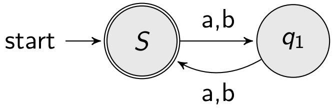
1. number of state is n because (length is divisible by n)
2. if remainder is 0 then final state is also initial state; if remainder is 1 then final state is next state of initial state ; and same for remainder 2, 3 and so on.

Construct a minimal DFA of all even length string $|\omega|$ mod 2 = 0

The String length is divisible by 2, we get two remainder either 0 or 1 then we having 2 state. In that case remainder is 0 so the language contains ε, and initial state is also final state.

$L = \{length\ 0, 2, 4, \dots\}$

$L = \{\varepsilon, aa, bb, ab, ba, aaaa, bbbb, \dots\}$

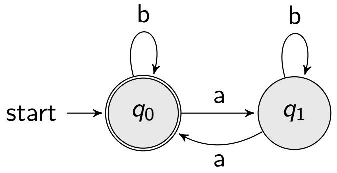


4. Based on the no. of a's or b's in a string which is divisible by N(OR Remainder) OR $(n_a(\omega)\ mod\ n = k)$)

$(n_a(\omega) \cong K \quad mod \quad N)$
In that case K is remainder and N is divisible number, so the number of state is N. And final state based on remainder.

Construct a minimal DFA for $(n_a(\omega)\ mod\ 2 = 0)$ or $(n_a(\omega) \cong 0\ mod\ 2)$

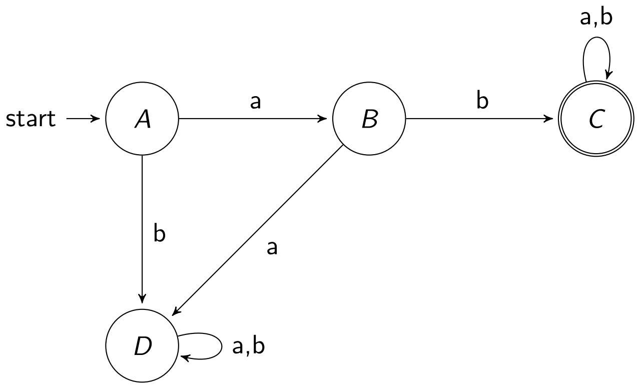
Number of a's is divisible by 2 or number of a's is multiple of 2 or even number of a.
In that case remainder is 0 so initial state is also final state and number of state is 2 because divisible by 2.



5. Starts with some symbol(like a) or string(ab)

Construct a dfa which accepts all strings starting from ab on $\Sigma = \{a, b\}$

$L = \{ab, aba, abb, aab, abaa, \ldots\}$ which is infinite language

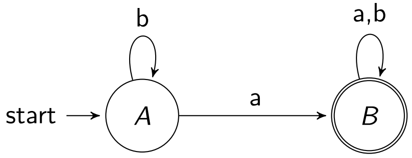


D is Dead State

6. String contains some symbol(like a) or string(ab)

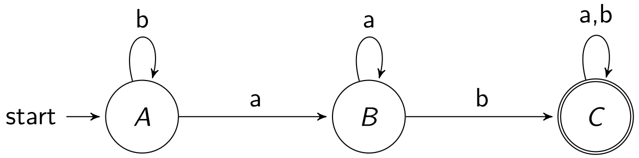
Ex1. Construct a dfa which accepts sting contains 'a'.

$L = \{a, aa, ab, ba, bba, aab, aba, \ldots\}$ which is infinite language



Ex2. Construct a dfa which accepts sting that contains 'ab' as a substring..

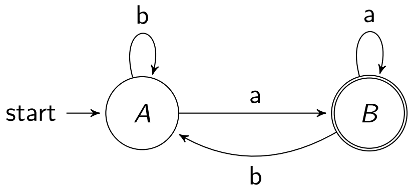
$L = \{ab, aba, abb, aab, abaa, bab \ldots\}$ which is infinite language



7. Ends with some symbol(like a) or string(ab)

Ex1. Construct a dfa which accepts sting that ends with 'a'.

$L = \{a, aa, ba, bba, aba, \ldots\}$ which is infinite language



Ex2. Construct a dfa which accepts sting that ends with 'ab'.

$L = \{ab, aab, abab, baab, abaab, \ldots\}$ which is infinite language

Do Your self

Deterministic Finite Automata

Operations

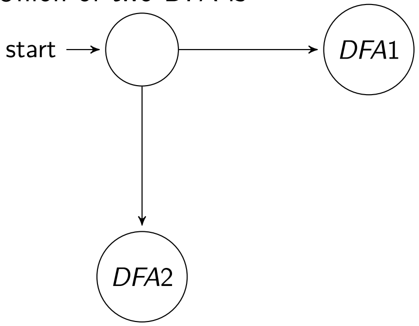
Hemant Kumar

July 22, 2020

Outline

Union

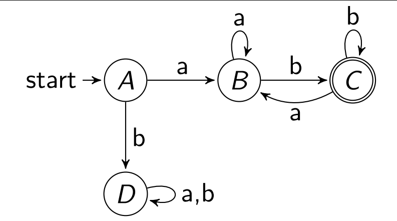
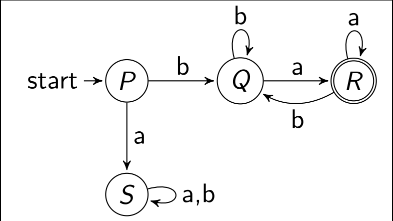
Let two languages are L_1 and L_2 and its corresponding dfa are DFA1 and DFA2 respectively.
Union of two languages are $(L_1 \cup L_2)$
Union of two DFA is

