CHAPTER

1

Introduction

Computational Problems:

In theoretical computer science, a computational problem is a mathematical object representing a collection of questions that computers might want to solve. For example, the problem of factoring "Given a positive integer n, find a nontrivial prime factor of n." is a computational problem. Computational problems are one of the main objects of study in theoretical computer science. The field of algorithms studies methods of solving computational problems efficiently. The complementary field of computational complexity attempts to explain why certain computational problems are intractable for computers

Types of computational problems:

A decision problem is a computational problem where the answer for every instance is either yes or no. An example of a decision problem is primality testing: "Given a positive integer n, determine if n is prime."

A **decision problem** is typically represented as the set of all instances for which the answer is yes. For example, primality testing can be represented as the infinite set L = {2, 3, 5, 7, 11, ...}.

In a **search problem**, the answers can be arbitrary strings. For example, factoring is a search problem where the instances are (string representations of) positive integers and the solutions are (string representations of) collections of primes.

A search problem is represented as a relation over consisting of all the instance-solution pairs, called a search relation. For example, primality can be represented as the relation

$$R = \{(4, 2), (6, 2), (6, 3), (8, 2), (8, 4), (9, 3), ...\}$$

which consist of all pairs of numbers (n, p), where p is a nontrivial prime factor of n.

A **counting problem** asks for the number of solutions to a given search problem. For example, the counting problem associated with primality is "Given a positive integer n, count the number of nontrivial prime factors of n."

An **optimization problem** asks for finding the "best possible" solution among the set of all possible solutions to a search problem. One example is the maximum independent set problem: "Given a graph G, find an independent set of G of maximum size."

Optimization problems can be represented by their search relations.

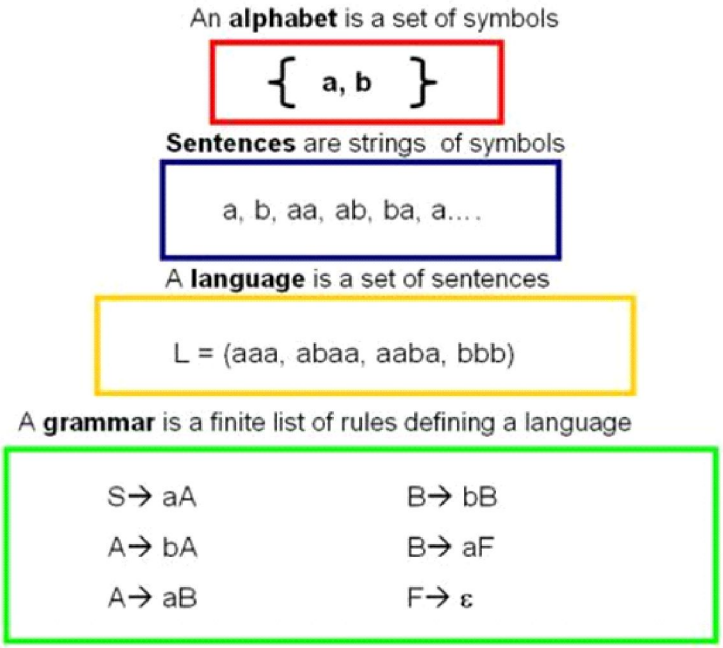
Computational models

In computability theory and computational complexity theory, a **model of computation** is the definition of the set of allowable operations used in computation and their respective costs. It is used for measuring the complexity of an algorithm in *execution time* and or *memory space*: by assuming a certain model of computation, it is possible to analyze the computational resources required or to discuss the limitations of algorithms or computers. Some examples of models include **Turing machines**, **recursive functions**, **lambda calculus**, and **production systems**.

Language :

A **language** is a set of words, i.e. finite strings of letters, symbols, or tokens. The set from which these letters are taken is called the **alphabet** over which the language is defined. A formal language is often defined by means of a **formal grammar** (also called its formation rules); accordingly, words that belong to a formal language are sometimes called **well-formed formulas**. The field of formal language theory studies purely syntactical aspects of language (i.e. their internal structural patterns).

