ACADEMIC YEAR 2021-22

6.5.2. The Institution reviews its teaching learning process, structures and methodologies of operations and learning outcomes at periodic intervals through IQAC setup as per norms and recorded the internal improvement in various activities for the first cycle and incremental improvements made for the preceding one year with regard to quality.

S. No.	Example of Institutional Review	Subject	Page No.(s)
1	Course File and Evaluation	Software Testing Methodologies	2-123

PRINCIPAL

Sri Indu Institute of Engineering and Technology

Department of Computer Science and Engineering

COURSE FILE

PREPARED BY	:E.RUPA
DEPARTMENT	: COMPUTER SCIENCE AND ENGINEERING
ACADEMIC YEAR	: 2021-2022
SUBJECT	:CS615PE-SOFTWARE TESTING METHODOLOGIES(C324)
CLASS	: III YEAR /II SEM – CSE – C – SECTION

PRINCIPAL
Sri Indu Institute of Engineering & Technology
Sheriguda(V), Ibrahimpatnam
R.R. Dist. Telangana -501 510

SRI INDU INSTITUTE OF ENGINEERING & TECHNOLOGY
Sheriguda (V), Ibrahimpatnam (M), Hyderabad,
R.R. Dist., Telangana State – 501 510.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT VISION

To become prominent knowledge hub for learners, strive for educational excellence with innovative and industrial techniques so as to meet the software industry needs.

DEPARTMENT MISSION

DM1 To Provide educational ambience that enhances innovations, problem solving skills, leadership qualities, decision making, team-spirit and ethical responsibilities.

DM2 Imparting quality education with professional and personal ethics, so as to attain with challenging technological needs of industry and society.

DM3 To provide academic infrastructure and develop linkage with the world class organizations to strengthen industry-academia relationships for learners.

DM4 Provide platform to strengthen novel research in thrust area of Computer Science and Engineering to serve the needs of Government and Society.

Head of the Department
Computer Science & Engg. Dept.
SRI INDU INSTITUTE OF ENGG & TECH.
Sheriguda(V), Ibrahimpatnam(M), R.R.Dist-501 510.

PRINCIPAL
Sri Indu Institute of Engineering & Technology
Sheriguda(V), Ibrahimpatnam
R.R. Dist. Telangana-501 510

SRI INDU INSTITUTE OF ENGINEERING & TECHNOLOGY
Sheriguda (V), Ibrahimpatnam (M), Hyderabad,
R.R. Dist., Telangana State – 501 510.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PROGRAM EDUCATIONAL OBJECTIVES(PEO's)

PEO1 Graduates with strong academic and technical skills of modern computer science and engineering.

PEO2 Graduate with leadership qualities and ability to solve real time problems using current techniques & tools in interdisciplinary environment.

PEO3 Graduates with attitude towards lifelong learning through continuing education and professional development.

PROGRAM SPECIFIC OUTCOMES(PSO's)

PSO1 **Professional Skills:** The ability to implement computer programs of varying complexity in the areas related to web design, cloud computing and networking.

PSO2 **Problem-Solving Skills:** The ability to develop quality products using open ended programming environment

Head of the Department
Computer Science & Engg. Dept.
SRI INDU INSTITUTE OF ENGG & TECH.
Sheriguda(V), Ibrahimpatnam(M), R.R.Dist-501 510.

Principal
Sri Indu Institute of Engineering &
Sheriguda(V), Ibrahimpatnam(M)
R.R. Dist. Telangana -501 510

PROGRAMME OUTCOMES (POs):

PO1 **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2 **Problem analysis:** Identify, formulate, review research literature, and analyse complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3 **Design/development of solutions:**Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4 **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5 **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.

PO6 **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7 **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8 **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9 **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10 **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11 **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12 **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PRINCIPAL
Sri Indu Institute of Engineering & Tech
Sheriguda(V), Ibrahimpatnam(M)
R.R. Dist. Telangana -501 510

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD

CS615PE: (Professional Elective - III)
SOFTWARE TESTING METHODOLOGIESIII Year B.Tech. CSE II-Sem

L T P C
3 0 0 3

Prerequisites

1. A course on "Software Engineering"

Course Objectives

1. To provide knowledge of the concepts in software testing such as testing process, criteria,strategies, and methodologies.
2. To develop skills in software test automation and management using latest tools.

Course Outcomes:

Design and develop the best test strategies in accordance to the development model.

UNIT - I

Introduction: Purpose of testing, Dichotomies, model for testing, consequences of bugs, taxonomy of bugs Flow graphs and Path testing: Basics concepts of path testing, predicates, path predicates and achievable paths, path sensitizing, path instrumentation, application of path testing.

UNIT - II

Transaction Flow Testing: transaction flows, transaction flow testing techniques.
Dataflow testing: Basics of dataflow testing, strategies in dataflow testing, application of dataflow testing.
Domain Testing:domains and paths, Nice & ugly domains, domain testing, domains and interfaces testing, domain and interface testing, domains and testability.

UNIT - III

Paths, Path products and Regular expressions: path products & path expression, reduction procedure, applications, regular expressions & flow anomaly detection.
Logic Based Testing: overview, decision tables, path expressions, kv charts, specifications.

UNIT - IV

State, State Graphs and Transition testing: state graphs, good & bad state graphs, state testing, Testability tips.

UNIT - V

Graph Matrices and Application: Motivational overview, matrix of graph, relations, power of a matrix, node reduction algorithm, building tools. (Student should be given an exposure to a tool like JMeter or Win-runner).

Text Books:

1. Software Testing techniques - Baris Beizer, Dreamtech, second edition.
2. Software Testing Tools – Dr. K. V. K. K. Prasad, Dreamtech.

References:

1. The craft of software testing - Brian Marick, Pearson Education.
2. Software Testing Techniques – SPD(Oreille)
3. Software Testing in the Real World – Edward Kit, Pearson.
4. Effective methods of Software Testing, Perry, John Wiley.
5. Art of Software Testing – Meyers, John Wiley

PRINCIPAL
Sri Indu Institute of Engineering & T
Sheriguda(V), Ibrahimpatnam,
R R Dist. Telangana -501 510

SRI INDU INSTITUTE OF ENGINEERING AND TECHNOLOGY
Accredited by NAAC A+ Grade, Recognized under 2(f) of UGC Act 1956.
(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)
Khalsa Ibrahimpatnam, Sheriguda(V), Ibrahimpatnam(M), Ranga Reddy Dist.,Telangana
https://siiet.ac.in/

Department of Computer Science and Engineering
2021-22; 2nd Semester

Course Outcomes

Course: Software Testing Methodologies (C324) **Class: III – II SEM – C - Section**

After completing this course the student will be able to:

C324.1 Recognize the importance , purpose of testing and its applications in software development life cycle. **(Knowledge)**

C324.2 List transaction flows ,transaction flow techniques and implementation comments in software testing **(Analysis)**

C324.3 Compare domain testing and path testing and explains various domain techniques. **(Analysis)**

C324.4 Design reduction procedure and its applications , lists regular expressions and data flow anomaly detection. **(Synthesis)**

C324.5 Design and implement state graph, state testing, good state graph, bad state graph and their testability tips. **(Synthesis)**

C324.6 Describe graph Matrices, matrix properties and node reduction algorithm. **(Knowledge)**

Mapping of course outcomes with program outcomes:

High -3 Medium -2 Low-1

PO/PSO/CO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2
C324.1	3	1	2	2	2	-	-	-	1	-	-	-	1	-
C324.2	1	2	2	1	2	-	-	-	1	-	-	-	2	-
C324.3	1	2	1	2	-	-	-	-	-	-	-	-	1	1
C324.4	2	1	1	1	-	-	-	-	-	-	-	-	-	1
C324.5	1	2	1	1	-	-	-	-	1	-	-	-	-	-
C324.6	3	2	1	2	-	-	-	-	2	-	-	-	-	2
C324	1.83	1.67	1.33	1.5	2	-	-	-	1.25	-	-	-	1.3	1.3

Faculty Signature

PRINCIPAL
Sri Indu Institute of Engineering &
Sheriguda(V), Ibrahimpatnam(
R.R. Dist. Telangana -501 510

SRI INDU INSTITUTE OF ENGINEERING AND TECHNOLOGY
Accredited by NAAC A+ Grade, Recognized under 2(f) of UGC Act 1956.
(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)
Khalsa Ibrahimpatnam, Sheriguda(V), Ibrahimpatnam(M), Ranga Reddy Dist.,Telangana
https://siiet.ac.in/

Department of Computer Science and Engineering
2021-22; 2nd Semester

CO – PO / PSO Mapping Justification

Course: Software Testing Methodologies (C324) **Class: III – II SEM – C - Section**

PROGRAMME OUTCOMES (POs):

PO1 **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2 **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3 **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4 **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5 **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO7 **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO9 **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO12 **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES(PSOs):

PSO1 **Professional Skills:** The ability to implement computer programs of varying complexity in the areas related to web design, cloud computing and networking.

PSO2 **Problem-Solving Skills:** The ability to develop quality products using open ended programming environment.

PRINCIPAL
Sri Indu Institute of Engineering &
Sheriguda(V), Ibrahimpatnam
R.R. Dist. Telangana-501 510

Department of Computer Science and Engineering
2021-22; 2nd Semester
CO-PO Mapping Justification

C324.1 Recognizethe importance , purpose of testing and its applications in software development life cycle. **(Knowledge)**

	Justification
PO1	Apply the knowledge of testing .
PO2	To identify the types of testing and its applications.
PO3	Design solutions for the software development life cycle.
PO4	Use research-based knowledge purpose of testing and its applications.
PO5	Appropriate techniques are used to classify the bugs into different categories
PO9	Effectively as an individual work for the testing and understand the process of testing.
PSO1	To implement computer program consequences and understand the importance of bugs.

C324.2 List transaction flows ,transaction flow techniques and implementation comments in software testing **(Analysis)**

	Justification
PO1	Apply the knowledge of transaction techniques.
PO2	To identify the types of transaction flow techniques.
PO3	Design solutions to identify the complications in a transaction flow testing method and anomalies in data flow testing
PO4	Use research-based knowledge of comments in software testing.
PO5	Apply appropriate techniques for data flow anomaly state graphs and control flow graphs and represent the state of the data objects.
PO9	Function effectively to analyse various strategies of data flow testing.
PSO1	To implement software testing for transaction flow testing and data flow testing.

C324.3 Compare domain testing and path testing and explains various domain techniques. **(Analysis)**

	Justification
PO1	Apply the knowledge for concept of domain testing
PO2	Identify the ugly and nice domains.
PO3	Design the domain testing strategy for different dimension domains.
PO4	Use research methods various domain techniques.
PSO1	Ability to implement the mathematical laws (distributive, associative, commutative etc)
PSO2	To develop the control flow graph and identify the path products, path sums and path expressions.

PRINCIPAL
Sri Indu Institute of Engineering &
Sheriguda(V), Ibrahimpatnam
R.R. Dist. Telangana 501 510

C324.4 Design reduction procedure and its applications , lists regular expressions and data flow anomaly detection. **(Synthesis)**

	Justification
PO1	Apply reduction procedure algorithm to a control flow graph and simplify it into a single path expression
PO2	Identify the probability of paths and understand the need for finding the probabilities.
PO3	Design complimentary operations such as PUSH / POP or GET / RETURN are interpreted in a flow graph.
PO4	Use research methods data flow anomaly detection.
PSO2	The ability to calculate mean processing time of a routine of a given flow graph.

C324.5Design and implement state graph, state testing, good state graph, bad state graph and their testability tips. **(Synthesis)**

	Justification
PO1	Apply the knowledge for graphical representation of state graphs.
PO2	Analyse the Software implementation of sate graphs
PO3	Design the state graph for good state graph and state graph.
PO4	Use research methods implement state graph.

C324.6 Describe graph Matrices, matrix properties and node reduction algorithm. **(Knowledge)**

	Justification
PO1	Apply the Knowledge of graph matrices, understanding testing theory
PO2	Implementation of node-reduction algorithms.
PO3	Design the graph matrices for node reduction algorithm.
PO4	Use research methods implement graph matrices and node reduction algorithm.

Faculty Signature

PRINCIPAL
Sri Indu Institute of Engineering & Tech
Sheriguda(V), Ibrahimpatnam(M),
R.R. Dist. Telangana-501 510

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD

Academic Calendar 2021-22

B. TECH./B.PHARM. III & IV YEARS I & II SEMESTERS

I SEM

S. No	Description	Duration	
		From	To
1	Commencement of I Semester classwork	06.09.2021	
2	1st Spell of instructions (including Dussehra Recess)	06.09.2021	06.11.2021 (9 Weeks)
3	Dussehra Recess	11.10.2021	16.10.2021 (1 Week)
4	First Mid Term Examinations	08.11.2021	13.11.2021 (1 Week)
5	Submission of First Mid Term Exam Marks to the University on or before	20.11.2021	
6	2nd Spell of Instructions	15.11.2021	08.01.2022 (8 Weeks)
7	Second Mid Term Examinations	10.01.2022	15.01.2022 (1 Week)
8	Preparation Holidays and Practical Examinations	19.01.2022	25.01.2022 (1 Week)
9	Submission of Second Mid Term Exam Marks to the University on or before	25.01.2022	
10	End Semester Examinations	27.01.2022	09.02.2022

II SEM

S. No	Description	Duration	
		From	To
1	Commencement of II Semester classwork	16.03.2022	
2	1st Spell of Instructions	10.03.2022	09.04.2022 (5 Weeks)
3	First Mid Term Examinations	07.04.2022	12.04.2022 (1 Week)
4	Submission of First Mid Term Exam Marks to the University on or before	20.04.2022	
5	2nd Spell of Instructions (including Summer Vacation)	16.04.2022	24.06.2022 (10 Weeks)
6	Summer Vacation	09.05.2022	21.05.2022 (2 Weeks)
7	Second Mid Term Examinations	25.06.2022	01.07.2022 (1 Week)
8	Preparation Holidays and Practical Examinations	02.07.2022	09.07.2022 (1 Week)
9	Submission of Second Mid Term Exam Marks to the University on or before	09.07.2022	
10	End Semester Examinations	11.07.2022	23.07.2022 (2 Weeks)

PRINCIPAL
Sri Indu Institute of Engineering & T
Sheriguda(V), Ibrahimpatnam
R.R. Dist. Telangana 501 5

Department of Computer Science and Engineering

2021-22; 2nd Semester

Course Title	SOFTWARE TESTING METHODOLOGIES
Course Code	CS615PE
Programme	B.Tech
Year & Semester	III-year II-semester
Regulation	R18
Course Faculty	Mrs.E.RUPA, Assistant Professor , CSE

LESSON PLAN

S.NO	Unit	TOPIC	Number of Sessions Planned	Teaching method/Aids	REFERENCE
1	I	Introduction	1	Black Board	T1
2		Purpose of testing	1	Black Board	T1
3		Dichotomies	1	Black Board	T1
4		model for testing	1	Black Board	T1
5		**Tutorial1(** Dichotomies)	1	Black Board	T1
6		consequences of bugs	1	Black Board	T1
7		taxonomy of bugs Flow graphs	1	Black Board	T1
8		Path testing Introduction	1	Black Board	T1
9		Basics concepts of path testing	1	Black Board	T1
10		**Tutorial2(** Basics concepts of path testing)	1	Black Board	T1
11		Predicates Examples	1	Black Board	T1
12		path predicates and achievable paths	1	Black Board	T1
13		path sensitizing	1	Black Board	T1
14		path instrumentation, application of path testing. **Tutorial3(** path sensitizing)	1	Black Board	T1
15		Transaction Flow Testing Introduction	1	Black Board	T1

PRINCIPAL
Sri Indu Institute of Engg. & Tech
[redacted]

16		transaction flows	1	Black Board	T1
17		transaction flow testing techniques	1	Black Board	T1
18		Dataflow testing	1	Black Board	T1
19		**Tutorial4**(transaction flow testing techniques)	1	Black Board	T1
20		Basics of dataflow testing	1	Black Board	T1
21	II	strategies in dataflow testing	1	PPT	T1
22		application of dataflow testing	1	PPT	T1
23		Domain Testing Introduction	1	PPT	T1
24		domains and paths	1	PPT	T1
25		**Tutorial5(** application of dataflow testing)	1	Black Board	T1
26		Nice & ugly domains	1	PPT	T1
27		domain testing	1	PPT	T1
28		domains and interfaces testing	1	PPT	T1
29		domain and interface testing	1	PPT	T1
30		**Tutorial6**(domainsand testability)	1	Black Board	T1
31		Paths, Path products and Regular expressions	1	Black Board	T1
32	III	path products & path expression	1	Black Board	T1
33		reductionprocedure, applications	1	Black Board	T1
34		regular expressions & flow anomaly detection	1	Black Board	T1
35		**Tutorial7**(regular expressions & flow anomaly detection)	1	Black Board	T1
36		Logic Based Testing Introduction	1	Black Board	T1
37		decision tables, path expressions	1	Black Board	T1
38		kv charts, specifications	1	Black Board	T1
39		State, State Graphs and Transition testing Introduction	1	Black Board	T1
40		**Tutorial7(** Transition testing)	1	Black Board	T1

PRINCIPAL
Sri Indu Institute of Engineering & Technology
Sheriguda(V), Ibrahimpatnam
R.R. Dist.

41	IV	good & bad state graphs	1	Black Board	T1
42		state testing	1	Black Board	T1
43		Testability tips	1	Black Board	T1
44		good & bad state graphs Examples	1	Black Board	T1
45		**Tutorial8**(State Testing)	1	Black Board	T1
46	V	Motivational overview	1	Black Board	T1
47		matrix of graph	1	Black Board	T1
48		relations, power of a matrix	1	Black Board	T1
49		node reduction algorithm	1	Black Board	T1
50		**Tutorial9**(node reduction algorithm)	1	Black Board	T1
		building tools	1	Black Board	T1

Text Books:

1. Software Testing techniques - Baris Beizer, Dreamtech, second edition.
2. Software Testing Tools – Dr. K. V. K. K. Prasad, Dreamtech.

References:

1. The craft of software testing - Brian Marick, Pearson Education.
2. Software Testing Techniques – SPD(Oreille)
3. Software Testing in the Real World – Edward Kit, Pearson.
4. Effective methods of Software Testing, Perry, John Wiley.
5. Art of Software Testing – Meyers, John Wiley

Faculty Signature

PRINCIPAL
Sri Indu Institute of Engineering & Tec
Sheriguda(V), Ibrahimpatnam(M),
R.R. Dist. Telangana -501 510

Code No: 156CW **R18**

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD
B. Tech III Year II Semester Examinations, February/March - 2022
SOFTWARE TESTING METHODOLOGIES
(Common to CSE, IT)

Time: 3 hours **Max. Marks: 75**

Answer any five questions
All questions carry equal marks

* * *

1.a) Discuss about implementation and application of path testing.
b) Describe instantiation and sensitization in transaction flow testing. [8+7]

2.a) Explain about approximate number of paths with suitable example.
b) Discuss in detail about state bugs. [8+7]

3.a) Describe software implementation of state graphs.
b) Explain about partition algorithm with example. [8+7]

4.a) Briefly explain about consequences of bugs.
b) Discuss about three and four variable KV charts with examples. [8+7]

5.a) What are dataflow anomalies? Explain about dataflow anomaly state graph.
b) Write and explain testability tips of state testing. [8+7]

6.a) Distinguish the following: i) Modularity vs efficiency ii) Function vs structure
b) Describe motivational overview of graph matrices. [8+7]

7.a) Explain basic concepts of path products and path expressions.
b) Discuss about node-reduction algorithm. [7+8]

8. Explain the following:
a) Testing one-dimensional domains
b) Coding bugs
c) Identity elements. [5+5+5]

—ooOoo—

Code No: 126AH

R15

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD
B. Tech III Year II Semester Examinations, April - 2018
SOFTWARE TESTING METHODOLOGIES
(Common to CSE, IT)

Time: 3 hours

Max. Marks: 75

Note: This question paper contains two parts A and B.
Part A is compulsory which carries 25 marks. Answer all questions in Part A. Part B consists of 5 Units. Answer any one full question from each unit. Each question carries 10 marks and may have a, b, c as sub questions.

PART - A

(25 Marks)

1.a) What is meant by testing? Why we need it [2]
b) Define a model for software testing. [3]
c) Explain various loops. Give example for each. [5]
d) Write the applications of data flow testing. [3]
e) In what a nice domain differs from and ugly domain. [3]
f) Define domain testing with example. [3]
g) Explain Regular Expressions. [2]
h) Explain sum of product form and product of sum form. [3]
i) Define good state and bad state graphs. [2]
j) How can the graph be represented in Matrix form? [3]

PART - B

(50 Marks)

2. State and explain various dichotomies in software testing. [10]

OR

3.a) What is meant by program's control flow? How is it useful for path testing?
b) Discuss various flow graph elements with their notations. [5+5]

4.a) What is meant by transaction flow testing. Discuss its significance.
b) Compare data flow and path flow testing strategies. [5+5]

OR

5.a) Explain data-flow testing with an example. Explain its generalizations and Limitations.
b) Explain the terms Dicing, Data-flow and Debugging. [5+5]

6.a) State and Explain various restrictions at domain testing processes.
b) With a neat diagram, explain the schematic representation of domain testing. [5+5]

OR

7. Discuss the domains and interface testing in detail. [10]

Sri Indu Institute of Engineering & Technology

Sheriguda (V), Ibrahimpatnam (M), R.R.Dist-501 510

I - Mid Examinations, MAY -2022

Set – I

Year & Branch: III-CSE(A,B,C) Date: 5-5 -22(AN)

Subject: STM Marks: 10 Time: 60 min

Answer any **TWO** Questions. All Question Carry Equal Marks 2*5=10 marks

(This question paper is prepared with Course Outcome and BT's mapping)

1. Explain about taxonomy for bugs. (Comprehension)(C324.1)(5M)
2. Discuss Path instrumentation with an example. (Knowledge)(C324.1)(5M)
3. Explain Nice and Ugly domains with an example. (Comprehension)(C324.2)(5M)
4. Discuss Path products and path expressions with an example. (Knowledge)(C324.3)(5M)

PRINCIPAL
Sri Indu Institute of Engineering & Tech
Sheriguda(V), Ibrahimpatnam(M)
R.R. Dist. Telangana -501 510

Sri Indu Institute of Engineering & Technology

Sheriguda (V), Ibrahimpatnam (M), R.R.Dist-501 510

I - Mid Examinations, MAY -2022

Set – I

Year & Branch: III-CSE(A,B,C) Date: 5-5 -22(AN)

Subject: STM Marks: 10 Time: 60 min

ANSWER KEY PAPER

1. Explain about taxonomy for bugs. (Comprehension)(C324.1)(5M)

Ans :REQUIREMENTS, FEATURES AND FUNCTIONALITY BUGS: Various categories in Requirements, Features and Functionlity bugs include:

1. **Requirements and Specifications Bugs:**
   - Requirements and specifications developed from them can be incomplete ambiguous, or self-contradictory. They can be misunderstood or impossible to understand.
2. **Feature Bugs:**
   - Specification problems usually create corresponding feature problems.
   - A feature can be wrong, missing, or superfluous (serving no useful purpose). A missing feature or case is easier to detect and correct. A wrong feature could have deep design implications.
3. **Feature Interaction Bugs:**
   - Providing correct, clear, implementable and testable feature specifications is not enough.
   - Features usually come in groups or related features. The features of each group and the interaction of features with in the group are usually well tested.

