# Chapter 7
# First-Order Logic: Formulas, Models, Tableaux

## 7.1 Relations and Predicates

The axioms and theorems of mathematics are defined on sets such as the set of integers $\mathscr{Z}$. We need to be able to write and manipulate logical formulas that contain relations on values from arbitrary sets. *First-order logic* is an extension of propositional logic that includes predicates interpreted as relations on a domain.

Before continuing, you may wish to review Appendix on set theory.

*Example 7.1* $\mathscr{P}(x) \subset \mathscr{N}$ is the unary relation that is the subset of natural numbers that are prime: $\{2, 3, 5, 7, 11, \ldots\}$. ■

*Example 7.2* $\mathscr{S}(x, y) \subset \mathscr{N}^2$ is the binary relation that is the subset of pairs $(x, y)$ of natural numbers such that $y = x^2$: $\{(0, 0), (1, 1), (2, 4), (3, 9), \ldots\}$. ■

It would be more usual in mathematics to define a unary function $f(x) = x^2$ which maps a natural number $x$ into its square. As shown in the example, functions are special cases of relations. For simplicity, we limit ourselves to relations in this chapter and the next; the extension of first-order logic to include functions is introduced in Sect. 9.1.

**Definition 7.3** Let $\mathscr{R}$ be an $n$-ary relation on a domain $D$, that is, $\mathscr{R}$ is a subset of $D^n$. The relation $\mathscr{R}$ can be *represented* by the Boolean-valued function $P_{\mathscr{R}} : D^n \mapsto \{T, F\}$ that maps an $n$-tuple to $T$ if and only if the $n$-tuple is an element of the relation:

$$P_{\mathscr{R}}(d_1, \ldots, d_n) = T \text{ iff } (d_1, \ldots, d_n) \in \mathscr{R},$$
$$P_{\mathscr{R}}(d_1, \ldots, d_n) = F \text{ iff } (d_1, \ldots, d_n) \notin \mathscr{R}.$$

■

*Example 7.4* The set of primes $\mathscr{P}$ is represented by the function $P_{\mathscr{P}}$:

$$P_{\mathscr{P}}(0) = F,\ P_{\mathscr{P}}(1) = F,\ P_{\mathscr{P}}(2) = T,$$
$$P_{\mathscr{P}}(3) = T,\ P_{\mathscr{P}}(4) = F,\ P_{\mathscr{P}}(5) = T,$$
$$P_{\mathscr{P}}(6) = F,\ P_{\mathscr{P}}(7) = T,\ P_{\mathscr{P}}(8) = F,\ \ldots$$

■

M. Ben-Ari, *Mathematical Logic for Computer Science*,
DOI 10.1007/978-1-4471-4129-7_7, © Springer-Verlag London 2012

*Example 7.5* The set of squares $\mathscr{S}$ is represented by the function $P_{\mathscr{S}}$:

$$P_{\mathscr{S}}(0,0) = P_{\mathscr{S}}(1,1) = P_{\mathscr{S}}(2,4) = P_{\mathscr{S}}(3,9) = \cdots = T,$$

$$P_{\mathscr{S}}(0,1) = P_{\mathscr{S}}(1,0) = P_{\mathscr{S}}(0,2) = P_{\mathscr{S}}(2,0) =$$
$$P_{\mathscr{S}}(1,2) = P_{\mathscr{S}}(2,1) = P_{\mathscr{S}}(0,3) = P_{\mathscr{S}}(2,2) = \cdots = F.$$

■

This correspondence provides the link necessary for a logical formalization of mathematics. All the logical machinery—formulas, interpretations, proofs—that we developed for propositional logic can be applied to predicates. The presence of a domain upon which predicates are interpreted considerably complicates the technical details but not the basic concepts.

