Cause-Effect Graphing: Rigorous Test Case Design

Gary E. Mogyorodi, B.Math., M.B.A.
Certified Tester, Foundation Level (CTFL)
Certified Tester, Advanced Level – Functional Tester (CTAL-FT)
Certified Tester, Advanced Level – Test Manager (CTAL-TM)
President – Canadian Software Testing Board

Software Testing Services

Cause-Effect Graphing: Rigorous Test Case Design

For authorization to photocopy items for internal corporate use, personal use, or for educational and/or classroom use, please contact:

Gary Mogyorodi, B.Math., M.B.A. CTFL, CTAL-TM, CTAL-FT
President
Software Testing Services
Toronto, Ontario, Canada
www.softestserv.ca
garym@softestserv.ca

Agenda

- The Requirements-Based Testing Overview
  - Ambiguity Reviews
  - Cause-Effect Graphing
- Review of Test Case Design Techniques – Manual Techniques and those Supported with Tools
- Comparison of Cause-Effect Graphing to other Test Case Design Techniques Supported with Tools
- Benefits of Cause-Effect Graphing

Requirements-Based Testing – First Major Differentiator

1. Ambiguity Reviews

Performed in the requirements phase of software development to identify anything that is unclear, ambiguous or incomplete in the requirements. The elimination of these ambiguities improves the quality of those requirements.

The Ambiguity Review Checklist

- Dangling else
- Ambiguity of reference
- Scope of action
- Omissions
  - Causes without effects
  - Missing effects
  - Effects without causes
  - Complete omissions
  - Missing causes
- Ambiguous logical operators
  - Or, And, Nor, Nand
  - Implicit connectors
  - Compound operators
- Negation
  - Scope of negation
  - Unnecessary negation
  - Double negation
- Ambiguous statements
  - Verbs, adverbs, adjectives
  - Variables, unnecessary aliases
- Random organization
  - Mixed causes and effects
  - Random case sequence
- Built-in assumptions
  - Functional/environmental knowledge
- Ambiguous precedence relationships
- Implicit cases
- Etc.
- I.E. versus E.G.
- Temporal ambiguity
- Boundary ambiguity

(Bender RBT Inc.)

Why Do Ambiguity Reviews?
Relative Cost To Fix An Error

Phase In Which Found	Cost Ratio
Requirements	1
Design	3-6
Coding	10
System/Integration Testing	15-40
User Acceptance Testing	30-70
Operation	40-1000

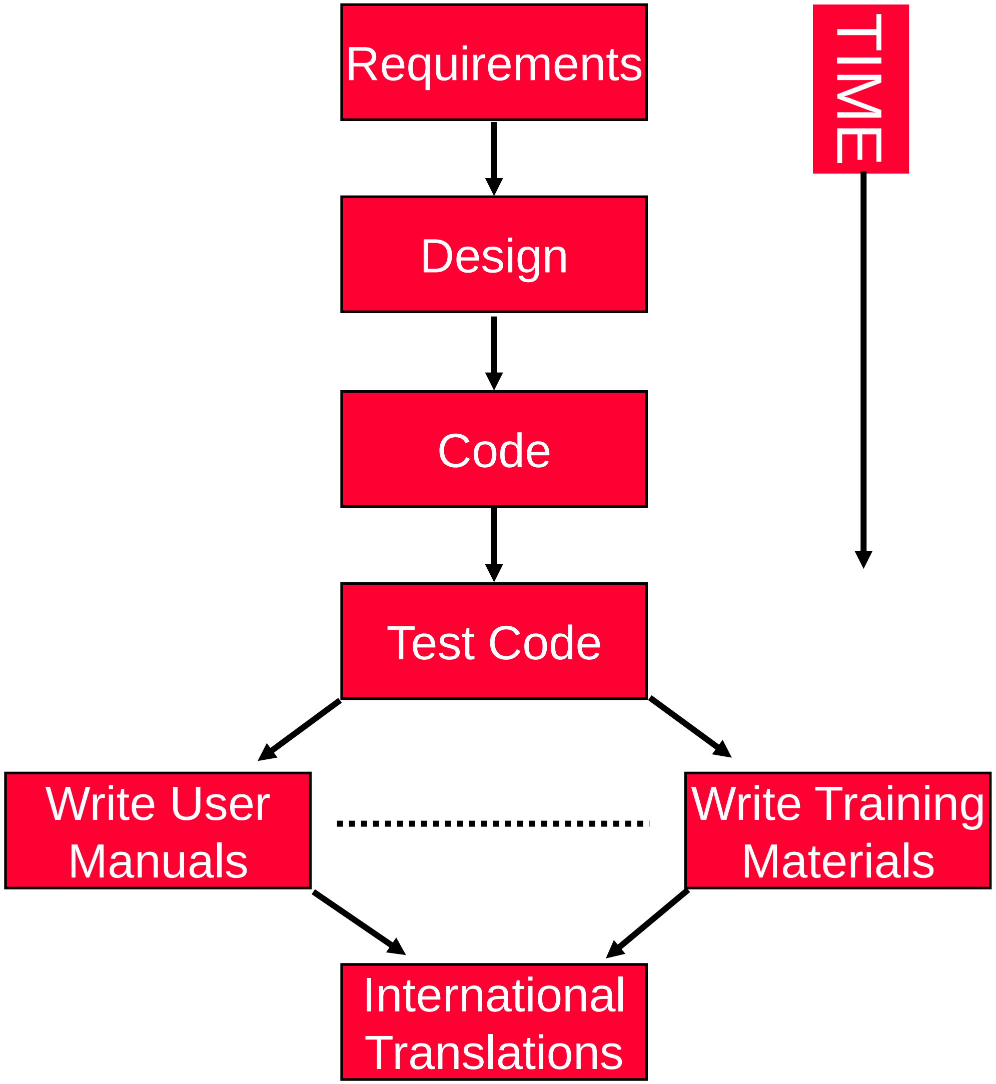
(IBM, et. al.)

Requirements-Based Testing – Second Major Differentiator

2. Cause-Effect Graphing

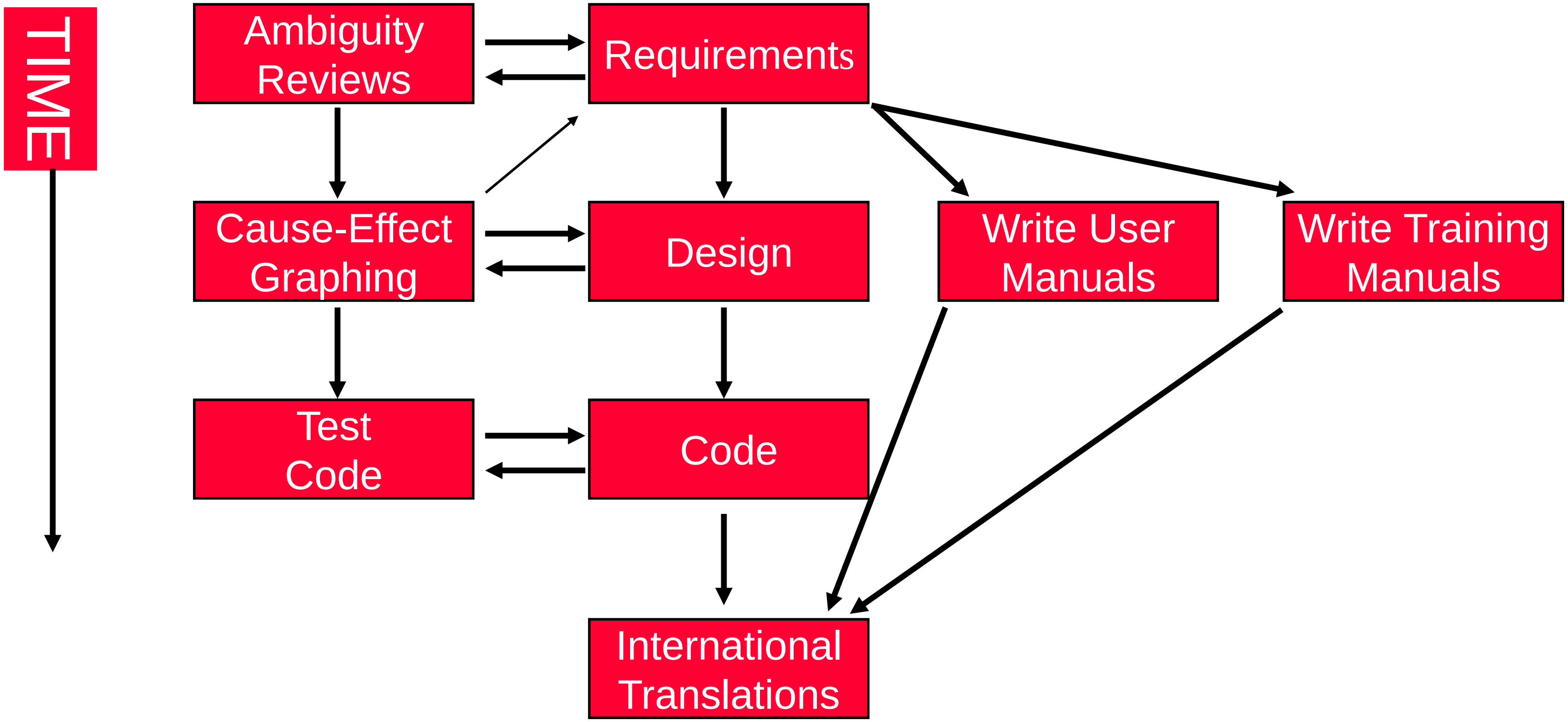
A test case design technique that is performed once requirements have been corrected for ambiguities. The Cause-Effect Graphing technique derives the minimum number of test cases to cover 100% of the functional requirements to improve the quality of test coverage.

The "Standard" Development Lifecycle



(Bender RBT Inc.)

Lifecycle Using Requirements-Based Testing



(Bender RBT Inc.)

Test Case Design Approaches

The Goal:

Design a necessary and sufficient set of test cases to ensure system integrity.