- **STRUCTURAL BUGS:** Various categories in Structural bugs include:
  1. **Control and Sequence Bugs:**
     - Control and sequence bugs include paths left out, unreachable code, improper nesting of loops, loop-back or loop termination criteria incorrect, missing process steps, duplicated processing, unnecessary processing, rampaging, GOTO's, ill-conceived (not properly planned) switches, sphagetti code, and worst of all, pachinko code.
  2. **Logic Bugs:**
     - Bugs in logic, especially those related to misundertanding how case statements and logic operators behave singly and combinations
     - Also includes evaluation of boolean expressions in deeply nested IF-THEN-ELSE constructs.
  3. **Processing Bugs:**
     - Processing bugs include arithmetic bugs, algebraic, mathematical function evaluation, algorithm selection and general processing.
     - Examples of Processing bugs include: Incorrect conversion from one data representation to other, ignoring overflow, improper use of grater-than-or-eual etc

PRINCIPAL
Sri Indu Institute of Engineering & Technology
Sheriguda(V), Ibrahimpatnam
R.R. Dist. Telangana -501 510

4. **Initialization Bugs:**
    - Initialization bugs are common. Initialization bugs can be improper and superfluous.
    - Superfluous bugs are generally less harmful but can affect performance.

DATA BUGS:

- Data bugs include all bugs that arise from the specification of data objects, their formats, the number of such objects, and their initial values.
- Data Bugs are atleast as common as bugs in code, but they are foten treated as if they didnot exist at all.
- **Dynamic Data Vs Static data:**
    - Dynamic data are transitory. Whatever their purpose their lifetime is relatively short, typically the processing time of one transaction. A storage object may be used to hold dynamic data of different types, with different formats, attributes and residues.

CODING BUGS:

- Coding errors of all kinds can create any of the other kind of bugs.
- Syntax errors are generally not important in the scheme of things if the source language translator has adequate syntax checking.

INTERFACE, INTEGRATION, AND SYSTEM BUGS:

Various categories of bugs in Interface, Integration, and System Bugs are:

- **External Interfaces:**
    - The external interfaces are the means used to communicate with the world.
    - These include devices, actuators, sensors, input terminals, printers, and communication lines.
- **Internal Interfaces:**
    - Internal interfaces are in principle not different from external interfaces but they are more controlled.
- **Hardware Architecture:**
    - Bugs related to hardware architecture originate mostly from misunderstanding how the hardware works.
    - Examples of hardware architecture bugs: address generation error, i/o device operation / instruction error, waiting too long for a response, incorrect interrupt handling etc.
- **Operating System Bugs:**
    - Program bugs related to the operating system are a combination of hardware architecture and interface bugs mostly caused by a misunderstanding of what it is the operating system does.
- **Software Architecture:**
    - Software architecture bugs are the kind that called - interactive.
- **Control and Sequence Bugs (Systems Level):**
    - These bugs include: Ignored timing, Assuming that events occur in a specified sequence, Working on data before all the data have arrived

from disc, Waiting for an impossible combination of prerequisites, Missing, wrong, redundant or superfluous process steps.

- **Resource Management Problems:**
  - Memory is subdivided into dynamically allocated resources such as buffer blocks, queue blocks, task control blocks, and overlay buffers.
  - External mass storage units such as discs, are subdivided into memory resource pools.
- **Integration Bugs:**
  - Integration bugs are bugs having to do with the integration of, and with the interfaces between, working and tested components.
  - These bugs results from inconsistencies or incompatibilities between components.
- **System Bugs:**
  - System bugs covering all kinds of bugs that cannot be ascribed to a component or to their simple interactions, but result from the totality of interactions between many components such as programs, data, hardware, and the operating systems.

2. Discuss Path instrumentation with an example. (Knowledge)(C324.1)(5M)

Ans: PATH INSTRUMENTATION:

- Path instrumentation is what we have to do to confirm that the outcome was achieved by the intended path.

Co-incidental Correctness: The coincidental correctness stands for achieving the desired outcome for wrong reason.

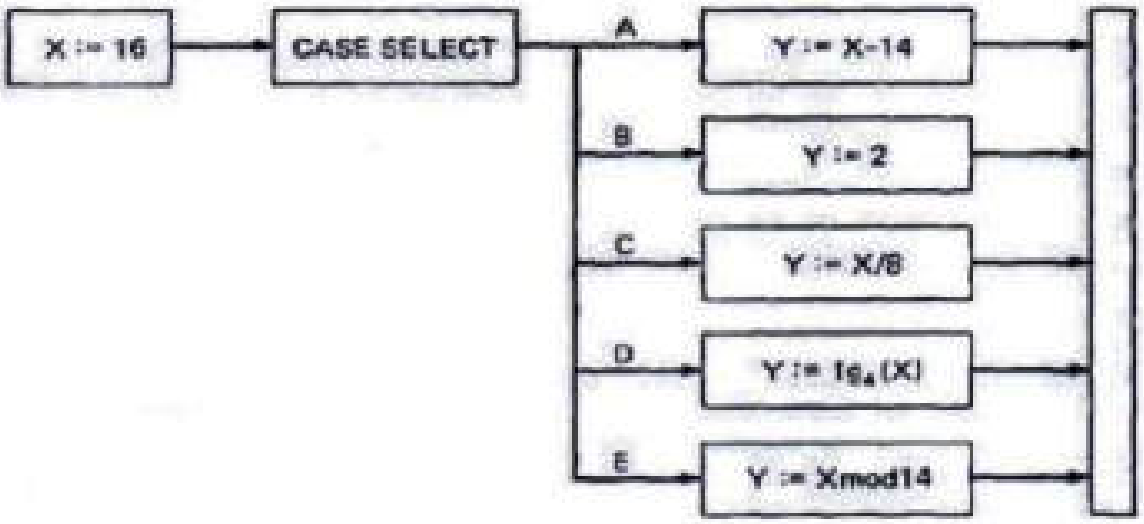
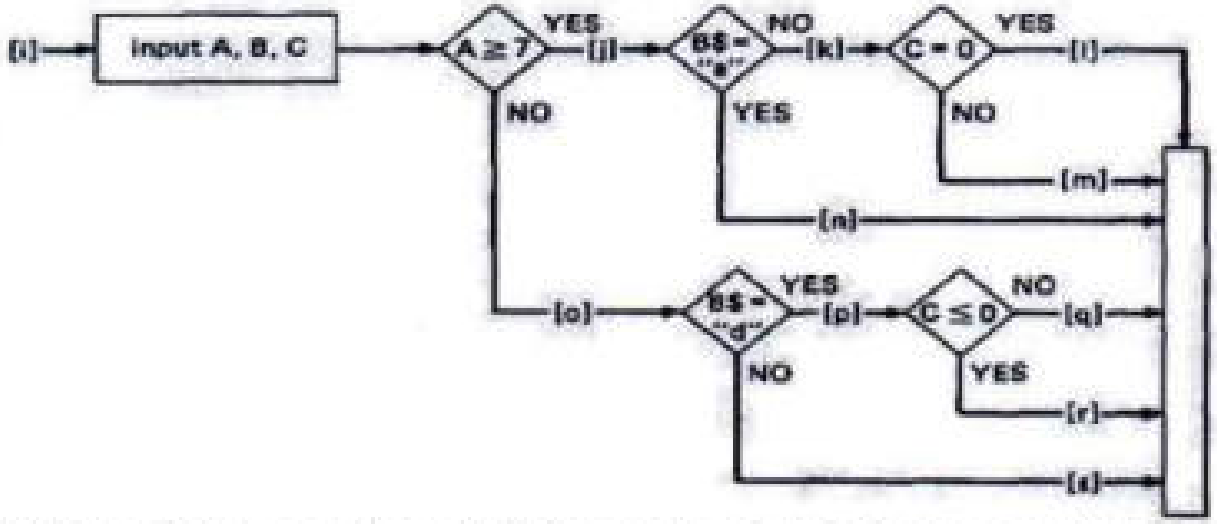


Figure : Coincidental Correctness

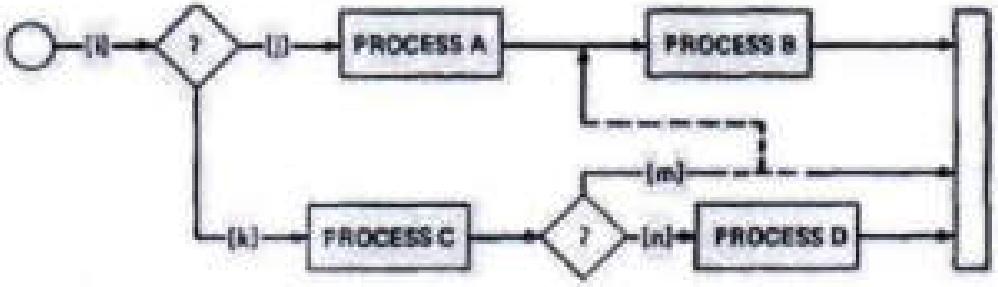
PRINCIPAL
Sri Indu Institute of Engineering & T...
Sheriguda(V), Ibrahimpatnam(M),
R.R. Dist. Telangana -501 510

Traversal Marker or Link Marker:

- A simple and effective form of instrumentation is called a traversal marker or link marker.
- Name every link by a lower case letter.
- Instrument the links so that the link's name is recorded when the link is executed.
- The succession of letters produced in going from the routine's entry to its exit should, if there are no bugs, exactly correspond to the path name.



Single Link Markers aren't enough: Unfortunately, a single link marker may not do the trick because links can be chewed by open bugs.



Two Link Marker Method:

- The solution to the problem of single link marker method is to implement two markers per link: one at the beginning of each link and on at the end.
- The two link markers now specify the path name and confirm both the beginning and end of the link.

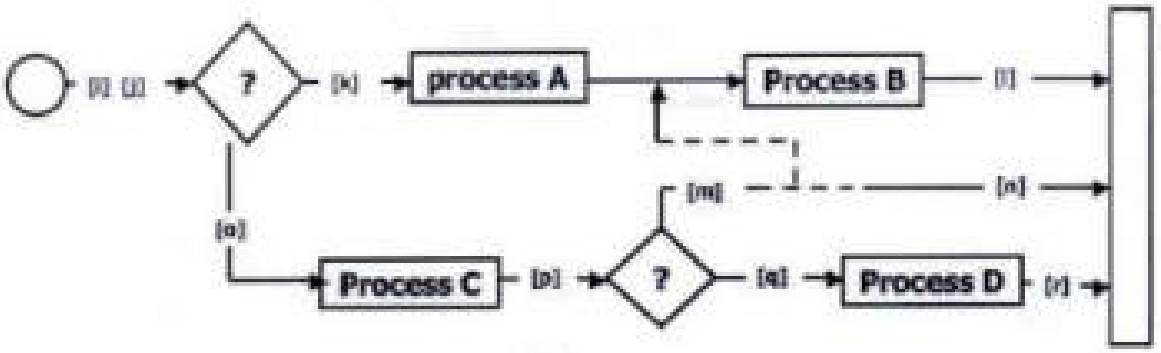


Figure : Double Link Marker Instrumentation.

3. Explain Nice and Ugly domains with an example. (Comprehension)(C324.2)(5M)

Ans:NICE DOMAINS:

- Domains are and will be defined by an imperfect iterative process aimed at achieving (user, buyer, voter) satisfaction.
- Some important properties of nice domains are: **Linear, Complete, Systematic, Orthogonal, Consistently closed, Convex and Simply connected.**

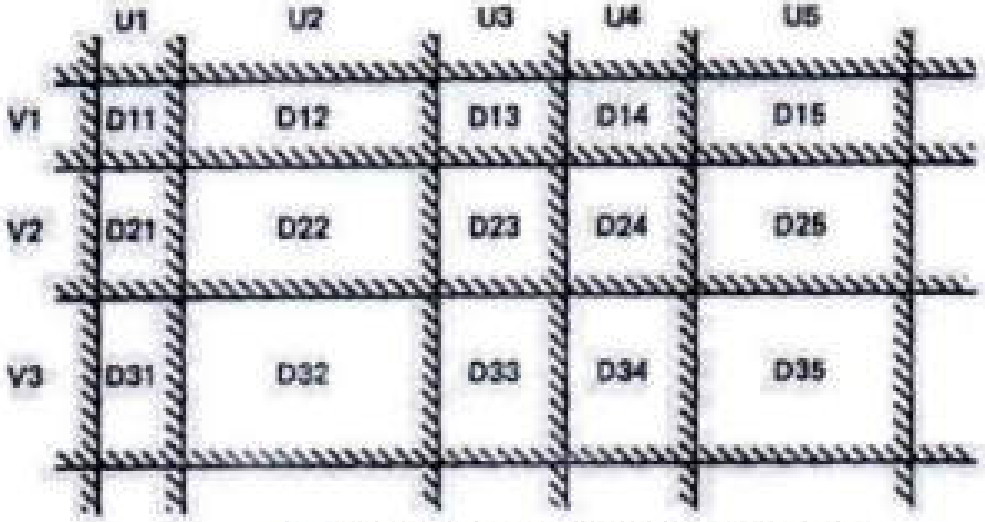


Figure: Nice Two-Dimensional Domains.

- **LINEAR AND NON LINEAR BOUNDARIES:**
  - Nice domain boundaries are defined by linear inequalities or equations.
  - The impact on testing stems from the fact that it takes only two points to determine a straight line and three points to determine a plane and in general n+1 points to determine a n-dimensional hyper plane.
- **COMPLETE BOUNDARIES:**
  - Nice domain boundaries are complete in that they span the number space from plus to minus infinity in all dimensions.
  - Figure shows some incomplete boundaries. Boundaries A and E have gaps.
  - Such boundaries can come about because the path that hypothetically corresponds to them is unachievable, because inputs are constrained in such a way that such values can't exist, because of compound predicates

that define a single boundary, or because redundant predicates convert such boundary values into a null set.

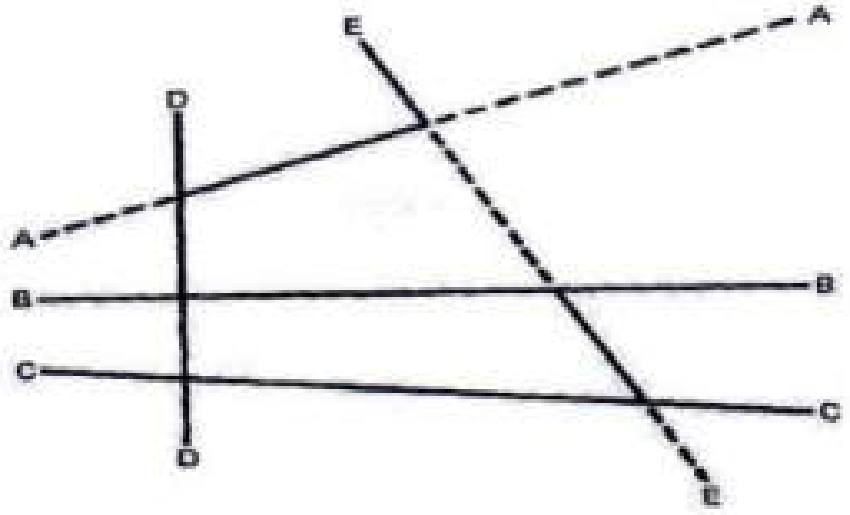


Figure 4.4: Incomplete Domain Boundaries.

SYSTEMATIC BOUNDARIES:

- Systematic boundary means that boundary inequalities related by a simple function such as a constant.
- In Figure for example, the domain boundaries for u and v differ only by a constant. We want relations such as

$$f_1(X) >= k_1 \text{ or } f_1(X) >= g(1,c)$$
$$f_1(X) >= k_2 \quad f_2(X) >= g(2,c)$$
$$\ldots\ldots\ldots\ldots \quad \ldots\ldots\ldots\ldots$$
$$f_i(X) >= k_i \quad f_i(X) >= g(i,c)$$

where f_i is an arbitrary linear function, X is the input vector, k_i and c are constants, and $g(i,c)$ is a decent function over i and c that yields a constant, such as $k + ic$.

ORTHOGONAL BOUNDARIES:

- Two boundary sets U and V are said to be orthogonal if every inequality in V is perpendicular to every inequality in U.
- If two boundary sets are orthogonal, then they can be tested independently

we have six boundaries in U and four in V. We can confirm the boundary properties in a number of tests proportional to 6 + 4 = 10 (O(n)). If we tilt the boundaries to get Figure, we must now test the intersections. We've gone from a linear number of cases to a quadratic: from O(n) to $O(n^2)$.

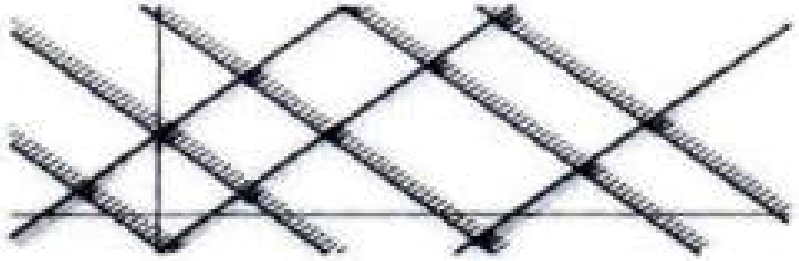


Figure : Tilted Boundaries.

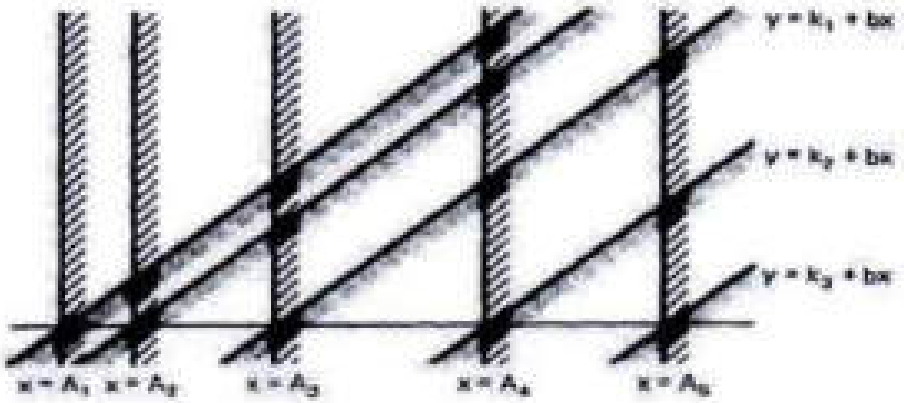


Figure: Linear, Non-orthogonal Domain Boundaries.

UGLY DOMAINS:

- Some domains are born ugly and some are uglified by bad specifications.
- Every simplification of ugly domains by programmers can be either good or bad.

AMBIGUITIES AND CONTRADICTIONS:

- Domain ambiguities are holes in the input space.

The holes may lie with in the domains or in cracks between domains.

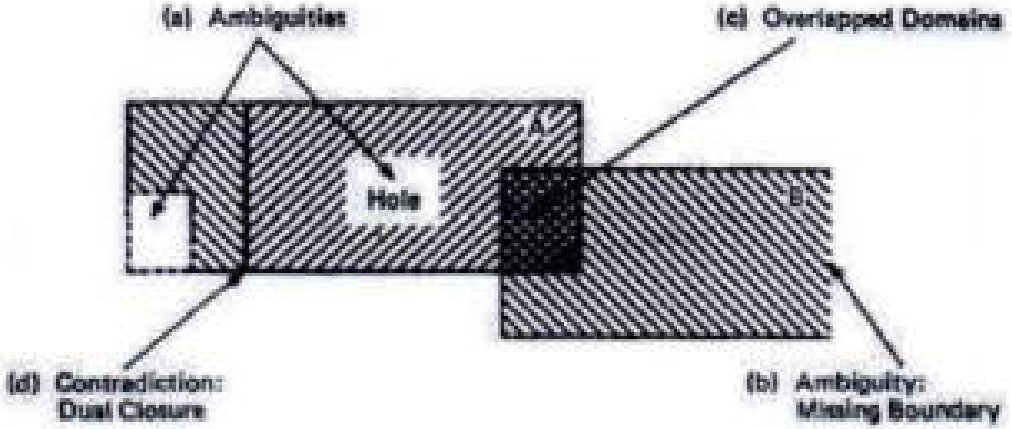


Figure : Domain Ambiguities and Contradictions.

PRINCIPAL
Sri Indu Institute of Engineering
Sheriguda(V), Ibrahimpatnam
R.R. Dist. Telangana -501 510

SIMPLIFYING THE TOPOLOGY:

- The programmer's and tester's reaction to complex domains is the same - simplify
- There are three generic cases: **concavities, holes** and **disconnected pieces**.

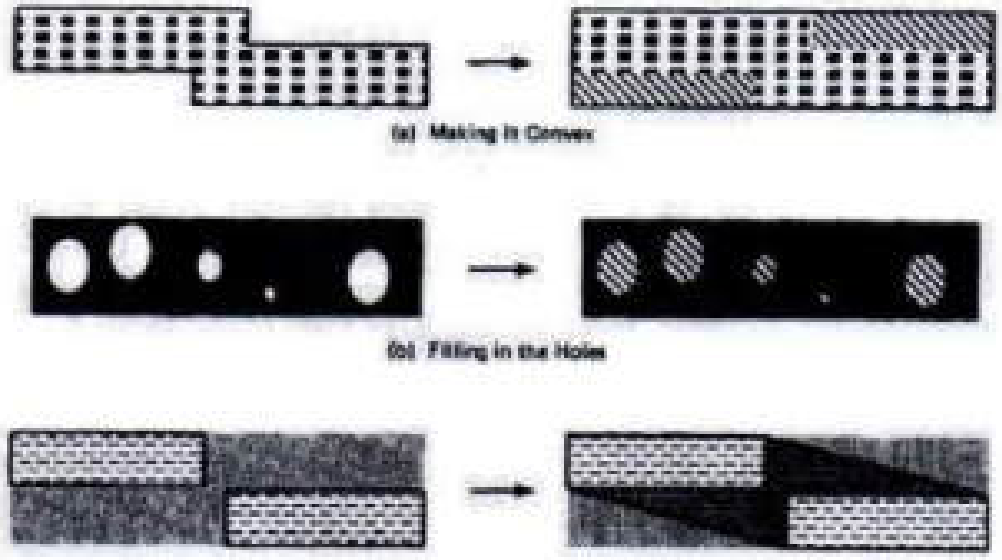


Figure : Simplifying the topology.

4. Discuss Path products and path expressions with an example.

(Knowledge) (C324.2)(5M)

Ans :

PATH PRODUCTS:

- Normally flow graphs used to denote only control flow connectivity.
- The simplest weight we can give to a link is a name.
- Every link of a graph can be given a name.
- The link name will be denoted by lower case italic letters.
- For example, if you traverse links a,b,c and d along some path, the name for that path segment is abcd. This path name is also called a **path product.** Figure shows some examples:

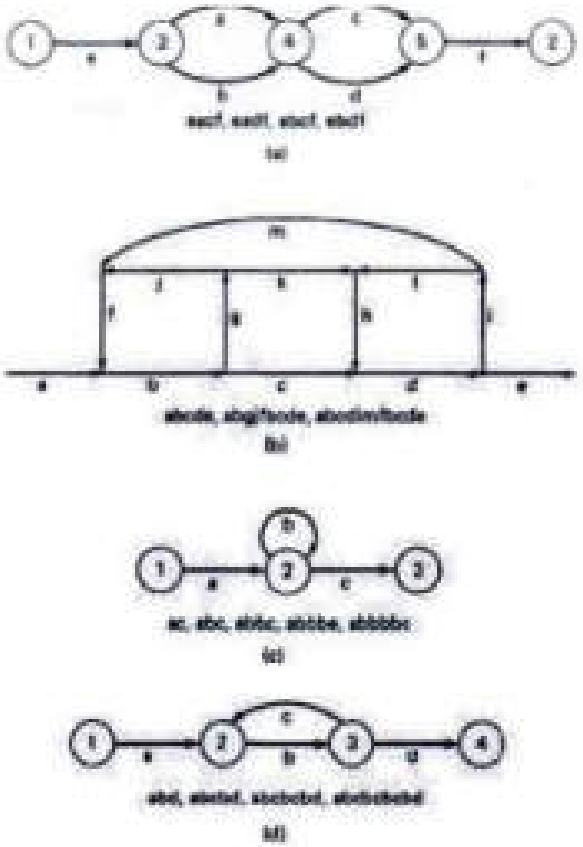


Figure : Examples of paths.

