UNIT-I

Flowgraphy and Path Testing

Part-(2).

2.1 Basic Concepts of Path Testing:-

Path Testing:- A sequence of statements which starts at an entry and ends at an exit by passing all the existing functions, decisions etc. is known as path. Path may end at the same function or the other or at any decision or exit.

2.1.1 Control Flowgraphy:-

A control flowgraph is a form of a flowchart which does not deal with the internal structure of the process rather it shows the data flow and the control flow between the processes. Every control flowgraph has some mandatory elements they are,

1) Process blocks
2) Decisions and Case statements
3) Junctions.

1) Process Blocks:- Process block is a block which consists of sequence of statements, which once initiated results in the

execution of all the statements in that block.

~~Every process blocks are neither~~

Every process has an entry and an exit and consists of a single or series of statements. Control flowgraphs are not concerned with the details of operations in a process block, so the test cases are designed accordingly.

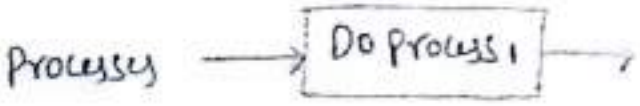


fig: Process Block

2) Decisions and case statements :-

Decisions :- Decision is a point in a program at which the control flow can split. The flow gets diverted in one of the many options available.

Mostly, decisions split in two-way, occasionally they split in three-way branches. The test case design is comparatively easy for two-way than with three-way branches.

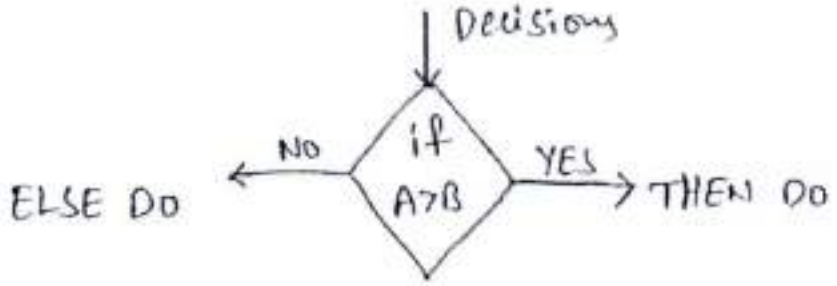


fig: Two-way Decisions.

Case statements:- Any decision can split the control flow into different way branches. This multiway branches can be termed as case statements are same.

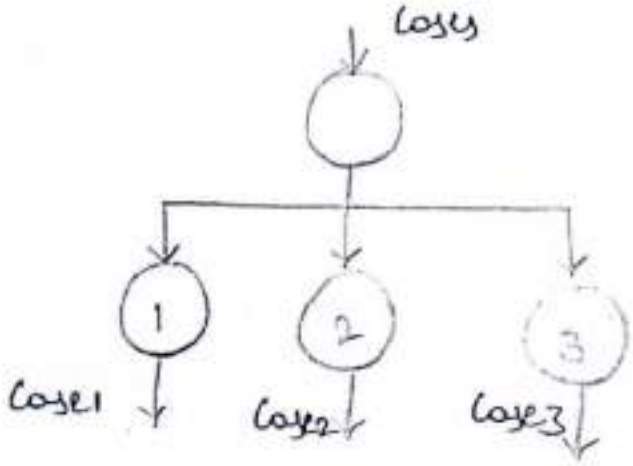


fig: Case Statements

3) Junctions:- A junction is just contradict to decision. All the control flows can merge at a point in a program which can be termed as junction. In other words, a node with more than one input line is known as junction.

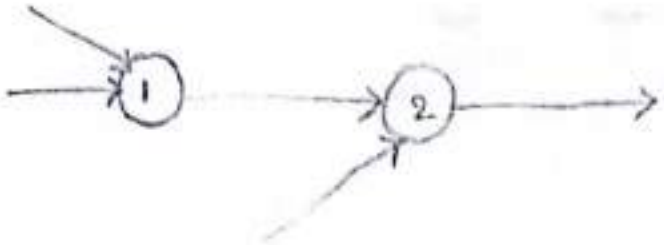


fig: Junctions

2.1.2 Control flowgraphs versus Flowcharts:-

Difference between control flowgraphs and flowcharts are as follows,

1) flow chart is a graph which represents the control structure of the program, as well as the internal structure of each and every process or process block.

2) Control flowgraph is also a graph which represents the control structure of a program, but is excludes the detailed structure of process blocks.

3) All the steps inside a process are shown using ~~flowcd~~ flowchart in addition to the control flow, but control flowgraph considers all the steps as a single process entity and shows only the control flows to and from that process entity.

4) flowchart also shows the internal ~~structure~~ flows of each process so, it is difficult to identify the actual control flows between different processes.

