The Graph Theoretic Approach for Structural Analysis of Query Based Test Generation System

Thesis submitted in partial fulfillment of the requirements for the award of degree of

Master of Engineering
in
Software Engineering

Submitted By
Bhagesh Kumar
(801231005)

Under the supervision of:
Mr. Ashish Aggarwal
Assistant Professor

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR UNIVERSITY
PATIALA – 147004

July 2014

CERTIFICATE

I hereby certify that the work which is being presented in the thesis entitled, **"The Graph Theoretical Approach For Structural Analysis Of Query Based Test Generation System"**, in partial fulfillment of the requirements for the award of degree of Master of Engineering in Software Engineering submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of **Mr. Ashish Aggarwal** and refers other researcher's work which are duly listed in the reference section.

The matter presented in the thesis has not been submitted for award of any other degree of this or any other University.

Bhagesh Kumar

(Bhagesh Kumar)
M.E(Software Engineering)
801231005

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

(Ashish Aggarwal)
Assistant Professor
Computer Science and Engineering Department
Thapar University, Patiala

Countersigned by

(Dr. Deepak Garg)
Head
Computer Science and Engineering Department
Thapar University
Patiala

(Dr. S. K. Mohapatra)
Dean (Academic Affairs)
Thapar University
Patiala

Acknowledgement

I express my gratitude and appreciation to all those who have helped me throughout the duration of my research work. It would not have been possible to complete this research without guidance and the help of several individuals who in one way or another contributed and extended their valuable assistance in the preparation and completion of this study.

First and foremost, I would like to express my deep and sincere gratitude to my supervisor Mr. Ashish Aggarwal, Assistant Professor, Computer Science and Engineering Department, Thapar University, Patiala. It gives me immense pleasure to pay my gratitude for his valuable advice, constructive criticism and great patience at all the times.

I owe my sincere thanks to Dr. Deepak Garg, Professor and Head of Computer Science and Engineering Department, Thapar University, Patiala, for his continuous motivation, cooperation and for providing facilities.

I wish to express my sincere thanks to all the staff members and my friends who always supported and encouraged me. I was very fortunate to have an unconditional support from my family. I want to acknowledge the contributions of my parents, for their constant motivation, inspirations and for supporting me spiritually throughout my life. Last but not the least, I would like to thank God for giving me inner peace and strength.

Bhagesh Kumar

M.E (Software Engineering)

(801231005)

Abstract

An attempt is made to develop an integrated systems model for the structure of the query based test generation system in terms of its components and interactions between the components using graph theory and matrix algebra. The test generations system is first modeled with the help of a graph theory, then by a variable adjacency matrix and then by a multinomial known as a permanent function. The permanent function provides an opportunity to carry out structural analysis of the test generation system in terms of program running time and storage optimization by correlating the properties of a test generation system with its structure.

A physical meaning has been associated with each term of the permanent function. Different structural attributes of the test generation system are identified to develop a graph theoretic model, a matrix model, and a multinomial permanent model of the test generation system. This matrix representation is a powerful tool for storage and retrieval of test generations in computer database and also for computer processing. A top-down approach for complete analysis of any test generation system is also given.

Table of Contents

List of Figures

Chapter 1 Introduction

The databases are the soul of operations performed in all the applications in any of the modern organizations. The DBMS (database management systems) available for purchase by the organization/users provide access to a large amount of data and in addition to this also maintain the data integrity and enables the user to know the details of the storage and retrieval procedures. To make an efficient use of this, an off-the-shell DBMS is available in the market, the organization can buy it and then design the database schemas and other application programs desired for the business use.

To meet all these specifications the database systems are required to work correctly and provide desired results. The efforts have been made by the organizations to make sure that the algorithms and the data structures present in the database management systems works as desired and in addition to this the integrity of data is also maintained.

1.1 Structural Analysis

The structural analysis is an ancient phenomenon and had been practiced since the early civilizations. There have been many civilizations where the skilled craftsmen have given examples of their analytical work by building great structures like The Pyramids of Egypt (2000 B.C.), The Parthenon of Athens, The Taj mahal and many more. These structures themselves speak of the great feats of the craftsmen in the field of the analysis, design and construction.

- A system having various parts connected together is called a Structure.
- A Structure is identified on the basis of their form or on the basis of the functions performed by it i.e. taking into account the elements present in it.
- In order to make perfect structural analysis the geometry of the structure and the properties of the structure must be considered.

1.2 Computer Applications and Graph Theory

One of the important aspects of understanding Graph Theory is the usage of the digital computers for solving the Graph-theoretical problems, especially when the

application programs are involved. Large part of the practical problems present now-a-days that involve graph theory is concerned with the large graphs- the graphs that are too difficult for solving manually. The computer applications have been built to deal with large graphs that have to be encountered during the problems like PERT, electrical networks etc. As far as the combinatorial problems are concerned, the analysis of the graphs is generally non-numerical. In graph-theoretical application program, the decision taking ability of the computer is more beneficial than solving the arithmetic problems. So the Graph theory can become a powerful tool for computer oriented problems and applications.

1.3 Graph Theory

The study that deals with the graphs, their organization, their structure, operations etc is called Graph Theory.

1.3.1 Graph

A graph is an entity consisting of the vertices and edges connected together. The graphs are the structures that are used to portray the pair wise relationship between the various objects. A Graph G (V,E) is formed by the pair of sets V and E with V being the vertex set and E being the edge set. Every element have a multiplicity because the elements can occur more than once so E is a multi-set. A graph with odd number of vertices have even degree and even number of vertices with odd degree [22].

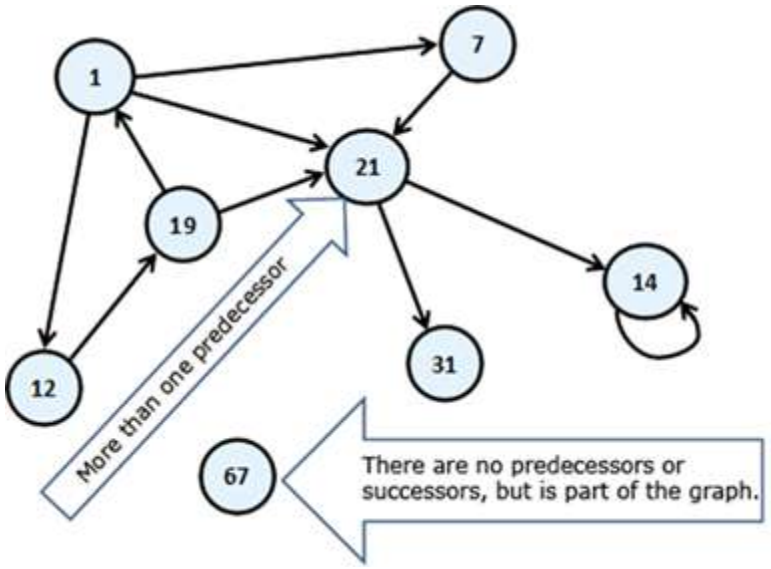


Fig. 1 Showing elements of graph with connectivity and no connectivity [22]

- Complete graph: A graph in which all the vertices are connected to every other vertex is called Complete Graph.
- Walk : Consider a graph G = (V,E).

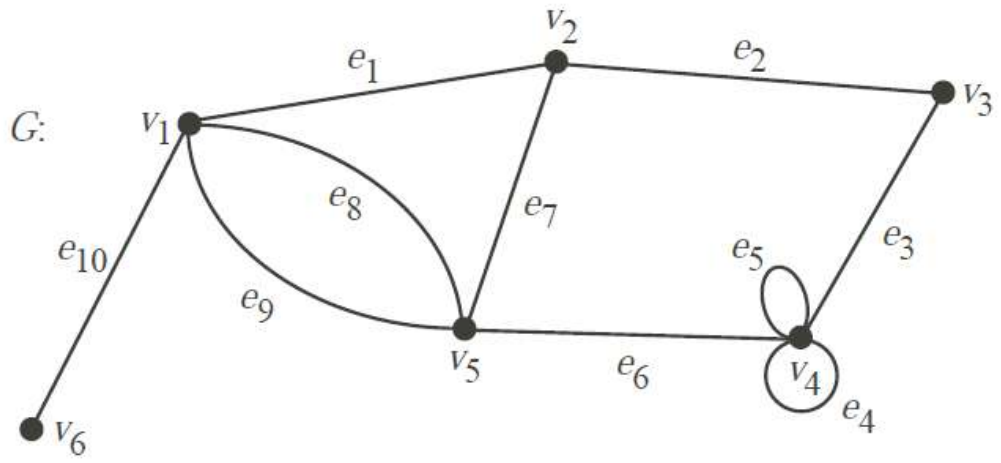


Fig. 2 Graph showing edges and vertices [23]

- A walk in this graph is a sequence which consists of alternating edges and vertices of G. The walk can start at any initial vertex and ends at terminal vertex. The number of vertices covered between initial and terminating vertex is called the length of the walk. If the walk terminates at the starting vertex then it is called closed walk and if the walk terminates at any other vertex then it is called open walk[22]. For example

  v2,e7,v5,e8,v1,e8,v5 is an open walk.

  v4,e5,v4,e3,v3,e2,v2,e7,v5,e6,v4 is a closed walk.
- Trail : A trail is a type of walk when the edges are traversed only once. For example

  v1,e8,v5,e9,v1,e1,v7,v5 is aTrail.
- Path : A trail called a path when each vertex is traversed only once. For example v2,e7,v5,e6,v4,e3,v3 is a Path.
- Circuit : The path that is closed is called a circuit i.e. it have same initial and final vertex. For example

v2,e7,v5,e6,v4,e3,v3,e2,v2 is a Circuit.

- Subgraph : A graph X1 =(V1,E1) is called the subgraph of X2 =(V2,E2) if
    1. V1 ⊆ V2 and
    2. Each edge of X1 is also an edge of X2.

1.3.2 Matrix

A matrix is a rectangular array of numbers or symbols or expressions that are arranged in the form of rows and columns. The items in the matrix are called elements of the matrix. The size of the matrix is the number of rows and columns included in it. A m-by- n matrix (m×n) have m rows and n columns the m,n are the dimensions of the matrix.[23]

For example a m×n matrix is

$$\begin{bmatrix} a11 & a12 & \cdots & \cdots & \cdots \\ a21 & a22 & \cdots & \cdots & \cdots \\ \vdots & \vdots & \ddots & \cdots & \cdots \\ \vdots & \vdots & \vdots & \ddots & \cdots \\ an1 & \vdots & \vdots & \vdots & anm \end{bmatrix}$$

The matrices having only single row are known as row matrices, and the matrices having only one column are called column matrices.

Matrix with equal number of rows and columns is called square matrix and the matrix with no rows or columns is called empty matrix.

