Rule Modeling Challenge – Collections – Mike Parish

Given an arbitrary number of collections of cars:

- find the make and model that is common to all collections
- Identify makes and models that are in only one collection.
- Find the most expensive car in each collection (regardless of make and model)

Group	Make	Model
G1	Ford	Pickup
G1	Ford	T
G1	Ford	Taurus
G1	Austin	Mini
G1	Hyundai	Santa Fe
G2	Ford	Pickup
G2	Ford	T
G2	Ford	Taurus
G2	Hyundai	Tucson
G2	Hyundai	Santa Fe

Difference	
Make	Model
Hyundai	Tucson
Austin	Mini

Intersection	
Make	Model
Ford	Pickup
Ford	T
Ford	Taurus
Hyundai	Santa Fe

Introduction

The way the data is organized can have a significant impact on how you model the rules.

Two solutions are presented based on different data models.

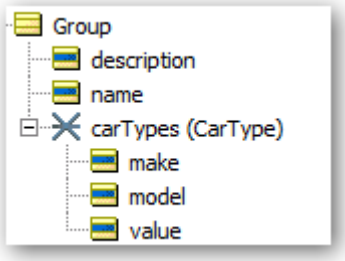
Part I uses an N:N association data model and that leads to a very simple rule model.

Part II uses a “flat” record oriented data model and converts the flat data into structured data (using 1:N associations).

Part I Association Model

The Vocabulary

Let's assume that the same car type (make and model) can be a member of one or more groups.
So make and model are not duplicated but are referenced by (or associated with) the groups they belong to. This is how several of the other solutions interpreted it.
This can be modeled as a many-to-many association in Corticon as indicated by the X symbol.



In part II I'll discuss an alternative way to model this that does allow make and model to be repeated as they are in the sample data (possibly with different pricing). This becomes clearer if you add the vehicle VIN to the model.

The Rule Sheet

Normally you'd divide up a problem into a number of rule sheets to make it easier for humans to understand the logic, but for this example we'll do it all in one rule sheet to save space.

Here's the natural language view of the decision table. It expresses how a human might describe the problem and its solution.
It's written from the point of view of an individual car type.
Corticon automatically applies it to all car types or groups as appropriate.

Conditions	0	1	2	3	4
Is this make and model in all groups?		T	F	-	-
Is this make and model in more than one group?		-	T	-	-
Is this make and model in exactly one group?		-	-	T	-
Is this make and model in no groups?		-	-	-	T
Actions					
Post Message(s)					
Add the make and model to the common collection		✓			
Add the make and model to the unclassified collection					✓
Add the make and model to the unique collection				✓	
Add the make and model to the duplicate collection			✓		
Add the most expensive make and model in each group to the expensive collection	✓				

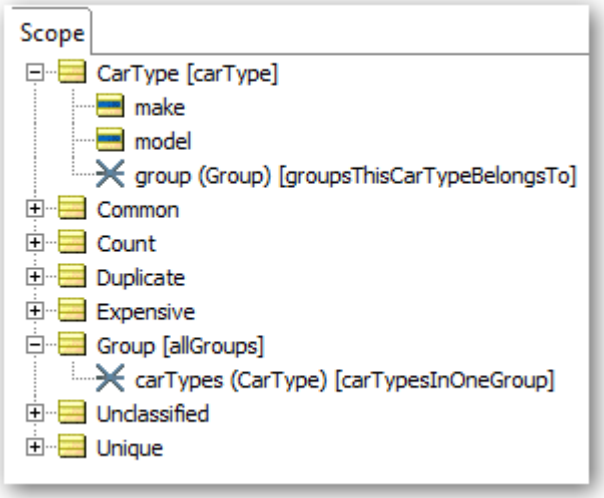
And here is one possible implementation using the Corticon equivalent of FEEL:

	Conditions	0	1	2	3	4
a	groupsThisCarTypeBelongsTo->size = allGroups->size		T	F	-	-
b	groupsThisCarTypeBelongsTo->size > 1		-	T	-	-
c	groupsThisCarTypeBelongsTo->size = 1		-	-	T	-
d	groupsThisCarTypeBelongsTo->size = 0		-	-	-	T
	Actions					
	Post Message(s)					
A	common+=carType		✓			
B	unclassified+=carType					✓
C	unique+=carType				✓	
D	duplicate+=carType			✓		
E						
F	expensiveCarTypes+=carsTypesInOneGroup->sortedBy Desc(value).first	✓				

Interestingly, apart from the reference to value in line F (which deals with the expensive cars and should really be on a separate sheet), the rule uses no attributes at all – it's all done using associations!