A hierarchy is defined for any language(or formal language)



Well formed formula:

A mathematical logic a well formed formula. This is a part of formal languages.

A well formed formula is defined recursively as

(a) If P is a proportional variable then it is a well formed.

(b) If α is a well formed then Σa is a well formed.

(c) If α and β are well formed formula's then $(\alpha \cup \beta)$, $(\alpha \cap \beta)$, $(\alpha \Rightarrow \beta)$ and $(\alpha \Leftrightarrow \beta)$ are well formed formula.

(d) A string of symbols is a well formed if and only if it is obtained by finite number of applications of above three cases.

e.g. $\alpha = (P \vee Q) \wedge (P \Rightarrow Q) \wedge (Q \Rightarrow P)$

$P \Rightarrow Q$ is P if and only if Q

Truth table for $P \Rightarrow Q$

P	Q	$P \Rightarrow Q$
T	T	T
T	F	F
F	T	F
F	F	T

Terminal symbols: One denoted by small letters/digits and special symbols

(a, b, c, x, y .....) (0, 1, 2, ...), (#, @, θ .....)

Non-terminal symbols: Denoted by capital letters.
(A, B, S, ...)

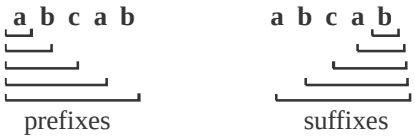
String: An alphabet is a non-empty finite set of symbols deonted by Σ

e.g. $\Sigma = \{a, b, c\}$ is an alphabet

A string is a finite sequence of symbols. Strings are usually deonted by u, v and w.

e.g. U = abcab is a string on $\Sigma = \{a, b, c\}$

The empty string (no symbols at all) is denoted λ
A part of a string is a substring.
e.g. bca is a substring of abcab.
A begininning of a string (up to any symbol) is a prefix, and an ending is a suffix.



Note: A string is a prefix and suffix of itself. λ is a prefix and suffix of any string.

Operation on the springs/words.

1. **Finding the length:** $\Sigma = \{a, b\}$

   $\therefore \quad w = abba$

   $\therefore \quad |w| = 4$

2. **Concatenation:** wv – appending v to the end of w.

   e.g. w = abc, v = ab; $wv = \underbrace{abc}_{w}\ \underbrace{ab}_{v}$

   Note: $\lambda w = w\lambda = w$

3. **Power :** $w^0 = \lambda$ (null string)

   $w' = w \Rightarrow w^2 = ww \Rightarrow w^3 = w \cdot w^2 = w \cdot w \cdot w$

   $w^n = w \cdot w^{n-1} = w \cdot w \cdot w \ldots\ldots (\text{n times})$

e.g. $|w^n| = ?$

   $n * |w|, \ w^n = |w| * |w| * |w| \ldots\ldots n$ times

4. **Reverse**: $w^R \rightarrow w$ in reverse order

e.g. w = abc, w^R = cba

5. **Palindrome :** $|w| = |w^R|$

   The word and its reversal have same value.

   $\langle w = aba \rangle$
   $\langle w^R = aba \rangle$

   Even Palindrome:

   (i) $w = w^R$ (ii) $|w|$ is even

   Odd Palindrome:

   (i) $w = w^R$ (ii) $|w|$ is odd

e.g. $\lambda^R = \lambda (\text{null}) \rightarrow 0$ is even palindrome

   $a^R = a \Rightarrow \text{odd}\ (1) \rightarrow$ odd palindrome

e.g.Number of palindrome of length 8 over $\Sigma = (0,1)$

2 ways: 2 2 / – – –

$\therefore \quad 2^4 = 16$

Length $\Sigma\{0,1\}$:

2	2	2	2	2.....
(0, 1)	(0, 1)	(0,1)	(0, 1)	(0, 1)

$\therefore \quad 2^5 = 32$

Number of palindromes of length n over Σ with $\Sigma = k$ is $\boxed{k^{\lceil n/2 \rceil}}$

e.g. n = 9, $\Sigma = \{0,1\} \therefore \quad 2^{[9/2]} = 2^5 = 32$

6. Kleen Star/Kleen's Closure:

If $\Sigma = \langle a, b \rangle$

Σ^* = the set of all string which can be conducted by using the symbols from Σ including λ.