5) flowcharts had lost its importance because of the detailed information it provides, which is not in use for process design.

6) We can also use flowchart for representing the control and data flows in a traditional way and control flowgraphs as the modern approach for representation of flows.

7) flowchart has a box to represent each and every process step which is not the case with control flowgraph only the outline of process block is shown in control flowgraph.

2.1.3 Advantages and Disadvantages of control flowgraph:

The control flowgraph is a graph which represents the control structure of a program in such a way that the

detailed structure of a process block is hidden. (26)

There are some major advantages, as well as disadvantages of control flowgraphs.

Advantages :-

1) Control flowgraph eliminates the occurence of some problems which results from expanding the visual complexities.

2) Control flowgraph treats all the steps inside a process as a single process entity and shows only data and control flow to and from that entity there by reducing the complexity of structure.

3) Control flowgraphs can be referred to as a modern approach for representation of flows.

4) Control flowgraphs gives the precise and clear view of the program structure, the directions of data flow etc.

Disadvantages:-

1) Control flowgraph plays an important role in representing the program control structure, but are sparsely due to the scarcity of control flowgraphs generators.

2) The information needed to produce a control flowgraph is not provided by most of the compilers.

3) Control flowgraph structure is similar to many programming structures and is very difficult to differentiate.

2.1.4 Path Testing:-

i) Paths:- A series of statements initiating from an entry and terminating at an exit, there by passing through the junctions and decisions is known as a path.

A path may initiate or terminate at the same or probably different entry, junction, decision or exit.

Every path consists of a set of Processes known as links.

En:- Consider the following example as shown in figure:

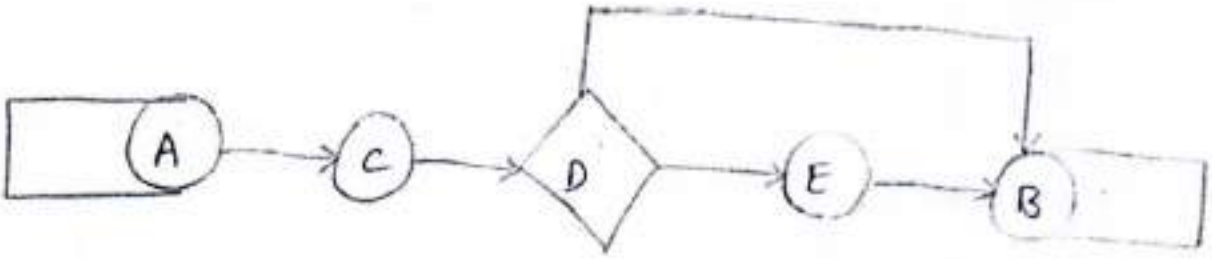


fig: An example of path selection.

There are two different paths from an entry (A) to an exit (B) they are 'ACDEB' and 'ACDB' respectively. Both path are simple, the most obvious among the two is 'ACDB' because, it is the shortest path between an entry and an exit.

ii) Nodes :- Nodes are the graphical representation of the real world entities (or) the abstract objects in any graph which resembles the objects in a real world are known as nodes.

Nodes are mainly denoted by small circles. A node which has more than one input link is known as a junction, and a node which has more than one output link is referred to as a decision. Nodes can be labelled by an alphabets or numbers.

Ex:- Consider the following example as shown in fig

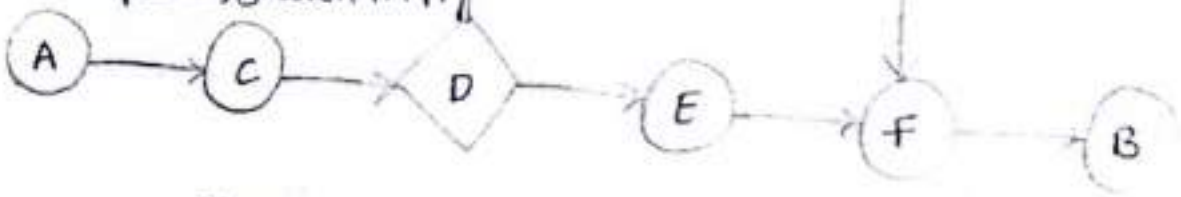


fig: A flowgraph with Nodes A, B, C, D, E, F

In above fig- all the nodes ~~(A, F, C, D~~, (A, B, C, E, f), D is the decision maker which has 2 output links, and f is a junction which has 2 input links.

iii) Links:- Every path consists of a set of Processes known as links. A link is the mediator for any two nodes, links can be denoted by an 'arrow' and can be represented by the lower case letters.

Ex:- In the following example as shown in fig a, b, c, d, e, f are all the available links.

