CS402-Theory Of Autometa
FINAL TERM SUBJECTIVE
Prepared by: JUNAID MALIK

AL-JUNAID INSTITUTE GROUP

Q No 1.How Moore and Mealy machine works in Computer Memory what is their importance in Computing ?

Mealy & Moore Machines work in computing as incrementing machine & 1's complement machine etc. These operations as basic computer operations so these machines are very important.

Q No 2.What is sequential circuit ?

Sequential Circuit:

A sequential circuit contains a memory component.

The memory component provides a state input. A flip-flop is often used as a memory component.

The state variable indicates the states of the sequential machine, i.e. the status or stage or progress of the whole event.

The state of a sequential circuit is indicated by the output of a flip-flop. A single flip-flop can be used to indicate two states (q=0 and q=1). When there are more than two states, additional flip-flops are used. Given n flip-flops, a total of 2^n states can be represented.

In other words, a sequential machine can be put into a number of different states depending on the particular inputs given.

The output is a function of both the Present Inputs and the Present States.

In addition to the outputs, the circuit must also generate an update to the memory components so that the state of the machine can also be changed with respect to the new inputs.

The update is called the Next State Function and is also a function of the Present Inputs and the Present States.

Both the output functions and the Next State Functions are combinational circuits. $Z=f(X,S^t)$

$S=g(X,S^t)$ The superscript t indicates the present time period while the superscript (t+1) indicates the next time period.
The characteristic of a sequential circuit is completely defined by a state transition diagram that enumerates all possible transitions for every possible input combination.

Q No 1.What is the concept of Pumping Lemma I and II and what is the difference between pumping Lemma 1 and pumping Lemma 2 ?

In fact PLI & PLII are same (A way to recognize Non Regular language). The only difference is that the conditions in pumping lemma II are more stricter than Pumping Lemma I some language that are difficult to proof Non Regular by Pumping Lemma I are proved Non Regular by pumping Lemma II easily.
Further mare in pumping lemma I we have to generate all words to of a language but in Pumping Lemma II we have to generate a single word to prove a language non regular.
Explanation:
Some languages like PALINDROME that are proved to be regular by first version
due to some of their symmetrical words when we pump these words they remain to be the parts of the language like bbabb
By pumping
lemma 1 Let y
= a

Now repeating y three
times results in bbaaabb
That is also a valid word of PALINDROME so by pumping lemma I PALINDROME can not be proved non regular, so there was the need of pumping lemma version version 2.
Now consider for
the word bbabb
if we take N=2
Then by pumping y (let we take it b)
two times results in bbbbabb
That word is not in
PALINDROME. But if
we take N=3 and y = a
Then by pumping y two
times results in bbaaabb
That word is in PALINDROME. So be careful in taking total no of states of the FA and also the repeating factor (y) to prove an infinite language non regular you need to prove only one word that is not part of the language.

Q No 2. What is the significance of Pumping Lemma II ?

The significance of 2nd version of 'pumping lemma' is that there are some infinite non regular languages like PALINDROME we can built FA that can accept there certain words but if we increase the length of their words that FA don't accept these words so by pumping lemma version I it is very difficult to prove them non regular but with the second version we can prove that a language is Non regular even it's some words may be accepted by some FA's. See page 195 of the book for further example.

Q No 3.Moore and Mealy machine?

1. In order to run a string on a Mealy or Moore machine, you can take directions from transition table. Running string on Mealy or Moore machine is similar to running string on a FA. For example, if want to run abba on the machine, take start from initial state. Check what is the transition for a, what state it goes. After that check what is the path of b from that state and so on. In this way you will be able to run whole of the string. Note that there is no final state in Mealy or Moore machine. So there is no case of acceptance or rejection of string. You just have to determine what the output is. I hope that will clear your mind for further clarification please listens to your lecture carefully.
2. The string is taken for the testing purposes. You can take any sort of string and determine its output using machine.

Q No 1.What is the difference between semiword and word please also give an example regarding this?

Word:

A word is complete combinations of terminals only e.g. abba or ab or a or null string.

Semiword:

A semiword is a string of terminals (may be none) concatenated with exactly one nonterminal on the right i.e. a semi word, in general, is of the following form

(terminal)(terminal)--- (terminal)(nonterminal)

For example

aaaaaaB , aabbaaaA , A.

What is the difference between derivation tree and total tree?

A Derivation tree is the one that shows how to derive any specific word of the language described by CFG but Total Language Tree shows all words of the

Q No 2.What does mean the LANGUAGE IS CLOSED?

When we say that a Language is closed it is always with respect to certain operation.

A simple example may be that the set of integers is closed under addition. It means when we take two numbers from set of integers say 3, 7 the result of their addition would also be in the set of integers. Similarly if the result of an operation on the words of a language results in the word of the same language we say that the language is closed under that operation.

Q No 3.What are the Productions?

Productions are the grammatical rules and regulations. These rules express the behavior of CFG. Using production in CFG terminals are converted into non-terminals and when all the terminals are converted using productions, a word is acquired.

Q No 4.What is the difference between concatenation and intersection of two FA's also what is the difference among Union of two FA's and addition of them?

In intersection of two FA's only those strings are accepted which are independently accepted by both FA’s, while in concatenation of two FA’s only those strings will be accepted in which first part of string is accepted by first FA and remaining part of string is accepted by the second FA.

While taking union of two FA’s one can represent it using + sign. So (FA1 U FA2) and (FA + FA2) both are same. There is no difference between them.

FAQ's about

Lectures 36 to 40

Automata Theory

FAQ's about Lectures 36 to 40

Q No 1.What is the Difference between Nullable and Null production? How to make eliminate Nullable and for Null Productions from the CFG ?

The production of

the form

nonterminal ® L

is said to be ***null production***.

Example:

Consider the following CFG

S ® aA|bB|L, A ® aa|L, B ® aS

Here S ® L and A ® L are null productions.

