Chapter 4

Decidable Theories of First-Order Logic

In this part of the book, we will look into constraint-based techniques for verification. The idea is to take a correctness property and encode it as a set of constraints. By solving the constraints, we can decide whether the correctness property holds or not.

The constraints we will use are formulas in *first-order logic* (FOL). FOL is a very big and beautiful place, but neural networks only live in a small and cozy corner of it—the corner that we will explore in this chapter.

4.1 **Propositional Logic**

We begin with the purest of all, *propositional logic*. A formula φ in propositional logic is over Boolean variables that are traditionally given the names $p, q, r, \ldots$. A formula φ is defined using the following grammar:

$$
\begin{array}{rlr}
\varphi :- & \mathsf{true} & \\
& \mathsf{false} & \\
& \mathit{var} & \text{Variable} \\
& \mid \varphi \land \varphi & \text{Conjunction (and)} \\
& \mid \varphi \lor \varphi & \text{Disjunction (or)} \\
& \mid \neg \varphi & \text{Negation (not)}
\end{array}
$$

Essentially, a formula in propositional logic defines a circuit with Boolean variables, AND gates ($\land$), OR gates ($\lor$), and NOR gates ($\neg$). At the end of the day, all programs can be defined as circuits, because everything is a bit on a computer and there is a finite amount of memory, and therefore a finite number of variables.

As an example, here is a formula $\varphi \triangleq (p \wedge q) \vee \neg r$. Observe the use of $\triangleq$; this is to denote that we are syntactically defining φ to be the formula on the right of $\triangleq$, as opposed to saying that the two formulas are semantically equivalent (more on this in a bit). We will use $fv(\varphi)$ to denote the set of *free* variables appearing in the formula. For now, it is the set of all variables that are syntactically present in the formula; for example, in $fv(\varphi) = \{p, q, r\}$.

Interpretations

Let φ be a formula over a set of variables $fv(\varphi)$. An interpretation I of φ is a map from variables $fv(\varphi)$ to true or false. For example,

$$I = \{p \mapsto \mathsf{true}, q \mapsto \mathsf{true}, r \mapsto \mathsf{false}\}$$

Given an interpretation I of a formula φ, we will use $I(\varphi)$ to denote the formula where we have replaced each variable $fv(\varphi)$ with its interpretation in I. For example, applying I above to $(p \wedge q) \vee \neg r$, we get

$$(\mathsf{true} \wedge \mathsf{true}) \vee \neg\mathsf{false}$$

Satisfiability

We will define the following evaluation or simplification rules for a formula:

$$\begin{aligned}
\mathsf{eval}(\mathsf{true}) &= \mathsf{true} \\
\mathsf{eval}(\mathsf{false}) &= \mathsf{false} \\
\\
\mathsf{eval}(\mathsf{true} \wedge \varphi) &= \mathsf{eval}(\varphi) \\
\mathsf{eval}(\varphi \wedge \mathsf{true}) &= \mathsf{eval}(\varphi) \\
\\
\mathsf{eval}(\mathsf{false} \wedge \varphi) &= \mathsf{false} \\
\mathsf{eval}(\varphi \wedge \mathsf{false}) &= \mathsf{false} \\
\\
\mathsf{eval}(\mathsf{false} \vee \varphi) &= \mathsf{eval}(\varphi) \\
\mathsf{eval}(\varphi \vee \mathsf{false}) &= \mathsf{eval}(\varphi) \\
\mathsf{eval}(\mathsf{true} \vee \varphi) &= \mathsf{true} \\
\mathsf{eval}(\varphi \vee \mathsf{true}) &= \mathsf{true}
\end{aligned}$$

$$\begin{aligned}\mathsf{eval}(\neg\mathsf{true}) &= \mathsf{false}\\ \mathsf{eval}(\neg\mathsf{false}) &= \mathsf{true}\end{aligned}$$

If a given formula has no free variables, then applying these rules repeatedly, you will get true or false. We will use $\mathsf{eval}(\varphi)$ to denote the simplest form of φ we can get by repeatedly applying the above rules.

A formula φ is *satisfiable* (SAT) if and only if there exists an interpretation I such that

$$\mathsf{eval}(I(\varphi)) = \mathsf{true}$$

in which case we will say that I is a *model* of φ and denote it

$$I \models \varphi$$

We will also use $I \not\models \varphi$ to denote that I is not a model of φ. It follows from our definitions that $I \not\models \varphi$ iff $I \models \neg\varphi$.

Equivalently, a formula φ is *unsatisfiable* (UNSAT) if and only if for every interpretation I we have $\mathsf{eval}(I(\varphi)) = \mathsf{false}$.

Validity and equivalence

To prove properties of neural networks, we will be asking *validity* questions. A formula φ is valid iff every possible interpretation I is a model of φ. It follows that a formula φ is valid if and only if $\neg\varphi$ is unsatisfiable.

We will say that two formulas, φ_1 and φ_2, are *equivalent* if and only if every model I of φ_1 is a model of φ_2, and vice versa. We will denote equivalence as $\varphi_1 \equiv \varphi_2$.

Implication and bi-implication

We will often use an *implication* $\varphi_1 \Rightarrow \varphi_2$ to denote the formula

$$\neg\varphi_1 \vee \varphi_2$$

Similarly, we will use a *bi-implication* $\varphi_1 \iff \varphi_2$ to denote the formula

$$(\varphi_1 \Rightarrow \varphi_2) \wedge (\varphi_2 \Rightarrow \varphi_1)$$

4.2 **First-Order Theories**

We can now extend propositional logic using *theories*.