

First-Order Logic

We consider a language with variables and the quantifiers “for all” and “there exists.” We enrich this language with predicates and functions.

1. Motivation

1.1. Quantifiers

The true/false logic of the previous chapters is ultimately rather limited and cannot express the natural statements found in a mathematics course. In this chapter, we consider a more powerful language: first-order logic. It allows us to write the following statements:

- “ $\forall n \in \mathbb{Z} \quad n^2 \geq 0$ ”: “the square of every integer is nonnegative.”
- “ $\exists z \in \mathbb{C} \quad z^2 = -1$ ”: “there exists a complex number whose square is -1 .”
- “ $\forall g \in G \quad \exists g' \in G \quad g \circ g' = e$ ”: “every element of G has an inverse,” for a group $(G, \circ)$ with identity element e .

With this language, we can define complicated notions with subtle nuances. For example, let $f : \mathbb{R} \rightarrow \mathbb{R}$ be a function.

- Continuity on $\mathbb{R}$ is defined by:

$$\forall x \in \mathbb{R} \quad \forall \varepsilon > 0 \quad \exists \delta > 0 \quad \forall y \in \mathbb{R} \quad |x - y| < \delta \rightarrow |f(x) - f(y)| < \varepsilon$$

- Whereas uniform continuity is defined by:

$$\forall \varepsilon > 0 \quad \exists \delta > 0 \quad \forall x \in \mathbb{R} \quad \forall y \in \mathbb{R} \quad |x - y| < \delta \rightarrow |f(x) - f(y)| < \varepsilon$$

1.2. New features

The main new feature is, of course, the appearance of the quantifiers “ $\forall$ ” and “ $\exists$.” The single statement “ $\forall n \in \mathbb{N} \quad n^2 \geq 0$ ” expresses infinitely many truths. In true/false logic, we would not know how to prove such a claim, because we would have to consider the case $n = 0$, then $n = 1$, and so on, requiring infinitely many steps. Here we treat the statement “ $\forall n \in \mathbb{N} \quad n^2 \geq 0$ ” as a single claim to be proved.

The second advantage of first-order logic is that it applies to a wide variety of situations. In propositional logic, we considered formulas, and satisfaction consisted in replacing the variables with the values T/F. Here we still want to decide whether a formula is true or false, but the variables correspond to more general objects. For example, consider the formula:

$$(\forall x \quad P(x)) \vee (\exists x \quad \neg P(x))$$

It applies to x in any set, and also to any predicate $P(x)$. In words, it means “a property is true for every element, or there exists an element for which it is not true.” Here is what this formula becomes in three different structures:

- $(\forall x \in \mathbb{R} \quad x^2 \neq 0) \vee (\exists x \in \mathbb{R} \quad x^2 = 0)$
- $(\forall n \in \mathbb{N} \quad n + 1 > n) \vee (\exists n \in \mathbb{N} \quad n + 1 \leq n)$
- $(\forall g \in G \quad g \circ g = e) \vee (\exists g \in G \quad \neg(g \circ g = e))$, in a group G .

First-order languages adapt to a wide variety of structures (sets, groups, fields, vector spaces...) and to the corresponding proofs.

1.3. Limitations

First-order logic is powerful; however, some things cannot be expressed in these languages. The most striking limitation is that there is no formula expressing whether a set is finite (which means, for example, that we cannot define the dimension of a vector space). Working in first-order logic means that we consider elements in a set E , written “ $x \in E$,” but we cannot quantify over subsets $A \subset E$. Thus, the statement “every nonempty subset has a least element” (which is the definition of a *well-ordered* set) is expressed in *second-order logic*, because it concerns subsets rather than elements. In conclusion, first-order logic describes local properties of objects well, but it is less effective at describing global properties related to infinity.

2. Language

A first-order language should be understood as a common general framework that we will then specialize to each situation (integers, real numbers, groups...).

2.1. Elements of the language

The formulas of first-order logic are constructed from languages made up of the following elements.

