CHAPTER 7

Relational Analysis

7.1 Introduction

In Relational Logic, it is possible to analyze the properties of sentences in much the same way as in Propositional Logic. Given a sentence, we can determine its validity, satisfiability, and so forth by looking at possible truth assignments. And we can confirm logical entailment or logical equivalence of sentences by comparing the truth assignments that satisfy them and those that don't.

The main problem in doing this sort of analysis for Relational Logic is that the number of possibilities is even larger than in Propositional Logic. For a language with n object constants and m relation constants of arity k, the Herbrand base has $m*n^k$ elements; and consequently, there are 2^{m*n^k} possible truth assignments to consider. If we have 10 objects and 5 relation constants of arity 2, this means 2^{500} possibilities.

Fortunately, as with Propositional Logic, there are some shortcuts that allow us to analyze sentences in Relational Logic without examining all of these possibilities. In this chapter, we start with the truth table method and then look at some of these more efficient methods.

7.2 Truth Tables

As in Propositional Logic, it is in principle possible to build a truth table for any set of sentences in Relational Logic. This truth table can then be used to determine validity, satisfiability, and so forth or to determine logical entailment and logical equivalence.

As an example, let us assume we have a language with just two object constants a and b and two relation constants p and q. Now consider the sentences shown below, and assume we want to know whether these sentences logically entail $\exists x.q(x)$.

$$p(a) \vee p(b)$$
$$\forall x.(p(x) \Rightarrow q(x))$$

A truth table for this problem is shown below. Each of the first four columns represents one of the elements of the Herbrand base for this language. The two middle columns represent our premises, and the final column represents the conclusion.

$p(a)$	$p(b)$	$q(a)$	$q(b)$	$p(a) \vee p(b)$	$\forall x.(p(x) \Rightarrow q(x))$	$\exists x.q(x)$
1	1	1	1	1	1	1
1	1	1	0	1	0	1
1	1	0	1	1	0	1
1	1	0	0	1	0	0
1	0	1	1	1	1	1
1	0	1	0	1	1	1
1	0	0	1	1	0	1
1	0	0	0	1	0	0
0	1	1	1	1	1	1
0	1	1	0	1	0	1
0	1	0	1	1	1	1
0	1	0	0	1	0	0
0	0	1	1	0	1	1
0	0	1	0	0	1	1
0	0	0	1	0	1	1
0	0	0	0	0	1	0

Looking at the table, we see that there are 12 truth assignments that make the first premise true and nine that make the second premise true and five that make them both true (rows 1, 5, 6, 9, and 11). Note that every truth assignment that makes both premises true also makes the conclusion true. Hence, the premises logically entail the conclusion.

7.3. Semantic Trees

While the Truth Table method works in principle, it is impractical when the tables get very large. As with Propositional Logic, we can sometimes avoid generating such tables by incrementally constructing the corresponding "semantic trees". By interleaving unit propagation and simplification with tree generation, we can often prune away unrewarding subtrees before they are generated and thereby reduce the size of the trees.

7.4. Boolean Models

Truth tables and semantic trees are good ways of explicitly representing multiple models for a set of sentences. In some cases, there is just one model.

In this approach, we write out an empty table for each relation and then fill in values based on the constraints of the problem. For example, for any unit constraint, we can immediately enter the corresponding truth value in the appropriate box. Given these partial assignments, we then simplify the constraints (as in the semantic trees method), possibly leading to new unit constraints. We continue until there are no more unit constraints.

As an example, consider the Sorority problem introduced in Chapter 1. We are given the constraints shown below, and we want to know whether Dana likes everyone that Bess likes. In other words, we want to confirm that, in every model that satisfies these sentences, Dana likes everyone that Bess likes.

Dana likes Cody.
Abby does not like Dana.
Dana does not like Abby.
Abby likes everyone that Bess likes.
Bess likes Cody or Dana.
Abby and Dana both dislike Bess.
Cody likes everyone who likes her.
Nobody likes herself.

In this particular case, it turns out that there is just one model that satisfies all of these sentences. The first step in creating this model is to create an empty table for the *likes* relation.

	Abby	**Bess**	**Cody**	**Dana**
Abby				
Bess				
Cody				
Dana				

The data we are given has three units - the fact that Dana likes Cody and the facts that Abby does not like Dana and Dana does not like Abby. Using this information we can refine our model by putting a one into the third box in the fourth row and putting zeros in the fourth box of the first row and the first box of the fourth row.

	Abby	**Bess**	**Cody**	**Dana**
Abby				**0**
Bess				
Cody				
Dana	**0**		**1**	

Now, we know that Abby likes everyone that Bess likes. If Bess likes Dana, then we could conclude that Abby likes Dana as well. We already know that Abby does not like Dana, so Bess must not like Dana either.