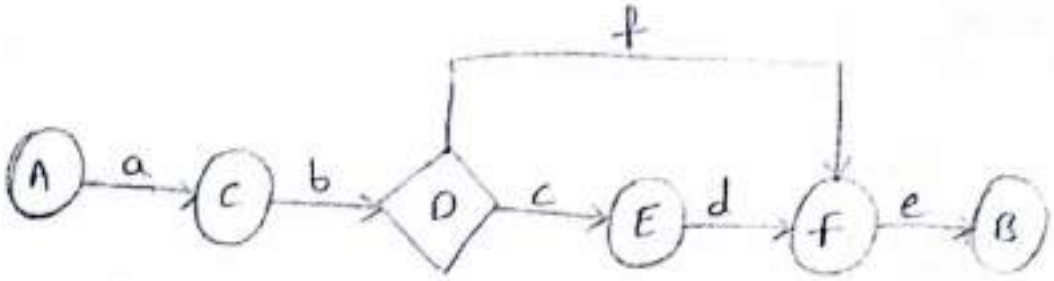


fig: A flowgraph with links a, b, c, d, e, f

The paths from A to B are (abcde) and (abfe), which are the concatenation of some selected links.

iv) Multi-Entry / multi-Exit Routines:-

Multi-entry means, multiple entry points and multiple-exit refers to multiple exit points. There are certain situations in which it is appropriate to change the routine and choose an alternate way to normal control structure.

You may want to choose an alternative routine when an illegitimate condition occur and will damage the system's data, if that path is continued further.

The other reason might be the occurrence of several fluctuations during the processing of same path. Hence, so changing of routine is advantageous in such situations by placing an entry point in a routine which sends the flow to appropriate location.

If multiple choice outlomes are possible, then place an exit parameter in that routine.

The main drawback of multi-entry and multi-exit routines is that all the test cases are difficult to cover because the control flow between various processes can't be determined easily due to ~~multiple~~ multiple entry/exit points, there by test coverage problem arises.

v) fundamental path selection criteria:-

Every routine may have several number of different path's available from its entry to its exit. path selection mainly deals with the selection of an optimal

Path between its entry and exit.

If a routine contains decisions or loops inside it, then there will be more number of paths. For example the number of paths become double on each decision and gets multiplied with the number of iterations on each occurrence of a loop.

Complete testing usually involves testing every path, statement and branch in each direction at least once. If we have tested every path then every statement and branch will automatically be covered.

Example:-

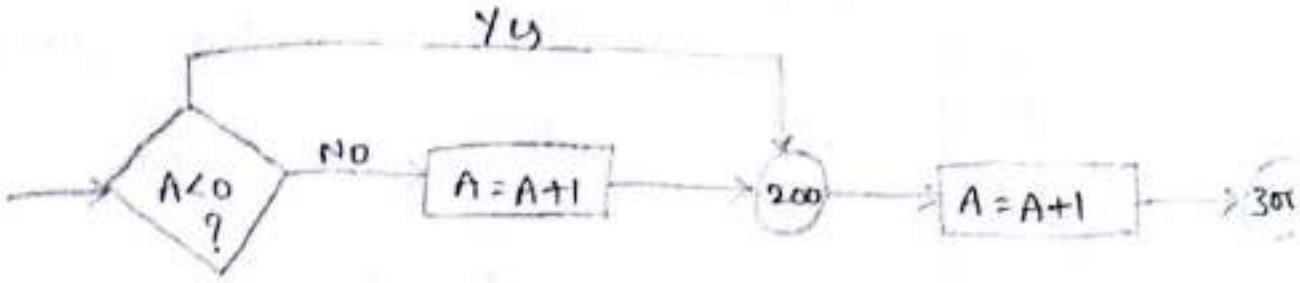


fig: Control flowgraph.

In the above example, if A is less than 0, then the output will be A+1 and if A is greater than 0, then the output will be A+2, because decision doubles

the number of paths.

vi) Path Testing Criteria:

There are three different path testing strategies/criterion. They are as follows,

i) Path Testing (P_∞)

ii) Statement Testing (P_1)

iii) Branch Testing (P_2)

Path testing with the execution of paths, if we have tested all the available control flowgraphs, we have achieved 100% path coverage which is mostly impossible.

Statement testing deals with the execution of all the statements inside a program at least once.

If we do enough tests to achieve this, 100% statement coverage is said to have achieved. In alternate way 100% node coverage which is denoted by C_1.

Branch testing deals with the execution of all the (5) branches in a program.

If enough tests are done to achieve this, then 100% branch coverage is said to have achieved. In alternate way 100% link coverage, which is denoted by C_2.

vii) Which path to be selected / chosen:

a) During the process of path selection, always prefer simple and easily achievable paths, i.e, complicated paths, even if they are few in number must be given the least priority while selection.

b) Pick additional paths as small variations from previous paths, that do not have loops.

c) Paths should not be chosen blindly following the rules but implement them to achieve C_1+C_2.