- **PATH EXPRESSION:**
  - Consider a pair of nodes in a graph and the set of paths between those node.
  - Denote that set of paths by Upper case letter such as X,Y. From Figure:c, the members of the path set can be listed as follows:

    ac, abc, abbc, abbbc, abbbbc.....

  - Alternatively, the same set of paths can be denoted by :

    ac+abc+abbc+abbbc+abbbbc+.....

  - The + sign is understood to mean "or" between the two nodes of interest, paths ac, or abc, or abbc, and so on can be taken.
  - Any expression that consists of path names and "OR"s and which denotes a set of paths between two nodes is called a **"Path Expression."**.
- **PATH PRODUCTS:**
  - The name of a path that consists of two successive path segments is conveniently expressed by the concatenation or **Path Product** of the segment names.
  - For example, if X and Y are defined as X=abcde,Y=fghij,then the path corresponding to X followed by Y is denoted by

    XY=abcdefghij

PRINCIPAL
Sri Indu Institute of Engineering & Tec
Sheriguda(V), Ibrahimpatnam(M),
R.R. Dist. Telangana -501 510

Sri Indu Institute of Engineering & Technology

Sheriguda (V), Ibrahimpatnam (M), R.R.Dist-501 510

I - Mid Examinations, MAY -2022

Set – II

Year & Branch: III-CSE(A,B,C) Date: 5-5 -22(AN)

Subject: STM Marks: 10 Time: 60 min

Answer any **TWO** Questions. All Question Carry Equal Marks 2*5=10 marks

(This question paper is prepared with Course Outcome and BT's mapping)

1.Discuss various Consequences of bugs ? (Knowledge)(C324.1)(5M)

2.a) Explain various kinds of testing blindness with examples?
(Comprehension)(C324.1)(2M)

b) Explain various loops with an example? (Comprehension)(C324.1)(3M)

3. How does Transaction flow occurs, illustrate with help of examples.
(Comprehension)(C324.2)(5M)

4.Discuss Path products and path expressions with an example.(Knowledge)(C324.3)(5M)

PRINCIPAL
Sri Indu Institute of Engineering & T
Sheriguda(V), Ibrahimpatnam(M)
R.R. Dist. Telangana -501 510

Sri Indu Institute of Engineering & Technology

Sheriguda (V), Ibrahimpatnam (M), R.R.Dist-501 510

I - Mid Examinations, MAY -2022

Set – II

Year & Branch: III-CSE(A,B,C) Date: 5-5 -22(AN)

Subject: STM Marks: 10 Time: 60 min

ANSWER KEY PAPER

1. Discuss various Consequences of bugs ? **(Knowledge)(C324.1)(5M)**

Ans: CONSEQUENCES OF BUGS: The consequences of a bug can be measure in terms of human rather than machine. Some consequences of a bug on a scale of one to ten are:

1. **Mild:** The symptoms of the bug offend us aesthetically (gently); a misspelled output or a misaligned printout.
2. **Moderate:** Outputs are misleading or redundant. The bug impacts the system's performance.
3. **Annoying:** The system's behaviour because of the bug is dehumanizing. E.g. Names are truncated orarbitarily modified.
4. **Disturbing:** It refuses to handle legitimate (authorized / legal) transactions. The ATM wont give you money. My credit card is declared invalid.
5. **Serious:** It loses track of its transactions. Not just the transaction itself but the fact that the transaction occurred. Accountability is lost.
6. **Very Serious:** The bug causes the system to do the wrong transactions. Instead of losing your paycheck, the system credits it to another account or converts deposits to withdrawals.
7. **Extreme:** The problems aren't limited to a few users or to few transaction types. They are frequent and arbitrary instead of sporadic infrequent) or for unusual cases.
8. **Intolerable:** Long term unrecoverable corruption of the database occurs and the corruption is not easily discovered. Serious consideration is given to shutting the system down.
9. **Catastrophic:** The decision to shut down is taken out of our hands because the system fails.
10. **Infectious:** What can be worse than a failed system? One that corrupt other systems even though it doesnot fall in itself ; that erodes the social physical environment; that melts nuclear reactors and starts war.

PRINCIPAL
Sri Indu Institute of Engineering & Tec
Sheriguda(V), Ibrahimpatnam(M)
R.R. Dist, Telangana -501 510

2.a) Explain various kinds of testing blindness with examples?

(Comprehension)(C324.1)(2M)

Ans: TESTING BLINDNESS:

- Testing Blindness is a pathological (harmful) situation in which the desired path is achieved for the wrong reason.
- There are three types of Testing Blindness:

1) **Assignment Blindness:**

- Assignment blindness occurs when the buggy predicate appears to work correctly because the specific value chosen for an assignment statement works with both the correct and incorrect predicate.
- For Example:

Correct	Buggy
X = 7 ..... if Y > 0 then ...	X = 7 ..... if X+Y > 0 then ...

- If the test case sets Y=1 the desired path is taken in either case, but there is still a bug.

2) **Equality Blindness:**

- Equality blindness occurs when the path selected by a prior predicate results in a value that works both for the correct and buggy predicate.
- For Example:

Correct	Buggy
if Y = 2 then ..... if X+Y > 3 then ...	if Y = 2 then ..... if X > 1 then ...

- The first predicate if y=2 forces the rest of the path, so that for any positive value of x. the path taken at the second predicate will be the same for the correct and buggy version.

3) **Self Blindness:**

- Self blindness occurs when the buggy predicate is a multiple of the correct predicate and as a result is indistinguishable along that path.
- For Example:

Correct	Buggy
X = A ..... if X-1 > 0 then ...	X = A ..... if X+A-2 > 0 then ...

- The assignment (x=a) makes the predicates multiples of each other, so the direction taken is the same for the correct and buggy version.

PRINCIPAL
Sri Indu Institute of Engineering & Tec
Sheriguda(V), Ibrahimpatnam(M)
R.R. Dist. Telangana -501 510

b) Explain various loops with an example? (Comprehension)(C324.1)(3M)

Ans: There are three different kinds of loops. They are as follows

i)Nested loops

ii)Concatenated loops

iii)Horrible loops

i)Nested loops: The nested loops are quite complicated 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.

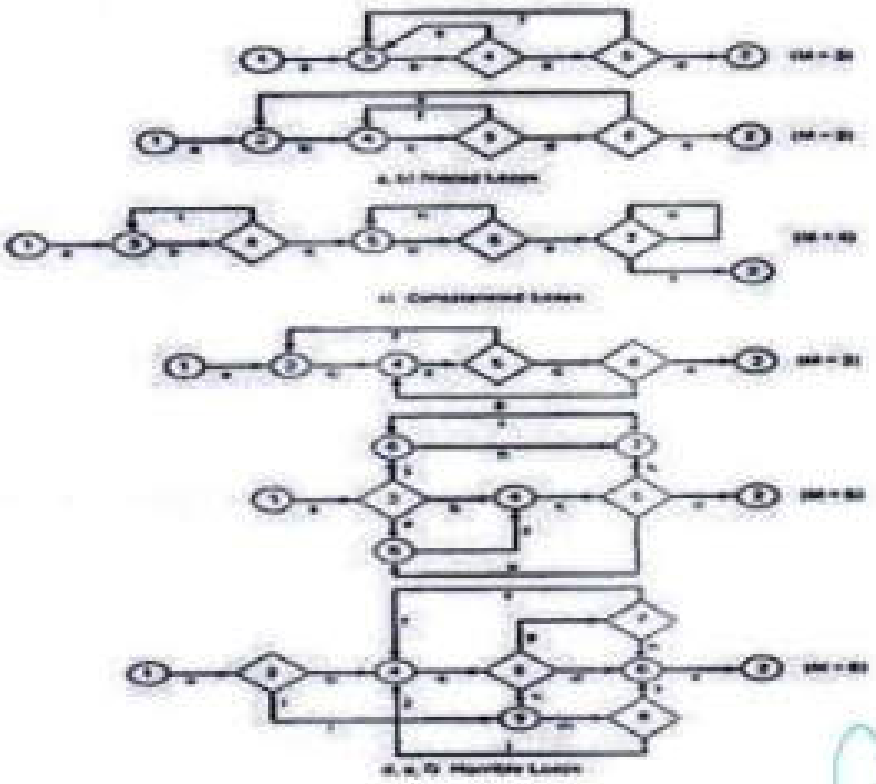
a)Begin your test from the most internal loop.move towards the external /outer loop there by keeping its value to be maximum.

ii) Concatenated Loops:

- Concatenated loops fall between single and nested loops with respect to test cases. Two loops are concatenated if it's possible to reach one after exiting the other while still on a path from entrance to exit.
- If the loops cannot be on the same path, then they are not concatenated and can be treated as individual loops.

iii) Horrible Loops:

- A horrible loop is a combination of nested loops, the use of code that jumps into and out of loops, intersecting loops, hidden loops, and cross connected loops.
- Makes iteration value selection for test cases an awesome and ugly task, which is another reason such structures should be avoided.



PRINCIPAL
SR Indu Institute of Engineering & Tc..
Sheriguda(V), Ibrahimpatnam(M),
R.R. Dist. Telangana -501 510

3. How does Transaction flow occurs, illustrate with help of examples.

(Comprehension)(C324.2)(5M)

Ans: A transaction is a unit of work seen from a system user's point of view.

- A transaction consists of a sequence of operations, some of which are performed by a system, persons or devices that are outside of the system.
- Transaction begin with Birth-that is they are created as a result of some external act.
- At the conclusion of the transaction's processing, the transaction is no longer in the system.
- **Example of a transaction:** A transaction for an online information retrieval system might consist of the following steps or tasks:
    - Accept input (tentative birth)
    - Validate input (birth)
    - Transmit acknowledgement to requester
    - Do input processing
    - Search file
    - Request directions from user
    - Accept input
    - Validate input
    - Process request
    - Update file
    - Transmit output
    - Record transaction in log and clean up (death)

TRANSACTION FLOW GRAPHS:

- Transaction flows are introduced as a representation of a system's processing.
- The methods that were applied to control flow graphs are then used for functional testing.
- Transaction flows and transaction flow testing are to the independent system tester what control flows are path testing are to the programmer.
- The transaction flow graph is to create a behavioral model of the program that leads to functional testing.
- The transaction flowgraph is a model of the structure of the system's behavior (functionality).

An example of a Transaction Flow is as follows:

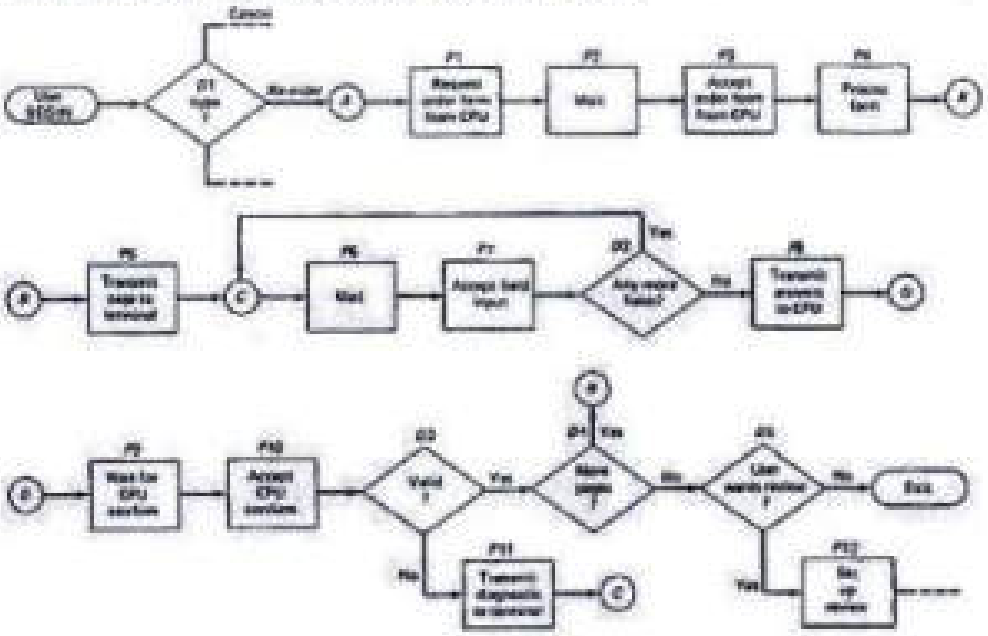


Figure : An Example of a Transaction Flow

4. Discuss Path products and path expressions with an example.

(Knowledge)(C324.3)(5M)

Ans: PATH PRODUCTS:

- Normally flow graphs used to denote only control flow connectivity.
- The simplest weight we can give to a link is a name.
- Every link of a graph can be given a name.
- The link name will be denoted by lower case italic letters.
- For example, if you traverse links a,b,c and d along some path, the name for that path segment is abcd. This path name is also called a **path product.** Figure shows some examples:

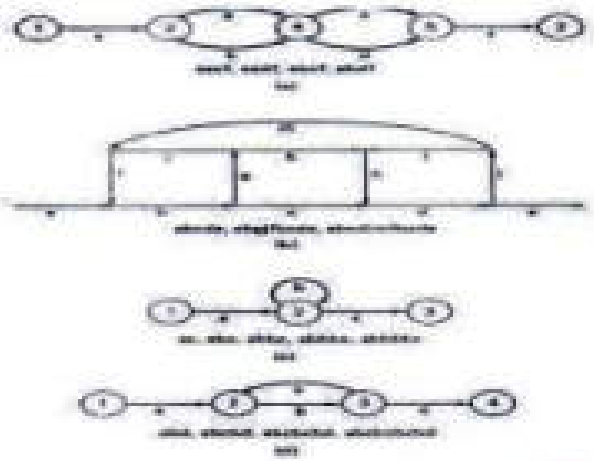


Figure : Examples of paths.

PRINCIPAL
Sri Indu Institute of Engineering & T
Sheriguda(V), Ibrahimpatnam
R.R. Dist. Telangana -501 510

- **PATH EXPRESSION:**
  - Consider a pair of nodes in a graph and the set of paths between those node.
  - Denote that set of paths by Upper case letter such as X,Y. From Figure:c, the members of the path set can be listed as follows:

    ac, abc, abbc, abbbc, abbbbc.....

  - Alternatively, the same set of paths can be denoted by :

    ac+abc+abbc+abbbc+abbbbc+.....

  - The + sign is understood to mean "or" between the two nodes of interest, paths ac, or abc, or abbc, and so on can be taken.
  - Any expression that consists of path names and "OR"s and which denotes a set of paths between two nodes is called a **"Path Expression."**.
- **PATH PRODUCTS:**
  - The name of a path that consists of two successive path segments is conveniently expressed by the concatenation or **Path Product** of the segment names.
  - For example, if X and Y are defined as X=abcde,Y=fghij,then the path corresponding to X followed by Y is denoted by

    XY=abcdefghij

PRINCIPAL
Sri Indu Institute of Engineering & Te
Sheriguda(V), Ibrahimpatnam(
R.R. Dist. Telangana -501 510

SRI INDU INSTITUTE OF ENGINEERING AND TECHNOLOGY
Accredited by NAAC A+ Grade, Recognized under 2(f) of UGC Act 1956.
(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)
Khalsa Ibrahimpatnam, Sheriguda(V), Ibrahimpatnam(M), Ranga Reddy Dist.,Telangana
https://siet.ac.in/

Department of Computer Science and Engineering
2021-22; 2nd Semester

Assignment Questions-I

(Assignment Questions are mapped with CO's, BT)

1.a) .Distinguish between different Dichotomies (C324.1)(Comprehension)
b) Explain phases in Tester's mental life. (C324.1)(Comprehension)

2.Explain about path instrumentation. (C324.1)(Comprehension)

3.Explain the transaction flow testing with an example. (C324.2)(Comprehension)

4.Explain the basics of dat flow testing and the strategies in data flow testing. (C324.2)(Comprehension)

5.Explain each of the following (C324.3)(Comprehension)
a) Path Expression
b) Path Product
c) Path Sum
d) Loops

PRINCIPAL
Sri Indu Institute of Engineering &
Sheriguda(V), Ibrahimpatnam
R.R. Dist. Telangana-501 510

Software Testing Methodologies

19X31A05E4
CSE-C

Assignment-01

1) a. Distinguish between different Dichotomies

1. Testing versus Debugging

Testing	Debugging
1. The goal of Testing is to detect errors in a program	The goal of Debugging is to detect errors and correct them.
2. Testing is initiated with known conditions	Debugging is initiated with unknown conditions.
3. The output can be anticipated	cannot be anticipated.
4. It is neccessary to have planned, designed & scheduled	It is not neccessary to have these constraints.
5. Testing finds out the reason for program's failure.	Debugging is the programmers justification.
6. It is not necessary to have design knowledge while performing testing.	It is sufficient to have detailed design knowledge about for debugging.
7. The test design and execution are done automatically	Designing and execution cannot be done automatically.

2. Small versus Large

Small	Large
1. Small programs have only few lines of code	Large programs have more number of lines of code.
2. They consists of few components.	They consists of Large number of components.
3. Small programs does not require any technique for testing.	They require different types of techniques for testing.
4. Small programs are more efficient	Large programs are Less efficient
5. small programs are written by single programmer	Large programs are written by different programmers.

PRINCIPAL
Sri Indu Institute of Engineering & Tech.
Sheriguda(V), Ibrahimpatnam(M),
R.R. Dist. Telangana 501 510

3. Functional versus Structural Testing.

Structural Testing	Functional Testing
1. structural Testing is also known as white-box, glass-box testing	Functional Testing is also known as black-box, glased-box
2 structural Tests are performed based on the knowledge of internal structure of sourcecode.	Functional Tests are performed without internal structure of the software.
3 The test cases take finite time, but cannot detect all bugs	The Test cases take infinite time and detect all errors
4 structural Testing is less effective when compared to functional Testing.	Functional Testing is more effective than glass-box testing
5 different methods for performing white-box Testing are as follows. i) statement Testing ii) Decision Testing iii) condition Testing.	Different methods for performing black-box testing are as follows i) expected inputs method ii) Boundary values method iii) Illegal values method

4 The Designer versus Tester.

Designer	Tester
1. Designer is based on the structural specifications of the system.	Functional specification of the system.
2 He/she depends on the implementation details	Independent on implementation details.
3 A s/w design is responsible for designing + executing the tests	A tester is responsible for designing and executing the tests.

b) Explain phases in a Tester's Mental Life

A tester consider 5 phases of Thinking. These phases are classified as follows

i) phase 0 Thinking
ii) phase 1 Thinking
iii) phase 2 Thinking
iv) phase 3 Thinking
v) phase 4 Thinking

i) phase 0 Thinking :- (Thinking criteria)

Testing and Debugging are similar Testing is usefull in debugging when testing come into existence, phase0 Thinking was the criteria for software development This thinking was applicable to an area forom which the following resources can be determined which are as follows.

i) High cost and inadequate computing resources
ii) Low cost software resources
iii) single programmers
iv) small profect resources
v) Irrelavent software resources

ii) phase 1 :- (Thinking the s/w or program works)

The aim of this Technique is to display the working of a program or s/w phase1 Thinking identifies The distinct relationship between testing and debugging. In this testing phase the number of tests performed on a software will not be executed even if the subsequent tests make its execution possible

iii) phase 2 :- (Thinking the program doesnt work)

The aim of this type of thinking is to demonstrate that the program doesnt execute. The Thinking of testers in phase 2 opposes the Thinking of designers If a test fails, then the goal of phase 2 thinking. bug can be detected by testers

programmers and designers.

i) The testes detects an error (bug)

ii) The programmer verifies and corrects the error.

iii) The designer designs test

iv) The designer aims to different tests to identify different errors.

iv) phase 3:- (Thinking Test for reducing the Risk)

The aim of Testing is to reduce the risks that are predictable. The phase 3 Thinking accepts the rules of statistical quality control to improve the quality of s/w. In the process of phase 3 Thinking, the testes detects, the bugs and eliminate them.

v) phase 4:- (Thinker's knowledge)

The aim of phase 4 Thinking is to reduce the risks of s/w with minimal Testing effort s/w that require minimum testing in order to attain two goals of lower phases testing is obtained by integrating the capability of testing with the knowledge of knowing the conditions under which s/w is made testable.

2. Explain about path instrumentation

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

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

To overcome this drawback, many different instrumentat-ions methods have evolved.

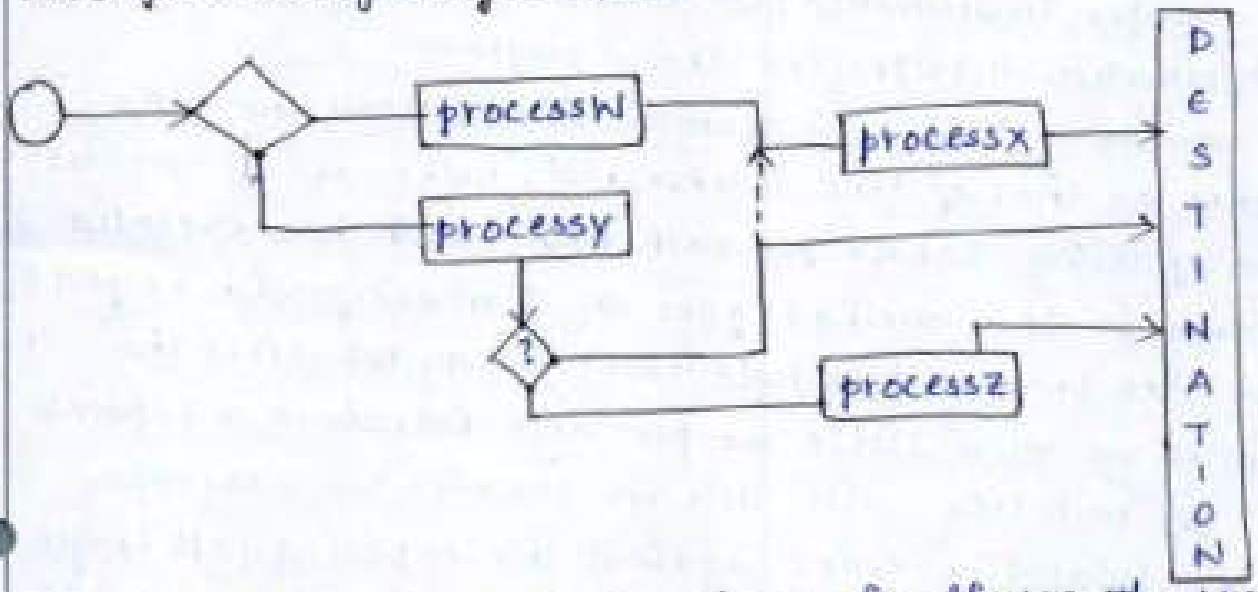
Link Marker Method of the Path Instrumentation:-

Link Marker is also known as Traversal Marker which uses small case Letters for naming every link.

whenever a Link is passed, it's 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 server marker may not serve the purpose because there is every possibility of bug which may result in a new link in the middle of the link being traversed.

