Lecture 8

First-order logic

Syntax and semantics

Print version of the lecture in *Logic and Proof*

presented on 15 May 2019

by Dr Christoph Haase

8.1

First-order logic can be understood as an extension of propositional logic. In propositional logic the atomic formulas have no internal structure—they are propositional variables that are either true or false. In first-order logic the atomic formulas are *predicates* that assert a relationship among certain elements. Another significant new concept in first-order logic is *quantification*: the ability to assert that a certain property holds *for all elements* or that it holds *for some element*.

1 Syntax of First-Order Logic

The syntax of first-order logic is defined relative to a signature. A **signature** σ consists of a set of **constant symbols**, a set of **function symbols** and a set of **predicate symbols**. Each function and predicate symbol has an **arity** $k > 0$. We will often refer to predicates as **relations**. Typically we use letters c, d to denote constant symbols, f, g to denote function symbols and P, Q, R to denote predicate symbols. Note that the elements of a signature are *symbols*; only later will will we interpret them as concrete functions or relations. Independent of the signature σ we also have a countably infinite set of *variables* $x_0, x_1, x_2, \ldots$.

Definition 1. Given a signature σ, the set of σ-**terms** is defined by structural induction as follows:

- Each variable is a term.
- Each constant symbol is a term.
- If $t_1, \ldots, t_k$ are terms and f is a k-ary function symbol then $f(t_1, \ldots, t_k)$ is a term.

The set of **formulas** is defined inductively as follows:

- Given terms $t_1, \ldots, t_k$ and a k-ary predicate symbol P then $P(t_1, \ldots, t_k)$ is a formula.
- For each formula F, $\neg F$ is a formula.
- For each pair of formulas F, G, $(F \vee G)$ and $(F \wedge G)$ are both formulas.
- If F is a formula and x is a variable then $\exists x\, F$ and $\forall x\, F$ are both formulas.

Atomic formulas are those constructed according to the first rule above. The atomic formula $P(t_1, \ldots, t_k)$ is read "$t_1, \ldots, t_k$ *are in relation* P". The symbol $\exists$ is called the **existential quantifier**. The formula $\exists x\, H$ is read "*there exists* x, H". The symbol $\forall$ is called the **universal quantifier**. The formula $\forall x\, H$ is read "*for all* x, H". A general first-order formula is built up from atomic formulas using the Boolean connectives and the two quantifiers. If a formula F occurs as part of another formula G then F is called a **subformula** of G. We assume that the quantifiers bind more tightly than any of the Boolean operators, e.g., $\forall x\, F \wedge G$ denotes the formula $(\forall x\, F) \wedge G$.

One important measure of the complexity of a formula F is its **quantifier depth**, which is denoted $\mathrm{qd}(F)$. We define this by induction on F as follows. Atomic formulas have quantifier depth 0; given formulas F and G,

$$\begin{aligned}\mathrm{qd}(\neg F) &:= \mathrm{qd}(F)\\ \mathrm{qd}(F \wedge G) = \mathrm{qd}(F \vee G) &:= \max(\mathrm{qd}(F), \mathrm{qd}(G))\\ \mathrm{qd}(\exists x\, F) = \mathrm{qd}(\forall x\, F) &:= \mathrm{qd}(F) + 1.\end{aligned}$$

Example 2. Consider a signature with a single binary relation symbol R. Since there are no constant symbols or function symbols, the only terms are variables. An example of a formula is

$$\forall x\, \forall y\, \forall z\, (R(x,y) \wedge R(y,z) \rightarrow R(x,z))\,.$$

This formula expresses that R is a transitive relation.

Example 3. Consider a signature with a constant symbol 0, unary function symbol s, and unary predicate symbol E. Terms over this signature include the *ground terms* (i.e., variable-free terms) $0, s(0), s(s(0)), \ldots$ as well as terms that mention variables, such as $s(x)$. An example of a formula is $E(0) \wedge \forall x(E(x) \leftrightarrow \neg E(s(x)))$.

Sometimes we write function symbols and predicate symbols *infix* to improve readability:

Example 4. Consider a signature with a constant symbol 1, binary function symbol $+$, and a binary relation symbol $<$, both written infix. Then $x + 1$ is a term and $\forall x(x < (y + 1))$ is a formula.