- The Transpose of a matrix is another matrix created by replacing rows with columns and column with rows. The transpose of an m×n matrix **A** is the n×m matrix $\mathbf{A}^{\mathrm{T}}$ [24].

$$(\mathbf{A}^{\mathrm{T}})_{i,j} = \mathbf{A}_{j,i}.$$

$$\begin{bmatrix} 1 & 5 \\ 7 & 9 \\ 3 & 8 \end{bmatrix}^{T} = \begin{bmatrix} 1 & 7 & 3 \\ 5 & 9 & 8 \end{bmatrix}$$

- A Submatrix is the subset of a matrix and is obtained by deleting a row/column or a collection of rows/columns. For example, for the given 3×4 matrix, a 2×3 submatrix can be made by deleting a row 3 and a column 2 [24].

$$\text{Matrix A} = \begin{bmatrix} 1 & 4 & 7 & 5 \\ 6 & 9 & 2 & 1 \\ 5 & 3 & 4 & 8 \end{bmatrix}$$

$$\text{and its Submatrix is} \quad \begin{bmatrix} 1 & 4 & 7 \\ 6 & 9 & 2 \end{bmatrix}$$

1.3.2.1 Operations on matrices:

Only matrices of the same size can be added orsubtracted. But for multiplication, the number of columns of first should be equal to number of rows of the second i.e only a A= m×n matix can be multiplied to B= n×p matrix. The product of the matrix is AB a $m \times p$ matrix having items formed by taking the dot product of the rows of A and the columns of B [24].

$$[AB]_{i,j} = A_{i,1}B_{1,j} + A_{i,2}B_{2,j} + \ldots\ldots\ldots + A_{i,n}B_{n,j} = \sum_{r=1}^{n} \mathrm{A_{i,r}B_{r,j}}$$

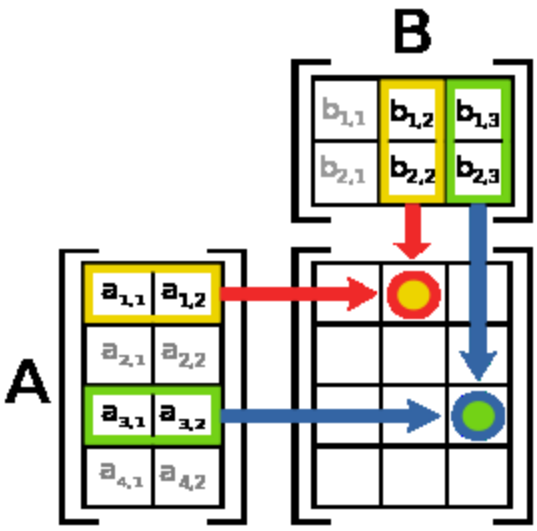


Fig. 3 showing the multiplication of the matrices [25]

The matrix product is not commutative in nature i.e AB is not equal to BA.

1.3.3 Determinant

The determinant is a well established term in the matrix algebra. The determinant calculated as the sum of products of the items of matrix with each product having n terms and coefficient of every product is 1,-1 or 0 taken by the rule: it's a polynomial expression of matrix items [25].

Let us consider a square matrix M with m rows and m columns so it can be shown as:

$$M = \begin{bmatrix} a11 & a12 & \cdots & \cdots & a1m \\ a21 & a22 & \cdots & \cdots & \cdots \\ \vdots & \vdots & \ddots & \cdots & \cdots \\ \vdots & \vdots & \vdots & \ddots & \cdots \\ an1 & \vdots & \vdots & \vdots & anm \end{bmatrix}$$

The terms present in the matrix can be both numbers or expressions. The determinant of the matrix can be added or subtracted or multiplied in the commutative manner. The determinant of M is denoted as det (M).

$$\det(M) = \begin{vmatrix} a11 & a12 & \cdots & \cdots & a1m \\ a21 & a22 & \cdots & \cdots & \cdots \\ \vdots & \vdots & \ddots & \cdots & \cdots \\ \vdots & \vdots & \vdots & \ddots & \cdots \\ an1 & \vdots & \vdots & \vdots & anm \end{vmatrix}$$

1.3.3.1 Physical meaning of determinant:

Let us consider a 2×2 matrix

$$M = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

then its determinant is $\begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc$

This can be represented by area of the parallelogram with vertices (a,b),(c,d),(0,0),(a+c,b+d). The matrix formed by these vertices gives the absolute value of the determinant of M. The determinant's absolute value including the signs becomes the oriented area of the parallelogram. The oriented area and simple area are same things except for the fact that oriented area is negative when angle between the

first and second vector (starting from first to the second) defining the parallelogram goes in the clockwise direction[25].

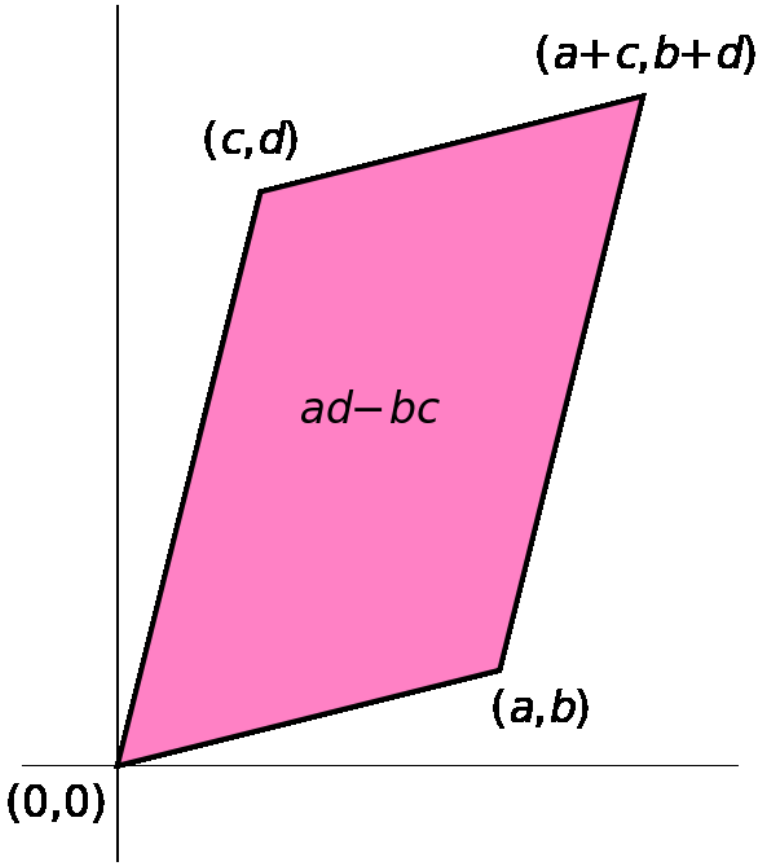


Fig. 4 Representation of the elements of a 2×2 matrix as vertices of the parallelogram [25]

Now let us consider a 3×3 matrix

A= $\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$ then its determinant is

$$\begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix} = a \begin{vmatrix} e & f \\ h & i \end{vmatrix} - b \begin{vmatrix} d & f \\ g & i \end{vmatrix} + c \begin{vmatrix} d & e \\ g & h \end{vmatrix}$$

= a(ei - fh) – b(di - fg) + c(dh - eg) = aei + bfg + cdh – ceg – bdi – afh.

The value of the determinant is calculated by the rule of the Sarrus. It's a mnemonic for 3-by- 3 matrix determinant: the sum of the products of the three diagonal north-west to south-east lines of matrix elements is taken and subtracted from the sum of the products of three diagonal south-west to north-east lines of entities and the copies of first two columns of the matrix are written beside it.

The volume of the parallelepiped constructed by the rows from vectors r1,r2,r3 gives the absolute value of the determinant.

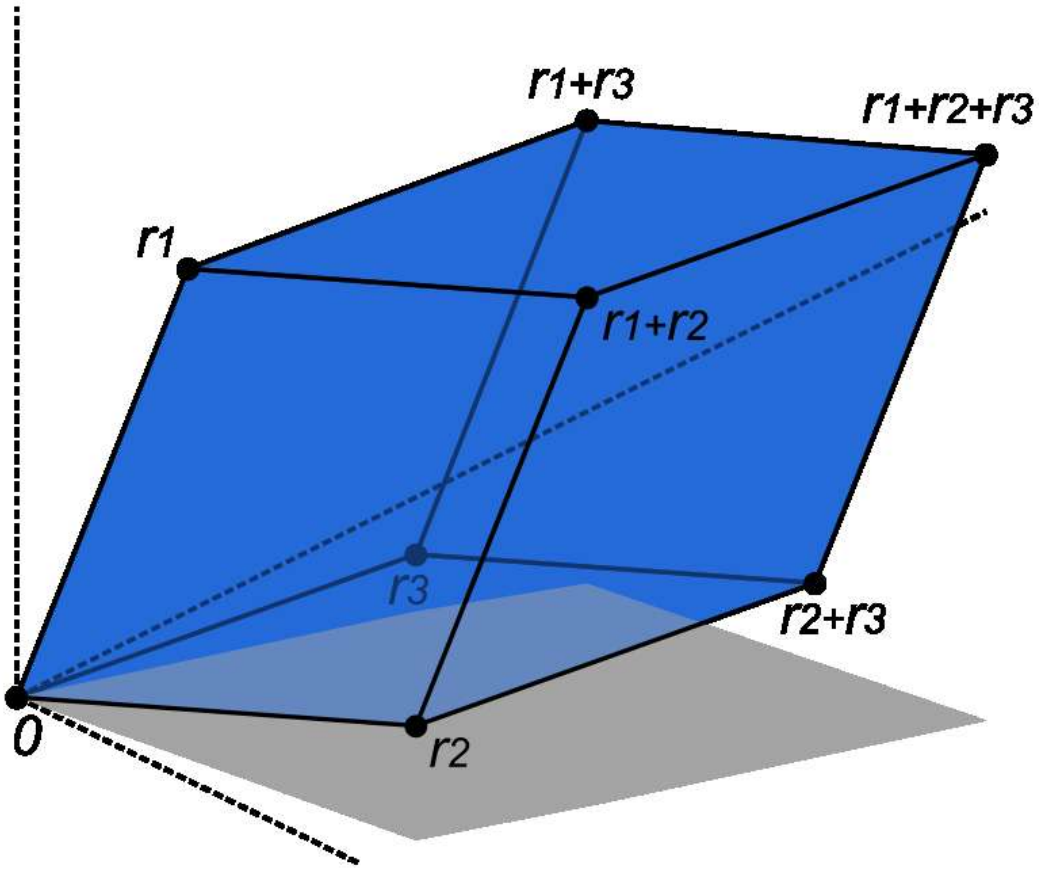


Fig. 5 Representation of the elements of a 3×3 matrix as vertices of the parallelogram [25]

1.3.4 Permanent

The permanent is a well known term in linear algebra. The word permanent was given by Cauchy in 1814 for interrelated functions. The permanent of the matrix can be viewed as a map that takes m entities as arguments. It is multilinear and symmetric in

nature. The permanent can be calculated for a square matrix. Some of its properties are:

- The permanent is not affected by the permutations of the rows and columns of the matrix.
- If we multiply a matrix with some scalar quantity s its determinant becomes s times the determinant of the previous i.e perm(M) changes to s.perm(M).
- The permanent of the matrix is same as permanent of its transpose.
- Perm(A)×perm(B) is not equal to perm(A×B).