Here is an overview of our development of first-order logic:

- Syntax (Sect. 7.2): Predicates are used to represent functions from a domain to truth values. Quantifiers allow a purely syntactical expression of the statement that the relation represented by a predicate is true for *some* or *all* elements of the domain.
- Semantics (Sect. 7.3): An interpretation consists of a domain and an assignment of relations to the predicates. The semantics of the Boolean operators remains unchanged, but the evaluation of the truth value of the formula must take the quantifiers into account.
- Semantic tableaux (Sect. 7.5): The construction of a tableau is potentially infinite because a formula can be interpreted in an infinite domain. It follows that the method of semantic tableaux is not decision procedure for satisfiability in first-order logic. However, if the construction of a tableau for a formula $A$ terminates in a closed tableau, then $A$ is unsatisfiable (soundness); conversely, a systematic tableau for an unsatisfiable formula will close (completeness).
- Deduction (Sects. 8.1, 8.2): There are Gentzen and Hilbert deductive systems which are sound and complete. A valid formula is provable and we can construct a proof of the formula using tableaux, but given an *arbitrary* formula we cannot decide if it is valid and hence provable.
- Functions (Sect. 9.1): The syntax of first-order logic can be extended with function symbols that are interpreted as functions on the domain. With functions we can reason about mathematical operations, for example:

$$((x > 0 \wedge y > 0) \vee (x < 0 \wedge y < 0)) \rightarrow (x \cdot y > 0).$$

- Herbrand interpretations (Sect. 9.3): There are canonical interpretations called Herbrand interpretations. If a formula in *clausal form* has a model, it has a model which is an Herbrand interpretation, so to check satisfiability, it is sufficient to check if there is an Herbrand model for a formula.
- Resolution (Chap. 10): Resolution can be generalized to first-order logic with functions.

## 7.2 Formulas in First-Order Logic

### 7.2.1 Syntax

**Definition 7.6** Let $\mathscr{P}$, $\mathscr{A}$ and $\mathscr{V}$ be countable sets of *predicate symbols*, *constant symbols* and *variables*. Each predicate symbol $p^n \in \mathscr{P}$ is associated with an *arity*, the number $n \geq 1$ of *arguments* that it takes. $p^n$ is called an $n$-ary predicate. For $n = 1, 2$, the terms *unary* and *binary*, respectively, are also used. ■

**Notation**

- We will drop the word 'symbol' and use the words 'predicate' and 'constant' by themselves for the syntactical symbols.
- By convention, the following lower-case letters, possibly with subscripts, will denote these sets: $\mathscr{P} = \{p, q, r\}$, $\mathscr{A} = \{a, b, c\}$, $\mathscr{V} = \{x, y, z\}$.
- The superscript denoting the arity of the predicate will not be written since the arity can be inferred from the number of arguments.

**Definition 7.7**
$\forall$ is the *universal quantifier* and is read *for all*.
$\exists$ is the *existential quantifier* and is read *there exists*. ■

**Definition 7.8** An *atomic formula* is an $n$-ary predicate followed by a list of $n$ arguments in parentheses: $p(t_1, t_2, \ldots, t_n)$, where each argument $t_i$ is either a variable or a constant. A *formula* in first-order logic is a tree defined recursively as follows:

- A formula is a leaf labeled by an atomic formula.
- A formula is a node labeled by $\neg$ with a single child that is a formula.
- A formula is a node labeled by $\forall x$ or $\exists x$ (for some variable $x$) with a single child that is a formula.
- A formula is a node labeled by a binary Boolean operator with two children both of which are formulas.

A formula of the form $\forall x\, A$ is a *universally quantified formula* or, simply, a *universal formula*. Similarly, a formula of the form $\exists x\, A$ is an *existentially quantified formula* or an *existential formula*. ■

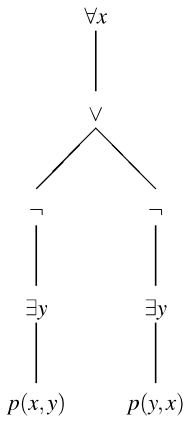
The definition of derivation and formation trees, and the concept of induction on the structure of a formula are taken over unchanged from propositional logic. When writing a formula as a string, the quantifiers are considered to have the same precedence as negation and a higher precedence than the binary operators.

