GPCET
Pioneering Innovative Education

G.PULLAIAH COLLEGE OF ENGINEERING & TECHNOLOGY

(Accredited by NAAC with 'A' Grade of UGC Approved by AICTE, New Delhi & Permanently Affiliated to JNTUA, Ananthapuramu)
(Recognized by UGC under 2(f) & 12(B) &ISO 9001:2008 Certified Institution)
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NAME OF THE SUBJECT: SOFTWARE TESTING METHODOLOGIES

SUBJECT CODE: 13A05605

YEAR/SEM: III B.Tech II SEM

NAME OF THE FACULTY: R.Sandeep Kumar

2 MARKS QUESTIONS

UNIT-4 PATH, PATH PRODUCTS AND PATH EXPRESSIONS & LOGIC BASED TESTING

1. Define path products?

Any expression that consists of path names and "OR"s and which denotes a set of paths between two nodes is called a "**Path Expression.**"

2. Define path products?

The name of a path that consists of two successive path segments is conveniently expressed by the concatenation or **Path Product** of the segment names.

3. Write short notes on path sums?

The "+" sign was used to denote the fact that path names were part of the same set of paths.

The "PATH SUM" denotes paths in parallel between nodes.

Links a and b in Figure are parallel paths and are denoted by a + b. Similarly, links c and d are parallel paths between the next two nodes and are denoted by c + d.

The set of all paths between nodes 1 and 2 can be thought of as a set of parallel paths and denoted by eacf+eadf+ebcf+ebdf.

If X and Y are sets of paths that lie between the same pair of nodes, then X+Y denotes the UNION of those set of paths. For example, in Figure

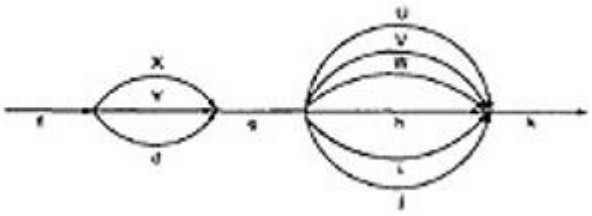


Figure: Examples of path sums.

The first set of parallel paths is denoted by X + Y + d and the second set by U + V + W + h + i + j. The set of all paths in this flowgraph is f(X + Y + d)g(U + V + W + h + i + j)k

4. List about the different law on path products?

The path is a set union operation, it is clearly Commutative and Associative.

- RULE 2: X+Y=Y+X
- RULE 3: (X+Y)+Z=X+(Y+Z)=X+Y+Z
  - RULE 6: $X^{\underline{n}} + X^{\underline{m}} = X^{\underline{n}}$ if n>m
  - RULE 6: $X^{\underline{n}} + X^{\underline{m}} = X^{\underline{m}}$ if m>n
  - RULE 7: $X^n X^m = X^{\underline{n+m}}$
  - RULE 8: $X^n X^* = X^* X^n = X^*$
  - RULE 9: $X^n X^+ = X^+ X^n = X^+$
  - RULE 10: $X^* X^+ = X^+ X^* = X^+$
  - RULE 11: 1 + 1 = 1
  - RULE 12: 1X = X1 = X

Following or preceding a set of paths by a path of zero length does not change the set.

RULE 13: $1^n = 1^{\underline{n}} = 1^* = 1^+ = 1$

No matter how often you traverse a path of zero length,It is a path of zero length.

RULE 14: $1^+ + 1 = 1^* = 1$

The null set of paths is denoted by the numeral 0. it obeys the following rules:

RULE 15: X+0=0+X=X

RULE 16: 0X=X0=0

If you block the paths of a graph for or aft by a graph that has no paths , there won't be any paths.

- **DISTRIBUTIVE LAWS:**
  - The product and sum operations are distributive, and the ordinary rules of multiplication apply; that is

- RULE 4: A(B+C)=AB+AC and (B+C)D=BD+CD
  - Applying these rules to the below Figure 5.1a yields
  - e(a+b)(c+d)f=e(ac+ad+bc+bd)f = eacf+eadf+ebcf+ebdf

- **ABSORPTION RULE:**
  - If X and Y denote the same set of paths, then the union of these sets is unchanged; consequently,
    - RULE 5: X+X=X (Absorption Rule)
  - If a set consists of paths names and a member of that set is added to it, the "new" name, which is already in that set of names, contributes nothing and can be ignored.
  - For example,
  - if X=a+aa+abc+abcd+def then
    - X+a = X+aa = X+abc = X+abcd = X+def = X

5. Write short notes on loops in a path with an example?

Loops can be understood as an infinite set of parallel paths. Say that the loop consists of a single link b. then the set of all paths through that loop point is b0+b1+b2+b3+b4+b5+.....