The permanent of a m×m matrix M = $(a_{i,j})$ is defined as

$$\text{Perm}(M) = \sum_{\sigma \in Sn} \prod_{i=1}^{n} a_{i,\sigma(i)}$$

The sum is for all elements σ of symmetric group Sn i.e. over all the permutations of 1, 2, ..., n. e.g.

$$\text{Perm}\begin{pmatrix} a & b \\ c & d \end{pmatrix} = ad + bd \quad \text{and}$$

$$\text{Perm}\begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix} = aei + bfg + cdh + ceg + bdi + afh.$$

The permanent do not have simple geometric interpretation like that of determinant. It is used in combinatorial mathematics and also for treating the boson Green's function in quantum field theory. In graph theory it have two interpretations: the sum of weights of cycle cover of the directed graph and as the sum of weights of perfect matching in the bipartite graph[26].

The permanent of the matrix M is different from its determinant in the sense that permutations are not considered.

Both permanent and determinant are polynomial in the entries of the matrix. Both of these are the special cases of the term Immanent.

1.3.5 Immanent

The concept of the Immanent was brought by the Dudley Littlewood and Archibald Richardson. The immanent of a matrix is defined for a square matrix.

Let f: Sn $\rightarrow$ C is a complex character. For a m×m matrix M = $(a_{ij})_{i,j=1}^{n}$

the immanent of a matrix is defined as

$$\text{Imm(M)} = \sum_{\sigma \epsilon Sn} f(\sigma) \prod_{j=1}^{n} A_{j\ \sigma(j)}$$

When Sn is a constant character , the Imm(M) is called the permanent of M. When f is taken as the representer of the permutation (it is a character of the permutation results related to the (non-trivial) one-dimensional representation). The Imm(M) is considered as the determinant of M. The determinant is defined taking into account the parity of the permutation[27].

1.3.6 Multinomial Distribution

The multinomial distribution is the parent set of the binomial distribution i.e. it is the general term for the binomial distribution.

It is the collection of the probability distribution having many random variables that represents the amount of occurance of an exclusive event in the consecutive multinomial trial. It indicates the chances of getting the required result after the number of trials. As the binomial distribution can be viewed as 1D slice of the Pascal's triangle, the multinomial distribution can be viewed as 2D slices of the Pascal's pyramid.

The binomial distribution consists of the polynomial coefficient of the expansion of the term:

$$(px_1 + (1-p)x_2)^n$$

Similarly multinomial distribution is obtained after the expansion of the term:

$$(p_1x_1 + p_2x_2 + \cdots p_kx_k)^n$$

In the expansion of the multinomial distribution, the sum of the coefficients must be equal to 1.

Chapter 2 Literature survey

The Database application programs have a wide usage, despite that relatively very less work has been done in the direction of how to test these programs to help check their correctness. The functioning of the program is decided by the database state and the user's inputs. These type of programs invokes new problems for the software testers. The present techniques used by the software testers works the following way: they divide the input space into subsets (possibly overlapping) and then makes sure that all of these subsets is represented. A number of techniques are available to make this subdivision, few are dependent on the source code of the application under test, others are dependent on the specification, and some largely automated, and some relying on manual test specifications. To apply these approaches to testing database applications, one must take into consideration the database state and also the user's input.

2.1 Aspects for checking the correctness of DBMS:

1. Is the application working as desired?
2. Is database schema correctly reflecting the organization of real world data being modeled?
3. Are the security and privacy constraints fulfilled?
4. Is the database having the correct data?
5. Are all the operations, that is the insertion, deletion, updation performed correctly on the data?

The aspects discussed above along with other aspects of the system performance are very important to the organizations whose working depend on the DBMS.

2.2 Issues arising for testing Database Applications:

Various issues arise while testing the applications based on the database management systems. Consider a database including a customer-feature table having information about the customers, customer ID number, customer address, the services desired by the customers, a billing table, with customer ID numbers and the information

pertinent for giving bills, and also some of other tables. The features desired by the application are :

Input the customer ID and the phone service which is desired by the customer. If any of these two are invalid , code 0 is returned; else if the customer is residing in an area where the demanded feature is available and also there is no issue of incompatibility of that this service with any other service that is already subscribed by the customer, then add the service to the list of customer feature records. Update the billing table accordingly and notify the in-charge department. The department will provide the desired service; return code 1. If the customer is not residing in the area where the desired service available, return code 2; If the customer is residing in the area where the service is available but this service is not compatible with the other services already subscribed by the customer; code 3 will be returned.

- The role of the database state: A database application is like a partial function with input set I and output set O. But the sense of database state (the role of database state) must also be taken into account. The specification is seen as a function from I to O. So for testing database state, values must be selected from I, executing the application for these, and finally checking whether the result of O match with the specifications. In traditional programs seen number of times in the testing literature and also in many existing tools, the input and output space have a complex outlook. This leads to difficulty in selecting values for the test cases and also in checking the results.

We can deal with the database states in many ways:

i) We can ignore the database state that is the application can be viewed as the function relating the user's input with the user's outputs. But practically it is not feasible because its mapping would be non-deterministic and making it very difficult to validate and re execute the test results.

ii) We can assume the database state as a feature of the environment under which the application is running and also include the role of the environment to find the expected and realistic results.

We have to treat the database state as part of input as well as output state.

- Choosing the desired database states form the available states: This means we have to populate the database with only the meaningful values those are obtained by interaction of different tables. It could be possible to generate one database state involving many customers having variety of features. Let there be 2 tables table T1 showing the incompatible features and the table T2 show the feature available in different areas. Let each row of T1 shows incompatible services and the rows of T2 shows a zip code and the feature associated with it. For making a good initial database state we need to have the record of the customers which have currently subscribed to the to the feature X and lives in the zip code Z. (X,Z) is present in T1 and (Z,Y) is present in T2,showing that the feature X and Y are incompatible and the feature Y is present in Z. When the customer demands feature Y, the permission should be denied by the application because even if the feature Y is available in customer's area but its not compatible with the feature X which has been already subscribed by the customer.
- Keeping a check on the database states after the execution of the operation performed:One aspect of maintaining the integrity is to check whether the operation done on the application is correctly performed? If every query have run correctly has customer been added to the table of the persons desiring the service ? have the database billing table been updated? If the service is not available to the customer or there is conflict with the service already possessed by the customer is the database is attaining its previous state?
- Deciding the type of data (LIVE OR SEMANTIC) used to populate the database: The data practically needed for the testing purpose is the live data i.e. the data actually present in the database at the time of the testing database application. An approach is suggested to generate that data which can be used specifically for testing and running the tests in the suitable environment. As database state is the combination of the relation states which are the subsets of the Cartesian product of the domains, so all we have to do is to generate the values for testing and glue them accordingly to make tables. But while doing this an important fact is underestimated that is the integrity constraints. The database needs to be populated not just by any tables but by the tables that have suitable and consistent data. In a consistent database state all constraints

that were specified are followed. If the integrity constraints have to be maintained, database state is filled with arbitrary data, so the data is rejected in which the constraints are not satisfied. But instead one tries to generate the suitable data and then fill it in the database. The data values that shows faults or loopholes should be selected. To ensure the consistency of the selected data, the database schema can be used. This information in the database schema is expressed using formal language, SQL's Data Definition Language (DDL), which makes it possible to automate much of the process.

- Generating synthetic data: To use the live data for the purpose of testing is very difficult and tedious. We want to generate data specifically for testing and also to run these in an isolated environment. The database states are made up of the relation states. The approach that states to generate values from the domains and joining them together to make tables have a problem. That is in this approach it is ignoring an important aspect that is to maintain the integrity constraint. Populating the databases with any type of data is not desired in fact it should be filled with the data that is interesting and valid. The database should reject the data that does not satisfy the integrity constraints.

In earlier work, a group developed a prototype testing tool AGENDA, A test GENerator for DAtabase Applications [1]. AGENDA works by populating a database with data, generating inputs to the application under test, and checking few aspects of the correctness of database updates performed by the application

2.3 AGENDA overview:

A tool AGENDA have been developed for testing the database applications. The input to this tool are: the database schema on which the application is running, the source code of the application, and the sample value files having some expected value of the attributes. The user gives the test heuristics and also gives information about the behavior of the test cases. The information provided thus is now used by the AGENDA to fill the database, provide input values for the applications, execute the application program for those input values and also check the some parameters for the correctness of the database and output of the application.

The technique used in this is based on the Category-partitioning method [24], that is the user provides the attribute values that are divided into different groups called data groups. The data is made available in the form of sample-value files. The AGENDA makes some meaningful combinations of the data and fill the database tables and also provide input parameters to the application program. The data groups are separate values that result in altering of the application behavior. Making use of the data groups and the heuristics provided by the user, the AGENDA fill the database and generate the group of test templates showing the abstract test cases. Now the expected behavior of the application under test is provided by the tester using the templates. To control the number of test cases generated and to make sure that the templates of only certain kind is produced, the tester selects the heuristics. When all this is done, AGENDA fill the templates with some specific test values, executes these test cases and check that the state of the database and the output are matching with the desired results.

AGENDA is made up of the five components that work by the guidance of the tester. The components are:

- The parser: It extracts the useful information from the database schema of the application, and provide this data to the other four components. It is done by making an internal database called AGENDA database which stores the extracted data. The AGENDA database is modified by the other four components.
- The state generator: This component makes use of the database schema and some other information about the sample-value files indicating useful values for the attributes and then fill the database with the data that maintain the integrity constraints. The information about the database tables is extracted from the AGENDA database and an initial database state for the application is generated called Application database.
- The input generator: This component produces the input data that will be used by the application. This data is produced by using information produced by AGENDA parser and state generator and some additional information derived from parsing the SQL statements. By making use of the AGENDA database and the heuristics provided by the user, the input generator instantiates the

input parameters of the application with original values, hence creating the test inputs.

- The state validator: This component keeps a check on how the state of the database application is modified after execution of the test. It automatically makes the respective changes in the application tables semi-automatically keeps a check on state change.
- The output validator: This component validates the output that is it compare whether the output produced matching with the pre and post conditions that have been specified.

In the update query the database state of the application is altered so the working components are: The parser, The state generator, The input generator, The state validator. In the select query, the database state is not affected, so the working components are: The parser, The state generator, The input generator, The output validator.

