Assignment 3

Task 1

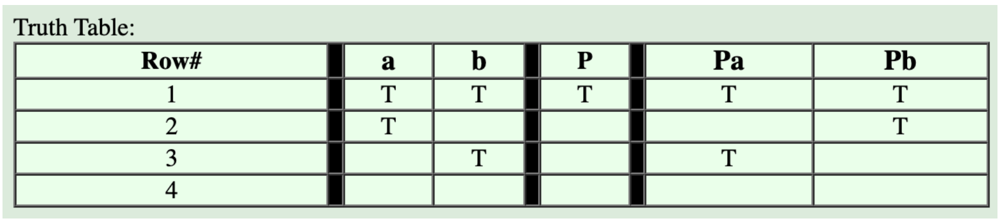
Preconditions

- A customer must have between 1 and 10000 bonus points
- A customer is either a gold customer or not

Logic Coverage

Predicate:
(1 <= bonus_points && bonus_points <= 10000) && (gold_customer: true OR gold_customer: false)

As instructed in the lab, I will only focus on the first part concerning bonus_points, since true or false evaluates to true: a is “1 <= bonus_points” and b is “bonus_points <= 10000”.



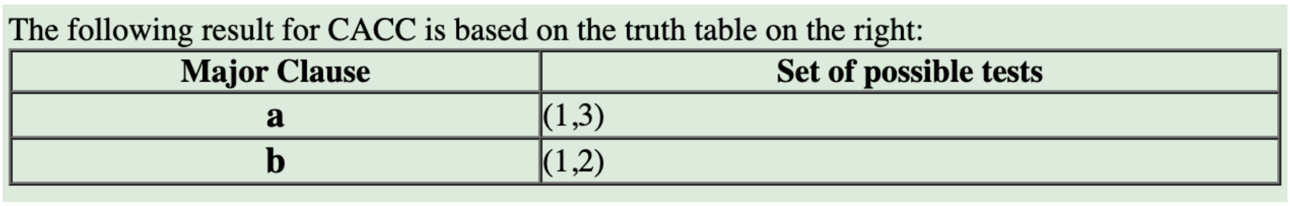
Truth Table:

Row#	a	b	P	Pa	Pb
1	T	T	T	T	T
2	T				T
3		T		T	
4					

Predicate Coverage is achieved by flipping the whole predicate true once, and false once, such as rows 1 and 2.

Clause Coverage is achieved by flipping each clause once, without caring about the value of the predicate. Rows 2 and 3 are a minimum requirement, but this does not give us predicate coverage.

Correlated Active Clause Coverage is achieved by having different major clauses, and the whole predicate must change its value.



The following result for CACC is based on the truth table on the right:

Major Clause	Set of possible tests
a	(1,3)
b	(1,2)

In row 1, each major clause is false. Then, we choose rows 2 and 3, ensuring we have made both a and b a major clause.

Enumerating tests with BVA:

Test	input	expected output	passed
ab_valid_1	{1}{true}	0	pass
ab_valid_2	{10000}, {true}	1	pass
a_valid	{10001}{true}	-1	fail
b_valid	{0}, {true}	-1	pass

I added a test for the condition where both a and b are valid, to test both boundary values 1 and 10000. The test a_valid with a bonus_point input of 10001 does not return -1 as expected, thus a fault was found.

Task 2

Modeling the input space using ISP

(1 <= bonus_points < 80 XOR 80 <= bonus_points < 120 XOR 120 <= bonus_points <= 10 000) && (gold_customer: true OR gold_customer: false)

Equivalence classes covering domains of the partitions bonus_points (b) and gold_customer (g):

	b	g
1	1 to 79	TRUE
2	80 to 119	FALSE
3	120 to 10000	

Each Choice: *at least one value from each block*
[b1, g1], [b2, g2], [b3, g1]

Pair-Wise: *a value from each block for each partition must be combined with a value from every block for each partition*
[b1, g1], [b1, g2], [b2, g1], [b2, g2], [b3, g1], [b3, g2]

The Pair-Wise criterion also covers the pairs for Each Choice. However, the expected output for b1 and b3 is the same, regardless of the golden status of the customer.

Boundary Value Analysis

There are three groups of points and some overlap in the values, for example, more than 79 can be 80 in the second block.

Blocks	Values	Chosen inputs
1 to 79	0, 1-79, > 79	0, 1
80 to 119	< 80, 80-119, > 119	80
120 to 10000	< 120, 120-10000, > 10000	120, 10000, 10001

Tests

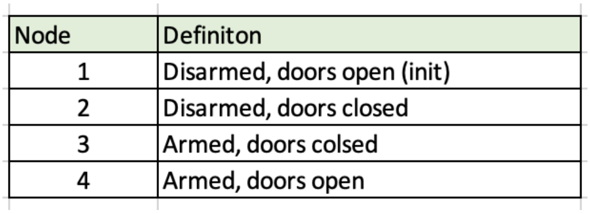
Here is a combination of tests generated with Pair-Wise modeling and boundary value analysis. I started with 6 tests and added some extra tests based on the boundary value analysis.

Test	input	expected output	passed
b1g1_0	{0},{true}	-1	pass
b1g1_1	{1},{true}	0	pass
b1g2_1	{1},{false}	0	pass
b2g1_80	{80},{true}	1	pass
b2g2_80	{80},{false}	0	pass
b3g1_120	{120},{true}	1	pass
b3g1_10000	{10000},{true}	1	pass
b3g2_120	{120},{false}	1	pass
b3g2_10000	{10000},{false}	1	pass
b3g1_10001	{10001},{true}	-1	fail

As seen from the table above all tests passed except the last one. The tenth test tested whether or not the system allows for illegal input exceeding 10000. It failed as in Task 1, implying a fault in the implementation.