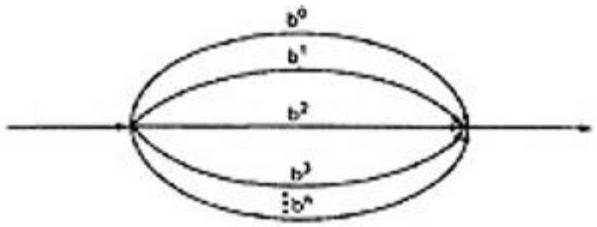


Figure: Examples of path loops.

This potentially infinite sum is denoted by b* for an individual link and by X* when X is a path expression.

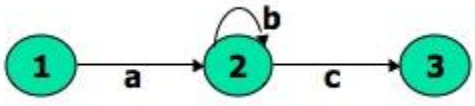


Figure: Another example of path loops.

- The path expression for the above figure is denoted by the notation:
  - ab*c=ac+abc+abbc+abbbc+.....
- Evidently,

- aa*=a*a=a+ and XX*=X*X=X+

- It is more convenient to denote the fact that a loop cannot be taken more than a certain, say n, number of times.
- A bar is used under the exponent to denote the fact as follows:

$$X^{\underline{n}} = X^0+X^1+X^2+X^3+X^4+X^5+\ldots\ldots\ldots\ldots\ldots+X^n$$

6. Write the steps involved in reduction procedure algorithm?

The steps in Reduction Algorithm are as follows:

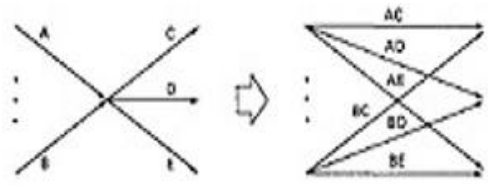
1. Combine all serial links by multiplying their path expressions.
2. Combine all parallel links by adding their path expressions.
3. Remove all self-loops (from any node to itself) by replacing them with a link of the form X*, where X is the path expression of the link in that loop.

STEPS 4 - 8 ARE IN THE ALGORIHTM'S LOOP:

4. Select any node for removal other than the initial or final node. Replace it with a set of equivalent links whose path expressions correspond to all the ways you can form a product of the set of inlinks with the set of out links of that node.
5. Combine any remaining serial links by multiplying their path expressions.
6. Combine all parallel links by adding their path expressions.
7. Remove all self-loops as in step 3.
8. Does the graph consist of a single link between the entry node and the exit node? If yes, then the path expression for that link is a path expression for the original flowgraph; otherwise, return to step 4.

7. Write short notes on cross term step?

- The cross - term step is the fundamental step of the reduction algorithm.
- It removes a node, thereby reducing the number of nodes by one.
- Successive applications of this step eventually get you down to one entry and one exit node. The following diagram shows the situation at an arbitrary node that has been selected for removal:

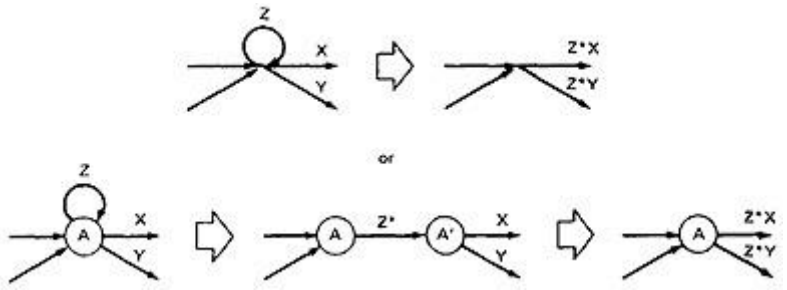


From the above diagram, one can infer:

(a + b)(c + d + e) = ac + ad + + ae + bc + bd + be

8. What are the two different operations of removing a loop?

There are two ways of looking at the loop-removal operation:



In the first way, we remove the self-loop and then multiply all outgoing links by Z*.

In the second way, we split the node into two equivalent nodes, call them A and A' and put in a link between them whose path expression is Z*. Then we remove node A' using steps 4 and 5 to yield outgoing links whose path expressions are Z*X and Z*Y

9. What are the applications of node removal algorithm?

The purpose of the node removal algorithm is to present one very generalized concept- the path expression and way of getting it.