A production is called ***nullable production*** there is a derivation that starts at Non Terminal and leads to L

i.e.

S ---- > *aA* | *bB* | *aa*

A -----> *C* | *bb*

C -----> L

Here A nullable Non Terminal due to Nullable production A > C as C leads to null.

Example:

Consider the
following CFG S
® XY, X ® Zb, Y
® bW
Z ® AB, W ® Z,
A ® aA|bA|L B
®Ba|Bb|L.
Here A ® L and B ® L are null productions, while Z ®
AB, W ® Z are nullable productions.

Method:

Delete all the Null productions and add new productions *e.g.*
Consider the following productions of a certain
CFG X ® aNbNa, N ® L, delete the production N
® L and using the production
X ® aNbNa, add the following
new productions X ® aNba, X ®
abNa and X ® aba
Thus the new CFG will contain the following
productions X ® Nba|abNa|aba|aNbNa

Note: It is to be noted that X ® aNbNa will still be included in the new CFG.

Method:

Consider the
following CFG S
® XY, X ® Zb, Y
® bW
Z ® AB, W ® Z, A
® aA|bA|L B
®Ba|Bb|L.

Here A ® L and B ® L are null productions, while Z ® AB, W ® Z are nullable productions. The new CFG after, applying the method, will be

S ® XY

X ®

Zb|b Y

®

bW|b

Z ®

AB|A|B

W ® Z

A ®

aA|a|bA|b

B

®Ba|a|Bb

|b

Note: While adding new productions all Nullable productions should be handled with care. All Nullable productions will be used to add new productions, but only the Null production will be deleted

Q No 2. Is it possible to make CFG for infix and postfix expression's using derivation tree ?

Derivation tree is only used to derive words of language that is described by a CFG. Yes, we can create CFG for languages infix expressions, postfix expressions.

Q No 3 what is the uses of push down automata in computing ?

PDA is just an enhancement in FAs. i.e Memory is attached with machine that recognizes some language. FA is basic structure for most advanced electronic machines such as computer etc.

Q No 4 What is difference between PUSH DOWN STACK and PUSH DOWN STORE ?

No difference at all. Both terms are used to describe memory structure attached with FAs to store some characters in it.

Q No 5 How we can distinguish between "CFG" and "CNF" in the questions ?

Chomsky Normal Form (CNF)

If a CFG has only productions of the form

string of two nonterminals□nonterminal ---------

or

one terminal□Nonterminal -----------

Then the CFG is said to be in Chomsky Normal Form (CNF). Thus if the given CFG is in the form specified above it will be called in CNF.

Q No 6.What is meant by the terms stack consistence and input tape consistence ?

Term Stack consistent means we can pop any character from the top of the stack only. PDA should not be able to pop any character other than that is present on the top of the stack.

Term Tape consistent means we can read only the first letter on the tape not any other letter of the tape after the first one.

Q No 7 What is the concept of unit production ?

The productions of the form

one Nonterminal□one Nonterminal --------

Are called unit productions. For example

A (Unit Produciton)□S --------

a | b□A--------

A. we can directly write□Here there is no need of Unit Production S S □ a | b

Q No 8 Why Context Free Grammars are called "Context Free? Context Free Grammars are called context free because the words of the languages of Context Free Grammars have words like "aaabbb"(PALINDROME). In these words the value of letters (a , b) is the same on whatever position they appear. On the other hand in context sensitive grammars their value depend on the position they appear in the word a simple example may be as follows

Suppose we have a decimal number 141 in our language . When compiler reads it, it would be in the form of string. The compiler would calculate its decimal equivalent so that we can perform mathematical functions on it. In calculating its decimal value , weight of first "1" is different than the second "1" it means it is context sensitive (depends on in which position the "1" has appeared).

i.e.

$1*10^2 + 4*10^1 + 1*10^0 = 14$

(value of one is 100) (value of one is just one)

That is not the case with the words of Context Free Languages. (The value of "a" is always same in whatever position "a" appears).

Q No 9. What is Unit Production?

The production in which one non-terminal leads to only one non-terminal.

Q No 10.What is Left most Derivation in CFG?

It is a method of generation of strings from a CFG starting from left most letter of the string.

Q No 1.Give a example of converting a CFG to CNF?

Consider the CFG
given below S}
ABC
A} aa
| bB}
c
C}d

Its CNF

will be

S} DC

D } AB
A}
EE | b
E}a
B}c
C}d

Q No 2.In the lecture 41 's example, we have converted PDA to conversion form and a word 'aaaabb' is derived from this conversion form PDA. What are the derivation steps.

The PDA converted to conversion form has some specific features that are important to understand first. These features are

The states named START, READ, HERE and ACCEPT are called joints of the machine.

With the help of the conversion form we have been able to achieve that POP state has only one path out of it and the path taking (multiple paths) decisions take place only on the READ state.

The word 'aaaabb' is generated as follows from the PDA START-

POP4-PUSH $

This step pops $ and then pushes it to ensure that stack contains $ at the beginning.

READ1-POP6-PUSH $-PUSH a

As first time after reading "a" there is $ at the top

of stack so we will follow path segment READ1-
POP6-PUSH $-PUSH a
READ1-POP5-PUSH a-PUSH a
Now a is on the top of the stack so we will follow
READ1-POP5-PUSH a-PUSH a
READ1-POP5-PUSH a-PUSH a
Again following same
segment for a READ1-
POP5-PUSH a-PUSH

a
Again following same segment for a
READ1-POP1- HERE-POP2
As we read b on
input tape.
READ2-POP1-
HERE-POP2
As we read b on
input tape.
READ2-POP3-
ACCEPT.
As we read Δ from the input tape

Q No 3.How to differentiate between "wanted" and "unwanted branch" ? When we derive a word in Top down parsing beginning with the starting Non Terminal the branches of the tree that do not lead to our required word are left aside these branches are called unwanted branches.