Example: using single Link Markers.



In the above example as shown in figure. 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 Link.

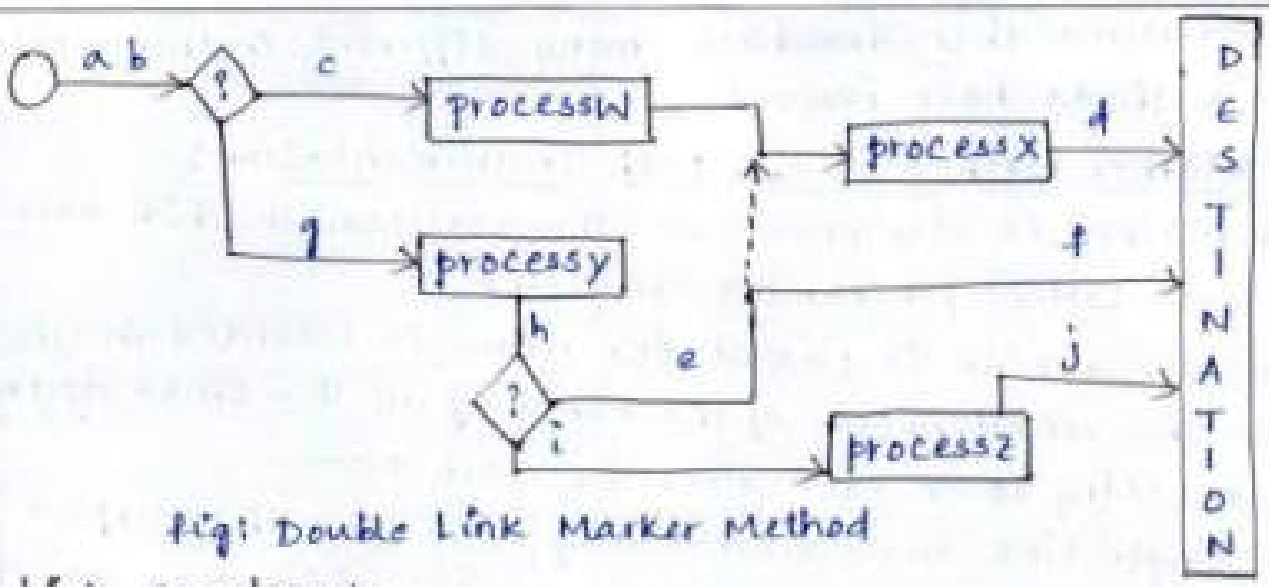


fig: Double Link Marker Method

Link counters:-

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

Link counter instrumentation method provides comparitively less information 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 each link with this in practice we can cross check the total link count against the expected path length this format is not reliable because there is every possibility of having a bug. which may result in a newlink in the midale 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.

3 Explain the Transaction Flow Testing with an example

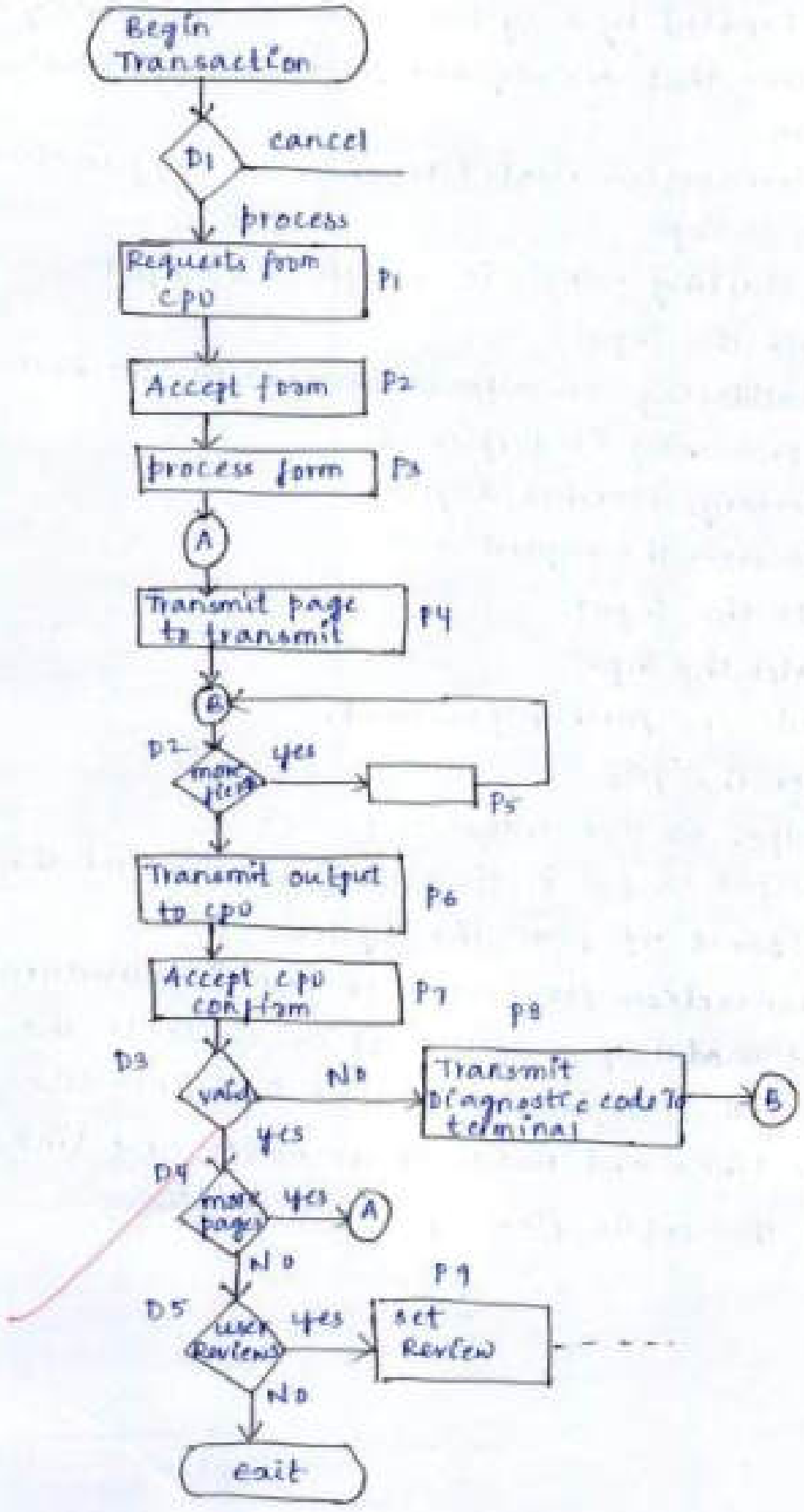
A Transaction is defined as a set of statements or a unit of work handled by a system user. It is actually a collection of operations that are required for handling a particular transaction

Every transaction related to an on-line information system consists of 12 steps

1 At the 'starting point', it accepts the input
2. validate the input
3 After validating, an acknowledgement is sent back to the user
4 Input processing is performed.
5 If necessary, searches a file.
6 user processes the request
7 Accepts the input
8 validates the input
9 Requests are further processed.
10 updates the file.
11 The output is then transmitted
12 The output is fed in the form of records and the transaction is cleaned up from the system.

The transaction flow occurs to test the structural and behavioural model of a system It is similar to the control flow graph the components involved in transaction flow graph are links and nodes these nodes and links represent the entire flow of a transaction.

The following flowgraph illustrate the processing of a transaction using forms



4. Explain the basics of Data Flow Testing and the strategies in Data Flow Testing.

Data Flow Testing is the name given to a family of test strategies based on selecting paths through the program's control flow in order to explore sequences of events related to the status of data objects.

Strategies in Data Flow Testing:

Terminology:

Different Terminologies have been defined to fulfill the graphs in path Testing. The terminologies are,

1. Definition - clear path segment
2. Loop - free path segment
3. simple path segment
4. Du path.

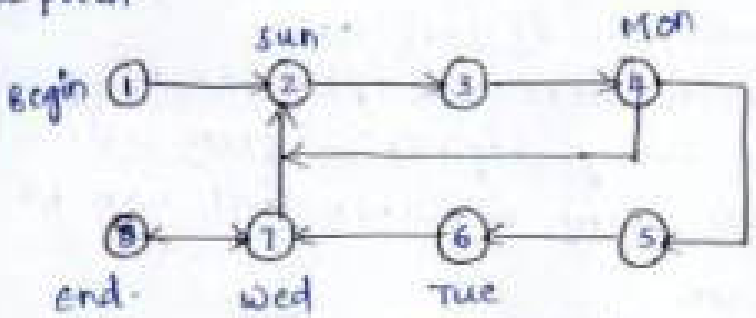


fig: control Flow graph.

1) Definition clear path segment: A path segment is a sequence of connected links between nodes. The first link of the path is defined and the subsequent link of that path is killed.

The path (1,2) (3,4) (5,6) are definition clear. The path (1,2,3,4,5,6) is not definition clear because the variable is defined on (2,3) (4,5) and again on (6,7). A definition-clear-sub path doesn't include Loops.

PRINCIPAL
Sri Indu Institute of Engineering & Technology
Sheriguda(V), Ibrahimpatnam(M)
R.R. Dist. Telangana-501 510

2) Loop-free path segment:- The path which doesn't have any loop and every node of which is visited atmost once is known as Loop-free path segment

Dataflow testing strategies are based on the program's control flow graphs

1) All du-path (ADUP) strategy:-

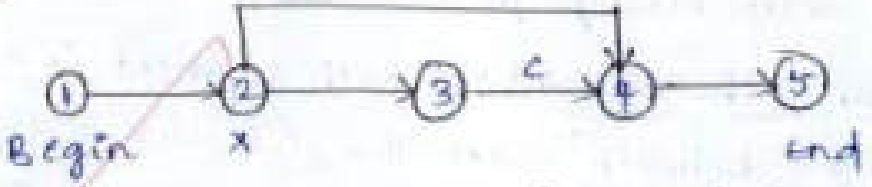
A du-path is a path which has a defined variable and has a computational use as a predicate use of that variable

Def:- All du-paths strategy is a data-flow testing strategy that requires every definition of every variable from every du-path to use every variable of that definition

2) All uses (AU) strategy:-

All users helps in reducing no of test cases.

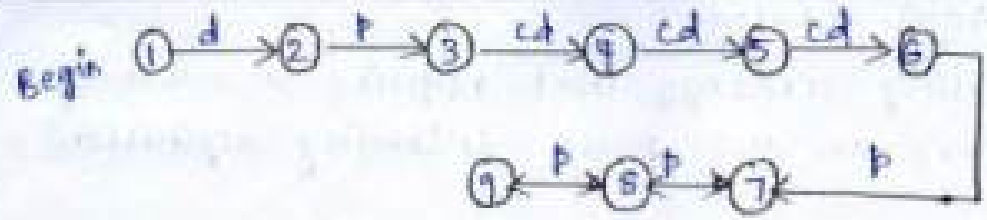
Def:- All uses strategy includes dataflow testing strategy that requires every definition from atleast one path segment to use every variable that can be reach by that definitation



3) All p-uses or some c-uses (APU+C) strategy:-

If a variable has a predicate use then there is no need to select computational use.

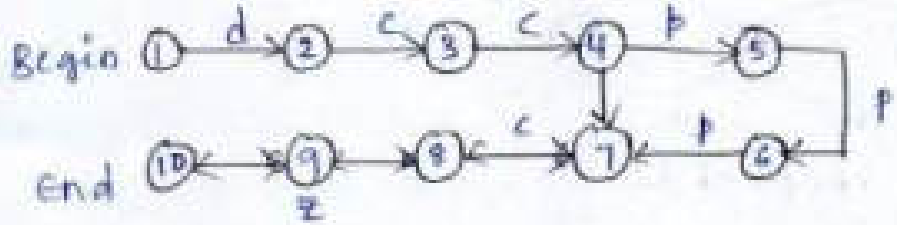
Def:- APU+C is a strategy that requires every definition of every variable included atleast one path from that definition to every predicate use.



4) All c-uses or some p-uses (ACU+P) strategy:-

In this testing strategy first ensure that every definition has a computational use of that definition and if any definition is not covered, predicate usecases added to assure that every definition is included is some test case.

Def:- ACU+P is a testing strategy that requires every def of every variable included atleast one path from that definition to every computational use.



5) All definitions (AD) strategy:-

AD strategy is weaker than ACU+P and APUC strategies.

Def:- AD is a testing strategy that requires every definition of every variable to cover atleast one use of that variable the use can be computational use 'c' or a predicate use 'p'.

6) All p-uses (APU) strategy:-

In this strategy every def of every variable has a path to every p-use of that def. If there is no p-use in that def then it is droped from contention.

Def:- APU is a testing strategy that requires a p-use of def if there are no c-use in the following definition.

7) All c-use (ACU) strategy:-

Def:- ACU is testing strategy that require a c-use of definition if there are not p-use following definition

5 Explain each of the following

a) path expression

b) path product

c) path sum

d) Loops.

a) path expressions:-

path expression is defined as an expression which represents set of all the possible paths between an entry and exit nodes.

Ex:

eacf, eadf, ebcf, ebdf.

b) path products:-

The concatenation of names of two consecutive path segments can be termed as path product.

Ex: Let A and B are 2 path segments which has 'abcd' and 'efgh' paths respectively, then the path product with A preceeding B will be,

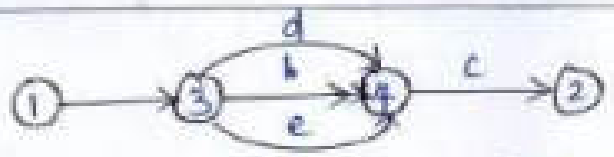
AB = abcd * efgh = abcdefgh

c) path sums:-

path sum is the sum of all the parallel paths available between 2 nodes in which paths passing through intermediate nodes are also considered as parallel.

path sum is denoted by '+' sign.

ex:

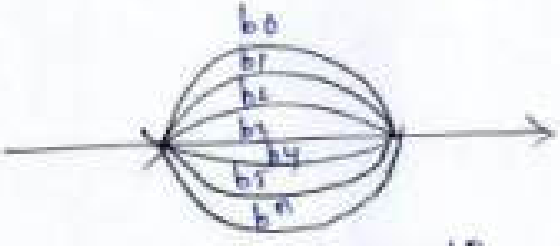


d,b,e are links between the nodes 3 and 4 and hence they are refered as parallel paths and represented by d+b+e.

Loops:-

Loops can be defined as a situation which occurs when a single link path expression is being traversed indefinite numbers of times leading to infinite number of parallel paths.

Loops may result in never ending path expression for any flow graph. If a loop is a single link, it can be termed as self loop and indefinite numbers of repitions of that link is denoted by an asterisk (*) placed on the link name. If the loop must be taken atleast once, then it is denoted by 'at' or 'xt'.



$b^0+b^1+b^2+b^3+\cdots b^n$

Ex: $ab^*c = ac+abc+abbc+abbbc+\cdots$

SRI INDU INSTITUTE OF ENGINEERING AND TECHNOLOGY
Accredited by NAAC A+ Grade, Recognized under 2(f) of UGC Act 1956.
(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)
Khalsa Ibrahimpatnam, Sheriguda(V), Ibrahimpatnam(M), Ranga Reddy Dist.,Telangana
https://siiet.ac.in/

Department of Computer Science and Engineering
2021-22; 2nd Semester

Assignment Questions-II

(Assignment Questions are mapped with CO's, BT)

1 . Differentiate between good state graph and bad state graph.

2. What is a decision table ? Discuss the role of decision table in a test case design.

3. Explain the rules of Boolean Algebra.

4. What are principles of state testing ?Explain its advantages and disadvantages.

5.Write an algorithm for Node Reduction algorithm.

PRINCIPAL
Sri Indu Institute of Engineering &
Sheriguda(V), Ibrahimpatnam
R.R. Dist. Telangana -501 5

Assignment-02 S. Mamatha Reddy

① Difference between good state graph and bad state graph.

Difference between good state graph and bad state graph

	Good state Graph	Bad state Graph
1)	State graph is said to be good, when every state i/p transition & o/p is specified clearly and understandable	1) A state graph is said to be bad when every state i/p, transition o/p is not specified clearly and difficult to understand.
2.	In good state graph, the sequence of i/ps is specified for every state in order to perform some action that will help the system back to the initial state.	2) In bad state graph, there is no sequence of i/p's specified, that result to the in correct o/p
3	In good state graph there is Exactly one transition specified for every state and i/p combination so that the transition bugs may not occur	3) In bad state graph, there might be either none or more than one transitions specified for every state and i/p combination that results to
4	In good state graph, the bugs are less and easy to identify.	4) In bad state graph, the bugs are more and difficult to identify.

② What is a decision table? Discuss the role of decision table in a test case design.

Decision Tables.

A decision table is a tabular form that consists of a set of conditions and their respective actions. A decision table consists of four parts. They are

→ condition stub
→ Condition entry
→ Action stub
→ Action entry.

PRINCIPAL
Sri Indu Institute of Engineering & Technology
Sheriguda(V), Ibrahimpatnam
R.R Dist. Telangana-501 510

Condition stub:- It consists of a list of causes for a rule to be satisfied. conditions are marked with Yes/No. "Yes" indicates that the condition is satisfied and "No" Specifies that the condition is not satisfied. The variable 'I' specifies the immaterial test case, which means that a particular condition doesn't have any role in that rule.

Action Stub:- It describes the possible actions that are to be executed and "No" Specifies that an action need not be executed.

Table 1: An Example of a Decision Table

	RULE 1	RULE 2	RULE 3	RULE 4
Condition Stub	CONDITION 1	YES	YES	NO	NO
	CONDITION 2	YES	I	NO	I
	CONDITION 3	NO	YES	NO	I
	CONDITION 4	NO	YES	NO	YES
Action Stub	ACTION 1	YES	YES	NO	NO
	ACTION 2	NO	NO	YES	NO
	ACTION 3	NO	NO	NO	YES

ACTION ENTRY

Applications Of Decision Tables

→ Decision tables are in business data processing

→ Decision table processors are generally used for generating a high-level code. The translator in the processor is used to examine the decision table for consistency and completeness. These processors are implemented using boolean algebra.

③ Explain the rules of Boolean algebra.

Boolean Algebra Rules:-

The different operators that are used while solving any boolean algebra are,

i) **Multiplication (x):-** This symbol specifies the AND operation. A statement 'pxq' is said to be true if both statements p and q are true, otherwise it is false.

pxq Truth Table

p	q	pxq
T	T	T
T	F	F
F	T	F
F	F	F

ii) **Addition (+):** This symbol specifies the "OR" operation. A statement 'p+q' is said to be true if either p is true or both are true.

p+q Truth Table

p	q	p+q
T	T	T
T	F	T
F	T	T
F	F	F

Negation (-):- It specifies the complement operation. It is referred as "NOT" operator. It is represented as $\bar{p}$ which means that statement is true if the statement p is false.

$\bar{p}$ Truth Table

p	$\bar{p}$
T	F
F	T

PRINCIPAL
Sri Indu Institute of Engineering &
Sheriguda(V), Ibrahimpatnam
R.R. Dist. Telangana-501 510

④ What are the principles of state testing? Explain its advanges and disadvantages?

State Testing:-

State Testing is defined as a functional testing technique to test the functional bugs in the entire system.

For path testing it is not possible to test every possible path in a flow graph. Similarly for state testing, it is not possible path in a flow graph. In a state graph a path is a sequence of transitions caused by a sequence of i/p's like path testing state testing examine each transition in a stategraph to guarantee complete testing.

A bug can manifest itself as one or more of the following symptoms

→ Wrong number of states.
→ Wrong transition for a given state i/p combination
→ Wrong o/p for a given transition.
→ States or sets of states that are split to create inequivalent duplicates.
→ States or sets of states that have become dead.
→ States or sets of states that have become unreachable.

Advantages:-

The advantages of states testing are as follows,

→ State testing is useful when the error corrections are less.

Disadvantages:-

→ state testing does not provide through testing because when a test completed, these might be some bugs remained in the system
→ Testers requires large number of input sequencess to catch transition error missing state, extra states and the link.

5 Write an algorithm for Node Reduction algorithm

Node Reduction Algorithm:-

→ The matrix powers usually tell us more than we want to know about most graphs.

→ In the context of testing, we usually interested in establishing a relation between two nodes typically the entry and exit nodes.

→ In a debugging context it is unlikely that we would want to know the path expression between every node and every other node.

→ The advantage of matrix reduction method is that it is more methodical than the graphical method called as node by node removal algorithm

1. Select a node for removal; replace the node by Equivalent links that bypass that node and add those links to the links they parallel.

2. Combine the parallel terms and simplify as you can.

3. Observe loop terms and adjust the out links of every node that had a self loop to account for the effect of the loop.

4. The result is a matrix whose size has been reduced by 1. Continue until only the two nodes of interest exist.

Node Reduction Algorithm

	1	2	3	4	5
1			a		
2					
3		d		b	
4		c			f
5		g	e		h

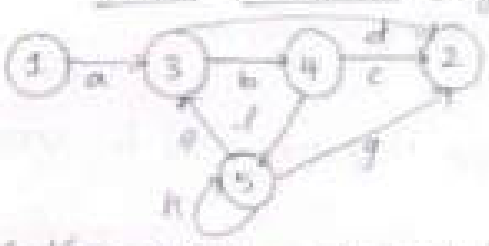
Step 1: Eliminate a node and replace it with a set of Equivalent links...

Say, we start with eliminating node 5

I/P: The out-link of the node removed will correspond to the row and the in-link will correspond to the column

PRINCIPAL
Sri Indu Institute of Engineering & T...
Sheriguda(V), Ibrahimpatnam(M),
R. Dist. Telangana-501 510

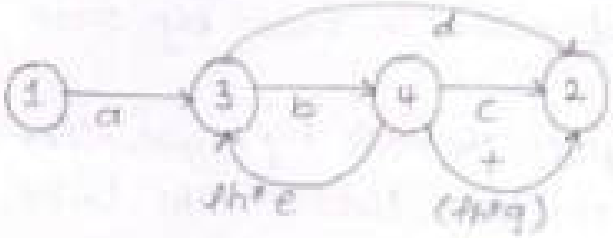
Node Reduction Algorithm



	1	2	3	4	5
1			a		
2					
3		d		b	
4		c			f
5		h*g	h*e		

Eliminating node 5... the e*h loop is represented with e* and the outgoing link from the node is multiplied.

