

## CS402 Short Notes

Q No 1

What is the difference between the strings and the words of a language?

**Answer:** A string is any combination of the letters of an alphabet where as the words of a language are the strings that are always made according to certain rules used to define that language. For example if we take

Alphabet  $\Sigma = \{ a , b \}$  Here a , b are the letters of this alphabet. As you can see we can make a lot of strings from these letters a and b.

For example a,b,aa,ab,ba,bb,aaa,aab,aba,baa,.....  
.... and so on.

But when we define a language over this alphabet having no a's and only odd number of b's. Then the words of this

language would have only those strings that have only odd number of b's and no a's. some example words of our defined language are b , bbb , bbbbbb , bbbbbbb  
,.....and so on.

So we can say that all the words are strings but all the strings may not be the words of a language. Hence strings are any combination of letters of an alphabet and the words of a language are strings made according to some rule.

QNo.2 What is the difference between an Alphabet and an element of a set.

Whether Alphabet is an element of a set or it is a set itself?

**Answer:** An Alphabet is a set in itself. The elements of an Alphabet are called letters .

For example

Binary Alphabet  $\Sigma = \{0,1\}$

Here 0,1 are the letters of binary alphabet.

Binary Alphabet is very important because it the Alphabet used by the computer.

Set of Natural Numbers  $N = \{1,2,3,4,5, \dots\}$

Here 1,2,3..... are the elements of set of Natural Numbers.

Q No.3 What is Null String () ?

**Answer:**

The string with zero occurrences of symbols (letters) from

$\Sigma$ .

It is denoted by (Small Greek letter Lambda)  $\lambda$  or (Capital Greek letter Lambda)  $\Lambda$ , is called an empty string or null string.

The capital lambda will mostly be used to denote the empty string, in further discussion.

**Q No.4 What is PALINDROME ?**

**Answer:**

The language consisting of (Null String) and the strings  $s$  defined over an Alphabet  $\Sigma$  such that  $\text{Rev}(s)=s$ .

Some example words of this language are aa

As  $\text{Rev}(aa) = aa$

As  $\text{Rev}(aba) = aba$

As  $\text{Rev}(bbb) = bbb$

As  $\text{Rev}(aabaa) = aabaa$

$bbbbaabbb$

As  $\text{Rev}(bbbbaabbb) = bbbbaabbb$

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

**QNo5.What is the concept of valid and invalid alphabets ?** **Answer:**While defining an alphabet of letters consisting of more than one symbols, no letter should be started with any other the letter of the same alphabet i.e. one letter should not be the prefix of another. However, a letter may be ended in the letter of same alphabet i.e. one letter may be the suffix of another.  $\Sigma = \{ a, b \}$  ( Valid Alphabet)  $\Sigma = \{ a, b, cd \}$  ( Valid Alphabet)  $\Sigma = \{ a, b, ac \}$  ( Invalid Alphabet)

QNo 6. What is ALGOL ?

**Answer:**ALGOL (ALGOritmic Language) is one of several high level languages designed specifically for programming scientific computations. It started out in the late 1950's, first formalized in a report titled ALGOL 58, and then progressed through reports ALGOL 60, and ALGOL 68. It was designed by an international committee to be a universal language. Their original conference, which took place in Zurich, was one of the first formal attempts to address the issue of software portability. ALGOL's machine independence permitted the designers to be more creative, but it made implementation much more difficult. Although ALGOL never reached the level of commercial popularity of FORTRAN and COBOL, it is considered the most important language of its era in terms of its influence on later language development.

ALGOL's lexical and syntactic structures became so popular that virtually all languages designed since have been referred to as "ALGOL - like"; that is they have been hierarchical in structure with nesting of both environments and control structures. **Answer:Sequencing**

**QNo7. What are the Sequential Operators?**

Sequencing operators

a >> b                      Sequence                      Match a and b in sequence

Sequential- and

a && b

Sequential-and. Same as above, match a and b in sequence

a || b                      Sequential-or Match a or b in sequence

The sequencing operator >> can alternatively be thought of as the sequential-and operator. The expression a && b reads as match a and b in sequence. Continuing this logic, we can also have a sequential-or operator where the expression a || b reads as match a or b and in sequence. That is, if both a and b match, it must be in sequence; this is equivalent to a >> !b | b.

