Who Killed Aunt Agatha – A Corticon Solution – by Mike Parish

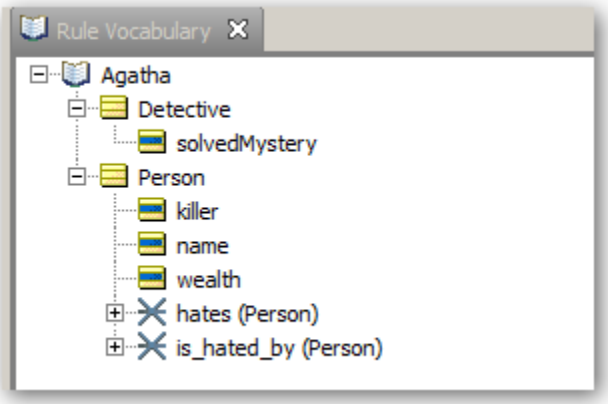
Problem Definition

Someone in Dreadsbury Mansion killed Aunt Agatha. Agatha, the butler, and Charles live in Dreadsbury Mansion, and are the only ones to live there. A killer always hates, and is no richer than his victim. Charles hates no one that Agatha hates. Agatha hates everybody except the butler. The butler hates everyone not richer than Aunt Agatha. The butler hates everyone whom Agatha hates. No one hates everyone. Who killed Agatha?

Observation

Although this is not typical of the kinds of rules that we see in business it can still be modeled using decision tables and the model can be executed to produce a solution.

The Vocabulary



The vocabulary is modeled to allow a person to hate and to be hated by multiple people (a many to many association)

The Main Rule Sheet

Here is a natural language view of the decision

Conditions	1	2	3	4	5
Consider a person (A)	'Agatha'	'Butler'	'Charles'	'Butler'	'Agatha'
Consider any person (B)	not 'Butler'	-	-	-	-
Is B no richer than the victim (Agatha)?	-	T	-	-	T
Is B hated by Agatha?	-	-	T	T	-
Is Agatha hated by B?	-	-	-	-	T
Actions					
Post Message(s)	✉	✉	✉	✉	✉
A hates B	☑	☑		☑	
A does not hate B			☑		
B is a potential killer					T
Detective may have solved the mystery					T

And here is the technical expression language that implements the natural language

Conditions	1	2	3	4	5
person.name	'Agatha'	'Butler'	'Charles'	'Butler'	'Agatha'
anyPerson.name	not 'Butler'	-	-	-	-
anyPerson.wealth <= Agatha.wealth	-	T	-	-	T
anyPerson.is_hated_by = Agatha	-	-	T	T	-
person.is_hated_by = anyPerson	-	-	-	-	T
Actions					
Post Message(s)	✉	✉	✉	✉	✉
person.hates+=anyPerson	☑	☑		☑	
person.hates-=anyPerson			☑		
person.killer					T
Detective.solvedMystery					T

The Rule Statements

1	Rule1: Agatha hates everybody except the butler. Deduction: Agatha hates {anyPerson.name}
2	Rule2: The butler hates everyone not richer than Aunt Agatha. Deduction: the butler hates {anyPerson.name}
3	Rule3: Charles hates noone that Agatha hates. Deduction: Charles does not hate {anyPerson.name}
4	Rule4: The butler hates everyone whom Agatha hates. Deduction: The butler hates {anyPerson.name}
5	Rule5: A killer always hates (the victim), and is no richer than his victim. Deduction {person.name} is the killer.

Note the use of substituted variable references {anyPerson.name} to customize the message for the particular persons that triggered the rule. If the rule applies multiple times then multiple messages will be generated. – See the test case for an example

How the Rules Work

The rules work in conjunction with the scope and filters.
The **scope** defines various collections (aliases) of persons that are of interest to us (like those that hate someone or those that are hated by someone)
The **filters** enable us to put constraints on which persons might belong to any collection (or alias)

What the first four rule columns are doing is determining who hates whom – the actions of the rules simply add or remove people from a person's list of hates.

Essentially we represent in Corticon exactly what the business rules said in English and then let Corticon try all the possible combinations. We do not try to figure out a specific procedural algorithm that will solve the problem (which is what we might do if using java). We try to use a declarative approach to specifying the business rules and let Corticon generate the conclusions.
There aren't very many combinations in this particular problem, but if we were using Corticon to solve Sudoku then the number of possible combinations would be huge!
The board game CLUE (Professor Plum in the study with the dagger) would be a more interesting problem since there are a lot more variables.