	Abby	Bess	Cody	Dana
Abby				**0**
Bess				**0**
Cody				
Dana	**0**		**1**	

At the same time, we know that Bess likes Cody or Dana. Since Bess does not like Dana, she must like Cody. Once again using the fact that Abby likes everyone whom Bess likes, we know that Abby also likes Cody.

	Abby	Bess	Cody	Dana
Abby			**1**	**0**
Bess			**1**	**0**
Cody				
Dana	**0**		**1**	

Abby and Dana both dislike Bess. Using this fact we can add 0s to the first and last cells of the second column.

	Abby	Bess	Cody	Dana
Abby		**0**	**1**	**0**
Bess			**1**	**0**
Cody				
Dana	**0**	**0**	**1**	

On the other hand, Cody likes everyone who likes her. This allows us to put a 1 in every column of the third row where there is a 1 in the corresponding rows of the third column.

	Abby	Bess	Cody	Dana
Abby		0	1	0
Bess			1	0
Cody	1	1		1
Dana	0	0	1	

Since nobody likes herself, we can put a 0 in each cell on the diagonal.

	Abby	Bess	Cody	Dana
Abby	0	0	1	0
Bess		0	1	0
Cody	1	1	0	1
Dana	0	0	1	0

Finally, using the fact that Abby likes everyone that Bess likes, we conclude that Bess does not like Abby. (If she did then Abby would like herself, and we know that that is false.)

	Abby	Bess	Cody	Dana
Abby	0	0	1	0
Bess	0	0	1	0
Cody	1	1	0	1
Dana	0	0	1	0

At this point, we have a complete model, and we can check our conclusion to see that this model satisfies the desired conclusion. In this case, it is easy to see that Dana indeed does like everyone that Bess likes.

We motivated this method by talking about cases where the given sentences have a unique model, as in this case. However, the method can also be of value even when there are multiple possible

models. For example, if we had left out the belief that Cody likes everyone who likes her, we would still have eight models (corresponding to the eight possible combinations of feelings Cody has for Abby, Bess, and Dana). Yet, even with this ambiguity, it would be possible to determine whether Dana likes everyone Bess likes using just the portion of the table already filled in.

7.5 Non-Boolean Models

As defined in Chapter 6, a model in Relational Logic is an assignment of truth values to the ground atoms of our language. We treat each ground atom in our language as a variable and assign it a single truth value (1 or 0). In general, this is a good way to proceed. However, we can sometimes do better.

Consider, for example, a theory with four object constants and two unary relation constants. In this case, there would be eight elements in the Herbrand base and 2^8 (256) possible truth assignments. Now, suppose we had the constraint that each relation is true of at most a single object. Most of these assignments would not satisfy the single value constraint and thus considering them is wasteful.

Luckily, in cases like this, there is a representation for truth assignments that allows us to eliminate such possibilities and thereby save work. Rather than treating each *ground atom* as a separate variable with its own Boolean value, we can think of each *relation* as a variable with 4 possible values. In order to analyze sentences in such a theory, we would need to consider only 4^2 (16) possibilities.

Even if we search the entire space of assignments, this a significant saving over the pure truth table method. Moreover, we can combine this representation with the techniques described earlier to find assignments for these non-Boolean variables in an even more efficient manner.

The game of Sukoshi illustrates this technique and its benefits. (Sukoshi is similar to Sudoku, but it is smaller and simpler.) A typical Sukoshi puzzle is played on a 4x4 board. In a typical instance of Sukoshi, several of the squares are already filled, as in the example below. The goal of the game is to place the numerals 1 through 4 in the remaining squares of the board in such a way that no numeral is repeated in any row or column.

	4		**1**
2			
			3
		4	

We can formalize the rules of this puzzle in the language of Logic. Once we have done that, we can use the techniques described here to find a solution.

In our formalization, we use the expression *cell*(1,2,3) to express the fact that the cell in the first row and the second column contains the numeral 3. For example, we can describe the initial board shown above with the following sentences.

cell(1,2,4)
cell(1,4,1)
cell(2,1,2)
cell(3,4,3)
cell(4,3,4)

We use the expression *same*(*x*,*y*) to say that *x* is the same as *y*. We can axiomatize *same* by simply stating when it is true and where it is false. An abbreviated axiomatization is shown below.

same(1,1)	¬*same*(2,1)	¬*same*(3,1)	¬*same*(4,1)
¬*same*(1,2)	*same*(2,2)	¬*same*(3,2)	¬*same*(4,2)
¬*same*(1,3)	¬*same*(2,3)	*same*(3,3)	¬*same*(4,3)
¬*same*(1,4)	¬*same*(2,4)	¬*same*(3,4)	*same*(4,4)