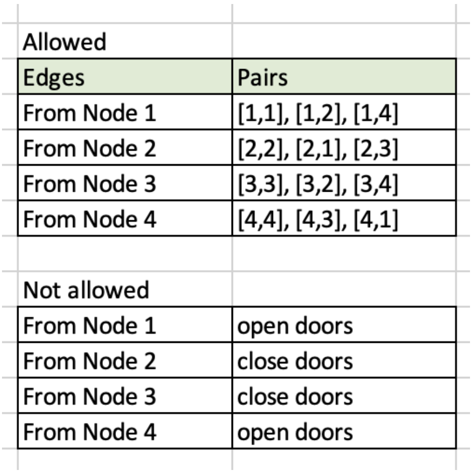
Task 3 – State Machine

The nodes and the edges of the state machine were enumerated:



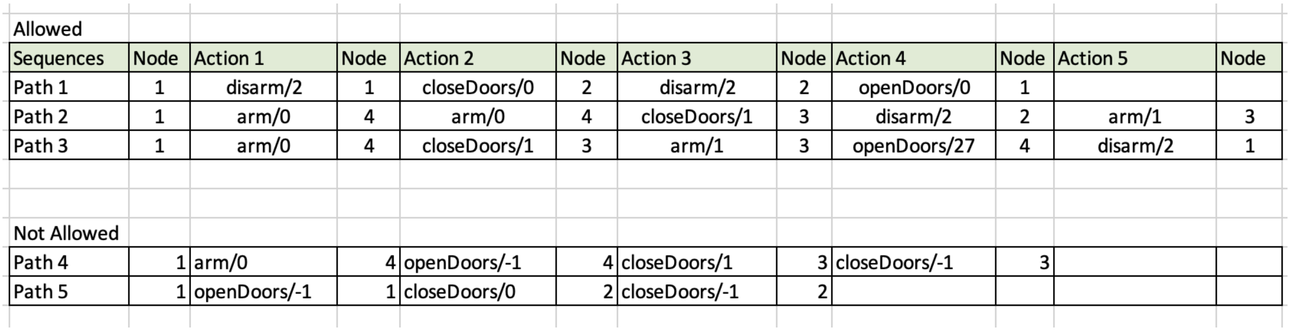
Node	Definiton
1	Disarmed, doors open (init)
2	Disarmed, doors closed
3	Armed, doors colsed
4	Armed, doors open

The state machine did not include the opening and closing of doors when the return is expected to be -1 if the action does not change the doors' state. I have listed the actions that are not allowed here as well:



Allowed	
Edges	Pairs
From Node 1	[1,1], [1,2], [1,4]
From Node 2	[2,2], [2,1], [2,3]
From Node 3	[3,3], [3,2], [3,4]
From Node 4	[4,4], [4,3], [4,1]
Not allowed	
From Node 1	open doors
From Node 2	close doors
From Node 3	close doors
From Node 4	open doors

I then created test paths with a maximum of five actions per test case. I have also listed the start and end nodes along with the nodes that are traversed on each test path.



Allowed											
Sequences	Node	Action 1	Node	Action 2	Node	Action 3	Node	Action 4	Node	Action 5	Node
Path 1	1	disarm/2	1	closeDoors/0	2	disarm/2	2	openDoors/0	1		
Path 2	1	arm/0	4	arm/0	4	closeDoors/1	3	disarm/2	2	arm/1	3
Path 3	1	arm/0	4	closeDoors/1	3	arm/1	3	openDoors/27	4	disarm/2	1
Not Allowed											
Path 4	1	arm/0	4	openDoors/-1	4	closeDoors/1	3	closeDoors/-1	3		
Path 5	1	openDoors/-1	1	closeDoors/0	2	closeDoors/-1	2				

State coverage is achieved by visiting each node at least once, and transition coverage is achieved by visiting each edge between the states.

The table above also specifies the actions and their expected outputs, as provided in the state machine. The not-allowed actions are edges not visualized in the state machine and return -1 when performed. They are all about opening or closing a door, in a way that does not change the state.

The table below represents all edges in the paths:

Sequence	Transitions	Passed
Path 1	[1,1], [1,2], [2,2], [2,1]	pass
Path 2	[1,4], [4,4], [4,3],[3,2] ,[2,3]	pass
Path 3	[1,4], [4,3], [3,3], [3,4], [4,1]	pass
Path 4	[1,4], [4,4], [4,3], [3,3]	pass
Path 5	[1,1], [1,2], [2,2]	pass
Path 6	[1,2], [2,3], [3,4], [4,2]	fail at [3,4]

Each transition is visited at least once. Paths 4 and 5 are added to test transitions not visualized in the state machine: opening and closing doors without changing the state. Thus, with this set of paths, no fault was found.

Fault found

This is how I first conducted my tests. However, I got suspicious when I did not find faults. I suspected something was still off between nodes 3 and 4, as sounding the alarm is crucial, and decided to check another path:

Added path											
Path 6	1	closeDoors/0	2	arm/1	3	openDoors/27	4	disarm/2	1		

At this path, there is a fault when traversing from node 3 to 4, that did not appear in path 3 for the same transition. This implies an issue with the implementation when coming from “disarmed, doors closed” to “armed, doors closed”, and then opening the door. The expected output is 27, but the actual output is 0, with no alarm sound.

This fault was almost not detected, even with transition coverage, and it could have been better to have a transition pair coverage approach to be sure to visit the edge openDoors/27 in both sequences [4,3,4], which worked, and [2,3,4] that did not work.