Logical symbols

- **variables**: $x, y, z, x_1, x_2, \dots$
- parentheses: (and)
- connectives: $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$
- **quantifiers**: $\forall$ (for all), $\exists$ (there exists)

Nonlogical symbols

- **constants**: $a, b, c, c_1, c_2, \dots$
- **predicates**: $P(x), P(x, y), \dots, P(x_1, \dots, x_n)$; including equality $x = y$
- **functions**: $f(x), f(x, y), \dots, f(x_1, \dots, x_n)$

We discuss the new notions in detail below. We will first consider languages without functions; we will add them at the end of this chapter.

2.2. Logical symbols

The language is much richer than that of true/false logic. Of course, we again find the logical connectives (and parentheses), which will play the same role here as in propositional logic. Let us examine the new features in detail.

- **Universal quantifier** $\forall$. It is an upside-down A, introduced by Gerhard Gentzen in 1935 as the initial letter of *all* or *alle*. It is read “for all” or “for every.”
- **Existential quantifier** $\exists$. It is a backward E, introduced by Giuseppe Peano at the end of the nineteenth century. It is read “there exists” or “there is at least one.”
- **Variables.** They are used to discuss the objects under consideration without naming them individually. Thus, “ $\forall x P(x)$ ” or “ $\exists x P(x)$ ” applies to every x (or to some x), but there is not really an object named x . This is called a **bound variable** (or a *dummy variable*). Thus, “ $\forall y P(y)$ ” would have the same meaning.

Note: for variables, we reserve the letters at the end of the alphabet, x, y, z , with subscripts if needed.

2.3. Nonlogical symbols

This is another new feature of first-order logic. We want constructions and theories that apply to many situations. Constants, predicates, and functions are like empty shells whose meaning must be specified in each application.

- **Constants.** Written $a, b, c, c_1, c_2, \dots$, these are symbols of the language that will distinguish certain elements. For example, we can define a language suited to groups with a constant c that will play the role of the identity element $e, 0, 1, \dots$ depending on the group under consideration. When defining the language, we choose the names and number of constants.
- **Predicates.** A predicate is somewhat like a super-proposition in true/false logic: it returns a T/F value, but that value depends on the values of one or more parameters. Their symbols in the language will often be P, Q , or R .
 - Unary predicate P , written $P(x)$: “does x have property P ?” (this may be true or false depending on the value assigned to x).
For example, such a predicate could mean “ $x^2 \geq x$,” “ $x^2 = 1$,” or “ $x^5 + x^3 + 1 = 0$ ” ... It may be true or false depending on x .
 - Binary predicate P , written $P(x, y)$: “does (x, y) have property P ?” This can also be understood as “is x related to y by P ?”
For example, a binary predicate could mean “ $x < y$ ” or “ $x^2 + y^2 = 1$ ”...
 - **Equality** “ $x = y$ ” is a special binary predicate. Unless otherwise stated (which will be rare), we will always assume that equality is present in the language.
 - An n -ary predicate $P(x_1, \dots, x_n)$ (also called an n -place predicate).

Once again, the concrete meaning of each predicate must be specified in each structure. At the level of the language, it is just a symbol P , but we will allow ourselves to write it as $P(x)$ or $P(x_1, \dots, x_n)$ to make the number of places clear.

- **Functions.** At the end of the chapter, we will add functions to the language. For example, $f(x)$ (which may represent $f(x) = 2x + 3$ or $f(x) = 1/x, \dots$), and $f(x, y)$ (for $f(x, y) = x + y$ or $f(x, y) = x \times y, \dots$). Functions make the definition of a formula somewhat more technical, so for now our languages will not have functions.

3. Formulas

3.1. Definitions

Because the language is rich, there is a specific vocabulary that must be mastered.

Let $\mathcal{L}$ be a language without functions.

Definition.

A **term** of a language is a variable or a constant.

Remark: we will modify this definition for languages with functions (see the end of the chapter).

Definition.

An **atomic formula** is a word of the form $P(t_1, \dots, t_n)$, where P is an n -ary predicate and $t_1, \dots, t_n$ are terms.

Definition.

