

syn.1 Terms and Formulas

fol:syn:frm:
sec Once a first-order language \mathcal{L} is given, we can define expressions built up from the basic vocabulary of \mathcal{L} . These include in particular *terms* and *formulas*.

fol:syn:frm:
defn:terms **Definition syn.1 (Terms).** The set of *terms* $\text{Trm}(\mathcal{L})$ of \mathcal{L} is defined inductively by:

1. Every **variable** is a term.
2. Every **constant symbol** of \mathcal{L} is a term.
3. If f is an n -place **function symbol** and t_1, \dots, t_n are terms, then $f(t_1, \dots, t_n)$ is a term.
4. Nothing else is a term.

A term containing no **variables** is a *closed term*.

The **constant symbols** appear in our specification of the language and the terms as a separate category of symbols, but they could instead have been included as zero-place **function symbols**. We could then do without the second clause in the definition of terms. We just have to understand $f(t_1, \dots, t_n)$ as just f by itself if $n = 0$. explanation

fol:syn:frm:
defn:formulas **Definition syn.2 (Formula).** The set of *formulas* $\text{Frm}(\mathcal{L})$ of the language \mathcal{L} is defined inductively as follows:

1. \perp is an atomic **formula**.
2. \top is an atomic **formula**.
3. If R is an n -place **predicate symbol** of \mathcal{L} and t_1, \dots, t_n are terms of \mathcal{L} , then $R(t_1, \dots, t_n)$ is an atomic **formula**.
4. If t_1 and t_2 are terms of \mathcal{L} , then $=(t_1, t_2)$ is an atomic **formula**.
5. If φ is a **formula**, then $\neg\varphi$ is **formula**.
6. If φ and ψ are **formulas**, then $(\varphi \wedge \psi)$ is a **formula**.
7. If φ and ψ are **formulas**, then $(\varphi \vee \psi)$ is a **formula**.
8. If φ and ψ are **formulas**, then $(\varphi \rightarrow \psi)$ is a **formula**.
9. If φ and ψ are **formulas**, then $(\varphi \leftrightarrow \psi)$ is a **formula**.
10. If φ is a **formula** and x is a **variable**, then $\forall x \varphi$ is a **formula**.
11. If φ is a **formula** and x is a **variable**, then $\exists x \varphi$ is a **formula**.
12. Nothing else is a **formula**.

explanation

The definitions of the set of terms and that of **formulas** are *inductive definitions*. Essentially, we construct the set of **formulas** in infinitely many stages. In the initial stage, we pronounce all atomic formulas to be formulas; this corresponds to the first few cases of the definition, i.e., the cases for \top , \perp , $R(t_1, \dots, t_n)$ and $=(t_1, t_2)$. “Atomic **formula**” thus means any **formula** of this form.

The other cases of the definition give rules for constructing new **formulas** out of **formulas** already constructed. At the second stage, we can use them to construct **formulas** out of atomic **formulas**. At the third stage, we construct new formulas from the atomic formulas and those obtained in the second stage, and so on. A **formula** is anything that is eventually constructed at such a stage, and nothing else.

By convention, we write $=$ between its arguments and leave out the parentheses: $t_1 = t_2$ is an abbreviation for $=(t_1, t_2)$. Moreover, $\neg=(t_1, t_2)$ is abbreviated as $t_1 \neq t_2$. When writing a formula $(\psi * \chi)$ constructed from ψ , χ using a two-place connective $*$, we will often leave out the outermost pair of parentheses and write simply $\psi * \chi$.

intro

Some logic texts require that the **variable** x must occur in φ in order for $\exists x \varphi$ and $\forall x \varphi$ to count as **formulas**. Nothing bad happens if you don’t require this, and it makes things easier.

If we work in a language for a specific application, we will often write two-place **predicate symbols** and **function symbols** between the respective terms, e.g., $t_1 < t_2$ and $(t_1 + t_2)$ in the language of arithmetic and $t_1 \in t_2$ in the language of set theory. The successor function in the language of arithmetic is even written conventionally *after* its argument: t' . Officially, however, these are just conventional abbreviations for $A_0^2(t_1, t_2)$, $f_0^2(t_1, t_2)$, $A_0^2(t_1, t_2)$ and $f_0^1(t)$, respectively.

Definition syn.3 (Syntactic identity). The symbol \equiv expresses syntactic identity between strings of symbols, i.e., $\varphi \equiv \psi$ iff φ and ψ are strings of symbols of the same length and which contain the same symbol in each place.

The \equiv symbol may be flanked by strings obtained by concatenation, e.g., $\varphi \equiv (\psi \vee \chi)$ means: the string of symbols φ is the same string as the one obtained by concatenating an opening parenthesis, the string ψ , the \vee symbol, the string χ , and a closing parenthesis, in this order. If this is the case, then we know that the first symbol of φ is an opening parenthesis, φ contains ψ as a substring (starting at the second symbol), that substring is followed by \vee , etc.

Photo Credits

Bibliography