**QNo 8.What is Non-Determinism and Determinism and what is the difference between them ?**

**Answer:**Determinism means that our computational model (machine) knows what to do for every possible inputs. Non determinism our machine may or may not know what it has to do on all possible inputs.

As you can conclude from above definition that Non-Deterministic machine can not be implemented ( used ) on computer unless it is converted in Deterministic machine.

**QNo 9. What is meant by equivalent FA's ?**

**Answer:**FA's that accept the same set of languages are called Equivalent FA's.

**QNo 10. What is the difference between Palindrome and Reverse function?**

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

Reverse (w) = w

Example:  $\Sigma = \{a,b\}$ ,

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 **QNo11.Define Kleene Star?**

**Answer:** Given  $\Sigma$ , then the Kleene Star Closure of the alphabet  $\Sigma$ , denoted by  $\Sigma^*$ , is the collection of all strings defined over  $\Sigma$ , including

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

Examples

If  $\Sigma = \{x\}$

Then  $\Sigma^* = \{x, xx, xxx, xxxx, \dots\}$  If  $\Sigma = \{0,1\}$

Then  $\Sigma^* = \{0, 1, 00, 01, 10, 11, \dots\}$

If  $\Sigma = \{aaB, c\}$

Then  $\Sigma^* = \{aaB, c, aaBaaB, aaBc, caaB, cc, \dots\}$  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) **QNo12.Valid/In-Valid alphabets?**

**Answer:** Any alphabet is valid if any of its letter does not appear in the start of any other letter otherwise it is invalid.

**QNo13.What is Reverse of a string?**

**Answer:** Alphabet provides only a set of symbols. A string is a concatenation of these symbols. Reverse of the string means to write the string in reverse order. It has no effect on alphabet.

Alphabet will remain same.

**QNo14.Differentiate Kleene Star Closure and PLUS? Answer:** Given  $\Sigma$ , then the Kleene Star Closure of the alphabet  $\Sigma$ , denoted by  $\Sigma^*$ , is the collection of all strings defined over  $\Sigma$ , including .

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

You can use other symbol for alphabet but we are mostly use sigma symbol.

**QNo15.Define Regular Expression?**

**Answer:** Regular Expression is the generalized form of any regular language through which you can construct any string related to that language. Take an example from your handouts  $L_1 = \{a, aa, aaa, \dots\}$  and  $L_2 = \{a, aa, aaa, aaaa, \dots\}$  can simply be expressed by  $a^*$  and  $a^+$ , respectively. so  $a^*$  and  $a^+$  are the generalized form of Languages  $L_1, L_2$ .

And  $a^*$  and  $a^+$  are called the regular expressions (RE) for  $L_1$  and  $L_2$  respectively.

computer science, a finite-state machine (FSM) or finite-state automaton (FSA) is an abstract machine that has only a finite, constant amount of memory. The internal states of the machine carry no further structure. This kind of model is very widely used in the study of computation and languages.

**Q No.2 What is the difference between FA , TG , GTG. ?**

In every FA, we mark transitions with single letter of the given alphabet but in TG transitions can be marked with letters or strings (combination of letters).

In every FA, every state shows transition for all letters of given alphabet but in any TG it is not necessary to show all transition for all letters of given alphabet. In TG, we may or may not show all letter transitions according to requirement. We can also show transitions on reading any strings in TGs but it is not possible in FA's. In GTG Directed edges connecting some pair of states are labeled with regular expressions . 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. Every FA is also a TG but not every TG is FA.

**Q No.3 What is the difference between FA's and TG's .Why we need TG's when we have FA's?**

The Transition Graphs (TG) differ from FA in the following areas

TG's are generalizations of FA's.

TG's can change state without an input ( Null transition).

Can read more than one letter (words of the language they are accepting) along the transition edges at a time. Can have a regular expression as a edge label. Can have more then one start state.

We have been given more freedom in TG's. But this freedom is on the cost of more memory and processing power it means that if we implement TG's on computer using some programming language it will need more memory and processing power of computer than used in the implementation of FA's.

**Q No.4 What is the concept of the Union of FA's ?**

When we take Union of two FA's it means that resultant FA's should accept all the words that were accepted by the two FA's individually. It is like taking union of two sets, the resultant set contain members of both sets.