2.4 Test Generation Query Generator (TGQG)

The TGQG is a tool that generate the queries that are used for choosing values for input generation of the application satisfying the type A requirements and also the templates having the combination of necessary data groups including the parameter location (i.e. WHERE and SET/VALUES clauses), the constraints to be obeyed like uniqueness constraint or referential constraint etc and also the ease and comfort to handle the limitations of the older approaches. The algorithm for the tool TGQG is put forward

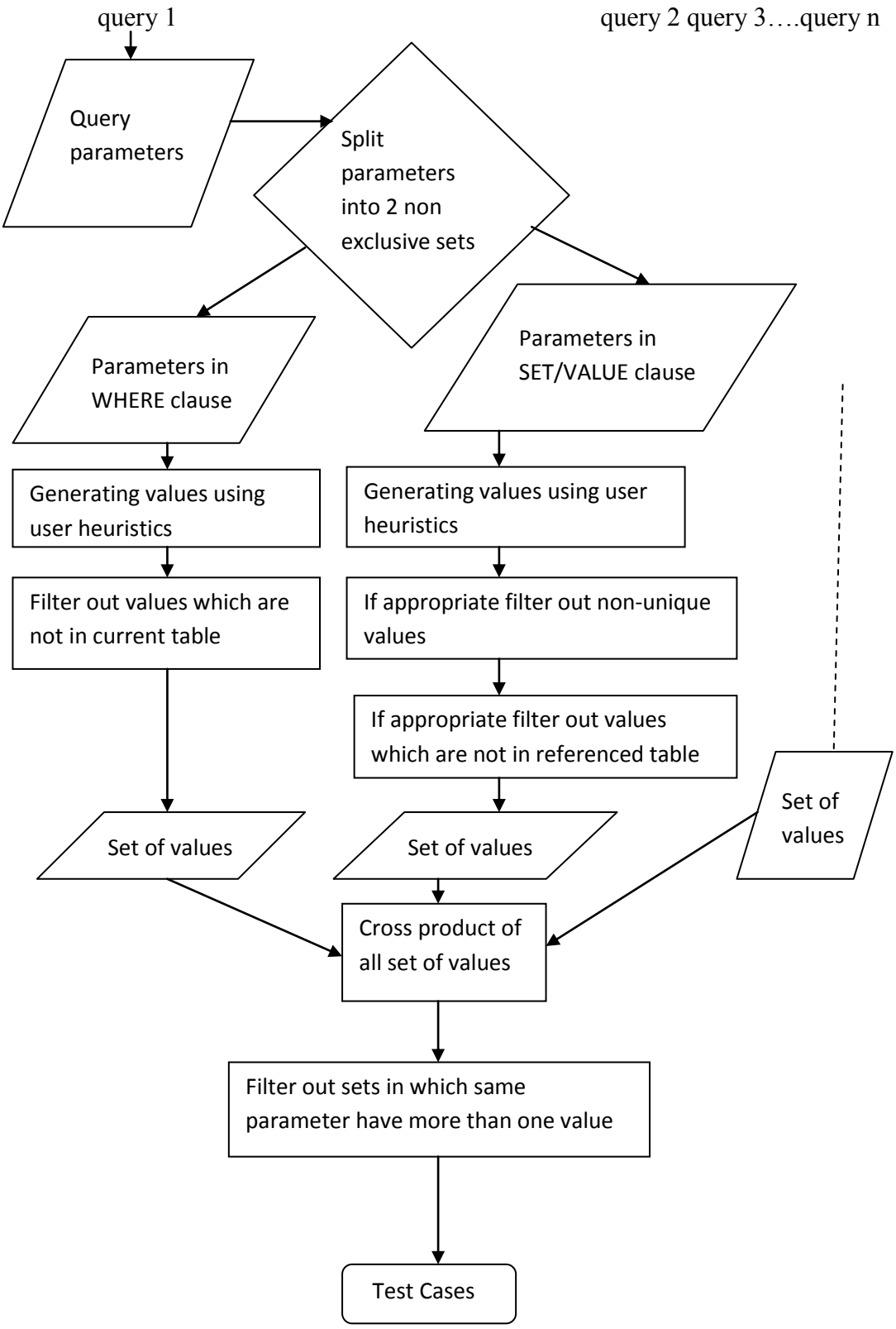


Fig. 6 Arrangement of the number of queries according to their clause location in the statements [28]

2.4.1 How to generate the queries

We take an example for this purpose. This example will show how the TGQG generate the actual queries.

Here is a schema:

- The elements of the table t1 are a1, a2…..a7.
- The table t1 also have composite uniqueness constraints on a1, a2, a3.
- It also have composite referential constraint on a1, a2 referring to table t2.
- Absence of constraints on any other element of t1.

Thus the application query is:

UPDATE t1

SET a1 = :in1, a2 = :in2, a3 = :in3 + :in4,

a5 = a5 * :in5

WHERE a6 = :in6 and a7 = :in7 / 100

The input file given to the TGQL contain the information about the context where the parameters are present in the form of the queries. The information like the parameter names, their data groups, the attributes of the table are collectively called the parameter value recs.

2.4.1.1 In case of WHERE clause :

The TGQG generate similar queries for the parameters in same WHERE clause for example :in6 and :in7 in the application queries have similar generation of queries. The query generated by the TGQG is:

1. Initialize: generatedQuery = "SELECT ".
2. Add a list of temporary variables to the generatedQuery for all the parameters of the same WHERE clause like temp1 for :in6 and temp2 for :in7 .
3. Add " FROM (".
4. The operations performed on all the temporary variables temp i in accordance to the parameters in WHERE clause are:

a) Add "(SELECT value AS ".

b) Add "temp$i$".

c) Add " FROM parameter value recs WHERE parameter name=".
d) Add the suitable parameter name (e.g. :in6).
e) Add "' AND group name = '".
f) Add the name of the group with respect to template.
g) Add "')".
h) Add " CROSS JOIN " if more parameters are there.

5. Add ") WHERE EXISTS (SELECT * FROM ".
6. Add the table names those are included in the application query (for example t1).
7. Put temporary variables in the place of the parameter names (for example. "WHERE a6 = temp1 and a7 = temp2 / 100") and add it to the WHERE clause.

The TGQG generate query for the parameters in WHERE clause. The generated query is as follows:

```
SELECT temp1, temp2 FROM (
  (SELECT value AS temp1 FROM parameter_value_recs
   WHERE parameter_name = ':in6' AND group_name = ?)
  CROSS_JOIN
  (SELECT value AS temp2 FROM parameter_value_recs
   WHERE parameter_name = ':in7' AND group_name = ?)
)
WHERE EXISTS (
   SELECT * FROM t1
   WHERE a6 = temp1 AND a7 = temp2 / 100
)
```

The name of the group is chosen by the AGENDA Squencer, that also takes care of the data groups that were represented.[25].

All the data groups and their combinations can be represented: if the number of the combinations are very large then using combinatorial testing technique, a subset is chosen.[26].

2.4.1.2 In case of SET/VALUE clause:

The TGQG generate similar queries for the parameters in same SET/VALUE clause for example :in1, :in2, :in3, :in4, :in5 in the application queries have similar generation of queries.

The query generated by the TGQG is:

1. Initialize: generatedQuery = "SELECT ".
2. Add a list of temporary variables to the generatedQuery for all the parameters of the same SET/VALUE clause like temp1 for :in1 and temp2 for :in2.
3. Add " FROM (".
4. The operations performed on all the temporary variables temp i in accordance to the parameters in WHERE clause are:

a) Add "(SELECT value AS ".
b) Add "temp$_i$".
c) Add " FROM parameter value recs WHERE parameter name=".
d) Add the suitable parameter name (e.g. :in6).
e) Add "' AND group name = '".
f) Add the name of the group with respect to template.
g) Add "')".
h) Add "CROSS JOIN" if more parameters are there.

5. When a table is updated, if all attributes of the uniqueness constraints have input parameters like a1,a2,a3 etc. then add to each constraint a WHERE NOT EXIST clause that filter out values present in the same tuple in the application database state.
6. When a table is updated, if all attributes of the uniqueness constraints have input parameters like a1,a2,a3 etc. then add to each constraint a WHERE EXIST clause to make sure that the referential constraints are satisfied.

In steps 5 and 6 if the expression containing SET clause is like "attribute =<non-trivial expression involving input parameters>". For example "a3 = :in3 + :in4", and the attribute obeys uniqueness or referential integrity constraint, then the expression will be present in the NOT EXISTS or EXISTS clause.

For the parameters of the SET clause, TGQG generates the following query:

SELECT temp1, temp2, temp3, temp4, temp5 FROM (

```
  (SELECT value AS temp1 FROM parameter_value_recs
   WHERE parameter_name = ':in1' AND group_name = ?)
  CROSS_JOIN
  (SELECT value AS temp2 FROM parameter_value_recs
   WHERE parameter_name = ':in2' AND group_name = ?)
  CROSS_JOIN
  (SELECT value AS temp3 FROM parameter_value_recs
   WHERE parameter_name = ':in3' AND group_name = ?)
  CROSS_JOIN
  (SELECT value AS temp4 FROM parameter_value_recs
   WHERE parameter_name = ':in4' AND group_name = ?)
  CROSS_JOIN
  (SELECT value AS temp5 FROM parameter_value_recs
   WHERE parameter_name = ':in5' AND group_name = ?)
)
WHERE NOT EXISTS (
  SELECT a1, a2, a3 FROM t1
  WHERE a1 = temp1 AND a2 = temp2 AND
        a3 = temp3 + temp4
)
AND EXISTS (
  SELECT a1, a2 FROM t2
  WHERE a1 = temp1 AND a2 = temp2
)
```

The last filtering step is a follows:

- For a parameter present in WHERE clauses of more than one query, generate a query using the given 7 steps of the first algorithm, then join the queries which have same common parameters.
- For a parameter present in SET/VALUE clauses of more than one query, generate a query using the given 1-6 steps of the second algorithm, then join the queries which have same common parameters.

- For a parameter present in WHERE clause and SET/VALUE clause of different queries, generate a query each for the first algorithm and second algorithm, then join the queries which have same common parameters.

2.5 Finding the structural and interaction features

The David's [2] approach deals with automatically generating queries for test generation based on SQL statements in transaction under test. Fig.6 show how input parameters of a no. of queries are arranged according to their clause location in statements. The input parameters of the queries are split into 2 sets (could be non-disjoint): parameters in WHERE clause and parameters in SET or VALUES clauses.

The parameters in the SET or VALUES clause are split into two sets (could be non-disjoint) : those associated with unique attributes and those associated with referential attributes. These features in input parameters are needed by the algorithms that produces type A test cases.

The test generation query that AGENDA produces for a given application SELECT query with two inputs, input_eid and input_did is shown (Fig. 7). A table is returned by the generated query with each row representing a (eid,did) pair in the given data groups, and producing a test case for the application query.

Suppose the two groups are specified for parameter input_eid, "part-time", and "full-time", and two groups for parameter input_did, "foreign" and "domestic", then its given that template for each combination of data groups is desired, the queries will be generated to produce test cases for every combination.