The set of **formulas** $\text{Form}(\mathcal{L})$ of the language is constructed according to the following rules:

- atomic formulas are formulas,
- if $\mathcal{P}$ and $\mathcal{Q}$ are formulas, then $\neg\mathcal{P}$, $(\mathcal{P} \wedge \mathcal{Q})$, $(\mathcal{P} \vee \mathcal{Q})$, $(\mathcal{P} \rightarrow \mathcal{Q})$, $(\mathcal{P} \leftrightarrow \mathcal{Q})$ are also formulas,
- if x is a variable and $\mathcal{P}$ is a formula, then

$$\forall x \mathcal{P} \quad \text{and} \quad \exists x \mathcal{P}$$
 are also formulas,
- no other words are formulas.

3.2. Remarks

- The expression “ $\forall x \mathcal{P}$ ” is a word of the language, and in this language x is an abstract variable, not an element of a set. When we later interpret the formula, we will define a domain E , for example $\mathbb{N}$, $\mathbb{R}$, G , ... and then the formula will become “ $\forall x \in \mathbb{N} \mathcal{P}$ ” or “ $\forall x \in \mathbb{R} \mathcal{P}$,” etc.
- The interesting case is, of course, when the formula $\mathcal{P}$ actually depends on the variable x . We will often allow ourselves to write a formula as $\mathcal{P}(x)$ to show its dependence on the variable x . Thus, we will usually write “ $\forall x \mathcal{P}(x)$ ” or “ $\forall x \exists y \mathcal{P}(x, y)$ ”...
- As usual, we will allow ourselves to add and remove parentheses to make formulas clearer. For example, we will write “ $(\forall x \mathcal{P}) \rightarrow (\exists y \mathcal{Q})$ ” instead of the official formula “ $(\forall x \mathcal{P} \rightarrow \exists y \mathcal{Q})$.”

The definition of a formula given above was the inductive definition (“from above”) of $\text{Form}(\mathcal{L})$. We can also give a recursive definition (“from below”).

We define $\text{Form}(\mathcal{L}) = \bigcup_{n \geq 0} \text{Form}_n(\mathcal{L})$, with:

$\text{Form}_0(\mathcal{L})$ is the set of atomic formulas

and then, for $n \geq 0$:

$$\begin{aligned} \text{Form}_{n+1}(\mathcal{L}) = & \text{Form}_n(\mathcal{L}) \\ & \cup \{ \neg \mathcal{P} \mid \mathcal{P} \in \text{Form}_n(\mathcal{L}) \} \\ & \cup \{ (\mathcal{P} \wedge \mathcal{Q}) \mid \mathcal{P}, \mathcal{Q} \in \text{Form}_n(\mathcal{L}) \} \\ & \cup \dots \\ & \cup \{ \forall x_k \mathcal{P} \mid x_k \text{ a variable, } \mathcal{P} \in \text{Form}_n(\mathcal{L}) \} \\ & \cup \{ \exists x_k \mathcal{P} \mid x_k \text{ a variable, } \mathcal{P} \in \text{Form}_n(\mathcal{L}) \} \end{aligned}$$

where, of course, the union also includes the connectives $\vee$, $\rightarrow$, $\leftrightarrow$, and where $x_1, x_2, \dots$ denotes the set of variables of $\mathcal{L}$.

4. Examples

Let us get a little ahead of the next chapter by looking at examples of structures in order to better understand the preceding notions.

4.1. With integers

We want to model the natural numbers with a zero and an order relation. We define a language $\mathcal{L} = (c, P)$ with a constant c and a binary predicate P , also written $P(x, y)$ (the equality predicate $x = y$ is also included by default). Strictly speaking, we should also specify the names of the variables, but we will use the naming convention $x, y, \dots$ throughout.

Here is a formula in the language:

$$\forall x \exists y \ P(x, y)$$

To use this language, we specify a domain, for example $E = \mathbb{N}$ here. We must then define all the nonlogical symbols:

- the constant: $c = 0$,
- the predicate: $P(x, y)$: “ $x < y$.”

The logical symbols, on the other hand, do not need to be specified: the variable x remains a variable, the connective $\wedge$ remains AND, etc. We have thus defined a **structure**, denoted by $\mathcal{S}$. In this structure, the preceding formula is interpreted as:

$$\forall x \in \mathbb{N} \exists y \in \mathbb{N} \ x < y,$$

that is, “for every natural number, there exists an even larger natural number.”