Possible approaches:

- Gut Feel
- Production Files
- ISTQB Foundation Level Techniques
- Test Case Design Techniques Supported by Tools

Testing By Gut Feel

Totally dependent on who is doing the testing:

- How experienced they are at testing
- How experienced they are in the application
- How experienced they are in the technology that the application runs on
- How they are feeling today

Even if all the tests run successfully, all you know is that ***those*** tests run -- not that the system runs successfully

(Bender RBT Inc.)

Testing With Production Files

- May covers less than 30% of the code
- Exception cases are not covered since data is already scrubbed of exceptions
- Time-dependent functions are not covered
- Expected results are not determined for every output field
- Might find some missing cases
- Have value in performance testing
- Have value in helping build test cases

(Bender RBT Inc.)

ISTQB Foundation Level Test Case Design Techniques

- Black Box techniques:
  - Equivalence Partitioning
  - Boundary Value Analysis
  - State Transition Diagrams
  - Decision Tables
  - Use Case Testing
- Each of these techniques is performed manually. There is no guarantee that test coverage is optimized, and no guarantee that the number of tests is minimized.

Test Case Design Techniques Supported by Tools

- Black Box techniques supported by tools:
  - 1. Classification Tree Method (ISTQB)
  - 3. Pairwise Testing (ISTQB)
  - 4. Combinatorial Testing
  - 5. Cause-Effect Graphing (ISTQB)
- NOTE: Only Cause-Effect Graphing produces complete test cases (inputs and outputs). All of these other techniques only produce input combinations. The tester has to manually derive the expected outputs from those input combinations to create complete test cases.

Test Case Design Comparison

Two important aspects of test case design: Efficiency and Effectiveness.

- Efficiency is measured by the number of test cases derived.
- Effectiveness is measured by the amount of coverage provided by the test cases.

Using an example set of requirements, compare the efficiency and effectiveness of the four test case design techniques supported by tools.

Example Requirements

This banking function has sixty-four possible combinations of inputs from which to select test cases:

If the customer is a business client or a preferred personal client,

and they have a checking account,

and they have $100,000 or more in deposits,

and they do not have overdraft protection,

and they have fewer than 5 overdrafts in the last 12 months,

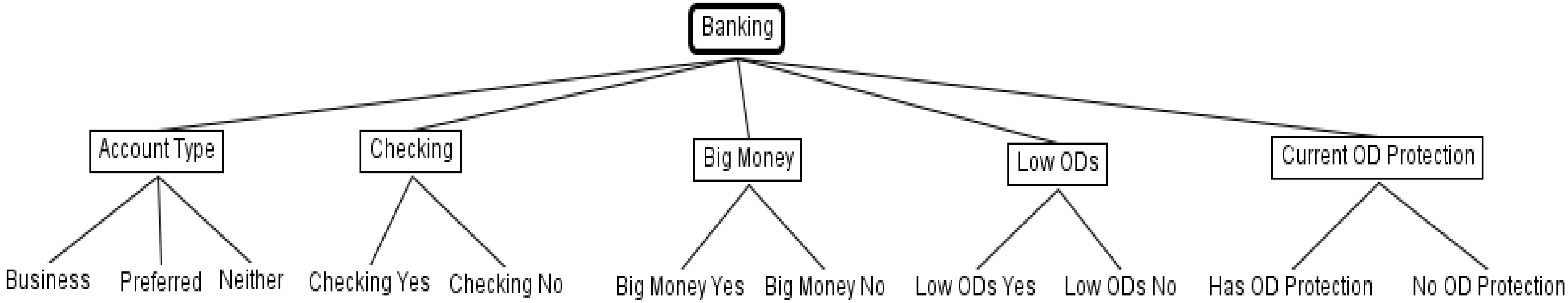
then set up free overdraft protection.

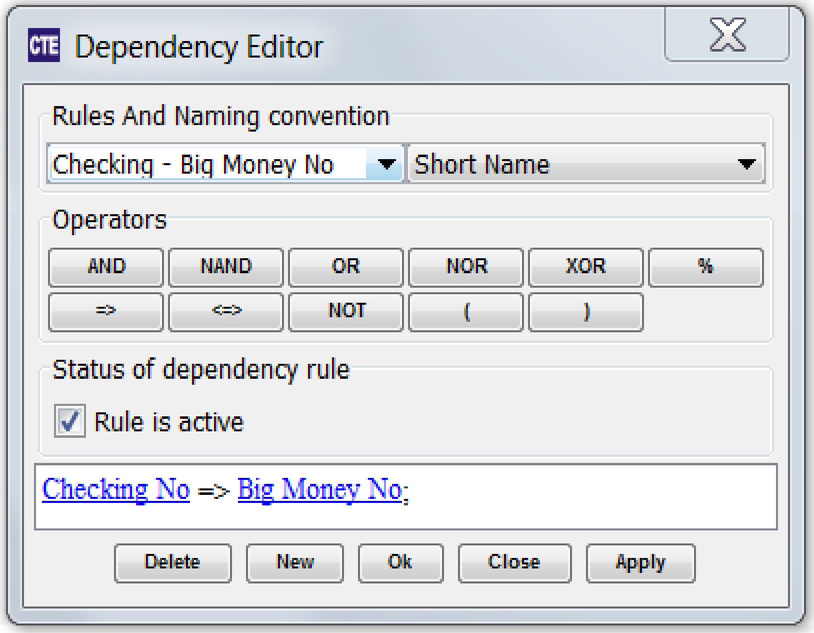
Otherwise, do not provide overdraft protection.

How many test cases are required to confirm that the function works?

(Bender RBT Inc.)

1. Classification Tree Method Using CTE XL





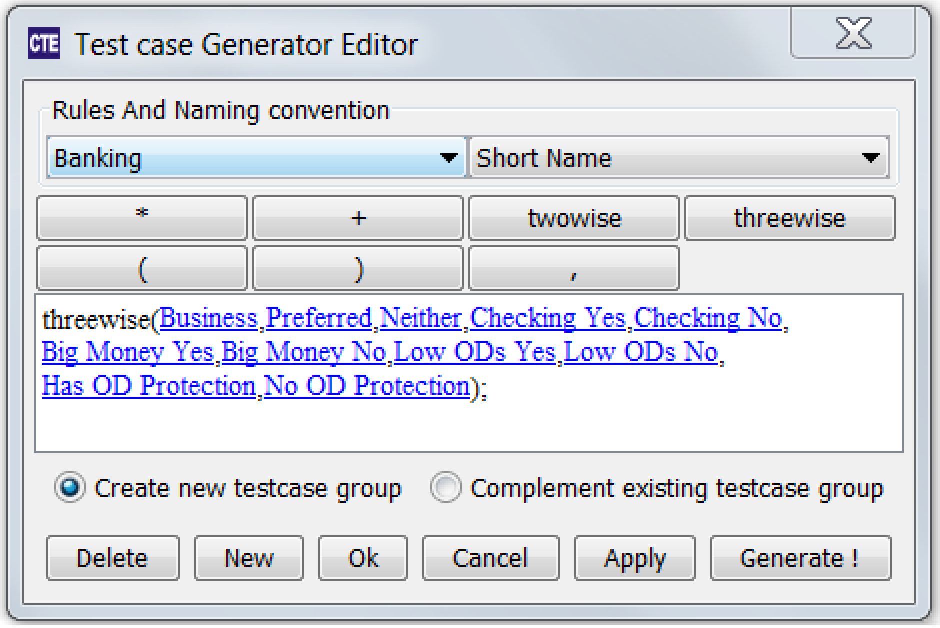
Rules And Naming convention

Checking - Big Money No | Short Name

Status of dependency rule

☑ Rule is active

Checking No => Big Money No;



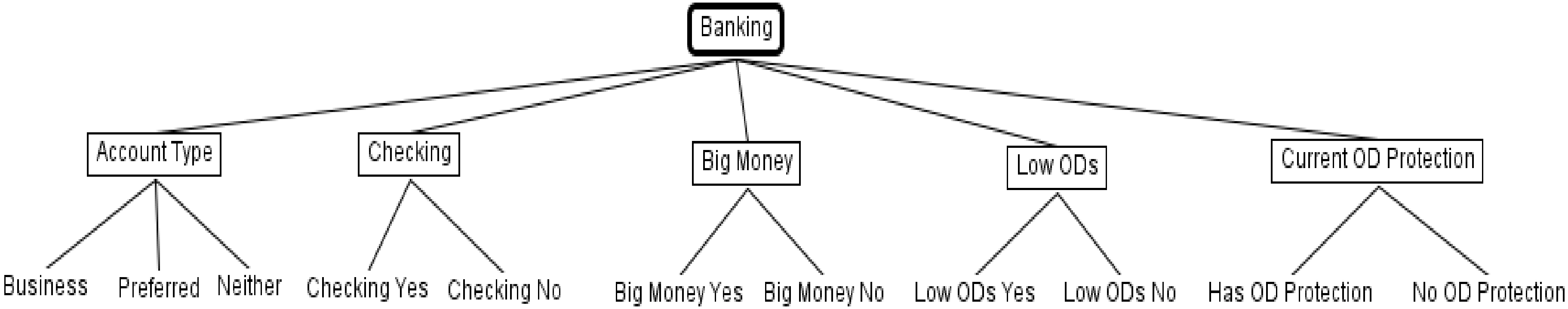
Rules And Naming convention

Banking | Short Name

threewise(Business,Preferred,Neither,Checking Yes,Checking No,
Big Money Yes,Big Money No,Low ODs Yes,Low ODs No,
Has OD Protection,No OD Protection);