The Fig. 7 shows query generated for the combination of the "part-time" group for eid and the "foreign group" for did. Further, to satisfy the Type A requirement, we should further constrsin the choice to select an (eid,did) pair that satisfies the WHERE clause of the application query. This is done by adding a WHERE EXISTS, clause to the generated query followed by essentially the original SELECT query, with temporary variables as inputs, shown in Fig. 7.

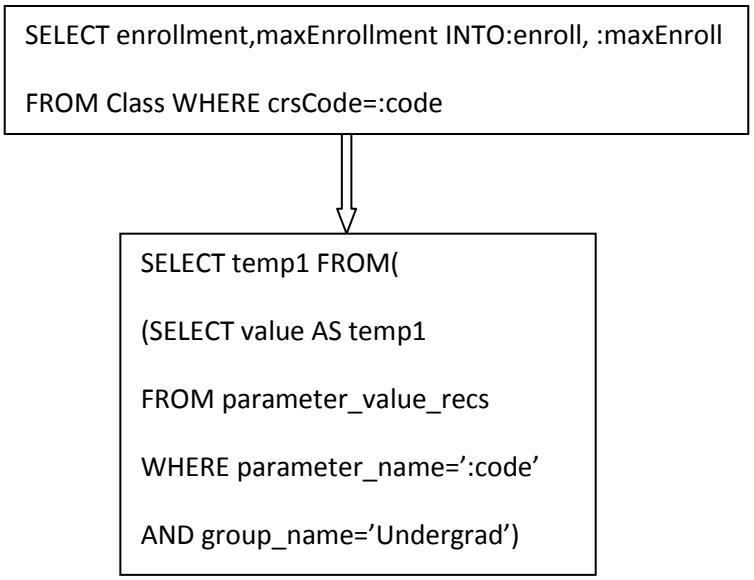


Fig. 7 Input parameters of multiple queries

Chapter 3 Problem statement

While experimental evaluation of the prototype made above showed promising results, it was observed that the initial input generator produces the test values that were not useful, either because they caused queries to return empty result sets or because they caused updates to violate the given constraints. David [2] etc. describes an improved approach for the input generation in which the SQL DML statements in the application and some additional information, are used to generate test generation queries automatically. By executing these queries we get the inputs values in which the above problems were reduced considerably and in addition also satisfy some other desirable properties. The test generation system obtained having the required properties largely the outcome of the parameters in WHERE clauses and also the cross-product of all values. A suitable value based on the heuristics, and design aspects of the test generation system is chosen. Some work had been done [3-7] on the interactions between the parameters in WHERE clauses/generate values using heuristics. Till now no publications [8-10] have taken into account the structural constituents [26] and analysis of test generation system.

From these studies, it can be observed that no one has considered the structural aspects for evaluating a test generation system and the engagements of it at the conceptual and design phases, above that no theory have been given taking integrated systems approach in evaluating test generation system i.e. by taking into account the subsystems as the constituents and their engagements, interdependence and interconnections. A mathematical model is needed for analysis and synthesis of test generation system at the conceptual stage and to develop a virtual model for that. The features of test generation system are combined features of the properties of the constituents and the properties of the inter and intra dependencies among them when running the final product. So an integrated model need to be developed for the designing and examination test generation system.

Chapter 4 Proposed solution to the problem

The work done in this thesis focuses on developing an integrated systems model for the structure of the query based test generation system in terms of its components and interactions between the components making use of the graph theory and also the matrix algebra. A test generations system is first constructed using graph theory, then modeled using variable adjacency matrix and after that using multinomial called permanent function. The permanent function enables us to perform the structural analysis of the test generation system taking into account the program running time and the storage optimization by relating the test generation system features with its design. A physical meaning has been assigned to every term of the permanent function. The structural attributes of the test generation system are found and a graph theoretic model is developed, a matrix model and also a multinomial permanent model are developed. A top-down method for complete analysis of any test generation system is also illustrated. The features of present approach is also given.

Digraph models provides a well-structured system approach and this is used in no. of areas of modern day technology [14-18]. The work done so far is derived from different fields of research such as quantitative structure activity relationship (QSAR) and quantitative structure properties relationship (QSPR), virtual prototyping [19], and also a lot of research had been carried out on the test generation system.

The constituents of the test generation system product and also the amount of interactions present in the various constituents should be identified [8-10]. From this study it can be observed that to model the finished test generation system product mathematically, all these attributes, those are contributing in the final product should be identified. All the structural attributes are identified and are arranged under five subgroups as shown in Fig. 8.

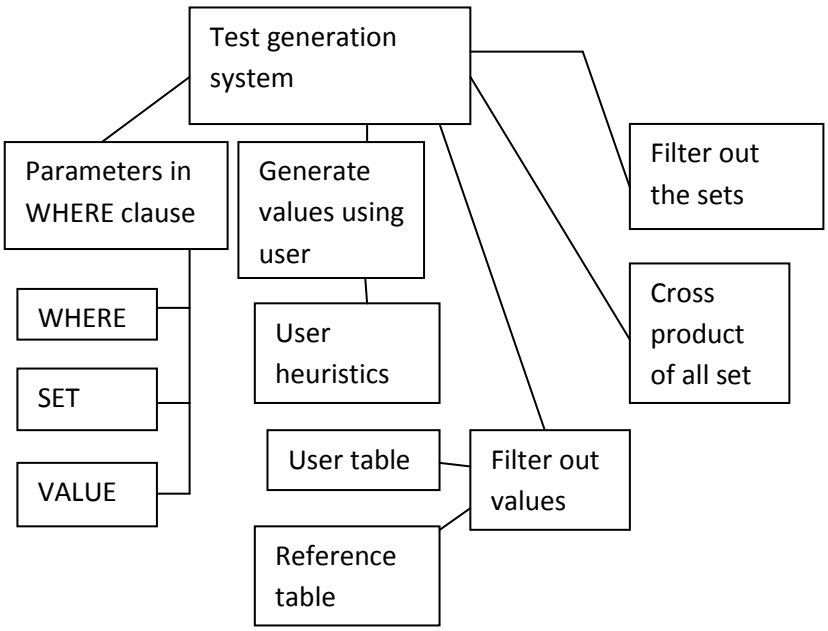


Fig. 8 Arrangement of the structural attributes

The generic categorization is ultimately reduced to Fig. 9.

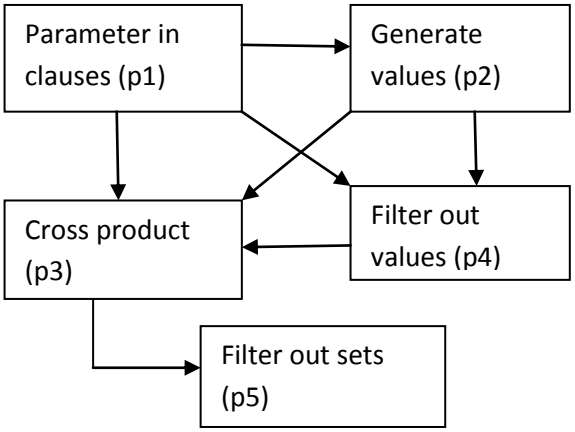


Fig. 9 The generic categorization

4.1 Graph Theoretic Representation

A test generation system is basically the many basic structural constituents put together, e.g., Parameters in WHERE clauses, Generate Values using User Heuristics etc. (Fig. 9).

Different forms of bonding and interactions connects these constituents with each other. Fig. 8 shows the constituents and interactions contributing to a test generation system. Blocks corresponding to constituents, lines to the interactions, and arrows representing directional interactions. Even though schematic diagram is an efficient way of representing a test generation system for a good understanding of its structure but its not a mathematical entity. So it is difficult to derive different results as mathematical operations cannot be carried out. Therefore mathematical representation of test generation system attributes is proposed in order to develop a systems model for examination and development of the test generation system, which will enable us to carry out the inspection of an existing product for improvement and cost reduction in addition to other advantages that it provides.

A test generation system is considered to be a system [S, R] with constituent set {S} = {S_1,S_2, …..Sn} and connectivity set {R} = {R_1, $R_{2,...}$ R_n}, Si is the ith constituent (and features included in it) Rj is the jth connectivity present among the two constituents of the test generation system. As far as the system representation of the test generation system is concerned, it will be suitable using graph theory [35] and matrix algebra for the mathematical representation [19] (Fig. 7).

A graph G[V, E] is a set of vertex and edges with vertex set {V} = { V_1 $V_{2.....}$$V_n$ } and edge set {E} = {e1,e2…en} an G have these as subsets. For the mathematical representation of test generation system, let the vertex set V represent its constituents and edge set E represent the connectivites and the directional properties are shown by directed edge.

4.2 Developing Algorithm

Let us suppose five constituents {S1,S2,S3,S4,S5} form a test generation system(Fig. 8) represented by five vertices {v1, v2, v3, v4, v5} i.e., S1 is represented by the vertex v1,S2 by v2 and so on. Connectivity between Si and Sj, i.e., between vi and vj is

represented by edge eij (edge joining vi and vj). If all five constituents are interacting with each other and have general directional characteristics, then test generation system have graph theoretic representation in which eij !=eji. The eij !=eji implies that the influence of ith vertex on jth vertex and influence of jth vertex on ith vertex is not equal. If the directional property is not significant, the test generation system consists of an undirected graph, with eij=eji.

But a case can occur in some test generation systems, in which some of the constituents have no connectivity with each other. So no edge will be present between the vertices in that case. In some of the test generation system, there is no directional interaction present between two constituents,i.e., vertices, those edges are undirected and other edges which have interactions/connectivity will be directed. So a graph theoretic representation of a test generation system could have both the directed and undirected edges.

A test generation system essentially consists of five subsystems: (i) Parameters in where clauses,(ii) Generate Values using User heuristics, (iii) Fill out Value, (iv) Cross Product of all set of values, and (v) Filter out the sets. The influence of the subsystems on each other is illustrated in Fig. 8. Generate values using user affects the structure of the product, the structure also affect the prior but it is not that significant. The diagram shown in Fig. 8 (non-mathematical entity), is represented by a mathematical entity called test generation system digraph (Fig. 10) (digraph representation of the test generation system design).

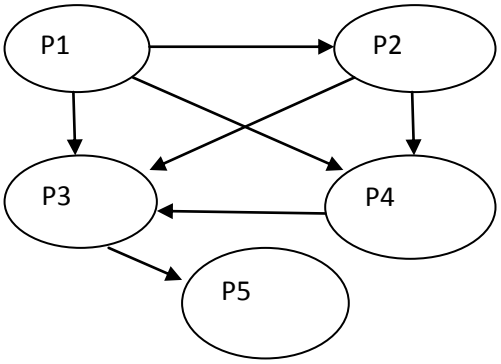


Fig. 10 The test generation system diagraph

Fig. 11 shows non-directional relationship between the constituents of the test generation system. Equation (6) gives the permanent function and Fig. 12 shows graphical representation of the associated terms. The five subsystems given in the figure represent the connectivity only. The degree of influence and degree of interactions(unidirectional or bi-directional) cannot be shown in an undirected graph as Pij=Pji.

