1. Propositional Logic: Syntax

Logic is the study of reasoning. The British mathematician and philosopher George Boole (1815–1864) is the man who made logic mathematical. His book *The Mathematical Analysis of Logic* was published in 1847.

Logic can be used in programming, and it can be applied to the analysis and automation of reasoning about software and hardware. This is why it is sometimes considered a part of theoretical computer science. Since reasoning plays an important role in intelligent behavior, logic is closely related to artificial intelligence.

The short book by the German philosopher Gottlob Frege (1848–1925) with the long title *Ideography, a Formula Language, Modeled upon that of Arithmetic, for Pure Thought* (1879), introduced notation that is somewhat similar to what is now called first-order formulas. Frege wrote:

> I believe that I can best make the relation of my ideography to ordinary language clear if I compare it to that which the microscope has to the eye. Because of the range of its possible uses and the versatility with which it can adapt to the most diverse circumstances, the eye is far superior to the microscope... But, as soon as scientific goals demand great sharpness of resolution, the eye proves to be insufficient.
>
> ... I am confident that my ideography can be successfully used wherever special value must be placed on the validity of proofs, as for example when the foundations of the differential and integral calculus are established.

In logic, we distinguish between two languages: the one that is the object of our study and the one that we use to talk about that object. The former is called the *object language*; the latter is the *metalanguage*. In the first part of this course, the object language is the formal language of propositional formulas defined below. The metalanguage is the usual informal language of mathematics and theoretical computer science, which is a mixture of the English language and mathematical notation. The importance of distinguishing between the object language and the metalanguage was emphasized by the mathematician and logician Alfred Tarski (1902–1983), who taught logic at Berkeley since 1942.

Propositional Formulas: Syntax

A *propositional signature* is a set of symbols called *atoms*. (In examples, we will assume that p, q, r are atoms.) The symbols

$$\wedge \quad \vee \quad \rightarrow \quad \leftrightarrow \quad \neg \quad \bot \quad \top$$

are called *propositional connectives*. Among them, the symbols $\wedge$ *(conjunction)*, $\vee$ *(disjunction)*, $\rightarrow$ *(implication)* and $\leftrightarrow$ *(equivalence)* are called *2-place*, or *binary* connectives; $\neg$ *(negation)* is a *1-place*, or *unary* connective; $\bot$ *(false)* and $\top$ *(true)* are *0-place*.

Take a propositional signature σ which contains neither the propositional connectives nor the parentheses $(,)$. The alphabet of propositional logic consists of the atoms from σ, the propositional connectives, and the parentheses. By a *string* we understand a finite string of symbols in this alphabet. We define when a string is a *(propositional) formula* recursively, as follows:

- every atom is a formula,
- both 0-place connectives are formulas,
- if F is a formula then $\neg F$ is a formula,
- for any binary connective $\odot$, if F and G are formulas then $(F \odot G)$ is a formula.

For instance,

$$\neg(p \rightarrow q)$$

and

$$(\neg p \rightarrow q) \tag{1}$$

are formulas; the string

$$\neg p \rightarrow q \tag{2}$$

is not a formula. But very soon (see next page) we are going to introduce a convention according to which (2) can be used as an abbreviation for (1).

Properties of formulas can be often proved by induction. One useful method is strong induction on length (number of symbols). In such a proof, the induction hypothesis is that every formula which is shorter than F has the property P that we want to prove. From this assumption we need to derive that F has property P also. Then it follows that all formulas have property P.

In another useful form of induction, we check that all atoms and 0-place connectives have property P, and that the property is preserved when a new formula is formed using a unary or binary connective. More precisely, we show that

- every atom has property P,
- both 0-place connectives have property P,
- if a formula F has property P then so does $\neg F$,
- for any binary connective $\odot$, if formulas F and G have property P then so does $(F \odot G)$.

Then we can conclude that property P holds for all formulas. This is called "structural induction."

Prove the following assertions.

1.1 In any prefix of a formula, the number of left parentheses is greater than or equal to the number of right parentheses.

(A *prefix* of a string $a_1 \cdots a_n$ is any string of the form $a_1 \cdots a_m$ where $0 \leq m \leq n$.)

1.2 Every prefix of a formula F

- is a string of negations (possibly empty), or
- has more left than right parentheses, or
- equals F.

1.3 No formula can be represented in the form $(F \odot G)$, where F and G are formulas and $\odot$ is a binary connective, in more than one way.

By representing a formula in the form $\neg F$ or $(F \odot G)$ we start "parsing" it. The assertion of the previous problem shows that a formula can be parsed in only one way.

From now on, we will abbreviate formulas of the form $(F \odot G)$ by dropping the outermost parentheses in them. We will also agree that $\leftrightarrow$ has a lower binding power than the other binary connectives. For instance,

$$p \vee q \leftrightarrow p \rightarrow r$$

will be viewed as shorthand for

$$((p \vee q) \leftrightarrow (p \rightarrow r)).$$

Finally, for any formulas $F_1, F_2, \ldots, F_n$ $(n > 2)$,

$$F_1 \wedge F_2 \wedge \cdots \wedge F_n$$

will stand for

$$(\cdots (F_1 \wedge F_2) \wedge \cdots \wedge F_n).$$

The abbreviation $F_1 \vee F_2 \vee \cdots \vee F_n$ will be understood in a similar way.