The expression on line F takes each group, sorts by descending value and then takes the first car type (which will be the most expensive in that group) and then adds it to the collection of expensive car types.

The context (or data view) for the rule sheet is defined in the Scope section.
It summarizes the associations, aliases, objects and attributes used in the rule sheet.
It's created automatically as you build the rule sheet – you just have to choose suitable alias names as shown in [ ]



The alias *groupsThisCarTypeBelongsTo* is the reverse association to the *carTypesInOneGroup* association and enables us to easily find the groups to which a particular car type belongs.

In Corticon you can specify either or both.

The other collections, **Common**, **Unique**, **Duplicate**, **Unclassified** and **Expensive** are one-to-many associations to car types that we'll use to classify the car types.

We could also have done it by setting attributes in the car type records to indicate how it was classified.

In general working with collections is much more powerful since a rich set of operators is available to determine such things as the size of the collection or the average, max or min of any of the attributes within any of the collections.

That's all there is to the Corticon solution!

Summary

This solution makes use of the reverse association (from car types to the groups(s) they belong to). When collections of related objects are modeled this way, complex logic can be expressed quite succinctly. Several of the other solutions use a similar technique.

Technically when we refer to "car" in the example above it's not a specific car we're referring to but rather the class of cars that share that make and model. That's why we made the simplifying assumption that the make and model is not duplicated but instead referred to from the groups to which it belongs.

But what if the data is not structured but instead comes in as "flat" records that simply contain a field indicating which group the make and model belong to? Each record now might represent a specific vehicle (especially if we add the VIN to the record). This is actually how the data is described in the original challenge statement. In this data structure the same make and model may be repeated in different records (for the same vehicle or a different vehicle) so we'll need a slightly different approach.

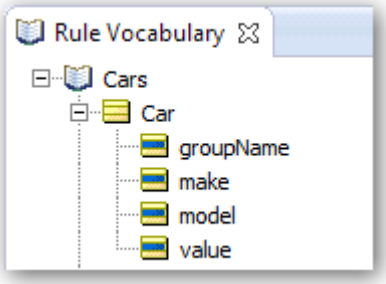
For the flat data model solution read on….

For sample test cases see the Appendix

Part II "Flat" Record Model

The Vocabulary

In Corticon the basic data model for the flat records can be represented as shown in the vocabulary.



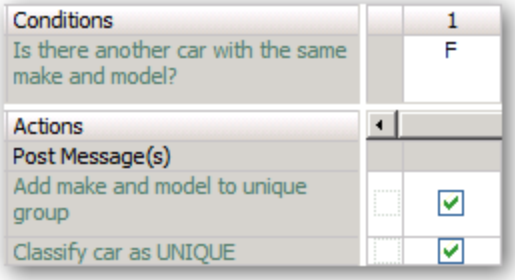
The group name is simply another attribute of the car record and there's nothing to prevent the same make and model being repeated in the same or different groups with the same or different value.

The rule model should be able to handle this.

Later we'll add some other variables and objects to store the results

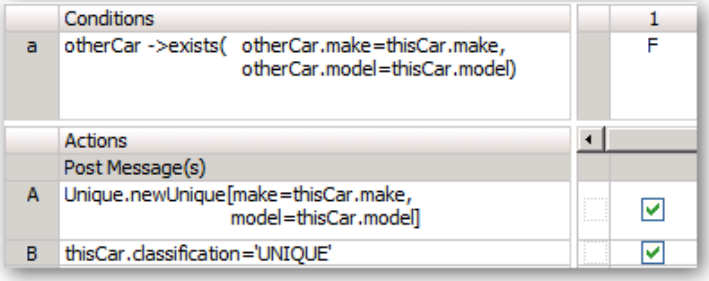
Find Unique Make and Model

We'll start with finding the unique make and model since it doesn't need any additional vocabulary items. The rules operate directly on the flat records.



