1 Fill in the Blanks

1. Grammar consists of four touplesSet of non-terminals.....set of production rule, and .....

2. According to the Chomsky hierarchy, there are .....types of grammars.

3. Type 1 grammar is called.....

4. Type 2 grammar is called.....

5. According to the Chomsky hierarchy, regular grammar is type.....grammar.

6. All languages are accepted by.....

7. The machine format of context-free language is.....

8. Linear bounded automata is the machine format of .....

9. The machine format of type 3 language is.....

10. Grammar where production rules are in the format is..... grammar

11. In a context-free grammar at the left hand side, there is.....non-terminal.

12. Type 3 language is called.....

13. an bn cn is an example of.....language in particular.

14. The grammar S $\rightarrow$ aSb/A, A $\rightarrow$Ac/c is an example of .....grammar in particular.

15. The grammar S $\rightarrow$ Abc/ABSc, BA $\rightarrow$ AB, Bb $\rightarrow$ bb, A $\rightarrow$ a is an example of grammar.....in particular.

16. The grammar A $\rightarrow$ aA/bB/a/b, B $\rightarrow$ bB/b is an example of.....grammar in particular.

17. The language a*(a + b)b* is an example of .....language in particular.

2 Answer the question above sentence

1. set of terminals, start symbol

2. Four

3. Context-sensitive grammar

4. Context-free grammar

5. Three 6. Turing machine

7. Push down automata

8. Context-sensitive grammar

9. Finite automata

10. Context-sensitive

11. Single

12. Regular expression

13. Context-sensitive

14. Context-free

15. Context-sensitive

16. Regular grammar

17. Type 4

3 Define grammar. Is any grammar possible without a start symbol?

4 Is it possible to guess the type of language from a given grammar? Is the reverse true? Give reasons in support of your answer

5 3. What is the Chomsky hierarchy? What is the usefulness of it?

6 4. Find the languages generated by the following grammars.

a) S → aSb/A, A →Ac/c
b) S → aSb/aAb, A → Ac/e
c) S → aSb/aAb, A → bA/b
d) S → S1/S2, S1 → 0S11/0A, A → 0A/, S2 → 0S21/B1, B → B1/e
e) S → AB/CD, A → aA/a, B → bB/bC, C→ cD/d, D → aD/AD
Justify your answer for this.
f) S → AA, A → BS, A → b, B → SA, B→ a
g) E → E + E— E menhha E— E * E— E/E—id

7 Construct a grammar for the following languages.

a) L = tohi
b) L = (a, b)*, where all 'a' appears before 'b'
c) L = (a, b)*, where all 'b' appears before 'a'
d) L = (a, b)*, where there are equal number of 'a' and 'b'
e) L = (a, b)*, where ab and ba appear in an alternating sequence.
f) L = (a, b)*, where the number of 'b' is one more than the number of 'a'
g) L = (a, b)*aa(a, b)*

8 Construct a grammar for the following languages.

a) L = ambn , where m not equal to n.
b) L = axbycz, where y = x + z
c) L = axbycz, where z = x + y
d) L = axbycz, where x = y + z

e) L = Set of all string over a, b containing aa or bb as substring
f) L = Set of all string over a, b containing at least two ‘a’
g) L = Set of all string over 0, 1 containing 011 as substring

9 Construct a grammar for the following languages.

a) anbn— n 0 u cmdm — m0
b) anbn— n 0 u ambm — m0
c) axbycz, where x = y + z u L = axbycz, where z = x + y

10 Construct a grammar for the following languages and fi nd the type of the grammar in particular.

a) L = (0 + 1)* 11 (11)*
b) L = (Set of all string of ‘a’, ‘b’ beginning and ending with ‘a’)
c) L = a2n + 1, where n ¿ 0

