BenderRBT Test Case Design Product Overview

Richard Bender
Bender RBT Inc.
17 Cardinale Lane
Queensbury, NY 12804
Phone: 518-743-8755
rbender@BenderRBT.com
www.BenderRBT.com

Test Case Design Challenges

- Testing is comparing an expected result to the observed result – implies clear specifications
- The number of potential tests exceeds the number of molecules in the universe
- Did you get the right answer for the right reason

Test Case Design Challenge

- Make the big number a small number:
  - If you have **just 6 variables** and they have only two states each and then factor in all of the unique orders then:

$$2^6! = 64! = 1.27 * 10^{89}$$

- Did you get the right answer for the right reason
  - Two or more defects may sometimes cancel each other out
  - Something going right can hide something going wrong

Information Needed to Design Test Cases

- Identify all of the variables
- Resolve aliases within/across processes
- Identify the possible states of the variables
  - Both positive and negative states
- Know which variables are mandatory versus optional
- Identify all of the preconditions
  - Based on the physical structure of the data
  - Based on the post conditions of prior functions

Information Needed to Design Test Cases

- Understand the precedence relationships
- Understand concurrency
- Know which variables are observable
- Identify implicit information and get it clarified
- Identify the transforms
- Identify the expected results

Requirements Based Testing Process

- VALIDATE That The Requirements Are:
  - Correct
  - Complete
  - Unambiguous
  - Logically Consistent

- Design Sufficient Tests To VERIFY That The Design And Code Correctly Implement The Requirements

Requirements-Based Testing

Quality Filters

1. Validate requirements against objectives.
2. Apply scenarios against requirements.
3. Perform initial Ambiguity Review.
4. Perform domain expert reviews.
5. Create cause-effect graph.
6. Logical consistency check and test cases designed by RBT.
7. Review test cases with requirements authors.
8. Validate test cases with users/domain experts.
9. Review test cases with developers.
10. Walk test cases through design.
11. Walk test cases through code.
12. Verify code against test cases designed from the requirements.

Designing Test Cases

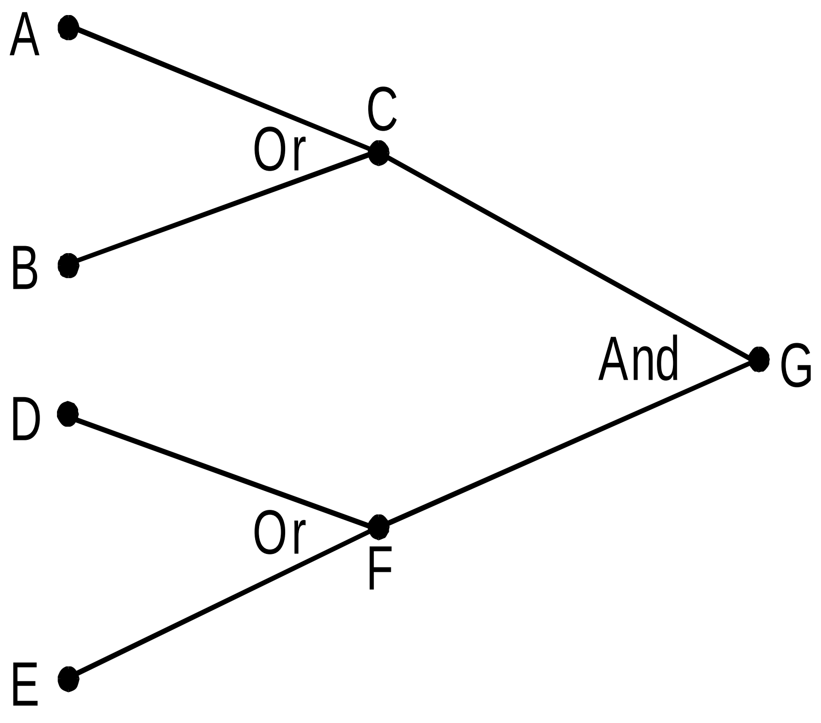
- Software has 5 defects per thousand lines of code at delivery.
- Hardware has less than 1 defect per many billions of logic gates at delivery.

Challenge:
How do we apply hardware logic testing to software?

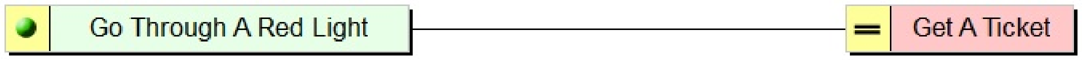
Cause-Effect Graphing

If A or B, then C.
If D or E, then F.
If C and F, then G.

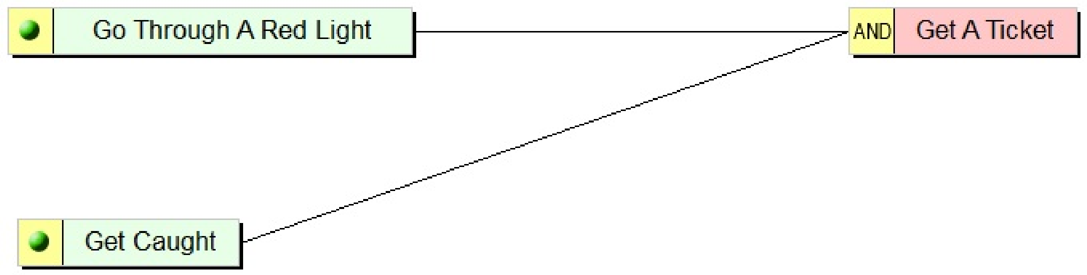
- Resolve Aliases
- Clarify Precedence Rules
- Clarifies Implicit Information
- Begin Integration Test



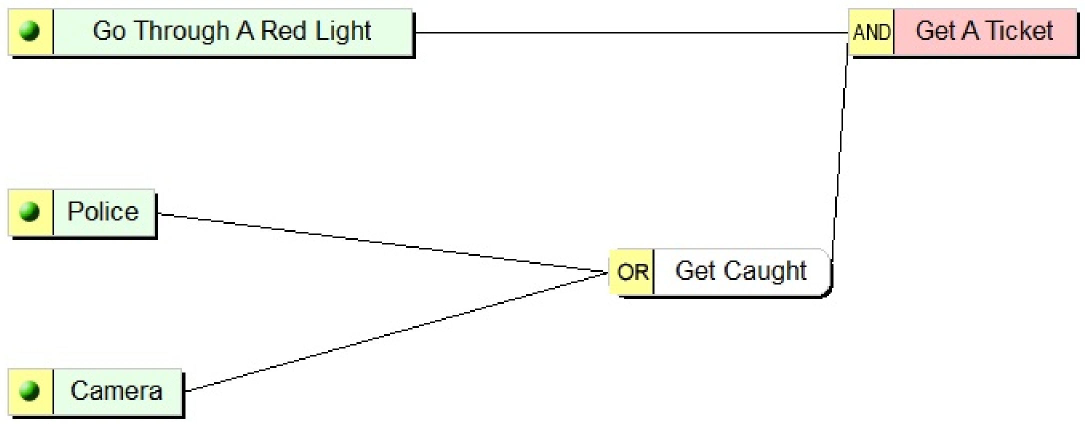
Clarifying Requirements Via Cause-Effect Graphing



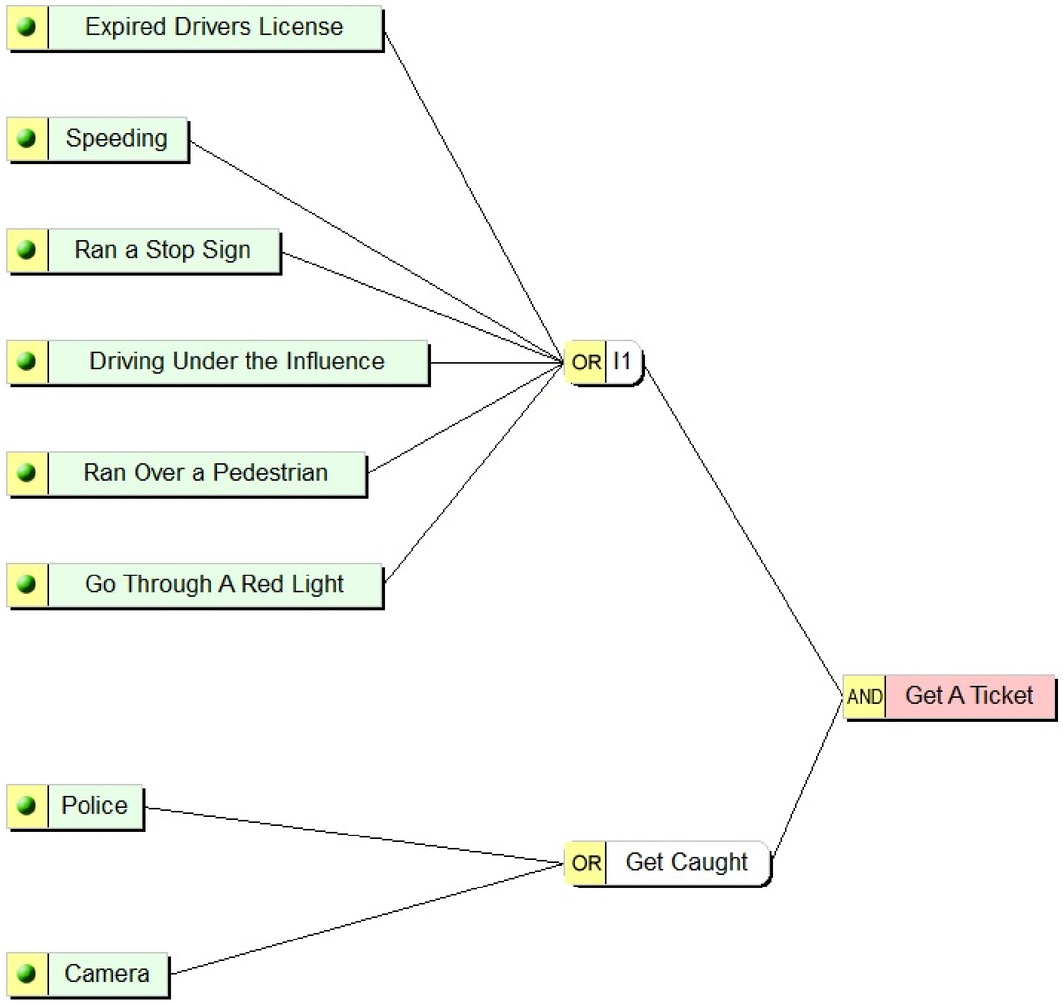
Clarifying Requirements Via Cause-Effect Graphing



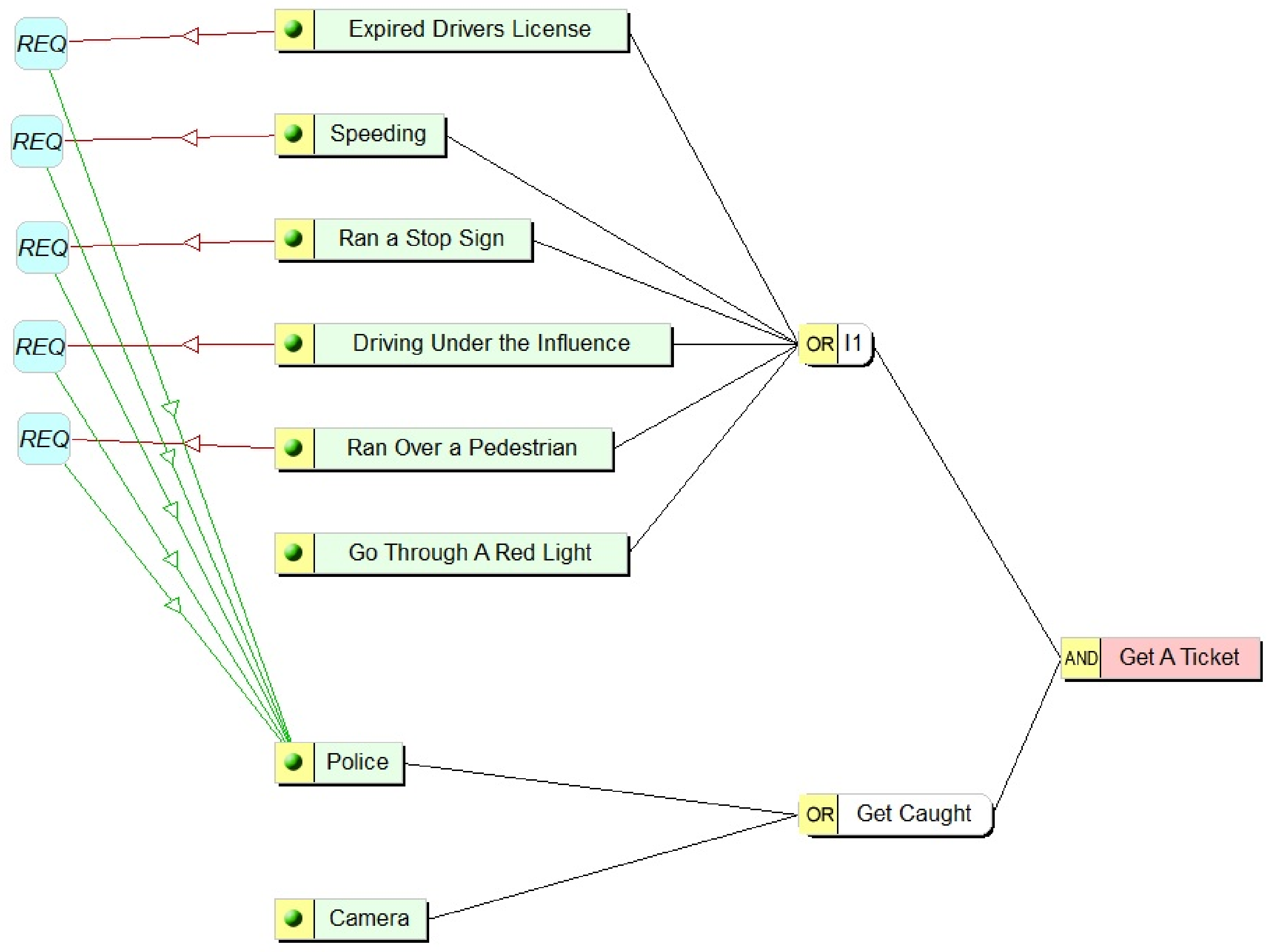
Clarifying Requirements Via Cause-Effect Graphing