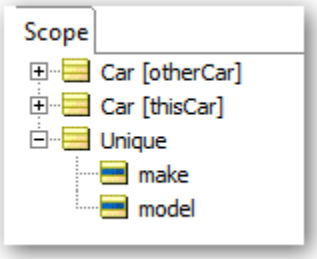
Basically we need to answer the question "Does each make and model exist in any other group".
If it does not then that make and model is unique.

Corticon's expression language is based on set theory;
It supports the concepts of "exists" and "forAll" with built in operators so we can create a rule sheet like this:

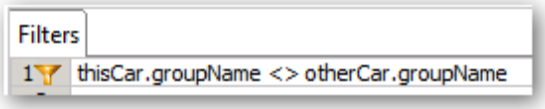


Action (A) creates a new object to record the make and model (without duplication)
Action (B) marks the car as unique

The scope defines two aliases (**thisCar** and **otherCar**) to the set of cars.



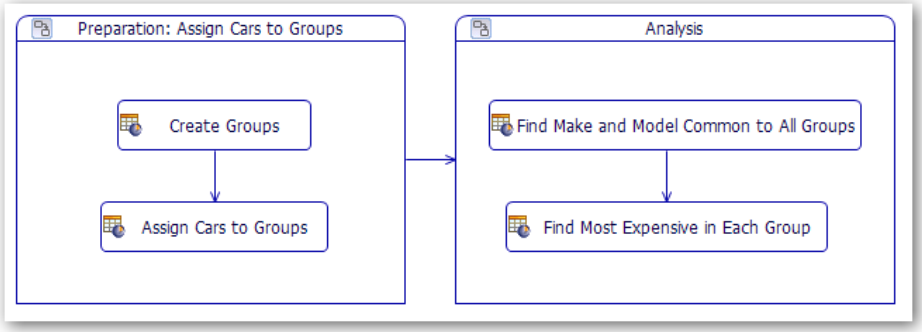
The filter ensures that **thisCar** and **otherCar** are in different groups.



Find Make and Model that Exist in All Groups

This requires that we arrange the cars into groups based on their group name. Each car record (most likely identified by its VIN) is linked with one group (because the record only contains one field for the group name) but the vin, make and model values may be repeated in other groups or even in the same group with a different price with an additional record. We need to find the make and models (not vehicles) that are in all groups.

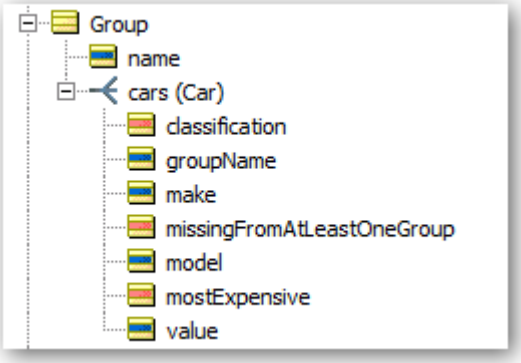
The problem is solved in two parts, Preparation and Analysis



The preparation section transforms the flat car record data into structured data that associates cars with the groups to which they belong. This enables us to take advantage of Corticon's powerful collection processing abilities.

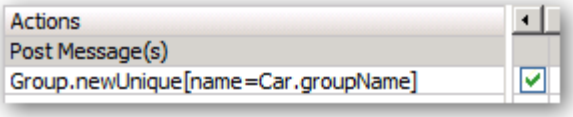
We'll therefore need to modify the vocabulary to introduce the notion that cars can be organized into groups identified by the name of the group.

We also introduce some transient variables (orange).



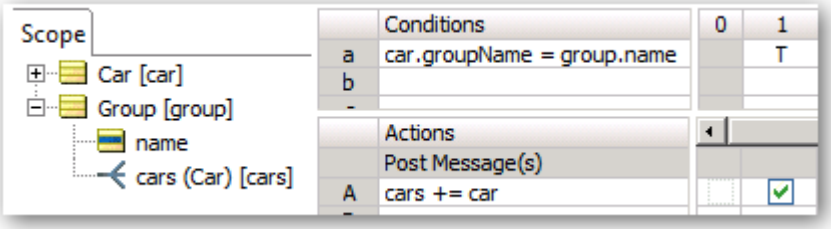
Create Car Groups

This statement will create a unique new group for each group named in the car's data record.



Assign Cars to Groups

This sheet compares the car's group name with the name of the groups we created previously and, if they match, associates that car with the group.