In some conditions we pin down a specific person by name and in other conditions we use the reference anyPerson – Corticon will automatically try the rules for all instances that are in that collection (determined by any filters we might have defined). The association name "is_hated_by" also identifies a collection that may have many members – Again Corticon will automatically try them all.

Rule 1 is simple: it says the person named "Agatha" hates anyone who is not named "Butler" (even Agatha herself)

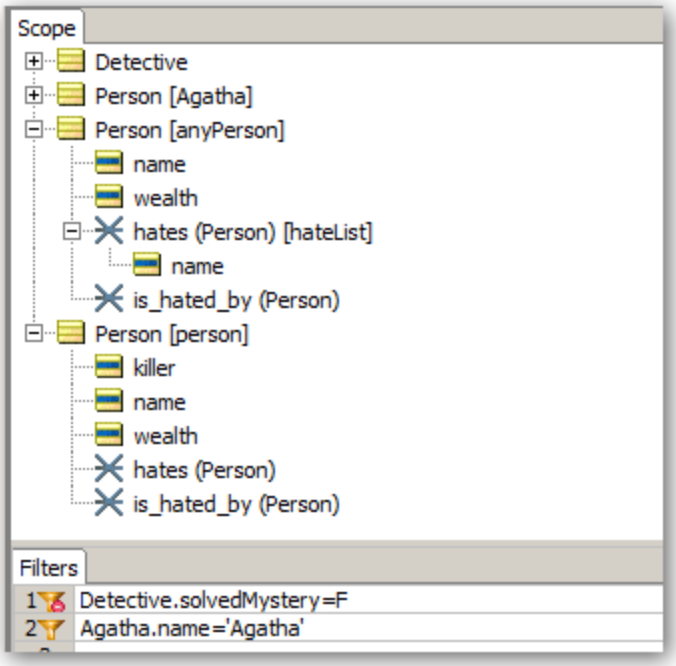
Rule 2 is a bit more complex: it says the person named "Butler" hates any person whose wealth is less than Agatha's wealth. (and since anyPerson has no filter it could also refer to Agatha herself)

Rule 3 is more complex again: the condition selects anyPerson who is hated_by Agatha (the other direction on the association) and removes them from the list of people that Charles hates (if they are on his hate list)

Rule 4 does the same this with Butler except that the Butler hates anyone that Agatha hates

Rule 5 Finally asks for the person named "Agatha" does anyone hate her and is their wealth less than or equal to her wealth. If so then they are a potential killer.

The references to person, anyPerson and Agatha are defined in the scope section of the rule sheet and populated by the filters.