For example

Let A

= {1,3,5,7,9}

and

B = {0,2,4,6,8,10} then,  $A \cup B = \{ 0,1,2,3,4,5,6,7,8,9,10 \}$  you can see that A U B

contain elements of both sets similar is the case with FA's. **Q No.5 What is the difference between is TG and GTG ?**

In TG, there are letter transitions for the strings. While in GTG, one can write whole RE as a transition from one state to another one.

**Q No.6 How one can create RE of a particular language?**

First thing about RE and FA is that there is no hard and fast formula or method to generate these. One can generate them by its mental approach. And this mental approach can be acquired through only PRACTICE. Here are some useful tips to write RE's,

□

Let our language consist of the words of length three exactly over alphabet  $\Sigma = \{a,b\}$  then it consists of the words  $L = \{aaa, aab, aba, abb, baa, bab, bba, bbb\}$ . Its RE can be simply written as

$RE = aaa + aab + aba + abb + baa + bab + bba + bbb$  which simply means that our language consists of **only** these words. So we can make RE for a finite language by writing its all words with + operator between them.

□

We should also keep the null string in our mind. If our language generates null string than our RE should also generate it)

For example language having all the words of even length has null string in it as well so we can write its RE as follows

$$RE = ((a+b)(a+b))^*$$

This RE also generates null string.

If a language generates all strings starting with a. then strings will be of type a , aa, ab, aab, aaa, aba, abb,....

Here RE should start with 'a' and then all strings including null. So this will be  $(a + b)^*$  and complete RE is  $a(a + b)^*$ .

Similarly languages of strings ending in b will have RE  $(a + b)^*b$ .

**Q No.7 What is the diagrammatic difference between FA's and TG's?**

The main differences between FA's and TG's are as follows

□

□

□

**Q No.8 What is the corresponding FA for RE  $aa((a+b)(a+b))^*$**  RE is  $aa((a + b)(a + b))^*$ . Its corresponding FA is as follows.

**Q No.9 What is difference between FA's and NFA's. Are they opposite to each other ?**

FA stands for finite automata while NFA stands for non-deterministic finite automata In FA there must be a transition for each letter of the alphabet from each state. So in FA number of transitions must be equal to (number of states \* number of letter in alphabet).

While in NFA there may be more than one transition for a letter from a state. And finally every FA is an NFA while every NFA may be an FA or not.

**Q No.10 Differentiate between (a,b) and (a+b)?**

(a, b) = Represents a and b.

(a + b) = Represents either a  
or b. FAQ's about Lectures

11 to 15

**Q No.1 What is the difference between how's FA and TG .Why we need TG's when we have FA's?**

The Transition Graphs (TG) differ from FA in the following areas TG's can change state without an input ( Null transition).

We have been given more freedom in TG's. But this freedom is on the cost of more memory and processing power it means that if we implement TG's on computer using some programming language it will need more memory and processing power of computer than used in the implementation of FA's.

**Q No.2 What is the concept of the Union of FA's ?**

When we take Union of two FA's it means that resultant FA's should accept all the words that were accepted by the two FA's individually. It is like taking union of two sets the resultant set contain members of both sets.

For example

Let A

= {1,3,5,7,9}

and

B = {0,2,4,6,8,10} then,  $A \cup B = \{ 0,1,2,3,4,5,6,7,8,9,10 \}$  you can see that  $A \cup B$  contain elements of both sets similar is the case with FA's.

**Q No.3 What is the difference between GT and GTG ?**

In TG, there are transitions for the strings. While in GTG, one can write whole RE as a transition from one state to another one.

**Q No.4 How to create a RE of a particular Language?**

Regular expression is used to express the infinite or finite language, these RE are made in such a way that these can generate the strings of that unique language also for the cross check that the defined RE is of a specified language that RE should accept all the string of that language and all language strings should be accepted by that RE.

**Q No.5 How diagrams of FA's are created ?**

It depends upon the question how many states involve in a FA. There is not any formal procedure to design FA for a language. This ability just improves with time and practice. 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.

#### Q No.6 How one can create RE of a particular language?

First thing about RE and FA is that there is no hard and fast formula or method to generate these. One can generate them by their mental approach. And this mental approach can be acquired through only PRACTICE.

I am giving you few tips. I hope those will help you.