In a formula $\exists x\, G$ we say that G is the *scope* of the quantifier $\exists x$. The scope of an occurrence of the universal quantifier is defined similarly. An occurrence of a variable x in a formula F is *bound* if that occurrence is within the scope of either $\exists x$ or $\forall x$. An occurrence that is not bound is said to be *free*. Note that the same variable can occur both bound and free in a given formula, e.g., variable x occurs both bound and free in the formula $P(x) \wedge \exists x\, P(x)$. A formula with no free variables is said to be *closed* or a *sentence*. The formulas in Examples 2 and 3 were closed, whereas the formula in Example 4 has a free variable y.

We will consider one important variant of first-order logic as described above, namely *first-order logic with equality*. This variant admits equality as built-in binary relation symbol. Thus, regardless of the signature, we admit $t_1 = t_2$ as an atomic formula for all terms t_1 and t_2.

2 Semantics of First-Order Logic

The semantics of formulas of first-order logic is given in terms of σ-structures.

Definition 5. Given a signature σ, a σ**-structure** (or **assignment**) $\mathcal{A}$ consists of:

- a non-empty set $U_{\mathcal{A}}$ called the **universe** of the structure;
- for each k-ary predicate symbol P in σ, a k-ary relation

$$P_{\mathcal{A}} \subseteq \underbrace{U_{\mathcal{A}} \times \cdots \times U_{\mathcal{A}}}_{k};$$

- for each k-ary function symbol f in σ, a k-ary function,

$$f_{\mathcal{A}} \colon \underbrace{U_{\mathcal{A}} \times \cdots \times U_{\mathcal{A}}}_{k} \rightarrow U_{\mathcal{A}};$$

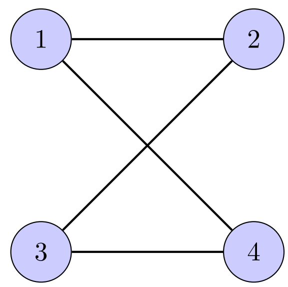

Figure 1: An undirected graph

- for each constant symbol c, an element $c_{\mathcal{A}}$ of $U_{\mathcal{A}}$;
- for each variable x an element $x_{\mathcal{A}}$ of $U_{\mathcal{A}}$.

The above definition treats constant symbols and variables identically. However a key difference is that the interpretation of variables can be overwritten. Given a structure $\mathcal{A}$, variable x, and $a \in U_{\mathcal{A}}$, we define the structure $\mathcal{A}_{[x \mapsto a]}$ to be exactly the same as $\mathcal{A}$ except that $x_{\mathcal{A}_{[x \mapsto a]}} := a$.

Often one specifies a structure as a tuple consisting of a set, some relations, some functions and some constants. For example, $(\mathbb{N}, <, 0)$ denotes the structure with universe $\mathbb{N}$, binary relation $<$ (understood as the usual order on $\mathbb{N}$) and constant 0. Note though that this convention does not specify which values are assigned to variables.

We define the **value** $\mathcal{A}(t)$ of each term t as an element of the universe $U_{\mathcal{A}}$ inductively as follows:

- For a constant symbol c we define $\mathcal{A}(c) := c_{\mathcal{A}}$.
- For a variable x we define $\mathcal{A}(x) := x_{\mathcal{A}}$.
- For a term $f(t_1, \ldots, t_k)$, where f is a k-ary function symbol and $t_1, \ldots, t_k$ are terms, we define $\mathcal{A}(f(t_1, \ldots, t_k)) := f_{\mathcal{A}}(\mathcal{A}(t_1), \ldots, \mathcal{A}(t_k))$.

We define the **satisfaction relation** $\mathcal{A} \models F$ ($\mathcal{A}$ **satisfies** F, or $\mathcal{A}$ **models** F) between a σ-structure $\mathcal{A}$ and σ-formula F by induction over the structure of formulas.

1. $\mathcal{A} \models P(t_1, \ldots, t_k)$ if and only if $(\mathcal{A}(t_1), \ldots, \mathcal{A}(t_k)) \in P_{\mathcal{A}}$.
2. $\mathcal{A} \models (F \wedge G)$ if and only if $\mathcal{A} \models F$ and $\mathcal{A} \models G$.
3. $\mathcal{A} \models (F \vee G)$ if and only if $\mathcal{A} \models F$ or $\mathcal{A} \models G$.
4. $\mathcal{A} \models \neg F$ if and only if $\mathcal{A} \not\models F$.
5. $\mathcal{A} \models \exists x\, F$ if and only if there exists $a \in U_{\mathcal{A}}$ such that $\mathcal{A}_{[x \mapsto a]} \models F$.
6. $\mathcal{A} \models \forall x\, F$ if and only if $\mathcal{A}_{[x \mapsto a]} \models F$ for all $a \in U_{\mathcal{A}}$.