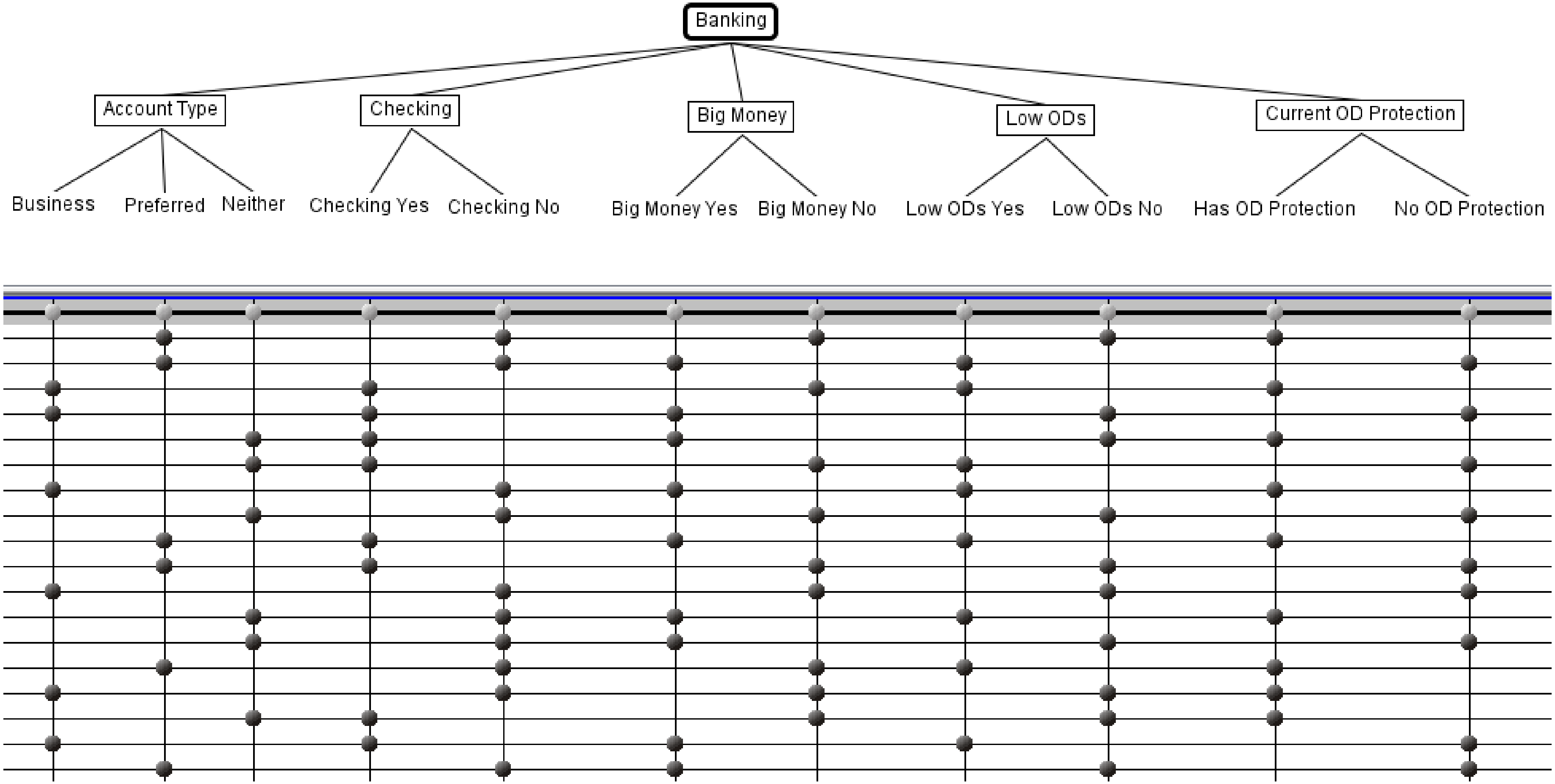
● Create new testcase group ○ Complement existing testcase group

1. Classification Tree Method Using CTE XL (Twowise) = 9 tests



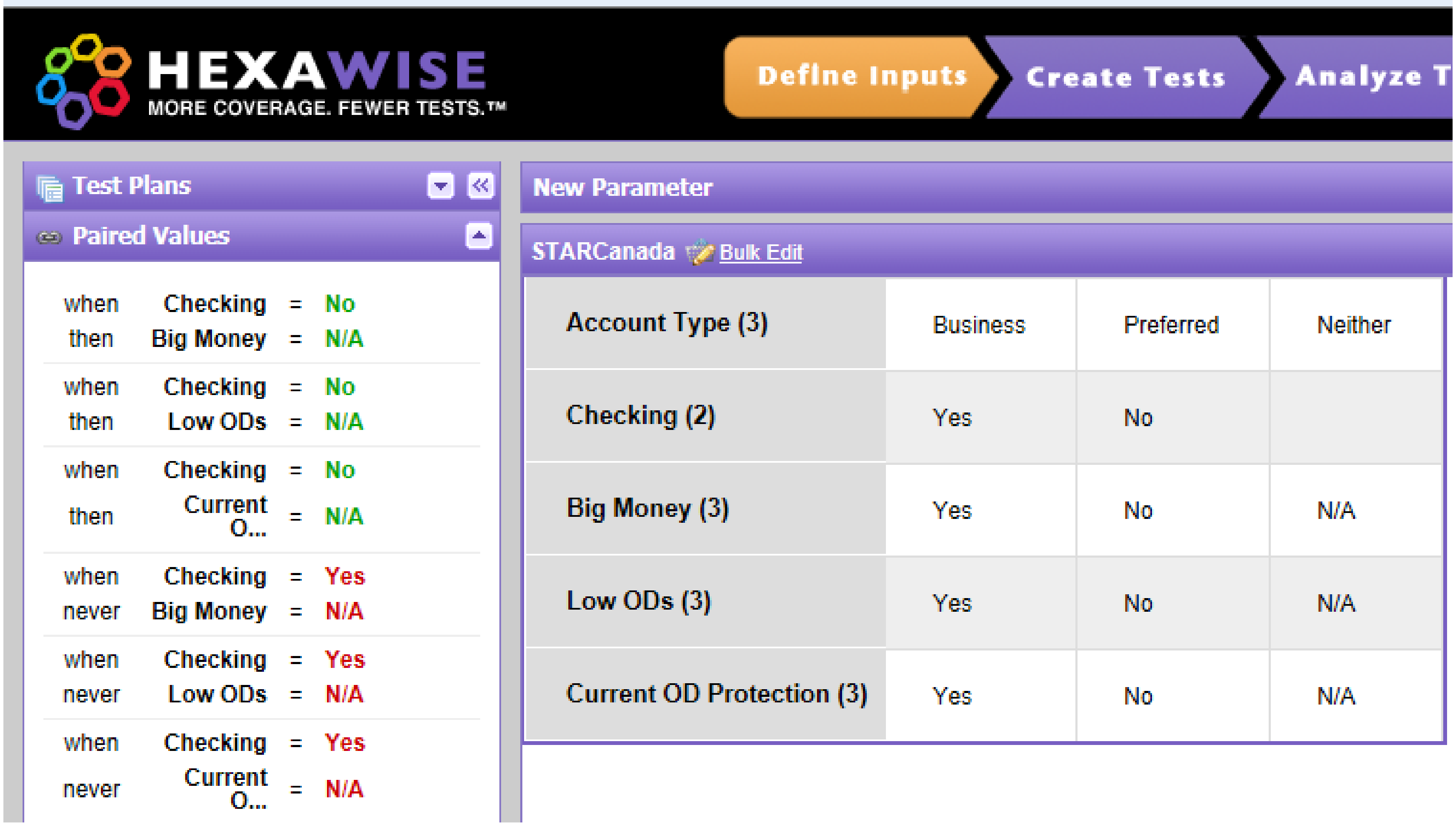
Business	Preferred	Neither	Checking Yes	Checking No	Big Money Yes	Big Money No	Low ODs Yes	Low ODs No	Has OD Protection	No OD Protection
	●			●	●			●	●	
●			●		●		●			●
		●	●			●	●		●	
		●		●		●		●		●
	●			●		●	●			●
●			●			●		●	●	
		●		●	●		●			●
●				●		●		●		●
	●		●		●		●		●	

1. Classification Tree Method Using CTE XL (Threewise) = 18 tests



Account Type			Checking		Big Money		Low ODs		Current OD Protection	
Business	Preferred	Neither	Checking Yes	Checking No	Big Money Yes	Big Money No	Low ODs Yes	Low ODs No	Has OD Protection	No OD Protection
	●			●		●		●	●	
	●			●	●		●			●
●			●			●	●		●	
●			●		●			●		●
		●	●		●			●	●	
		●	●			●	●			●
●				●	●		●		●	
		●		●		●		●		●
	●		●		●		●		●	
	●		●			●		●		●
●				●		●		●		●
		●		●	●		●		●	
		●		●	●			●		●
	●			●		●	●		●	
●				●		●		●	●	
		●	●			●		●	●	
●			●		●		●			●
	●			●	●			●		●