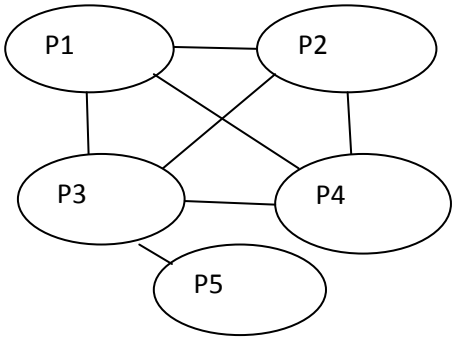


Fig. 11 Non-directional relationship between the constituents of the test generation system

4.3 Matrix Representation

As diagraph gives diagramatic representation only the mathematical analysis cannot be done. To make a computer understandable representation of test generation system, matrix is used for representation of the digraph. The computer processing becomes easy by this [36].The matrix representation allows storage, retrieval and examination of test generation systems.

Let us consider a digraph with 'n' subsystems having nth-order symmetric (0, 1) matrix A=[Pij]. The rows and columns in the matrix are the subsystems attributes, i.e., Pij showing ith subsystem's relation with the jth subsystem.

Traditionally Pij =Pji as test generation attributes are not undirected and Pii=0, because a subsystem cannot have an interaction with itself (in some cases it may be possible,but for now lets assume it cannot have). The test generation system matrix is a non-symmetric square matrix and can be used in the place of adjacency matrix in graph theory. The test generation system matrix (CSM) representing the digraph is given in Fig. 10 :

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Pij = {1, if i is connected to j

0, otherwise:

} **(1)**

In matrix A, the values of Pii, P_{15}, P_{25}, P_{15}, P_{45}, P_{51}, P_{52}, P_{53} are assumed to be zero. The non-diagonal elements with value 0 and 1 shows the interdependency and connectivity between subsystems, Pi, i=1, 2, . . . , 5 of the test generation system. The value of the diagonal elements are 0 because there is no connectivity/interaction of a subsystem with itself. For identification/characterization of the test generation system, a characteristic matrix is introduced.

4.4 Test generation System Characteristic Matrix (CM-TEST generation)

Take an identity matrix I, and a variable P to represent the test generation system. The characteristic matrix used in mathematics [19] is now used to represent/identify the test generation system. Fig. 10 shows the test generation system characteristic matrix B, for the digraph. The characteristic matrix B can be expressed as [PI-A], with A being the CSM shown in Equation (1).

$$B = \begin{bmatrix} p & -1 & -1 & -1 & 0 \\ -1 & p & -1 & -1 & 0 \\ -1 & -1 & p & -1 & 0 \\ -1 & -1 & -1 & p & -1 \\ 0 & 0 & p & -1 & p \end{bmatrix}$$

We have identity matrix I and P as the variable for the test generation system, then

B= [P-I] where A is the CSM. **(2)**

In matrix B, the value of all diagonal elements is the same (i.e., all test generation subsystems are assumed to be identical which may or may not be true in practice, since all test generation subsystems have different characteristics depending on various parameters affecting them). Interdependencies between the subsystems have been assigned values of 0 and 1 depending on whether it is there or not. This does not

represent varying degree of influence of one subsystem over the other subsystems. To consider this, another matrix called the test generation system variable characteristic matrix is proposed.

4.5 Test Generation System Variable Characteristic Matrix (VCM-TEST generation)

The test generation system variable characteristic matrix takes into account ,the effect of different test subsystems and their degrees of connectivity.

The digraph in Fig. 10 is used to define VCM-TEST GENERATION. Lets nodes are shown by Pis and edges by Pijs. Take a square matrix C with non-diagonal elements Pij showing connectivity between the test generation subsystems, i.e., in place of 1 (as in Equation (1)), a different matrix D is considered with diagonal elements P1,P2,P3,P4,P5 the Pi's represents the five subsystems as given in Equation 3. As the systems are all discrete, Pi gives the varying degree of inheritance of structural attributes of the subsystems. Let us consider matrices C and D,

VCM-TEST GENERATION is given by H=[D-C]

$$H = \begin{bmatrix} p & -p12 & -p13 & -p14 & 0 \\ -p21 & p2 & -p23 & -p24 & 0 \\ -p31 & -p32 & p3 & -p34 & 0 \\ -p41 & -p42 & -p43 & p4 & -p45 \\ 0 & 0 & 0 & -p54 & p5 \end{bmatrix} \quad \textbf{(3)}$$

● The test generation system variable matrix consider both the effect of different test generation subsystem and their varying degree of interaction.

● The information can be used for examination, designing and generation of new test generation system product which can be used for optimization purposes.

The matrix H, allows us to portray all the information about the five subsystems and also the connectivity between them which can have any industrial use test generation product.

The determinant of a matrix gives us a useful tool, called the variable characteristic test generation multinomial (VC-TEST GENERATION). This gives the characteristic of the whole system and also shows the complete test generation product system,

taking into consideration the impact of test generation subsystems and their connectivity. As we take into account the selective interaction between the test generation subsystems (as per Fig. 8), some diagonal elements in the matrix, H, i.e., in Equation (3), becomes zero.

There are positive and negative symbols associated with few terms of the determinant of H(the variable characteristic test generation multinomial). The terms of the multinomial contains all the information associated with the test generation product system. Replacing Pi and Pij with their respective numerical values can lead to loss of information. Therefore the complete information in the test generation system cannot be retrieved as there will be loss of information due to the addition and subtraction of numerical values of the diagonal and non-diagonal elements (i.e., Pis and Pijs). Thus the multinomial in Equation (3), is not a source of the complete information regarding the test generation system under given conditions.

To prevent the loss of structural information in the mathematical processing, a test generation system variable permanent matrix (VPM-TEST GENERATION) is introduced.

4.6 Test Generation System Variable Permanent Matrix (VPM-TEST generation)

In order to get a distinct and complete model of test generation product system represented by Fig. 8 and a digraph (Fig. 10), some other terms permanent and permanent matrix, regularly used in combinatorial mathematics is introduced. Consider a permanent matrix of five-subsystem test generation system product defined as:

$$E = \begin{matrix} p1 & p12 & p13 & p14 & 0 \\ p21 & p2 & p23 & p24 & 0 \\ p31 & p32 & p3 & p34 & 0 \\ p41 & p42 & p43 & p4 & p45 \\ 0 & 0 & 0 & p54 & p5 \end{matrix} \quad \textbf{(4)}$$

The above is the basic representation in matrix form for a test generation product which was sketched using five-subsystems. So the VPM-TEST generation for the five

subsystems represented in digraph form in Fig. 8 shown by VPM-TEST generation as:

The Pi's , i=1,2,…,5 are the diagonal elements showing the role played by the five critical subsystems in producing a test generation system product. The inter connectivity of the subsystems are shown by the non-diagonal elements. This model representation throws light at the role of each subsystem and its connectivity quantitatively and the loss of information (multinomial form) is also avoided.

4.7 Permanent Function Representation

The main limitation of digraph and matrix representations is that they do not give the guarantee of uniqueness as these models are liable to be changed if the nodes are modified. To restrain the uniqueness constraint and to generate a labeling independent test generation system product, a permanent function of the matrix VPM_TEST GENERATION is introduced. There is a standard matrix function referred as Permanent which is used in the combinatorial mathematics [19, 20]. It is extracted from the matrix same way as the determinant. The only difference is that the negative signs appear when determinant is expanded but when permanent is expanded, positive signs are replaced by the negative. The process result gives a multinomial (shown in Equation (6)) having different terms each having some physical significance corresponding to the test generation system. The multinomial restores all the information corresponding to the constituents as subsystems and connectivity between them.The variable permanent function for a test generation product modeled as five subsystem test generation product system extracted from the matrix (Equation (4)) is given.The variable permanent function for general test generation product system is:

$$\begin{aligned}\text{Per (p)} = {} & P_1P_2P_3P_4P_5 + (P_1P_2P_3P^2_{45} + P_1P_2P_4P^2_{35} + P_1P_2P_5P^2_{34} + P_1P_3P_4P^2_{25} + \\ & P_1P_3P_5P^2_{24} + P_1P_4P_5P^2_{23} + P_2P_3P_4P^2_{15} + P_2P_3P_5P^2_{14} + P_2P_4P_5P^2_{13} + \\ & P_3P_4P_5P^2_{12}) + (2\ P_1P_2P_{34}P_{45}P_{53} + 2\ P_1P_3P_{24}P_{45}P_{52} + 2\ P_1P_4P_{23}P_{35}P_{52} + \\ & 2\ P_1P_5P_{23}P_{34}P_{42} + 2\ P_3P_4P_{12}P_{25}P_{51} + 2\ P_3P_5P_{12}P_{24}P_{41} + 2\ P_2P_3P_{14}P_{45}P_{51} + \\ & 2\ P_2P_5P_{13}P_{34}P_{41} + 2\ P_2P_4P_{13}P_{35}P_{51} + 2\ P_4P_5P_{12}P_{23}P_{31}) + P_1P^2_{23}P^2_{45} + \\ & P_1P^2_{24}P^2_{35} + P_1P^2_{25}P^2_{34} + P_2P^2_{45}P^2_{13} + P_2P^2_{14}P^2_{35} + P_2P^2_{15}P^2_{34} + \\ & P_3P^2_{12}P^2_{45} + P_3P^2_{14}P^2_{25} + P_3P^2_{15}P^2_{24} + P_4P^2_{12}P^2_{35} + P_4P^2_{13}P^2_{25} + \\ & P_4P^2_{15}P^2_{23} + P_5P^2_{12}P^2_{34} + P_5P^2_{13}P^2_{24} + P_5P^2_{14}P^2_{23}) + (2\ P_1P_{23}P_{35}P_{54}P_{42} + \\ & 2\ P_1P_{23}P_{34}P_{45}P_{52} + 2\ P_1P_{24}P_{43}P_{35}P_{52} + 2\ P_2P_{13}P_{34}P_{45}P_{51} + \\ & 2\ P_2P_{14}P_{43}P_{35}P_{51} + 2\ P_2P_{13}P_{35}P_{54}P_{41} + 2\ P_3P_{12}P_{24}P_{45}P_{51} + \\ & 2\ P_3P_{14}P_{42}P_{25}P_{51} + 2\ P_3P_{12}P_{25}P_{54}P_{41} + 2\ P_4P_{12}P_{23}P_{35}P_{51} + \\ & 2\ P_4P_{13}P_{32}P_{25}P_{51} + 2\ P_4P_{12}P_{25}P_{53}P_{31} + 2\ P_5P_{13}P_{32}P_{24}P_{41} + \\ & 2\ P_5P_{12}P_{23}P_{35}P_{51} + 2\ P_5P_{13}P_{34}P_{42}P_{21}) + (2\ P^2_{12}P_{34}P_{45}P_{53} + \\ & 2\ P^2_{13}P_{24}P_{45}P_{52} + 2\ P^2_{14}P_{23}P_{35}P_{52} + 2\end{aligned}$$

