

# Logic for Computer Science - Lecture Notes

Alexandru Ioan Cuza University, Iași  
Faculty of Computer Science  
University Year 2021-2022

Ștefan Ciobâcă  
Andrei Arusoaie  
Rodica Condurache  
Cristian Masalagiu



# Contents

<b>1</b>	<b>Motivation and introduction</b>	<b>5</b>
<b>2</b>	<b>Structures and signatures</b>	<b>7</b>
2.1	Exercise sheet . . . . .	10
<b>3</b>	<b>The Syntax of First-Order Logic</b>	<b>13</b>
3.1	The Alphabet of First-Order Logic . . . . .	13
3.2	Terms . . . . .	14
3.3	Atomic formulae . . . . .	15
3.4	First-Order Formulae . . . . .	15
3.5	Model English sentences as formulae in <b>FOL</b> . . . . .	18
3.6	Model arithmetic sentences as formulae in <b>FOL</b> . . . . .	19
3.7	Exercises . . . . .	20



# Chapter 1

## Motivation and introduction

First-order logic, what we will be studying next, is an extension of propositional logic, extension that brings more expressivity. The additional expressivity is necessary in order to model certain statements that cannot be expressed in propositional logic.

In propositional logic, we cannot express naturally the following statement: *All men are mortal*.

To model a statement in propositional logic, we identify the atomic propositions. Then we associate to each atomic proposition a propositional variable. The atomic propositions are the propositions that cannot be split into one or more smaller propositions, linked among them by the logical connectives of propositional logic:  $\neg$ ,  $\wedge$ ,  $\vee$ ,  $\rightarrow$  and  $\leftrightarrow$ .

We notice that the statement *All men are mortal* cannot be decomposed into smaller statements linked among them by the logical connectives of propositional logic, as is described above. Therefore, in propositional logic, the statement is atomic. So we associate to the entire statement a propositional variable  $\mathbf{p} \in A$ .

Let us now model the statement *Socrates is a man*. Obviously, to this second statement we must associate another propositional variable  $\mathbf{q} \in A$ . Let us assume that  $\mathbf{p}$  and  $\mathbf{q}$  are true. Formally, we work in a truth assignment  $\tau : A \rightarrow B$  where  $\tau(\mathbf{p}) = 1$  and  $\tau(\mathbf{q}) = 1$ . Can we draw the conclusion that *Socrates is mortal* in the truth assignment  $\tau$ ?

No, because to the statement *Socrates is mortal* we should associate a third propositional variable  $\mathbf{r} \in A$ . We cannot draw any conclusion on  $\tau(\mathbf{r})$  from  $\tau(\mathbf{p}) = 1$  and  $\tau(\mathbf{q}) = 1$ . So, from the semantics of propositional logic, we cannot draw the conclusion that  $\mathbf{r}$  is true in any truth assignment that makes

both **p** and **q** true. This is despite the fact that, in any world where *All men are mortal* and *Socrates is a man*, we can draw the conclusion that *Socrates is mortal* without failure. This difference between reality and our modelling indicates that our modelling is not sufficient for our purposes.

First-order logic includes, in addition to propositional logic, the notion of *quantifier* and the notion of *predicate*. The universal quantifier is denoted by  $\forall$  and the existential quantifier is denoted by  $\exists$ .

A predicate is a statement whose truth value depends on zero or more parameters. For example, for the statements above, we will be using two predicates: **Man** and **Mortal**. The predicate **Man** is the predicate that denotes the quality of being a man: **Man**(**x**) is true iff **x** is a man. The predicate **Mortal** is true when its argument is mortal. As the predicates above have only one argument/parameter, they are called *unary* predicates. Predicates generalize propositional variables by the fact that they can take arguments. Actually, propositional variable can be regarded as predicates with no arguments.

In this way, the statement *All men are mortal* will be modelled by the formula

$$(\forall x.(\text{Man}(x) \rightarrow \text{Mortal}(x))),$$

which is read as follows: *for any x, if Man of x, then Mortal of x*. The statement *Socrates is a man* shall be modelled by the formula **Man**(**s**), where **s** is a *constant* that denotes Socrates, just like 0 denotes the natural number zero. For example, **Man**(**s**) is true (as **s** stands for a particular man – Socrates), but **Man**(**l**) is false if **l** is a constant standing for the dog *Lassie*.

The statement *Socrates is mortal* shall be represented by **Mortal**(**s**) (recall that the constant **s** stands for Socrates). The statement **Mortal**(**s**) is true, as Socrates is mortal; likewise, the statement **Mortal**(**l**) is also true.