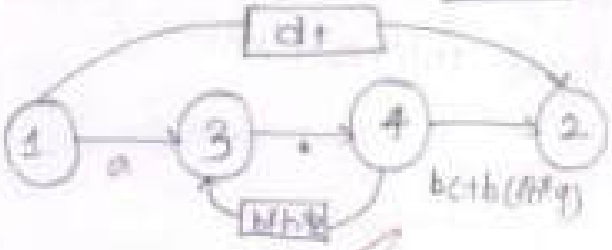
Node Reduction Algorithm



	1	2	3	4	c
1			a		
2					
3		d		b	
4		d+b c*g	bh*g		
c					

Now we eliminate node 5

Node Reduction Algorithm



	1	2	3	4	5
1			a		
2					
3		d+bc+ bh*e	bh*g		
4					
5					

Now we eliminate Node 4

Removing the loop term yields (bh*e)

		a
	bh*e*x (d+bc+bh*g)	

The final result yields to:

a (bh*e)* (d+bc+bh*g)

SRI INDU INSTITUTE OF ENGINEERING AND TECHNOLOGY
Accredited by NAAC A+ Grade, Recognized under 2(f) of UGC Act 1956.
(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)
Khalsa Ibrahimpatnam, Sheriguda(V), Ibrahimpatnam(M), Ranga Reddy Dist.,Telangana
https://siiet.ac.in/

Department of Computer Science and Engineering
2021-22; 2nd Semester

Tutorial Topics

Tutorial1: Dichotomies

Tutorial2: Basics concepts of path testing

Tutorial3: Path sensitizing

Tutorial4: Transaction flow testing techniques

Tutorial5: Application of dataflow testing

Tutorial6: Domainsand testability

Tutorial7: Regular expressions & flow anomaly detection

Tutorial8: Transition testing

Tutorial9: State Testing

Tutorial10: Node reduction algorithm

PRINCIPAL
Sri Indu Institute of Engineering &
Sheriguda(V), Ibrahimpatnam
R.R. Dist. Telangana -501 5

SRI INDU INSTITUTE OF ENGINEERING AND TECHNOLOGY

Department of Computer Science and Engineering

Course Outcome Attainment (Internal Examination-1)

Name of the faculty E.RUPA — Academic Year: 2021-22

Branch & Section: CSE-C — Examination: I Internal

Course Name: **Soft Ware Testing Methodologies** — Year: III — Semester: II

S.No	HT No.	Q1a	Q1b	Q1c	Q2a	Q2b	Q2C	Q3A	Q3b	Q3c	Q4a	Q4b	Q4c	Obj1	A1
Max. Marks ==>		5			5			5			5			10	5
1	18X31A05C8														5
2	18X31A05C9														5
3	18X31A05D4														5
4	18E31A05FB														5
5	18X31A05G4														5
6	19X31A05C1							5			4			7	5
7	19X31A05C2	5												5	5
8	19X31A05C3	3									2			6	5
9	19X31A05C4							2			4			6	5
10	19X31A05C5				4						3			6	5
11	19X31A05C6				5									5	5
12	19X31A05C7										2			6	5
13	19X31A05C8				3						4			6	5
14	19X31A05C9				3						3			6	5
15	19X31A05D0							1			2			6	5
16	19X31A05D1													6	5
17	19X31A05D2										4			6	5
18	19X31A05D3										4			5	5
19	19X31A05D4													5	5
20	19X31A05D5													6	5
21	19X31A05D6							5			5			5	5
22	19X31A05D7										4			6	5
23	19X31A05D8													7	5
24	19X31A05D9													7	5
25	19X31A05E0				1						3			6	5
26	19X31A05E1														5
27	19X31A05E2	5									4			6	5
28	19X31A05E3	2						2						5	5
29	19X31A05E4							5			5			8	5
30	19X31A05E5				5						4			7	5
31	19X31A05E6				3						3			6	5
32	19X31A05E7				5						5			8	5
33	19X31A05E8				4									6	5
34	19X31A05E9				5						5			8	5
35	19X31A05F0										5			5	5
36	19X31A05F1	4			2									6	5
37	19X31A05F2				4						3			6	5
38	19X31A05F3				3						5			6	5
39	19X31A05F4				2						5			6	5
40	19X31A05F5				3			3						6	5

PRINCIPAL
Sri Indu Institute of Engineering &
Sheriguda(V), Ibrahimpatnam,
R.R. Dist. Telangana -501 510

41	19X31A05F6	2													7	5
42	19X31A05F7				5										7	5
43	19X31A05F8				4						2				7	5
44	19X31A05F9										4				7	5
45	19X31A05G0				5						5				7	5
46	19X31A05G1										5				6	5
47	19X31A05G2										4				6	5
48	19X31A05G3	1									5				7	5
49	19X31A05G4	1									5				6	5
50	19X31A05G5				5						4				6	5
51	19X31A05G6										3				6	5
52	19X31A05G7										4				6	5
53	19X31A05G8														7	5
54	19X31A05G9														7	5
55	19X31A05H0															5
56	19X31A05H1				1						5				7	5
57	19X31A05H2							3							6	5
58	19X31A05H3							3			5				6	5
59	19X31A05H4				5						5				7	5
60	19X31A05H5	4									5				7	5
61	19X31A05H6	5			5										7	5
62	19X31A05H7				3						3				6	5
63	19X31A05H8															5
64	19X31A05H9														6	5
65	19X31A05I0				1						5				6	5
66	20X35A0513															5
67	20X35A0514				4										6	5
68	20X35A0515				2						2				7	5
69	20X35A0516	3									3				7	5
70	20X35A0517	4													6	5
71																
72																
73																
Target set by the faculty / HoD		3.00	0.00	0.00	3.00	0.00	0.00	3.00	0.00	0.00	3.00	0.00	0.00	6.00	3.00	
Number of students performed above the target		8	0	0	20	0	0	6	0	0	36	0	0	54	70	
Number of students attempted		12	0	0	26	0	0	9	0	0	41	0	0	61	70	
Percentage of students scored more than target		67%			77%			67%			88%			89%	100%	

PRINCIPAL
Sri Indu Institute of Engineering & Tec
Sheriguda(V), Ibrahimpatnam(M),
R.R. Dist. Telangana-501 510.

CO Mapping with Exam Questions:

CO - 1	Y	Y		Y			Y						Y	Y
CO - 2					Y			Y		Y			Y	Y
CO - 3														
CO - 4														
CO - 5														
CO - 6														

CO Attainment based on Exam Questions:

CO - 1	67%			77%									89%	100%
CO - 2										88%			89%	100%
CO - 3														
CO - 4														
CO - 5														
CO - 6														

CO	Subj	obj		Asgn	Overall	Level
CO-1	72%	89%		100%	87%	3.00
CO-2	88%	89%		100%	92%	3.00
CO-3						
CO-4						
CO-5						
CO-6						

Attainment Level	
1	60%
2	70%
3	>80%

Attainment (Internal I Examination) = **3.00**

Faculty Signature

PRINCIPAL
Sri Indu Institute of Engineering
Sheriguda(V), Ibrahimpatnam
R.R. Dist. Telangana -501 5

SRI INDU INSTITUTE OF ENGINEERING AND TECHNOLOGY

Department of Computer Science and Engineering

Course Outcome Attainment (Internal Examination-2)

Name of the faculty : E.RUPA | Academic Year: 2021-22

Branch & Section: CSE-C | Examination: II Internal

Course Name: **Soft Ware Testing Methodologies** | Year: III | Sem: II

S.No	HT No.	Q1a	Q1b	Q1c	Q2a	Q2b	Q2c	Q3a	Q3b	Q3c	Q4a	Q4b	Q4c	Obj2	A2
Max. Marks ==>		5			2	3		5			5				5
1	18X31A05C8														5
2	18X31A05C9														5
3	18X31A05D4														5
4	18X31A05F8														5
5	18X31A05G4														5
6	19X31A05C1				2	2					5			9	5
7	19X31A05C2				2	2					2			8	5
8	19X31A05C3				2	3								9	5
9	19X31A05C4				2	3								8	5
10	19X31A05C5				2	2		2						7	5
11	19X31A05C6				2	2					5			7	5
12	19X31A05C7					2		4						8	5
13	19X31A05C8	3			2	2								8	5
14	19X31A05C9				2	2		2						8	5
15	19X31A05D0				2	2		3						8	5
16	19X31A05D1				2	2								8	5
17	19X31A05D2				2	2		3						8	5
18	19X31A05D3	2			2	2								8	5
19	19X31A05D4					2		2						7	5
20	19X31A05D5					2		2						8	5
21	19X31A05D6	3			2	2								9	5
22	19X31A05D7	3			2	2								8	5
23	19X31A05D8	3			2	2								7	5
24	19X31A05D9				2			1						6	5
25	19X31A05E0				2	2								7	5
26	19X31A05E1														5
27	19X31A05E2	4			2	2								8	5
28	19X31A05E3							3			2			7	5
29	19X31A05E4				2	3					5			9	5
30	19X31A05E5	5			2	3								8	5
31	19X31A05E6				2	2					1			8	5
32	19X31A05E7				2	3					5			9	5
33	19X31A05E8				2	2								8	5
34	19X31A05E9				2	3		5						9	5
35	19X31A05F0							3			3			8	5
36	19X31A05F1				2	2		4						9	5
37	19X31A05F2				1						3			9	5
38	19X31A05F3	4			2	2								8	5
39	19X31A05F4	5				3								8	5
40	19X31A05F5	4			2									8	5
41	19X31A05F6				2			2						8	5
42	19X31A05F7				2	2								7	5
43	19X31A05F8							3			3			6	5
44	19X31A05F9				2	2		1						7	5

PRINCIPAL
Sri Indu Institute of Engineering &
Sheriguda(V), Ibrahimpatnam
R.R. Dist. Telangana 501

45	19X31A05G0				2	3		5						8	5
46	19X31A05G1				2						3			8	5
47	19X31A05G2				2						3			8	5
48	19X31A05G3	4			2	2								8	5
49	19X31A05G4	3			2									7	5
50	19X31A05G5				2	2		2						7	5
51	19X31A05G6				2			1						6	5
52	19X31A05G7				2						3			6	5
53	19X31A05G8							3						6	5
54	19X31A05G9							3			3			6	5
55	19X31A05H0							3			2			7	5
56	19X31A05H1	4			2	3								8	5
57	19X31A05H2	3			2									7	5
58	19X31A05H3				2	2		1						7	5
59	19X31A05H4				2	2		4						7	5
60	19X31A05H5				2	2		4						7	5
61	19X31A05H6				2	2		5						6	5
62	19X31A05H7				2	2								7	5
63	19X31A05H8														5
64	19X31A05H9				2	1								7	5
65	19X31A05I0				2	2		4						7	5
66	20X35A0513														5
67	20X35A0514				2	2								6	5
68	20X35A0515				2						3			7	5
69	20X35A0516				2			4						7	5
70	20X35A0517	2			2	2								8	5
71															
72															
73															
Target set by the faculty / HoD		3.00	0.00	0.00	1.20	1.80	0.00	3.00	0.00	0.00	3.00	0.00	0.00	0.00	3.00
Number of students performed above the target		13	0	0	51	43	0	17	0	0	9	0	0	62	70
Number of students attempted		15	0	0	52	44	0	27	0	0	16	0	0	62	70
Percentage of students scored more than target		87%			98%	98%		63%			56%			100%	100%

CO Mapping with Exam Questions:

CO - 1														
CO - 2														
CO - 3	Y													
CO - 4								Y		Y				Y
CO - 5				Y			Y							Y
CO - 6														Y

% Students Scored >Target %	87%			98%	98%		63%			56%			100%	100%

PRINCIPAL
Sri Indu Institute of Engineering & Tech.
Sheriguda(V), Ibrahimpatnam(M)
R.R. Dist. Telangana -501 510

CO Attainment based on Exam Questions:

CO - 1														
CO - 2														
CO - 3	87%													
CO - 4										56%				100%
CO - 5				98%			63%							100%
CO - 6														100%

CO	Subj	obj		Asgn	Overall	Level
CO-1						
CO-2						
CO-3	87%				87%	3
CO-4	56%			100%	78%	3.00
CO-5	81%			100%	90%	3.00
CO-6				100%	100%	3.00

Attainment (Internal Examination-2) = 3.00

Attainment Level		
1	60%	
2	70%	
3	>80%	

Faculty Signature

Principal
Sri Indu Institute of Engineering
Sheriguda(V), Ibrahimpatnam
R.R. Dist. Telangana-501 510

SRI INDU INSTITUTE OF ENGINEERING AND TECHNOLOGY

Department of Computer Science and Engineering

Course Outcome Attainment (University Examinations)

Name of the faculty : E.RUPA | Academic Year: 2021-22

Branch & Section: CSE-C | Year / Semester: III / II

Course Name: Soft Ware Testing Methodologies

S.No	Roll Number	Marks Secured
1	18X31A05C8	3
2	18X31A05C9	-1
3	18X31A05D4	4
4	18X31A05F8	14
5	18X31A05G4	-1
6	19X31A05C1	36
7	19X31A05C2	35
8	19X31A05C3	29
9	19X31A05C4	31
10	19X31A05C5	42
11	19X31A05C6	34
12	19X31A05C7	8
13	19X31A05C8	29
14	19X31A05C9	4
15	19X31A05D0	26
16	19X31A05D1	10
17	19X31A05D2	27
18	19X31A05D3	6
19	19X31A05D4	26
20	19X31A05D5	14
21	19X31A05D6	34
22	19X31A05D7	21
23	19X31A05D8	0
24	19X31A05D9	2
25	19X31A05E0	27
26	19X31A05E1	-1
27	19X31A05E2	36
28	19X31A05E3	6
29	19X31A05E4	48
30	19X31A05E5	39
31	19X31A05E6	26
32	19X31A05E7	36
33	19X31A05E8	2
34	19X31A05E9	39
35	19X31A05F0	29
Max Marks		75

S.No	Roll Number	Marks Secured
36	19X31A05F1	26
37	19X31A05F2	28
38	19X31A05F3	26
39	19X31A05F4	40
40	19X31A05F5	18
41	19X31A05F6	17
42	19X31A05F7	26
43	19X31A05F8	26
44	19X31A05F9	0
45	19X31A05G0	38
46	19X31A05G1	10
47	19X31A05G2	16
48	19X31A05G3	21
49	19X31A05G4	15
50	19X31A05G5	40
51	19X31A05G6	16
52	19X31A05G7	1
53	19X31A05G8	18
54	19X31A05G9	21
55	19X31A05H0	2
56	19X31A05H1	26
57	19X31A05H2	15
58	19X31A05H3	28
59	19X31A05H4	35
60	19X31A05H5	26
61	19X31A05H6	31
62	19X31A05H7	10
63	19X31A05H8	-1
64	19X31A05H9	0
65	19X31A05I0	20
66	20X35A0513	-1
67	20X35A0514	11
68	20X35A0515	26
69	20X35A0516	20
70	20X35A0517	30

PRINCIPAL
Sri Indu Institute of Engineering & T
Sheriguda(V), Ibrahimpatnam
R.R. Dist. Telangana -501 5

Class Average mark	20
Number of students performed above the target	37
Number of successful students	70
Percentage of students scored more than target	53%
Attainment level	**1**

Attainment Level	% students
1	60%
2	70%
3	>80%

Faculty Signature

SRI INDU INSTITUTE OF ENGINEERING AND TECHNOLOGY

Department of Computer Science and Engineering

Course Outcome Attainment

Name of the faculty : E.RUPA
Branch & Section: CSE-C
Course Name: **Soft Ware Testing Methodologies**

Academic Year: 2021-22
Examination: I Internal
Year: III
Semester: II

Course Outcomes	1st Internal Exam	2nd Internal Exam	Internal Exam	University Exam	Attainment Level
CO1	3.00		3.00	1.00	1.50
CO2	3.00		3.00	1.00	1.50
CO3	0.00	3.00	1.50	1.00	1.13
CO4		3.00	3.00	1.00	1.50
CO5		3.00	3.00	1.00	1.50
CO6		3.00	3.00	1.00	1.50
Internal & University Attainment:			2.75	1.00	
Weightage			25%	75%	
CO Attainment for the course (Internal, University)			0.69	0.75	
CO Attainment for the course (Direct Method)			1.44		

Overall course attainment level **1.44**

Faculty Signature

PRINCIPAL
Sri Indu Institute of Engineering & T.
Sheriguda(V), Ibrahimpatnam(M),
R.R. Dist. Telangana -501 510

SRI INDU INSTITUTE OF ENGINEERING & TECHNOLOGY

Department of Computer Science and Engineering

Program Outcome Attainment (from Course)

Name of Faculty:	E.RUPA	Academic Year:	2021-22
Branch & Section:	CSE-C	Year:	III
Course Name:	Soft Ware Testing Methodologies	Semester:	I

CO-PO mapping

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO1	3	1	2	2	2				1			
CO2	1	2	2	1	2				1			
CO3	1	2	1	2								
CO4	2	1	1	1								
CO5	1	2	1	1					1			
CO6	3	2	1	2					2			
Course	**1.83**	**1.67**	**1.33**	**1.50**	**2.00**	**0.00**	**0.00**	**0.00**	**1.25**	**0.00**	**0.00**	**0.00**

CO	Course Outcome Attainment
CO1	1.50
CO2	1.50
CO3	1.13
CO4	1.50
CO5	1.50
CO6	1.50
Overall course attainment level	**1.44**

PO-ATTAINMENT

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO Attainment	0.88	0.80	0.64	0.72	0.96	0.00	0.00	0.00	0.60	0.00	0.00	0.00

CO contribution to PO - 33%, 67%, 100% (Level 1/2/3)

Faculty Signature

Sri Indu Institute of ...
Sheriguda(V), Ibrahimpa...
R.R Dist. Telangana -501 5...

UNIT - 2

1. The Purpose of Testing:-

Testing:- It is the process of executing a program (or) system with the intent of finding errors.

* Testing is basically a task of locating errors.

> Software Testing = Software Verification + Software Validation.

Software testing is an activity that is used to evaluate the capability of a system and determine if it is in accordance with the specifications. The purpose of testing is more than just debugging and detecting bugs. testing is usually performed for the following ~~steps~~ purposes. They are as follows,

1) For improving and assuring software quality.
2) For verification and validation.
3) For estimating reliability.

2) Software Quality Assurance (SQA):-

Quality is used to determine if the software meets all the requirements that were specified during design phase.

Software quality assurance is used to monitor the software engineering processes and methods for ensuring its quality.

2) Verification and Validation:-

Once the product is developed the software has to be tested to ensure that it meets the requirements of the customer and also the internal requirements. Validation is done to check whether the software conforms to all the specifications.

Both Verification and Validation is done to test the quality of a software. Based on the test results, testers make decisions whether the product will work properly or not.

Software quality has the following three aspects such as functionality, engineering and adaptability.

1) Functionality:- It is an exterior quality factor. It is determined by considering correctness, reliability, usability and integrity of a product.

ii) Engineering:- It is an interior quality factor. It is determined by considering efficiency, testability documentation and structure of a product.

iii) Adaptability:- It is a future quality factor. It is determined by considering flexibility, reusability and maintainability of a product. When a validation test is performed on a product, it is referred to either as a clean or positive tests.

The disadvantage of validation is that the software works only for some particular test cases. The finite number of test cases cannot ensure that the software works properly under all circumstances. On the other hand, only one failed test shows that the software doesn't work. Negative tests are performed to show that the software fails to meet the expected requirements of the user.

3) Reliability Estimation:-

Testing is an important factor to quantify software reliability. Software reliability is related to the different aspects of software such as structure, documentation etc. It is difficult to estimate software reliability only by considering its related aspects.

1.1.1 Goals for testing:-

The two major goals of testing are as follows,

1) Bug Prevention

2) Bug Discovery

1) Bug Prevention:- Which is considered as the primary goal for testing. When a bug is detected, appropriate method should be used to remove it. And if the bugs are not prevented, then certain ~~Sympoty~~ Symptoms are discovered that are the major causes for the occurrence of bugs. When a perticular bug is prevented there's no need to perform testing again to confirm

accuracy of the program. Also a prevented bug won't affect the execution schedule of the program.

2) Bug Discovery:- Which is considered as the secondary goal for testing. It is performed when the primary goal fails to prevent the bugs. Since bugs are not static, they always change their states. Even if all the information regarding the expectations, procedures and output of test are maintained, there is a probability of errors to occur.

1.1.2 phases in a Tester's Mental Life:-

A tester considers five phases of thinking. These phases are classified as follows,

1) phase 0 thinking
2) phase 1 thinking
3) phase 2 thinking
4) phase 3 thinking
5) phase 4 thinking.

1) Phase 0 (Thinking Criteria):-

Testing and debugging are similar. Testing is useful in debugging i.e it supports debugging. When testing came into existence, phase 0 thinking was the criteria for the software development. This thinking was applicable to an area from which the following resources can be determined which are as follows,

i) High-cost and inadequate computing resources

ii) Low-cost software resources

iii) Single Programmers.

iv) Small Projects resources.

v) Irrelevant software resources.

2) phase 1 (Thinking the software or program works):-

the aim of this testing is to display the working of a Program or software. Phase 1 thinking identifies the distinct relationship between testing and debugging. In this testing phase the number of tests performed on a software does not determine the execution of the software.

If a single test among the infinite number of available tests fails, then the tester concludes that the software will not be executed even if the subsequent tests make its execution possible.

3) Phase 2 (Thinking the program doesn't work):-

The aim of this type of thinking is to demonstrate that the program doesn't execute. The thinking of testers in phase 2 opposes the thinking of designers. If a test fails, then the goal of phase 2 thinking is accomplished. In the process of phase 2 thinking, bug can be detected by testers, programmers and designers.

i) The tester detects an error (bug).

ii) The programmer verifies and corrects the error.

iii) The designer designs test.

iv) The designer aims to perform different tests to identify different errors.

4) phase 3 (Thinking Test for Reducing the Risk):-

the aim of testing to reduce the risks that are Predictable. The phase 3 thinking accepts the rules of statistical quality control to improve the quality of software. In the process of phase 3 thinking, the tester detects the bugs and eliminate them.

5) phase 4 (Thinking Knowledge):-

The aim of phase 4 thinking is to reduce the risks of software with minimal testing effort. Software that require minimum testing in order to attain goals of lower phases testing is obtained by integrating the capability of testing with the knowledge of knowing the conditions under which software is made testable.

Testing is done for the following reasons,

i) To decrease the manpower required for testing.

ii) To ensure that a testable code has less number of bugs when compared to the code that is difficult to test.

1.1.3 Testing isn't Everything:

Testing is an effective method for detecting errors. Alternatively, other methods are taken to detect errors and remove them. The other methods include reviews, inspections, walk through etc. After processing these methods the software is tested. The methods are as follows:

1) walkthroughs, Inspection and Review Methods:

Inspections, walkthroughs and review methods are performe to detect errors. These methods are capable of determini only some errors. The drawback of these methods is that they cannot detect all the errors present in the softwar

2) Program Design Method:- the design model of a program determines whether the quality of the program is good or not. An improper design model degrades the quality of the software program.

3) Syntax checking method:- During the analysis of source code, the syntax errors are checked by the compile The static analysis methods such as strong typing and type checking are considered

Thuse two methods detect all the errors in the Program and correct them.

4) Software Development methods :-

A Software development method is an internal process which uses various methods for developing a Software. for example, a statistical control method, Configuration control method and automatic distribution of informatic method are used to develop software. Thuse methods detect and remove most of the errors in the Softwa and the programmer is unaware of the changes that take place after removing those errors.

1.2 Dichotomies:

1.2.1 Testing versus Debugging:

Testing	Debugging
1) The goal of testing is to detect errors in a program.	1) The goal of debugging is to detect errors and correct them.
2) Testing is initiated with known conditions.	2) Debugging is initiate with unknown conditions.
3) The output can be anticipated	3) The output cannot be anticipated.
4) It is necessary to have planned, designed and scheduled constraints.	4) While in debugging, it is not necessary to have this constraints.
5) Testing finds out the reason for program's failure.	5) Debugging is the programmer's justification.
6) It is not necessary to have design knowledge while performing testing.	6) It is sufficient to have detailed design knowledge for debugging.
7) The test design and execution are done automatically.	7) Designing and execution can't be done automatically in debugging.

1.2.2 function versus structure Testing:-

Structural Testing	Functional Testing
1) Structural testing is also known as white box, glass-box testing.	1) Functional testing is also known as black box or closed box testing.
2) Structural tests are performed based on the knowledge of internal structure of the source code.	2) Functional tests are performed without the knowledge of internal structure of the software.
3) The test cases take finite time but cannot detect all bugs.	3) The test cases take infinite time and detect all errors.
4) Structural testing is less effective when compared to functional testing.	4) Functional testing is more effective than glass-box testing.
5) Different methods for performing white box testing are as follows, i) Statement testing ii) Decision testing iii) Condition testing	5) Different methods for performing black box testing are as follows, i) Expected inputs method ii) Boundary value method. iii) Illegal values method.