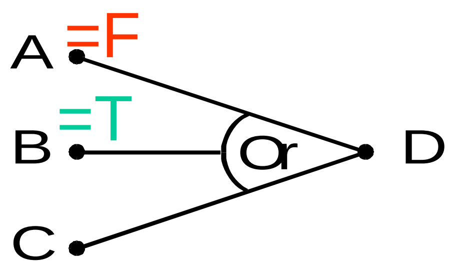
Clarifying Requirements Via Cause-Effect Graphing



Clarifying Requirements Via Cause-Effect Graphing



Cause-Effect Graphing

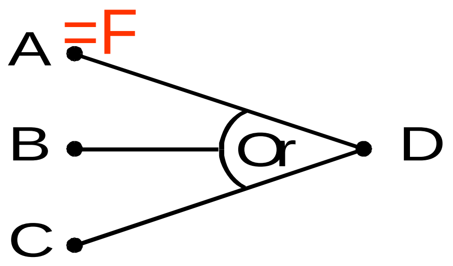


1.	A	—	—	D
2.	—	B	—	D
3.	—	—	C	D
4.	—	—	—	—

Assume A is stuck at FALSE and B is stuck at TRUE.
The machine would interpret:

1.	A	—	—	as	—	B	—	D
2.	—	B	—	as	—	B	—	D
3.	—	—	C	as	—	B	C	D
~~4.~~	—	—	—	as	—	B	—	Ⓓ

Cause-Effect Graphing



1.	A	—	—	D
2.	—	B	—	D
3.	—	—	C	D
4.	—	—	—	—

Assume A is still stuck at FALSE.
The machine would interpret:

~~1.~~ X	A — —	as	—	—	—	⊖
2.	— B —	as	—	B	—	D
3.	— — C	as	—	—	C	D
4.	— — —	as	—	—	—	—

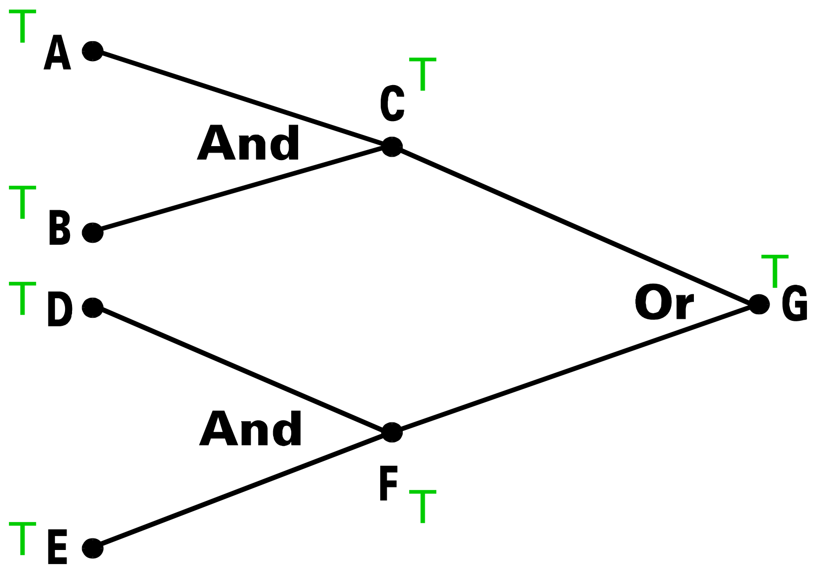
Fix the bug found by #4 and #1 fails.

Must rerun ALL tests until ALL pass!

Cause-Effect Graphing

Observable Events and Path Sensitizing

- Assume C and F are not observable events.
- Assume A is stuck at FALSE.
- Enter as a test case A(T), B(T), D(T), E(T).
- Results should be C(T), F(T) and G(T).

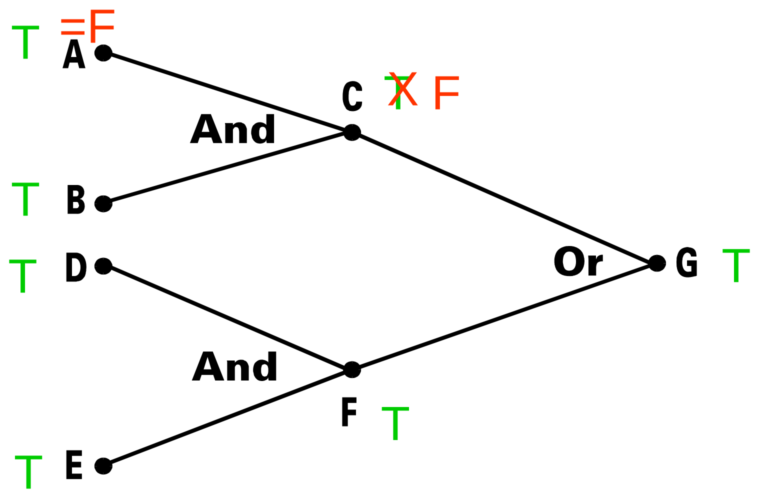


RBT24261

Cause-Effect Graphing

Observable Events and Path Sensitizing

- Results should be C(T), F(T) and G(T).
- A, stuck at FALSE, causes C to be (F).
- The error is not detected since G is still (T) due to F(T).
- Therefore, no test of C can be combined with tests of F which would result in F(T).



Test Design Challenge

Challenge:

- Design a set of test cases, factoring in:
  - The relations between the variables
  - Constraints between the data attributes
  - Functional variations required to test
  - Node observability

… such that if any logical defect or any combination of defects are present, at least one test case will fail at an observable point.

Cause-Effect Graphing
Observable Events and Path Sensitizing

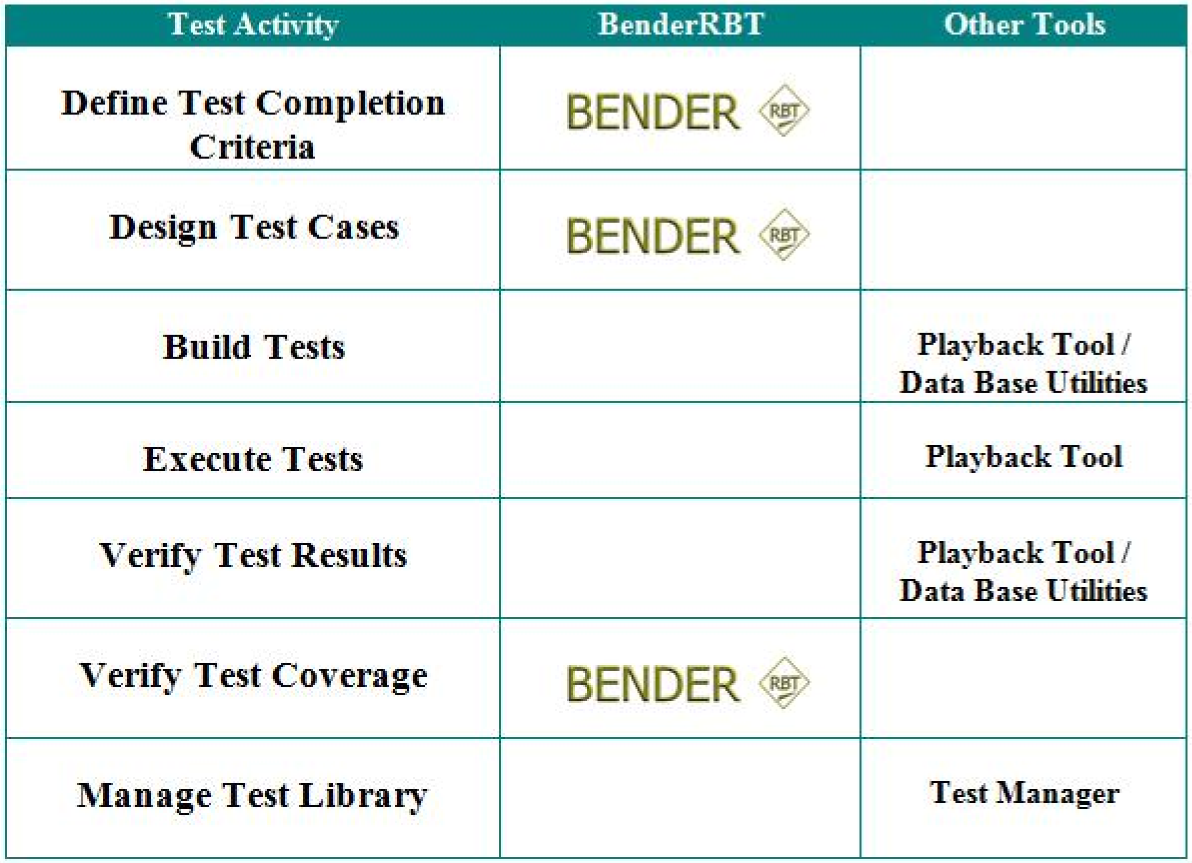
Diagnostic probe points

- RESULT:
  - Some functional variations still untestable
- SOLUTION:
  - Diagnostic probe points
  - I.E., force normally unobservable nodes to be observable.