The following is the example of path selection.

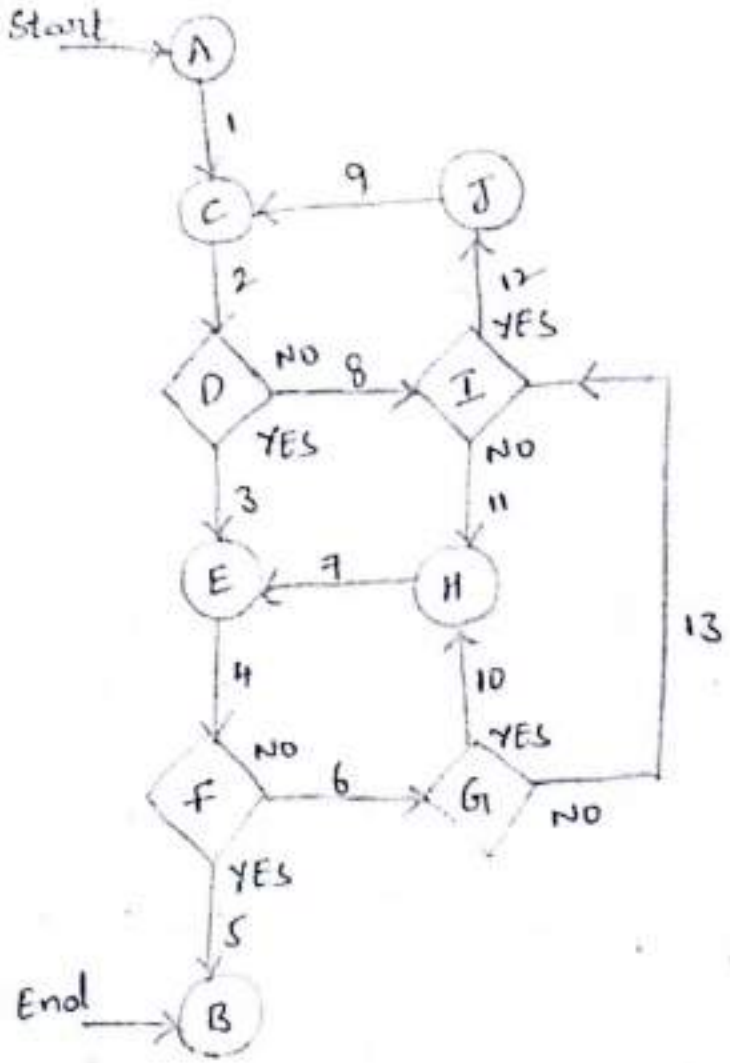


fig: Path Selection

The most obvious path the A to B interms of nodes (i.e start to end) is (A, C, D, E, F, B). Interms of links it will be (1, 2, 3, 4, 5). By using links the second most appropriate path is (1, 2, 8, 11, 7, 4, 5). All other paths lead to loops. Consider the first loop (1, 2, 8, 12, 9, 2, 3, 4, 5), second loop (1, 2, 3, 4, 6, 10, 7, 4, 5) and then the last loop

(1,2,3,4,6,13,9,2,3,4,5) · choose the optimal path from all the available paths.

2.1.5 Loops:-

There are three different kinds of loops. They are as follow

i) Nested Loop

ii) Concatenated loops

iii) Horrible loops

i) Nested Loops:- The nested loops are quite complicate i.e a loop with in another loop is known as nested loop. It is very expensive to test the path which contains nested loop because of its complexity. To overcome this complexity we have to follow some crucial steps.

a) Begin your test from the most internal loop, move towards the external / outer loop there by keeping its value to be minimum.

b) Test all the internal loop's values, explore the test if any value is missed.

c) After testing the innermost loop come out of it and conduct a test on the immediate outer loop.

Example:

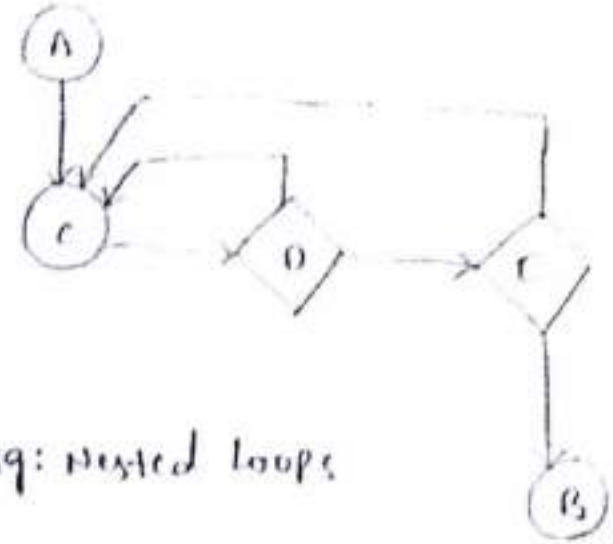
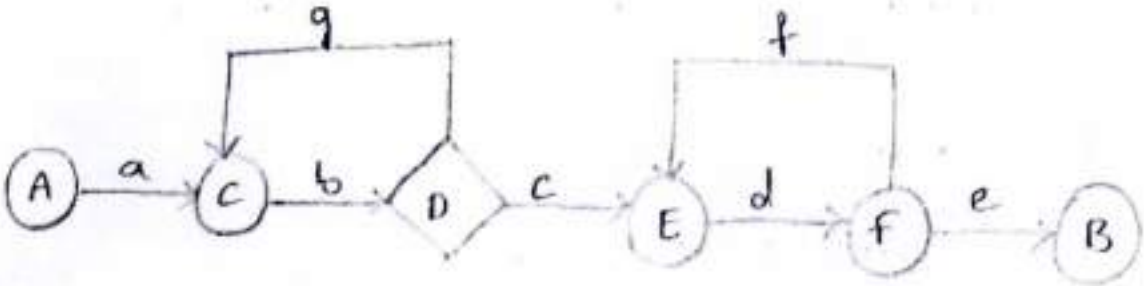


Fig: Nested loops

ii) Concatenated loops:- Concatenated loops are the loops which reside the one beside the other on the same path. In other words, when there exists two adjacent loops on the same path such that, on exit of one loop serves as an entry point for the other loop, then the loops are said to be concatenated.

If the loops are not on the same path they are said to be individual loops but not concatenated loops.



iii) Horrible Loops:- Horrible loops are the complexed of all the three loops and may involve nested loops, intersecting loops, cross connected loops all in one structure.

This complex structure of horrible loops makes it very difficult to be tested. The design of test cases for horrible loops are indefinite and are too many to execute. Hence, horrible loops must be avoided.

2.1.6 More on Testing Multi-Entry / Multi-Exit Routines:-

The issues related to multi-entry and multi-exit routines are as follows,

i) Weak Approach:- To test the program with multi-entry and multi-exit routines, we need to follow certain procedures.

first, built the imaginary single entry routine and imaginary exit routine with pseudo case statements

and processes respectively. Secondly concentrate on hypothetical common junction. This overall structure of the code will enable you to create test case design for multi-entry/multi-exit routines.

Multi-exit designers are the people responsible for choosing an exit, but selecting an entry will be quite clumsy, because many different programmers can initialize an entry point.

a) Multi-entry routine is converted to single entry with case statements as shown in below figure:

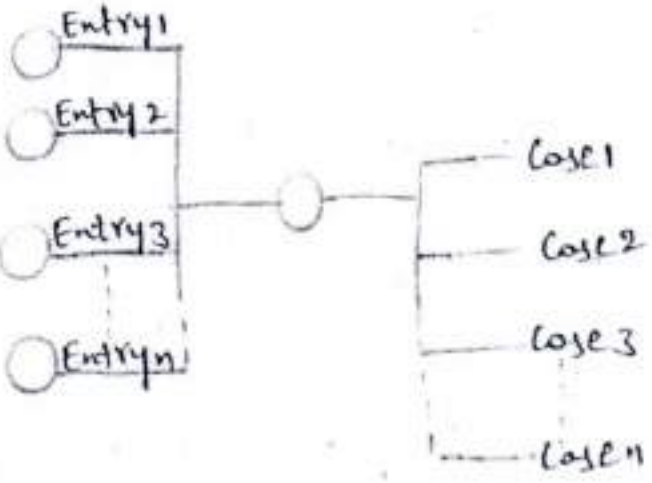


Fig: multi-entry to single Entry using Case Statements

b) Multi-exit routine is converted to single exit with processes as shown in below figure:

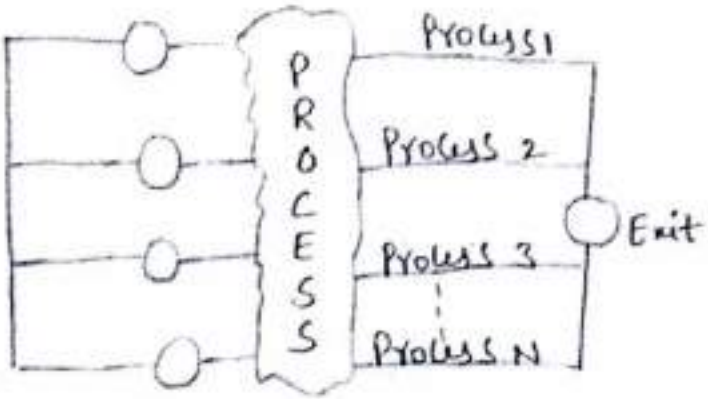


fig: Multi-exit to Single Exit using processes.

ii) Path Testing Strategies:-

a) Avoid multi-entry|multi-exit routines as much as possible, because it is difficult to cover all the test cases for these routines.

b) If it is mandatory for you to use multi-entry|multi-exit routines, try to achieve maximum control over these routines.

c) Each and every entry|exit pair is considered as a separate routine, which is tested individually using powerful unit testing mechanism.

d) multi-entry and multi-exit routines are assumed to be more unusual and dangerous, so integration testing is performed with more efforts and concentration.