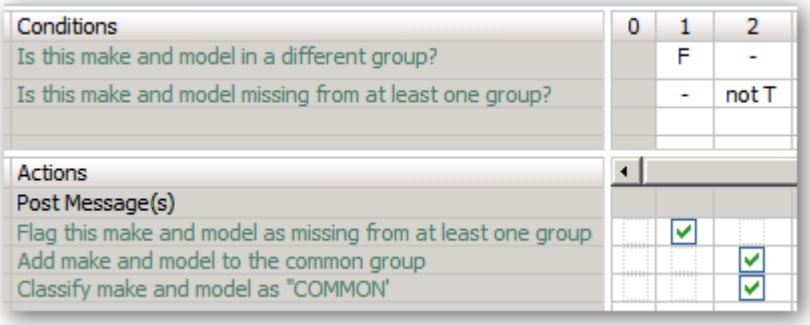


Note that the crows foot symbol shows a one to many association that enforces that a specific physical car (record) may belong to only one group at a time (though its make and model are not so constrained). Moving it to a new group will move it out of its old group.

Analysis

Find Common Make and Model

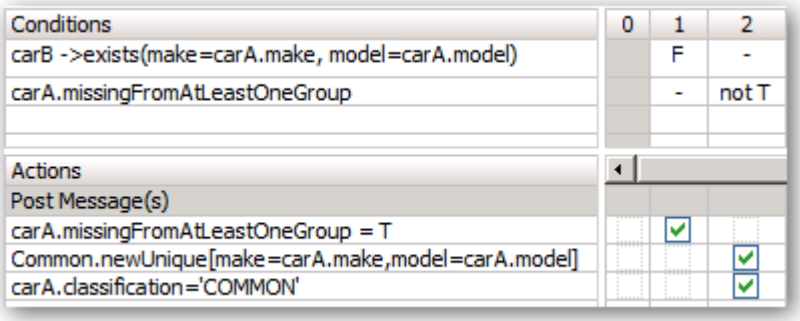
This time we ask the question "Is the make and model of a car in one group missing from any other group?"



We cannot do the simpler counting methods that worked when we had a many to many association because it's not necessarily the same car in the other group (even if the make and model are the same).

If a make and model is missing from any other group then that make and model cannot be common to all groups so we flag it as "missing from at least one group". Any make and model that does NOT get flagged is therefore in all groups.

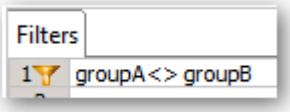
This can be achieved using the ->**exists** operator.

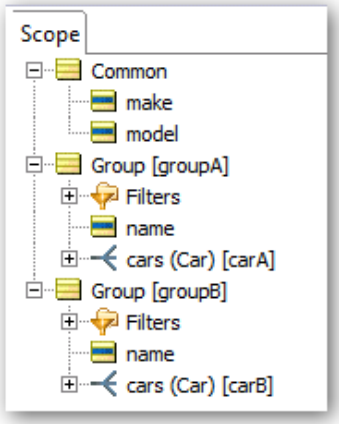


The ->exists operator is very powerful and enables us to find out if a particular make and model (referred to as CarA) matches another (CarB) in a different group.

The exists test will be applied to all cars. No need for loops. Its all automatic. Corticon will process all cars through rule 1 before moving on to rule 2

This time we have aliases to define two different groups of cars (**groupA** and **groupB**) and the cars within those groups (**carA** and **carB**). Again the filter ensures that **groupA** and **groupB** are different.



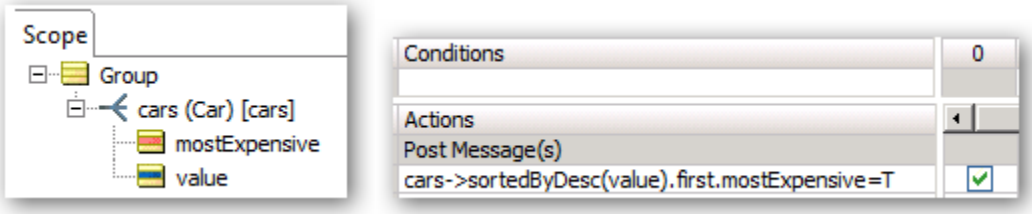


Note: Although the **->exists** statement looks very similar to the one used to find the unique cars, there is an important difference in the scope. For the unique rule sheet the **->exists** operated on the entire set of car records using aliases **thisCar** and **otherCar**. For the common rule sheet the **->exists** is operating on sets of cars (**carA** and **carB**) but they are within different groups. It's a subtle but very important point. It's why we needed to arrange the cars in groups in the first two rule sheets.