For example

for CFG S

---->AA

A ->a | b

If we want to generate the word "aa" we will leave the branch generated by the production A >b.

Q No 4.What is the difference between intersection and union of a language?

Intersection of two languages will consist of all those words which are in both languages while union of two languages will consist of all those words which are present in at least one language.

Symbol for intersection is 、and for union is U.

Q No 5.What is the difference between Context free languages and regular languages?

Regular languages can be represented by FA's because we do not need any memory to recognize (accept or reject them on FA) them but there is another class of languages that can not be represented by FA's because these languages require that we have some memory (with the help of memory we can store letters of the string we are checking so that we can compare them with next coming letters in the string).

For example language anbn requires that we must store a's and then compare their count with next coming b's so that we can check whether a's are equal to b's or not.

Due to this reason we use Context Free Grammars to represent them because we can5t write RE's for them. So Context Free Languages represent a broader category this category also include regular languages as subcategory. It means that context free languages include regular languages as well as some other languages.

Q No.6 What is the difference between Moore and Mealey machines?

In Mealy Machine we read input string letters and generate output while moving along the paths from one state to another while in Moore machine we generate output on reaching the state so the output pattern of Moore machine contains one extra letter because we generated output for state q0 where we read nothing.

Q No 7.What does the following terms mean

- STACK Consistent
- Y-able Paths
- Working string
- Semi Word means

Stack consistence means that in the PDA converted in the conversion form, when we follow a path segment (which is formed by combining start, read or here state with next read, here or accept state on the path) along the PDA its pop state should have the path for the same letter that is present on the top of the stack at that stage. If this doesn't happen our PDA will crash because in conversion form of the PDA the pop state has only one letter path, so if we could not be able to find that letter on the top of the stack our PDA will crash (if will not find path where to go from that state)

Working string means the string present on the input tape.

Y-able Paths means that when we follow a certain sequence of rows from the row table to generate a path for a word form start state to accept state. The path (sequence of rows) should be stack as well as joint consistent it means that rows should end at the same read or here state (join consistency) and the rows should be able to pop the letter from the top that is indicated in the pop state of the row.

Semi word is the string of terminals it may be null string ending with a Non terminals on the right.

For example some
semi words are aaS
aab
bA
B

Is Automata Theory is a Programming Subject or theoretical?

Automata theory is the study of *abstract* computing devices, or "machines". This topic goes back to the days before digital computers and describes *what is possible to compute using an abstract machine*. These ideas directly apply to creating compilers, programming languages, and designing applications. They also provide a formal framework to analyze new types of computing devices, e.g. biocomputers or quantum computers

What are practical Examples of the implications of Automata Theory and the formal Languages?

Grammars and languages are closely related to automata theory and are the basis of many important software components like:

- Compilers and interpreters
- Text editors and processors
- Text searching
- System verification

What are the Types of Automata?

- The Types of Automata Theory are
- Finite Automata

- Regular Languages
- Linear-bounded Automata
- Context Sensitive Languages
- Push-Down Automata
- Context Free Languages
- Turing Machines
- Recursively innumerable languages

There are others as well like,

- Random Access Machines
- Parallel Random Access Machines
- Arrays of Automata

Question: How types of Automata Differ?

Answer:

They differ in the following areas Complexity (or Simplicity)

Power

In the function that can be computed. In the languages that can be accepted.

Question: What is the difference between the alphabet and an element of a set?

Answer: Alphabets is a set of letters nothing else but a set of strings (elements) can have more than one letters in one string.

Question: Difference between Palindrome and Reverse function?

Answer: The language consisting of ⋯ and the strings s defined over ¦ such that Rev(s)=s.

It is to be denoted that the words of PALINDROME are called palindromes. Reverse (w) = w

Example: ¦={a,b},

PALINDROME={⋯, a, b, aa, bb, aaa, aba, bab, bbb, ...}

If a is a word in some language L, then reverse (a) is the same string of letters spelled backwards, called the reverse of a.

e.g

reverse (xxx)

= xxx reverse

(623) =

326

reverse (140) = 041

Question: Define Strings?

Answer: Concatenation of finite letters from the alphabet is called a string.

e.g If ¦={a,b} then a language L can be defined as

L = {a, abab, aaabb, abababababababab,. }

it's mean all words with a's more or equal to b's

AL-JUNAID INSTITUTE GROUP

Question: Define empty or null strings?

Answer: Concatenation of finite letters from the alphabet is called a string.

Sometimes a string with no symbol at all is used, denoted by (Small Greek letter Lambda) ├ or (Capital Greek letter Lambda) ¨, is called an empty string or null string.

Question: Difference between string and word?

Answer: Any combination of letters of alphabet that follows rules of language is called a word.

A string is a finite sequence of symbols from an alphabet.

Question: There are as many palindromes of length 2n as there are of length 2n-1, please explain?

Answer: If we try to create palindromes then middle elements (2 in even palindromes & 1 in odd palindrome) does not cause any change in no. of palindromes

Defining the language PALINDROME, of length 2n and 2n-1 defined over S = {a,b}

e.g if we take n= 2 for 2n

Length (2n) = 4 and string can be written as {aaaa, abba, baab, bbbb} And if we take n = 2 for 2n-1

Length (2n-1) = 3 and string can be written as {aaa, aba, bab, bbb}

AL-JUNAID INSTITUTE GROUP

Question: Define Kleene Star?

Answer: Given ┆, then the Kleene Star Closure of the alphabet ┆, denoted by ┆*, is the collection of all strings defined over ┆, including ┈.