Therefore, e.g. $\Sigma = \{a,b\} \Rightarrow \Sigma^* \{\lambda, a, b, aa, bb, ba, aaa, bbb, aba, aab, baa.....\}$

(it is a universal language)

e.g. $\Sigma = \{a\}$

$a^* = \{a^*\} = \Sigma^* = \{\lambda, a, a^2, ...\}$

7. Kleen Plus/Positive Closure (Σ^+) : The set of all string which can be constructed using the symbols of Σ excluding λ

$\therefore \quad \Sigma^+ = \{a, b, aa, bb, ab, ba, aaa.....\}$

$\therefore \quad \boxed{\Sigma^* - \Sigma^+ = \{\lambda\}}$

We can say,

$\Rightarrow \left[\Sigma^+ \cup \{\lambda\} = \Sigma^*\right] \Rightarrow \left[\Sigma^* \cup \{\lambda\} = \Sigma^*\right]$

Formal Langauge/Language (L):

Collection of strings of Σ, usually denoted by *L*. It may be finite or infinite.

e.g. $\Sigma = \{a, b\}$

$\left[L = \{\lambda, ab, bba, bab\}\right] \rightarrow$ finite language

$\Sigma^* = \{\lambda, a, b, aa, bb, ab, ba, aaa.....\} \rightarrow$ universal language/infinite language

$\therefore \quad \boxed{L \subset \Sigma^*}$, L is the subset of Σ^*

Operations on Language:

(i) Finding the cardinality:

e.g. (a) $L = \{\lambda, ab, ba\} \quad \therefore |L| = 3 \rightarrow$ cardinality

(b) $L = \{\lambda\}, \Rightarrow |L| = 1$

(c) $L = \{\ \} \quad \therefore |L| = 0 = \phi$

(ii) Reversal of a Language:

$L^R = \{w^R \mid w \in L\}$

e.g. $L = \{\lambda, ab, bba\} \Rightarrow L^R = \{(\lambda)^R, (ab)^R, (bba)^R\}$

$\therefore \quad L^R = \{\lambda, ba, abb\}$

$\therefore \quad \boxed{|L| = |L^R|} \rightarrow$ equal cardinality but if $(L = L^R) \rightarrow \alpha$

[It will be possible only in the case of palindrome string]

(iii) Concatenation:

$L_1 \cdot L_2 = \{w_1 w_2 \mid w_1 \in L_1 \text{ and } w_2 \in L_2\}$

Or, $L_2 \cdot L_1 = \{w_2 w_1 \mid w_2 \in L_2 \text{ and } w_1 \in L_1\}$

$L_1 = \{a, b, ba\} \Rightarrow L_2 = \{abb, bb\}$

$L_1 L_2 = \{ab, abbb, abbb, baabb, babb\}$

$L_2 L_1 = \{abbab, abbba, bbab, bbba\}$

Note: $L_1 L_2 = L_2 L_1$ *(its not true)

It will be true only when $\boxed{(L_1 = L_2)}$

(iv) Union:

$L_1 \cup L_2 = \{w \mid w \in L, \textbf{ OR } w \in L_2\}$

(v) Intersection:

$L_1 \cap L_2 = \{w \mid w \in L_1 \textbf{ AND } w \in L_2\}$

(vi) Difference :

$L_1 \cap L_2 = \{w \mid w \in L_1 \textbf{ AND } w \in L_2\}$

$L_2 - L_1 = \{w \mid w \in L_2 \textbf{ AND } w \notin L_1\}$

(vii) Symmetric difference:

$L_1 \oplus L_2 = (L_1 - L_2) \cup (L_2 - L_1)$

(XOR operation)

Note:

Union, intersection, difference, symmetry are binary operators.