$P^2_{15}P_{23}P_{34}P_{42}$ + 2 $P^2_{23}P_{14}P_{45}P_{51}$ + 2 $P^2_{24}P_{13}P_{35}P_{51}$ + 2 $P^2_{25}P_{13}P_{34}P_{41}$+ 2 $P^2_{34}P_{12}P_{25}P_{51}$ + 2 $P^2_{35}P_{12}P_{24}P_{41}$ + 2 $P^2_{45}P_{12}P_{23}P_{31}$ + 2 $P_{12}P_{23}P_{35}P_{54}P_{41}$+ 2 $P_{12}P_{23}P_{34}P_{45}P_{51}$ + 2 $P_{12}P_{24}P_{45}P_{53}P_{31}$) + (2 $P_{12}P_{24}P_{43}P_{35}P_{51}$+ 2 $P_{12}P_{25}P_{53}P_{34}P_{41}$ + 2 $P_{12}P_{25}P_{54}P_{43}P_{31}$ + 2 $P_{13}P_{32}P_{24}P_{45}P_{51}$+ 2 $P_{13}P_{34}P_{42}P_{25}P_{51}$ + 2 $P_{13}P_{35}P_{52}P_{24}P_{41}$ + 2 $P_{13}P_{32}P_{25}P_{54}P_{41}$+ 2 $P_{14}P_{42}P_{23}P_{35}P_{51}$ +2 $P_{15}P_{52}P_{23}P_{34}P_{41}$)

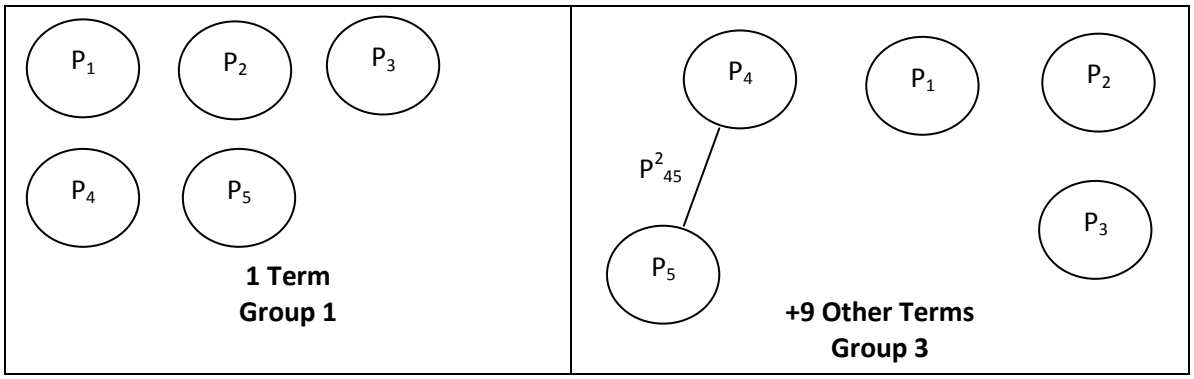
(5)

This is generic representation of test generation system having five subsystems, that include 120 terms. The terms are reduced to 28 if the permanent function for equation (5) is used. The values of P13 and P32 are 0 that implies the connectivity is zero.The variable permanent function for test generation product system is given by:

Per (E) = $P_1P_2P_3P_4P_5$ + $P_1P_2P_3P^2_{45}$ +$P_1P_2P_5P^2_{34}$ + $P_1P_3P_5P^2_{24}$ + $P_1P_4P_5P^2_{23}$ + $P_2P_3P_5P^2_{14}$ + $P_2P_4P_5P^2_{13}$+ $P_3P_4P_5P^2_{12}$ + 2 $P_1P_5P_{23}P_{34}P_{42}$ + 2 $P_3P_5P_{12}P_{24}P_{41}$ + 2 $P_2P_5P_{13}P_{34}P_{41}$ + 2 $P_4P_5P_{12}P_{23}P_{31}$ + $P_1P^2_{23}P^2_{45}$ + $P_2P^2_{45}P^2_{13}$ + $P_3P^2_{12}P^2_{45}$ + $P_5P^2_{12}P^2_{34}$ + $P_5P^2_{13}P^2_{24}$ + $P_5P^2_{14}P^2_{23}$ + 2 $P_5P_{13}P_{32}P_{24}P_{41}$ + 2 $P_5P_{13}P_{34}P_{42}P_{21}$ + 2 $P^2_{45}P_{12}P_{23}P_{31}$

(6)

This equation independently shows the test generation product system given in Fig. 7. Each term has a unique significance every one of it correspond to a unique subset of the test generation product system. The equations can be simply written just by visually observing the test generation system in Fig. 8.We begin by writing the permanent function (Equation (6)) in the standard form. These are divided into (N+1) groups. The combinations and connectivity among subsystems and interactions is given in Fig. 10.



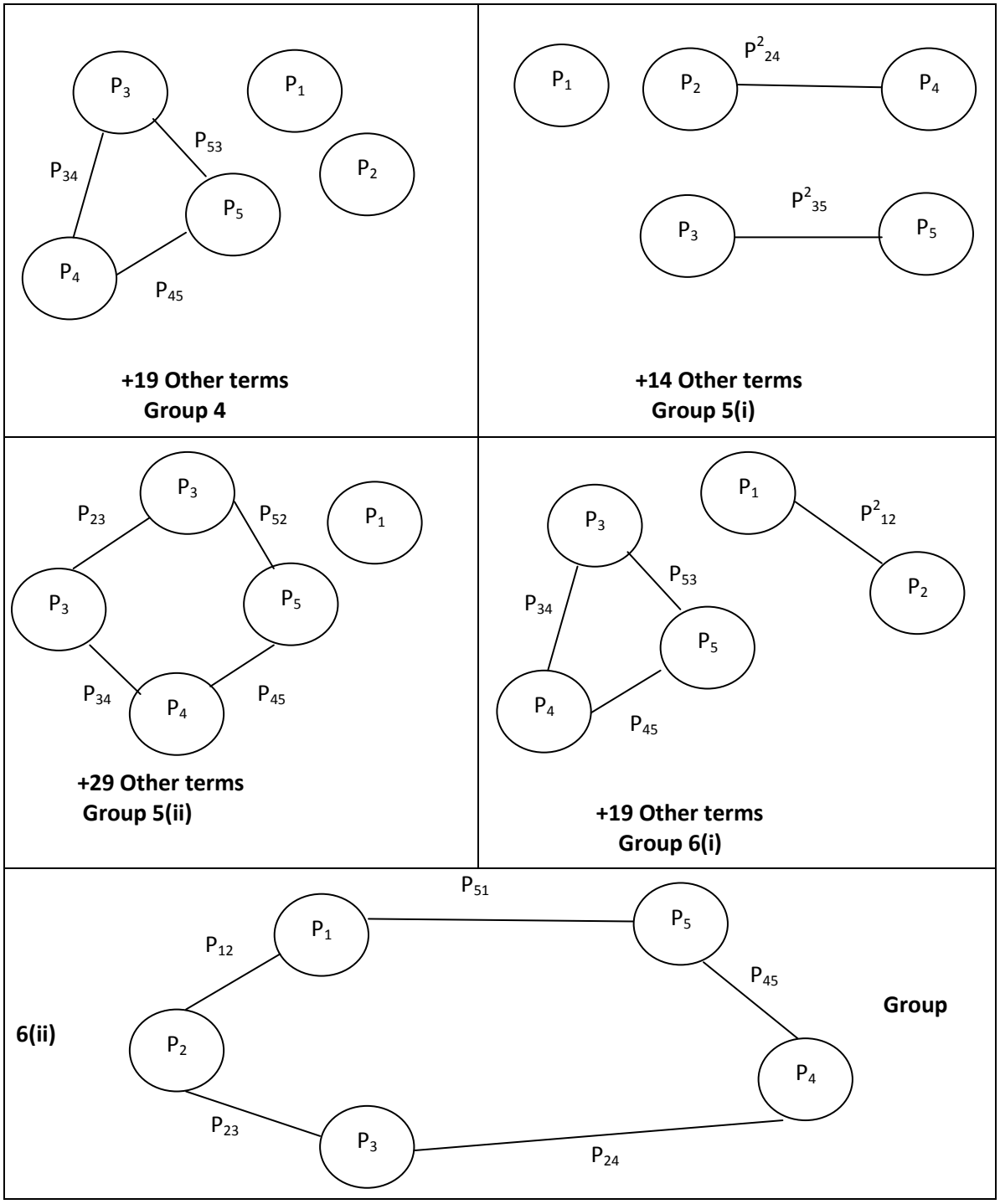


Fig. 12 The graphical representation of the associated terms

The division in (N+1) groups , gives us an exhaustive technique for inspection of a test generation system product. The identification of constituents, the parameters,

design structure and the connectivity between the subsystems of test generation systems becomes easier.

Group 1: have one term associated with it of the five isolated vertices i.e., all five subsystems are taken as independent quantities.

Group 2: this group is absent because a specific subsystem do not have self connectivity .

Group 3: it consist of six terms, each one of which is a set of one dyad (dyad is the combination two subsystems i and j,

which is taken as one unit), P2 ij or a two-subsystem loop i.e.

Pij Pij and three independent subsystems.

Group 4: it have total of 8 terms each one of which having a set of a 3-subsystem loop and the independent subsystems. A 3-subsystem loop is a system taken as one unit.

Group 5: it have only 2 subgroups: 5(i) having 3 terms, each one of which is the subset of 2 independent dyads or 2-subsystem loops and an independent subsystem. 5(ii) contains 4 terms, the terms are constructed as the set of four-subsystem loop and an independent subsystem.

Group 6: this group have 2 subgroups: 6(i) is not present because there is no loop having three subsystems and one dyad or a 2-subsystem loop. 6(ii) is not present because no loop have 5-subsystems.

Basically 5-subsystem permanent function contains 5!, i.e. 120 terms contained in (N+1) groups. Through graphical representation in Fig. 10 it is possible to analyse and improve the test generation product system. It fives opportunity to make SWOT analysis (strength-weakness-opportunities-threats) on the test generation product system and also enables to make strategic decisions in their favor.