We shall see that in first-order logic, the formula **Mortal**(**s**) is a logical consequence of the formulae  $(\forall x.(\text{Man}(x) \rightarrow \text{Mortal}(x)))$  and respectively **Man**(**s**). Therefore, first-order logic is sufficiently expressive to explain theoretically the argument by which we deduce that *Socrates is mortal* from the facts that *All men are mortal* and *Socrates is a man*.

## Chapter 2

# Structures and signatures

You have certainly met already several first-order logic formulae, without necessarily knowing that you are dealing with first-order logic. Consider the following formula:

$$\varphi = (\forall x. (\forall y. (x < y \rightarrow \exists z. (x < z \wedge z < y))))).$$

The formula makes use of a binary predicate,  $<$ , that is defined as follows:  $<(x, y)$  is true if  $x$  is strictly smaller than  $y$ . In order to simplify our writing, we use the infix notation  $(x < y)$  instead of the prefixed notation  $(<(x, y))$  for many binary predicates (including for  $<$ ).

Is the formula  $\varphi$  above true? The formula states that between any two values of the variables  $x, y$  there is a third value, of the variable  $z$ . The formula is true if the domain of the variables  $x, y, z$  is  $\mathbb{R}$ , but it is false if the domain is  $\mathbb{N}$  (between any two real numbers there exists a third, but between two consecutive naturals there is no other natural number).

Generally, first-order formulas refer to a particular *mathematical structure*.

**Definition 1** (Mathematical structure). A mathematical structure is a tuple  $S = (D, Pred, Fun)$  where:

- $D$  is a non-empty set called the domain of the structure;
- each  $P \in Pred$  is a predicate (of a certain arity) over the set  $D$ ;
- each  $f \in Fun$  is a function (of a certain arity) over the set  $D$ .

Here are a few examples of mathematical structures:

1.  $(\mathbb{N}, \{<, =\}, \{+, 0, 1\})$ ;

The domain of the structure is the set of naturals. The structure contains two predicates:  $<$  and  $=$ , both of arity 2. The predicate  $<$  is the *smaller than* predicate on naturals, and the predicate  $=$  is the *equality* predicate over natural numbers.

The structure also contains three functions. The binary function  $+$  :  $\mathbb{N}^2 \rightarrow \mathbb{N}$  is the addition function for naturals, and the functions  $0$  :  $\mathbb{N}^0 \rightarrow \mathbb{N}$  and respectively  $1$  :  $\mathbb{N}^0 \rightarrow \mathbb{N}$  are the arity 0 functions (also called constant functions or simply constants) 0 and respectively 1.

2.  $(\mathbb{R}, \{<, =\}, \{+, -, 0, 1\})$ ;

This structure contains two binary predicates,  $<$  and  $=$ , as well as four functions over  $\mathbb{R}$ : the binary function  $+$ , the unary function  $-$  (unary minus) and the constants  $0, 1 \in \mathbb{R}$ .

3.  $(\mathbb{Z}, \{<, =\}, \{+, -, 0, 1\})$ ;

This structure is similar to that above, but the domain is the set of integers.

4.  $(B, \emptyset, \{\cdot, +, -\})$ ;

This structure is a boolean algebra, where the domain is the set truth values and the functions are those that we studied in the first half of the semester. Such structures, without any predicates, are called *algebraic structures*.

5.  $(\mathbb{R}, \{<\}, \emptyset)$ .

This structure contains only a predicate of arity 2 (the *less than* relation over  $\mathbb{R}$ ) and no function. Structures without functions are called relational structures. Relational structures with a finite domain are called relational data bases and you will study them in your second year.

Whenever we want to evaluate the truth value of a first-order formula we need a mathematical structure. Recall our previous formula:

$$\varphi = \left( \forall x. (\forall y. (x < y \rightarrow \exists z. (x < z \wedge z < y))) \right).$$

This formula is true in the structure  $(\mathbb{R}, \{<, =\}, \{+, -, 0, 1\})$  (between any two distinct real numbers there is another real number), but it is false in the structure  $(\mathbb{Z}, \{<, =\}, \{+, -, 0, 1\})$  (because it is not true that between any two distinct integers there is a third integer – for example there is no such integer between two consecutive integers).

It is possible for two different structure to have a set of predicates and a set of functions with the same names. For example, the structures above,  $(\mathbb{R}, \{<, =\}, \{+, -, 0, 1\})$  and respectively  $(\mathbb{Z}, \{<, =\}, \{+, -, 0, 1\})$ . Even if the



predicate  $<\in \mathbb{R}^2$  is different from the predicate  $<\in \mathbb{Z}^2$ , they both have the same name:  $<$ .