Example : $L_1 = (ab, ba, abb)$, $L_2 = \{\lambda, ba, bbb\}$

Union: $L_1 \cup L_2 = \{ab, ba, abb, \lambda, bbb\}$

Intersection: $L_1 \cap L_2 = \{ba\}$

Difference: $L_1 - L_2 = \{ab, abb\}$; $L_2 - L_1 = \{\lambda, bbb\}$

Symmetric difference : $L_1 \oplus L_2 = \{\lambda, ab, abb, bbb\}$

Note: $\boxed{|L_1 \cup L_2| \leq |L_1| + |L_2|}$ {equal (when nothing is common) < (when something is common)}

$\boxed{|L_1 \cap L_2| \leq |L_1| + |L_2|}$ {equal (when there is ϕ, empty string) < (when there is common)}

(viii) Power of Language:

$L^0 = \{\lambda\}$	$w^0 = \lambda$
$L^1 = L$	$w^1 = w$
$L^2 = L \cdot L$	$\vdots$
$L^3 = L \cdot L^2 = L \cdot L \cdot L$	$\vdots$
$\vdots$	$\vdots$
$L^n = L \cdot L^{n-1} = L.L.L....(\text{n times})$	$w^n = w \cdot w^{n-1}$

Note: $L^0 = \{\lambda\}$ (it's a langauge which contains a null string) $|L^0| = 1$

$w^0 = \lambda$ (its a null string) $\{|w^0| = |\lambda| = 0\}$

(ix) Complement of a language:

$\overline{L}$ or $L^c = \{w \mid w \notin L \text{ w.r.t. } \Sigma^*\} \Rightarrow [\overline{L} \text{ or } L^c = \Sigma^* - L]$

e.g. $L = 0^* = \{\lambda, 0, 0^2, 0^3, ...\}$, $\Sigma = \{0,1\}$

$L^c = \Sigma^* - L \Rightarrow \{0,1\}^* - 0^* \Rightarrow \{\lambda, 0,1, 00, 01, 10, 11, 111, 000, ...\} - \{\lambda, 0, 0^2, 0^3, ...\}$

$[\therefore \quad L^c = \{1, 01, 10, 111, ...\}]$

The set of all strings which contains atleats one '1'.

Problem: If $L = \{a^n b^n \mid n \geq 0\}$, then $L^2 = ?$

Soln. $L^2 = L \cdot L$

$L = \{a^n b^n\} \Rightarrow L\{\lambda, ab, a^2b^2, a^3b^3, ...\}$

(i) $\{a^n b^n a^n b^n \mid n \geq 0\}$, $L^2 = \{\lambda, ab, a^2b^2, a^3b^3, ...\} \cdot \{\lambda, ab, a^2b^2, a^3b^3, ...\}$

(ii) $\{a^{2n} b^{2n} \mid n \geq 0\} = \{\lambda, a^2b^2, a^4b^4, a^6b^6, ...\}$

Problem: If $L = \{a^n \mid n \geq 0\}$ then $L^2 = ?$

Soln. $L^2 = L \cdot L$

$L = \{\lambda, a, a^2, a^3.....\}$

$L^2 = \{\lambda, a, a^2, a^3.....\} \cdot \{\lambda, a, a^2, a^3.....\}$

$= \{\lambda, a, aa, a \cdot a^2, a \cdot a^3, a^2a, a^2a^2, a^2a^3, ...\}$

$L^2 = \{\lambda, a, a^2, a^3, a^4, ...\} \quad \therefore \{a^n \mid n \geq 0\}$

(x) Prefix (L) = $\{\text{Prefix}(w) \mid w \in L\}$

(xi) Suffix (L) = $\{\text{suffix}(w) \mid w \in L\}$

Example : $w = xy$ e.g. $w = 011$

We can write it as $\rightarrow w = \lambda.011$

Problem: How many prefix/suffix of a string w of length 'n' will be

Soln. $(n+1)$

(xii) Right Quotent/Right Derivative of a language:

$$\frac{dL}{da} = \frac{L}{a} = \{w \mid wa \in L\}$$

(We will consider only those strings which ends with a) $\rightarrow$ so remove such kind.