BenderRBT Test Case Design Tool

- Validates Functional Requirements
- Automates Test Case Design
- Rigorous Algorithm
- Visual Test Case Design Tool

BenderRBT Test Case Design

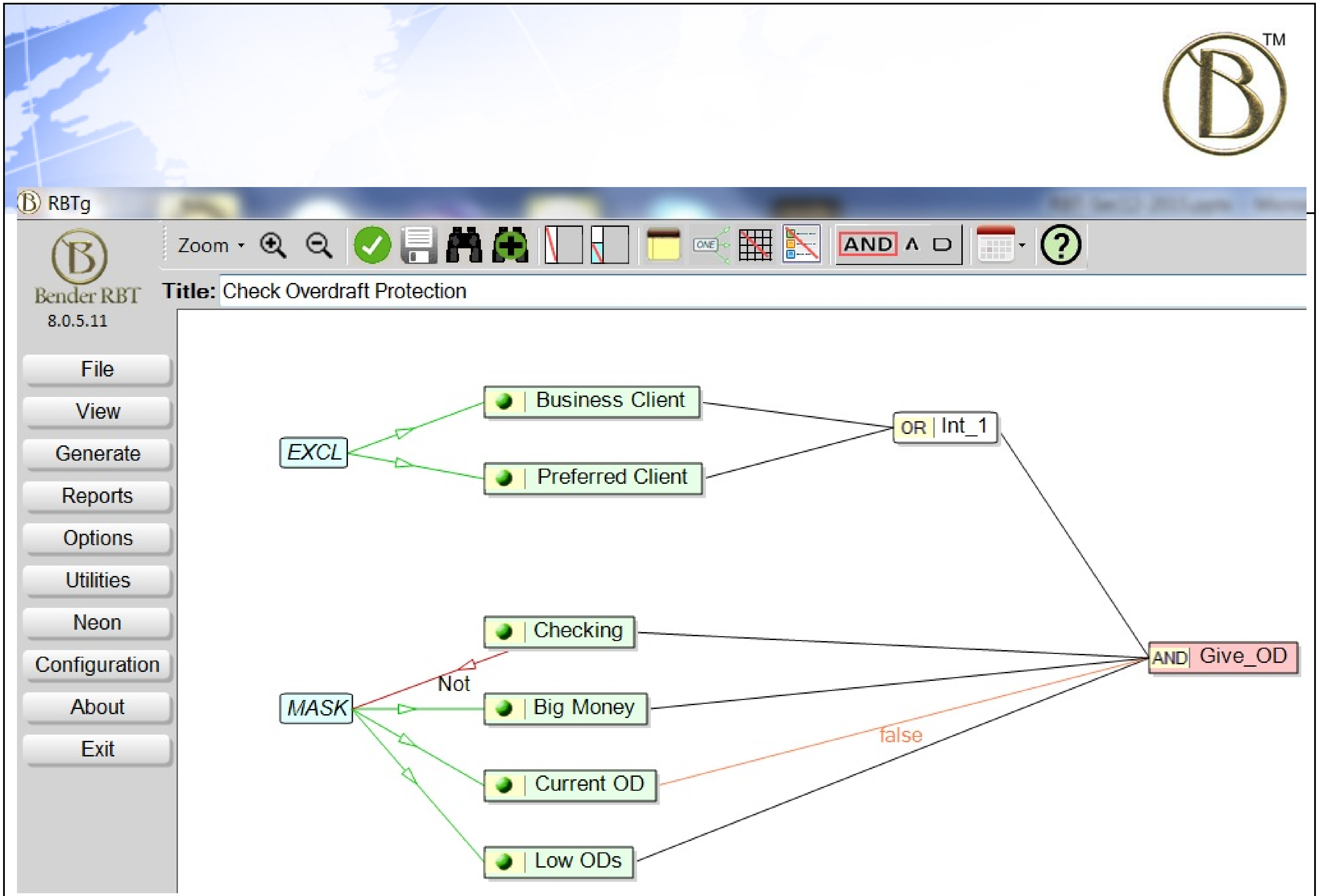


Test Activity	BenderRBT	Other Tools
Define Test Completion Criteria	BENDER RBT	
Design Test Cases	BENDER RBT	
Build Tests		Playback Tool / Data Base Utilities
Execute Tests		Playback Tool
Verify Test Results		Playback Tool / Data Base Utilities
Verify Test Coverage	BENDER RBT	
Manage Test Library		Test Manager

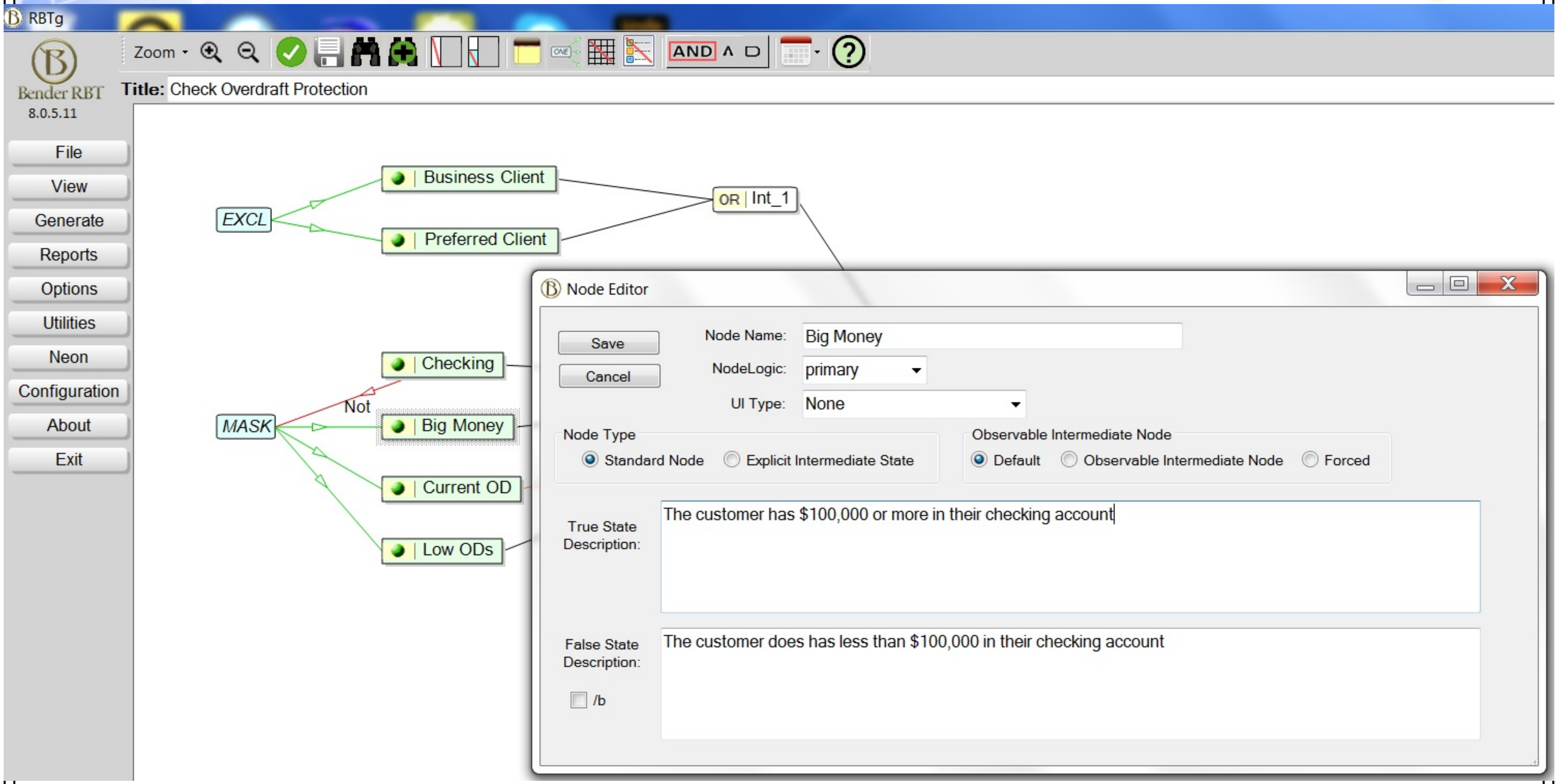
Designing Test Cases

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

If the customer is a business client or a preferred personal client and they have a checking account, $100,000 or more in deposits, no overdraft protection and fewer than 5 overdrafts in the last 12 months, set up free overdraft protection. Else, do not give overdraft protection.



Title: Check Overdraft Protection



Title: Check Overdraft Protection

Node Name: Big Money
NodeLogic: primary
UI Type: None

Node Type: [✓ Standard Node] Explicit Intermediate State

Observable Intermediate Node: [✓ Default] Observable Intermediate Node Forced

True State Description: The customer has $100,000 or more in their checking account

False State Description: The customer does has less than $100,000 in their checking account

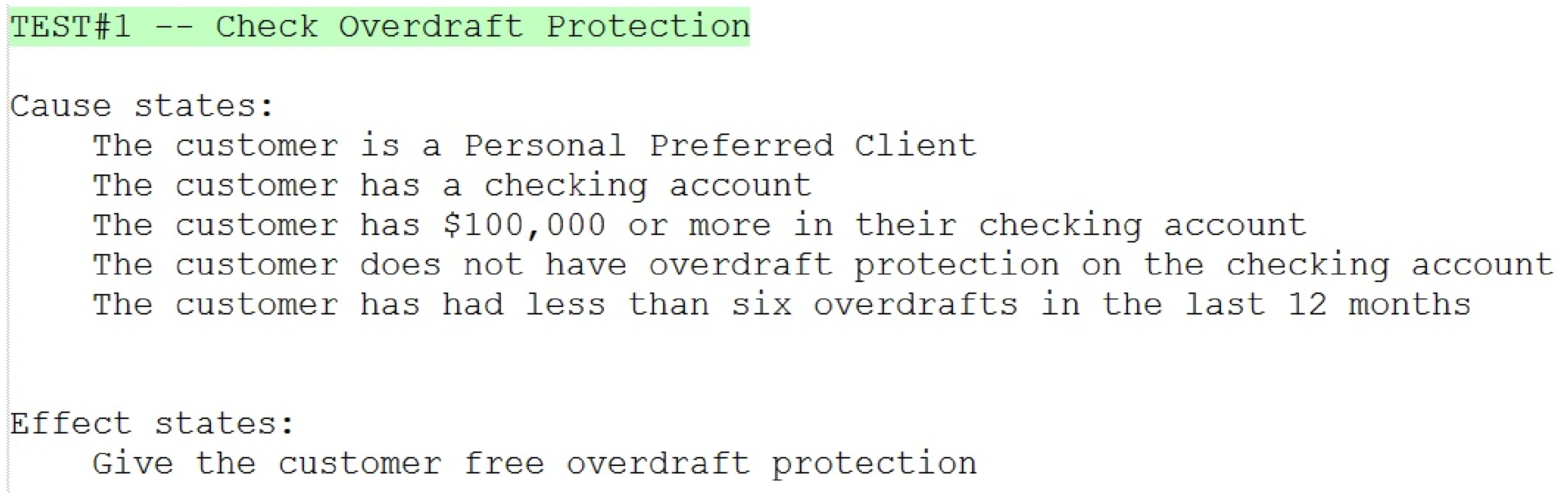
☐ /b

RBT then identifies the system's FUNCTIONAL VARIATIONS.

```
Functional Variations
Check Overdraft Protection
Run:  Synthesis of New Tests

Functional Variations for:
Int_1:-Business Client OR Preferred Client.
       1. If not Business Client and not Preferred Client
          then not Int_1.
       2. If Business Client
            (and not Preferred Client)
          then Int_1.
       3. If Preferred Client
            (and not Business Client)
          then Int_1.

Functional Variations for:
Give_OD:-Int_1 AND Checking AND Big Money AND not Current OD AND Low
     ODs.
       4. If Int_1 and Checking and Big Money and not Current OD and
     Low ODs
          then Give_OD.
       5. If not Int_1
            (and Checking and Big Money and not Current OD and Low
     ODs)
          then not Give_OD.
       6. If not Checking
            (and Int_1 and Big Money Masked and not Current OD
     Masked and Low ODs Masked)
          then not Give_OD.
```

```
 7. If not Big Money
            (and Int_1 and Checking and not Current OD and
    Low ODs)
          then not Give_OD.
       8. If Current OD
            (and Int_1 and Checking and Big Money and Low
    ODs)
          then not Give_OD.
       9. If not Low ODs
            (and Int_1 and Checking and Big Money and not
    Current OD)
          then not Give_OD.


Number of infeasible variations:   0
Number of untestable variations:   0
Maximum time to create a test is 1 seconds
Skip Time is 60 seconds
```

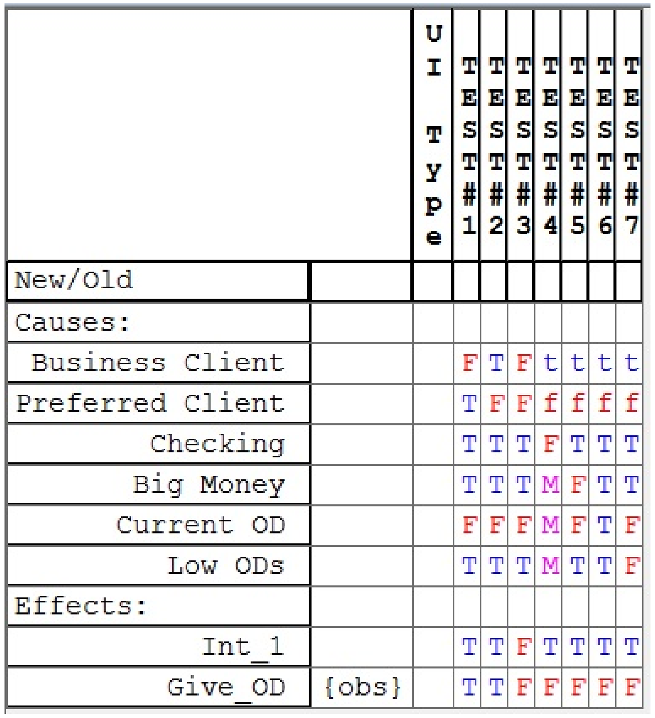
Next, RBT takes the FUNCTIONAL VARIATIONS and packages them into a complete set of TEST CASES.



```
TEST#1 -- Check Overdraft Protection

Cause states:
    The customer is a Personal Preferred Client
    The customer has a checking account
    The customer has $100,000 or more in their checking account
    The customer does not have overdraft protection on the checking account
    The customer has had less than six overdrafts in the last 12 months


Effect states:
    Give the customer free overdraft protection
```

Automatic Check for Overdraft Protection

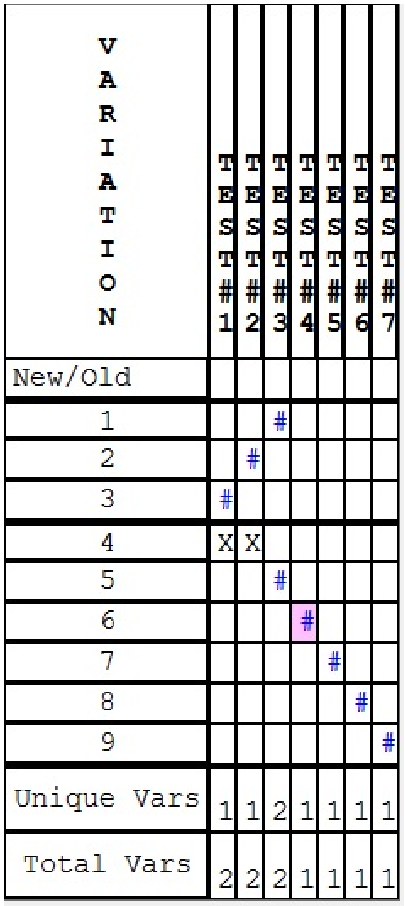
Definition Matrix



		UI Type	TEST #1	TEST #2	TEST #3	TEST #4	TEST #5	TEST #6	TEST #7
New/Old									
Causes:									
Business Client			F	T	F	t	t	t	t
Preferred Client			T	F	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

To track the testing process, RBT produces two test matrices and an assessment of the total test coverage.

