ENCODING PROBLEMS IN PROPOSITIONAL LOGIC

1. Overview

As we have seen in lectures, propositional logic is a mathematical theory governed by rules; however, it is also a language—albeit a very formal and restrictive one—that can be used for expressing and solving real world problems. Although learning the rules of formal logic is a relatively clear-cut task, the process of encoding problems in propositional logic is far more of an art. It is this encoding process that we examine here.

2. An Example

Three friends, Alice, Bob, and Cindy, are going to watch the latest Japanese monster film. As the cinema is reasonably empty, they are able to get good seats in the middle row, and have some room to spread out. Within their row, they can sit anywhere within the four consecutive seats 13, 14, 15, and 16. Our task is to distribute the three friends amongst the four seats.

To express this problem using propositional logic, we must firstly identify the basic information that we shall work with. Since we are only concerned about the friends' seating arrangements, and not extraneous information such as the film's length, country of origin, or the address of the cinema, we refer only to seats and friends: the three friends will be denoted with A, B, and C, while the seat numbers will denote the seats. As for our propositions, we shall let $P_{F,s}$ be the statement "Friend F is seated in seat s," for each friend F and each seat s.

Now that we have our raw materials, we need to think about how we are going to put them together. In total, there are 12 propositions (3 friends by 4 seats) to consider, which gives 2^{12} different ways to assign truth values to all 12 propositions. Although it is easy to express a *particular* seating of the friends with such an assignment, many other assignments will not make sense. In particular, when assigning truth values, we can't take any physically obvious features for granted. For example, having $P_{A,13}$ and $P_{A,14}$ both true would mean that Alice is sitting in seats 13 and 14, which is clearly nonsense, yet this contradiction appears in many of the 2^{12} truth assignments that we are considering. Therefore, if we are to express the *entire* seating problem, we must lay down general logical constraints that precisely define what it means for a particular truth assignment to be a valid solution to the seating problem.

A good place to start is to summarize, in English, the constraints that govern the problem. The following rules capture the situation exactly:

(1) Each friend gets a seat.
(2) No friend occupies more than one seat.
(3) No seat is occupied by more than one friend.

As all three friends are just that—friends—there are no restrictions on whom they sit next to. Keep in mind that the rules of logic mean that these three pieces of defining information could be expressed in many different, yet equivalent, ways; we shall determine an encoding that follows naturally from the above three statements.

To keep things simple, we separately deal with each rule above. Firstly, for each friend F, having F assigned to a seat is expressed as

$$P_{F,13} \vee P_{F,14} \vee P_{F,15} \vee P_{F,16}. \tag{1}$$

Also, for each friend F, we need to ensure that F does not get assigned any extra seats. This amounts to ensuring that no pair of seats is occupied by friend F; we impose this constraint with the more lengthy expression

$$\begin{aligned} &\neg\left(P_{F,13} \wedge P_{F,14}\right) \wedge \neg\left(P_{F,13} \wedge P_{F,15}\right) \wedge \neg\left(P_{F,13} \wedge P_{F,16}\right) \wedge \\ &\neg\left(P_{F,14} \wedge P_{F,15}\right) \wedge \neg\left(P_{F,14} \wedge P_{F,16}\right) \wedge \neg\left(P_{F,15} \wedge P_{F,16}\right). \end{aligned} \tag{2}$$

The third rule can be tackled in a similar manner: for each seat s, the expression

$$\neg\left(P_{A,s} \wedge P_{B,s}\right) \wedge \neg\left(P_{A,s} \wedge P_{C,s}\right) \wedge \neg\left(P_{B,s} \wedge P_{C,s}\right) \tag{3}$$

ensures that no pair of friends occupies the seat s. Taken together, these expressions precisely encode our seating problem, and result in a total of ten individual propositions.

3. Conjunctive Normal Form

Although expressing such a problem with propositional logic is a puzzle that must be dealt with on a case-by-case basis, there is a formatting convention that we can use to standardize the logical expressions that we produce. As it turns out, expression (1) already meets the convention. Firstly, note that it is a disjunction of one or more logical variables, each of which may be negated. Such an expression is called a *clause*. More generally, an expression comprising a sequence of one or more clauses joined by conjunction is said to be in *conjunctive normal form* (CNF). This means that a lone clause, such as (1), is automatically in CNF.

Our remaining two expressions, (2) and (3), are not in CNF, but we can easily convert them to CNF using the laws of logic. Fortunately, their similarity of form means that they yield to the same conversion process. Because each expression is a sequence of negated conjunctions bound together with conjunction, we simply apply de Morgan's laws to each negated conjunction; this yields a conjunction of clauses, which is precisely CNF. Consequently, applying the conversion process to expression (2) yields

$$\begin{aligned} &\left(\neg P_{F,13} \vee \neg P_{F,14}\right) \wedge \left(\neg P_{F,13} \vee \neg P_{F,15}\right) \wedge \left(\neg P_{F,13} \vee \neg P_{F,16}\right) \wedge \\ &\left(\neg P_{F,14} \vee \neg P_{F,15}\right) \wedge \left(\neg P_{F,14} \vee \neg P_{F,16}\right) \wedge \left(\neg P_{F,15} \vee \neg P_{F,16}\right), \end{aligned}$$

whereas applying it to expression (3) yields

$$\left(\neg P_{A,s} \vee \neg P_{B,s}\right) \wedge \left(\neg P_{A,s} \vee \neg P_{C,s}\right) \wedge \left(\neg P_{B,s} \vee \neg P_{C,s}\right).$$

Using CNF to uniformly encode problems in propositional logic is an important technique in complexity theory. Later lectures will explore this approach in more detail.