Problem: $L = \{a, ab, bba, babaa\} \left[\frac{dL}{d(aa)} = \{bab\}\right]$

Soln. $\left[\frac{dL}{da} = \{\lambda, bb, baba\}\right]$

Similarly, $\frac{dL}{db} = \{a\}$

(xii) Left Quotient:

$L \setminus a$ (left derivative with respect to a)

$\therefore\ L \setminus a = \{w \mid aw \in L\}$ (consider only those which starts with a)

e.g. $L = \{ab, bb, abba\}$; $L \setminus a = \{b, bba\}$

Note:

(i) Kleen star of a language:

$L^* = L^0 \cup L^1 \cup L^2 \cup L^3$

$\boxed{L^* = \bigcup_{i=0}^{\infty} L^i}$

(ii) Kleen Plus of a language :

$L^+ = L^1 \cup L^2 \cup L^3$

$\boxed{L^+ = \bigcup_{i=1}^{\infty} L^i}$; $\boxed{L^* - L^+ = L^0}$

Grammer :

- Grammer is a set of rules helps to generate valid strings with respect to an alphabet.
- A grammer is a four tuple system defined as

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

Where, V = the set of variables/non-terminals (finite)
T = the set of terminals (finite)
$S \in V$ is a start symbol.
P = the set of production rules (finite), helps to generate valid string with respect to S.

The form of the rules:

$X \to Y$

$X \in (V \cup T)^+ \quad Y \in (V \cup T)^*$

The value of Y can be null.

Example: $G : (V = \{S\}, T = \{a\})$

$\mathrm{P} : (1)\ \mathrm{S} \to \lambda;\ (2)\ \mathrm{S} \to \mathrm{aS}$

Soln. $\mathrm{L(G)}$ = Language generated by the grammer G

$$\left\{ \mathrm{w} \in \mathrm{T}^* \mid \mathrm{S} \overset{*}{\Rightarrow} \mathrm{w} \right\}$$

Applying rules:

$S \to \lambda$ (S can be replaced by λ)

$S \to aS$ (*S* can be replaced by *aS*)

$S \xrightarrow{(1)} \lambda$

$S \xrightarrow{(2)} aS$

This is not the terminal string (only small letters induced).

Note: A language can be generated by one or more then one grammer.

Chomsky Classification of Languages

Grammer is basically defined as (V_N, Σ, P, S)

V_N = set of variables.
Σ = set of symbols.
P = production rule
S = starting state.

Chomsky classified the grammar into four in terms of productions.

- A type 0 grammer is any phase structure grammer without any restrictions.
- A production is defined as a form of

  $\phi A \psi \rightarrow \phi \alpha \psi$

  A = variable

  ϕ is called left context.

  ψ is called right context.

  In type-1 production, if $\alpha \neq \Lambda$ in type-I erasing of A is not permitted.
- A type-I grammer is also called context sensitive or context dependent is its all production are type-I production.
- In type-I product $\phi A \psi \rightarrow \phi \alpha \psi$ are not increase the length of the string (because $\alpha \neq \Lambda$)

  Therefore, $|\alpha| \leq |\beta|$ for $\alpha \rightarrow \beta$ production.

  This grammer is also called monotonic grammer.
- In type-II, grammer if it contains only type-I productions. It is also called context free grammer.
  A type-II, production of the form

  $A \rightarrow \alpha \qquad A \in V_N$

  $\alpha \in (V_N \cup \Sigma)^*$

  A language generated by type-II grammer is called context free languages.
- In type-III, or regular grammer if all its productions are type-III productions.
  $S \rightarrow \Lambda$ is allowed in type-III. But S should not be apear at right hand side of production.