2.2 Predicates, Path Predicates and Achievable paths:-

2.2.1 Predicates:-

Predicates is a function which is logically executed during the decision Processing.

Example:- "A is less than or equal to 5"

2.2.2 Predicate Expressions:-

Predicate is a function which is executed during the decision Processing.

Predicate interpretation refers to the process of expressing the predicate in terms of the given input vector by performing various symbolic replacement of operations.

Consider an example, where the predicate is the sum of A, and C greater than zero, which is symbolically represented as $A+C>0$. Now, let the value of C be given

using another Predicate as "C := B+5".

The Substitution of C value in the first Predicate gives you another predicate which is "A+B+5 > 0". This Process is known as Predicate interpretation.

2.2.3 Predicate Coverage:-

Predicate Coverage is the Process of testing all the truth values related to a specific path in all the possible ways.

If all the values are tested in all possible directions then we can say that 100% Predicate Coverage is achieved which needs lots of efforts.

2.2.4 Testing Blindness:-

Blindness is a situation which results in the correct path via wrong route unintentionally.

There are three types of predicate blindness. They are as follows,

1) Assignment Blindness
2) Equality Blindness
3) Self Blindness

1) Assignment Blindness:-

Assignment blindness comes into consideration when both the predicates irrespective of their correctness are satisfied by a value assigned to the assignment statement.

Assignment blindness may also lead to wrong path selection.

Correct	Incorrect
A=5 ..... If B>0 then Do Something	A=6 ..... If A+B>=5 THEN DO SOMETHING

In the above example, for all the positive values of B, both the predicates work correctly but, both results in different values of A, which would lead to wrong path.

2) Equality Blindness:- Whenever a path is chosen by the predicate, some value is generated as an outcome. If that value satisfies the consecutive predicates

irrespective of its correctness, then it is called equality blindness.

Example:

Correct	Incorrect
If B=7 THEN	If B=7 THEN
Do SOMETHING	Do SOMETHING
If A+B>=9 THEN	If A>=2 THEN
Do SOME OTHER THING	Do SOME OTHER THING

In the above example, it is clear that the second Predicate will lead to the desired path irrespective of its correctness for any positive value of B.

3) Self Blindness:- Sometimes, both the correct and the wrong predicate becomes similar to each other depending on the value assigned to the assignment statement. Hence the path chosen by both the predicates will be same

Example:-

Correct	Incorrect
X=X1	X=X1
.....	.....
If X-3>=0 THEN	If X+X1-4>0 THEN
Do SOMETHING	Do SOMETHING

In the above example, both the predicates i.e correct and incorrect are interdependent based on the assignment value i.e $(X = X_1)$.

2.3 Path Sensitizing:-

Path predicate expressions are the collection of expressions that must be fulfilled in order to achieve the desired path.

If all the expressions are met then the path is said to be achievable else the path is not achievable. An attempt to find the set of solutions to the path predicate expressions is called path sensitization.

2.3.1 Heuristic procedures for Sensitizing Paths:-

Heuristic procedures are the most optimistic ways for Sensitizing paths.

The first preference for selecting a path must be given to the paths which can be easily sensitized thereby delaying the paths whose solution to path predicate expression is difficult to obtain.

1) All the Process dependent, Process independent and correlated input variables are first determined and classified accordingly.

2) After classifying the variables, determine and classify the Predicates depending on the input variables into dependent, independent or correlated Predicates and also show the type of relation that exists among them.

3) Consider the uncorrelated and independent Predicates for selection of path. During your selection, if you have found any dependent Predicate, then there may be a classification error or there might be a bug or complete Path coverage is not yet achieved.

4) Now Consider the correlated and independent predicates if they are not covered then start considering the dependent and uncorrelated, Predicates. If the complete coverage is not yet accomplished then move on the last selection i.e consider correlated, dependent Predicates.

5) Display all the input variables, its values, relationship among the variables, type of links for all independent, dependent and correlated variables respectively of every selected path.

6) Every path will produce some set of inequalities which must be met in order to select that path.