Generally, in Mathematics and in Computer Science, we do not make any difference between a predicate and its name or between a function and its name. However, in Logic, the difference is extremely important. In particular, if we refer to the name of a function, we shall use the phrase “functional symbol” (i.e., symbol standing for a function). When we refer to the name of a predicate, we shall use the phrase “predicate symbol” (or “relational symbol”). Why is the difference between a predicate and a predicate symbol important? Because we shall need to associate to the same predicate symbol several predicates, similarly to how we can associate several values to a program variable in an imperative language.

When we are interested only in the function and predicate names (not the function or predicates themselves), we work with signatures:

**Definition 2** (Signature). A signature  $\Sigma$  is a tuple  $\Sigma = (\mathcal{P}, \mathcal{F})$ , where  $\mathcal{P}$  is a set of predicate symbols and  $\mathcal{F}$  is a set of functional symbols. Each predicate or functional symbol  $s$  has an associate natural number called its arity denoted by  $ar(s)$ .

To a signature we can associate many structures:

**Definition 3** ( $\Sigma$ -structure). If  $\Sigma = (\mathcal{P}, \mathcal{F})$  is a signature, a  $\Sigma$ -structure is any structure  $S = (D, Pred, Fun)$  so that for each predicate symbol  $P \in \mathcal{P}$ , exists a predicate  $P^S \in Pred$  of corresponding arity, and for every functional symbol  $f \in \mathcal{F}$ , there is a function  $f^S \in Fun$  of corresponding arity.

**Example 4.** Let  $\Sigma = (\{\mathbf{P}, \mathbf{Q}\}, \{\mathbf{f}, \mathbf{i}, \mathbf{a}, \mathbf{b}\})$ , where  $\mathbf{P}$  and  $\mathbf{Q}$  are predicate symbols of arity  $ar(\mathbf{P}) = ar(\mathbf{Q}) = 2$  and  $\mathbf{f}, \mathbf{i}, \mathbf{a}, \mathbf{b}$  are function symbols having the following arities:  $ar(\mathbf{f}) = 2$ ,  $ar(\mathbf{i}) = 1$  and  $ar(\mathbf{a}) = ar(\mathbf{b}) = 0$ .

We have that  $(\mathbb{R}, \{<, =\}, \{+, -, 0, 1\})$  and  $(\mathbb{Z}, \{<, =\}, \{+, -, 0, 1\})$  are  $\Sigma$ -structures.

**Remark.** As you can observe in Example 4, for predicate symbols (e.g.,  $\mathbf{P}, \mathbf{Q}$ ) we use a different color than the color used for functional symbols (e.g.,  $\mathbf{f}, \mathbf{i}, \mathbf{a}, \mathbf{b}$ ). For predicates and functions we use the normal font for mathematical formulas.

To remember!

Structure = domain + predicates + functions

Signature = predicate symbols + functional symbols

To a signature  $\Sigma$  we can associate many structures, which are called  $\Sigma$ -structures.

**Notation.** The set of predicate symbols of arity  $n$  is denoted by  $\mathcal{P}_n = \{P \mid \text{ar}(P) = n\}$ , and the set of functional symbols of arity  $n$  is  $\mathcal{F}_n = \{f \mid \text{ar}(f) = n\}$ . For the particular case when  $n = 0$ ,  $\mathcal{F}_0$  is the set of constant symbols (that is, functional symbols with arity 0).

## 2.1 Exercise sheet

**Exercise 5.** Identify the predicates and the functions in the text below. What is their domain?

John is a student. Any student learns Logic. Anyone learning Logic passes the exam. Any student is a person. There is a person who did not pass the exam. Therefore: not all persons are students.

**Exercise 6.** Identify the predicates and the functions in the text below. What is their domain?

The sum of two even numbers is even.

**Exercise 7.** Identify the predicates and the functions in the text below. What is their domain?

In chess, the queen can make a move from one square to another iff the bishop or the rook can make the same move.

**Exercise 8.** Identify the predicates and the functions in the text below. What is their domain?

The sum of two numbers greater than zero is greater than zero.

**Exercise 9.** Identify the predicates and the functions in the text below. What is their domain?

The number 7 is prime.

**Exercise 10.** Identify the predicates and the functions in the text below. What is their domain?

Any even number greater than 2 is the sum of two primes.

**Exercise 11.** *Identify the predicates and the functions in the text below. What is their domain?*

If the Earth is flat, then  $2 + 2 = 5$ .

