Corticon Rule Modeling Challenge Xmas 2017 Santa's Reindeer

Rules are listed in the appendix

The Approach

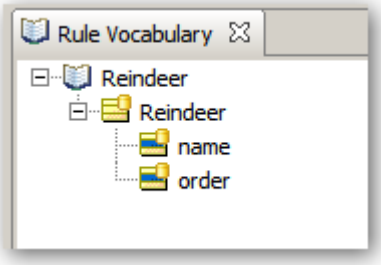
There are 15 rule statements that specify the conditions for the order of the reindeer. One approach is to write them as constraints to test all possible sequences of reindeer to locate a sequence which satisfies all the rules. "Monkey Business" from Nov 2015 is a good example of this approach.

An alternative approach is to start with an arbitrary sequence of reindeer and use the rules to see if any reindeer is breaking the rules. If it is then we adjust its position. We repeat the process until no reindeer breaks any of the rules. This approach enables us to solve the problem using a standard decision table approach with production rules. In fact, exactly the 15 rules specified in the problem.

The basic idea is to assign a sequence number to each reindeer (initially 1) and then compare reindeer two at a time. If two reindeer are found to be out order (i.e. they do not conform to the rules) then the action of the rules will adjust the sequence number.

And, because modifying a reindeer's sequence number may cause that reindeer to break other rules, we'll need to repeat the process of applying the rules until no more changes are made to the sequence numbers. Then we will have a solution.

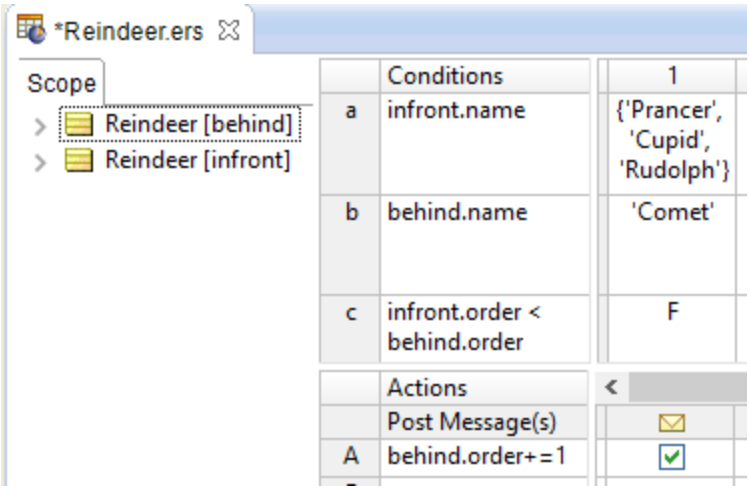
The vocabulary is very simple. It just has the reindeer name and order.



And there is just one decision table with 15 columns representing the 15 rules in the problem.

Here's what the first rule looks like (Comet must be behind Prancer, Cupid and Rudolph)

The scope section declares two aliases (infront and behind) to the collection of reindeer.



Lines (a) and (b) express the conditions laid down by the problem statement – that Comet should be behind Prancer, Cupid and Rudolph.

Condition (c) checks to see if this is in fact the case – the reindeer infront should have a lower order than the one behind. If Comet is not behind any of the other three then we move Comet back one place.

We keep applying the rule until Comet is behind all of them.

In a similar fashion we can express the infront and behind requirements of the other 14 rules.

Basically each rule says what to do if the specific reindeer are found out of sequence – namely bump up the sequence number of the reindeer that is out of sequence.

The complete set of 15 rules in a single decision table is shown below. I've just split it into two parts so it fits better on the page. Visually it shows which reindeer is supposed to be in front.

*Reindeer.ers

	Conditions	1	2	3	4	5	6	7	8
a	infront.name	{'Prancer', 'Cupid', 'Rudolph'}	'Cupid'	'Blitzen'	'Cupid'	{'Prancer', 'Dasher', 'Vixen'}	'Prancer'	'Rudolph'	'Vixen'
b	behind.name	'Comet'	'Blitzen'	{'Vixen', 'Donder', 'Dancer'}	{'Blitzen', 'Vixen', 'Comet'}	'Donder'	'Rudolph'	{'Dasher', 'Donder', 'Dancer'}	{'Comet', 'Dancer'}
c	infront.order < behind.order	F	F	F	F	F	F	F	F
	Actions								
	Post Message(s)	✉	✉	✉	✉	✉	✉	✉	✉
A	behind.order+=1	☑	☑	☑	☑	☑	☑	☑	☑

*Reindeer.ers

	Conditions	9	10	11	12	13	14	15
a	infront.name	{'Rudolph', 'Blitzen', 'Donder'}	'Prancer'	'Prancer'	'Dasher'	{'Cupid', 'Comet'}	'Cupid'	{'Prancer', 'Rudolph', 'Dasher'}
b	behind.name	'Dancer'	{'Cupid', 'Blitzen', 'Donder'}	'Dasher'	{'Blitzen', 'Vixen', 'Dancer'}	'Donder'	{'Rudolph', 'Dancer'}	'Vixen'
c	infront.order < behind.order	F	F	F	F	F	F	F
	Actions							
	Post Message(s)	✉	✉	✉	✉	✉	✉	✉
A	behind.order+=1	☑	☑	☑	☑	☑	☑	☑

The decision table will also be marked as "iterate" which tells Corticon to keep applying the decision table until nothing more changes. This is done on the rule flow:

That's all there is to the Corticon solution; a simple declarative representation of the problem which can be executed to produce a solution. The inference engine does all the real work.