Examples:-

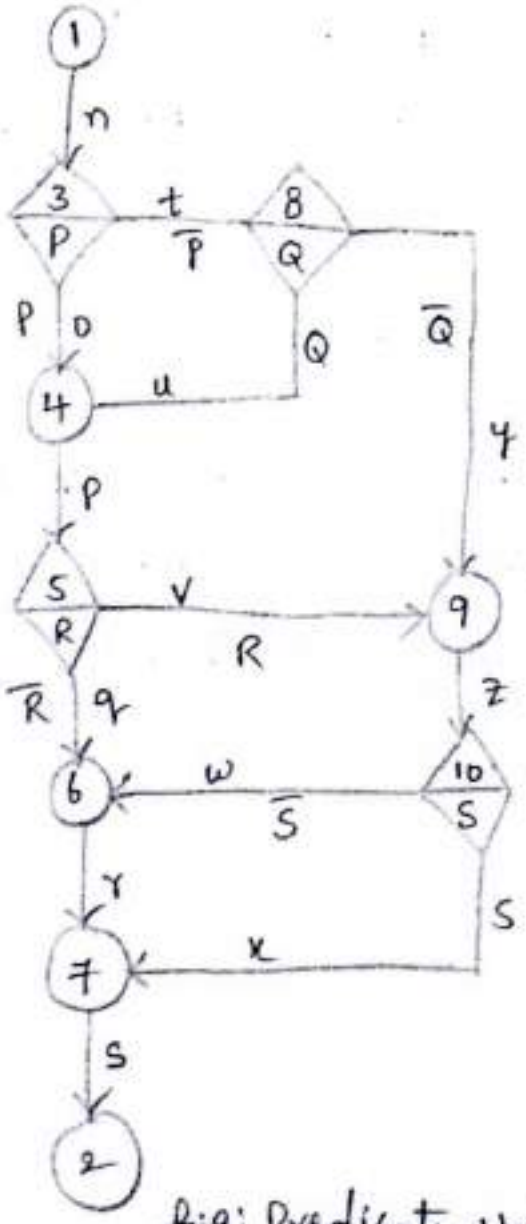


fig: Predicate Notation

Independent, Uncorrelated Predicates:-

Considering the independent, uncorrelated predicates. There are 4 predicates and 4 decisions which are denoted by "P, Q, R, S" and "◇" respectively.

False predicates are denoted by a bar on the variable. True predicates are represented by the variable (without any bar over them).

From the fig: We can retrieve all the covering path and their predicate values which can be represented in the tabular form as shown in Table:

Path	Predicate Values
n o p q r s	$P\bar{R}$
n t u p v z x s	$\bar{P}QRS$
n t y z w r s	$\bar{P}\bar{Q}\bar{S}$

We can also cover this control flowgraph with more number of simple paths i.e,

Path	Predicate values
n o p q r s	$P\bar{R}$
n o p v z w r s	$PR\bar{S}$
n o p v z x s	PRS
n t u p q r s	$\bar{P}Q\bar{R}$
n t y z x s	$\bar{P}\bar{Q}S$

2.4 Path Instrumentation:-

Path instrumentation is a technique used for identifying whether the outcome of a test is achieved through the desired path or a wrong path. Path instrumentation technique is another form of interpretive trace program, which will run each and every statement sequentially thereby storing all the labels and values of the statements covered so far.

The only drawback of the trace programs is its additional large information content, which is of no use.

To overcome this drawback, many different instrumentation methods have ~~ebv~~ evolved.

2.4.1 Link Marker method of Path Instrumentation:-

Link marker is also known as traversal marker which uses small case letters for naming every link.

Whenever a link is passed, its name is recorded in the marker. The concatenation of the names of all the links starting from an entry to an exit gives the path name.

The single link marker may not serve the purpose because there is every possibility of bug which may ~~boog~~ result in a new link in the middle of the link being traversed.