**Exercise 12.** *Identify the predicates and the functions in the text below. What is their domain?*

For any  $\epsilon \in (0, \infty)$ , there exists  $\delta_\epsilon \in (0, \infty)$  such that for any  $x \in \mathbb{R}$  with  $d(x_0, x) < \delta_\epsilon$ , we have  $d(f(x_0), f(x)) < \epsilon$ .



## Chapter 3

# The Syntax of First-Order Logic

In this chapter we present the syntax of first-order logic formula. The language of first-order logic is parameterised by a signature. A difference to propositional logic is that there are several first-order logic languages, one first-order language for each signature  $\Sigma$ . In propositional logic, there was just one language,  $\mathbb{PL}$ .

Next, we shall fix a signature  $\Sigma$  that contains the predicate symbols in  $\mathcal{P}$  and the functional symbols in  $\mathcal{F}$ .

### 3.1 The Alphabet of First-Order Logic

Just as propositional logic formulae, the formulae in first-order logic are strings of characters over a certain alphabet. Unlike propositional logic, the alphabet is now richer. The alphabet of first-order logic consists of the follows “characters”:

1. the logical connectives already known:  $\neg, \wedge, \vee, \rightarrow, \leftrightarrow, \perp$ , as well as two new *quantifiers*:  $\forall, \exists$ ;
2. variables: we will assume that a countably infinite set of variables  $\mathcal{X} = \{x, y, z, x', y', x_1, z'', \dots\}$  is also part of the alphabet (not to be confused with propositional variables in propositional logic – they are two fundamentally different notions);
3. auxilliary symbols:  $(, ), \cdot, \circ, (, ),$  and  $;$ ;
4. non-logical symbols, that are specific to each signature  $\Sigma = (\mathcal{P}, \mathcal{F})$ : the functional symbols in  $\mathcal{F}$  and the predicate symbols in  $\mathcal{P}$ .

## 3.2 Terms

**Definition 13.** *The set of terms,  $\mathcal{T}$ , is the smallest set having the following properties:*

1.  $\mathcal{F}_0 \subseteq \mathcal{T}$  (any constant symbol is a term);
2.  $X \subseteq \mathcal{T}$  (any variable is a term);
3. if  $f \in \mathcal{F}_n$  (with  $n > 0$ ) and  $t_1, \dots, t_n \in \mathcal{T}$ , then  $f(t_1, \dots, t_n) \in \mathcal{T}$  (a functional symbol of arity  $n$  applied to  $n$  terms is a term).

**Remark.** *The elements of the set  $\mathcal{T}$  are often called  $\Sigma$ -terms, because the definition of  $\mathcal{T}$  depends on  $\Sigma$ .*

Terms are essentially built by applying functional symbols to variables and constant symbols.

**Example 14.** Recall  $\Sigma = (\{\mathbf{P}, \mathbf{Q}\}, \{\mathbf{f}, \mathbf{i}, \mathbf{a}, \mathbf{b}\})$  from Example 4, where  $ar(\mathbf{P}) = ar(\mathbf{Q}) = 2$ ,  $ar(\mathbf{f}) = 2$ ,  $ar(\mathbf{i}) = 1$ ,  $ar(\mathbf{a}) = ar(\mathbf{b}) = 0$ . Here are several example of terms:  $\mathbf{a}$ ,  $\mathbf{b}$ ,  $\mathbf{x}$ ,  $\mathbf{y}$ ,  $\mathbf{x}_1$ ,  $\mathbf{y}'$ ,  $\mathbf{i}(\mathbf{a})$ ,  $\mathbf{i}(\mathbf{x})$ ,  $\mathbf{i}(\mathbf{i}(\mathbf{a}))$ ,  $\mathbf{i}(\mathbf{i}(\mathbf{x}))$ ,  $\mathbf{f}(\mathbf{a}, \mathbf{b})$ ,  $\mathbf{i}(\mathbf{f}(\mathbf{a}, \mathbf{b}))$ ,  $\mathbf{f}(\mathbf{f}(\mathbf{x}, \mathbf{a}), \mathbf{f}(\mathbf{y}, \mathbf{y}))$ .