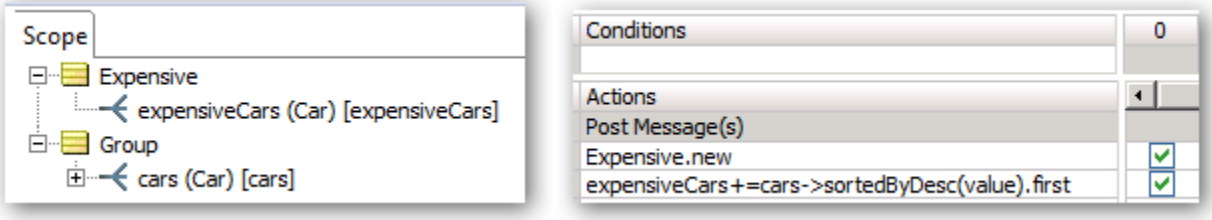
Find Most Expensive Car in Each Group

This time we are not concerned with the make and model, but rather with the actual car in each group and its value. The same make and model in different groups could have different values since they are physically different cars. Again putting a VIN in the data helps to make this clearer.
To keep track of which vehicle is the most expensive we'll introduce a Boolean attribute called **mostExpensive** which we'll set to True as necessary.

All we need in the rule is



Alternatively we could put the most expensive cars in a group of their own like we did with the unique and common ones. Again this rule will be applied across all groups and across all cars within each group.



We could of course do both.

The alias **cars** makes sure that the sorting by descending value and taking the first car is done separately within each group.

Summary

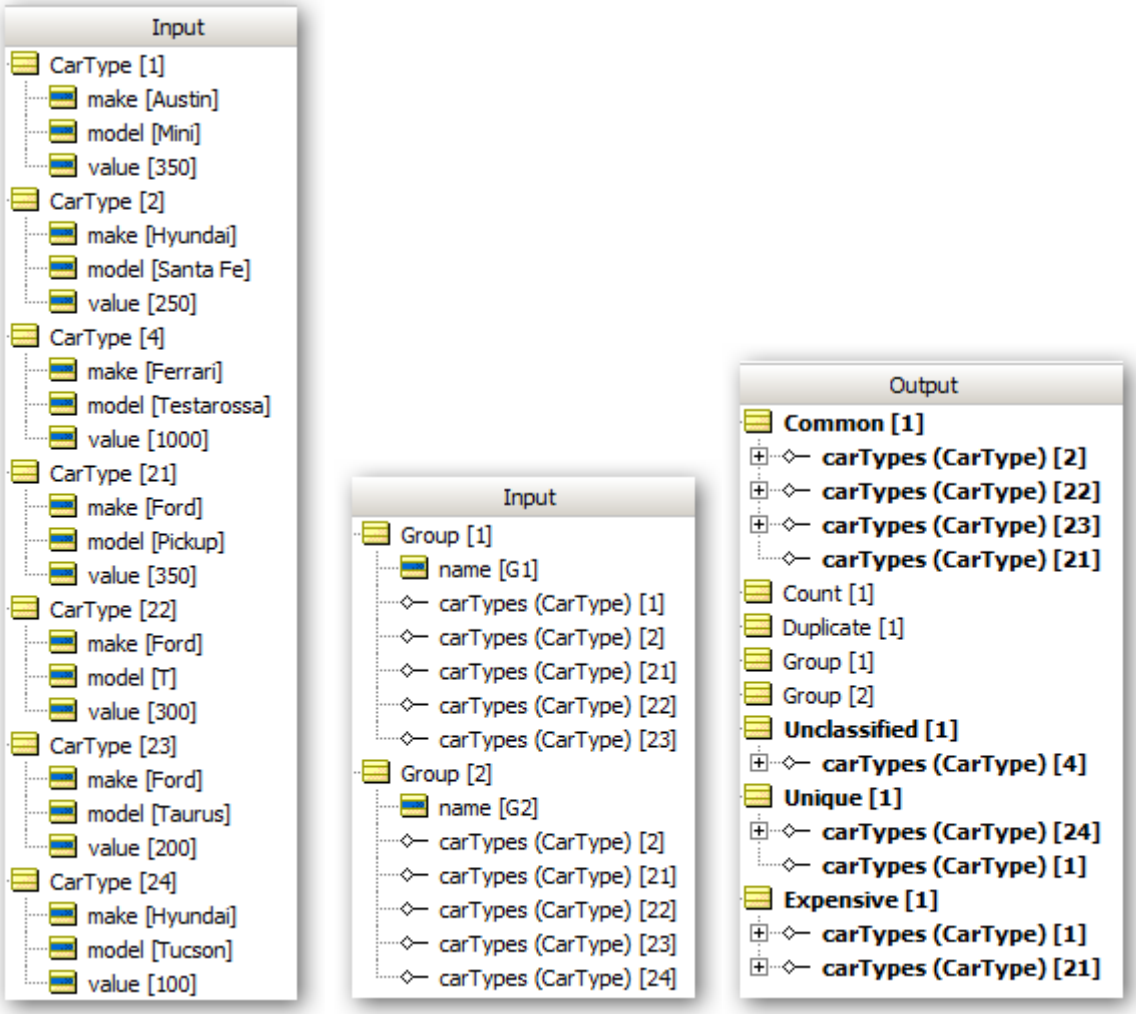
Corticon is easily able to handle complex rules that deal with set membership because it has two important built-in features:

1. The ability to model the relationships between objects 1:1, 1:N and N:N
2. Set theory operators that allow testing for **exists** and **forAll**

Appendix – Test Cases

Test Case – Using two groups (N:N Associations)

A slight variation on the sample data in the challenge to show more categories- there's a make and model that is not in any group.



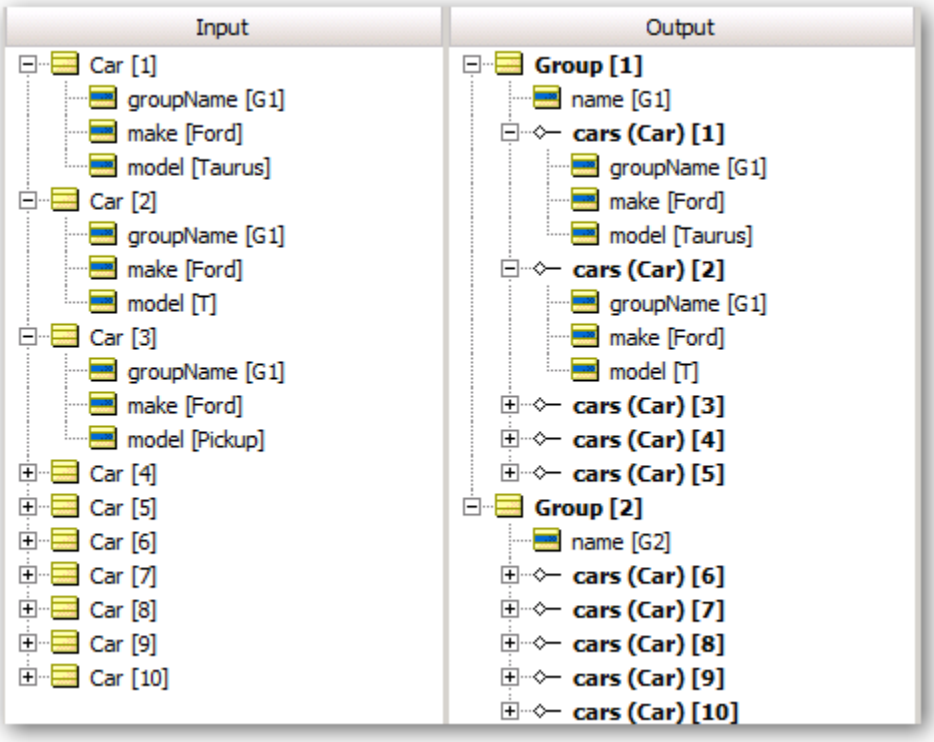
Note that the same car may be in more than one group (or just one, or none)