It is to be noted that Kleene Star Closure can be defined over any set of strings. Examples

If ┆ = {x}

Then ┆* = {┈, x, xx, xxx, xxxx, ….} If ┆ = {0,1}

Then ┆* = {┈, 0, 1, 00, 01, 10, 11, ….}

If ┆ = {aaB, c}

Then ┆* = {┈, aaB, c, aaBaaB, aaBc, caaB, cc, ….} Note:

Languages generated by Kleene Star Closure of set of strings, are infinite languages. (By infinite language, it is supposed that the language contains infinite many words, each of finite length).

Question: Why do not we can write" ba" in the set of PALINDROME while it is reverse of "ab".

Answer: The language consisting of ┈ and the strings s defined over ┆ such that

Rev(s)=s.

It is to be denoted that the words of PALINDROME are called palindromes.

Exampl e For ┆ ={a,b},

PALINDROME={┈ , a, b, aa, bb, aaa, aba, bab, bbb, ...}

All two length string cannot satisfied the palindrome. aa and bb in palindrome but ba and ab are not in palindrome.

AL-JUNAID INSTITUTE GROUP

Question: What are the steps of Recursive Definition of Languages?

Answer: A recursive definition is characteristically a three-step process. First, we specify some basic objects in the set.

Second, we give rules for constructing more objects in the set from ones we already know.

Third, we declare that no objects except those constructed in this way are allowed in the set.

Question: Strings that ending in "a " and strings containing exactly one "a".

Answer:

Its means all string ending in a

e.g ┆= {a, b}

{a, aa, ba, aba, baa,…….} Exactly a,

defined over ┆= {a, b}

{ab, ba, abb, bba,… }

Question: What is Lexical Analyzer?

Answer:

The first phase of the compiler is the lexical analyzer, also known as the scanner, which recognizes the basic language units, called tokens.

The exact characters in a token is called its lexeme.

Tokens are classified by token types, e.g. identifiers, constant literals, strings, operators, punctuation marks, and key words.

Different types of tokens may have their own semantic attributes (or values) which must be extracted and stored in the symbol table.

The lexical analyzer may perform semantic actions to extract such values and insert them in the symbol table.

Question: What is accepting string language?

Answer: The strings which follow rules for the language are accepted in language.

Let u and v be strings. Then uv denotes the string obtained by concatenating u with v, that is, uv is the string obtained by appending the sequence of symbols of v to that of u. For example if u = aab and v = bbab, then uv = aabbbab. Note that vu = bbabaab uv. We are going to use first few symbols of English alphabet such as a and b to denote symbols of an alphabet and those toward the end such as u and v for strings.

Question: What is transition table?

Answer: A complete transition table contains one column for each character. To save space, table compression may be used. Only non-error entries are explicitly represented in the table, using hashing, indirection or linked structures. Tabular representation of a function that takes two arguments and returns a value

	0	1
q0◊	q2	q0
*q1	q1	q1
q2	q2	q1

Rows correspond to states Columns correspond to inputs Entries correspond to next states.
The start state is marked with an arrow The accepting states are marked with a star

AL-JUNAID INSTITUTE GROUP

Question: What does it means by the transition?

Answer: Transition means which letter, after being read, is transfer from which place to which place. It is necessary to show transition of every letter from each and every state.

Question: What is Null?

Answer: Λ is a string having no letter in it. e.g (A box having no object in it).

Let us observe that if the alphabet has no letters, then its closure is the language with the null string as its only word, because Λ is always a word in a Kleen closure. Symbolically,

we write

if Σ = {the empty set} then Σ* = {Λ},

This is not the same as if S = {Λ}

then S* = {Λ}

which is also true but for a different reason, that is ΛΛ = Λ.

Example

If L is any language, then

LΛ = ΛL = L

If Λ string concatenates with any string S, it does not cause any change in the string S, if we use Λ in any string then it generates some result that is below here

Λ for both side then string is ΛaaΛ = aa

b for left and Λ for right then string is baaΛ = baa

... for left and b for right then string is ...aab = abb

Question: What is the difference between Regular Languages and Non Regular Languages?

Answer: The language determined by a regular expression is regular otherwise non regular.

Question: What is NFA?

Answer: Nondeterminism plays a key role in the theory of computing. A nondeterministic finite state automaton is one in which the current state of the machine and the current input do not uniquely determine the next state. This just means that a number of subsequent states (zero or more) are possible next states of the automaton at every step of a computation.

Of course, nondeterminism is not realistic, because in real life, computers must be deterministic. Still, we can simulate nondeterminism with deterministic programs. Furthermore, as a mathematical tool for understanding computability, nondeterminism is invaluable.

As with deterministic finite state automata, a nondeterministic finite state automaton has five components.

a set of *states*

a finite *input alphabet* from which input strings can be constructed

a *transition function* that describes how the automaton changes states as it processes an input string

a single designated *starting state*

a set of *accepting states*

The only difference lies in the transition function, which can now target subsets of the states of the automaton rather than a single next state for each state, input pair.

Question: What is a main difference between NFA and FA?

Answer: Finite Automata (FA)

A finite automaton with unique transitions from each state. There are no choices

There is only 1 letter of the alphabet per transition (the label on the edges in the graph is limited to 1)

ꟷ transitions are not allowed.

No implied trap states. That is, if the letter of alphabet has n letters in it, every state will have n edges coming out of it. If the letters are not part of a valid word, then the edges will go into a special state, called the trap states. Trap states are the NO states.

Nondeterministic Finite Automaton (NFA)

Has the freedom to do various different moves when in a state and seeing some input

This is modeled mathematically as

The ability to be in various states at once

Accepting a string whenever at least one of those states is accepting.

AL-JUNAID INSTITUTE GROUP

Question: How to obtain 9's complement?

Answer: The (r – 1)'s Complement

Given a positive number N is base r with an integer part of n digits and a fraction part of m digits, the (r-1)'s complements of N is defined as

$r^n - r^{-m} - N$. Some numerical examples follow:

The 9's complement of (52520)10 is (10^5 – 1 – 52520) = 99999 – 52520 = 47479.