Coverage Matrix



VARIATION	TEST #1	TEST #2	TEST #3	TEST #4	TEST #5	TEST #6	TEST #7
New/Old							
1			#				
2		#					
3	#						
4	X	X					
5			#				
6				#			
7					#		
8						#	
9							#
Unique Vars	1	1	2	1	1	1	1
Total Vars	2	2	2	1	1	1	1

```
Check Overdraft Protection
Run:  Synthesis of New Tests
Input Graph Filename:  C:\CEGRAPH\Cause Effect Graphing Examples - 8\Check-OD.rbt
Input Last Modified:    1 May 2015 @ 12:09

Design Tests Last Run:  14 May 2015 @ 10:31
BenderRBT Release:        8.0(443)

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 BenderRBT is:  7
The test case compression ratio is:
   (2^6)/7 = 9 : 1

Number of tested variations:       9
The tested variations to test case compression ratio is:
   9/7 = 1 : 1

BenderRBT Elapsed Time =  00:00:01  (hh:mm:ss)
```

Summary statistics are also produced to aid in project estimating and tracking.

Test Statistics For A Typical Screen

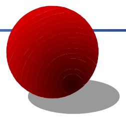
For n = 37 Primary causes, then
2^n = [a little more than] 137,438,953,472
THEORETICAL Maximum Number of Test Cases.

RBT generated 22 Test Cases, which yields a
6,247,225,157 to 1 Test Case Compression
Ratio.

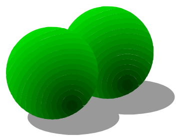
RBT Elapsed Time: 00:00:01 (hh:mm:ss)

Test Statistics

Thought Experiment



- Put 137,438,953,450 red balls in a giant barrel.
- Add 22 green balls to the barrel 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?

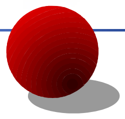
- Pull out 1,000,000 balls

What is the probability that you have the 22 green ones now?

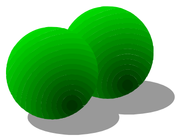
This is what "GUT FEEL" testing really is.

Test Statistics

Thought Experiment



- Put 137,438,953,450 red balls in a giant barrel.
- Add 22 green balls to the barrel 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 $7.3\text{X}10^{-180}$

What is the probability that you have the 22 green ones now?

- Pull out 1,000,000 balls $9.2\text{X}10^{-114}$

What is the probability that you have the 22 green ones now?

This is what "GUT FEEL" testing really is.

Test Statistics For A Large Problem

Input Graph Filename: Interface RBCDL0002 & RBCDL0013
(Gary Mogyorodi consulting to Royal Bank of Canada – Mortgage Processing)

Number of Primary Causes: 437
The THEORETICAL maximum number of test cases is:
2^437 =
354,901,720,847,464,300,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000

The number of test cases generated by RBT is: 96

RBT Elapsed Time = 00:00:27(hh:mm:ss)

This Requirement...

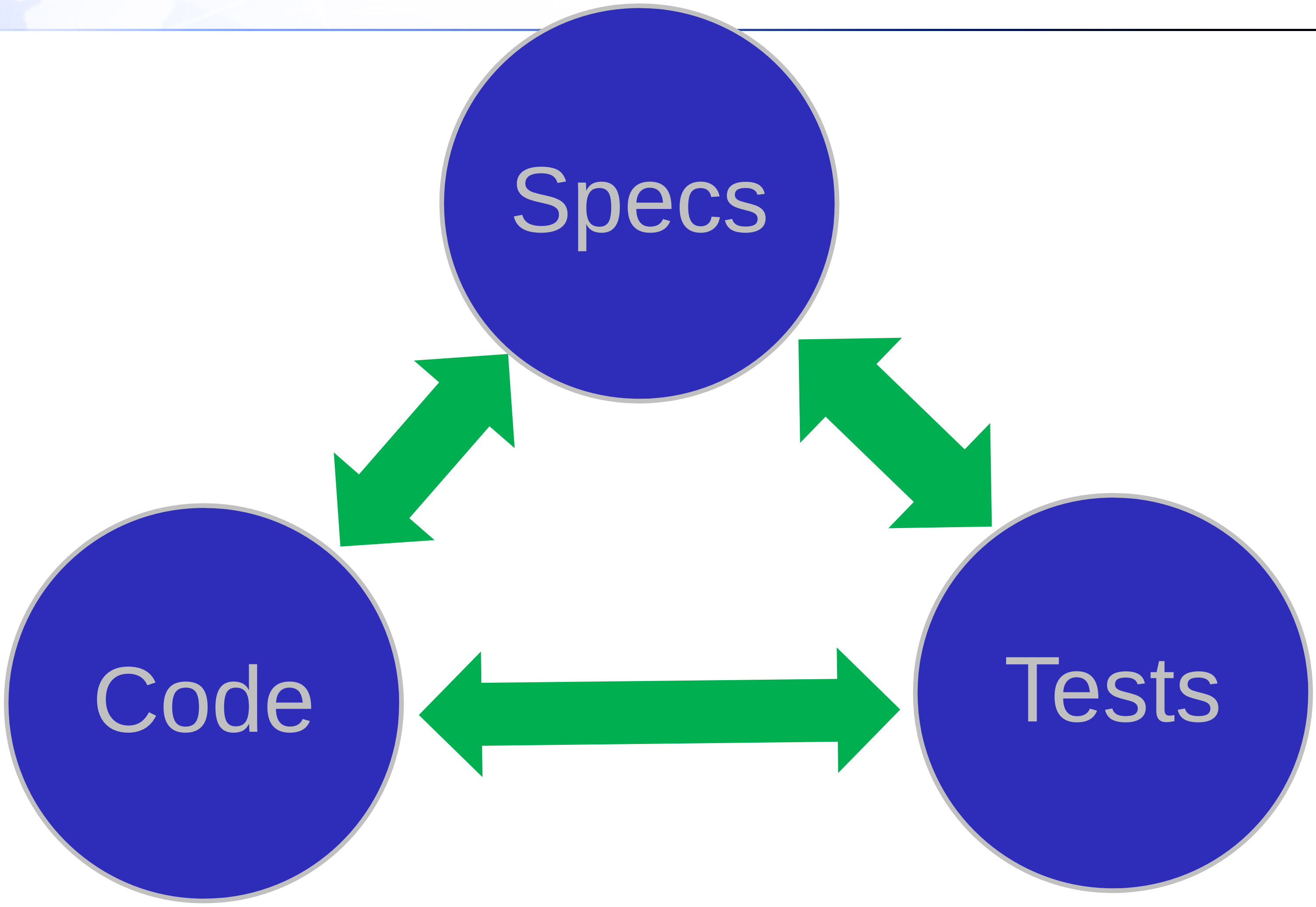
Dentists with membership codes of 2, 3, or 9 are member dentists. For claims referencing a non-member dentist or for procedures not within the referenced dentist's record, a system table is used to calculate the amount paid. Otherwise the amount submitted is paid. However, an override code of 1 or 9 allows the amount submitted to be paid for non-member dentists or for procedures not within the referenced dentist's record. When an override code is used an entry is made on the paid claims report.

…can be rewritten by RBT.

```
1. IF The member is a full member
     OR The member is an associate member
     OR The member is a temporary member
   THEN This is a member dentist
   ELSE This is a non-member dentist.

2. IF This is a member dentist
     AND The procedure was preauthorized
   THEN This is a valid procedure for the
     member dentist.

3. IF This is a member dentist
     AND The procedure was not preauthorized
   THEN This is not a valid procedure for the
     member dentist.

4. IF [This is a non-member dentist]
     OR This is not a valid procedure for the
     member dentist
   THEN This is a potential partial payment
     situation.

5. IF

   THEN The override code was accepted
   ELSE No override code was entered.

6. IF This is a potential partial payment situation
     AND The override code was accepted
   THEN Override the partial payment.

7. IF This is a valid procedure for the member dentist
     OR Override the partial payment
   THEN Pay the full amount of the claim.

8. IF This is a potential partial payment situation
     AND No override code was entered
   THEN Make a partial payment of the claim based
        on the system table.

9. IF Override the partial payment
   THEN Make an entry on the paid claims report
   ELSE Do not make an entry on the paid claims
        report.
```

RBT can also create the “as built” specification.

“As Delivered”

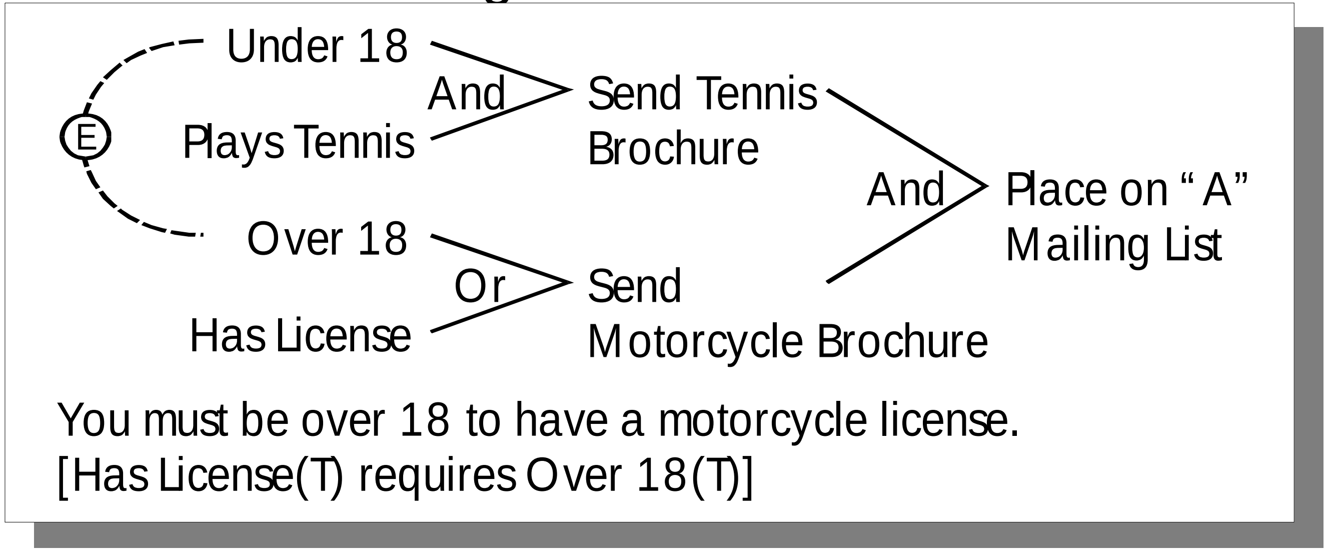


RBT Validates the Consistency of FUNCTIONAL REQUIREMENTS

If the person is under 18, and plays tennis,
then send them a tennis club brochure.