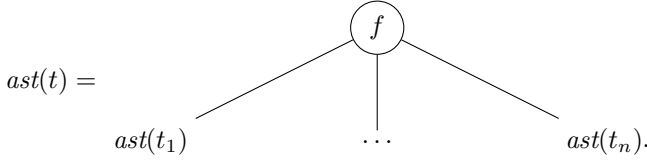
**Exercise 15.** *Identify the  $\Sigma$ -terms in the following list :*

1.  $\mathbf{i}(\mathbf{i}(\mathbf{x}))$ ;
2.  $\mathbf{i}$ ;
3.  $\mathbf{f}(\mathbf{x}, \mathbf{x})$ ;
4.  $\mathbf{P}(\mathbf{a}, \mathbf{b})$ ;
5.  $\mathbf{i}(\mathbf{a}, \mathbf{a})$ ;
6.  $\mathbf{f}(\mathbf{i}(\mathbf{x}), \mathbf{i}(\mathbf{x}))$ ;
7.  $\mathbf{f}(\mathbf{i}(\mathbf{x}, \mathbf{x}))$ ;
8.  $\mathbf{a}(\mathbf{i}(\mathbf{x}))$ .

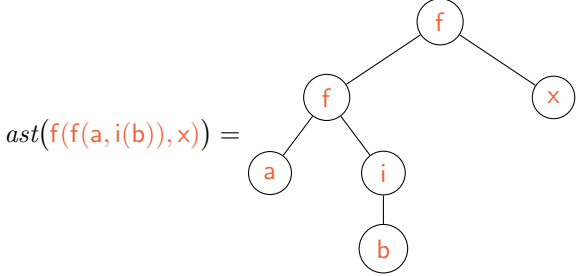
We denote terms by the letters  $t, s, t_1, t_2, s_1, t'$ , etc. Even if terms are usually written as a string of characters, they have an associated abstract syntax tree defined as follows:

1. if  $t = c$  and  $c \in \mathcal{F}_0$ , then  $ast(t) = \bigcirc c$ ;
2. if  $t = x$  and  $x \in \mathcal{X}$ , then  $ast(t) = \bigcirc x$ ;

3. if  $t = f(t_1, \dots, t_n)$  and  $f \in \mathcal{F}_n$  ( $n > 0$ ),  $t_1, \dots, t_n \in \mathcal{T}$ , then



**Remark.** Even if formally terms are defined as strings of characters over the alphabet described above, these must be understood as being trees. In any software program that handles terms, these are stored as a rooted tree. Here is the tree associated to the term  $f(f(a, i(b)), x)$ :



**Exercise 16.** Compute the abstract syntax trees for all terms in Example 14.

### 3.3 Atomic formulae

**Definition 17** (Atomic formula). An atomic formula is any string of characters of the form  $P(t_1, \dots, t_n)$ , unde  $P \in \mathcal{P}_n$ , where  $P \in \mathcal{P}_n$  is a predicate symbol of arity  $n$ , and  $t_1, \dots, t_n \in \mathcal{T}$  are terms.

**Example 18.** Continuing the previous example, we work over the signature

$$\Sigma = (\{P, Q\}, \{f, i, a, b\}),$$

where  $ar(P) = ar(Q) = 2$ ,  $ar(f) = 2$ ,  $ar(i) = 1$ ,  $ar(a) = ar(b) = 0$ .

Here are a few examples of atomic formulae:  $P(a, b)$ ,  $P(x, y)$ ,  $Q(i(i(x)), f(x, x))$ ,  $Q(a, b)$ ,  $P(f(f(a, i(x)), b), i(x))$ .

**Exercise 19.** Explain why  $P(a)$ ,  $P, i(i(x))$  are not atomic formulae over the signature in Example 18.

### 3.4 First-Order Formulae

**Definition 20** (First-Order Formula). The set of first-order formulae, written  $\text{FOL}$ , is the smallest set with the following properties:

1. (base case) any atomic formula is a formula (that is  $P(t_1, \dots, t_n) \in \text{FOL}$  for any predicate symbol  $P \in \mathcal{P}_n$  and any terms  $t_1, \dots, t_n$ ; if  $n = 0$ , we write  $P$  instead of  $P()$ );
2. (inductive cases) for any formulae  $\varphi, \varphi_1, \varphi_2 \in \text{FOL}$ , for any variable  $x \in \mathcal{X}$ , we have:
  - (a)  $\neg \varphi_1 \in \text{FOL}$ ;
  - (b)  $(\varphi_1 \wedge \varphi_2) \in \text{FOL}$ ;
  - (c)  $(\varphi_1 \vee \varphi_2) \in \text{FOL}$ ;
  - (d)  $(\varphi_1 \rightarrow \varphi_2) \in \text{FOL}$ ;
  - (e)  $(\varphi_1 \leftrightarrow \varphi_2) \in \text{FOL}$ ;
  - (f)  $(\forall x. \varphi) \in \text{FOL}$ ;
  - (g)  $(\exists x. \varphi) \in \text{FOL}$ .