If we have a finite language then it will always be regular and will not have \* in RE. e.g.  $L = \{aaa, aba, bb\}$ . L language generates given three strings then its RE will be  $(aaa + aba + bb)$ . So in finite language + of all strings can be it's RE.

If we have an infinite language, then there will be \* in it's RE.

We should also keep the null string in our mind.

For practice just try to create RE of simple languages. Don't try to confuse yourself with complex languages.

For example if a language generates all strings starting with a. then strings will be of type a, aa, ab, aab, aaa, aba, abb,.... Here RE should start with 'a' and then all strings including null. So this will be  $(a + b)^*$  and complete RE is  $a(a + b)^*$ .

Similarly languages of strings ending in b will have RE  $(a + b)^*b$ .

I hope now you will be able to generate the RE of simpler languages. Gradually, increase the complexity of languages to become a perfect in RE's.

Now as similar to RE, FA of finite language will not have any loop in it.

If language is infinite then there will always be at least one loop in its FA. From RE, if you want to generate its FA, then first get the smallest strings and generate their FA and then gradually get the strings of bigger length and keep amending the created FA. After some practice, you will be able to generate the FA's.

And the last thing nobody can do the new task accurately for the first time. Practice is the key to success. In the start you will have lot of mistakes but after practice you will be able to clear all of them.

#### Q No.7 What is the difference between FA's ,and TG's ?

There are two or three big differences between FA's and TG's.

In FA there can be maximum one initial or starting state while in TG there may be more than one initial state.

In FA there can be transition for letters only while in TG transitions from a state to another one can be for strings.

In FA there must be transition from each state for each letter (deterministic) while in TG there may be no transition for specific letter from a state and there may be more than one path for a string or letter from a state.

**Q No.8 What is the exact definition of FA ?**

Definition:

A Finite automaton (FA), is a collection of the followings

Finite number of states, having one initial and some (maybe none) final states. Finite set of input letters ( $\Sigma$ ) from which input strings are formed. Finite set of transitions i.e. for each state and for each input letter there is a transition showing how to move from one state to another.

**Q No.9 What is the difference between TG and GTG ?**

In TG, there are transitions for the strings. While in GTG, one can write whole RE as a transition from one state to another one.

For RE  $aa((a+b)(a+b))^*$  what will be its corresponding FA ? RE is  $aa((a + b)(a + b))^*$ . Its corresponding FA is as follows.

**Q No.10 What is the difference between FA and NFA ?**

FA stands for finite automata while NFA stands for non-deterministic finite automata

In FA there must be a transition for each letter of the alphabet from each state. So in FA number of transitions must be equal to (number of states \* number of letter in alphabet).

While in NFA there may be a transition for a letter from a state. In NFA there may be more than one transition for a letter from a state. And finally every FA is an NFA while every NFA may be an FA.

FA

NFA:

**Q No.11 What is the method to understanding FA's and NFA's** Firstly we know that an FA is used to describe a language. Now a language consists of strings. FA will describe the specific language only if it accepts all the strings of that particular language and all the strings generated by the FA are in the language. So confirmation is of two ways.

Notes Made by Vu-Gateway. . .

Be A part of Vu-Gateway by hitting the link. . .

<https://chat.whatsapp.com/Gecrau2Nit0D4F5DjDiGM>

for any info or help feel free cntct: +92 305 5868956

Now, how to traverse the FA. It is very easy. Every FA has one initial state (state with - sign). From every state of FA there is one transition for every letter of the alphabet. Read the string letter by letter and move according to transitions from state to state. If the string ends in the final state (state with a + sign), that particular string will be accepted otherwise rejected. So, every string ending in final state will be accepted by FA and will be a word of the language.

For NFA, there may be no path or more than one path for a letter from a specific state. As similar to FA just start traversing from the initial state and if the string ends in the final state, it will be accepted.

Remember, as there may be more than one path for a letter from a state. So any path can be used. Goal is to reach the final state. Remaining theory is same to the FA. Practice is the key to success. Just try simple FA's and NFA's in the start.

FAQ's about Lectures 16 to 20

**Q No 1. What is the concept of Nondeterministic Finite Automaton (NFA)**

? 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.

By Vu-Gateway

Notes Made by Vu-Gateway. . .

Be A part of Vu-Gateway by hitting the link. . .