2. Pairwise and Combinatorial Testing Using Hexawise

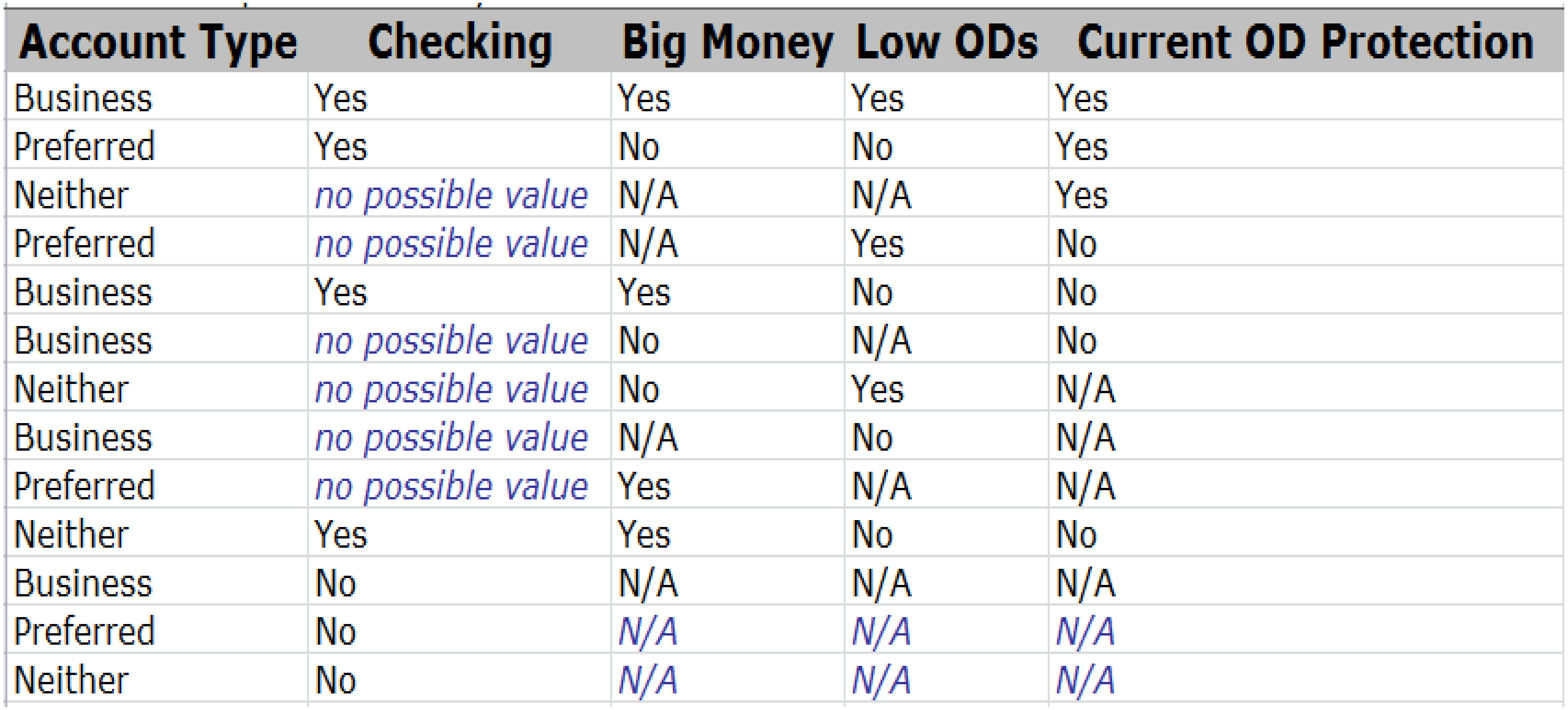


when	Checking	=	No
then	Big Money	=	N/A
when	Checking	=	No
then	Low ODs	=	N/A
when	Checking	=	No
then	Current O...	=	N/A
when	Checking	=	Yes
never	Big Money	=	N/A
when	Checking	=	Yes
never	Low ODs	=	N/A
when	Checking	=	Yes
never	Current O...	=	N/A

STARCanada

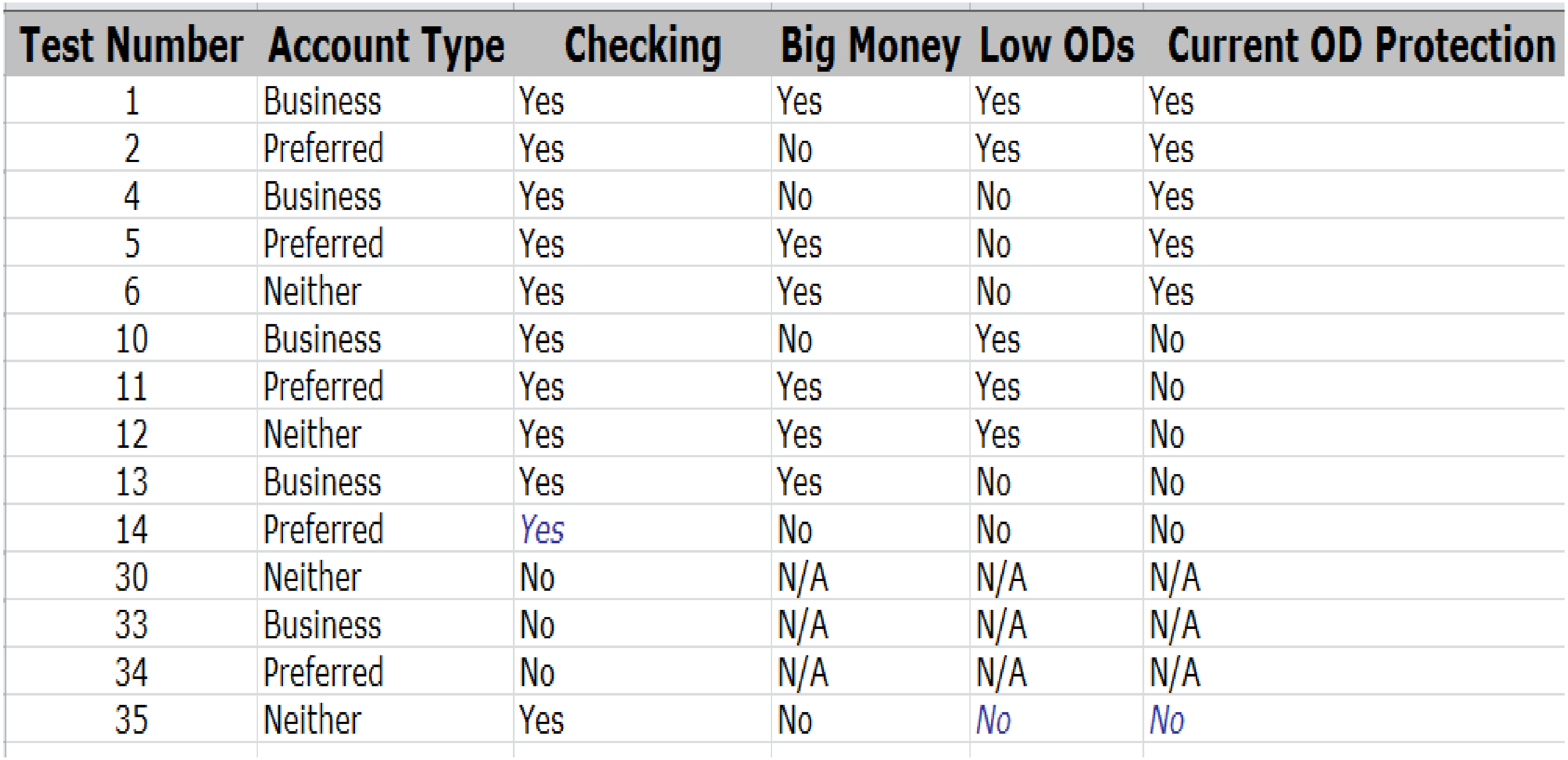
Account Type (3)	Business	Preferred	Neither
Checking (2)	Yes	No	
Big Money (3)	Yes	No	N/A
Low ODs (3)	Yes	No	N/A
Current OD Protection (3)	Yes	No	N/A

2. Pairwise Testing Using Hexawise 2-way = 7 tests



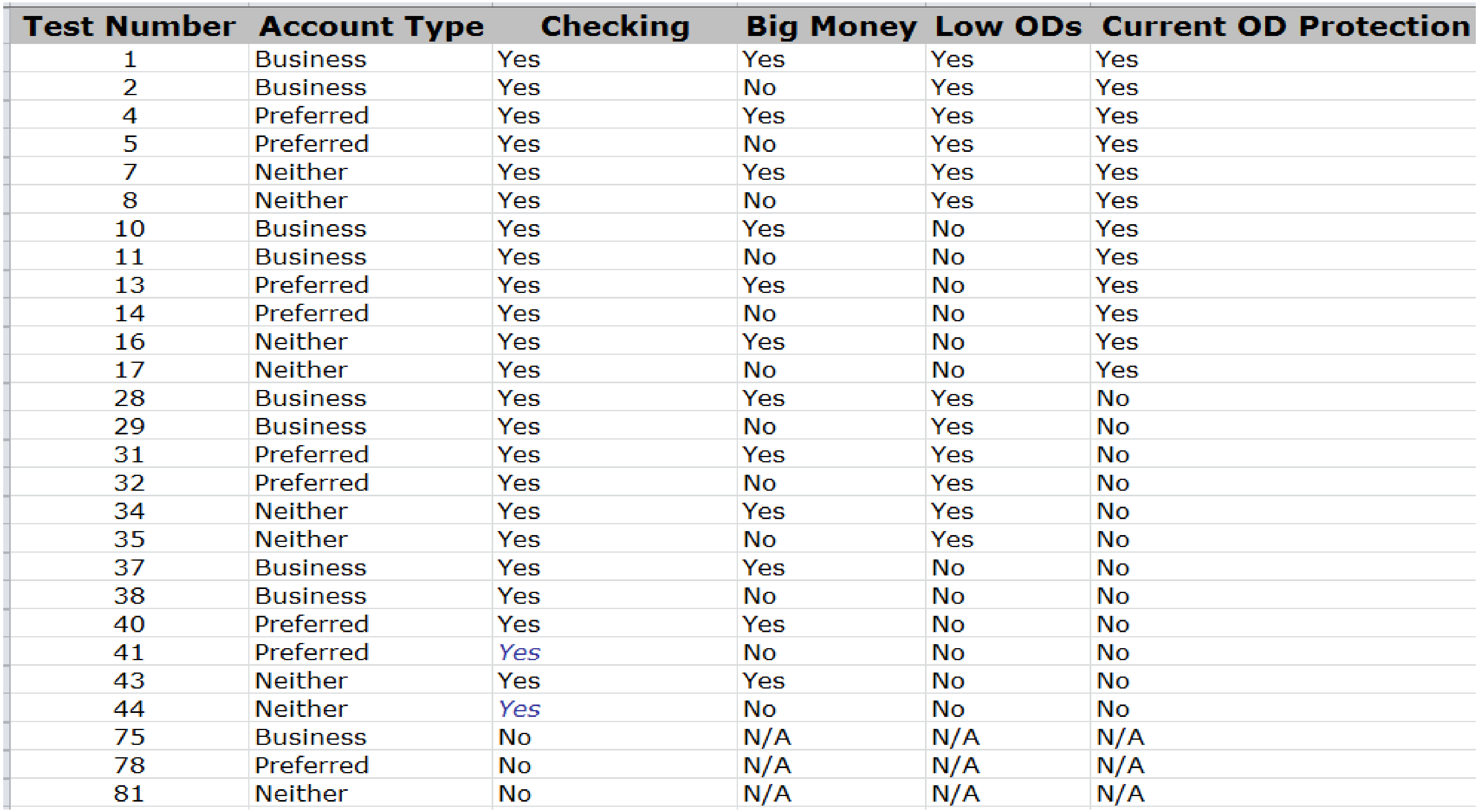
Account Type	Checking	Big Money	Low ODs	Current OD Protection
Business	Yes	Yes	Yes	Yes
Preferred	Yes	No	No	Yes
Neither	*no possible value*	N/A	N/A	Yes
Preferred	*no possible value*	N/A	Yes	No
Business	Yes	Yes	No	No
Business	*no possible value*	No	N/A	No
Neither	*no possible value*	No	Yes	N/A
Business	*no possible value*	N/A	No	N/A
Preferred	*no possible value*	Yes	N/A	N/A
Neither	Yes	Yes	No	No
Business	No	N/A	N/A	N/A
Preferred	No	*N/A*	*N/A*	*N/A*
Neither	No	*N/A*	*N/A*	*N/A*