**Remark.** In Definition 20, we find the logical connectives  $\neg, \wedge, \vee, \rightarrow$  and  $\leftrightarrow$  from propositional logic. The predicate symbols of arity 0 play the role of propositional variables (for now, at the syntactic level). The constructions  $(\forall x. \varphi)$  and  $(\exists x. \varphi)$  are new.

**Example 21.** Recall  $\Sigma = (\{\mathbf{P}, \mathbf{Q}\}, \{\mathbf{f}, \mathbf{i}, \mathbf{a}, \mathbf{b}\})$  from Example 14, where  $\text{ar}(\mathbf{P}) = \text{ar}(\mathbf{Q}) = 2$ ,  $\text{ar}(\mathbf{f}) = 2$ ,  $\text{ar}(\mathbf{i}) = 1$  și  $\text{ar}(\mathbf{a}) = \text{ar}(\mathbf{b}) = 0$ .

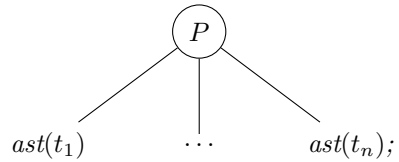
Here are several example of first-order formulae:

1.  $\mathbf{P}(\mathbf{a}, \mathbf{b})$ ;
2.  $\mathbf{Q}(\mathbf{a}, \mathbf{b})$ ;
3.  $\mathbf{P}(\mathbf{a}, \mathbf{x})$ ;
4.  $\neg \mathbf{P}(\mathbf{a}, \mathbf{b})$ ;
5.  $(\mathbf{P}(\mathbf{a}, \mathbf{b}) \wedge \neg \mathbf{Q}(\mathbf{a}, \mathbf{b}))$ ;
6.  $(\mathbf{P}(\mathbf{a}, \mathbf{b}) \vee \neg \mathbf{Q}(\mathbf{x}, \mathbf{y}))$ ;
7.  $(\mathbf{P}(\mathbf{a}, \mathbf{b}) \rightarrow \mathbf{P}(\mathbf{a}, \mathbf{b}))$ ;
8.  $((\mathbf{P}(\mathbf{a}, \mathbf{b}) \rightarrow \mathbf{P}(\mathbf{a}, \mathbf{b})) \leftrightarrow (\mathbf{P}(\mathbf{a}, \mathbf{b}) \rightarrow \mathbf{P}(\mathbf{a}, \mathbf{b})))$ ;
9.  $(\forall \mathbf{x}. \mathbf{P}(\mathbf{a}, \mathbf{x}))$ ;
10.  $(\exists \mathbf{x}. \neg \mathbf{Q}(\mathbf{x}, \mathbf{y}))$ .

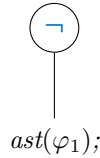
**Definition 22** (The Abstract Syntax Tree of formulae in FOL). *Formulae have an associated abstract syntax tree defined as follows:*



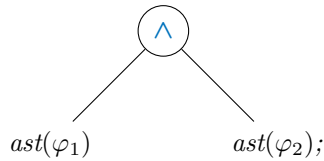
1. if  $\varphi = P(t_1, \dots, t_n)$ , then  $ast(\varphi) =$



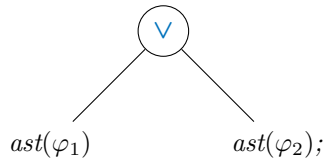
2. if  $\varphi = \neg \varphi_1$ , then  $ast(\varphi) =$



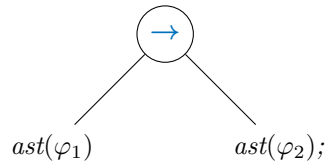
3. if  $\varphi = (\varphi_1 \wedge \varphi_2)$ , then  $ast(\varphi) =$



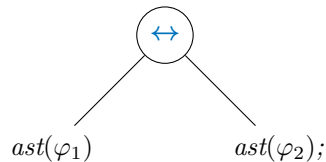
4. if  $\varphi = (\varphi_1 \vee \varphi_2)$ , then  $ast(\varphi) =$



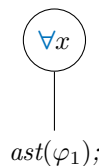
5. if  $\varphi = (\varphi_1 \rightarrow \varphi_2)$ , then  $ast(\varphi) =$



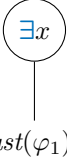
6. if  $\varphi = (\varphi_1 \leftrightarrow \varphi_2)$ , then  $ast(\varphi) =$



7. if  $\varphi = (\forall x. \varphi_1)$ , then  $ast(\varphi) =$



8. if  $\varphi = (\exists x.\varphi_1)$ , then  $ast(\varphi) =$



**Exercise 23.** Compute the abstract syntax tree of the formulae shown in Example 21.

### 3.5 Model English sentences as formulae in FOL

Next, we will explain the signature used to model in first-order logic the statements *All men are mortal*, *Socrates is a man* and respectively *Socrates is mortal*.