For the same language $\mathcal{L}$, we can make other choices of structure, and the same formula is interpreted differently.

Here are a few examples illustrating the vocabulary associated with formulas, each time giving a word of the language and its interpretation in the structure.

Let us begin with terms: there are only the constant c and the variables $x, y, \dots$. In the structure, we replace c with 0.

Term in $\mathcal{L}$	Interpretation in $\mathcal{S}$
c	0
x	x

Atomic formulas are formed using a predicate. Here, only the predicate P is defined, and it is interpreted as $x < y$ in the structure. We also always have the equality predicate “ $x = y$.”

Atomic formula in $\mathcal{L}$	Interpretation in $\mathcal{S}$
$P(c, x)$	$0 < x$
$P(c, c)$	$0 < 0$
$P(x, y)$	$x < y$
$P(y, x)$	$y < x$
$x = c$	$x = 0$

For formulas, connectives and quantifiers are allowed. A formula “ $\forall x \mathcal{P}$ ” is interpreted in the structure as “ $\forall x \in \mathbb{N} \mathcal{P}$.”

Formula in $\mathcal{L}$	Interpretation in $\mathcal{S}$
$\forall x \ P(c, x)$	$\forall x \in \mathbb{N} \ 0 < x$
$\exists y \ x = y$	$\exists y \in \mathbb{N} \ x = y$
$\forall x \ P(c, x) \vee (x = c)$	$\forall x \in \mathbb{N} \ (0 < x) \vee (x = 0)$
$\forall x \ \forall y \ P(x, y) \rightarrow (\exists z \ P(x, z) \wedge P(z, y))$	$\forall x \in \mathbb{N} \ \forall y \in \mathbb{N} \ (x < y) \rightarrow (\exists z \in \mathbb{N} \ (x < z) \wedge (z < y))$

For the integers, the last formula is interpreted as “between any two distinct integers, one can find an integer strictly between them.” This is clearly false (for example, there is no integer strictly between 7 and 8). It would be true for another structure whose domain is the set of real numbers $\mathbb{R}$ (between two distinct real numbers, there are even infinitely many real numbers).

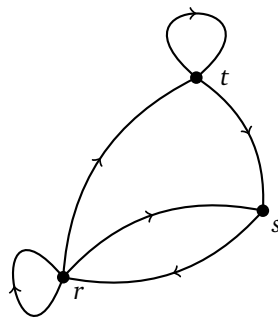
4.2. With graphs

We want to model graphs (such as the one in the figure below). We define a language $\mathcal{L}$ with the binary predicate R (and no constants).

In this language, the only terms are variables, for example x and y . The atomic formulas are $R(x, x)$, $R(x, y)$, $R(y, x)$, $R(y, y)$, as well as $x = x$, $x = y$, $y = x$, $y = y$. Here is an example of a formula:

$$\forall x \ \exists y \ R(x, y)$$

We have our language. How do we model a graph? Here is the directed graph we want to model:



There are three vertices, so we define a structure whose domain is the three-element set $E = \{r, s, t\}$. In the structure, it is the predicate that describes the edges: we interpret the predicate R by saying that $R(x, y)$ is true if there is an edge from x to y (we say that x is related to y). Thus, in our example, we define the predicate in the structure by:

$$R(r, s) = \text{T}, \quad R(s, t) = \text{F}, \quad R(t, s) = \text{T}, \dots$$

The preceding formula in the language $\mathcal{L}$ is interpreted in this structure as:

$$\forall x \in \{r, s, t\} \quad \exists y \in \{r, s, t\} \quad R(x, y)$$

In other words, “for every vertex x , there is a vertex y for which there is an edge from x to y ”; put differently, “at least one edge leaves every vertex,” which is true for the graph under consideration.

Here are some sentences in the language. It is up to you to determine what they mean for graphs and whether they are true or false for the example above.

