**THEORY OF AUTOMATA**

**ASSIGNMENT #4**

Aaiza Irfan (SP20-BCS-001)

Aliza Tanweer (SP20-BCS-013)

Kulsoom Kurshid (SP20-BCS-044)

**Question #1:**

**Chomsky Hierarchy**:

Chomsky Hierarchy represents the languages of a class that are accepted by different machines. The languages in Chomsky Hierarchy are categorized into following types.

1. Type 0 represents the Unrestricted Grammar.
2. Type 1 represents the Context Sensitive Grammar.
3. Type 2 represent the Context Free Grammar.
4. Type 3 represents the Regular Grammar.

Any language belonging to type 3 also belongs to type 2, 1 and 0. Similarly, any language of type 2 is also of type 1 and 0. Therefore, this makes a hierarchy.

**Type 0 Grammar:**

Type 0 also known as Unrestricted Grammar, which means that there are no restricted rules for these languages. Turing Machines can be used to efficiently model these languages.

For example:

bAa 🡪 aa

S 🡪 s

**Type 1 Grammar:**

Type 1 also known as Context Sensitive Grammar. This grammar is used to represent Context Sensitive language. Following rules are used for CSG.

* It may have more than one symbol on the left side of their production rules.
* The number of symbol on the left side should not exceed the number of symbol on the right side.
* The rule of the form A → ε is not allowed unless A is a start symbol. It does not occur on the right-hand side of any rule.
* Type 1 grammar should be type 0. In type 1, Production is in the form of V 🡪 T

For example:

S → AT

T → xy

A → a

**Type 2 Grammar:**

Context Free Grammar is used to represent the Context Free languages. Type 2 is also type 1.

A → α

A is a single non-terminal and is any combination of terminals and non-terminals.

For example:

A → aBb

A → b

B → a

**Type 3 Grammar:**

Regular languages can be described by any regular expression that can be modeled by NFA and DFA. It is the most restricted form of grammar. Type 3 should be in the form of

V → T\*V / T\*

For example:

A → xy

**Question #2:**

**Regular Language:**

A regular language is a language that can be expressed with a regular expression or a deterministic or a non-deterministic finite automata or state machine.

* Is regular if it can be expressed in terms of regular expression.
* Regular language are subset of the set of all the strings.
* Used in parsing and designing programming languages.

Language:

It is a set of strings which are made up of characters from a specified alphabet or set of symbols.

Rules for an instance:

* X\*  means zero or more occurrence of x. Example: {e,xx,xxx,xxxx,….}
* X+  means one or more occurrence of x. Example: {x,xx,xxx,xxxx,….}

Operations on Regular Language:

Following are the operations on regular language are:

**Union:** If L and M are two regular languages then their union L U M is also a union.

L U M = {s | s is in L or s is in M}

**Intersection:** If L and M are two regular languages then their intersection is also an intersection.

L ⋂ M = {st | s is in L and t is in M}

**Kleen closure:** If L is a regular language then its Kleen closure L1\* will also be a regular language.

L\* = Zero or more occurrence of language L.

Examples of Language:

**Example 1 –** All strings of length = 2 over {a, b}\* i.e. L = {aa, ab, ba, bb} is regular.

**Example 2–** The language over the alphabet {0,1} where strings contain an even number of 0’s can be constructed by = (1\* ((01\* )(01\* ))\* ).

**Context free language:**

Context free language is generated by context free grammar and is accepted by pushdown automata

Grammar:

A context-free grammar is a 4-tuple (V, T, S, P) where

1. V is a finite set called the variables
2. T is a finite set, disjoint from V, called the terminals
3. P is a finite set of rules, with each rule being a variable and a string of variables and terminals, and
4. S ∈V is the start variable.

If u, v and w are strings of variables and terminals, and A → w is a rule of the grammar,

we say that uAv will result in uwv, written uAv ⇒ uwv.

Example of CFG:

Given a grammar G = ({S}, {a, b}, R, S).

The grammar G = ({S}, {a, b}, S, P) with productions

S 🡪 aSa

S 🡪 bSb

S → λ

is context free.

S 🡪 aSa

🡪 aaSaa

🡪 aabSbaa

🡪aabbaa

CFG closure properties:

**Union** : If L1 and L2 are two context free languages, their union L1 ∪ L2 will also be context free

**Concatenation** : If L1 and If L2 are two context free languages, their concatenation L1.L2 will also be context free

**Kleene Closure** : If L1 is context free, its Kleene closure L1\* will also be context free

**Context sensitive language**

Context free language is generated by context sensitive grammar and is accepted by linear bounded automata.

Grammar:

A context-sensitive grammar is one whose productions are all of the form

xAy→ xvy

where A ∈v and x, v, y (V T ) \* ∈ ∪

“Context-sensitive” implies the fact that the actual string modification is given by A → v, while the x and y provide the context in which the rule may be applied.

This type of grammar is called “Non-contracting” as the derivation steps never decrease the length of the sentential form

Example of CSG:

The language L={anbncn|n≥1} is a context-sensitive language.

A kind of grammar is

S → abc|aAbc, Ab→ bA,  
Ac→ Bbcc,

bB → Bb,  
aB → aa | aa A

Let us see how this works by looking at the derivation of a 3 b3 c3 .

S ⇒ aAbc ⇒ abAc ⇒ abBbcc  
⇒ aBbbcc ⇒ aaAbbcc ⇒ aabAbcc ⇒ aabbAcc ⇒ aabbBbccc  
⇒ aabBbbccc ⇒ aaBbbbccc  
⇒ aaabbbccc.

This uses the variables A and B. Since the language is not context-free, it is said to be context-sensitive language.

Closure properties:  
Context sensitive languages are closed under

* Union
* Intersection
* Complement
* Concatenation
* Kleene star

**Recursively Enumerable Languages**

Any language accepted by a Turing machine is called recursively enumerable.

A Turing machine accepted a language may loop forever on some inputs.

Thus accepted guarantees that if a string is in the language the corresponding TM will halt on that string but it does not guarantee that the TM will crash on all other inputs; it may loop.

Example:

H is a r.e. language. a\* is a r.e. language.