First, we identify the predicates in the text. We have two unary predicates *is a man* and respectively *is mortal*. We choose the predicate symbol **H** for the first predicate and the predicate symbol **M** for the second predicate. We also have one constant in the text: Socrates. We choose the functional symbol **s** (of arity 0) for this constant. Therefore, to model the statements above, we shall work in the signature

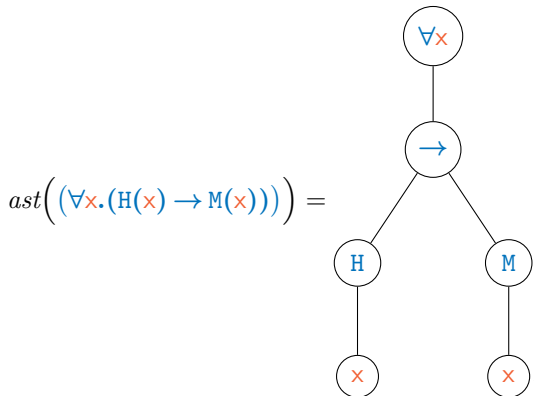
$$\Sigma = (\{\mathbf{H}, \mathbf{M}\}, \{\mathbf{s}\}),$$

where **H** and **M** are predicate symbols of arity  $ar(\mathbf{H}) = ar(\mathbf{M}) = 1$ , and **s** is a functional symbol of arity  $ar(\mathbf{s}) = 0$ .

The statement *All men are mortal* will be modelled by the first-order formula

$$(\forall x.(\mathbf{H}(x) \rightarrow \mathbf{M}(x))),$$

whose abstract syntax tree is:



The statement *Scorates is a man* shall be modelled by the atomic formula  $\mathbf{H}(\mathbf{s})$ , and the statement *Socrates is mortal* by the atomic formula  $\mathbf{M}(\mathbf{s})$ .

For the signature  $\Sigma = (\{\mathbf{H}, \mathbf{M}\}, \{\mathbf{s}\})$  fixed above, there exist several possible  $\Sigma$ -structures. An example of a  $\Sigma$ -structure would be  $S = (D, \{\mathbf{H}^S, \mathbf{M}^S\}, \{\mathbf{s}^S\})$  defined as follows:

1.  $D$  is the set of all beings on Earth;
2.  $\mathbf{H}^S(x)$  is true for any being  $x$  that is a man;
3.  $\mathbf{M}^S(x)$  is true of any being  $x$  (all of the elements in the domain are mortal);
4.  $\mathbf{s}^S$  is Socrates (Socrates, being a being, belongs to the set  $D$ ).

Anticipating a little bit (we shall discuss the semantics of first-order formulae in the next lecture), all tree formulae discussed in this section,  $(\forall x.(\mathbf{H}(x) \rightarrow \mathbf{M}(x)))$ ,  $\mathbf{H}(\mathbf{s})$  and respectively  $\mathbf{M}(\mathbf{s})$ , are true in the structure  $S$  defined above.

In fact, the quality of the argument *All men are mortal; Socrates is a man; so: Socrates is mortal* is given by the fact that the formula  $\mathbf{M}(\mathbf{s})$  is necessarily true in *any* structure in which the formulae  $(\forall x.(\mathbf{H}(x) \rightarrow \mathbf{M}(x)))$  and respectively  $\mathbf{H}(\mathbf{s})$  are true, not just in the structure  $S$  above.

### 3.6 Model arithmetic sentences as formulae in FOL

Consider the signature  $\Sigma = (\{<, =\}, \{+, -, 0, 1\})$ , where  $<$  and  $=$  are predicate symbols of arity 2,  $+$  is a functional symbol of arity 2,  $-$  is a functional symbol of arity 1, and  $0$  and  $1$  are constant symbols.

Here are a few first-order formulae in the first-order language associated to the signature  $\Sigma$ :

1.  $(\forall x.(\forall y.(<(x, y) \rightarrow \exists z.(<(x, z) \wedge <(z, y))))$ ;
2.  $(\forall x.(\forall y.(\exists z.(=(+(x, y), z))))$ ;
3.  $(\forall x.(<(0, x) \vee =(0, x)))$ ;
4.  $(\forall x.(\exists y.(=(x, -(y))))$ ;
5.  $=(+(x, y), z)$ .