- $\forall x \forall y \quad R(x, y)$
- $\exists x \exists y \quad R(x, y)$
- $\exists x \forall y \quad R(x, y)$
- $\forall x \forall y \quad R(x, y) \vee R(y, x)$
- $\exists x \exists y \quad R(x, y) \wedge R(y, x)$

4.3. There exists exactly one; there exist two

- “ $\exists x P(x)$ ” requires the existence of at least one x for which $P(x)$ holds. There may be one or more.
- Here is how to express that there is exactly one element for which $P(x)$ holds:

$$(\exists x \quad P(x)) \wedge (\forall x \forall y \quad (P(x) \wedge P(y) \rightarrow x = y))$$

On the one hand, we require an element to exist; on the other, if there is another one, it is the same element. We will not use the notation “ $\exists!$ ”.

- Here is how to express that there are at least two elements:

$$\exists x \exists y \quad (x \neq y) \wedge P(x) \wedge P(y)$$

We write $x \neq y$ for $\neg(x = y)$.

5. Language with functions

5.1. Functions

A **function** f takes one or more objects as input and returns an object as output. Functions must not be confused with predicates: predicates also take objects as input, but they return a T/F value.

Here are some examples of interpretations of functions:

- $f(x) = x^2$, a function of one variable (here, over the real numbers),
- $f(x, y) = x \% y$, the remainder in the Euclidean division of x by y (here, over the integers).

In contrast, $P(x)$: “ $x = 0$ ” and $Q(x, y)$: “ $x < y$ ” are predicates, not functions.

From now on, we allow n -ary functions (or functions of n arguments) to be added as nonlogical symbols of the language. As with predicates, we will often abuse notation and write functions as $f(x)$, $f(x, y)$, $f(x_1, \dots, x_n)$, even though the symbol in the language is just f .

5.2. Formulas

With the introduction of functions, we need to adapt our definition of a formula.

Let $\mathcal{L}$ be a language (with or without functions).

Definition.

The **terms** of a language are:

- the variables and constants,
- and if $t_1, \dots, t_n$ are terms and f is an n -ary function symbol, then $f(t_1, \dots, t_n)$ is also a term.

Note that the definition is recursive.

The definition of an **atomic formula** is unchanged: it is an expression of the form $P(t_1, \dots, t_n)$, where P is an n -ary predicate and $t_1, \dots, t_n$ are terms (according to our new definition).

The definition of a **formula** is unchanged: it is the same construction using atomic formulas, connectives, and quantifiers (except, of course, that terms and atomic formulas may now contain functions).

5.3. Examples

Let $\mathcal{L}$ be the language consisting of:

- constants a, b ,
- a binary predicate P , written (by an abuse of notation) as $x < y$,
- a unary function f and a binary function g .
- Examples of terms:

$$a, \quad y, \quad f(x), \quad g(a, y), \quad g(f(x), b), \quad f(g(y, x)), \quad \dots$$

- Examples of atomic formulas:

$$x < a, \quad f(x) < f(y), \quad g(a, f(b)) < x, \quad \dots$$

- Examples of formulas:

$$\neg(x < y), \quad (f(x) < a) \wedge (f(y) < b), \quad \forall x \, f(x) < b, \quad \exists x \, g(x, y) = a, \\ \forall x \, (f(x) < a) \rightarrow (\exists y \, g(x, y) < b)$$

As before, we anticipate the notion of a structure in order to make these functions concrete. For example, we may work over the set of real numbers $\mathbb{R}$ and consider the following interpretation of the nonlogical symbols:

- $a = 0, b = 1$,
- $P(x, y)$ denotes the usual strict inequality $x < y$,
- $f(x) = x^2$ and $g(x, y) = x + y$.
- Examples of terms in this interpretation:

$$x + 1, \quad y^2, \quad x^2 + y, \quad (x + y)^2, \quad x + (y^2 + 1)^2, \quad \dots$$

- Examples of atomic formulas in this interpretation:

$$x < 0, \quad x + y < x^2, \quad x^2 + y^2 = 1, \quad \dots$$

- Examples of formulas in this interpretation:

$$(x^2 + y^2 < 1) \rightarrow ((x < 1) \wedge (y < 1)), \quad \forall x \, x^2 < x + y, \quad (y < 1) \rightarrow (\exists x \, x^2 = y), \quad \dots$$