Example: Using Single link markers

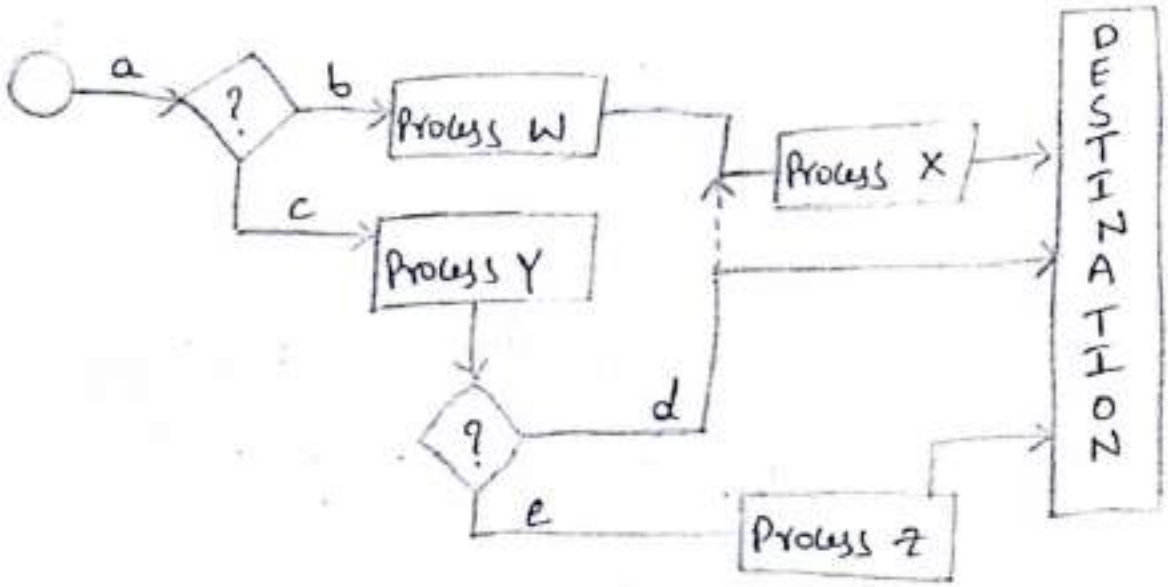


fig: Single link Marker method.

In the above example as shown in fig: the preferred route is through 'acd', due to an error in the structure, we reached the correct destination but via Process X.

To avoid this scenario, we need to use two markers on each link, one at the start of the link and other at the end of the link. Now, Path name includes both the markers of the links.

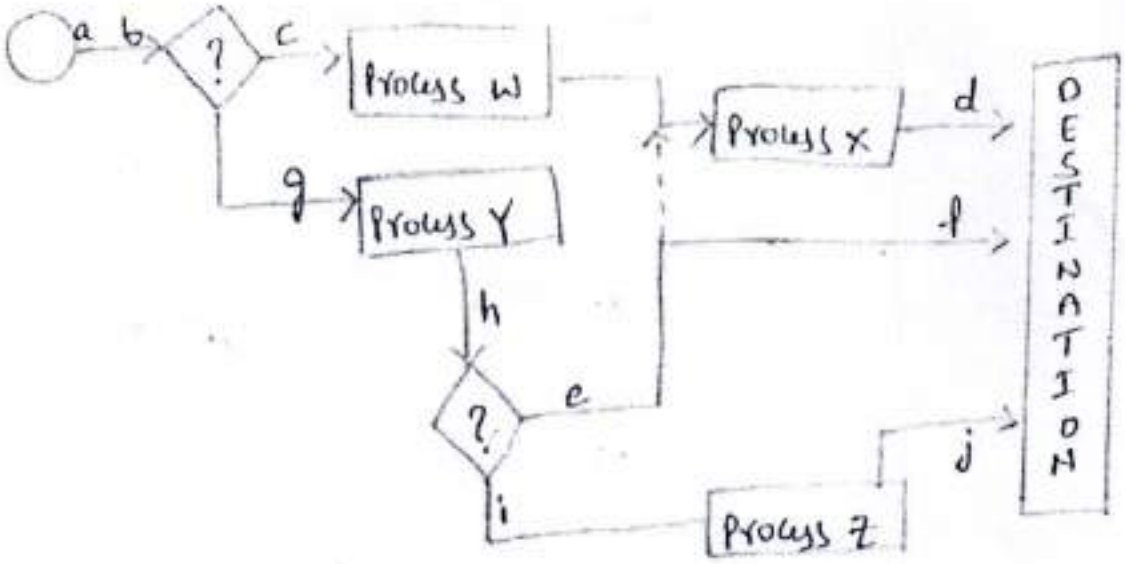


fig: Double Link Marker Method

Now, the same path as above will be 'abghef' instead of 'acd' So, even if we reached Process X, we could easily find, by observing the path name.

2.4.2 Link Counters:-

Link Counter is one of the instrumentation techniques which usually relies on the concept of counters.

Link counter instrumentation method provides comparatively less information than interpretive trace program.

Link counter method of instrumentation follows the same procedure as that of link marker but, make use of counters instead of using labels for each link which has executed.

Counters in this method goes on increasing with respect to each link traversed. Single counter may not serve the purpose so, we move little deeper and introduce a separate counter for every link. With this in practice, we can cross check the total link count against the expected path length. This format is not reliable because ~~this~~ there is every possibility of having a bug, which may result in a new link in the middle of the link being travelled.

To get rid of this type of bugs, we use double link counters i.e one counter at the entry and one counter at the exit of every link, traversed.

Now we can compare in the following way.

i) Are the counter values are equal for each entry and exit point of each link, decision and junctions?

ii) Does this value resembles the expected value?