Many times, in the case of binary predicate symbols and binary functional symbols, we use the infix notation (e.g.,  $x < y$  instead of  $<(x, y)$ ). In this case, we could write the formulae above as follows:

1.  $(\forall x. (\forall y. (x < y \rightarrow \exists z. (x < z \wedge z < y))))$ ;
2.  $(\forall x. (\forall y. (\exists z. (x + y = z))))$ ;
3.  $(\forall x. (0 < x \vee 0 = x))$ ;
4.  $(\forall x. (\exists y. (x = -(y))))$ ;
5.  $x + y = z$ .

Two of the possible  $\Sigma$ -structures are  $S_1 = (\mathbb{R}, \{<, =\}, \{+, -, 0, 1\})$  and  $S_2 = (\mathbb{Z}, \{<, =\}, \{+, -, 0, 1\})$ , where the predicates and functions are those known from mathematics (with the remark that  $-$  is the unary minus function).

Anticipating the next lecture, on the semantics of first-order formulae, the first formula is false in  $S_2$  and true in  $S_1$ . The second and the fourth formula are true both in  $S_1$  and in  $S_2$ . The third formulae is false both in  $S_1$  and in  $S_2$ . The truth value of the fifth formula depends not only of the structure where we evaluate the truth value of the formula, but also on the values of the variables  $x, y, z$ . Because the variables  $x, y, z$  are not protected by a quantifier in formula number 5, they are called *free variables*. Formula number 5 is *satisfiable* both in the structure  $S_1$  as well as in the structure  $S_2$ , because in both cases there are values for the variables  $x, y, z$  that make the formula true (e.g. the values 1, 2, 3 for  $x, y$ , and respectively  $z$ ).

### 3.7 Exercises

**Exercise 24.** *Identify a signature for the following statements and model the statements as formulae in first-order logic over that signature.*

John is a student. Any student learns Logic. Anyone learning Logic passes the exam. Any student is a person. There is a person who did not pass the exam. Therefore: not all persons are students.

**Exercise 25.** *Consider  $S = (\mathbb{R}, \{Nat, Int, Prime, Even, >\}, \{+, 0, 1, 2\})$ , a structure where  $Nat$ ,  $Int$ ,  $Prime$ ,  $Even$  are unary predicates with the following meaning:  $Nat(u) = "u \text{ is a natural number}"$ ,  $Int(u) = "u \text{ is an integer number}"$ ,  $Prime(u) = "u \text{ is a prime}"$  and  $Even(u) = "u \text{ is an even number}"$ . The binary predicate  $>$  is the "greater than" relation over real numbers. The*

function  $+$  is the addition function for real numbers. The constants  $0, 1, 2$  are what you would expect.

1. Find a signature  $\Sigma$  so that the structure  $S$  is a  $\Sigma$  structure;
2. Model the following statements as first-order formulae in the signature associated to the structure  $S$  above:
  - (a) Any natural number is also an integer.
  - (b) The sum of any two natural numbers is a natural number.
  - (c) No matter how we would choose a natural number, there is prime number that is greater than the number we chose.
  - (d) If any natural number is a prime number, then zero is a prime number.
  - (e) No matter how we choose a prime number, there is a prime number greater than it.
  - (f) The sum of two even numbers is an even number.
  - (g) Any prime number greater than 2 is odd.
  - (h) Any prime number can be written as the sum of four prime numbers.
  - (i) The sum of two even numbers is an odd number.
  - (j) Any even number is the sum of two primes.

**Exercise 26.** Give examples of 5 terms over the signature in Exercise 25 (1.) and compute their abstract syntax tree.

**Exercise 27.** Give examples of 5 formulae over the signature in Exercise 25 (1.) and compute their abstract syntax tree.

**Exercise 28.** Compute the abstract syntax tree of the following formulae:

1.  $(P(x) \vee (P(y) \wedge \neg P(z)))$ ;
2.  $((\neg \neg P(x) \vee P(y)) \rightarrow (P(x) \wedge \neg P(z)))$ ;
3.  $(\forall x. (\forall y. ((\neg \neg P(x) \vee P(y)) \rightarrow (P(x) \wedge \neg P(z))))$ ;
4.  $(\forall x. (\forall y. ((\neg \neg P(x) \vee P(y)) \rightarrow (\exists x. (P(x) \wedge \neg P(x)))))$ ;
5.  $(\forall x'. \neg (\forall x. (P(x) \wedge (\exists y. ((Q(x, y) \vee \neg Q(z, z)) \rightarrow (\exists z'. P(z'))))))$ .