1.2.3 The Designer Versus the Tester:-

Designer	Tester
1) Designer is based on the structural specification of the system.	1) Tester is based on function specification of the system.
2) He/She depends on the implementation details.	2) He/She is independent of the implementation details
3) A software designer is responsible for designing and executing the tests.	3) A tester is responsible for designing and executing the tests.

1.2.4 Small versus Large:-

Small	Large
1) Small programs have only few lines of code.	1) Large programs have more no of lines of code.
2) They consist of few components.	2) They consist of large number of components.
3) Small programs does not require any technique for testing.	3) They require different types of techniques for testing.
4) Small programs are more efficient.	4) Large programs are less efficient.
5) Small programs are written by single programmer.	5) Large programs are written by different programmers.

1.3 Model for Testing :-

1.3.1 A model Project for Testing Process :-

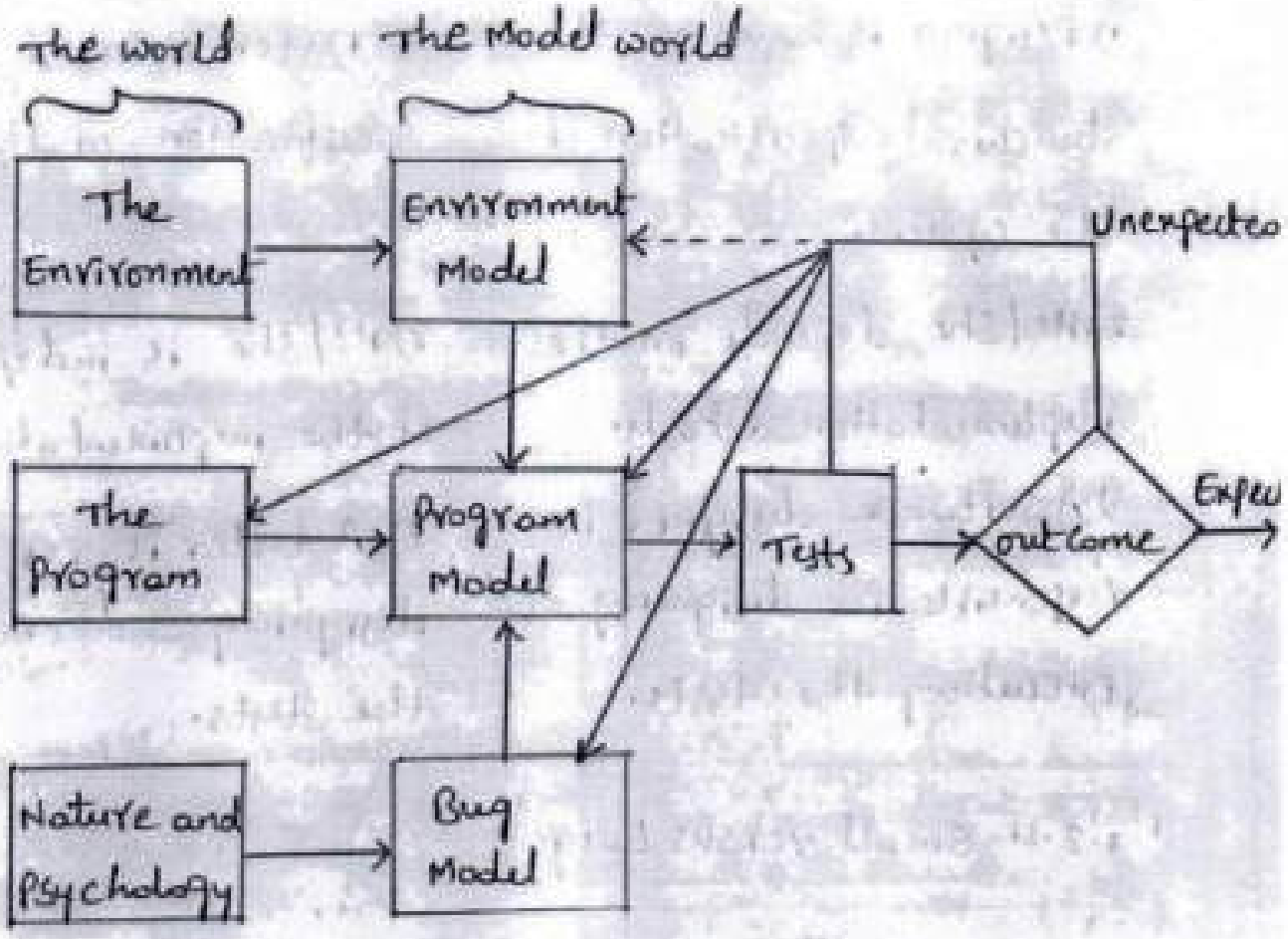


fig: A Model of Testing

Above figure is a model of the testing process. The process starts with a program embedded in an environment, such as a computer, an operating system, or a calling program. We understand human nature and its susceptibility to error. This understanding leads us to create three models, a model of the environment,

a model of the program, and model of the expected bugs. From these models we create a set of tests, which are then executed. The result of each test is either expected or unexpected. If unexpected, it may lead us to revise the test, our model or concept of how the program behaves, our concept of what bugs are possible or the program itself.

The program's environment is the hardware and software required to make it run. For online systems the environment may include communications lines, other systems, terminals and operators. The environment also includes all programs that interact with and are used to create the program under test, such as operating system, loader, linkage editor, compiler, utility routines.

Programmers should learn early in their careers that it's not smart to blame the environment (i.e hardware and firmware) for bugs.

Hardware bugs are rare. So are bugs in manufacture supplied software.

If testing reveals an unexpected results, we may have to change our beliefs (our model of the environment) to find out what went wrong. But sometimes the environment could be wrong, the bug could be in the hardware or firmware after all

~~139~~

1.3.2 Software Testing Strategy:

The overall strategy for software testing can be diagramatically represented in below figure using the Spiral model.

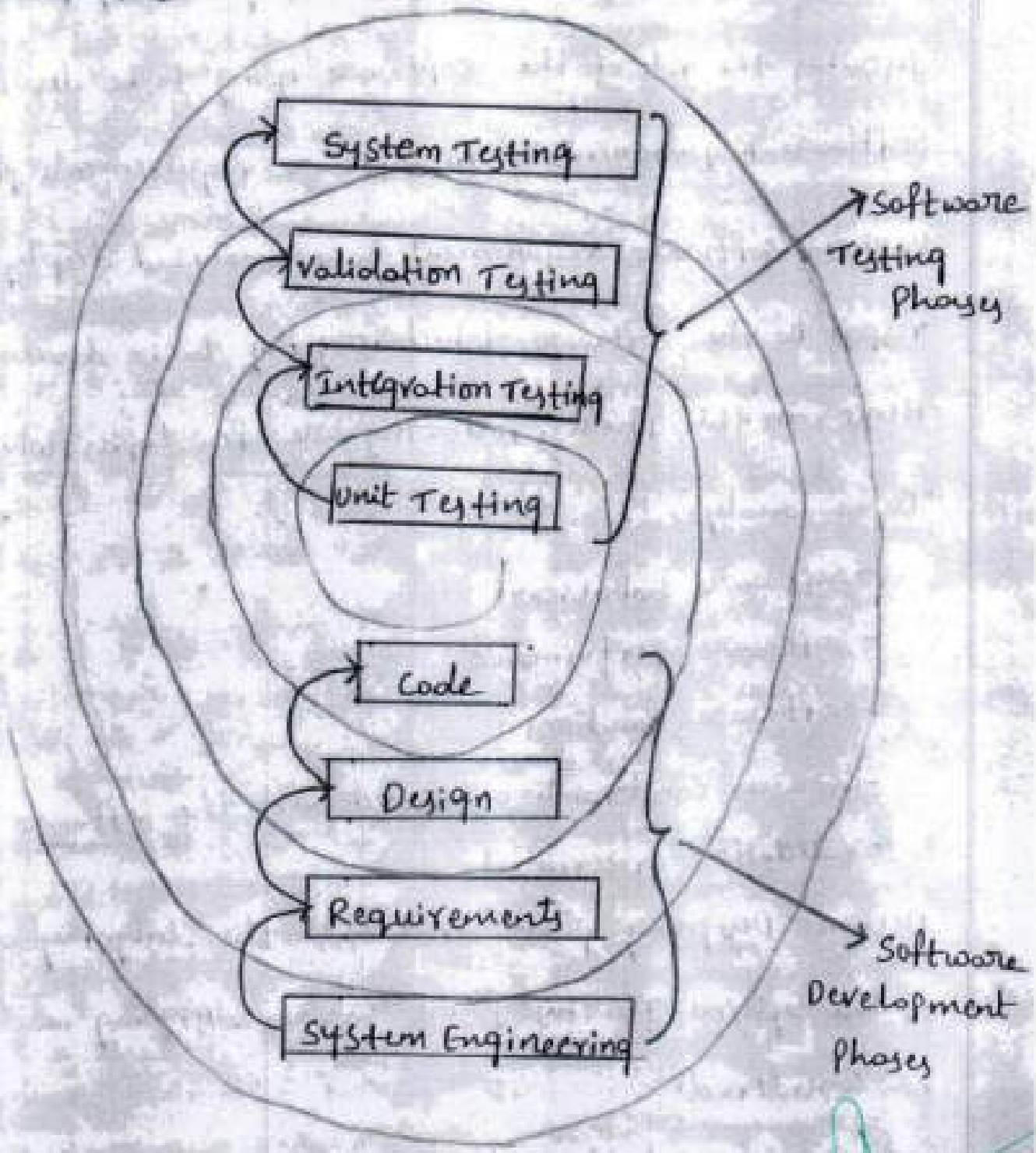


fig: Testing strategy using Spiral model.

The spiral model depicting the Software development process is diagramatically represented in above figure. the details of each entity in this model is as follows

System Engineering :- The system engineering phase describes the role of the software going to be developed.

Software Requirements :- In software requirements phase, various software requirements are analyzed deeply with respect to the software which is going to be developed. Hence, in this phase, following are few topics which are to be analyzed.

- Software behaviour
- Software performance
- Software functions
- Various constraints associated with the software
- Validation criteria etc

Design :- Designing phase deals with the designing aspects of the system. In this phase various designing models can be considered.

Coding :- As usual the coding phase deals with writing of code for the software.

The four phases described above come under the Software development phases. Now Consider the Software testing phases which are associated with each phase of the Software development phases.

i) The Unit Testing phase:- As the name suggests in this phase each unit or module of the code is tested.

ii) Integration Testing:- Integration testing concentrates on on testing the software design.

iii) Validation Testing:- In validation Testing, The code is tested to verify that the software satisfies all the requirements which were laid during the requirements phase of software development.

System Testing:- In system testing the entire System is tested on a whole.

Each of these four phases are sequentially adopted in order to ensure that the software which is delived to the given user satisfies all the functional, behavioural, performance, as well as the System requirements.

1.3.3 Testing and Levels:-

There are four different types of testing that can be performed on a software system. They are as follow

1) Unit Testing:-
2) Component Testing
3) Integration Testing
4) System Testing

1) Unit Testing:- A unit is the smallest piece of source code that can be tested. It is also known as a module which consists of several lines of code that are processed by a single programmer. The main purpose of performing unit testing to reveal that a particular unit doesn't fulfil the specified functional requirement and also to show that the structural implementations is not similar to the expected structured designed.

Unit tests can be both static tests and dynamic tests. At first, static tests are performed followed by the dynamic tests to check the tests paths, boundaries and branches. most of the unit tests are dynamic in

Logic Based Testing

1) Overview about Logic Based Testing:-

"Logic" is one of the most often used words in programm Vocabulary but one of their least used techniques.

This chapter concerns Logic in its simplest form, boolean algebra and its applications to program and specification test and design. Boolean algebra is to logic, as arithmeti is to mathematics.

2) Decision Tables:-

A decision table is a tabular form that consists of a set of conditions and their respective actions. A decision table consists of four parts. They are as follows,

i) Condition stub
ii) Condition entry
iii) Action stub
iv) Action entry.

Condition Stub:- It consists of a list of causes. for a rule to be satisfied, conditions are marked with Yes/No. "Yes" indicates that the condition is satisfied and "No" specifies that the condition is not satisfied. The variable 'I' specifies the immaterial test case, which means that a particular condition doesn't have any role in that rule.

Action Stub:- It describes the possible actions that are to be executed and "No" specifies that an action need not be executed.

Table1: An Example of a Decision Table

		RULE1	RULE2	RULE 3	RULE 4
CONDITION STUB	CONDITION1	YES	YES	NO	NO
	CONDITION2	YES	I	NO	I
	CONDITION3	NO	YES	NO	I
	CONDITION 4	NO	YES	NO	YES
ACTION STUB	ACTION 1	YES	YES	NO	NO
	ACTION 2	NO	NO	YES	NO
	ACTION 3	NO	NO	NO	YES

ACTION ENTRY

1.a) Action 1 will take place if conditions 1 and 2 are met and if conditions 3 and 4 are not met (rule 1), or if conditions 1, 3 and 4 are met (rule 2).

1.b) Action 1 will be taken if predicates 1 and 2 are true and if predicates 3 and 4 are false (rule 1), or if predicates 1, 3 and 4 are true (rule 2).

2. Action 2 will be taken if the predicates are all false (rule 3).

3. Action 3 will take place if predicate 1 is false and predicate 4 is true (rule 4).

	RULE 5	RULE 6	RULE 7	RULE 8
CONDITION1	I	NO	YES	YES
CONDITION2	I	YES	I	NO
CONDITION3	YES	I	NO	NO
CONDITION4	NO	NO	YES	I
DEFAULT ACTION	YES	YES	YES	YES

Table: the Default Rules for Table 1.

Applications of Decision Tables:-

1) Decision tables are used in business data processing.
2) Decision table processors are generally used for generating a high-level code. The translator in the processor is used to examine the decision table for consistency and completeness. These processors are implemented using boolean algebra.
3) Decision tables are used in many application areas like business analysis, programming, testing, design of hardware

Decision-Table Processors:-

Decision tables represent higher-level language, as they can be translated into program code automatically. The translator of decision table is responsible for checking whether the source decision table is reliable and complete. It also checks whether any default rules are required and specifies them if needed.

The advantages of using decision table as source language are,

i) It provides integrity of clarity.

ii) It provides explicit relation to specification.

iii) It provides high-level of maintainability.

The drawback of using decision table as source language is that it provides inefficient program logic.

Decision Table Based Testing :-

Decision tables are used for designing test-case when specification are specified as decision table. On the other hand, decision tables are used as a basis for designing test-case when program's logic is implemented as decision table. In this situation the reliability and completeness of decision table are examined using decision table processors. The processor automatically converts the decision table into specialized language

where a translator is responsible for checking reliability and specifying the default rules if required

The following are the situations where decision tables are restricted for designing test-case...

1) When specifications are specified or converted as decision table.

2) When the particular sequence in which conditions are estimated has no influence on the resulting action or definition.

3) When the particular sequence in which rules are estimat has no influence on resulting action.

4) If a test-case is true and leads to the execution of many actions, ~~is executed~~ the sequence in which actions are performed has no importance.

5) When a specific test case is true and its respective action is executed, it is not necessary to check the remaining test cases.

Expansion of Immaterial Cases:-

Immaterial entry (I) cause most decision-table contradictio

If a condition's truth value is immaterial in a ~~value~~ rule, satisfying the rule does not depend on the condition It doesn't mean that the case is impossible. for example

Rule 1: "If the persons are male and over 30, then they shall receive a 15% "raise".

Rule 2: "But if the persons are female, then they shall receive a 10% raise"

The above rules state that age is material for a male's raise, but immaterial for deter-mining a female's raise. No one would suggest that females either under or over 30 are impossible. If there are n predicates there are 2^n cases to consider. find the cases by expanding the immaterial cases.

This is done by converting each 'I' entry into a pair of

entries, one with a 'YES' and the other with a 'NO'. Each 'I' entry in a rule doubles the number of cases in the expansion of that rule. Rule 2 in Table 1 contains one 'I' entry and therefore expands into two equivalent subrules. Rule 4 contains two 'I' entries and therefore expands into four subrules. The expansion of rules 2 and 4 are shown in below table:

	RULE 2		RULE 4			
	Rule 2.1	Rule 2.2	Rule 4.1	Rule 4.2	Rule 4.3	Rule 4.4
CONDITION 1	YES	YES	NO	NO	NO	NO
CONDITION 2	YES	NO	YES	YES	NO	NO
CONDITION 3	YES	YES	YES	NO	NO	YES
CONDITION 4	YES	YES	YES	YES	YES	YES

Table: Expansion of Immaterial cases for Rules 2 and 4

	RULE 1	RULE 2
CONDITION1	YES	YES
CONDITION2	I	NO
CONDITION3	YES	I
CONDITION4	NO	NO
ACTION 1	YES	NO
ACTION 2	NO	YES

RULE 1·1	RULE 1·2	RULE 2·1	RULE 2·2
YES	YES	YES	YES
YES	NO	NO	NO
YES	YES	YES	NO
NO	NO	NO	NO
YES	YES	NO	NO
NO	NO	YES	YES

Table: The Expansion of an Inconsistent Specification

Decision Tables and Structure:-

Decision tables can also be used to examine a program's structure. Below figure shows a program segment that consists of a decision tree. These decisions, in various combinations, can lead to actions 1, 2, or 3.

If the decision appears on a path, put in 'YES' or 'NO' as appropriate. If the decision does not appear on the path, put in an 'I'.

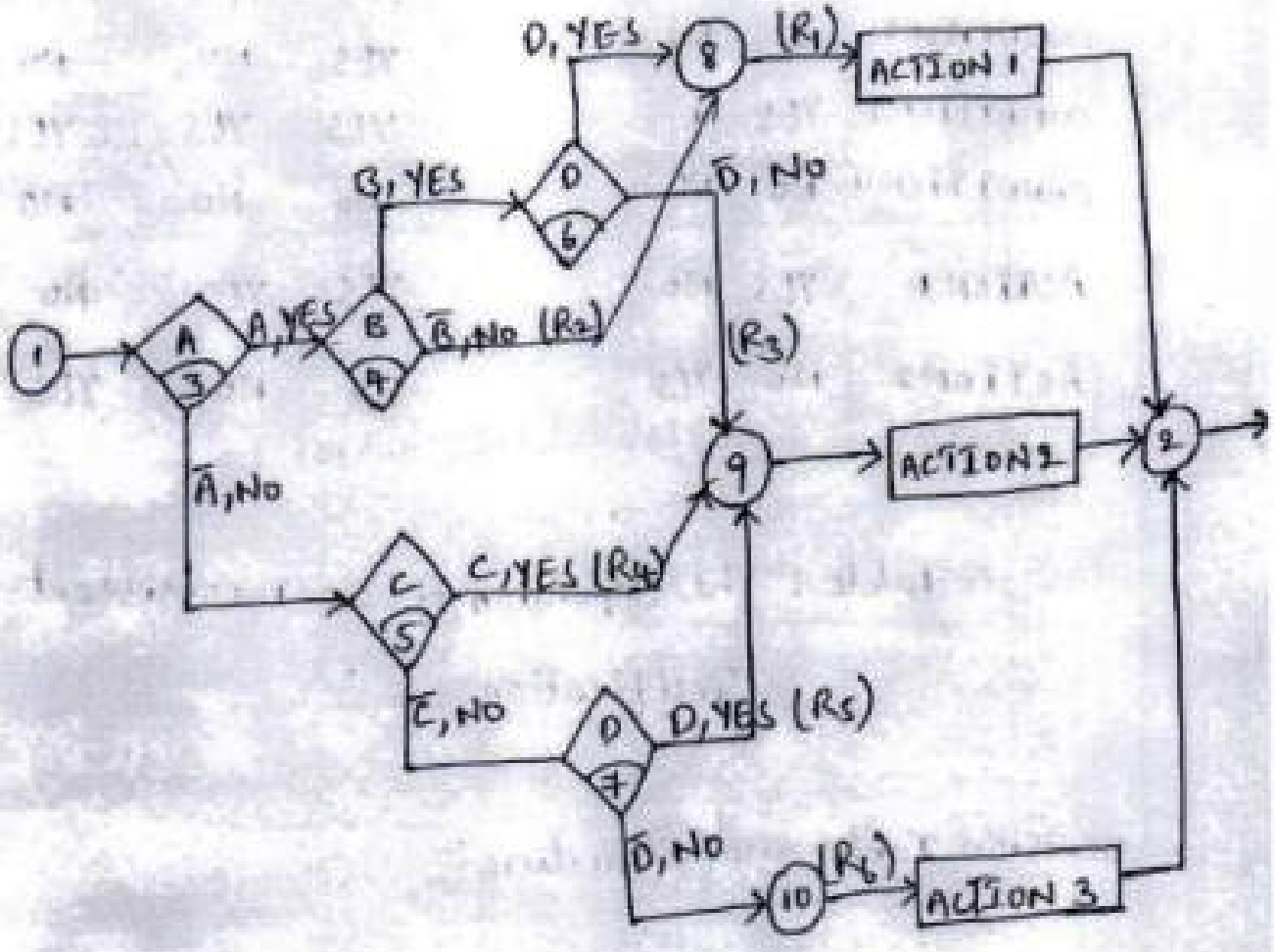


figure: A Sample Program.

	RULE 1	RULE 2	RULE 3	RULE 4	RULE 5	RULE 6
CONDITION A	YES	YES	YES	NO	NO	NO
CONDITION B	YES	NO	YES	I	I	I
CONDITION C	I	I	I	YES	NO	NO
CONDITION D	YES	I	NO	I	YES	NO
ACTION 1	YES	YES	NO	NO	NO	NO
ACTION 2	NO	NO	YES	YES	YES	NO
ACTION 3	NO	NO	NO	NO	NO	YES

	R1	RULE 2	R3	RULE 4	R5	R6
CONDITION A	Y Y	Y Y Y Y	Y Y	N N N N	N N	N N
CONDITION B	Y Y	N N N N	Y Y	Y Y N N	N Y	Y N
CONDITION C	Y N	N N Y Y	Y N	Y Y Y Y	N N	N N
CONDITION D	Y Y	Y N N Y	N N	N Y Y N	Y Y	N N

Table : The Expansion of Table 2.

3) Path Expressions:-

x

i) Predicates and Binary truth values:-

x x

Predicate is defined as a process which gives truth values as its output. Predicates are not confined to only arithmetic relations but also depend on relational operators such as $(>, >=, <, <=, =, \neq)$ is a subset o is a subgraph of, is above, is below.

Predicates is not only implemented using the basic arithmetic relation but also an equality predicate

this equality predicate is generally used in looping structure that is responsible for scanning the entire set.

ii) Case statements and multivalued logics:-

Predicates are not restricted to binary truth values which are True/false. predicates can be either be multiway predicates or multivalued logic.

There are many situations where something can go wrong with predicate.

a) The wrong relational operator is used e.g. > instead of <=

b) The predicate expression of a compound predicate is incorrect eg: A+B instead AB.

c) The wrong operands are used: eg: A>x instead of A>.

d) The processing leading to the predicate (along the predicate interpretation path) is faulty.

Boolean Algebra Rules:-

The different operators that are used while solving any boolean algebra are,

i) Multiplication (x): This Symbol Specifies the AND operation. A statement 'PxQ' is said to be true if both statements P and Q are true, otherwise it is false.