2. Combinatorial Testing Using Hexawise 3-way = 14 tests



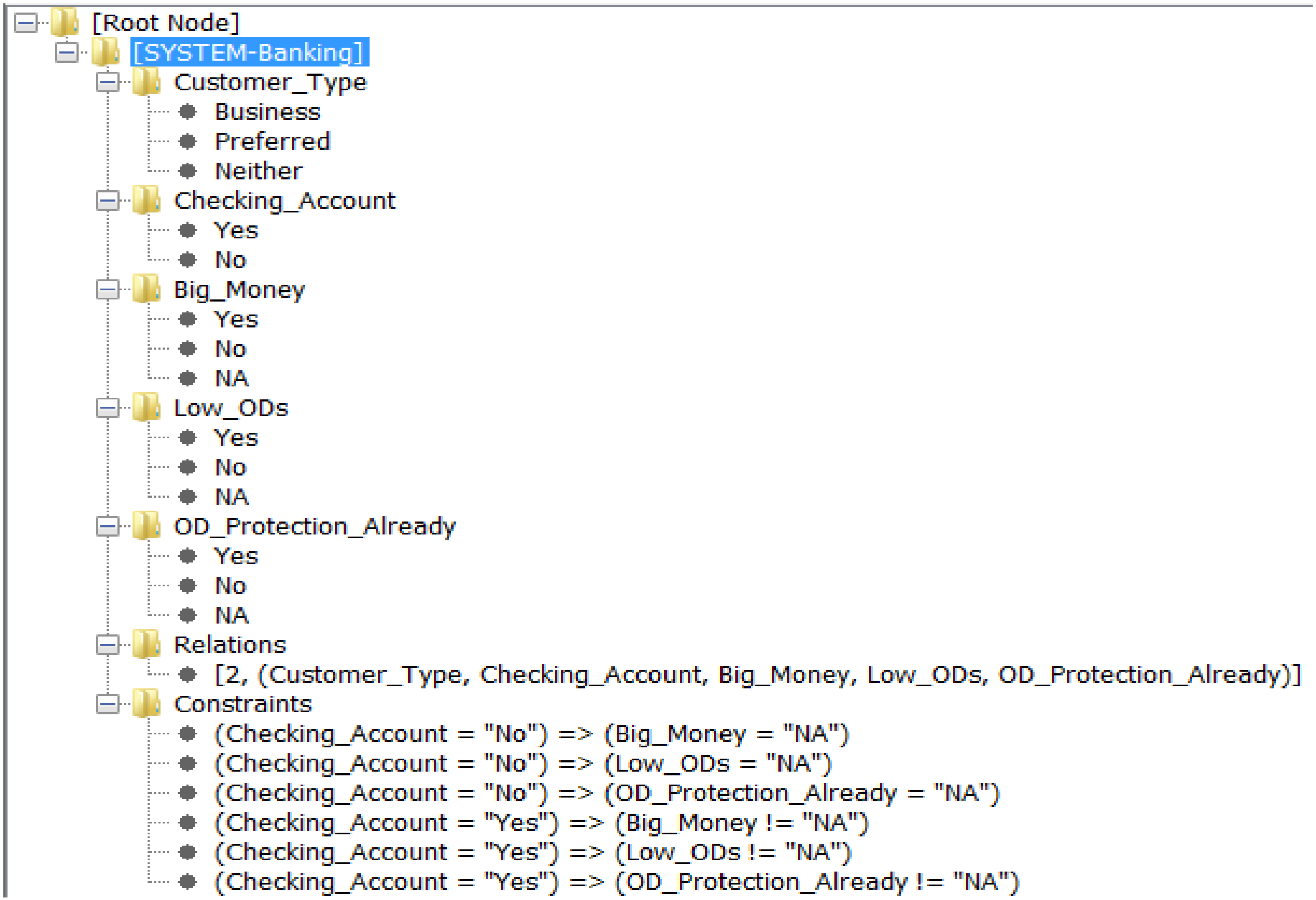
Test Number	Account Type	Checking	Big Money	Low ODs	Current OD Protection
1	Business	Yes	Yes	Yes	Yes
2	Preferred	Yes	No	Yes	Yes
4	Business	Yes	No	No	Yes
5	Preferred	Yes	Yes	No	Yes
6	Neither	Yes	Yes	No	Yes
10	Business	Yes	No	Yes	No
11	Preferred	Yes	Yes	Yes	No
12	Neither	Yes	Yes	Yes	No
13	Business	Yes	Yes	No	No
14	Preferred	*Yes*	No	No	No
30	Neither	No	N/A	N/A	N/A
33	Business	No	N/A	N/A	N/A
34	Preferred	No	N/A	N/A	N/A
35	Neither	Yes	No	*No*	*No*

2. Combinatorial Testing Using Hexawise 4-way = 27 tests



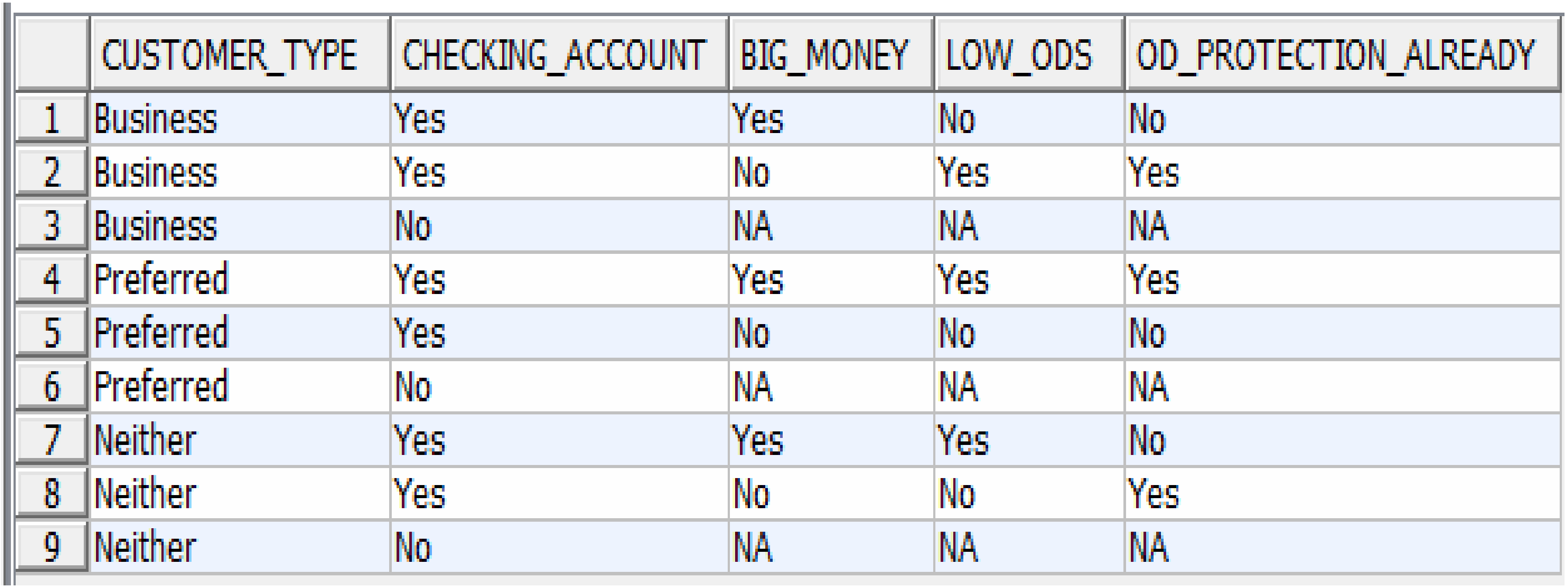
Test Number	Account Type	Checking	Big Money	Low ODs	Current OD Protection
1	Business	Yes	Yes	Yes	Yes
2	Business	Yes	No	Yes	Yes
4	Preferred	Yes	Yes	Yes	Yes
5	Preferred	Yes	No	Yes	Yes
7	Neither	Yes	Yes	Yes	Yes
8	Neither	Yes	No	Yes	Yes
10	Business	Yes	Yes	No	Yes
11	Business	Yes	No	No	Yes
13	Preferred	Yes	Yes	No	Yes
14	Preferred	Yes	No	No	Yes
16	Neither	Yes	Yes	No	Yes
17	Neither	Yes	No	No	Yes
28	Business	Yes	Yes	Yes	No
29	Business	Yes	No	Yes	No
31	Preferred	Yes	Yes	Yes	No
32	Preferred	Yes	No	Yes	No
34	Neither	Yes	Yes	Yes	No
35	Neither	Yes	No	Yes	No
37	Business	Yes	Yes	No	No
38	Business	Yes	No	No	No
40	Preferred	Yes	Yes	No	No
41	Preferred	*Yes*	No	No	No
43	Neither	Yes	Yes	No	No
44	Neither	*Yes*	No	No	No
75	Business	No	N/A	N/A	N/A
78	Preferred	No	N/A	N/A	N/A
81	Neither	No	N/A	N/A	N/A