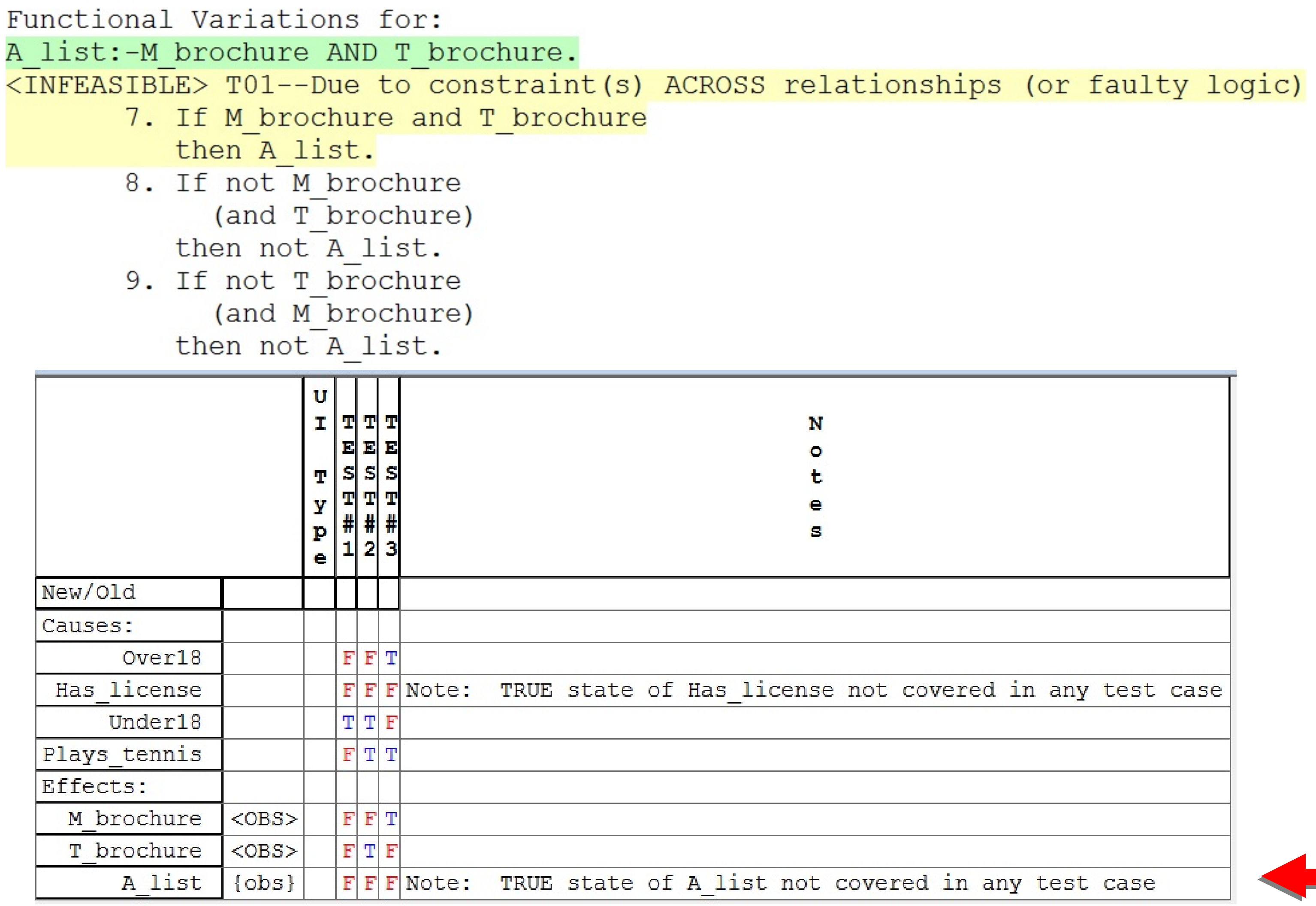
If the person is 18 or older, or has a motorcycle license,
then send them a motorcycle club brochure.

If the person was sent both brochures, then put them
on the “A” mailing list.



You must be over 18 to have a motorcycle license.
[Has License(T) requires Over 18(T)]

RBT Identifies Logic Errors.

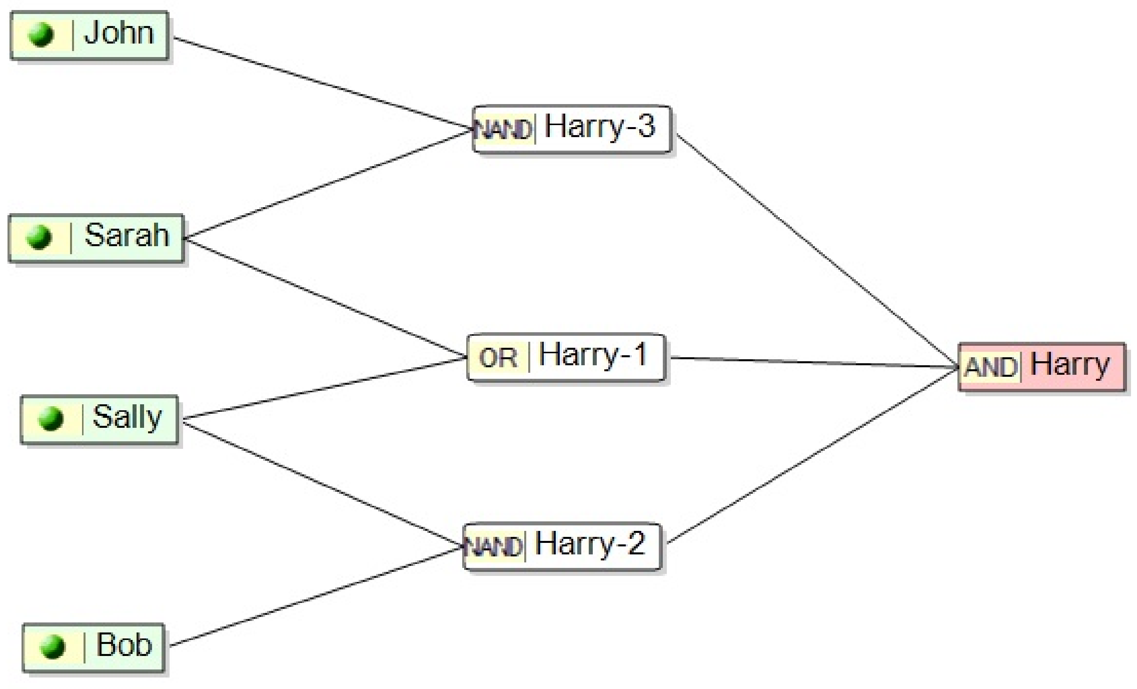


```
Functional Variations for:
A_list:-M_brochure AND T_brochure.
<INFEASIBLE> T01--Due to constraint(s) ACROSS relationships (or faulty logic)
       7. If M_brochure and T_brochure
          then A_list.
       8. If not M_brochure
              (and T_brochure)
          then not A_list.
       9. If not T_brochure
              (and M_brochure)
          then not A_list.
```

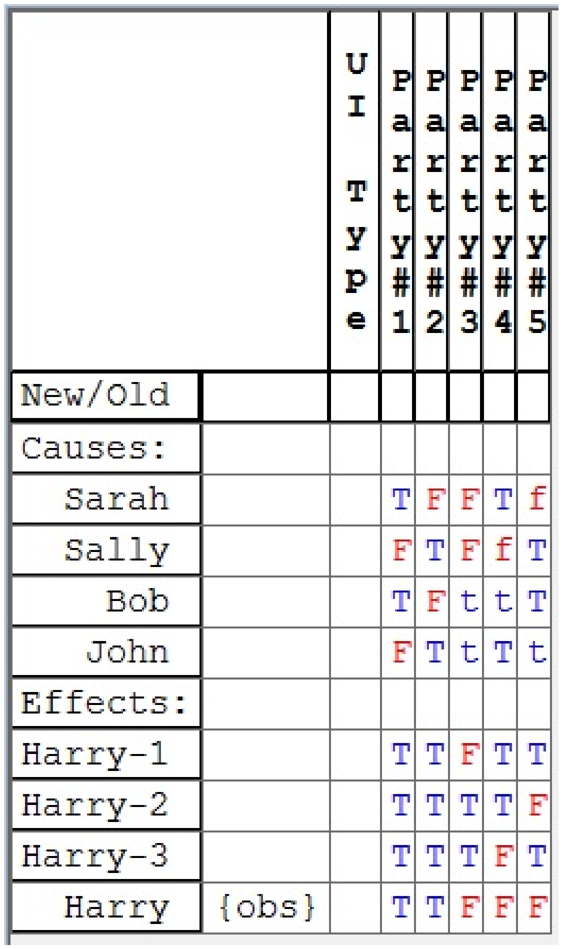
		UI Type	TEST #1	TEST #2	TEST #3	Notes
New/Old						
Causes:						
Over18			F	F	T	
Has_license			F	F	F	Note: TRUE state of Has_license not covered in any test case
Under18			T	T	F	
Plays_tennis			F	T	T	
Effects:						
M_brochure	<OBS>		F	F	T	
T_brochure	<OBS>		F	T	F	
A_list	{obs}		F	F	F	Note: TRUE state of A_list not covered in any test case

RBT Protects Your Investment in Previously Built Tests

The original graph



RBT Protects Your Investment in Previously Built Tests

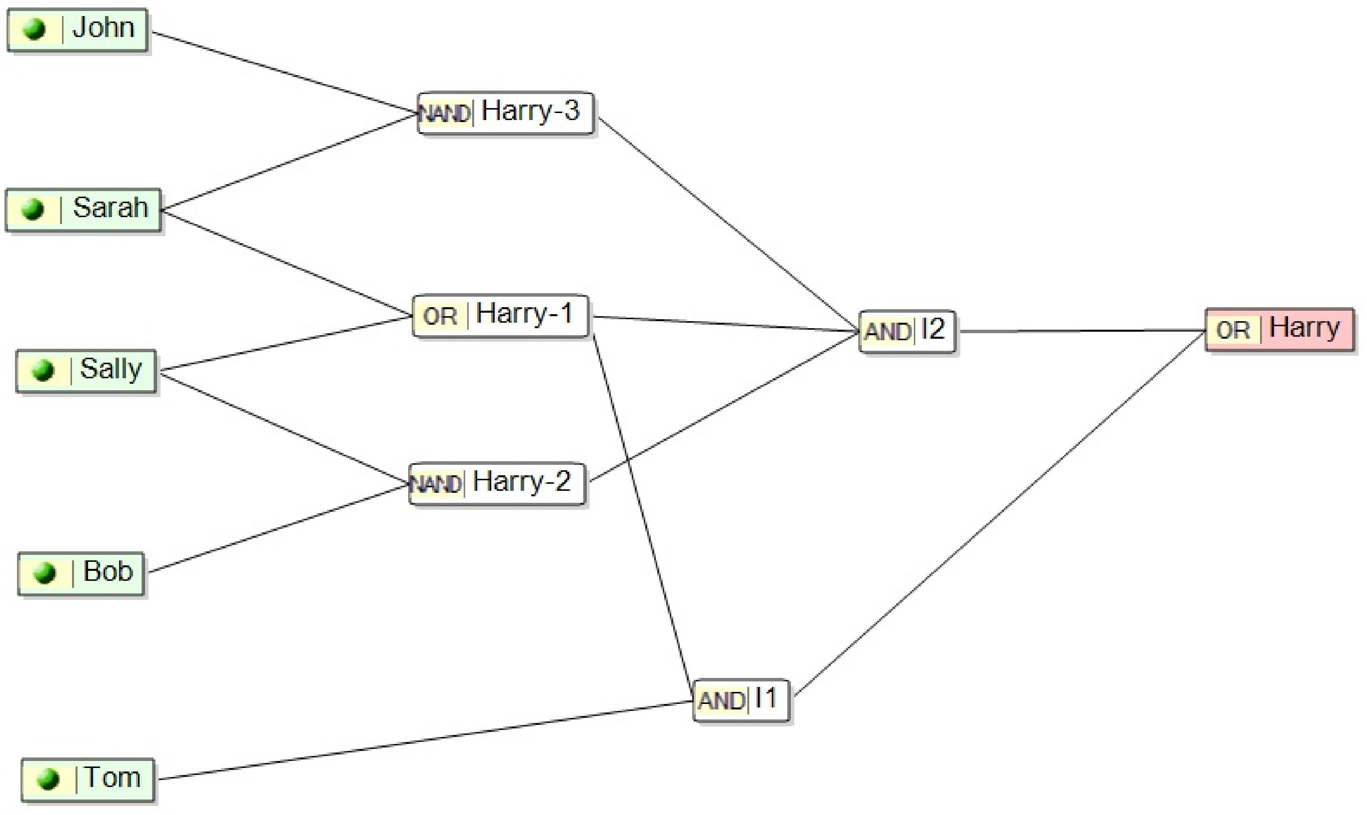


		UI Type	Party #1	Party #2	Party #3	Party #4	Party #5
New/Old							
Causes:							
Sarah			T	F	F	T	f
Sally			F	T	F	f	T
Bob			T	F	t	t	T
John			F	T	t	T	t
Effects:							
Harry-1			T	T	F	T	T
Harry-2			T	T	T	T	F
Harry-3			T	T	T	F	T
Harry	{obs}		T	T	F	F	F