<https://chat.whatsapp.com/Gec1rau2Nit0D4F5DjDiGM>

for any info or help feel free cntct: +92 305 5868956

**Q No 2. If a language can be expressed in the form of FA than why it is needed to use NFA ?**

NFA stands for non-deterministic FA and this sort of structure has relaxation compared with FA. So it is rather more easy to represent a language using NFA.

We have methods to convert NFA into FA's so sometimes it is easier to build NFA of a given language and than convert its NFA into FA using these methods rather than directly building an FA for a language which may be very difficult.

**Q No 3.How to made NFA corresponding to the closure of an FA ?**

While generating NFA corresponding to closure of an FA one should take care of the null string. Simple way to accept null string is declare initial state, final as well. But in this way a lot of other strings will also be accepted. Therefore, accurate way is draw another state. Declare the new state initial as well as final. Connect the new state with the states originally connected with the old start state with the same transitions as the old start state. Newly drawn diagram will be an NFA representing the language closure of the given FA

**Q No 4.What is the difference between Union of two FA's , Concatenation of two FA's and closure of two FA's ?**

Consider two FA's given

Here FA1 accepts all strings ending in a and FA2 accepts all strings ending in b. An FA corresponding to FA1UFA2 will accept all the strings ending in a or ending in b. for example, aba,bbaaab,bbb

An FA corresponding to FA1FA2 will accept all the strings whose first substring belongs to FA1 and second substring belongs to FA2. for example, ababab, bbabbb.

An FA corresponding to FA1\* will accepts all the strings of FA1 including null string. if FA1 represents RE r1 then FA1\* will correspond to RE r1\*.

FAQ's about Lectures 21 to 25

Automata Theory FAQ's about Lectures 21 to 25

**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:Join Us on <http://vustudents.ning.com/>

A sequential circuit contains a memory component.

**By Vu-Gateway**

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.

$$Z=f(X,St)$$

$$S=g(X,St)$$

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 prove Non Regular by Pumping Lemma I are proved Non Regular by pumping Lemma II easily.

Further more 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

Notes Made by Vu-Gateway. . .

Be A part of Vu-Gateway by hitting the link. . .

<https://chat.whatsapp.com/Gecrau2Nit0D4F5DjDiGM>

for any info or help feel free cntct: +92 305 5868956

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.

FAQ's about Lectures 31 to 35

**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

Language described by CFG on it

**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.

Notes Made by Vu-Gateway. . .

Be A part of Vu-Gateway by hitting the link. . .

<https://chat.whatsapp.com/Gec1rau2Nit0D4F5DjDiGM>

for any info or help feel free cntct: +92 305 5868956

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  $(FA_1 \cup FA_2)$  and  $(FA_1 + FA_2)$  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  $\rightarrow L$  is said to be *null production*.

**Example:**

Consider the following CFG  
 $S \rightarrow aA \mid bB \mid L$ ,  $A \rightarrow aa \mid L$ ,  $B \rightarrow aS$   
Here  $S \rightarrow L$  and  $A \rightarrow L$  are null productions.

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

$S \rightarrow aA \mid bB \mid L$   
 $A \rightarrow C \mid bb$   
 $C \rightarrow L$

Here A nullable Non Terminal due to Nullable production  $A \rightarrow C$  as C leads to null. **Example:**

Consider the following  
CFG  $S \rightarrow XY$ ,  $X \rightarrow Zb$ ,  $Y \rightarrow bW$   
 $Z \rightarrow AB$ ,  $W \rightarrow Z$ ,  $A \rightarrow aA \mid bA \mid L$   
 $B \rightarrow Ba \mid Bb \mid L$ .

Here  $A \rightarrow L$  and  $B \rightarrow L$  are null productions, while  $Z \rightarrow AB$ ,  $W \rightarrow Z$  are nullable productions.

**Method:**

Notes Made by Vu-Gateway. . .

Be A part of Vu-Gateway by hitting the link. . .

<https://chat.whatsapp.com/Gec1rau2Nit0D4F5DjDiGM>

for any info or help feel free cntct: +92 305 5868956

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

Consider the following productions of a certain CFG  $X^{\circ} aNbNa$ ,  $N^{\circ} L$ ,

delete the production  $N^{\circ} L$  and using the production  $X^{\circ} aNbNa$ , add the following new productions  $X^{\circ} aNb$ ,  $X^{\circ} abNa$  and  $X^{\circ} aba$

Thus the new CFG will contain the following productions

$X^{\circ} Nba | abNa | aba | aNbNa$

**Note:** It is to be noted that  $X^{\circ} aNbNa$  will still be included in the new CFG.

**Method:**

Consider the following

CFG  $S^{\circ} XY$ ,  $X^{\circ} Zb$ ,  $Y$

$^{\circ} bW$

$Z^{\circ} AB$ ,  $W^{\circ} Z$ ,  $A^{\circ} aA | bA | L$

$B^{\circ} Ba | Bb | L$ .