Every application follows this common pattern:

1. Convert the program or graph into a path expression.
2. Identify a property of interest and derive an appropriate set of "arithmetic" rules that characterizes the property.
3. Replace the link names by the link weights for the property of interest. The path expression has now been converted to an expression in some algebra, such as ordinary algebra, regular expressions, or boolean algebra. This algebraic expression summarizes the property of interest over the set of all paths.
4. Simplify or evaluate the resulting "algebraic" expression to answer the question you asked.

10. What are the different ways of methods are there in finding the number of paths?

The question is not simple. Here are some ways you could ask it:

- What is the maximum number of different paths possible?
- What is the fewest number of paths possible?
- How many different paths are there really?

- What is the average number of paths?

  - Determining the actual number of different paths is an inherently difficult problem because there could be unachievable paths resulting from correlated and dependent predicates.
  - If we know both of these numbers (maximum and minimum number of possible paths) we have a good idea of how complete our testing is.
  - Asking for "the average number of paths" is meaningless.

11. Write short notes on how to find the maximum path count of a flow graph?

- Label each link with a link weight that corresponds to the number of paths that link represents.
- Also mark each loop with the maximum number of times that loop can be taken. If the answer is infinite, you might as well stop the analysis because it is clear that the maximum number of paths will be infinite.
- There are three cases of interest: parallel links, serial links, and loops.

Case	Path expression	Weight expression
Parallels	A+B	W_A+W_B
Series	AB	W_AW_B
Loop	A^n	$\sum_{j=0}^{n} W_A^j$

11. Write short notes on how to find the minimum path count of a flow graph?

A lower bound on the number of paths in a routine can be approximated for structured flow graphs. The arithmetic is as follows:

Case	Path expression	Weight expression
Parallels	A+B	W_A+W_B
Series	AB	$\max(W_A,W_B)$
Loop	A^n	$1, W_1$

12. What is the probability of being at a certain point in a routine?

Weights, Notations and Arithmetic:

1. Probabilities can come into the act only at decisions (including decisions associated with loops).
2. Annotate each outlink with a weight equal to the probability of going in that direction.
3. Evidently, the sum of the outlink probabilities must equal 1
4. For a simple loop, if the loop will be taken a mean of N times, the looping probability is N/(N + 1) and the probability of not looping is 1/(N + 1).
5. A link that is not part of a decision node has a probability of 1.
6. The arithmetic rules are those of ordinary arithmetic.

Case	Path expression	Weight expression
Parallel	A+B	P_A+P_B
Series	AB	P_AP_B
Loop	A*	$P_A / (1-P_L)$

13. Write the arithmetic rules for calculating the mean processing time of a routine?

The model has two weights associated with every link: the processing time for that link, denoted by **T**, and the probability of that link **P**

Case	Path expression	Weight expression
Parallel	A+B	$T_{A+B}=(P_AT_A+P_BT_B)/(P_A+P_B)$ $P_{A+B}= P_A+P_B$
Series	AB	$T_{AB}=T_A+T_B$ $P_{AB} =P_AP_B$
Loop	A^n	$T_A= T_A+ T_LP_L/(1-P_L)$ $P_A =P_A/(1-P_L)$

14. Define decision table?

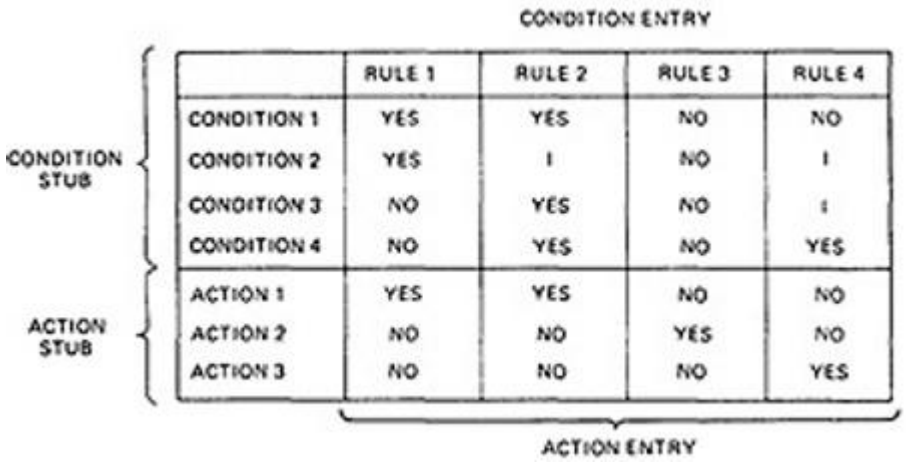
A **decision table** is used to represent conditional logic by creating a list of tasks depicting business level rules. **Decision tables** can be used when there is a consistent number of a condition that must be evaluated and assigned a specific set of actions to be used when the conditions are finally met.