Messages

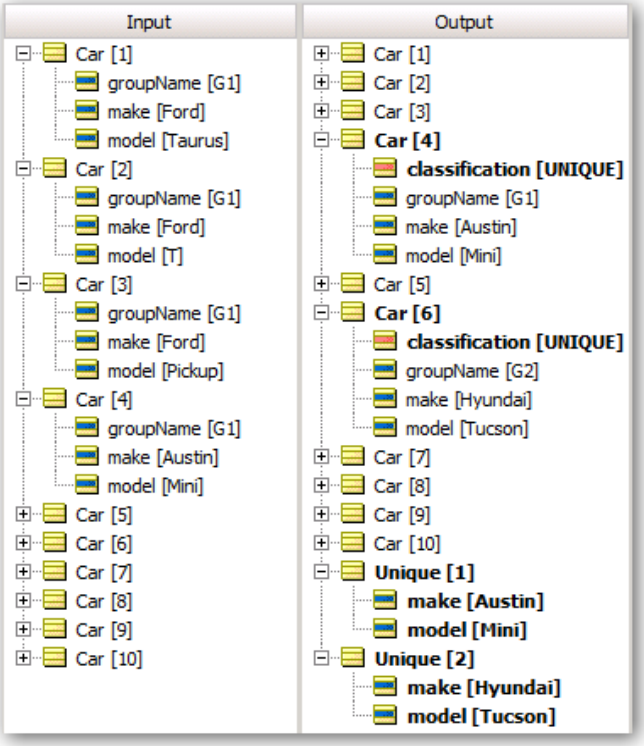
As each rule executes on one of the cars we can attach a suitable explanation of the outcome
The number [n] indicates which rule fired for that car

Message	Entity
[3] Austin Mini is in one group	CarType[1]
[1] Hyundai Santa Fe is in all groups	CarType[2]
[4] Ferrari Testarossa is not in any group	CarType[4]
[1] Ford Pickup is in all groups	CarType[21]
[1] Ford T is in all groups	CarType[22]
[1] Ford Taurus is in all groups	CarType[23]
[3] Hyundai Tucson is in one group	CarType[24]

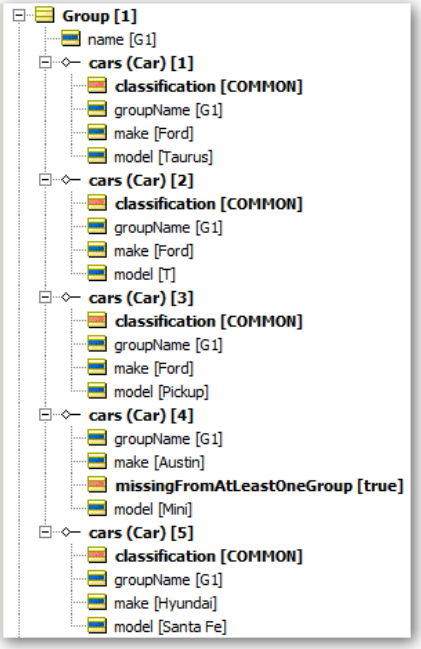
Test Case Create Hierarchical Data Structure from Flat Records

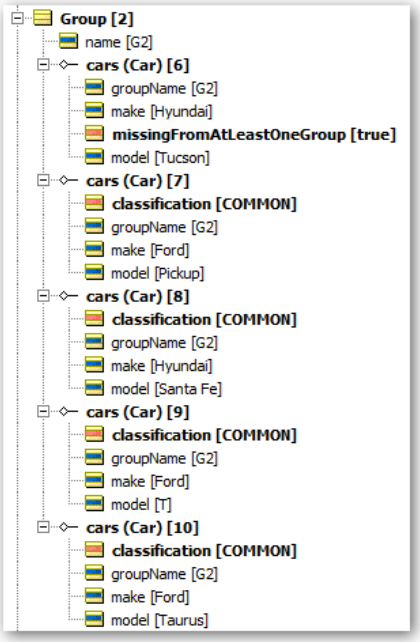


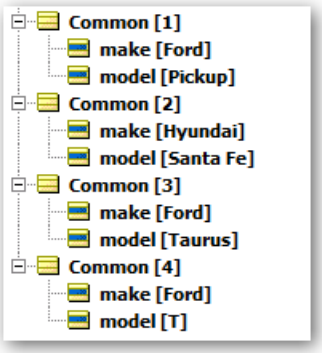
Test Case UNIQUE (using the Problem Example Data)



Test Case Output – Find Common Make and Model







Test Case (Expensive Cars)