Here  $A^{\circ} L$  and  $B^{\circ} L$  are null productions, while  $Z^{\circ} AB$ ,  $W^{\circ} Z$  are nullable productions. The new CFG after, applying the method, will be  $S^{\circ} XY$

$X^{\circ}$

$Zb |$

$bY$

$^{\circ}$

$bW$

$|b$

$Z^{\circ}$

$AB | A | B W$

$^{\circ} Z$

$A^{\circ} aA | a | bA | b$

$B^{\circ} Ba | a | Bb | b$

**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.

Notes Made by Vu-Gateway. . .

Be A part of Vu-Gateway by hitting the link. . .

<https://chat.whatsapp.com/Gecrau2Nit0D4F5DjDiGM>

for any info or help feel free cntct: +92 305 5868956

**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 Production)    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.

$$1 \cdot 10^2 + 4 \cdot 10^1 + 1 \cdot 10^0 = 14$$

(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.

FAQ's about Lectures 41 to 45

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

Consider the CFG given below  
 $S \rightarrow ABC$   
 $A \rightarrow aa \mid b$   
 $B \rightarrow c$   
 $C \rightarrow d$

Its CNF will

$S \rightarrow DC$   
 $D \rightarrow AB$   
 $A \rightarrow EE$   
 $E \rightarrow b \mid E_a$   
 $B \rightarrow c$   
 $C \rightarrow d$

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.

Notes Made by Vu-Gateway. . .

Be A part of Vu-Gateway by hitting the link. . .

<https://chat.whatsapp.com/Gec1rau2Nit0D4F5DjDiGM>

for any info or help feel free cntct: +92 305 5868956

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-

POP1HERE-POP2

As we read b on input

tape. READ2-

POP3ACCEPT.

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?**

Notes Made by Vu-Gateway. . .

Be A part of Vu-Gateway by hitting the link. . .

<https://chat.whatsapp.com/Gc1rau2Nit0D4F5DjDiGM>

for any info or help feel free cntct: +92 305 5868956

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  $\cap$  and for union is  $\cup$ .

**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  $a^n b^n$  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 can't 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  $q_0$  where we read nothing.

**Q No 7. What does the following terms mean i.**

ii. Y-able Paths iii. Working string

iv. 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

Notes Made by Vu-Gateway. . .

Be A part of Vu-Gateway by hitting the link. . .

<https://chat.whatsapp.com/Gec1rau2Nit0D4F5DjDiGM>

for any info or help feel free cntct: +92 305 5868956

the row table to generate a path for a word from 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

aabb

A 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

By Vu-Gateway

Notes Made by Vu-Gateway. . .

Be A part of Vu-Gateway by hitting the link. . .

<https://chat.whatsapp.com/Gec1rau2Nit0D4F5DjDiGM>

for any info or help feel free cntct: +92 305 5868956

- Recursively innumerable languages

There are others as well like,

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

Question: How types of Automata Answer:  
Differ?

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

Question: Difference between Palindrome and Reverse function?

Answer: Let  $\Sigma$  be an alphabet and  $s$  a string over  $\Sigma$  and the strings  $s$  defined over  $\Sigma$  such that

The language consisting of

$Rev(s)=s$ .

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

Example:  $\Sigma = \{a, b\}$ ,

PALINDROME =  $\{ \epsilon, a, b, aa, bb, aaa, aba, bab, bbb, \dots \}$

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  $\Sigma = \{a, b\}$  then a language  $L$  can be defined as  $L$

=  $\{a, abab, aaabb, ababababababababab, \dots\}$

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

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)  $\lambda$  or (Capital Greek letter Lambda)  $\Lambda$ , 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

$\{aaa, aba, bab, bbb\}$

Question: Define Kleene Star?

