Task 1: Create a test plan

1. There are 12 combinations of tests for 2-way interactions.
2. Theoretically, there are 1152 combinations for the above parameters.

I’m not sure about the question, but to provide some kind of database that tells whether the chosen car is present in the database, suppose we have a table “CAR\_BOOKING\_SERVICE\_MASTER” that contains all the information about the cars that we have in the workshop. Each of the cars has an ID number that we can call using our database management tool.

To see if we have the car or not, we can just use the query as simple as (just example):

SELECT \* FROM CAR\_BOOKING\_SERVICE\_MASTER WHERE CAR\_TYPE = ‘TRUCK’ AND COLOR = ‘LILAC’ AND …(and so on)

Task 2: Removing impossible test cases (adding constraints)

**Constraints:**

(CarType = "Truck") => (SideAirbagBack = "None")

(PassengerSeatAirbag = "Yes") => (PassengerSeatChildSeat = "No")

1. After adding the above constraints, I got 16 combinations of tests. Here’s the Excel file. (double click on the icon to open)



I also tried 3-way interactions for the test and got 41 combinations of tests. Here’s the Excel file.



1. The result wasn’t exactly like I would’ve expected, because I thought there would be a lesser combination of tests after adding constraints. I think this happened because the previous combination with the existing condition (before adding constraints), hadn’t covered up all the conditions after adding the constraints. Thus, the system fulfills all the conditions, adding the combination tests to the result. That’s why there are more combinations of tests after adding constraints than before adding constraints.
2. After comparing the first list with the list after adding the constraints, I found that adding constraints doesn’t necessarily mean reducing the combination of tests. Sometimes it can increase the number of combinations in order to fulfill all the conditions.

Task 3: Enforcing pairs (adding requirements)

**Constraints:**

1. Constraints 1

(CarType = "Truck") => (SideAirbagBack = "None")

(PassengerSeatAirbag = "Yes") => (PassengerSeatChildSeat = "No")

1. Constraints 2

(CarType = "Truck") => (SideAirbagBack = "None")

(PassengerSeatAirbag = "Yes") => (PassengerSeatChildSeat = "No")

(CarType = "Truck") && (Color = "Yellow") && (PassengerSeatAirbag = "Yes")

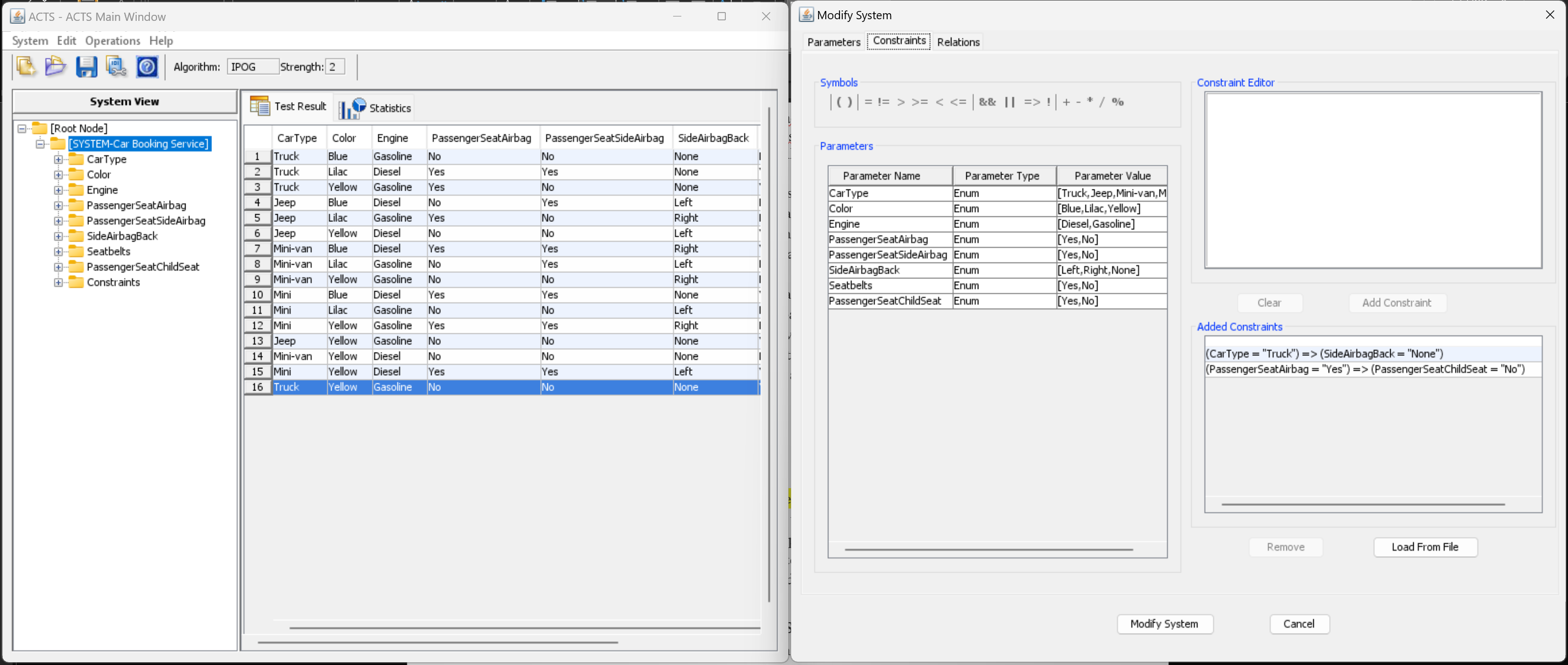
1. I got 22 tests. I got it by combing 2 sets of constraints above (Constraints 1 and 2). Constrains 1 was to cover the combination of tests for the traffic regulations which said that you cannot put a child seat in the passenger seat if there exists an airbag, while also a Truck never has a back seat, which implies that we cannot have any Side Airbag Back.

Constrains 2 was to cover the combination for “**most of** the customers want a yellow truck with Passenger Seat Airbag”, along with the mandatory traffic regulations. This second constraint gave only truck combinations in the result. That’s why I need to combine it with the first constraints to give all the possible choices for the customers. In this case all the possible combinations of tests.

Here’s the Excel file:



1. The hint said that ACTS has two modes for generating test suites. In this task, I use File-Based Testing (FBT) Mode which tells that ACTS generates test cases based on user-provided input files. So the user specified the inputs along with the constraints. ACTS then generates the test suites based on the inputs and the constraints. I tested the requirements directly using the ACTS app.



Task 4: Adding parameters to see combinatorial growth

* After adding a parameter with only one value, there are no additional pairwise tests.
* After adding one more value to the parameter, there are still no additional pairwise tests.
* With 3 values in the additional parameter, the number of pairwise tests increased to 15.
* After adding a few parameters with a different number of values, I got that what affects the number of 2-way interaction tests mostis **the** **number of values,** because 2-way interaction or X-way interaction corresponds to the X parameters with the highest number of values.