3. Combinatorial Testing Using ACTS



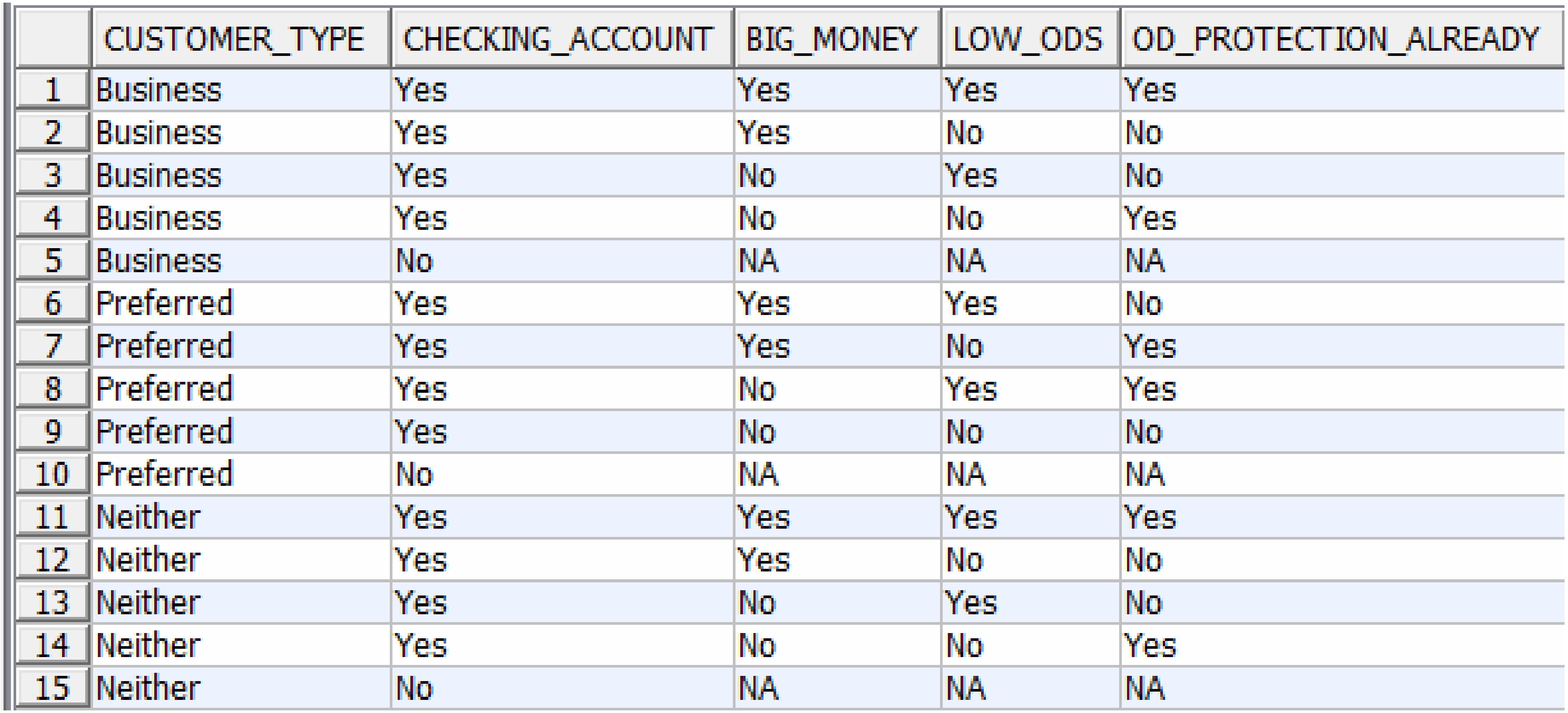
```
[Root Node]
  [SYSTEM-Banking]
    Customer_Type
      Business
      Preferred
      Neither
    Checking_Account
      Yes
      No
    Big_Money
      Yes
      No
      NA
    Low_ODs
      Yes
      No
      NA
    OD_Protection_Already
      Yes
      No
      NA
    Relations
      [2, (Customer_Type, Checking_Account, Big_Money, Low_ODs, OD_Protection_Already)]
    Constraints
      (Checking_Account = "No") => (Big_Money = "NA")
      (Checking_Account = "No") => (Low_ODs = "NA")
      (Checking_Account = "No") => (OD_Protection_Already = "NA")
      (Checking_Account = "Yes") => (Big_Money != "NA")
      (Checking_Account = "Yes") => (Low_ODs != "NA")
      (Checking_Account = "Yes") => (OD_Protection_Already != "NA")
```

3. Combinatorial Testing Using ACTS (Strength = 2) = 9 tests



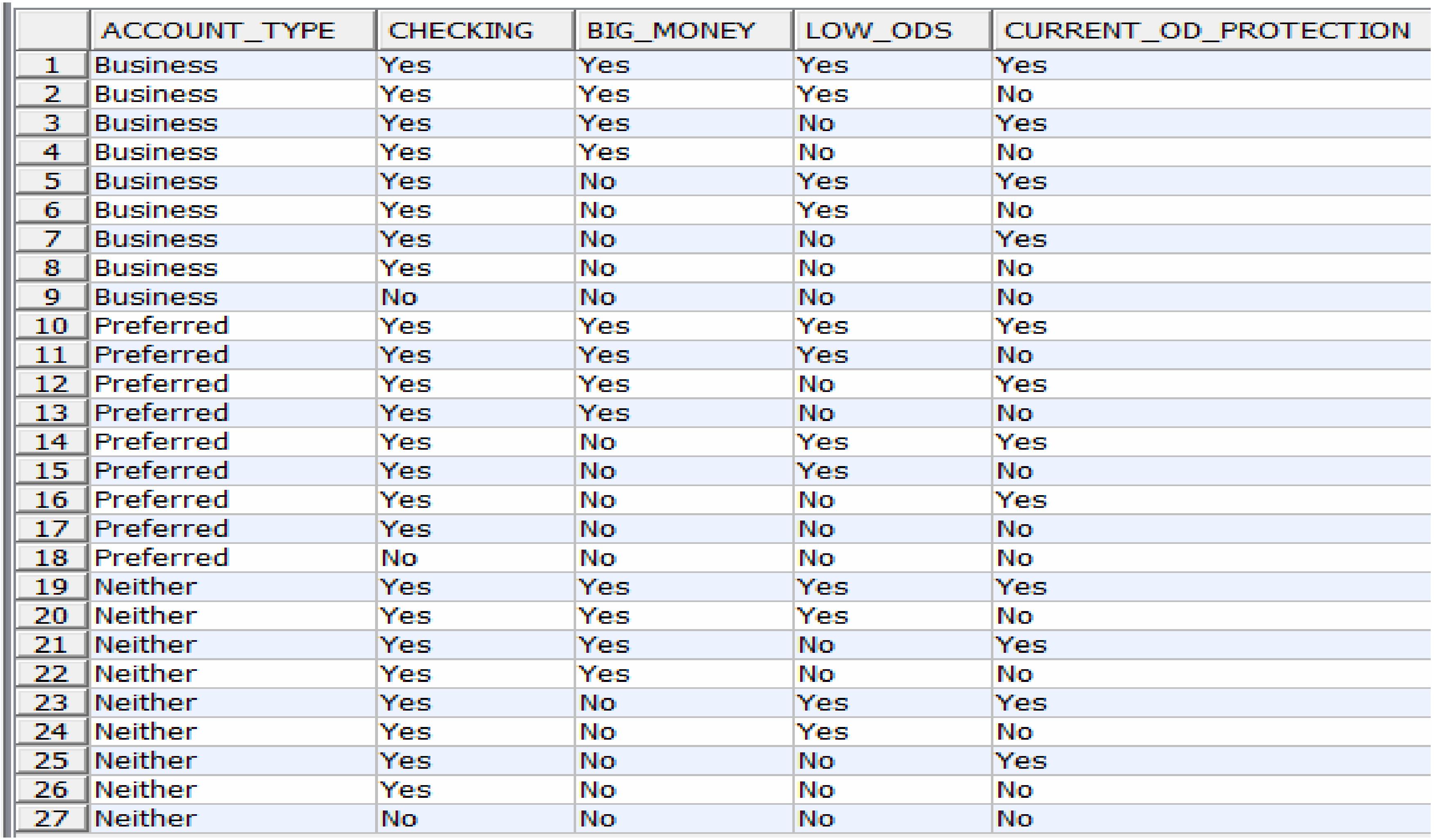
	CUSTOMER_TYPE	CHECKING_ACCOUNT	BIG_MONEY	LOW_ODS	OD_PROTECTION_ALREADY
1	Business	Yes	Yes	No	No
2	Business	Yes	No	Yes	Yes
3	Business	No	NA	NA	NA
4	Preferred	Yes	Yes	Yes	Yes
5	Preferred	Yes	No	No	No
6	Preferred	No	NA	NA	NA
7	Neither	Yes	Yes	Yes	No
8	Neither	Yes	No	No	Yes
9	Neither	No	NA	NA	NA