Exercises

Language, Formulas, Examples

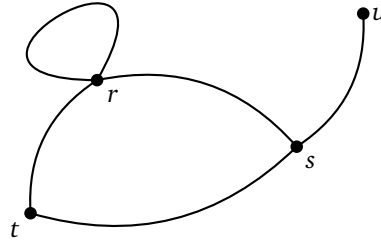
Exercise 1. Consider the language $\mathcal{L}$ consisting of the constants a and b , a unary predicate P , and a binary predicate Q (as well as the variables x, y , the connectives, and the quantifiers). Determine whether the following words are terms, atomic formulas, or formulas.

$$\neg a, \quad xy, \quad (x \wedge a), \quad P(a), \quad \neg Q(x) \\ \forall x Q(x, y), \quad \exists y P(x), \quad \forall a \exists Q(a, b) \\ \forall x \exists y P(x \rightarrow y), \quad \exists x ((\forall y Q(x, y)) \vee (\forall y \neg Q(x, y)))$$

Exercise 2. Consider the language $\mathcal{L}$ equipped with a constant c and a binary predicate $P(x, y)$. Determine whether the following words are terms, atomic formulas, or formulas. If they are, give their interpretation in the structure with domain $\mathbb{N}$, where $c = 0$ and $P(x, y)$ is defined by $x < y$. Then interpret them in the structure with domain $\mathbb{R}$, where $c = 1$ and $P(x, y)$ is defined by $x \leq y$.

$$P, \quad c^2, \quad P(c, c), \quad \neg P(c, 0), \quad P(x, P(y, c)), \quad P(x = c) \rightarrow x = c \\ P(x, c), \quad \exists x, \quad x > c \rightarrow \forall y P(y, c), \quad \forall y (P(x, c) \rightarrow y = x) \\ \exists x \vee y P(x, c) \vee P(y, c), \quad \forall x ((x = c) \leftrightarrow (\exists y P(y, c)))$$

Exercise 3. Consider a language with a binary predicate $R(x, y)$. Consider the structure with domain $E = \{r, s, t, u\}$, and interpret $R(x, y)$ as true if there is an (undirected) edge between the vertices x and y of a graph G (such as the one below).



Translate the following sentences into the language $\mathcal{L}$:

1. There is a vertex connected to itself.
2. There is a vertex connected to every other vertex.
3. Every vertex is connected to at least one other vertex.
4. There is a vertex that is not connected to any other vertex.
5. Not all the vertices are connected to one another.
6. There is exactly one vertex connected to itself.

Exercise 4. Consider a language $\mathcal{L}$ equipped with a predicate $P(x)$ and equality $x = y$. Translate the following sentences into formulas of $\mathcal{L}$:

1. No element satisfies P .
2. There is at most one element that satisfies P .

3. There are exactly two elements that satisfy P .
4. There are at least three elements that satisfy P .
5. All but one element satisfy P .

Language with functions

Exercise 5. Consider a language $\mathcal{L}$ having constants a, b , a predicate $P(x)$, and two functions $f(x)$ and $g(x, y)$. Determine whether the following words are terms, atomic formulas, or formulas of $\mathcal{L}$. If they are, give their interpretation in the structure with domain $\mathbb{R}$, where $a = 0$, $b = 1$, $P(x)$ is defined by $x \geq 0$, $f(x) = \cos(x)$, and $g(x, y) = x \times y$.

$$\begin{aligned}
 &f(x), \quad g(a, b), \quad f(P(a)), \quad \neg P(f(a)), \quad P(g(y, x)), \quad P(g(f(x), f(f(b)))) \\
 &P(x) \wedge P(y) \rightarrow g(x, y), \quad \exists x \, f(x) = b, \quad \forall y \, P(y) \leftrightarrow P(f(y)) \\
 &\forall x \forall b \, P(g(x, b)) \rightarrow (P(x) \wedge P(b)), \quad \exists x \, (P(x) \leftrightarrow (\forall y \, (P(y) \rightarrow P(g(x, y))))))
 \end{aligned}$$