If we are working in first-order logic with equality then we additionally have

7. $\mathcal{A} \models t_1 = t_2$ if and only if $\mathcal{A}(t_1) = \mathcal{A}(t_2)$.

While the function and predicate symbols in a signature can be interpreted as arbitrary functions and predicates in a given structure, the equality symbol is treated as a "built-in", and is always interpreted as equality.

Example 6. An undirected graph can be considered as a σ-structure for the signature σ with one binary relation symbol E, where E is interpreted as the edge relation. For example, the graph shown in Figure 2 can be represented by a structure $\mathcal{A}$ with universe $U_{\mathcal{A}} = \{1, 2, 3, 4\}$ and irreflexive symmetric binary relation

$$E_{\mathcal{A}} = \{(1,2), (2,3), (3,4), (4,1), (2,1), (3,2), (4,3), (1,4)\}.$$

The following sentence asserts that edge relation is irreflexive and symmetric:

$$\forall x\, \neg E(x,x) \wedge \forall x\, \forall y\, (E(x,y) \rightarrow E(y,x))$$

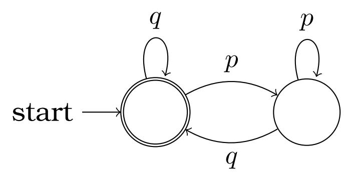

Figure 2: Automaton accepting all strings in which each q is followed by some p.

This sentence is satisfied by the structure in Figure 2.

The following sentence expresses that every pair of nodes are connected by a path of length 3.

$$\forall x\,\forall y\,\exists z_1\,\exists z_2\,(E(x,z_1)\wedge E(z_1,z_2)\wedge E(z_2,y)).$$

This sentence is not satisfied by the structure in Figure 2.

Exercise 7. Let signature σ comprise a single unary relation symbol P and let $\mathcal{A}$ be the assignment with $U_{\mathcal{A}}=\{0,1\}$ and $P_{\mathcal{A}}=\{1\}$. Does $\mathcal{A}$ satisfy the sentence

$$\forall x_1\ldots\forall x_n(P(x_1)\to(P(x_2)\to(P(x_3)\to\ldots\to(P(x_n)\to P(x_1))\ldots)))\,?$$

Have you seen this question before?

Example 8. Consider a signature σ with one binary relation symbol $<$. A *totally ordered set* satisfies the following sentences in first-order logic with equality:

1. Irreflexivity: $\forall x\,\neg(x<x)$.
2. Transitivity: $\forall x\,\forall y\,\forall z((x<y\wedge y<z)\to x<z)$.
3. Trichotomy: $\forall x\,\forall y\,(x<y\vee y<x\vee x=y)$.

The structures $(\mathbb{Z},<_{\mathbb{Z}})$, $(\mathbb{Q},<_{\mathbb{Q}})$ and $(\mathbb{R},<_{\mathbb{R}})$ all satisfy the above sentences.

Example 9. Let $w=w_0w_1\ldots w_{n-1}$ be a finite string over an alphabet $\{p,q\}$. Consider a signature σ with a binary relation symbol $<$ and unary predicate symbols P and Q. We can see w as a σ-structure $\mathcal{A}$ whose universe $U_{\mathcal{A}}$ is the set $\{0,1,\ldots,n-1\}$ of positions in w, $<_{\mathcal{A}}$ is the usual order on $U_{\mathcal{A}}$, $P_{\mathcal{A}}=\{i:w_i=p\}$ is the set of positions in which letter a occurs and likewise $Q_{\mathcal{A}}=\{i:w_i=q\}$ is the set of positions in which letter b occurs.

The sentence $F=\forall x(P(x)\to\exists y\,(x<y\wedge Q(y)))$ is satisfied by a string precisely when every letter p is followed by a letter q. It is easy to see that the set of finite strings that satisfy F is precisely the language of the automaton in Figure 2. In fact for *any* sentence F over this signature, the set of strings satisfying F defines a regular language.

A first-order formula F over signature σ is **satisfiable** if $\mathcal{A}\models F$ for some σ-structure $\mathcal{A}$. If F is not satisfiable it is called **unsatisfiable**. F is called **valid** if $\mathcal{A}\models F$ for every σ-structure $\mathcal{A}$. Given a set of formulas $\mathcal{S}$ we write $\mathcal{S}\models F$ to mean that every σ-structure $\mathcal{A}$ that satisfies $\mathcal{S}$ also satisfies F. The same relations exist among these notions as in propositional logic, e.g., F is unsatisfiable if and only if $\neg F$ is valid.