PxQ Truth Table

P	Q	PxQ
T	T	T
T	f	f
f	T	f
f	f	f

ii) Addition (+): This Symbol specifies the "OR" operation

A statement 'P+Q' is said to be true if either P is true or both are true.

P+q Truth Table

p	q	p+q
T	T	T
T	f	T
f	T	T
f	f	f

Negation (-): It specifies the complement operation. It is referred as "NOT" operator. It is represented as $\bar{p}$ which means that statement is true if the statement p is false.

$\bar{p}$ Truth Table

p	$\bar{p}$
T	f
f	T

Laws of Boolean Algebra:-

Law: (Idempotency law): $p + p = p$ [If both statements are true then the resultant statement is true].

$\bar{P} + \bar{P} = \bar{P}$ [If both statements are false, then the resultant statement is false].

Law 2 (Null Law): $P + 1 = 1$. Here '1' is universal true. According to '+' operation, if one statement is true then resultant is always true.

Law 3: $P + 0 = P$. Here '0' is a null value.

Law 4 (Commutative Law): $P + Q = Q + P$

Law 5: $A + \bar{A} = 1$. The resultant statement is universal true, if either A (or) its negation is true.

Law 6 (Idempotency Law): $P \times P = P$, $\bar{P} \times \bar{P} = \bar{P}$

Law 7: $P \times 1 = P$ [If a true statement is multiplied with universal true statement, then the resultant statement is also true].

Law 8 (Null Law): $P \times 0 = 0$ [If a true statement is multiplied using null value, then the resultant statement is also a null value].

Law 9 (commutative law of multiplication):

$P \times q = q \times p$

Law 10: $P \times \bar{P} = 0$. It is not possible for a single statement to be true and false simultaneously.

Law 11: $\bar{\bar{P}} = P$. This law represents that the negation of negation statement is the statement itself.

Law 12 (Involution): $\bar{0} = 1$

Law 13: $\bar{1} = 0$

Law 14 (Demorgan's law): $\overline{P+q} = \bar{P}\bar{q}$, $\overline{Pq} = \bar{P} + \bar{q}$

Law 15 (distributive law): $P(q+r) = Pq + Pr$

Law 16 (Associative Law of multiplication):

$(P \times q)r = P(q \times r)$

Law 17 (Associative Law Addition):

$(P+q)+r = P+(q+r)$

Law 18 (Absorptive Law): $P + \bar{P}q = P + q$

Law 19: $P + Pq = P$

4) KV charts (Karnaugh Veitch)

Simple Forms:- All the boolean functions of a single variable and their equivalent representation as a KV chart. A '1' means the variables value is '1' or TRUE. A '0' means that the variable value is '0' or FALSE.

KV chats for functions of a Single Variable

A

0	1
0	0

The function is never 'true'

A

0	1
0	1

The function is true when A is true

$\bar{A}$

0	1
1	0

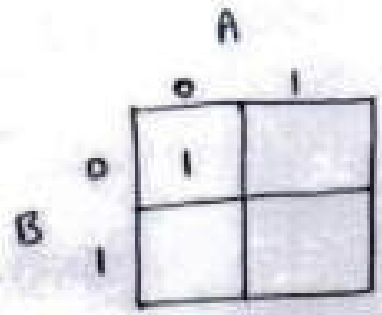
The function is true when A is false

A

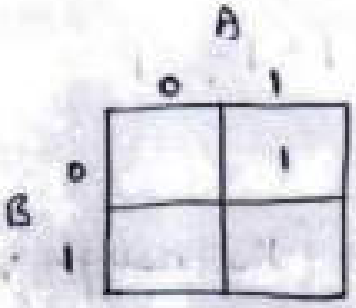
0	1
1	1

The function is always true

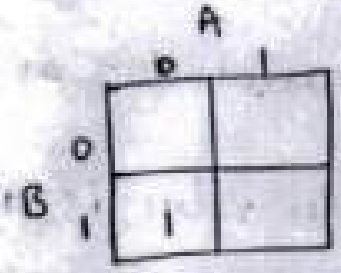
Functions of two variables



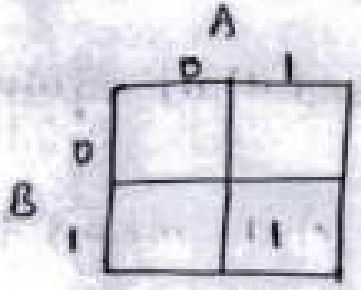
$\bar{A}B$ - NAND



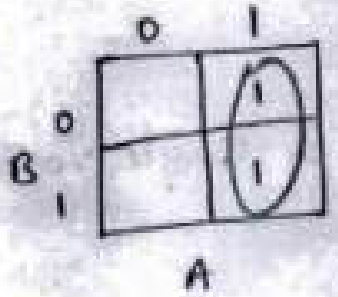
$A\bar{B}$ - A and not B

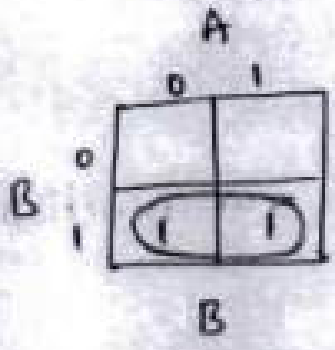


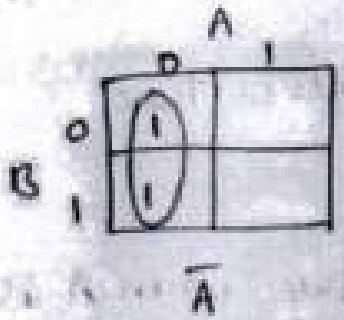
$\bar{A}B$ - B and not A

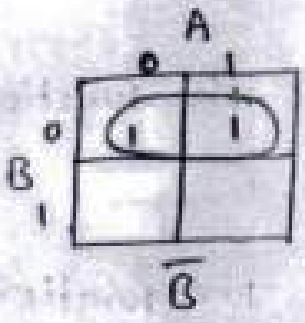


AB - A and B

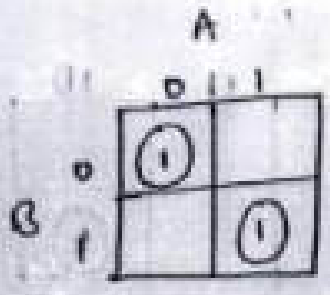




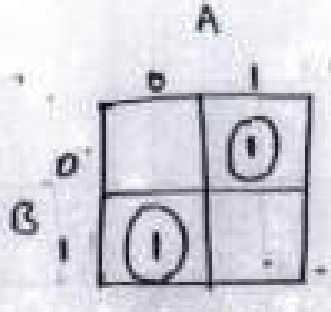




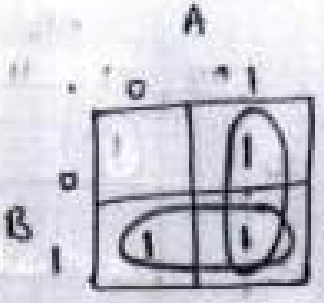
More functions of two variables



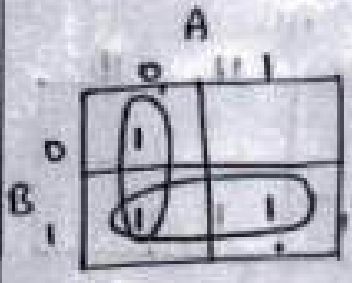
$\bar{A}\bar{B} + AB$



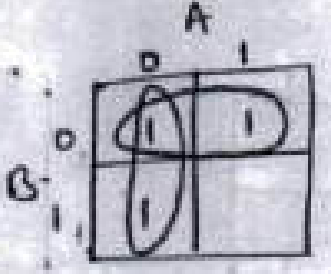
$A\bar{B} + \bar{A}B$



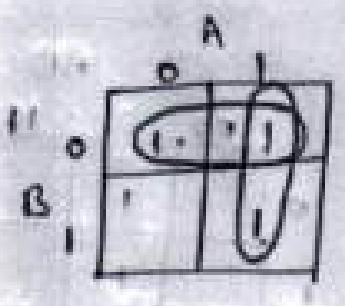
$A + B$



$\bar{A} + B$

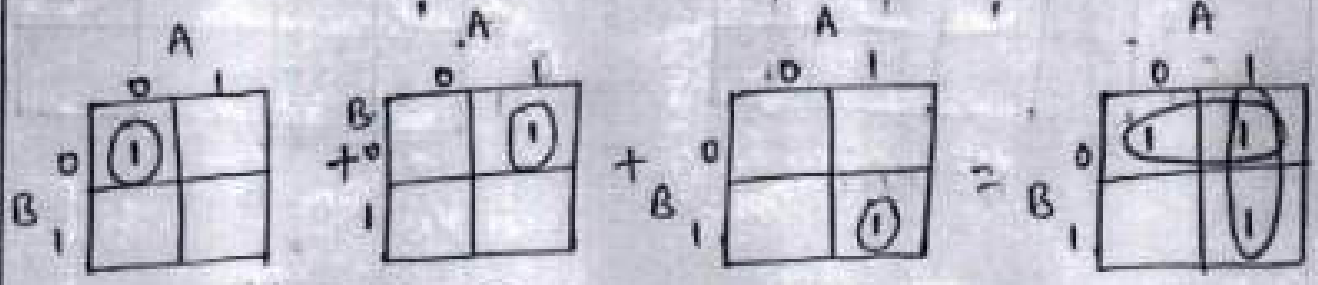


$\bar{A} + \bar{B}$



$\bar{B} + A$

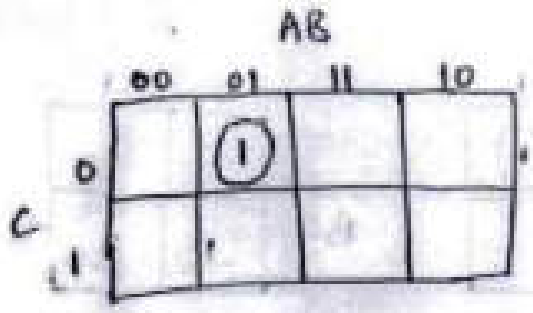
Ex:- $\bar{A}\bar{B} + A\bar{B} + AB = \bar{B} + A$



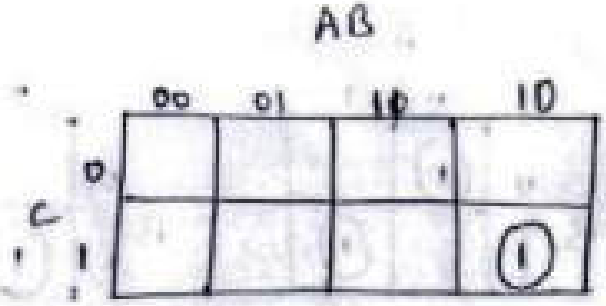
$\bar{A}\bar{B}$ $A\bar{B}$ AB $\bar{B} + A$

PRINCIPAL
Sri Indu Institute of Engineering &
Sheriguda(V), Ibrahimpatnam(M),
R.R. Dist. Telangana

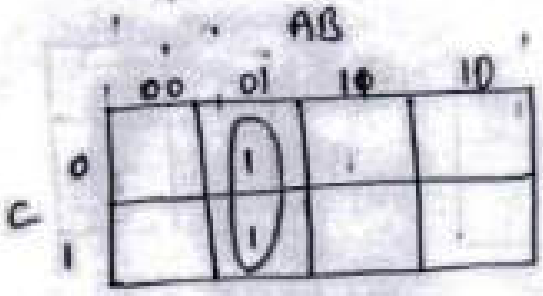
Three Variables:-



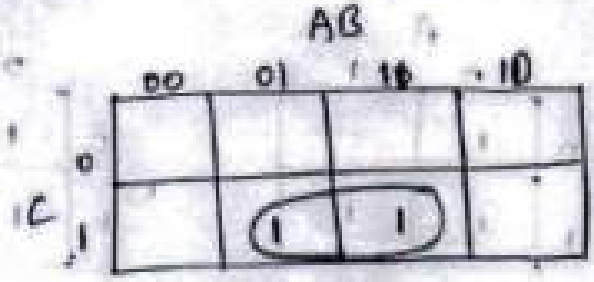
$\bar{A}B\bar{C}$



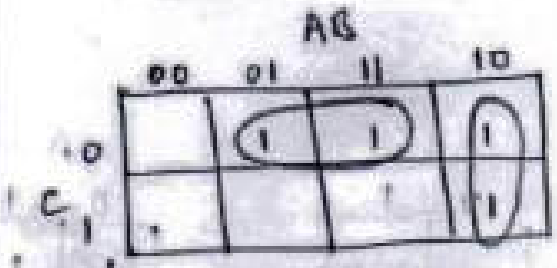
$A\bar{B}C$



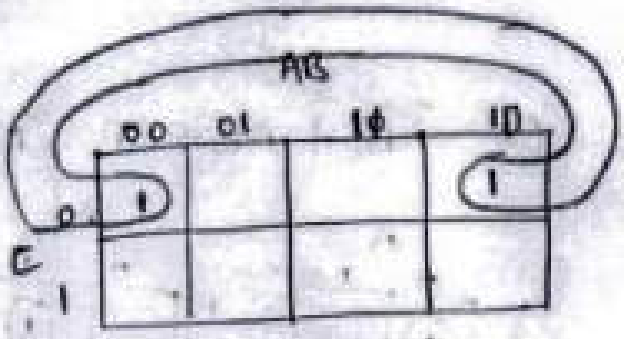
$\bar{A}B$



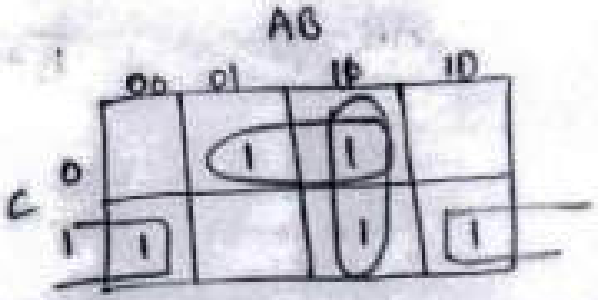
BC



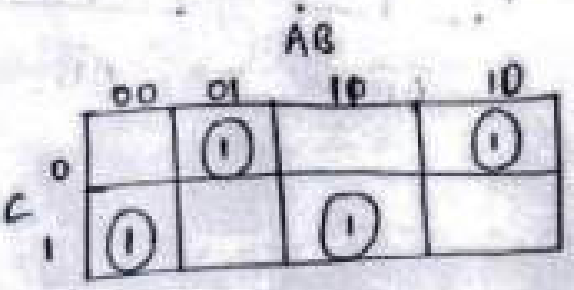
$B\bar{C} + A\bar{B}$



$\bar{B}\bar{C}$



$B\bar{C} + AB + \bar{B}C$



$\bar{A}\bar{B}C + \bar{A}B\bar{C} + ABC + A\bar{B}\bar{C}$

UNIT V
GRAPH MATRICES AND APPLICATIONS

Problem with Pictorial Graphs

- Graphs were introduced as an abstraction of softwarestructure.
- Whenever a graph is used as a model, sooner or later we trace paths through it- to find a set of covering paths, a set of values that will sensitize paths, the logic function that controls the flow, the processing time of the routine, the equations that define the domain, or whether a state is reachable ornot.
- Path is not easy, and it's subject to error. You can miss a link here and there or cover some links twice.
- One solution to this problem is to represent the graph as a matrix and to use matrix operations equivalent to path tracing. These methods are more methodical and mechanical and don't depend on your ability to see a path they are morereliable.

Tool Building

- If you build test tools or want to know how they work, sooner or later you will be implementing or investigating analysis routines based on thesemethods.
- It is hard to build algorithms over visual graphs so the properties or graph matrices are fundamental to toolbuilding.

The Basic Algorithms

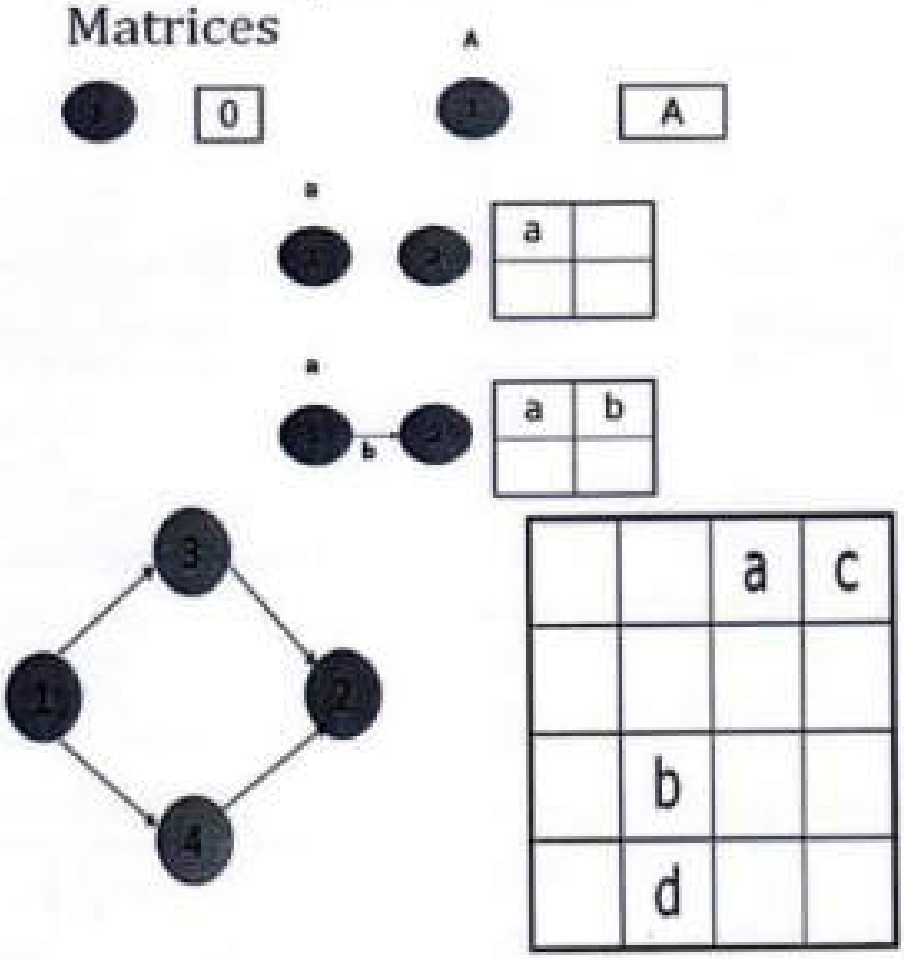
- The basic tool kit consistsof:
- Matrix multiplication, which is used to get the path expression from every node to every othernode.
- A partitioning algorithm for converting graphs with loops into loop free graphs or equivalence classes.
- A collapsing process which gets the path expression from any node to any othernode.

The Matrix of a Graph

- A graph matrix is a square array with one row and one column for every node in thegraph.
- Each row-column combination corresponds to a relation between the node corresponding to the row and the node corresponding to thecolumn.
- The relation for example, could be as simple as the link name, if there is a link between the nodes.
- Some of the things to beobserved:
- The size of the matrix equals the number of nodes.
- There is a place to put every possible direct connection or link between any and any othernode.
- The entry at a row and column intersection is the link weight of the link that connects the two nodes in thatdirection.
- A connection from node i to j does not imply a connection from node j to nodei.
- If there are several links between two nodes, then the entry is a sum; the "+" sign denotes parallel links asusual.

PRINCIPAL
Sri Indu Institute of Engineering &
Shenguda(V), Ibrahimpatnam,
R.R. Dist. Telangana -501 510

Some Graphs and their Matrices



A simple weight

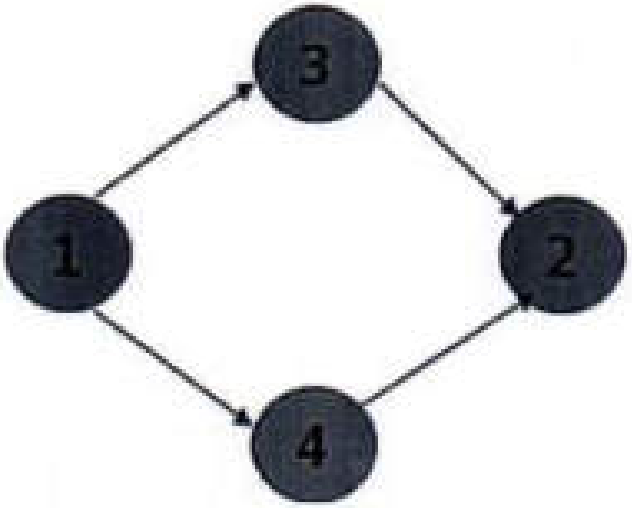
- A simplest weight we can use is to note that there is or isn't a connection. Let "1" mean that there is a connection and "0" mean that thereisn't.
- The arithmetic rulesare:
- 1+1=1 1*1=1
- 1+0=1 1*0=0
- 0+0=0 0*0=0
- A matrix defined like this is called connectionmatrix.

Connection matrix

- The connection matrix is obtained by replacing each entry with 1 if there is a link and 0 if there isn't.
- As usual we don't write down 0 entries to reduce theclutter.

		a	c
	b		
	d		

		1	1
	1		
	1		

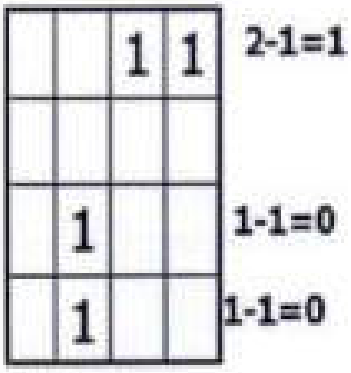


Connection Matrix-continued

- Each row of a matrix denotes the out links of the node corresponding to thatrow.
- Each column denotes the in links corresponding to thatnode.
- A branch is a node with more than one nonzero entry in itsrow.
- A junction is node with more than one nonzero entry in its column.
- A self loop is an entry along thediagonal.

Cyclomatic Complexity

- The cyclomatic complexity obtained by subtracting 1 from the total number of entries in each row and ignoring rows with no entries, we obtain the equivalent number of decisions for each row. Adding these values and then adding 1 to the sum yields the graph's cyclomaticcomplexity.



1+1=2 (cyclomatic complexity)

Relations

- A relation is a property that exists between two objects of interest.
- Forexample,
- "Node a is connected to node b" or aRb where "R" means "is connectedto".
- "a>=b" or aRb where "R" means greater than orequal".
- A graph consists of set of abstract objects called nodes and a relation R between thenodes.
- If aRb, which is to say that a has the relation R to b, it is denoted by a link from a tob.
- For some relations we can associate properties called as linkweights.

Transitive Relations

- A relation is transitive if aRb and bRc impliesaRc.
- Most relations used in testing aretransitive.
- Examples of transitive relations include: is connected to, is greater than or equal to, is less than or equal to, is a relative of, is faster than, is slower than, takes more time than, is a subset of, includes, shadows, is the bossof.
- Examples of intransitive relations include: is acquainted with, is a friend of, is a neighbor of, is lied to, has a du chainbetween.

Reflexive Relations

- A relation R is reflexive if, for every a,aRa.
- A reflexive relation is equivalent to a self loop at everynode.
- Examples of reflexive relations include: equals, is acquainted with, is a relativeof.
- Examples of irreflexive relations include: not equals, is a friend of, is on top of, isunder.

Symmetric Relations

- A relation R is symmetric if for every a and b, aRb implies bRa.
- A symmetric relation mean that if there is a link from a to b then there is also a link from b toa.
- A graph whose relations are not symmetric are called directedgraph.