No fraction part, so $10^{-m} = 10^0 = 1$.

The 9's complement of (0.3267)10 is (1 – 10^{-4} – 0.3267) = 0.9999 – 0.3267 =

0.6732

No integer part, so $10^n = 10^0 = 1$.

The 9's complement of (25.639)10 is (10^2 – 10^{-3} – 25.639) = 99.9999 - 25.63967

= 74.360

Question: What is DELAY box?

Answer: It is a component which held input for some time and then forwards it just a holder.

Question: What is the difference between pumping Lemma 1 and pumping Lemma 2?

Answer: Infact PLI & PLII are same (A way to recognize Non Regular language). The only difference is in PLII we take care about the substring x & y that length (x) + length (y) less than or equal no. of state of machine. This is because through PLI palindrome (that is Non Regular) is proved to be regular and through PLII this problem is fixed.

AL-JUNAID INSTITUTE GROUP

Question: What is pumping lemma? And what is history?

Answer: A theorem to check validity (Regularity) of an infinite language should not be used with finite languages. Whenever an infinite is regular then there must be a loop (circuit) because without a loop means infinite no. of states that is not possible practically. (Machine can have finite states only)

Question: What is the difference between semi word and word?

Answer: A word is complete combinations of terminals only e.g. abba or ab or a or null string.

Semiword: A semiword is a string of terminals (may be none) concatenated with exactly one nonterminal on the right i.e. a semiword, in general, is of the following form

(terminal)(terminal)… (terminal)(nonterminal)

Question: What is the difference between derivation tree and total tree?

Answer: A Derivation tree is the one that shows how to derive any specific word of the language described by CFG but Total Language Tree shows all words of the

Language described by CFG on it.

Question: How to identify a production by it, ambiguity will be removed?

Answer: It is a matter of practice that one can know how to remove ambiguity from it, only practice makes you efficient enough to do it in less time.

Question: Difference between Nullable and Null production? How to make CFG for Nullable and for Null?

Answer: The production of the form

nonterminal ® L

is said to be ***null production***.

Example:

Consider the following CFG

S ® aA|bB|L, A ® aa|L, B ® aS

Here S ® L and A ® L are null productions.

A production is called ***nullable production*** if it is of the form N ® L

or

there is a derivation that starts at N and leads to L *i.e.*

N1 ® N2, N2 ® N3, N3 ® N4, …, Nn ® L,

where N, N1, N2, …, Nn are non terminals.

Example:

Consider the following CFG S ® XY, X ® Zb, Y ® bW

Z ® AB, W ® Z,

A ® aA|bA|L B ®Ba|Bb|L.

Here A ® L and B ® L are null productions, while Z ® AB, W ® Z are nullable productions.

Method:

Delete all the Null productions and add new productions *e.g.*

Consider the following productions of a certain CFG X ® aNbNa, N ® L, delete the production N ® L and using the production X ® aNbNa, add the following new productions X ® aNba, X ® abNa and X ® aba

Thus the new CFG will contain the following productions X ® Nba|abNa|aba|aNbNa

Note: It is to be noted that X ® aNbNa will still be included in the new CFG.

Method:

Consider the following CFG S ® XY, X ® Zb, Y ® bW

Z ® AB, W ® Z,

A ® aA|bA|L B ®Ba|Bb|L.

Here A ® L and B ® L are null productions, while Z ® AB, W ® Z are nullable productions. The new CFG after, applying the method, will be

S ® XY

X ® Zb|b Y ® bW|b

Z ® AB|A|B

W ® Z

A ®
aA|a|bA|b
B
®Ba|a|Bb|
b

Note: While adding new productions all Nullable productions should be handled with care. All Nullable productions will be used to add new productions, but only the Null production will be deleted.

Question: Is it possible to make CFG for infix and postfix expression's using derivation tree?

Answer: Derivation tree is only used to derive words of language that is described by a CFG. Yes, we can create CFG for languages infix expressions, postfix expressions.

Question: What are the uses of push down automata in computing?

Answer: PDA is just an enhancement in FAs. i.e Memory is attached with machine that recognizes some language. FA is basic structure for most advanced electronic machines such as computer etc.

AL-JUNAID INSTITUTE GROUP

Question: What is difference between PUSH DOWN STACK and PUSH DOWN STORE?

Answer: No difference at all. Both terms are used to describe memory structure attached with FAs to store some characters in it.

Question: How to accommodate NULL string if it is part of language during converting from CFG to CNF and building FA's?

Answer: When we convert CFG to CNF and Null is a part of language then null string is not part of language in CNF. This is the only change a language gets in CNF. When we draw an FA for a CFG, there is no change in language and simply draws FA that accepts the language of CFG.

Question: How to accommodate NULL string if it is part of language during converting from CFG to CNF and in building PDA?

Answer: When we convert CFG to CNF and Null is a part of language then null string is not part of language in CNF. This is the only change a language gets in CNF. When we draw an PDA for a CFG, there is no change in language and simply draws PDA that accepts the language of CFG.

Question: What is Push down Automata?

Answer: Pushdown Automaton (PDA), consists of the following

An alphabet S of input letters.
An input TAPE with infinite many locations in one direction. Initially the input string is placed in it starting from first cell; the remaining part of the TAPE is empty.
An alphabet G of STACK characters.
A pushdown STACK which is initially empty, with infinite many locations in one direction. Initially the STACK contains blanks.
One START state with only one out-edge and no in-edge.
Two halt states i.e. ACCEPT and REJECT states, with in-edges and no out-edges.
A PUSH state that introduces characters onto the top of the STACK.
A POP state that reads the top character of the STACK (may contain more than one out-edges with same label).

A READ state that reads the next unused letter from the TAPE (may contain more than one out-edges with same label).

Question: Why we study Automata?