SOLVED EXAMPLES

1. Write the grammer for the following language

(1) $L=\left\{a^n b^n \mid n \geq 0\right\}$ (2) $L=\left\{a^n b^n \mid n \geq 1\right\}$

(3) $L=\left\{a^n b^{n+1} \mid n \geq 0\right\}$ (4) $L=\left\{a^{n+1} b^n \mid n \geq 1\right\}$

(5) $L=\left\{a^n b^{2n} \mid n \geq 0\right\}$ (6) $L=\left\{a^n b^{2n+1} \mid n \geq 0\right\}$

(7) $L=\left\{a^{n+2} b^{2n} \mid n \geq 0\right\}$

Soln. (1) $S \rightarrow \lambda;\ S \rightarrow aSb$ (2) $S \rightarrow ab;\ S \rightarrow aSb$

(3) $S \rightarrow b;\ S \rightarrow aSb$ (4) $S \rightarrow aSb;\ S \rightarrow a^2 b$

$L=\left\{a^n b^{2n} \mid n \geq 0\right\}$ $L=\left\{a^n b^{2n+1} \mid n \geq 0\right\}$

(5) $S \rightarrow \lambda$ (6) $S \rightarrow b$

$S \rightarrow aSb^2$ $S \rightarrow aSb^2$

$L=\left\{a^{n+2} b^{2n} \mid n \geq 0\right\}$

(7) $S \rightarrow a^2$

$S \rightarrow aSb^2$

2. $L=\left\{a^n b^m \mid n>m\right\}$

Soln. $S \rightarrow a; S \rightarrow aS; S \rightarrow aSb$

3. $L=\left\{a^n b^m \mid n<m\right\}$

Soln. $S \rightarrow b; S \rightarrow Sb; S \rightarrow aSb$

4. $L=\left\{w \in\{a, b\}^* \mid N_a(w)=N_b(w)\right\}$

Soln. Suppose $w=ab$ and $N_a(w)=N_b(w)=1$

$S \rightarrow \lambda; S \rightarrow aSb; S \rightarrow bSa; S \rightarrow SS$

5. Set of all binary strings which contain atleast two 1 is

Soln. $\{0,1\}^* \mid\{0,1\}^* \mid\{0,1\}^*$

$S \rightarrow A1A1A;\quad A \rightarrow \lambda;\ A \rightarrow 0A;\ A \rightarrow 1A$

6. Set of all palindromes $\left(w=w^R\right)$ over $\Sigma=\{0,1\}$

Soln. (1) $S \rightarrow \lambda$ (2) $S \rightarrow 0$ (3) $S \rightarrow 1$ (4) $S \rightarrow 1S1$ (5) $S \rightarrow 0S0$

7. Set of Even palindromes over $\Sigma=\{0,1\}$

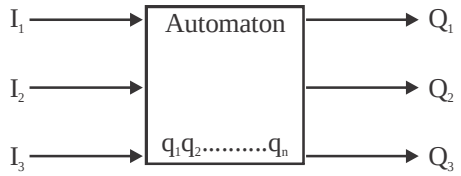
Soln. $S \rightarrow \lambda; S \rightarrow 1\lambda 1; S \rightarrow 0\lambda 0$

8. Set of odd palindromes over $\Sigma=\{0,1\}$

Soln. $S \rightarrow 0; S \rightarrow 1; S \rightarrow 0S0;\ S \rightarrow 1S1$

Automaton:

Automata Theory is an abstract model of computation. It is used for solving the computational problems. Automata theory is the study of self-operating virtual machine to help in logiccal understainding of input and output process with or without the intermediate stage of computation.



An automaton gets one input every time step that is picked up from a set of symbols or letters which is called an Alphabet and finite squence of symbols is called Word at each instance of time of run of input word. The automaton is in one of its state.

At each time step when the automaton reals the symbols of input word one after another and transitions from state to state according to the transition function.

Once the input word has been read the automation is said to have stopped and the state at which automata has stopped is called the final state.

Depending on final state it's said that automation either accepts or rejects an input word. Then as a subset of state's of the automation which is defined as the set of accepting states if the final state is an accepting state then the automata accepts the words otherwise the word is rejected. The set of all words which is accepted by automata is called the language recognized by the automation.

SOLVED PROBLEMS

1. How many substrings (of all lengths inclusive) can be formed a character string of length n? Assume all characters to be distinct. Prove your answer.