Answer: Given  $\Sigma$ , then the Kleene Star Closure of the alphabet  $\Sigma$ , denoted by  $\Sigma^*$ , is the collection of all strings defined over  $\Sigma$ , including  $\epsilon$ . It is to be noted that Kleene Star Closure can be defined over any set of strings. Examples

If  $\Sigma = \{x\}$

Then  $\Sigma^* = \{\epsilon, x, xx, xxx, xxxx, \dots\}$

If  $\Sigma = \{0,1\}$

Then  $\Sigma^* = \{\epsilon, 0, 1, 00, 01, 10, 11, \dots\}$

If  $\Sigma = \{aaB, c\}$

Then  $\Sigma^* = \{\epsilon, aaB, c, aaBaaB, aaBc, caaB, cc, \dots\}$  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  $\epsilon$  and the strings  $s$  defined over  $\Sigma$  such that

$Rev(s)=s$ .

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

Example

Notes Made by Vu-Gateway. . .

Be A part of Vu-Gateway by hitting the link. . .

<https://chat.whatsapp.com/Gecrau2Nit0D4F5DjDiGM>

for any info or help feel free cntct: +92 305 5868956

For  $\{$   
 $\} = \{a, b\}$ ,

PALINDROME =  $\{ \dots, a, b, aa, bb, aaa, aba, bab, bbb, \dots \}$  All two length string cannot satisfied the palindrome. aa and bb in palindrome but ba and ab are not in palindrome.

Question: What are the steps of Recursive Definition of

Answer: Languages? 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". Its means all string

Answer: ending in a e.g.  $\{ \} = \{a, b\}$   
 $\{a, aa, ba, aba, baa, \dots\}$   
Exactly a, defined over  $\{ \} =$   
 $\{a, b\}$   
 $\{ab, ba, abb, bba, \dots\}$

What is Lexical Analyzer?

Question: The first phase of the compiler is the lexical analyzer, also known as the scanner,

Answer: 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  $vu = bbabaab$ . 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

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  $(10^2 - 10^{-3} - 25.639) = 99.9999 - 25.63967 = 74.360$

Notes Made by Vu-Gateway. . .

Be A part of Vu-Gateway by hitting the link. . .

<https://chat.whatsapp.com/Gec1rau2Nit0D4F5DjDiGM>

for any info or help feel free cntct: +92 305 5868956

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  $(x) + \text{length}(y)$  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.

What is pumping lemma? And what is history?

Question: 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)

Answer:

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.

Notes Made by Vu-Gateway. . .

Be A part of Vu-Gateway by hitting the link. . .

<https://chat.whatsapp.com/Gec1rau2Nit0D4F5DjDiGM>

for any info or help feel free cntct: +92 305 5868956

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  $\rightarrow L$  is said to be **null production**.

**Example:**

Consider the following CFG

$S \rightarrow aA|bB|L, A \rightarrow aa|L, B \rightarrow aS$

Here  $S \rightarrow L$  and  $A \rightarrow L$  are null productions.

A production is called **nullable production** if it is of the form  $N \rightarrow L$

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

$N_1 \rightarrow N_2, N_2 \rightarrow N_3, N_3 \rightarrow N_4, \dots, N_n \rightarrow L$ , where  $N, N_1, N_2, \dots, N_n$  are non terminals.

**Example:**

Consider the following CFG  $S \rightarrow$

$XY, X \rightarrow Zb, Y \rightarrow bW$

$Z \rightarrow AB, W \rightarrow Z, A \rightarrow aA|bA|L, B \rightarrow Ba|Bb|L.$

Here  $A \rightarrow L$  and  $B \rightarrow L$  are null productions, while  $Z \rightarrow AB, W \rightarrow Z$  are nullable productions.

**Method:**

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

Consider the following productions of a certain CFG  $X \rightarrow aNbNa, N \rightarrow L$ , delete the production

$N \rightarrow L$  and using the production  $X \rightarrow aNbNa$ , add the following new productions  $X \rightarrow aNba, X \rightarrow abNa$  and  $X \rightarrow aba$

Thus the new CFG will contain the following productions  $X^{\circ}$

$Nba \mid abNa \mid aba \mid aNbNa$

**Note:** It is to be noted that  $X^{\circ} aNbNa$  will still be included in the new CFG. **Method:**

Consider the

following CFG  $S^{\circ}$

$XY, X^{\circ} Zb, Y^{\circ} bW$

$Z^{\circ} AB, W^{\circ} Z, A^{\circ} aA \mid bA \mid LB$

$^{\circ}Ba \mid Bb \mid L$ .