Answer: Automata theory is the study of abstract computing devices, or "machines". This topic goes back to the days before digital computers and describes what is possible to compute using an abstract machine. These ideas directly apply to creating compilers, programming languages, and designing applications. They also provide a formal framework to analyze new types of computing devices, e.g. biocomputers or quantum computers. Finally, the course should help to turn you into mathematically mature computer scientists capable of precise and formal reasoning. More precisely, we'll focus primarily on the following topics. Don't worry about what all the terms mean yet, we'll cover the definitions as we go:

1. Finite state automata: Deterministic and non-deterministic finite state machines; regular expressions and languages. Techniques for identifying and describing regular languages; techniques for showing that a language is not regular. Properties of such languages.

AL-JUNAID INSTITUTE GROUP

2. Context-free languages: Context-free grammars, parse trees, derivations and ambiguity. Relation to pushdown automata. Properties of such languages and techniques for showing that a language is not context-free.
3. Turing Machines: Basic definitions and relation to the notion of an algorithm or program. Power of Turing Machines.
4. Undecidability: Recursive and recursively enumerable languages. Universal Turing Machines. Limitations on our ability to compute; undecidable problems.
5. Computational Complexity: Decidable problems for which no sufficient algorithms are known. Polynomial time computability. The notion of NP-completeness and problem reductions. Examples of hard problems.

Question: What is valid and invalid alphabets explain with example?

Answer: Example 1

If s=abc is a string defined

over ¦= {a,b,c} then Rev(s) or

sr = cba

¦= {a,b}

s=abbaa

Rev(s)=a

abba

When more then letter in the alphabet you have to be quite careful that don't reverse the symbols however you write the letter from right to left.

Example 2

¦= {B, aB,

bab, d}

s=BaBbabB

d

Rev(s)=dBb

abaBB

Example 3

¦= {ab, b,

aa} s=abbaa

Rev(s)=aaba

b

¦1= {B, aB, bab, d} is valid alphabet as there is no letter in ¦1 that lies in start of any other letter means all the tokens of any word (string) will be unique.

Whereas in ¦2= {B, Ba, bab, d} letter B lies in start of letter Ba.

This makes it difficult to decide which token to select at some point if B occurs in any string.

Question: Why we use Capital Letters for Languages.

Is it possible to combine two languages together like EVEN-EVEN & EQUAL, and so on?

Answer: We use capital letter for our convenient and yes you can combine two languages.

Question: What are the rules to form WORDS in languages developed by Automata?

Are strings not following any rule?

Answer: Rules are different for different languages.

e.g ┆={a,b}

The language L of strings of even length, defined over ┆ = {a,b}, can be written as Valid for even length L= {aa, bb, aabb, bbaa, baab, abba …..}

Invalid for even length L= {a, b, aaa, bbb, aba, bab, …….}

The language L of strings of odd length, defined over ┆ = {a,b}, can be written as Valid for odd length L1= {a, b, aaa, bbb, aba, bab, …….}

Invalid for odd length L= {aa, bb, aabb, bbaa, baab, abba …..} Strings cannot follow any rule.

Question: What are graphs of palindromes of length 2n and length 2n-1?

Answer: Palindromes of even length are always symmetric about the middle line and

palindromes of odd length are always symmetric about middle letter.

If we try to create palindromes then middle elements (2 in even palindromes & 1 in odd palindrome) does not cause any change in no. of palindromes

Defining the language PALINDROME, of length 2n and 2n-1 defined over S = {a,b}

e.g if we take n= 2 for 2n

Length (2n) = 4 and string can be written as {aaaa, abba, baab, bbbb} And if we take n = 2 for 2n-1

Length (2n-1) = 3 and string can be written as {aaa, aba, bab, bbb}

Question: How can we write a RE for a given number of words?

Answer: In example let us consider a finite language L that contains all the strings of a's and b's of length three exactly:

L = {aaa aab aba abb baa bab bba bbb}

The first letter of each word in L is either an a or a b. The second letter of each word in L is either an a or a b. The third letter of each word in L is either an a or a b. So, we may write

L = language ((a+b)(a+b)(a+b)) Or

L = language $((a+b)^3)$

If we want to define the set of all seven-letter strings of a's and b's, we could write $(a+b)^7$. In general, if we want to refer to the set of all possible strings of a's and b's of any length whatsoever, we could write (a+b)*

This is the set of all possible strings of letters from the alphabet $\sum = \{a, b\}$ including the null string. This is a very important expression and we shall use it often.

Again, this expression represents a language. If we choose that * stands for 5, then (a+ b)* give s

$(a+b)^5 = (a+b)(a+b)(a+b)(a+b)(a+b)$

We now have to make five more choices: either a or b for the first letter, either a or b for second letter, and so on.

21. Please explain that in some expression having more then one plus sign the resulting string is only one from them or it can be more then one? For example if there is an expression: a +b + c, with out any small bracket between them the resultant string is "a or b or c" OR it cans both "a or b and c”,"a and b or c"

The resulting string is only one from them. Whenever we put + signs between n words that means one option (only) out of all available.

Question: One Language generates many REs?

Answer: Sometimes we generate many Regular Expressions (RE) for one language these REs are called Equal RE.

Example:

Consider the following regular expressions r1= (a + b)* (aa + bb)

r2= (a + b)*aa + (a + b)*bb then

both regular expressions define the language of strings ending in aa or bb. This RE (aa+bb) is separated by +. Whenever RE is separated by +, two possibilities occur

1. Before + part
2. After + part

(a+b)*aa comes before + part, means RE generate all strings that end with aa. (a+b)*bb comes after + part this means RE generate all strings that end with bb. In short we can say r2 generates the language of strings ending either aa or bb this is equal to r1.

Question: Rules for determining RE for a given language defined on a set?

Answer: The following rules define the language associated with any regular expression: Rule 1: The language associated with the regular expression that is just a single letter is that one-letter word alone and the language associated with null is just {null}, a one-word language.