3. Combinatorial Testing Using ACTS (Strength = 3) = 15 tests



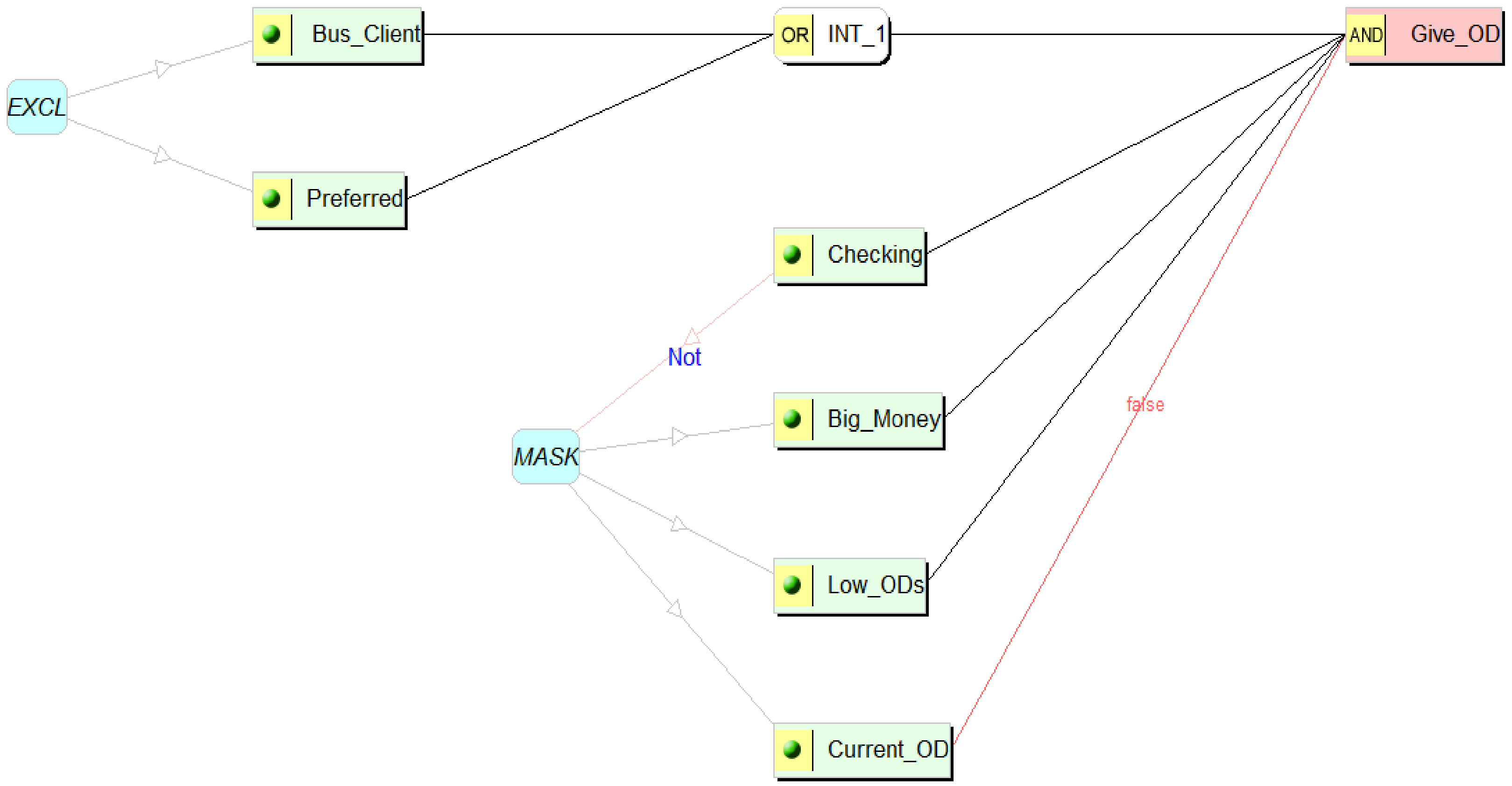
	CUSTOMER_TYPE	CHECKING_ACCOUNT	BIG_MONEY	LOW_ODS	OD_PROTECTION_ALREADY
1	Business	Yes	Yes	Yes	Yes
2	Business	Yes	Yes	No	No
3	Business	Yes	No	Yes	No
4	Business	Yes	No	No	Yes
5	Business	No	NA	NA	NA
6	Preferred	Yes	Yes	Yes	No
7	Preferred	Yes	Yes	No	Yes
8	Preferred	Yes	No	Yes	Yes
9	Preferred	Yes	No	No	No
10	Preferred	No	NA	NA	NA
11	Neither	Yes	Yes	Yes	Yes
12	Neither	Yes	Yes	No	No
13	Neither	Yes	No	Yes	No
14	Neither	Yes	No	No	Yes
15	Neither	No	NA	NA	NA

3. Combinatorial Testing Using ACTS (Strength = 4) = 27 tests



	ACCOUNT_TYPE	CHECKING	BIG_MONEY	LOW_ODS	CURRENT_OD_PROTECTION
1	Business	Yes	Yes	Yes	Yes
2	Business	Yes	Yes	Yes	No
3	Business	Yes	Yes	No	Yes
4	Business	Yes	Yes	No	No
5	Business	Yes	No	Yes	Yes
6	Business	Yes	No	Yes	No
7	Business	Yes	No	No	Yes
8	Business	Yes	No	No	No
9	Business	No	No	No	No
10	Preferred	Yes	Yes	Yes	Yes
11	Preferred	Yes	Yes	Yes	No
12	Preferred	Yes	Yes	No	Yes
13	Preferred	Yes	Yes	No	No
14	Preferred	Yes	No	Yes	Yes
15	Preferred	Yes	No	Yes	No
16	Preferred	Yes	No	No	Yes
17	Preferred	Yes	No	No	No
18	Preferred	No	No	No	No
19	Neither	Yes	Yes	Yes	Yes
20	Neither	Yes	Yes	Yes	No
21	Neither	Yes	Yes	No	Yes
22	Neither	Yes	Yes	No	No
23	Neither	Yes	No	Yes	Yes
24	Neither	Yes	No	Yes	No
25	Neither	Yes	No	No	Yes
26	Neither	Yes	No	No	No
27	Neither	No	No	No	No

4. Cause-Effect Graph Using Bender RBT



4. Cause-Effect Graphing Using Bender RBT = 7 test cases

TEST#1 -- Automatic Check For Overdraft Protection

Cause(s):

The customer is a business client
The customer has a checking account
The customer has $100,000 or more in deposits
The customer does not have overdraft protection
Overdrawn less than five times in last 12 months

Effect(s):

Set up free overdraft protection

TEST#2 -- Automatic Check For Overdraft Protection

Cause(s):

The customer is a preferred personal client
The customer has a checking account
The customer has $100,000 or more in deposits
The customer does not have overdraft protection
Overdrawn less than five times in last 12 months

Effect(s):

Set up free overdraft protection

TEST#3 -- Automatic Check For Overdraft Protection

Cause(s):

NOT The customer is a business client
NOT The customer is a preferred personal client
The customer has a checking account
The customer has $100,000 or more in deposits
The customer does not have overdraft protection
Overdrawn less than five times in last 12 months

Effect(s):

Do not set up free overdraft protection

TEST#4 -- Automatic Check For Overdraft Protection

Cause(s):

The customer is a business client
The customer does not have a checking account

Effect(s):

Do not set up free overdraft protection

4. Cause-Effect Graphing Using Bender RBT = 7 test cases

TEST#5 -- Automatic Check For Overdraft Protection

Cause(s):

The customer is a business client

The customer has a checking account

The customer has less than $100,000 in deposits

The customer does not have overdraft protection

Overdrawn less than five times in last 12 months

Effect(s):

Do not set up free overdraft protection

TEST#6 -- Automatic Check For Overdraft Protection

Cause(s):

The customer is a business client

The customer has a checking account

The customer has $100,000 or more in deposits

The customer currently has overdraft protection

Overdrawn less than five times in last 12 months

Effect(s):

Do not set up free overdraft protection

TEST#7 -- Automatic Check For Overdraft Protection

Cause(s):

The customer is a business client

The customer has a checking account

The customer has $100,000 or more in deposits

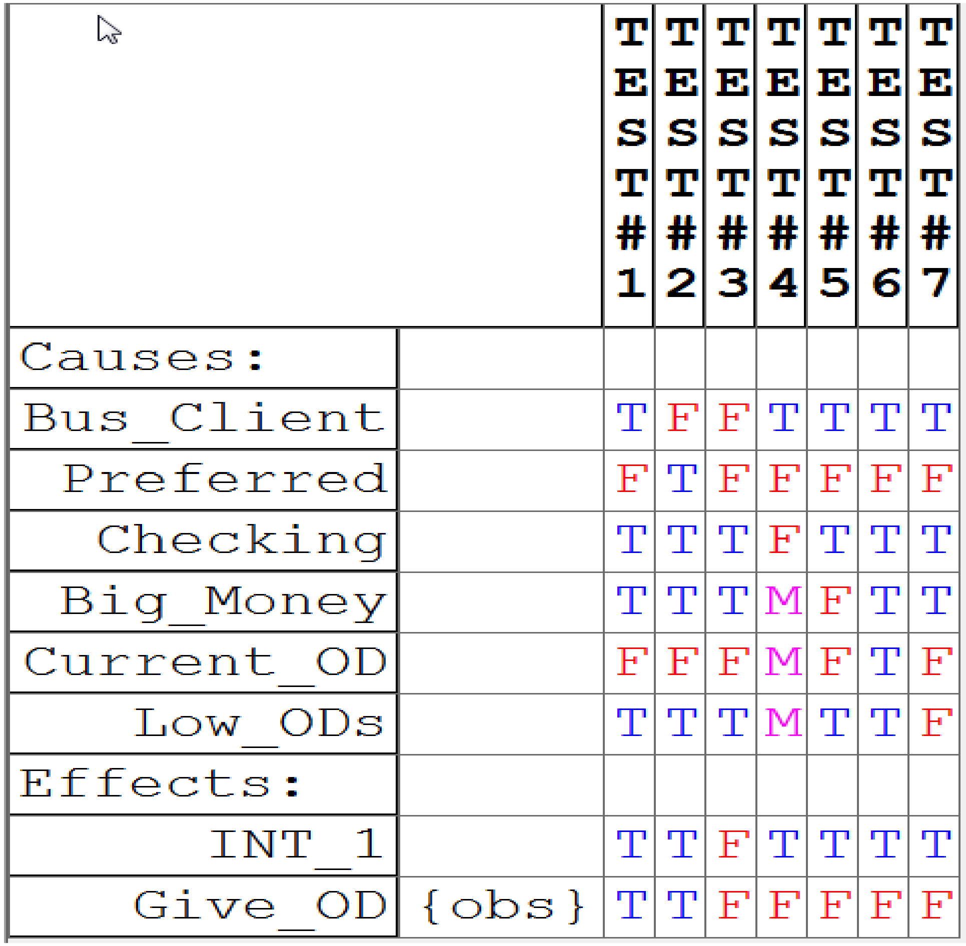
The customer does not have overdraft protection