The original tests

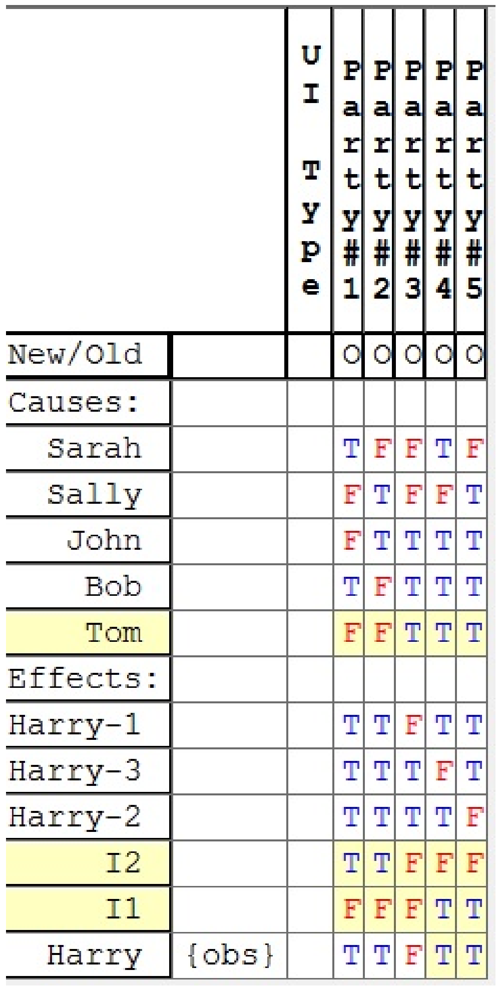
RBT Protects Your Investment in Previously Built Tests

The updated graph



RBT Protects Your Investment in Previously Built Tests

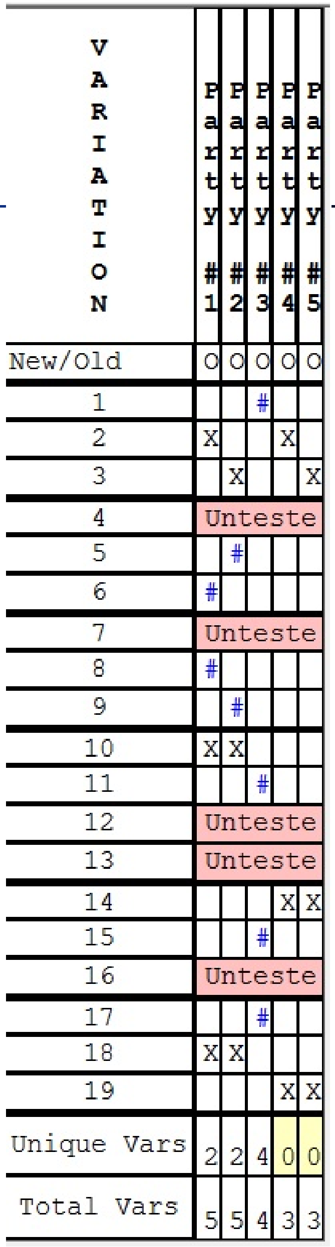
RBT adjusts the old tests to maximize coverage



		UI Type	Party#1	Party#2	Party#3	Party#4	Party#5
New/Old			O	O	O	O	O
Causes:							
Sarah			T	F	F	T	F
Sally			F	T	F	F	T
John			F	T	T	T	T
Bob			T	F	T	T	T
Tom			F	F	T	T	T
Effects:							
Harry-1			T	T	F	T	T
Harry-3			T	T	T	F	T
Harry-2			T	T	T	T	F
I2			T	T	F	F	F
I1			F	F	F	T	T
Harry	{obs}		T	T	F	T	T

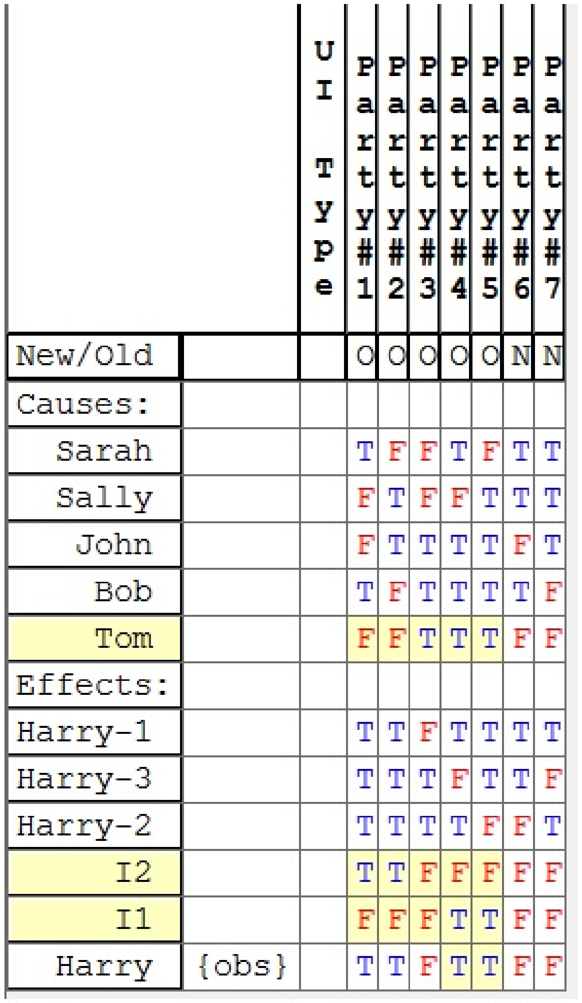
RBT Protects Your Investment in Previously Built Tests

RBT identifies untested variations



VARIATION	Party #1	Party #2	Party #3	Party #4	Party #5
New/Old	O	O	O	O	O
1			#		
2	X			X	
3		X			X
4	Unteste				
5		#			
6	#				
7	Unteste				
8	#				
9		#			
10	X	X			
11			#		
12	Unteste				
13	Unteste				
14				X	X
15			#		
16	Unteste				
17			#		
18	X	X			
19				X	X
Unique Vars	2	2	4	0	0
Total Vars	5	5	4	3	3

RBT Protects Your Investment in Previously Built Tests



		UI Type	Party#1	Party#2	Party#3	Party#4	Party#5	Party#6	Party#7
New/Old			O	O	O	O	O	N	N
Causes:									
Sarah			T	F	F	T	F	T	T
Sally			F	T	F	F	T	T	T
John			F	T	T	T	T	F	T
Bob			T	F	T	T	T	T	F
Tom			F	F	T	T	T	F	F
Effects:									
Harry-1			T	T	F	T	T	T	T
Harry-3			T	T	T	F	T	T	F
Harry-2			T	T	T	T	F	F	T
I2			T	T	F	F	F	F	F
I1			F	F	F	T	T	F	F
Harry	{obs}		T	T	F	T	T	F	F

RBT then supplements the old test library as needed

VARIATION	TEST#01	TEST#02	TEST#03	TEST#04	TEST#05	TEST#06	TEST#07	TEST#08	TEST#09	TEST#10	TEST#11	TEST#12	TEST#13
1	#												
2				X							X		
3		X	X				X	X					
4					#								
5	X	X								X			
6				X			X						
7								X			X		X
8		#											
9			#										
10											#		
11				X			X	X	X				
12		X	X										
13						X						X	X
14					#								
15							X	X					
16	#												
17		X	X										
18				X	X	X	X	X			X	X	X
19	#												
20		X	X										
21				X	X	X	X	X			X	X	X
22										#			

RBT Coverage Analysis

Coverage Analysis:

Coverage = 3 of 58 = 5%.

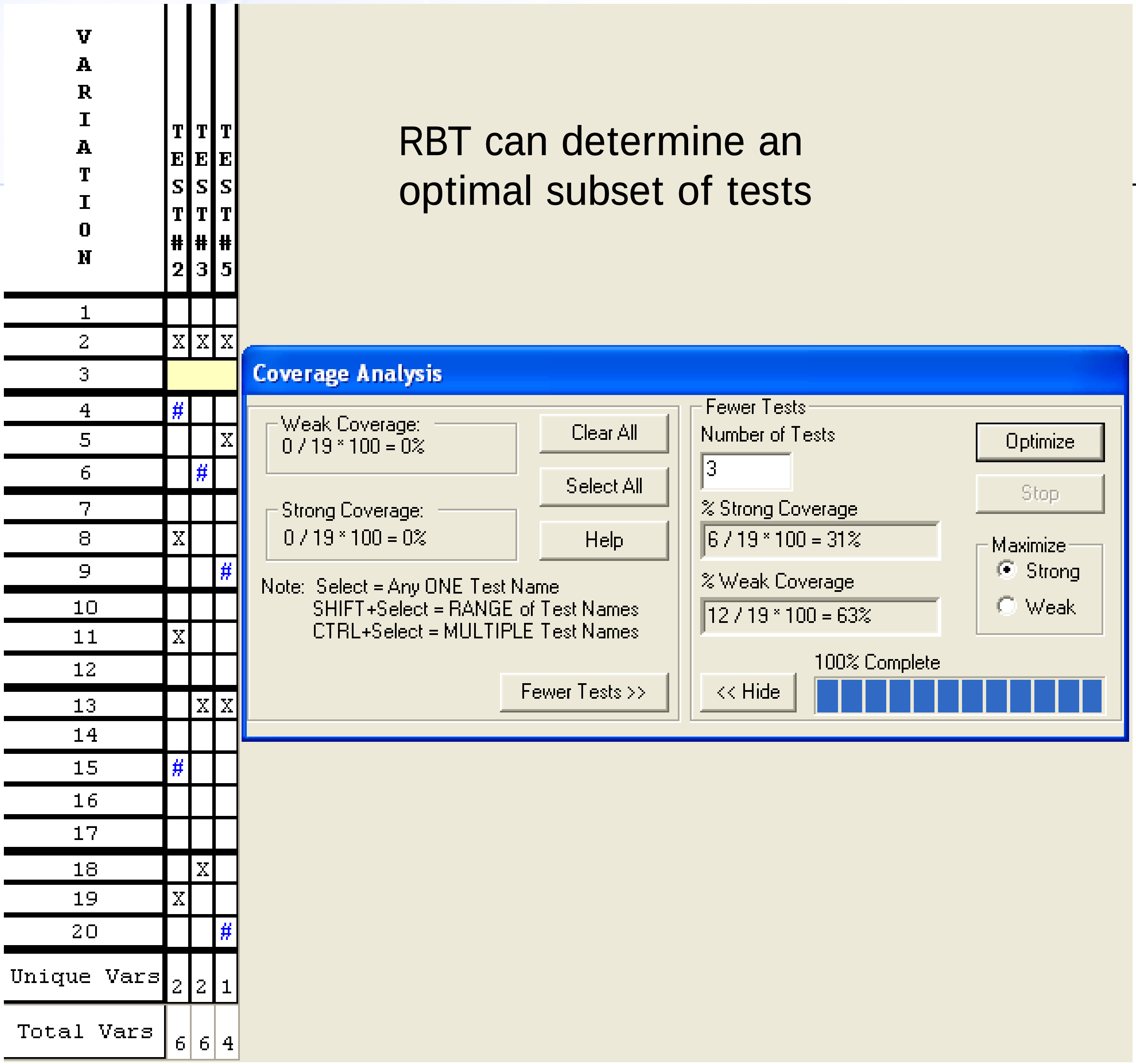
RBT Coverage Analysis

VARIATION	TEST #01	TEST #02	TEST #03	TEST #04	TEST #05	TEST #06	TEST #07	TEST #08	TEST #09	TEST #10	TEST #11	TEST #12	TEST #13
1	#												
2				X							X		
3		X	X				X	X					
4					#								
5	X	X								X			
6				X			X						
7								X			X		X
8		#											
9			#										
10											#		
11				X			X	X	X				
12		X	X										
13						X						X	X
14					#								
15							X	X					
16	#												
17		X	X										
18				X	X	X	X	X			X	X	X
19	#												
20		X	X										
21				X	X	X	X	X			X	X	X
22										#			

Coverage Analysis:

Coverage = 26 of 58 = 44%.

