

int.1 Formulas

Here is the approach we will use to rigorously specify **sentences** of first-order logic and to deal with the issues arising from the use of **variables**. We first define a *different* set of expressions: **formulas**. Once we've done that, we can consider the role **variables** play in them—and on the basis of some other ideas, namely those of “free” and “bound” **variables**, we can define what a **sentence** is (namely, a **formula** without free **variables**). We do this not just because it makes the definition of “**sentence**” more manageable, but also because it will be crucial to the way we define the semantic notion of satisfaction.

Let's define “**formula**” for a simple first-order language, one containing only a single **predicate symbol** P and a single **constant symbol** a , and only the logical symbols \neg , \wedge , and \exists . Our full definitions will be much more general: we'll allow infinitely many **predicate symbols** and **constant symbols**. In fact, we will also consider **function symbols** which can be combined with **constant symbols** and **variables** to form “terms.” For now, a and the variables will be our only terms. We do need infinitely many **variables**. We'll officially use the symbols v_0, v_1, \dots , as variables.

Definition int.1. The set of **formulas** Frm is defined as follows:

fol:int:fml:
fmls-atom

1. $P(a)$ and $P(v_i)$ are **formulas** ($i \in \mathbb{N}$).
2. If φ is a **formula**, then $\neg\varphi$ is **formula**.
3. If φ and ψ are **formulas**, then $(\varphi \wedge \psi)$ is a **formula**.
4. If φ is a **formula** and x is a **variable**, then $\exists x \varphi$ is a **formula**.

fol:int:fml:
fmls-limit

5. Nothing else is a **formula**.

(1) tell us that $P(a)$ and $P(v_i)$ are **formulas**, for any $i \in \mathbb{N}$. These are the so-called *atomic formulas*. They give us something to start from. The other clauses give us ways of forming new **formulas** from ones we have already formed. So for instance, we get that $\neg P(v_2)$ is a **formula**, since $P(v_2)$ is already a **formula** by (1), and then we get that $\exists v_2 \neg P(v_2)$ is another **formula**, and so on. (5) tells us that *only* strings we can form in this way count as **formulas**. In particular, $\exists v_0 P(a)$ and $\exists v_0 \exists v_0 P(a)$ *do* count as **formulas**, and $(\neg P(a))$ does not.

This way of defining **formulas** is called an *inductive definition*, and it allows us to prove things about **formulas** using a version of proof by induction called *structural induction*. These are discussed in ?? and ??, which you should review before delving into the proofs later on. Basically, the idea is that if you want to give a proof that something is true for all **formulas** you show first that it is true for the atomic **formulas**, and then that *if* it's true for any **formula** φ (and ψ), it's *also* true for $\neg\varphi$, $(\varphi \wedge \psi)$, and $\exists x \varphi$. For instance, this proves that it's true for $\exists v_2 \neg P(v_2)$: from the first part you know that it's true for the atomic **formula** $P(v_2)$. Then you get that it's true for $\neg P(v_2)$ by

the second part, and then again that it's true for $\exists v_2 \neg P(v_2)$ itself. Since all formulas are inductively generated from atomic formulas, this works for any of them.

Photo Credits

Bibliography