15. What are the advantages of constructing a decision table for an application?

- Conciseness

- Easy checking for completeness, consistency and correctness
- Flexibility
- Overview
- Readable in two ways (data oriented, goal oriented)
- Correctness and speed of decision making
- Uniform communication medium
- Automatic conversion into computer programs
- Simple specification of test data
- Ease of translation between languages

16. Depict the pictorial representation of decision table?



17. What are the four areas that decision table consists of?

The functional requirements of many programs can be specified by **decision tables**, which provide a useful basis for program and test design.

- It consists of four areas called the condition stub, the condition entry, the action stub, and the action entry.
- Each column of the table is a rule that specifies the conditions under which the actions named in the action stub will take place.
- The condition stub is a list of names of conditions.
- A rule specifies whether a condition should or should not be met for the rule to be satisfied. "YES" means that the condition must be met, "NO" means that the condition must not be met, and "I" means that the condition plays no part in the rule, or it is immaterial to that rule.
- The action stub names the actions the routine will take or initiate if the rule is satisfied. If the action entry is "YES", the action will take place; if "NO", the action will not take place.

18. How decision-tables are basis for test case design?

1. The specification is given as a decision table or can be easily converted into one.

2. The order in which the predicates are evaluated does not affect interpretation of the rules or the resulting action - i.e., an arbitrary permutation of the predicate order will not, or should not, affect which action takes place.
3. The order in which the rules are evaluated does not affect the resulting action - i.e., an arbitrary permutation of rules will not, or should not, affect which action takes place.
4. Once a rule is satisfied and an action selected, no other rule need be examined.
5. If several actions can result from satisfying a rule, the order in which the actions are executed doesn't matter.

19. List some rules of Boolean algebra?

- Boolean algebra has three operators: X (AND), + (OR) and $\overline{A}$(NOT)
- **X :** meaning AND. Also called multiplication. A statement such as AB (A X B) means "A and B are both true". This symbol is usually left out as in ordinary algebra.
- **+ :** meaning OR. "A + B" means "either A is true or B is true or both".
- $\overline{A}$ meaning NOT. Also negation or complementation. This is read as either "not A" or "A bar". The entire expression under the bar is negated.

The following are the laws of Boolean algebra:

1. $A + A$	$= A$	If something is true, saying it
$\overline{A} + \overline{A}$	$= \overline{A}$	twice doesn't make it truer, ditto for falsehoods.
2. $A + 1$	$= 1$	If something is always true, then "either A or true or both" must also be universally true.
3. $A + 0$	$= A$	
4. $A + B$	$= B + A$	Commutative law.
5. $A + \overline{A}$	$= 1$	If either A is true or not-A is true, then the statement is always true.
6. AA	$= A$	
$\overline{A}\,\overline{A}$	$= \overline{A}$	
7. $A \times 1$	$= A$	
8. $A \times 0$	$= 0$	
9. AB	$= BA$	
10. $A\overline{A}$	$= 0$	A statement can't be simultaneously true and false.
11. $\overline{\overline{A}}$	$= A$	"You ain't not going" means you are. How about, "I ain't not never going to get this nohow."?
12. $\overline{0}$	$= 1$	
13. $\overline{1}$	$= 0$	
14. $\overline{A + B}$	$= \overline{A}\,\overline{B}$	Called "De Morgan's theorem or law."
15. $\overline{AB}$	$= \overline{A} + \overline{B}$	
16. $A(B + C)$	$= AB + AC$	Distributive law.
17. $(AB)C$	$= A(BC)$	Multiplication is associative.
18. $(A + B) + C$	$= A + (B + C)$	So is addition.
19. $A + \overline{A}B$	$= A + B$	Absorptive law.
20. $A + AB$	$= A$	

20. What is the purpose of kv charts in software testing?

Consistency and completeness can be analyzed by using Boolean algebra, which can also be used as a basis for test design. Boolean algebra is trivialized by using **Karnaugh-Veitch charts**.