Overdrawn more than 4 times in last 12 months

Effect(s):

Do not set up free overdraft protection

4. Decision Table Output Using Bender RBT



		TEST #1	TEST #2	TEST #3	TEST #4	TEST #5	TEST #6	TEST #7
Causes:								
Bus_Client		T	F	F	T	T	T	T
Preferred		F	T	F	F	F	F	F
Checking		T	T	T	F	T	T	T
Big_Money		T	T	T	M	F	T	T
Current_OD		F	F	F	M	F	T	F
Low_ODs		T	T	T	M	T	T	F
Effects:								
INT_1		T	T	F	T	T	T	T
Give_OD	{obs}	T	T	F	F	F	F	F

4. Cause-Effect Graphing Test Statistics Using Bender RBT

```
Test Statistics
Automatic Check For Overdraft Protection

Run:  Synthesis of New Tests
Number of input statements:  16

Number of Functional Variations:  9
Number of infeasible variations:  0
Number of untestable variations:  0

Number of new test cases defined:  7
Number of tested variations:       9
Number of Feasible Variations:     9
Percentage of functional coverage of feasible variations:
   9/9*100 = 100%

Number of tested variations:       9
Percentage of functional coverage of testable variations:
   9/9*100 = 100%

Number of Primary Causes:  6
The THEORETICAL maximum number of test cases is:
   2^6 = 64

The number of test cases generated by Bender RBT is:  7
```

Bender RBT provides summary statistics to aid in project estimating and test tracking.

4. Functional Specification Output from the Cause-Effect Graph Using Bender RBT

```
1. IF [The customer is a business client
        OR The customer is a preferred personal client]
        AND The customer has a checking account
        AND The customer has $100,000 or more in deposits
        AND The customer does not have overdraft protection
        AND Overdrawn less than five times in last 12 months
      THEN Set up free overdraft protection
      ELSE Do not set up free overdraft protection.
```

****

In addition, the following constraints must be applied to the above specifications:

1. WHEN: The customer does not have a checking account
THEN the following condition(s) are Indeterminate:
    The customer has $100,000 or more in deposits.
    The customer currently has overdraft protection
    Overdrawn less than five times in last 12 months

2. At most ONE (or NONE) of the following conditions may exist:
    The customer is a preferred personal client
    The customer is a business client

4. Test Coverage Comparison Using Bender RBT

Bender RBT has the ability to compare the test coverage provided by other test case design techniques to its test coverage for the same problem.

Test Coverage Comparison

Test Case Design Technique	Number of Test Cases	Test Case Coverage
Classification Tree Twowise	9	22%
Classification Tree Threewise	18	55%
Hexawise Pairwise 2-way	7	33%
Hexawise Combinatorial 3-way	14	88%
Hexawise Combinatorial 4-way	27	100%
Combinatorial Testing Strength = 2	9	55%
Combinatorial Testing Strength = 3	15	66%
Combinatorial Testing Strength = 4	27	100%
Cause-Effect Graphing	7	100%

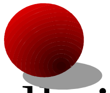
Test Statistics For a Large Problem Using Bender RBT

```
Test Statistics
CHP_PG5_26/TOBACCO USE STATISTICS

Run:  Synthesis of New Tests
Number of input statements:  112

Number of Functional Variations:  141
Number of infeasible variations:  0
Number of untestable Variations:  1

Number of new test cases defined:  22
Number of tested variations:       140
Number of Feasible Variations:     141
Percentage of functional coverage of feasible variations:
   140/141*100 = 99%

Number of tested variations:       140
Percentage of functional coverage of testable variations:
   140/140*100 = 100%

Number of Primary Causes:  37
The THEORETICAL maximum number of test cases is:
   2^37 = 137,438,953,472

The number of test cases generated by BenderRBT is:  22
```

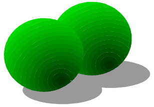
(Bender RBT Inc.)

Justification for Rigorous Testing

- Thought experiment:



  - Put 137,438,953,450 **red** balls in this room
  - Add 22 **green** balls to the room and mix well
  - Turn out the lights



- Pull out 22 balls
  - What is the probability that you have selected the 22 **green** ones?
- Pull out 1,000 balls
  - What is the probability that you have the 22 **green** ones now?
- Pull out 1,000,000 balls
  - What is the probability that you have the 22 **green** ones now?

** This is what "GUT FEEL" testing really is.**

(Bender RBT Inc.)

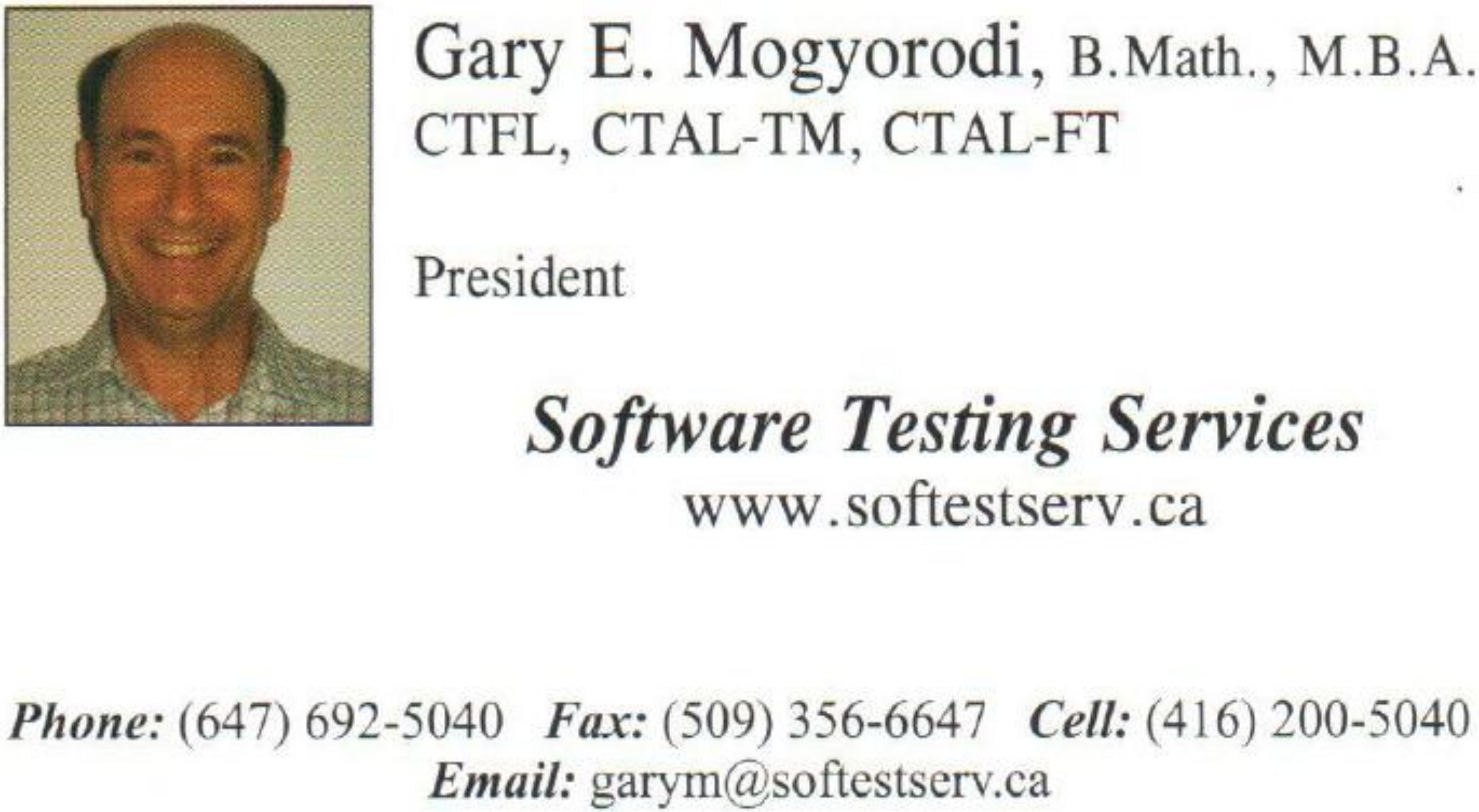
Benefits of Cause-Effect Graphing

- ***Maximum*** coverage with ***minimum*** test cases (better results than any other test case design technique)
  - **100%** functional coverage
  - 70-90% code coverage
- Identifies gaps in requirements as the test cases are being derived
- Test cases can be created for any application written in any language running on any platform

(Bender RBT Inc.)

Questions?

Contact Information



Gary E. Mogyorodi, B.Math., M.B.A.
CTFL, CTAL-TM, CTAL-FT

President

Software Testing Services
www.softestserv.ca

Phone: (647) 692-5040 ***Fax:*** (509) 356-6647 ***Cell:*** (416) 200-5040
Email: garym@softestserv.ca

References

- CTE XL 1.9.3 software by Berner & Mattner Systemtechnik GmbH http://www.berner-mattner.com/en/berner-mattner-home/company/index.html
- Bender RBT 2.1.264 0616b software by Bender RBT Inc. http://www.benderrbt.com/index.htm
- Hexawise software by Justin Hunter http://hexawise.com/
- ACTS Beta 2 - Revision 1.3 Release software by the Automated Combinatorial Testing for Software group at NIST and UT-Automation http://csrc.nist.gov/groups/SNS/acts/index.html