Rule 2: If r1 is a regular expression associated with the language L1 and r2 is regular expression associated with the language L2, then:

The regular expression (r□1)(r2) is associated with the product L1L2 that is the language L1 times L2

language (r1r2) = L1L2

The regular expression r□1 + r2 is associated with the language formed by the union of the sets L1 and L2:

language (r1 + r2) = L1 + L2

The language associated with the regular expression (r□1)* is L1*, the Kleene closure of the set L1 as a set of words:

language(r1*) = L1*

Once again, this collection of rules proves recursively that there is some language associated with every regular expression. As we build up a regular expression from the rules, we simultaneously are building up the corresponding language.

Question: What is EVEN - EVEN LANGUAGE?

Answer: Even-Even means count of a's is even and cout of b's is also even.

Even + Even = Even (Proved)

So we can divide any string excluding ... which is also in Even-Even in substrings of length 2 each. It gives us following combinations

aa, bb makes no change in string status ab, ba create disorder in string status so (aa+bb+(ab+ba)(aa+bb)*(ab+ba))*.

Question: What is tokenizing string?

Answer: Tokenize a string means make its valid units. Question: What is whole star?

Answer: Whole star of any RE means all possible combinations of that RE including ….

Question: How to use + operator in Automata?

Answer: Plus Operation is same as Kleene Star Closure except that it does not generate … (null string), automatically.

Question: How to know what is RE?

Answer: It is a matter of practice that you come to know what does an RE represent. You may start with simpler REs and later you will be able to recognize different REs gradually.

Question: Why we use null string in FA?

Answer: If null string is part of our language then we have to handle it in FA, its not compulsion.

Question: What is tree structure?

Answer: A tree is a connected undirected graph with no simple circuits. Since a tree cannot have a simple circuit, a tree cannot contain multiple edges or loops. Therefore any tree must be a simple graph.

AL-JUNAID INSTITUTE GROUP

Question:	What is difference between (a, b) & (a + b)?
Answer:	(a, b) & (a + b) are same and both represents either a or b.

AL-JUNAID INSTITUTE GROUP

Question: Difference between FA & TG?

Answer: Finite Automata (FA)

- A finite automaton with unique transitions from each state.
- There are no choices
- There is only 1 letter of the alphabet per transition (the label on the edges in the graph is limited to 1)
- ··· transitions are not allowed.
- No implied trap states. That is, if the letter of alphabet has n letters in it, every state will have n edges coming out of it. If the letters are not part of a valid word, then the edges will go into a special state, called the trap states. Trap states are the NO states.

Transition Graphs (TG)

- TG are generalizations of FA.
- Can change state without an input.
- Can read more than one letter at a time.
- Can have a regular expression as a edge label.
- Can have more then one start state.

We are not bound in TG. We are only given a freedom or relation which we are not forced to use. We may or may not use it on will.

In fact we enjoy freedom of staying at multiple places at one time in TGs while reading any letter strings which provides us a facility that any one available path it leads us to final state, word is accepted.

AL-JUNAID INSTITUTE GROUP

Question: Difference between $(a+b)^+$ and $(a+b)^*$?

Answer: $(a + b)^+$ means, we repeat RE a + b infinite (any) no. of times but minimum once, whereas $(a + b)^*$ means we may repeat a + b any no. of times even zero times. () are only used to clear (distinguish) the one RE from some other RE.

Question: What is length of string?

Answer: The length of a string indicates how many symbols are in that string.

For example, the string 0101 using the binary alphabet has a length of 4. The standard notation for a string w is to use |w|.

For example:

Length of string: |0101| is 4.

Length of string: |0010| = 4, |aa| = 2, | |= 0

Question: After looking the on diagram how can we say it is TG or is FA?

Answer: Every FA is also a TG but not every TG is FA. In every FA, every state shows transition of all letters of given alphabet but in any TG it is not must. In TG, we may or may not show all letters transition according to requirement. We can also show transitions on reading any strings in TGs but it is not possible in FAs.

Question: Differentiate between FA, TG and GTG?

Answer: Every FA is also a TG but not every TG is FA. In every FA, every state shows transition of all letters of given alphabet but in any TG it is not must. In TG, we may or may not show all letters transition according to requirement.

We can also show transitions on reading any strings in TGs but it is not possible in FAs. In GTG Directed edges connecting some pair of states labeled with regular expression. It may be noted that in GTG, the labels of transition edges are corresponding regular expressions. In TG we write strings and in GTG we are bound to write RE.

Question: Difference between even clumps and odd clumps?
Answer: Letters may be a's (for example) in even count at one place.

e.g.
aab =
valid baa
= valid
aba =
invalid
abab =
invalid

Question: Difference between containing and consisting?
Answer: Containing something (e.g. R1) means R1 is there (must), whatever other things are, we don't care. Consisting something (e.g. R1) means only R1 is there all other things should be rejected.

Question: Define the main formula of Regular expressions? Define the back ground of regular expression?

Answer: Regular expressions are a notation that you can think of similar to a programming language. In fact, regular expressions are quite fundamental in some programming languages like perl and applications like grep or lex.

Regular expressions are similar to NFA and end up describing the same things we can express with a finite automaton. However, regular expressions are declarative in what strings are accepted, while automata are machines that accept strings. We use Regular expressions for defining the languages.

Say that R is a regular expression if R is:

1. a for some a in the alphabet ¦, standing for the language {a}
2. ⋯ , standing for the language { ⋯}
3. R1+R2 where R1 and R2 are regular expressions, and + signifies union
4. R1R2 where R1 and R2 are regular expressions and this signifies concatenation
5. R* where R is a regular expression and signifies closure
6. (R) where R is a regular expression, then a parenthesized R is also a regular expression

A set of strings from an alphabet. The set may be empty, finite or infinite.

The building blocks of regular languages are symbols, concatenation of symbols to make strings (words), set union of strings and Kleene closure (denoted as *, also called the Kleene star, it should be typed as a superscript but this is plain text.)