*Example 7.9* Figure 7.1 shows the tree representation of the formula:

$$\forall x(\neg \exists y p(x, y) \vee \neg \exists y p(y, x)).$$

The parentheses in $p(x, y)$ are part of the syntax of the atomic formula. ■

**Fig. 7.1** Tree for $\forall x(\neg \exists y p(x, y) \vee \neg \exists y p(y, x))$




*Example 7.10* Here are some examples of formulas in first-order logic:

$$\forall x \forall y(p(x, y) \rightarrow p(y, x)),$$
$$\forall x \exists y p(x, y),$$
$$\exists x \exists y(p(x) \wedge \neg p(y)),$$
$$\forall x p(a, x),$$
$$\forall x(p(x) \wedge q(x)) \leftrightarrow (\forall x p(x) \wedge \forall x q(x)),$$
$$\exists x(p(x) \vee q(x)) \leftrightarrow (\exists x p(x) \vee \exists x q(x)),$$
$$\forall x(p(x) \rightarrow q(x)) \rightarrow (\forall x p(x) \rightarrow \forall x q(x)),$$
$$(\forall x p(x) \rightarrow \forall x q(x)) \rightarrow \forall x(p(x) \rightarrow q(x)).$$

For now, they are just given as examples of the syntax of formulas in first-order logic; their meaning will be discussed in Sect. 7.3.2. ■

### 7.2.2 The Scope of Variables

**Definition 7.11** A universal or existential formula $\forall x A$ or $\exists x A$ is a *quantified formula*. $x$ is the *quantified variable* and its *scope* is the formula $A$. It is not required that $x$ actually appear in the scope of its quantification. ■

The concept of the scope of variables in formulas of first-order logic is similar to the concept of the scope of variables in block-structured programming languages. Consider the program in Fig. 7.2. The variable `x` is declared twice, once globally and once locally in method `p`. The scope of the global declaration includes `p`, but the local declaration *hides* the global one. Within `p`, the value printed will be `1`, the

**Fig. 7.2** Global and local variables

```
class MyClass {
  int x;

  void p() {
    int x;
    x = 1;
    // Print the value of x
  }

  void q() {
    // Print the value of x
  }

  ... void main(...) {
  x = 5;
  p;
  q;
}
```

value of the local variable. Within the method `q`, the global variable `x` is in scope but not hidden and the value `5` will be printed. As in programming, hiding a quantified variable within its scope is confusing and should be avoided by giving different names to each quantified variable.

**Definition 7.12** Let $A$ be a formula. An occurrence of a variable $x$ in $A$ is a *free variable* of $A$ iff $x$ is not within the scope of a quantified variable $x$. A variable which is not free is *bound*.

If a formula has no free variables, it is *closed*. If $\{x_1, \ldots, x_n\}$ are all the free variables of $A$, the *universal closure* of $A$ is $\forall x_1 \cdots \forall x_n A$ and the *existential closure* is $\exists x_1 \cdots \exists x_n A$.

$A(x_1, \ldots, x_n)$ indicates that the set of free variables of the formula $A$ is a subset of $\{x_1, \ldots, x_n\}$. ■

*Example 7.13* $p(x, y)$ has two free variables $x$ and $y$, $\exists y p(x, y)$ has one free variable $x$ and $\forall x \exists y p(x, y)$ is closed. The universal closure of $p(x, y)$ is $\forall x \forall y p(x, y)$ and its existential closure is $\exists x \exists y p(x, y)$. ■

*Example 7.14* In $\forall x p(x) \wedge q(x)$, the occurrence of $x$ in $p(x)$ is bound and the occurrence in $q(x)$ is free. The universal closure is $\forall x(\forall x p(x) \wedge q(x))$. Obviously, it would have been better to write the formula as $\forall x p(x) \wedge q(y)$ with $y$ as the free variable; its universal closure is $\forall y(\forall x p(x) \wedge q(y))$. ■