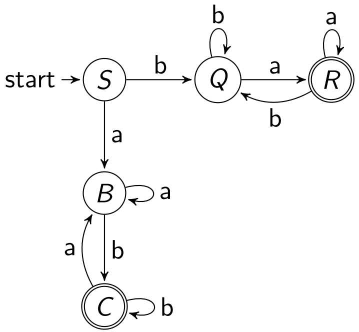


In that case, join both DFA's at starting state, i.e., Common starting point

Construct a minimal dfa which start and ends with different symbol.

Let input $\Sigma = \{a, b\}$.
problem in two parts first string start with a and end with b or start with b and end with a. so the languages

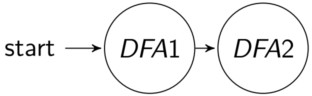
$L_1 = \{ab, aab, abb, abbb, \ldots\}$	$L_2 = \{ba, baa, bba, baaba, \ldots\}$
	



Concatenation

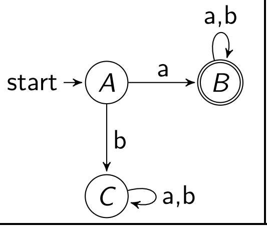
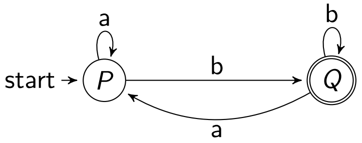
In concatenation of two DFA, we put first dfa, and final state of first dfa is the initial state of second dfa and second dfa have final state i.e.
Let two languages are L_1 and L_2 and its corresponding dfa are DFA1 and DFA2 respectively.

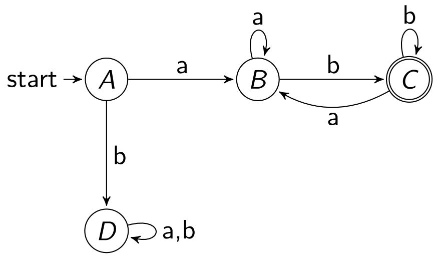
$$L_1 \cdot L_2 = DFA1 \cdot DFA2$$



Construct a of minimal DFA which start with a and end with b.

Let input $\Sigma = \{a, b\}$.
problem in two parts first string start with a and end with b. so the languages

$L_1 = \{a, aa, ab, aaa, aba \ldots\}$	$L_2 = \{b, ab, bb, bab, aab, \ldots\}$
	



Cross-Product

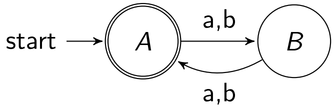
Complement

It is only for DFA. First we construct a DFA for that problem and next for complement of DFA, First we make non-final state to final state and final state to non-final state.

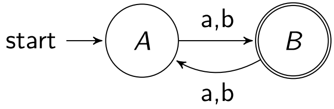
Construct a complement of minimal DFA which accepts set of all strings over Σ of even length.

First Construct DFA

$L_1 = \{aa, ba, bb, ab, aaaa, abab, abba, bbbb, \ldots\}$ = even length string



Next for complement, we just make final state to non-final state and non-final state to final state.



$L_2 = \{a, b, aaa, bbb, aba \ldots\}$ = odd length string

Complement of DFA

Definition

$$M = (Q,\ \Sigma,\ \delta,\ q_0,\ Q - F)$$

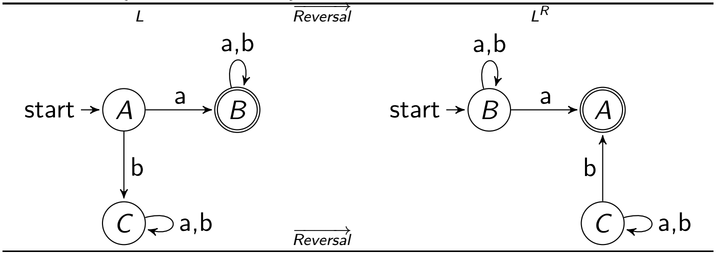
where,
Q is finite Set of states
Σ is input alphabet
q_0 is start state $q_0 \in Q$
$Q - F$ is set of final states and F is final state in given DFA
δ is transition function $\delta : Q \times F \rightarrow Q$

Reversal

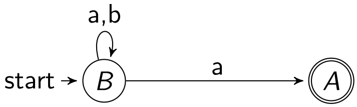
Reversal operation on DFA and output can be either DFA or NFA.
For Reversal operation , we make final state to start state and start state to final state and the direction of arrow is revert.
Dead state in DFA convert to unreachable or useless state and remove that state.

Reversal of a of minimal DFA which start with a.

languages $L_1 = \{a, aa, ab, aaa, aba \ldots\}$



Here state C is unreachable or useless state so remove it, and final automata is



This is NFA not DFA.