Informally, we use a syntax for regular expressions.

Using sigma as the set {0, 1} (an alphabet of two symbols)

01110 is a string starting with the symbol 0 and then concatenating 1, then 1, then 1, and finally concatenating 0. No punctuation is used between symbols or strings that are concatenated.

(01+10) is a union of the two strings 01 and 10. The set {01, 10}

(00+11)* is the Kleene closure of the union of 0 concatenated with 0 and 1 concatenated with 1.

The Kleene closure (star) is defined as the concatenation of none, one, two, or any countable number strings it applies to.

Examples of Kleene star:

1* is the set of strings {⋯, 1, 11, 111, 1111, 11111, etc. } This set is infinite.

(1100)* is the set of strings {⋯, 1100, 11001100, 110011001100, etc. } (00+11)* is the set of strings {⋯, 00, 11, 0000, 0011, 1100, 1111, 000000, 000011, 001100, etc. }

Note how the union (+ symbol) allows all possible choices of ordering when used with the Kleene star.

(0+1)* is all possible strings of zeros and ones, often written as ¦ * where ¦ = {0, 1}(0+1)*

(00+11) is all strings of zeros and ones that end with either 00 or 11. Note that concatenation does not have an operator symbol.

(w)+ is a shorthand for (w)(w)* w is any string or expression and the superscript plus, + , means one or more copies of w are in the set defined by this expression.

Question: Are S* and S^+same?

Answer: No because S+ means same as Kleene Star Closure except that it does not generate … (null string), automatically. So in the above example, there is no null string.
Concatenation of finite letters from the alphabet is called a string.
If we have S={a, bb, bab, abaab}, first we factorize the string like (a) (bb) (bab) (abaab) then concatenate to each other and make more string to its concatenation but the string abbabaabab is not in S* because the last member(ab) of the group does not belong to S, so abbabaabab is not in S.

Question: What is set?

Answer: Set is a collection of distinct objects.

AL-JUNAID INSTITUTE GROUP

Question: How many Methods of defining the languages?

Answer: 1. You can describe a language in English like statement

2. You can define a language by putting all its words in a set
3. You can define a language in a mathematical way
4. You can define a language by Recursive Definition
5. You can define a language by Regular Expression
6. You can define a language by Finite Automata
7. You can define a language by Transition Graph
8. You can define a language by Context Free Grammar

Question: What is difference between Palindrome, Kleene star closure and plus operation?

Answer: The language consisting of ⋯ and the strings s defined over ┆ such that Rev(s)=s.

or

A string x is a palindrome if $x^R=x$.

It is to be denoted that the words of PALINDROME are called palindromes. Reverse (w) = w

Example: ┆={a,b},

PALINDROME={⋯ , a, b, aa, bb, aaa, aba, bab, bbb, ...}

If a is a word in some language L, then reverse (a) is the same string of letters spelled backwards, called the reverse of a.

e.g

reverse (xxx)

= xxx reverse

(623) = 326

reverse (140) = 041

Given ¦, then the Kleene Star Closure of the alphabet ¦, denoted by ¦*, is the collection of all strings defined over ¦, including ….

Plus Operation is same as Kleene Star Closure except that it does not generate … (null string), automatically.

Question: What is inside language?

Answer: Any language contains words in it and certain rules to validate strings for that language.

Question: What is Equal RE?

Answer: Sometimes we generate many REs for one language these are called Equal RE.

Question: How can we make Finite Automaton from a language and a language from FA?

Answer: There is not any formal procedure to design FA for a language. This ability just improves with time and practice.

Question: Could we just use +, - symbols with x not with y in FA?

Answer: Yes, you can use only + or – for the place of x and y, but remember when you don't write + or –, So you should write start and final at the beginning or ending state Or you should indicated by an arrow and a final state by drawing box or another circle around its circle because if you don't write how can we indicate the start and final state.

Question: Explain the language L of string, defined over ∑= {0,1}, having double 0's and double 1's?

Answer: Language of strings with 00 or 11 means string that have substrings 00 or 11 in them at least.

Minimum words which are included in this language are 00 and 11. This language does not accept null and it also does not accept 0 or 1.

Question: What are the difference between single 1 and 0 and double 1's and 0's?

Answer: 1. Consider the language L of strings, defined over

If ¦={0, 1}, having words with either 0's or 1's without null. The language L may be expressed by RE

$(0 + 1)^+$

When you make string by above RE you have all possible combination of 0's and 1's except null

i.e {0, 1, 00, 01, 10, 11,… }. Minimum words which are included in this language are 0 and 1.

2. Consider the language L of strings, defined over

If ¦={0, 1}, having double 0's or double 1's, The language L may be expressed by the RE (0+1)* (00 + 11) (0+1)*

Double 1's and 0's means clumps of letter which will always come together.

Minimum words which are included in this language are 00 and 11.

AL-JUNAID INSTITUTE GROUP

Question: On what basis we select initial and final states?

Answer: It depends on the expression given to us.

Question: How we know that the given expression has how many states?

Answer: There is not any formal procedure to know the number of states. This ability just improves with time and practice.

Question: How will we develop the rules of transition?

Answer: Transition means which letter, after being read, is transfer from which place to which place. It is necessary to show transition of every letter from each and every state.

Question: Can we accept the strings going from final to initial?

Answer: It is to be noted that if any state start from the final state it does not accept any string. Even it does not accept the null string, because there is no path starting from initial state and ending in final state.

Question: What are the basic rules to build FA?

Answer: One and only rule is to build a Finite Automata (FA) should accept all words of the language and reject all the words which are not part of the language. Any FA that ensures these above things is the right FA for the language.

Note: One language can have many FA.

Question: What is Dead state?

Answer: The DEAD STATE is introduced to be able to make an automaton complete without altering its behavior.