[GATE-1989]

Soln. Number of substrings (of all lengths inclusive) that can be formed from a character strings of length n is

$$\frac{n(n+1)}{2}+1.$$

Example: Let the length of string be 4, |w| = 4 and w = ABCD
Number of substrings of length 0 is 1(ε)
Number of substrings of length 1 is 4(A, B, C, D)
Number of substrings of length 2 is 3(AB, BC, CD)
Number of substrings of length 3 is 2(ABC, BCD)
Number of substrings of length 4 is 1(ABCD)
So, number of substrings of all length by a string of length 4 is Σ4 + 1

i.e., $\frac{4\times(4+1)}{2}+1=\frac{4\times5}{2}+1=10+1=11.$

2. The number of substrings (of all lengths inclusive) that can be formed from a character string of length n is

(a) n (b) n^2 (c) $\frac{n(n-1)}{2}$ (d) $\frac{n(n+1)}{2}+1$

[GATE-1994]

Ans. (d)

Soln. Number of substrings (of all lengths inclusive) that can be formed from a character strings of length n is

$$\frac{n(n+1)}{2}+1.$$

Example: Let the length of string be 4, |w| = 4 and w = ABCD
Number of substrings of length 0 is 1(ε)
Number of substrings of length 1 is 4(A, B, C, D)
Number of substrings of length 2 is 3(AB, BC, CD)
Number of substrings of length 3 is 2(ABC, BCD)
Number of substrings of length 4 is 1(ABCD)
So, total number of substrings of all length by a string of length 4 is Σ4 + 1

i.e., $\frac{4\times(4+1)}{2}+1=\frac{4\times5}{2}+1=11.$

3. Given $\Sigma = \{a, b\}$, which one of the following sets is not countable?

(a) Set of all strings over Σ
(b) Set of all languages over Σ
(c) Set of all regular languages over Σ
(d) Set of all languages over Σ accepted by Turing Machines

[GATE-1997]

Ans. (b)

Soln. (a) The set of all strings over Σ is Σ* which is countably infinite.
(b) Set of all languages over Σ is 2^{Σ^*}. According to Cantor's theorem if S be an countably infinite set, then its power set 2^S is uncountable.
So 2^{Σ^*} is uncountable because Σ* is countably infinite.
(c) Set of all regular languages over Σ is countably infinite.
(d) Set of all languages over Σ accepted by turing machine is the set of all RE languages which is countably infinite.

4. How many substrings of different lengths (non-zero) can be formed from a character string of length n?

(a) n (b) n^2 (c) 2^n (d) $\frac{n(n+1)}{2}$

[GATE-1998]

Ans. (d)

Soln. Number of substrings (of all lengths inclusive) that can be formed from a character strings of length n is $\frac{n(n+1)}{2}$.

Since we do not want to count the substring of zero length (i.e., null string), the number of substrings becomes $\frac{n(n+1)}{2}$.

PRACTICE SET

1. The following grammar
$G = (N, T, P, S)$
$N = \{S, A, B, C\}$
$T = \{a, b, c\}$
$P : S \rightarrow aS$
$A \rightarrow bB$
$B \rightarrow cC$
$C \rightarrow a$ is
(a) is type 3
(b) is type 2 but not type 3
(c) is type 1 but not type 2
(d) is type 0 but not type 1

2. Which of the following denotes Chomskianhiearchy?
(a) REG $\subset$ CFL $\subset$ CSL $\subset$ type0
(b) CFL $\subset$ REG $\subset$ type0 $\subset$ CSL
(c) CSL $\subset$ type0 $\subset$ REG $\subset$ CFL
(d) CSL $\subset$ CFL $\subset$ REG $\subset$ type0

3. Let Q = {aa, abaaabb, bbaaaaa, bbbbbbbbbb} and R = {b, bbbb, bbbaaa, bbbaaaaa}
Pref (Q in R) is equal to,
(a) {b,bbba,bbbaaa} (b) {b,bba,bbaaa} (c) {ab,bba,bbbaa} (d) {b,bba,bbba}

4. A countable union of countable set is not
(a) Countable (b) Uncountable (c) Countably Infinite (d) Denumerable

ANSWER KEY

1. ( ) **2. (a)** **3. (a)** **4. (b)**