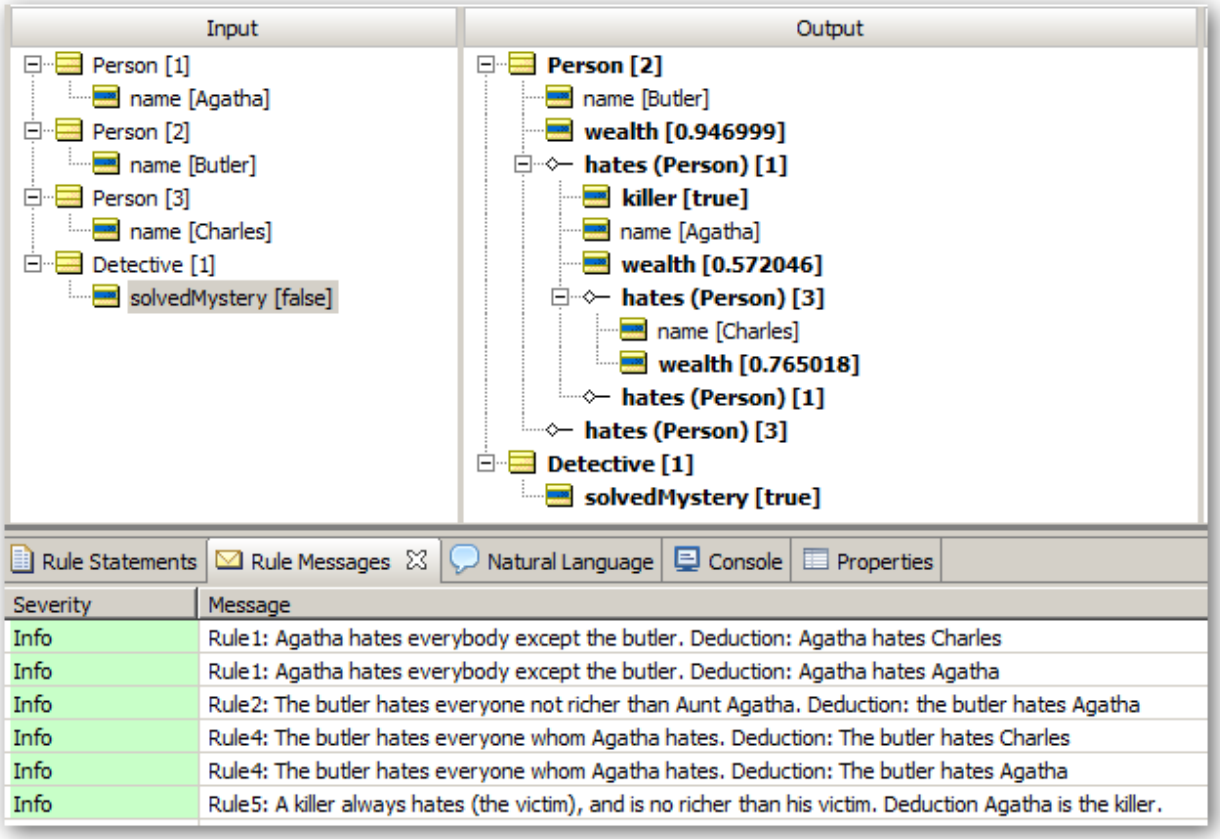


Filter 1 is a precondition which determines if the rule sheet even needs to be executed;
Filter 2 sets the collection alias Agatha to the (only) Person whose name is “Agatha”.
The aliases person and anyPerson are free variables (no filters) that Corticon can bind to any instance of Person when executing the rules.

Essentially what happens here is that Corticon’s default behavior is to apply the rules to every instance of the business objects that successfully pass though the filter. The alias provides a way to refer to different collections of instances that are formed by the various filters. The membership of these collections may change dynamically as the rules execute.

So in the case of the alias Agatha, the filter requires a Person whose name=’Agatha”. So the collection named Agatha will have exactly one member.

Test Case

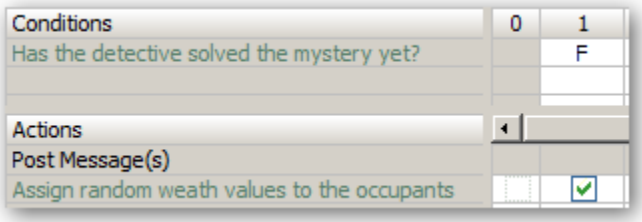


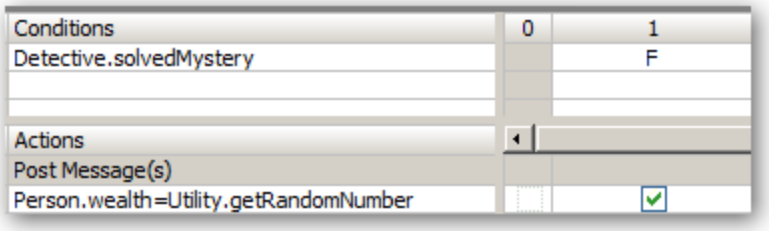
You can see in the output data structure that the Butler hates Charles and Agatha (who hates Charles and herself)

Complications

The problem is made a little trickier because nowhere do we have information about the relative wealth of the occupants of Dreadsbury Mansion. And this is required by rule 2 (The butler hates everyone not richer than Aunt Agatha)

One simple solution to this is to add a rule sheet that assigns random values for wealth:





Then running the rules a few times in the tester will result in a solution with only one killer

However, depending on the relative wealth, different results are possible. This is why there is an additional rule “No one hates everyone”. It’s possible (when Agatha is richer than the others) for the Butler to hate everyone (which is prohibited by the rules) – so in this case we need to adjust the wealth.

In this solution we do it simply by rerunning the rules. But it could be done under the control of the rules too.

Incidentally this gives us a clue as to why Agatha killed herself – she is poorer than the butler and Charles (presumable the gardener or the chauffeur)