RBT can determine an optimal subset of tests



VARIATION	TEST #2	TEST #3	TEST #5
1			
2	X	X	X
3			
4	#		
5			X
6		#	
7			
8	X		
9			#
10			
11	X		
12			
13		X	X
14			
15	#		
16			
17			
18		X	
19	X		
20			#
Unique Vars	2	2	1
Total Vars	6	6	4

Coverage Analysis

Weak Coverage:
0 / 19 * 100 = 0%

Strong Coverage:
0 / 19 * 100 = 0%

Note: Select = Any ONE Test Name
SHIFT+Select = RANGE of Test Names
CTRL+Select = MULTIPLE Test Names

Fewer Tests

Number of Tests: 3

% Strong Coverage: 6 / 19 * 100 = 31%

% Weak Coverage: 12 / 19 * 100 = 63%

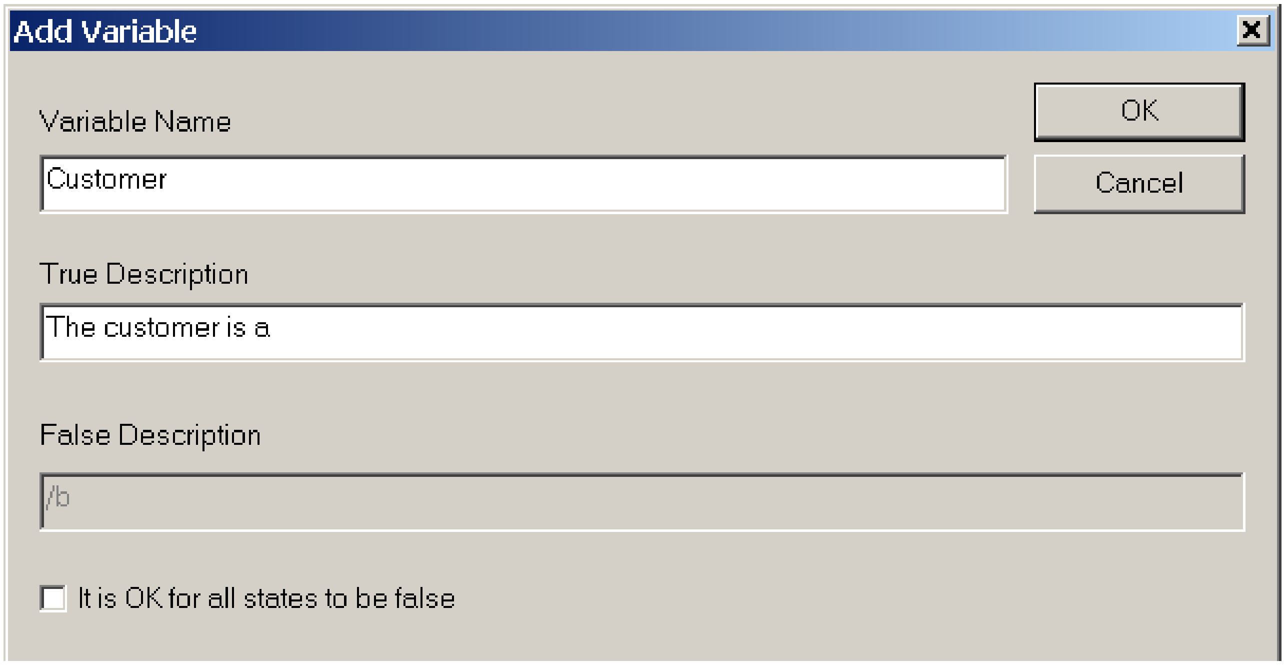
Maximize: [✓ Strong] Weak

100% Complete

Quick Design

(Supports Orthogonal and Optimized Pairs)

- Define Variables



Variable Name

Customer

True Description

The customer is a

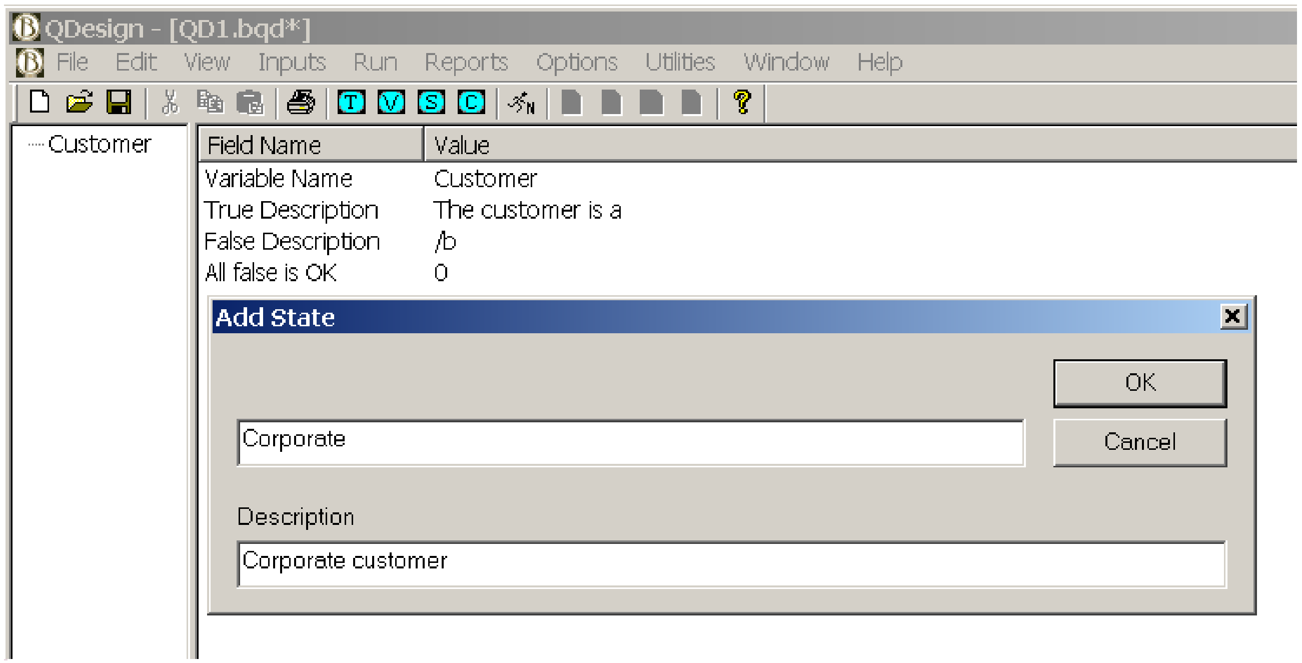
False Description

/b

☐ It is OK for all states to be false

Quick Design

- Define States


QDesign - [QD1.bqd*]
File Edit View Inputs Run Reports Options Utilities Window Help
Customer
Add State
OK
Cancel

Field Name	Value
Variable Name	Customer
True Description	The customer is a
False Description	/b
All false is OK	0

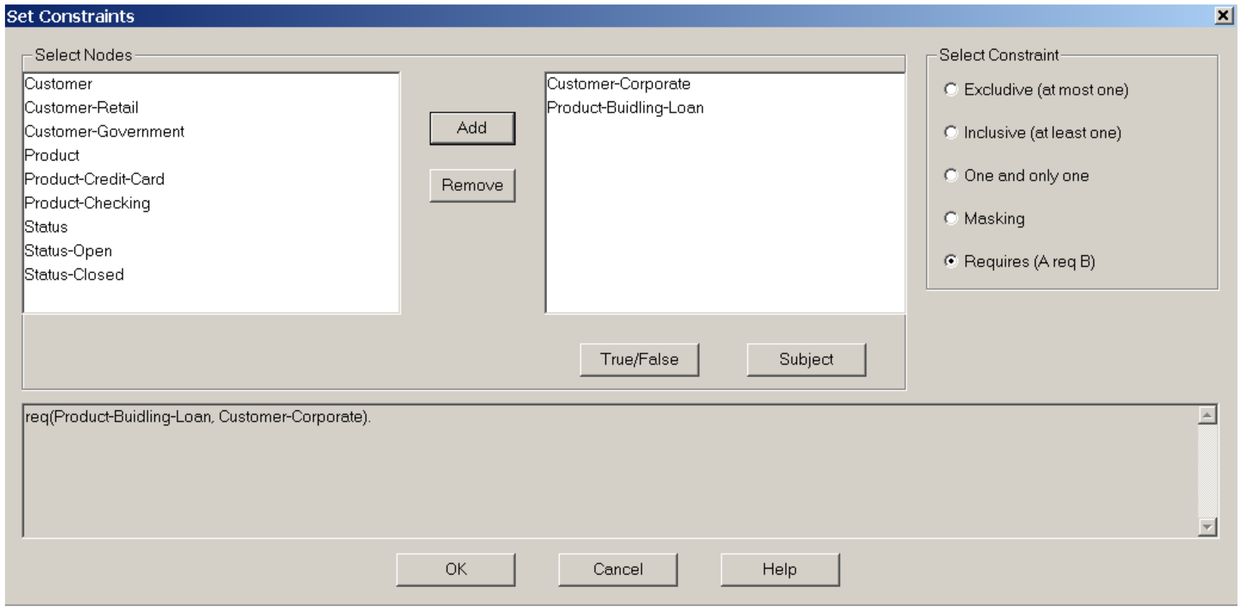
Corporate

Description

Corporate customer

Quick Design

- Define Constraints



Set Constraints

Select Nodes

Customer
Customer-Retail
Customer-Government
Product
Product-Credit-Card
Product-Checking
Status
Status-Open
Status-Closed

Customer-Corporate
Product-Buidling-Loan

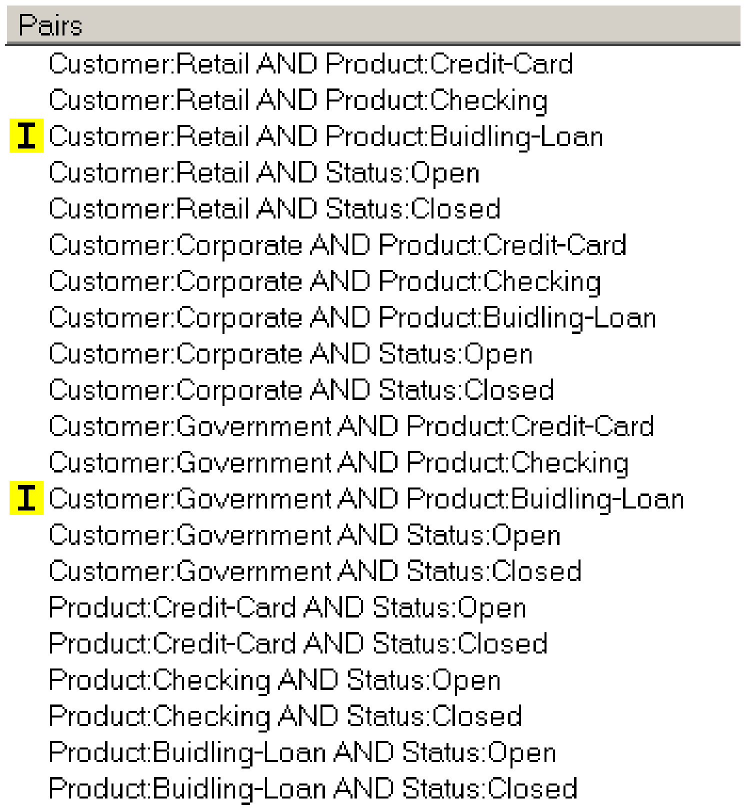
Select Constraint

( ) Excludive (at most one)
( ) Inclusive (at least one)
( ) One and only one
( ) Masking
[✓ Requires (A req B)]

req(Product-Buidling-Loan, Customer-Corporate).