- A graph over a symmetric relation is called an undirectedgraph.
- The matrix of an undirected graph is symmetric ($a_{ij}=a_{ji}$) for all(i,j)

Antisymmetric Relations

- A relation R is antisymmetric if for every a and b, if aRb and bRa, then a=b, or they are the same elements.
- Examples of antisymmetric relations: is greater than or equal to, is a subset of,time.
- Examples of nonantisymmetric relations: is connected to, can be reached from, is greater than, is a relative of, is a friendof

Equivalence Relations

- An equivalence relation is a relation that satisfies the reflexive, transitive, and symmetric properties.
- Equality is the most familiar example of an equivalencerelation.
- If a set of objects satisfy an equivalence relation, we say that they form an equivalence class over thatrelation.
- The importance of equivalence classes and relations is that any member of the equivalence class is, with respect to the relation, equivalent to any other member of thatclass.
- The idea behind partition testing strategies such as domain testing and path testing, is that we can partition the input space into equivalenceclasses.
- Testing any member of the equivalence class is as effective as testing themall.

Partial Ordering Relations

- A partial ordering relation satisfies the reflexive, transitive, and antisymmetricproperties.
- Partial ordered graphs have several important properties: they are loop free, there is at least one maximum element, and there is at least one minimumelement.

The Powers of a Matrix

- Each entry in the graph's matrix expresses a relation between the pair of nodes that corresponds to thatentry.
- Squaring the matrix yields a new matrix that expresses the relation between each pair of nodes via one intermediate node under the assumption that the relation istransitive.
- The square of the matrix represents all path segments two linkslong.
- The third power represents all path segments three linkslong.

atrix Powers and Products

- Given a matrix whose entries are aij, the square of that matrix is obtained by replacing every entrywith
  - n
  - $a_{ij}=\Sigma\, a_{ik}a_{kj}$
  - k=1
- more generally, given two matrices A and B with entries aik and bkj, respectively, their product is a new matrix C, whose entries are cij,where:
  - n
    - $C_{ij}=\Sigma\, a_{ik}b_{kj}$
    - k=1

3.1. The Set of AllPaths

Our main objective is to use matrix operations to obtain the set of all paths between all nodes or, equivalently, a property (described by link weights) over the set of all paths from every node to every other node, using the appropriate arithmetic rules for such weights. The set of all paths between all nodes is easily expressed in terms of matrix operations. It's given by the following infinite series of matrix powers:

This is an eloquent, but practically useless, expression. Let I be an *n* by *n* matrix, where *n* is the number of nodes. Let I's entries consist of multiplicative identity elements along the principal diagonal. For link names, this can be the number "1." For other kinds of weights, it is the multiplicative identity for those weights. The above product can be re-phrased as:

$$A(I + A + A^2 + A^3 + A^4 \ldots A^\infty)$$

But often for relations, $A + A = A$, $(A + I)^2 = A^2 + A + A + I\, A^2 + A + I$. Furthermore, for any

finite n, $(A + I)^n = I + A + A^2 + A^3 \ldots A^n$

Therefore, the original infinite sum can be replaced by

$$\sum_{i=1}^{\infty} A^i = A(A+I)^\infty$$

This is an improvement, because in the original expression we had both infinite products and infinite sums, and now we have only one infinite product to contend with. The above is valid whether or not there are loops. If we restrict our interest for the moment to paths of length $n - 1$, where *n* is the number of nodes, the set of all such paths is givenby

$$\sum_{i=1}^{n-1} A^i = A(A+I)^{n-2}$$

This is an interesting set of paths because, with *n* nodes, no path can exceed $n - 1$ nodes without incorporating some path segment that is already incorporated in some other path or path segment. Finding the set of all such paths is somewhat easier because it is not necessary to do all the intermediate

products explicitly. The following algorithm iseffective:

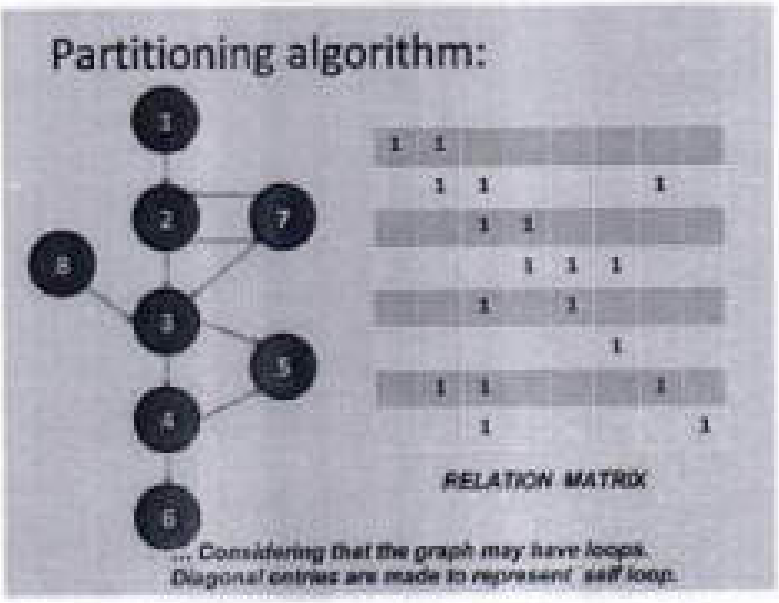
1. Express $n-2$ as a binarynumber.
2. Take successive squares of $(A+I)$, leading to $(A+I)^2$, $(A+I)^4$, $(A+I)^8$, and soon.
3. Keep only those binary powers of $(A+I)$ that correspond to a 1 value in the binary representation of$n-2$.
4. The set of all paths of length $n-1$ or less is obtained as the product of the matrices you got in step 3 with the original matrix.

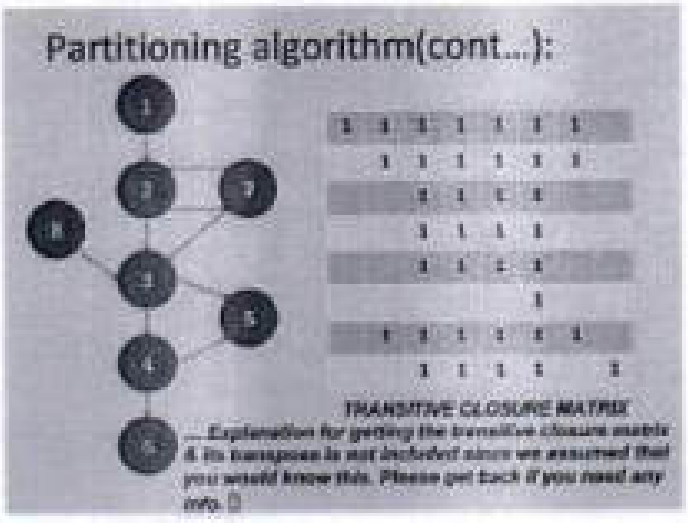
As an example, let the graph have 16 nodes. We want the set of all paths of length less than or equal to 15. The binary representation of $n-2$ (14) is 2^3+2^2+2. Consequently, the set of paths is given by

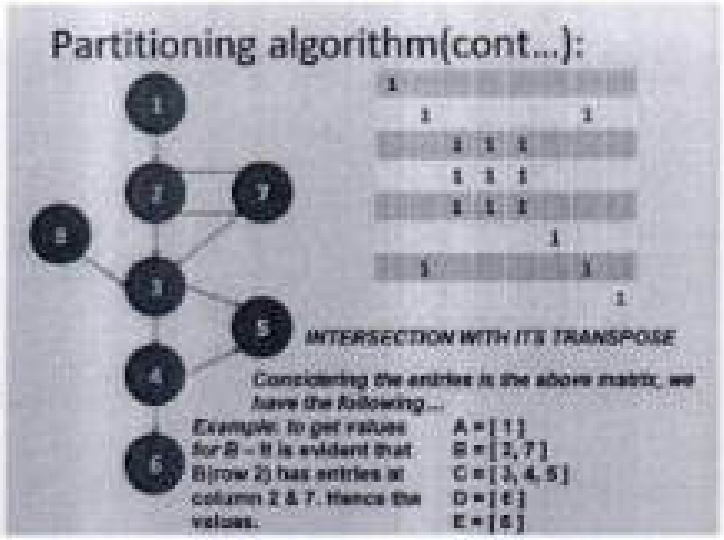
$$\sum_{i=1}^{15} A^i = A(A+I)^8(A+I)^4(A+I)^2$$

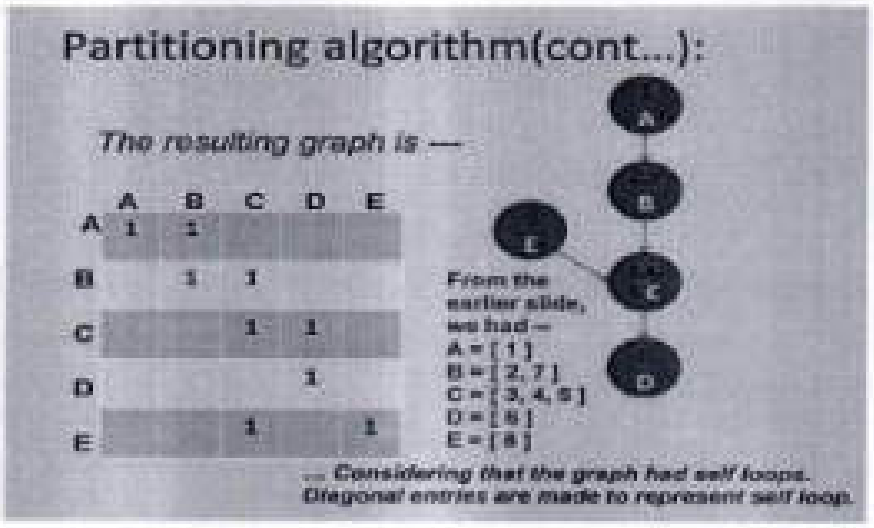
Partitioning Algorithm

- Consider any graph over a transitive relation. The graph may haveloops.
- We would like to partition the graph by grouping nodes in such a way that every loop iscontained within one group oranother.
- Such a graph is partiallyordered.
- There are many used for an algorithm that doesthat:
- We might want to embed the loops within a subroutine so as to have a resulting graph which is loop free at the toplevel.
- Many graphs with loops are easy to analyze if you know where to break theloops.
- While you and I can recognize loops, it's much harder to program a tool to do it unless you have a solid algorithm on which to base thetool.









Node Reduction Algorithm (General)

- The matrix powers usually tell us more than we want to know about mostgraphs.
- In the context of testing, we usually interested in establishing a relation between two nodes- typically the entry and exitnodes.
- In a debugging context it is unlikely that we would want to know the path expression between every node and every other node.
- The advantage of matrix reduction method is that it is more methodical than the graphical method called as node by node removalalgorithm.

1. Select a node for removal; replace the node by equivalent links that bypass that node and add those links to the links theyparallel.
2. Combine the parallel terms and simplify as youcan.
3. Observe loop terms and adjust the out links of every node that had a self loop to account for the effect of theloop.
4. The result is a matrix whose size has been reduced by 1. Continue until only the two nodes of interestexist.

Node Reduction Algorithm

	1	2	3	4	5
1			a		
2					
3		d		b	
4		c			f
5		g	e		h

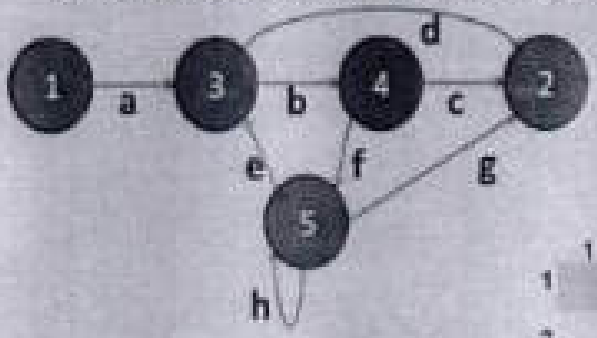
***STEP 1**: Eliminate a node and replace it with a set of equivalent links ...*

Say, we start with eliminating node 5-

TIP: The out-link of the node removed will correspond to the row and the in-link will correspond to the column

Node Reduction Algorithm

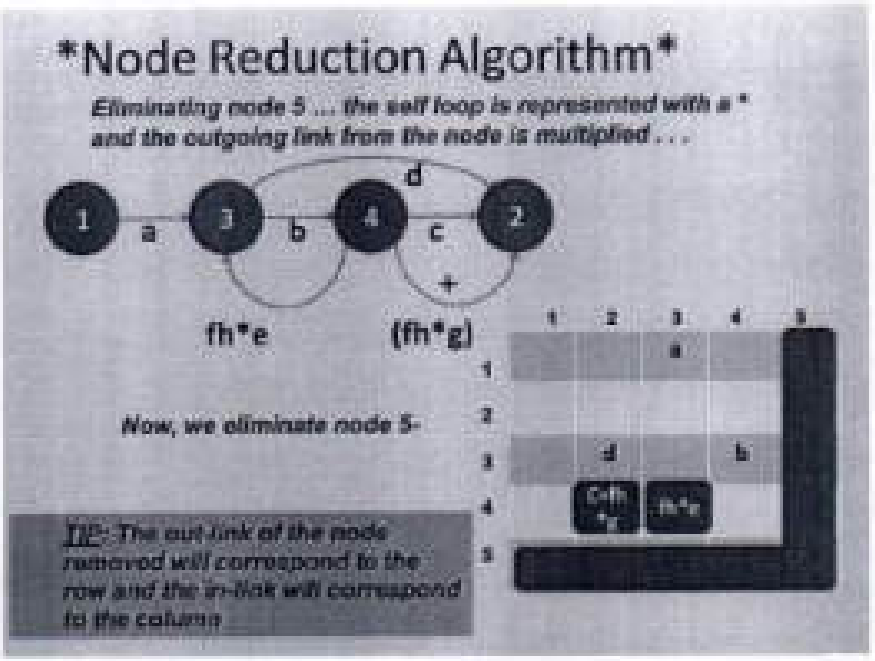
*Eliminating node 5 ... the self loop is represented with a * and the outgoing link from the node is multiplied . . .*

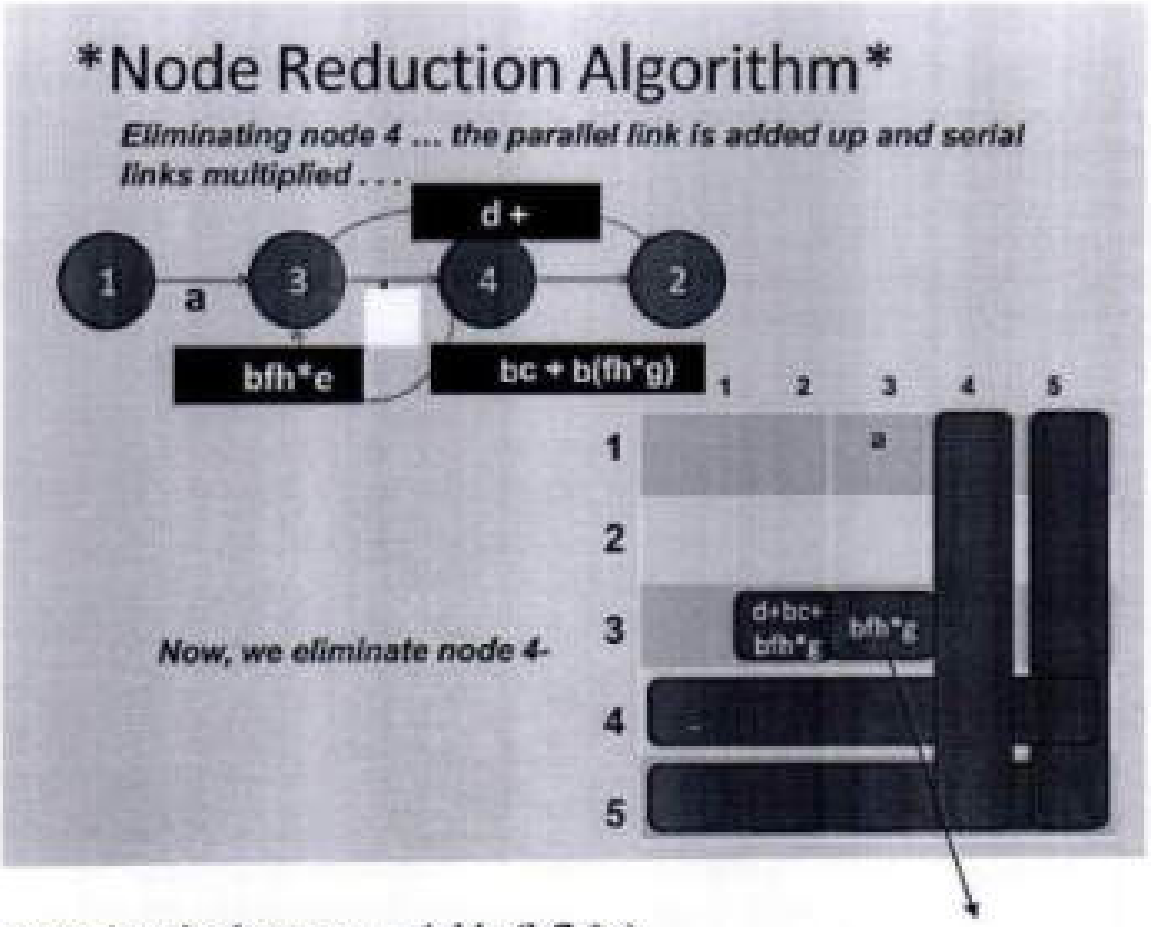


	1	2	3	4	5
1			a		
2					
3		d		b	
4		c			f
5		h^*g	h^*e		

Say, we start with eliminating node 5. First, we remove the self loop-

TIP: The out-link of the node removed will correspond to the row and the in-link will correspond to the column





Removing the loop term yields **(bfh*e)**

		a
	(bfh*e)*X(d+bc+bfh*g)	

The final result yields to :

a(bfh*e)*(d + bc + bfh * g)

BUILDING TOOLS:

Building tools *(node degree & graph density)*:

- The ***out-degree*** of a node is the number of out-links it has.
- The ***in-degree*** of a node is the number of in-links it has.
- The ***degree*** of a node is the sum of in-degree and out-degree.
- The ***average degree***(mean) of a node is b/w 3 and 4.
- Degree of a simple branch/junction is 3
- Degree of a loop contained in 1 statement is 4
- Mean degree of 4 or 5 ▯ very busy flow graph

What's wrong with arrays?

Matrix as a 2-dimensional array is not convenient for larger graphs. Herez why!...

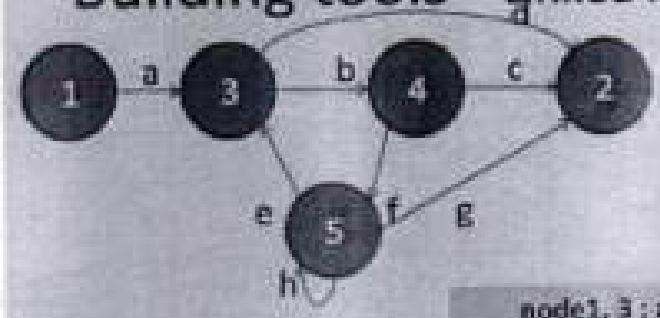
We have four reasons for the same-

- **Space**: For a matrix representation of an array, space grows as n^2 whereas, for a linked list, it grows only as kn, where k is a small number such as 3 or 4.
- **Weights**: Most weights in arrays are complicated and may have many components. This means that an additional weight matrix is required for each such weight.

What's wrong with arrays? (*cont...*)

- **Variable-length weights:** If the weights are regular expressions/algebraic expressions, we would then need a 2-dimensional string array (most of whose entries would be null).
- **Processing time:** Even though operations over null entries are fast, it still takes time to access such entries and discard them.
  The matrix representation forces us to spend a lot of time processing combinations of entries that we know will yield null results.

Building tools - Linked list representations:



Every node is a unique name/ number.
A link is a pair of node names.
The linked list entries for the above are ...

Format for linked list entries:
row, column ; data entry

node1, 3 ; a
node1,2,
node3, 2 ; d
node3, 4; b
node4, 2 ; c
node4, 5 ; f
node5, 2 ; g
node5, 3 ; e
node5, 5 ; h

	1	2	3	4	5
1			a		
2					
3		d		b	
4		c			f
5		g	e		h

The link names will usually be pointers to entries in a string array(where actual weight expressions are stored).

If the weights are fixed length, they can be associated with links in parallel-fixed entry length array.

Building tools - Linked list representations:

Clarifying entries by using node names & pointers...

1,3 ; a	1	node1, 3 ; a
2,	2	node2,exit
3,2 ; d	3	node3, 2 ; d
3,4; b		3, 4; b
4,2 ; c	4	node4, 2 ; c
4,5 ; f		4, 5 ; f
5,2 ; g	5	node5, 2 ; g
5,3 ; e		5, 3 ; e
5,5 ; h		5, 5 ; h

list entry	content
1	node1, 3 ; a
2	node2,exit
	3,
	4,
	5,
3	node3, 2 ; d
	3, 4; b
	1,
	5,
4	node4, 2 ; c
	4, 5 ; f
	3,
5	node5, 2 ; g
	3 ; e
	5 ; h
	4,
	5,

Fig.1 is represented in a more logical format in Fig.2 by representing the nodes under list entry. Fig 3 is more detailed the with in-links to that node added (highlighted in black).

Matrix operations:

- **Parallel Reduction**: the easiest operation. After sorting, parallel links are adjacent entries with the same pair of node names.

Example: Say we have 3 parallel links from node 17 to node 44. y, z & w are the pointers to the weight expressions.

Depicting all the entries for parallel links between node 17 & node 44, we have -

node 17, 21 ; x	[]	node 17, 21 ; x
, 44 ; y		, 44 ; y
, 44 ; z		*where, y = y + z + w*
, 44 ; w		

Matrix operations:

- **Loop Reduction**: the self loop is identified. To remove the loop, the link weight must be multiplied with all the out-links from that node.
  Start by identifying the out-links to be multiplied. Multiply the self loop (h*) *where h is the link weight of the self loop* with the out-link.

Example: *From the below entries, it is evident at entry(5,5;h) that it is a self loop at node 5 with link weight h. Also, from the 1st two entries, we see that the out-links from node 5 are 2 and 3. We need to multiply the self loop (h*) with the link weights of nodes going from node 5 to node 2 & 3.*

Refer fig. on slide 43 for the graph.

node 5, 2 ; g	node 5, 2 ; h*g
5, 3 ; e	5, 3 ; h*e
5, 5 ; h	

Matrix operations:

- **Cross-Term Reduction**: Select a node for reduction. The cross-term step requires that you combine every in-link to the node with every out-link from that node. The in-links are obtained by back pointers. The new links created be removing the node will be associated with the nodes of the in-links.
  Example: Say that the node to be removed was node 4.

list entry	content
2	node2, exit
	(inlink)4, 2
	(inlink)3,2
3	node3,2 ; d
	3,4 ; b
4	node4, 2 ; c
	4, 5 ; f
	(inlink)3,4; b
5	(inlink)4, 5

The changes are highlighted in dark red

list entry	content
2	node2, exit
	(inlink)3,2
3	node3,2 ; d
	3,2 ; bc
	3,5 ; bf
5	(inlink) 4, 5

NODE – REDUCTION OPTIMIZATION:

Node Reduction Optimization (*tips*):

- The optimum order for node reduction is to do the lowest degree nodes first.
- When a node with degree 3(*may be 1 in-link & 2out-links or 2 in-links & 1 out-link*) is removed , it reduces the total link count by 1 link.
- A degree 4 node keeps the link count the same & all higher degree nodes increase the link count.