Exercise 10. Consider a signature σ with constant symbol 0, unary function symbol s, and unary predicate symbol P. Is the σ-formula $P(0)\wedge\forall x(P(x)\to P(s(x)))\wedge\exists x\,\neg P(x)$ satisfiable?

3 Reasoning by Induction on Terms and Formulas

Often proofs about first-order logic involve induction on the structure of terms and formulas. We give the following simple lemma by way of example.

Lemma 11 (Relevance Lemma)**.** *Suppose that $\mathcal{A}$ and $\mathcal{A}'$ are σ-assignments with the same universe and identical interpretations of the predicate, function, and constant symbols in σ. If $\mathcal{A}$ and $\mathcal{A}'$ give the same interpretation to each variable occurring free in some σ-formula F then $\mathcal{A} \models F$ if and only if $\mathcal{A}' \models F$.*

Proof. We first show by induction on terms that if $\mathcal{A}(x) = \mathcal{A}'(x)$ for each variable x occurring in a term t occurring in F then $\mathcal{A}(t) = \mathcal{A}'(t)$.

Base cases: If t is either a constant symbol or a variable then $\mathcal{A}(t) = \mathcal{A}'(t)$ by assumption.

Induction steps: For f a k-ary function symbol we have:

$$\begin{array}{rcll} \mathcal{A}(f(t_1,\ldots,t_k)) & = & f_{\mathcal{A}}(\mathcal{A}(t_1),\ldots,\mathcal{A}(t_k)) & \\ & = & f_{\mathcal{A}'}(\mathcal{A}(t_1),\ldots,\mathcal{A}(t_k)) & \text{(since } f_{\mathcal{A}} = f_{\mathcal{A}'}\text{)} \\ & = & f_{\mathcal{A}'}(\mathcal{A}'(t_1),\ldots,\mathcal{A}'(t_k)) & \text{(induction hypothesis)} \\ & = & \mathcal{A}'(f(t_1,\ldots,t_k))\,. & \end{array}$$

We now show by induction on formulas F that if $\mathcal{A}$ and $\mathcal{A}'$ agree on the free variables in F then $\mathcal{A} \models F$ if and only if $\mathcal{A}' \models F$.

Base cases: Let $F = P(t_1,\ldots,t_k)$ be an atomic formula. By assumption, $\mathcal{A}(x) = \mathcal{A}'(x)$ for any variable x appearing in some term t_i. Then

$$\begin{array}{rll} \mathcal{A} \models P(t_1,\ldots,t_k) & \text{iff} & (\mathcal{A}(t_1),\ldots,\mathcal{A}(t_k)) \in P_{\mathcal{A}} \\ & \text{iff} & (\mathcal{A}(t_1),\ldots,\mathcal{A}(t_k)) \in P_{\mathcal{A}'} \quad \text{(Since } P_{\mathcal{A}} = P_{\mathcal{A}'}\text{)} \\ & \text{iff} & (\mathcal{A}'(t_1),\ldots,\mathcal{A}'(t_k)) \in P_{\mathcal{A}'} \quad \text{(by the above result for terms)} \\ & \text{iff} & \mathcal{A}' \models P(t_1,\ldots,t_k)\,. \end{array}$$

Induction steps: We omit the inductive case for the Boolean connectives. We just give the cases for the universal quantifier (the existential quantifier is similar).

$$\begin{array}{rl} \mathcal{A} \models \forall x\, F & \text{iff } \mathcal{A}_{[x\mapsto a]} \models F \text{ for all } a \in U_{\mathcal{A}} \\ & \text{iff } \mathcal{A}'_{[x\mapsto a]} \models F \text{ for all } a \in U_{\mathcal{A}'} \quad \text{(induction hypothesis)} \\ & \text{iff } \mathcal{A}' \models \forall x\, F \end{array}$$

Notice that a variable occurring free in F is either identical to x or it already occurs free in $\forall x\, F$. Thus $\mathcal{A}_{[x\mapsto a]}$ and $\mathcal{A}'_{[x\mapsto a]}$ agree on the free variables of F and we may indeed apply the induction hypothesis above. $\square$

A special case of the relevance lemma is that if F is a closed formula and $\mathcal{A}$ and $\mathcal{A}'$ are assignments that only differ on the interpretation of variables, then $\mathcal{A} \models F$ if and only if $\mathcal{A}' \models F$. For this reason we sometimes don't bother to specify the interpretation of variables when describing assignments in first-order logic.