How it Works

By specifying two aliases (infront and behind) to the collection of Reindeer, Corticon will automatically try every possible combination (9 x 9 = 81) of two reindeer from the set of reindeer (this is a lot less than trying every possible sequence of nine reindeer 9 x 8 x 7 x 6 x 5 x 4 x 3 x 2 = 4,376,970). The rules then modify the sequence number of one of the reindeer if it is found to break the rules. This process repeats until no more rules fire (i.e. no reindeer is out of sequence). If we want to get a detailed explanation of what's happening when the rules are executing we can attach rule messages such as:

Ref		Post	Alias	Text
1		Info	behind	R1. Comet is not behind {infront.name} so move Comet back one place
2		Info	behind	R2. Blitzen is not behind {infront.name} so move Blitzen back one place
3		Info	behind	R3. Blitzen is not in front of {behind.name} so move {behind.name} back one place
4		Info	behind	R4. Cupid is not front of {behind.name} so move {behind.name} back one place
5		Info	behind	R5. Donder is not behind {infront.name} so move Donder back one place
6		Info	behind	R6. Rudolph is not behind Prancer so move Rudolph back one place
7		Info	behind	R7. Rudolph is not in front of {behind.name} so move {behind.name} back one place
8		Info	behind	R8. Vixen is not in front of {behind.name} so move {behind.name} back one place
9		Info	behind	R9. Dancer is not behind {infront.name} so move Dancer back one place
10		Info	behind	R10 Prancer is not in front of {behind.name} so move {behind.name} back one place
11		Info	behind	R11. Dasher is not behind Prancer so move Dasher back one place
12		Info	behind	R12. Dasher is not in front of {behind.name} so move {behind.name} back one place
13		Info	behind	R13. Donder is not behind {infront.name} so move Donder back one place
14		Info	behind	R14. Cupid is not in front of {behind.name} so move {behind.name} back one place
15		Info	behind	R15. Vixen is not behind {infront.name} so move Vixen back one place

The attribute inside the {} will be replaced by the actual value during execution.

For example here's a partial listing of the actions taken by the various rules (given that all the reindeer are initially at position 1):

Message
R1. Comet is not behind Rudolph so move Comet back one place
R2. Blitzen is not behind Cupid so move Blitzen back one place
R3. Blitzen is not in front of Vixen so move Vixen back one place
R3. Blitzen is not in front of Dancer so move Dancer back one place
R3. Blitzen is not in front of Donder so move Donder back one place
R5. Donder is not behind Vixen so move Donder back one place
R6. Rudolph is not behind Prancer so move Rudolph back one place
R7. Rudolph is not in front of Dasher so move Dasher back one place
R7. Rudolph is not in front of Dancer so move Dancer back one place
R8. Vixen is not in front of Comet so move Comet back one place
R9. Dancer is not behind Donder so move Dancer back one place
R10 Prancer is not in front of Cupid so move Cupid back one place

Corticon solves the problem in 36 moves requiring three passes of the decision table. (So 3x9x9 reindeer pairings)

Test Case

The final order is

1 Prancer
2 Cupid
3 Rudolph
4 Dasher
5 Blitzen
6 Vixen
7 Comet
8 Donder
9 Dancer

Playing with the Rule Model

If we apply the Corticon compression feature to the rule sheet we'll see that Corticon is automatically able to express the same logic in fewer rules. 10 rules instead of 15.

Rules 2, 4 and 14 can be merged since they all apply to Cupid

Rules 6, 10 and 11 can be merged since the all apply to Prancer.

Rules 5 and 13 can be merged since they all apply to Donder.

Conditions	1	2	3	4	5	6	7	8	9	10
infront.name	{'Cupid', 'Prancer', 'Rudolph'}	'Cupid'	'Blitzen'	{'Comet', 'Cupid', 'Dasher', 'Prancer', 'Vixen'}	'Prancer'	'Rudolph'	'Vixen'	{'Blitzen', 'Donder', 'Rudolph'}	'Dasher'	{'Dasher', 'Prancer', 'Rudolph'}
behind.name	'Comet'	{'Blitzen', 'Comet', 'Dancer', 'Rudolph', 'Vixen'}	{'Dancer', 'Donder', 'Vixen'}	'Donder'	{'Blitzen', 'Cupid', 'Dasher', 'Donder', 'Rudolph'}	{'Dancer', 'Dasher', 'Donder'}	{'Comet', 'Dancer'}	'Dancer'	{'Blitzen', 'Dancer', 'Vixen'}	'Vixen'
infront.order < behind.order	F	F	F	F	F	F	F	F	F	F
Actions										
Post Message(s)	✉	✉	✉	✉	✉	✉	✉	✉	✉	✉
behind.order+=1	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑

Appendix

Santa always leaves plans for his elves to determine the order in which the reindeer will pull his sleigh. This year, for the European leg of his journey, his elves are working to the following schedule, which will form a single line of nine reindeer

The Rules

1. Comet behind Rudolph, Prancer and Cupid.
2. Blitzen behind Cupid
3. Blitzen in front of Donder, Vixen and Dancer.
4. Cupid in front of Comet, Blitzen and Vixen.
5. Donder behind Vixen, Dasher and Prancer.
6. Rudolph behind Prancer
7. Rudolph in front of Donder, Dancer and Dasher.
8. Vixen in front of Dancer and Comet.
9. Dancer behind Donder, Rudolph and Blitzen.
10. Prancer in front of Cupid, Donder and Blitzen.
11. Dasher behind Prancer
12. Dasher in front of Vixen, Dancer and Blitzen.
13. Donder behind Comet and Cupid.
14. Cupid in front of Rudolph and Dancer.
15. Vixen behind Rudolph, Prancer and Dasher.

Create a rule model that determines the order of the reindeer.