Here  $A^{\circ} L$  and  $B^{\circ} L$  are null productions, while  $Z^{\circ} AB, W^{\circ} Z$  are nullable productions. The new CFG after, applying the method, will be

$S^{\circ} XY$

$X^{\circ}$

$Zb \mid bY$

$^{\circ}bW \mid b$

$Z^{\circ}$

$AB \mid A \mid B W$

$^{\circ}Z A^{\circ}$

$aA \mid a \mid b$

$A \mid b B$

$^{\circ}Ba \mid a \mid Bb \mid b$

**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.

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

Answer: Example 1

If  $s=abc$  is a string defined over  $\Sigma =$

$\{a,b,c\}$  then  $\text{Rev}(s)$  or  $sr = cba$

$\Sigma =$

$\{a,$

$b\}$

$s=a$

$bb$

$aa$

$\text{Rev}(s)=aabba$

When more than one 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

$\Sigma = \{B, aB, bab,$

$d\}$

$s=BaBbabBd$

$\text{Rev}(s)=dBbab$

$a$

BB Example 3

$\Sigma = \{ab, b,$

$aa\}$

$s=abbaa$

$\text{Rev}(s)=aa$

$b ab$

$\Sigma_1 = \{B, aB, bab, d\}$  is valid alphabet as there is no letter in  $\Sigma_1$  that lies in start

of any other letter means all the tokens of any word (string) will be unique.

Whereas in  $\Sigma_2 = \{B, Ba, bab, d\}$  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  $\Sigma = \{a, b\}$

The language L of strings of even length, defined over  $\Sigma = \{a, b\}$ , 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  $\Sigma = \{a, b\}$ , 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

$L = \text{language}$

$((a+b)(a+b)(a+b))$  Or

$L = \text{language } ((a+b)^3)$

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  $\Sigma = \{a, b\}$  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  
)  
\*  
g  
i  
v  
e  
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 than one plus sign the resulting string is only one from them or it can be more than 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  $r_1 = (a + b)^* (aa + bb)$

$r_2 = (a + b)^* aa + (a + b)^* bb$  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  $r_2$  generates the language of strings ending either  $aa$  or  $bb$  this is equal to  $r_1$ .

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  $\{\text{null}\}$ , a one-word language.

Rule 2: If  $r_1$  is a regular expression associated with the language  $L_1$  and  $r_2$  is regular expression associated with the language  $L_2$ , then:

The regular expression  $(r_1 r_2)$  is associated with the product  $L_1 L_2$  that is the language  $L_1$  times  $L_2$  language  $(r_1 r_2) = L_1 L_2$

The regular expression  $r_1 + r_2$  is associated with the language formed by the union of the sets  $L_1$  and  $L_2$ : language  $(r_1 + r_2) = L_1 + L_2$

The language associated with the regular expression  $(r_1)^*$  is  $L_1^*$ , the Kleene closure of the set  $L_1$  as a set of words: language  $(r_1)^* = L_1^*$

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 count of b's is also even.

Even + Even = Even (Proved)

So we can divide any string excluding  $\epsilon$  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

- 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 than 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.

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:  $|0010| = 4$ ,  $|aa| = 2$ ,  $|\epsilon| = 0$

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. 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  $\dots$  and the strings  $s$  defined over  $\Sigma$  such that  $\text{Rev}(s)=s$ .

What is Equal RE?

Sometimes we generate many REs for one language these are called Equal RE. How can we make Finite Automaton from a language and a language from FA?

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  $\Sigma = \{0,1\}$ , 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  
Join Us on <http://vustudents.ning.com/>

If  $\Sigma = \{0, 1\}$ , 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  $\{0, 1, 00, 01, 10, 11, \dots\}$ . Minimum words which are included in this language are 0 and 1.

2. Consider the language L of strings, defined over

If  $\Sigma = \{0, 1\}$ , 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.

Notes Made by Vu-Gateway. . .

Be A part of Vu-Gateway by hitting the link. . .

<https://chat.whatsapp.com/Gc1rau2Nit0D4F5DjDiGM>

for any info or help feel free cntct: +92 305 5868956

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

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.