4.8 Evaluating Pi

The values corresponding to the diagonal elements of the five subsystems of the test generation system Pi's i=1,2,…..,5 are

$P_1 = per(EP_1)$; $P_2 = per(EP_2)$; $P_3 = per(EP_3)$;

$P_4 = per(EP_4)$; $P_5 = per(EP_5)$

the EPi's i=1,2,….,5 are the variable permanent matrices corresponding to the 5 subsystems of the test generation product system. The Pi's are calculated in the same way as per(E) of Equation (6a). The following procedure is carried out for the calculation taking into account, the subsystems of the test generation system:

1. The different subsubsystems are taken into account to represent the structure of these subsystems.

2. The amount of interactions, connectivity, etc. among the subsubsytems is obtained.

Firstly the matrix equations are extracted by making Digraphs (Fig. 10) of each of the five subsystems i.e. EPis and the corresponding permanent functions are also obtained, Pis i¼1, . . . , 5. The interactions among the systems Pi and Pj are shown by non-diagonal elements Pij(i, j=1, 2, . . . , 5). Pij can take the form of a multinomial,or a graph, or a matrix or some other analytical model depending on selection of structure analysis used.

Chapter 5 Test Generation System Analysis

.

The procedure for study described earlier for completely analyzing the test generation system is given below:

Step 1: Take a test generation product, go through its subsystems (Parameters in WHERE clause, Generate Values using user huerisitics) and the connectivity.

Step 2: Now make the block diagram of the test generation system, taking into account its subsystems and their connectivity.

Step 3: Using subsystems for nodes and the edges connecting them as interaction make a graph of the test generation system (Fig. 8).

Step 4: Now make a matrix and multinomial representations of test generation system

Step 5: Find the values corresponding to the diagonal elements from the permanent functions respective to each subsystems of the test generation and repeat procedure.

Step 6: Also find the values corresponding to non-diagonal entities at different levels of hierarchy of the test generation , that is its systems, their subsystems, their subsubsystems,their subsubsubsystems etc.

The values of the connectivities Pij's among the subsystems P_1,P_2, . . . , P_N can be represented by the multinomial or a matrix, depending on what type of connectivity is there among the subsystems. Now a subsubsystem can also be taken as system. So the subsystems can be decomposed into subsubsystems and the graphs, matrix, and permanent representations could be made. Depending on how deep the analysis is to be done, the process could be taken deeper and deeper. In some cases, Pij's could be found experimentally or by making use of some mathematical models. From the data thus obtained, complete multinomial corresponding to the test generation system can be calculated..

Chapter 6 Conclusions and Future Scope

This thesis describes an approach to develop a test generation product by taking into account all the features of design, production, and the process parameters, also the connectivity among the constituents is given using a digraph and matrix approach. This provides a powerful feature to alter the amount of interaction between the test generation entities. The effect of the interactions on the process parameters, design, and production attributes is also given. The work in this thesis is based on five characteristics, that describe the whole test generation product system. The approach in this thesis is comprised of the test generation system digraph, test generation system matrix, and the test generation system permanent function. The test generation digraph shows the structural characteristics, their connectivity, dependencies used in visual modeling and analysis mathematically. The test generation system matrix converts the digraph into some other mathematical form. The storage and retrieval of test generations and the computer processing becomes easy with the help of matrix representation. Another term test generation system permanent function is also given which is a mathematical model showing the test generation product and also used to find the test generation system index. The various terms associated with the permanent function gives a path for analysis to check the strength, weakness and also for the improvement of test generations. When Pis and Pjs are represented by numeric values in the variable adjacency matrix the characterization of test generation is converted into a single numerical index. Therefore the numerical index of a test generation system thus obtained from a multinomial is used for comparison and desired selection. The work done till now makes use of numerical approach for test generation product system that can be further used for the optimization of structure and production parameters. A generalized methodology is also given for modeling a system with N subsystems and their connectivity. The above approach also suggest a method on the basis of which two test generation product systems can be compared making use of the permanent function on design basis. An attempt could be made in future publications to correlate test generation performance characteristics with the test generation structural model.

Future scope

This approach gives a powerful tool to the developer with its various features. This approach can also be used with morphological chart/tree, to develop more designs solutions and convenient one could be choosed. This approach can also be used in other areas of quality improvement, cost scheduling and time to market. It is also possible to exploit the methodology to extend the useful product life in the market by making strategic changes in the product. This approach gives the insight view to the user about test generation products and also helps to choose desired product at correct time at right appropriate cost from market. Using the features of the given approach Cost scheduling , Run-time scheduling for various algorithms can be done. By using this approach, research can be done in the areas of global projects of quantitative structure activity relationship (QSAR) and quantitative structure properties relationship (QSPR) [21, 22]. The given methodology moves towards the generation of a technique for virtual fabrication [23] of test generation system and virtual designing of the whole test generation product system taking account of product design, generate values using user's heuristics, Parameter in WHERE/SET clauses, and the other process parameters.

References

[1]C.Binning, D.Kossmann and E.Lo, "Reverse query processing", *International Conference on Data Engineering (ICDE-2007),* pp. 506-515, 2007.

[2]C.Binning, D.Kosmann, E. Lo, and M.T.A Ozsu.Qagen, "Generating query-aware test databases", I*nternational conference on Management of data(SIGMOD-2007),* pp. 341-352, 2007.

[3]D. Chays, " Test Data Generation For Relational Database Applications", Phd thesis: Computer and Information Science, Polytechnic University, 2006.

[4]D. Chays, Y. Deng, P. G. Frankl, S. Dan, F. I. Vokolos, and E. J. Weyuker, "An agenda for testing relational database applications", *Software testing verification and reliability*, vol.14(1), pp. 17-44,2004.

[5]M. B. Cohen, P. B. Gibbons, W. B. Mugridge, and C. J. Colbourn, " Constructing test suites for interaction testing", *25th International Conference on Software Engineering(ICSE-2003)*, pp. 38-48, 2003.

[6]Y. Deng, P. Frankl, and D. Chays, "Testing database transactions with agenda", *27th international conference on Software engineering (ICSE-2005)*, pp. 78-87, 2005.

[7]M. Emmi, R. Majumdar, and K. Sen, "Dynamic test input generation for database applications", I*nternational symposium conference on Software testing and analysis (ISSTA-2007)*, pp. 151-162, 2007.

[8]T. J. Ostrand and M. J. Balcer, "The category- partition method for specifying and generating fuctional tests," *Commun. ACM*, vol.31(6), pp. 676-686, 1988.

[9]D. Willmor and S. M. Embury," An intensional approach to the specification of test cases for database applications", *28th international conference on Software engineering(ICSE-2006),* pp. 102-111, 2006.

[10]R.Venkataswamy, and V.P.Agrawal, "System and Structural Analysis of an Automobile Vehicle – A Graph Theoretic Approach", *International Journal of Vehicle Design*, vol. 16(4/5), pp. 477–503, 1995.

[11]M. Mohan, O.P. Gandhi, and V.P. Agrawal, "Systems Modeling of a Coal-based Steam Power Plant", *Journal of Power and Energy*, vol. 217(3), pp. 259–277, 2003.

[12]O.P. Gandhi, and V.P. Agrawal, "Failure Cause Analysis – A Structural Approach",*Journal of Pressure Vessel Technology Trans ASME*, vol.118(4), pp. 434–440, 1996.

[13]R. Venkataswamy, and V.P. Agrawal, "A Digraph Approach to Quality Evaluation of an Automatic Vehicle", *Quality Engineering*, vol. 9(3), pp.405–417, 1997.

[14]O.P. Gandhi, and V.P. Agrawal, "FMEA – A Digraph and Matrix Approach, Reliability Engineering and System Safety", *Reliability Engineering & System Safety,* vol. 35(2), pp. 147–158, 1992.

[15]G. Sandeep, V.P. Agrawal and I.A. Khan, "A Digraph Approach to TQM Evaluation of an Industry", *International journal of Production Research*, vol. 42(19), pp. 4031–4053, 2004.

[16]N. Deo, "Graph Theory with Application to Engineering and Computer Science", *Prentice-Hall, New Delhi, India*, 2000.

[17]D.F. Robinson, and L.R. Foulds, "Digraphs: Theory and Techniques", *Gordon and Breach Science Publishers, London*, 2006.

[18]W.B. Jurkat, and H.J. Ryser, "Matrix Factorization of Determinants and Permanents", *Journal of Algebra*, vol. 3(1), pp. 1–27, 1966.

[19]Thomas J. Ostrand and Mare J. Balcer, "The category patitioning method for specifying and generating functional tests", *Communication of ACM*, vol.31 (6), Pp .676-686, 1998.

[20]D. Chays, "Test Data Generation for Relational Database Applications" PhD thesis, Computer and Information Science, Polytechnic University, 2003.

[21]M. B. Cohen, P. B. Gibbons, W. B. Mugridge, and C. J. Colbourn, "Constructing Test suites for interaction testing", *25th International Conference on Software Engineering*, pp.38–48 , USA, 2003.

[22] (2014, Jan) http://math.tut.fi/~ruohonen/GT_English.pdf.

[23](2014, Jan) http://en.wikipedia.org/wiki/Matrix_(mathematics).

[24](2014, Feb) http://en.wikipedia.org/wiki/Determinant.

[25] (2014, March) http://planetmath.org/permanent.

[26] (2014, March) http://planetmath.org/immanent.

[27]David Chays, "Query-based Test Generation for Database Applications", *In ACM SIGSOFT international symposium on Software testing and analysis,* 2000.

[28]Hicham G. Elmongui and Vivek Narasayya, “A Framework for Testing Query Transformation Rules”, *ACM SIGMOD International Conference on Management of data Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, 2010.

[29]D. Chays, “Generating Test Databases for the functional testing of OLTP applications”, *1st international workshop on Testing database systems*, 2010.

[30]Kai Pan, Xintao Wu, Tao Xie, “Database state generation via dynamic symbolic execution for coverage criteria”, *Fourth International Workshop on Testing Database Systems*, 2011.

[31]Micheal Marcozzi, Wim Vanhoof, Jean-Luc Hainaut, “Test input generation for Database programs using relational constraints”, *Fifth International Workshop on Testing Database Systems*, 2012.

[32]Kai Pan, Xintao Wu, Tao Xie, “Generating program input for database application testing”, *Fifth International conference on Testing Database Systems*, 2012.

[33]Kai Pan, Xintao Wu, Tao Xie, “Guided test generation for database applications via synthesized database interactions”, *ACM Transactions on Software Engineering and Methodology (TOSEM),* vol. 23 , 2014.

[34]Fei Li, Hosagrahar V Jagadish, “NaLIR: an interactive natural language interface for querying relational databases”, *ACM SIGMOD international conference on Management of data,* 2014.

[35]Jose M. Faleiro, Alexander Thomson, Daniel J. Abadi, “Lazy evaluation of transactions in database systems”, *ACM SIGMOD international conference on Management of data,* 2014.

List of Publications

[1] Bhagesh Kumar, Ashish Aggarwal ,“ Implementation of Graph Theoretic Approach for Structural Modeling and Analysis of Query Based Test Generation in Database Applications,” IEEE International Conference on Advances in Engineering and Technology Research (ICAETR-2014), IEEE, 2014 [Communicated].