11 Finite Automata

12 Introduction

The term ‘automata’ has some typical pronounceable symmetry with the term ‘automatic’. In a computer, all pro cesses appear to be done automatically. A given input is processed in the CPU, and an output is generated. We are not concerned about the internal operation in the CPU, but only about the given input and the received output. However, in reality, the input is converted to ‘0’ and ‘1’ and assigned to the process for which the input was given. It then performs some internal operation in its electronic circuit and generates the output in ‘0’ and ‘1’ format. The output generated in the ‘0’ and ‘1’ format is converted to a user-understandable format and we get the output. From the discussion, it is clear that the CPU performs machine operations internally. In automata, we shall learn about how to design such machines. To get a clear idea about the automata theory, we need to travel back to the history of computer science. The engineering branch ‘electrical engineering’ stands on the base of physics. Electronics engineering came from electrical engineering. The hardware of a computer is a complex electronics circuit. The circuit understands only binary command, which is impossible for a human user to memorize. Then, mathematics extends its hand for developing the logic. Mathematics and electronics have jointly given birth to a new branch of engineering called ‘computer science and engineering’.

13

The name of the subject is formal language and automata theory. We have a basic idea about the automata theory. Now what is formal language? Let us discuss what language is. Language is a communication medium through which two persons communicate. For every nation, there is some language to communicate such as Hindi, English, Bengali, etc. It is the same for communicating with a computer, wherein the user needs some language called the programming language. C, C ++, and Java are some examples of programming languages. The characteristics of these types of languages are that they are similar to the English language and are easily understandable by the user. However, the computer does not understand English-like statements. It understands only binary numbers. Therefore, the computers have a compiler that checks the syntax and acts as a converter between English-like statements to binary numbers and vice versa. But, to design the compiler, some logic is needed. That logic can be designed using mathematics. For each language, there is a grammar. Grammar is the constructor for any language. Similarly, the languages that are used for computer programming purpose have some grammar to construct them. These rules and grammar and the process of converting these grammar and languages to machine format are the main objectives of this subject.

14 History of the Automata Theory

Theoretical computer science or automata theory is the study of abstract machines and the computational problems related to these abstract machines. These abstract machines are called automata. The origin of the word 'automata' is from a Greek word which means that something is doing something by itself. Around 1936, when there were no computers, Alen Turing proposed a model of abstract machine called the Turing machine which could perform any computational process carried out by the present day's computer. A formal version of the fi nite automata model was proposed in 1943 in the McCulloch–Pitts neural network models. Finite automata and fi nite state machines got a complete form by the efforts of George H. Mealy at the Bell Labs and Edward F. Moore in IBM around 1960. The machines are named as Mealy and Moore machines, respectively. In mid-1950s, Noam Chomsky at the Harvard University started working on formal languages and grammars. Grammar is a set of rules that are performed to produce sentences in the language. Chomsky structured a hierarchy of formal languages based on the properties of the grammar required to generate the languages. The Chomsky hierarchy drives the automata theory one step forward. In 1971, Stephen Arthur Cook in his research paper 'The Complexity of Theorem Proving Procedures' formalized the notions of polynomial-time reduction and NP-completeness. He separated those problems that can be solved effi ciently by the computer from the set of problems that can be solved but with so much time-taking. These second classes of problems are called NP-hard. This subject is sometimes called the

'Theory of Computation' because it includes all of these rules for constructing a computer language and converts them into machine format. That is the theory of computer science. Basically formal language and automata theory and the theory of computation are different names for a single subject that covers all the aspects of the theoretical part of computer science.

15 Use of Automata

It is unfruitful to study a subject without knowing its usefulness. The subject 'automata theory' is called one of the core subjects of computer science. Some of the uses of this subject in different fi elds of computer science and engineering are as follows:

number1: The main usefulness of the automata theory is in the compiler design. The fi rst phase of compiler called lexical analyser is designed by the fi nite automata. In syntax analysis part, a common error called syntax error is checked by the rules of context-free grammar.

number2: The working principle of the software used for checking the behaviour of digital circuits is based on the automata theory.

number3: The software for natural language processing (NLP) takes the help of the automata theory. Automata is useful for designing the software for counting the occurrence of a particular word, pattern, etc., in a large text.

16 Characteristics of Automaton

The word 'automata' is plural and the singular is 'automaton'. A Finite Automaton, in short FA, provides the simplest model of a computing device. It is called fi nite because it has a fi nite number of states. The state of the system memorizes the information concerning the past input. It is necessary to determine the future behaviour of the system.

17 Finish Ostad razavi thanks for watching.