Quick Design

- RBT Quick Design generates the pairs

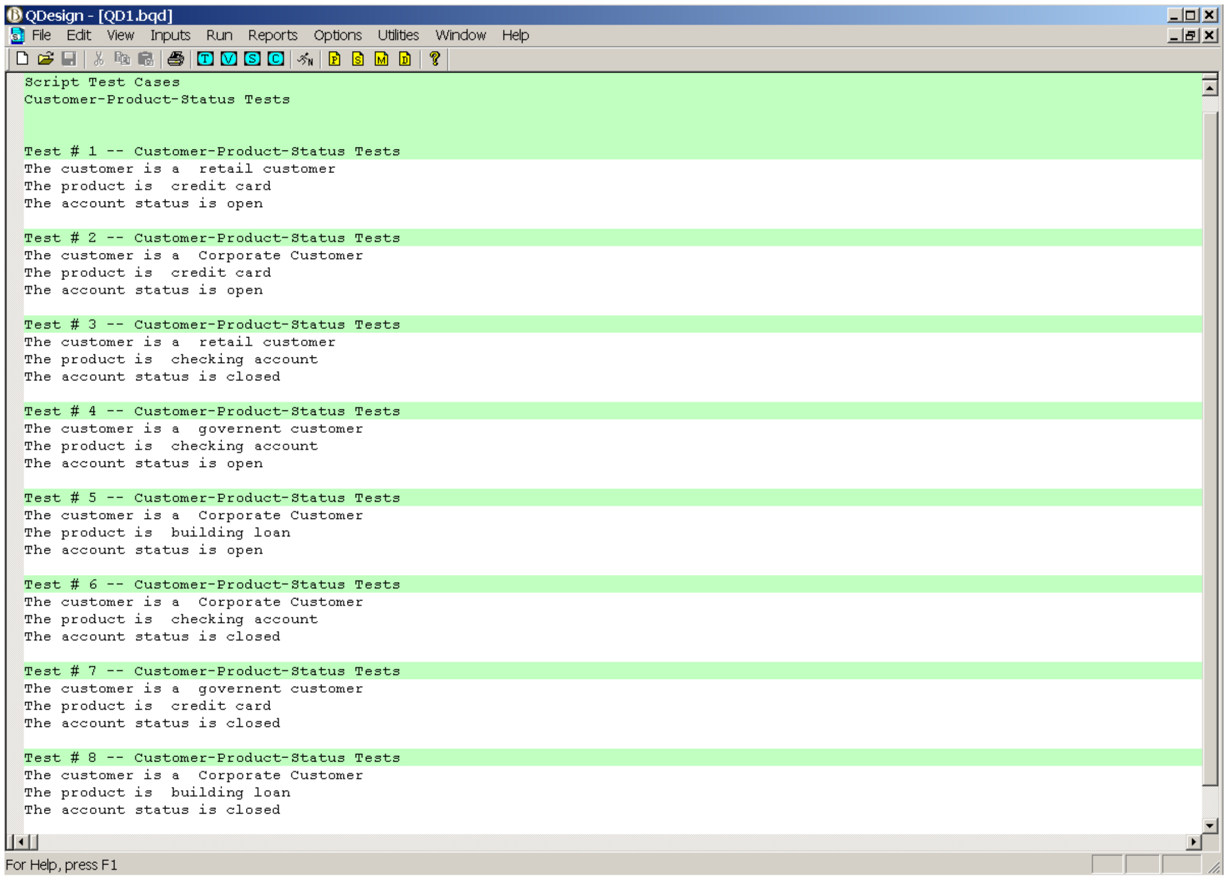


Pairs

```
  Customer:Retail AND Product:Credit-Card
  Customer:Retail AND Product:Checking
I Customer:Retail AND Product:Buidling-Loan
  Customer:Retail AND Status:Open
  Customer:Retail AND Status:Closed
  Customer:Corporate AND Product:Credit-Card
  Customer:Corporate AND Product:Checking
  Customer:Corporate AND Product:Buidling-Loan
  Customer:Corporate AND Status:Open
  Customer:Corporate AND Status:Closed
  Customer:Government AND Product:Credit-Card
  Customer:Government AND Product:Checking
I Customer:Government AND Product:Buidling-Loan
  Customer:Government AND Status:Open
  Customer:Government AND Status:Closed
  Product:Credit-Card AND Status:Open
  Product:Credit-Card AND Status:Closed
  Product:Checking AND Status:Open
  Product:Checking AND Status:Closed
  Product:Buidling-Loan AND Status:Open
  Product:Buidling-Loan AND Status:Closed
```

Quick Design

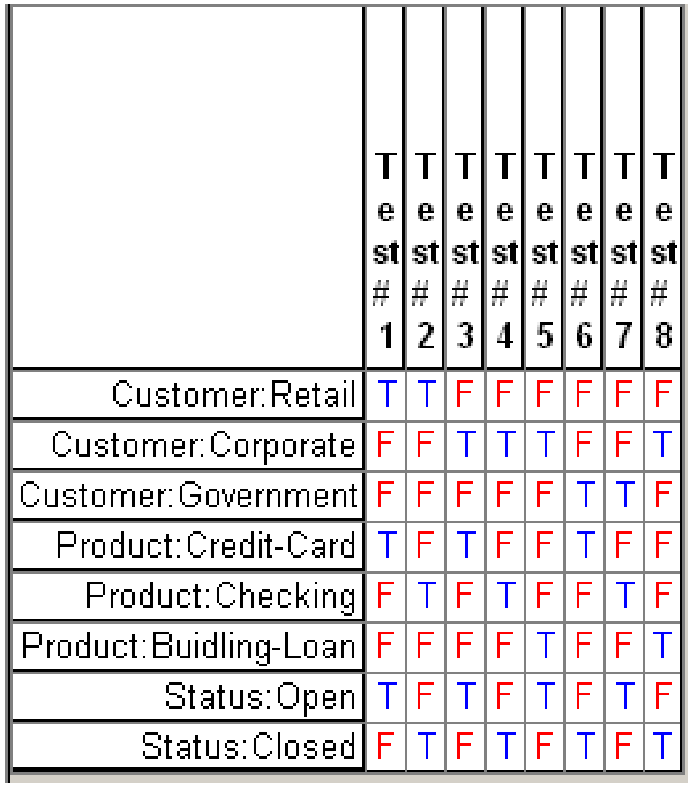
- The Pairs are merged into tests


QDesign - [QD1.bqd]
File Edit View Inputs Run Reports Options Utilities Window Help
For Help, press F1

```
Script Test Cases
Customer-Product-Status Tests


Test # 1 -- Customer-Product-Status Tests
The customer is a  retail customer
The product is  credit card
The account status is open

Test # 2 -- Customer-Product-Status Tests
The customer is a  Corporate Customer
The product is  credit card
The account status is open

Test # 3 -- Customer-Product-Status Tests
The customer is a  retail customer
The product is  checking account
The account status is closed

Test # 4 -- Customer-Product-Status Tests
The customer is a  governent customer
The product is  checking account
The account status is open

Test # 5 -- Customer-Product-Status Tests
The customer is a  Corporate Customer
The product is  building loan
The account status is open

Test # 6 -- Customer-Product-Status Tests
The customer is a  Corporate Customer
The product is  checking account
The account status is closed

Test # 7 -- Customer-Product-Status Tests
The customer is a  governent customer
The product is  credit card
The account status is closed

Test # 8 -- Customer-Product-Status Tests
The customer is a  Corporate Customer
The product is  building loan
The account status is closed
```

Quick Design

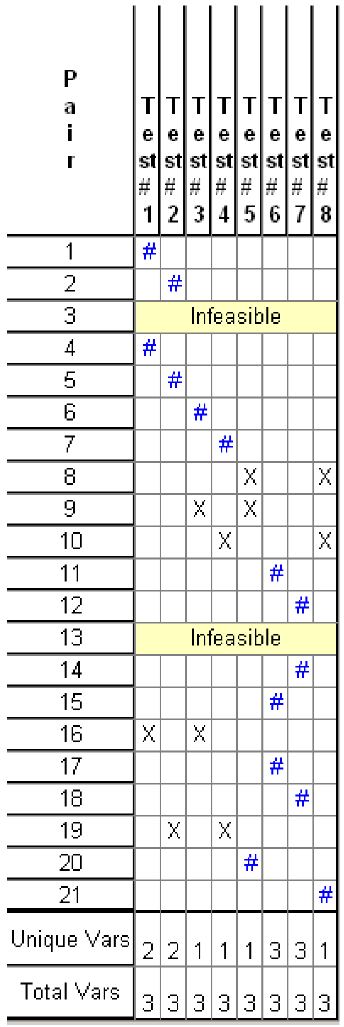
- The tests can be viewed in matrix format



	Test #1	Test #2	Test #3	Test #4	Test #5	Test #6	Test #7	Test #8
Customer:Retail	T	T	F	F	F	F	F	F
Customer:Corporate	F	F	T	T	T	F	F	T
Customer:Government	F	F	F	F	F	T	T	F
Product:Credit-Card	T	F	T	F	F	T	F	F
Product:Checking	F	T	F	T	F	F	T	F
Product:Buidling-Loan	F	F	F	F	T	F	F	T
Status:Open	T	F	T	F	T	F	T	F
Status:Closed	F	T	F	T	F	T	F	T

Quick Design

- Quick Design generates a coverage matrix



Pair	Test #1	Test #2	Test #3	Test #4	Test #5	Test #6	Test #7	Test #8
1	#							
2		#						
3	Infeasible							
4	#							
5		#						
6			#					
7				#				
8					X			X
9			X		X			
10				X				X
11						#		
12							#	
13	Infeasible							
14							#	
15						#		
16	X		X					
17						#		
18							#	
19		X		X				
20					#			
21								#
Unique Vars	2	2	1	1	1	3	3	1
Total Vars	3	3	3	3	3	3	3	3

Synergy With Other Tools

Test Activity	BenderRBT	Other Tools
Define Test Completion Criteria	BENDER RBT	
Design Test Cases	BENDER RBT	
Build Tests		Playback Tool / Data Base Utilities
Execute Tests		Playback Tool
Verify Test Results		Playback Tool / Data Base Utilities
Verify Test Coverage	BENDER RBT	
Manage Test Library		Test Manager

Synergy of BenderRBT and Requirements Repositories

- Basic Links:
  - Derivative
  - Change Notification
  - Functional Coverage
- Provides Traceability from Requirements in Repository
  - To The Logical Tests In RBT
  - To the Physical Tests In The Test Managers / Playback Tools
- Allows Users to View a Given Function in Test Case Format
- Allows Users to Review Test Status by Function
- (Much more to come in this area)

Synergy of BenderRBT and Playback Tools

- RBT process stabilizes the functional definition and user interface earlier.
  - Allows timely implementation of the scripts.
  - Minimizes scripts rework.
- RBT tool minimizes the required number of scripts.
  - 4X reduction for equivalent coverage.
  - Test scripting to test design time ratio 3:1 to 5:1 regardless of the test case design approach.
  - Minimizing the number of scripts reduces test implementation effort significantly.
  - Minimized script library reduces test execution/validation time.

Synergy of BenderRBT and Test Data Utilities

- Export to Grid Tools' Datamaker for automatic test data base creation.
- RBT ensures that the expected results are identified.
- Minimized number of tests means fewer items to compare and smaller test bases to manage.

Synergy of BenderRBT and Code Coverage Monitors

- Industry average code coverage at test "completion" is under 50%.
- People do not like to use tools that give them (and their managers) bad news.
- RBT results in 70% to 90% coverage during the initial pass.
  - Minor supplement required to complete code coverage.
- Combined with RBT's functional coverage analyzer gives full picture of functional testing status.

Synergy of BenderRBT and Test Library Managers

RBT Tests Automatically Exported Into Test Managers

When rules change:

- Identifies new tests required.
- Identifies necessary changes to existing tests.
- Identifies potentially redundant tests.
- Identifies tests no longer viable - i.e., violate constraints.

Synergy of BenderRBT and Defect Tracking

- Easier to do root cause analysis.
- Improves defect removal efficiency.
  - Phase defect introduced versus phase defect identified.
- Improves defect removal rate
  - Ratio of defects found during development versus total defects.

Test Design Summary

Validate Requirements	Cause-Effect Graphing	Path Coverage	Pair-Wise
Flexible Requirements Format	X		X
Ambiguity Eliminated	X		
Implicit Requirements Clarified	X		
Sequencing Clarified	X	X	
Concurrency Clarified	X		
Logical Relationships Clarified	X	x	
Intra-Functional Logical Consistency Verified	X	x	
Inter-Functional Logical Consistency Verified	X		

Test Design Summary

Test Design	Cause-Effect Graphing	Path Coverage	Pair-Wise
Expected Results Included	X	x	
Boundary Constraints Factored In	X		x
Observability of Defects	X		
Reduce Number of Tests	X	x	x
Test Coverage	100%	<50%	<50%
Can Support Agile Projects	X		

What BenderRBT Delivers:

- Maximum coverage with minimum tests:
  - 100% functional coverage.
  - 70-90% code coverage.
- Quantitative test progress metrics.
- Testing no longer a bottleneck.
- Highly portable test scripts.
- Tests any application written in any language running on any computer.

What BenderRBT Delivers

- Time to deliver
  - Reduced 20% to 30%
- Cost to deliver
  - Reduced 20% to 30%
- Residual defect rate
  - Reduce to zero or near zero