Using this vocabulary, we can write the rules defining Sukoshi as shown below. The first sentence expresses the constraint that two cells in the same row cannot contain the same value. The second sentence expresses the constraint that two cells in that same column cannot contain the same value. The third constraint expresses the fact that every cell must contain at least one value.

$$\forall x.\forall y.\forall z.\forall w.(cell(x,y,w) \wedge cell(x,z,w) \Rightarrow same(y,z))$$
$$\forall x.\forall y.\forall z.\forall w.(cell(x,z,w) \wedge cell(y,z,w) \Rightarrow same(x,y))$$
$$\forall x.\forall y.\exists w.cell(x,y,w)$$

As a first step in solving this problem, we start by focussing on the fourth column, since two of the cells in that column are already filled. We know that there must be a 4 in one of the cells. It cannot be the first, since that cell contains a 1, and it cannot be the third since that cell contains a 3. It also cannot be the fourth, since there is already a 4 in the third cell of the fourth row. By process of elimination, the 4 must go in the fourth cell of the second row, leading to the board shown below.

	4		1
2			4
			3
		4	

At this point, there is a four in every row and every column except for the first column in the third row. So, we can safely place a four in that cell.

	4		1
2			4
4			3
		4	

Since there is just one empty cell in the fourth column, we know it must be filled with the single remaining value, viz. 2. After adding this value, we have the following board.

	4		1
2			4
4			3
		4	2

Now, let's turn our attention to the first column. We know that there must be a 1 in one of the cells. It cannot be the first, since there is already a 1 in that row, and it cannot be the second or third

since those cell already contain values. Consequently, the 1 must go in the first cell of the fourth row.

	4		1
2			4
4			3
1		4	2

Once again, we have a column with all but one cell filled. Column 1 has a 2 in the second cell, a 4 in the third, and a 1 in the fourth. So, we can place a 3 in the first cell of that column.

3	4		1
2			4
4			3
1		4	2

At this point, we can fill in the single empty cell in the first row, leading to the following board.

3	4	2	1
2			4
4			3
1		4	2

And we can fill in the single empty cell in the fourth row as well.

3	4	2	1
2			4
4			3
1	3	4	2

Now, let's consider the second column. We cannot put a 2 in the second cell, since there is already a 2 in that row. Since the first and last cells are already full, the only option is to put the 2 into the third cell.

3	4	2	1
2			4
4	2		3
1	3	4	2

Finishing off the third row leads to the board below.

3	4	2	1
2			4
4	2	1	3
1	3	4	2

Finishing off the third column leads to the following board.

3	4	2	1
2		3	4
4	2	1	3
1	3	4	2

Finally, we can place a 1 in the second cell of the second row. And, with that, the board is full. We have a distinct numeral in every row and every column, as required by the rules.

3	4	2	1
2	1	3	4
4	2	1	3
1	3	4	2

Given the initial assignment in this case, it is fairly easy to find a complete assignment that satisfies the Sukoshi constraints. For other initial assignments, solving the problem is more difficult. However, the techniques described here still work to cut down on the amount of work necessary. In fact, virtually all Sukoshi puzzles can be solved using these techniques without any form of trial and error.

Exercises

Exercise 7.1: Mr. Red, Mr. White, and Mr. Blue meet for lunch. Each is wearing a red shirt, a white shirt, or a blue shirt. No one is wearing more than one color, and no two are wearing the same color. Mr. Blue tells one of his companions, "Did you notice we are all wearing shirts with different color from our names?", and the other man, who is wearing a white shirt, says, "Wow, that's right!" Use the Boolean model technique to figure out who is wearing what color shirt.

Exercise 7.2: Amy, Bob, Coe, and Dan are traveling to different places. One goes by train, one by car, one by plane, and one by ship. Amy hates flying. Bob rented his vehicle. Coe tends to get seasick. And Dan loves trains. Use the Boolean models method to figure out which person, used which mode of transportation.

Exercise 7.3: Sudoku is a puzzle consisting of a 9x9 board divided into nine 3x3 subboards. In a typical puzzle, several of the squares are already filled, as in the example shown below. The goal of the puzzle is to place the numerals 1 through 9 into the remaining squares of the board in such a way that no numeral is repeated in any row or column or 3x3 subboard.

5	8	6					1	2
				5	2	8	6	
2	4		8	1				3
			5		3		9	
				8	1	2	4	
4		5	6			7	3	8
	5		2	3			8	1
7					8			
3	6				5			

Use the techniques described in the Chapter to solve this puzzle.