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

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 $\mathtt{p} \in A$.

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

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

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

which is read as follows: *for any* x*, if* Man *of* x*, then* Mortal *of* x. The statement *Socrate is a men* shall be modelled by the formula $\mathtt{Man}(s)$, 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 $(\forall x.(\mathtt{Man}(x) \rightarrow \mathtt{Mortal}(x)))$ 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:

$$\varphi = \Big(\forall \mathsf{x}.(\forall \mathsf{y}.(\mathsf{x} < \mathsf{y} \rightarrow \exists \mathsf{z}.(\mathsf{x} < \mathsf{z} \wedge \mathsf{z} < \mathsf{y})))\Big).$$

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 infixed notation $(x < y)$ instead of the prefixed notation $(<(x, y))$ for many binary predicates (including for $<$).

Is the formula φ 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 $\mathsf{x}, \mathsf{y}, \mathsf{z}$ 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* $f \in Pred$ *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 $+ : \mathbb{N}^2 \to \mathbb{N}$ is the addition function for naturals, and the functions $0 : \mathbb{N}^0 \to \mathbb{N}$ and respectively $1 : \mathbb{N}^0 \to \mathbb{N}$ 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:

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

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 Σ *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 (Σ-structure)**.** *If* $\Sigma = (\mathcal{P}, \mathcal{F})$ *is a signature, a* Σ-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* $\Sigma = (\{\mathtt{P}, \mathtt{Q}\}, \{\mathtt{f}, \mathtt{i}, \mathtt{a}, \mathtt{b}\})$*, where* P *and* Q *are predicate symbols of arity* $ar(\mathtt{P}) = ar(\mathtt{Q}) = 2$ *and* f, i, a, b *are function symbols having the following arrities:* $ar(\mathtt{f}) = 2$*,* $ar(\mathtt{i}) = 1$ *and* $ar(\mathtt{a}) = ar(\mathtt{b}) = 0$*.*

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

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

To remember!

Structure = domain + predicates + functions

Signature = predicate symbols + functional symbols

To a signature Σ we can associate many structures, which are called Σ-structures.

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

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 Σ. In propositional logic, there was just one language, $\mathbb{PL}$.

Next, we shall fix a signature Σ 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: $\neg, \wedge, \vee, \rightarrow, \leftrightarrow, \bot$, as well as two new *quantifiers*: $\forall$, $\exists$;
2. variables: we will assume that a countably infinite set of variables $\mathcal{X} = \{\mathsf{x}, \mathsf{y}, \mathsf{z}, \mathsf{x}', \mathsf{y}', \mathsf{x}_1, \mathsf{z}'', \ldots\}$ 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: $($, $)$, $.$, $,$, $($, $)$, 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 5. *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* $t_1, \ldots, t_n \in \mathcal{T}$*, then* $f(t_1, \ldots, t_n) \in \mathcal{T}$ *(a functional symbol of arity* n *applied to* n *terms is a term).*

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

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

Example 6. *Recall* $\Sigma = (\{\mathtt{P}, \mathtt{Q}\}, \{\mathtt{f}, \mathtt{i}, \mathtt{a}, \mathtt{b}\})$ *from Example 4, where* $ar(\mathtt{P}) = ar(\mathtt{Q}) = 2$*,* $ar(\mathtt{f}) = 2$*,* $ar(\mathtt{i}) = 1$*,* $ar(\mathtt{a}) = ar(\mathtt{b}) = 0$*. Here are several example of terms:* $\mathtt{a}$*,* $\mathtt{b}$*,* $\mathtt{x}$*,* $\mathtt{y}$*,* $\mathtt{x}_1$*,* $\mathtt{y}'$*,* $\mathtt{i(a)}$*,* $\mathtt{i(x)}$*,* $\mathtt{i(i(a))}$*,* $\mathtt{i(i(x))}$*,* $\mathtt{f(a,b)}$*,* $\mathtt{i(f(a,b))}$*,* $\mathtt{f(f(x,a),f(y,y))}$*.*

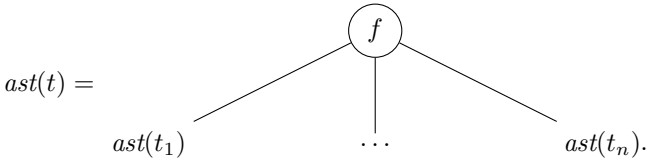
Exercise 7. *Identify the* Σ*-terms in the following list :*

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

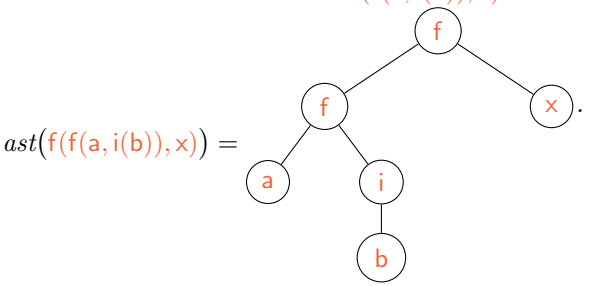
We denote terms by the letters t, s, t_1, t_2, s_1, t', etc. Even if terms are usually written as a string of characters, they have an associated abstract syntax tree defined as follows:

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

3. if $t = f(t_1, \ldots, t_n)$ and $f \in \mathcal{F}_n$ $(n > 0)$, $t_1, \ldots, t_n \in \mathcal{T}$, 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* $\mathsf{f(f(a,i(b)),x)}$*:*



Exercise 8. *Compute the abstract syntax trees for all terms in Example 6.*

3.3 Atomic formulae

Definition 9 (Atomic formula)**.** *An* atomic formula *is any string of characters of the form* $P(t_1, \ldots, t_n)$*, unde* $P \in \mathcal{P}_n$ *, where* $P \in \mathcal{P}_n$ *is a predicate symbol of arity* n*, and* $t_1, \ldots, t_n \in \mathcal{T}$ *are terms.*

Example 10. *Continuing the previous example, we work over the signature*

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

where $ar(\mathsf{P}) = ar(\mathsf{Q}) = 2$*,* $ar(\mathsf{f}) = 2$*,* $ar(\mathsf{i}) = 1$*,* $ar(\mathsf{a}) = ar(\mathsf{b}) = 0$*.*

Here are a few examples of atomic formulae: $\mathsf{P(a,b)}$*,* $\mathsf{P(x,y)}$*,* $\mathsf{Q\Big(i(i(x)),f(x,x)\Big)}$*,* $\mathsf{Q(a,b)}$*,* $\mathsf{P\Big(f\big(f(a,i(x)),b\big),i(x)\Big)}$*.*

Exercise 11. *Explain why* $\mathsf{P(a)}$*,* P*,* $\mathsf{i(i(x))}$ *are not atomic formulae over the signature in Example 10.*

3.4 First-Order Formulae

Definition 12 (First-Order Formula)**.** *The set of first-order formulae, written* $\mathbb{FOL}$*, is the smallest set with the following properites:*

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

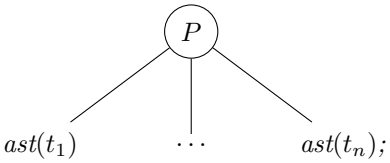
Remark. *In Definition 12, we find the logical connectives* $\neg, \wedge, \vee, \to$ *and* $\leftrightarrow$ *from propositional logic. The predicate symbols of arity* 0 *play the role of propositional variables (for now, at the syntactic level). The constructions* $(\forall x.\varphi)$ *and* $\exists x.\varphi)$ *are new.*

Example 13. *Recall* $\Sigma = (\{\mathtt{P}, \mathtt{Q}\}, \{\mathtt{f}, \mathtt{i}, \mathtt{a}, \mathtt{b}\})$ *from Example 6, where* $ar(\mathtt{P}) = ar(\mathtt{Q}) = 2$*,* $ar(\mathtt{f}) = 2$*,* $ar(\mathtt{i}) = 1$ *și* $ar(\mathtt{a}) = ar(\mathtt{b}) = 0$*.*
Here are several example of first-order formulae:

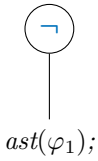
1. $\mathtt{P}(\mathtt{a},\mathtt{b})$*;*
2. $\mathtt{Q}(\mathtt{a},\mathtt{b})$*;*
3. $\mathtt{P}(\mathtt{a},\mathtt{x})$*;*
4. $\neg\mathtt{P}(\mathtt{a},\mathtt{b})$*;*
5. $(\mathtt{P}(\mathtt{a},\mathtt{b}) \wedge \neg\mathtt{Q}(\mathtt{a},\mathtt{b}))$*;*
6. $(\mathtt{P}(\mathtt{a},\mathtt{b}) \vee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

Definition 14 (The Abstract Syntax Tree of formulae in $\mathbb{FOL}$)**.** *Formulae have an associated abstract syntax tree defined as follows:*

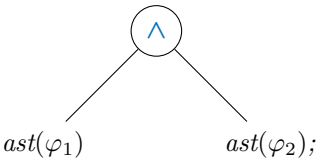
1. *if* $\varphi = P(t_1, \ldots, t_n)$, *then* $ast(\varphi) =$



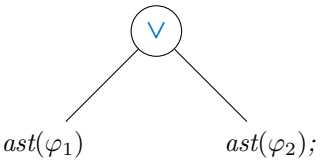
2. *if* $\varphi = \neg\varphi_1$, *then* $ast(\varphi) =$



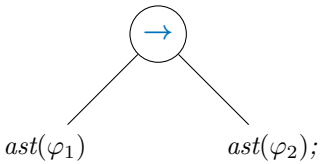
3. *if* $\varphi = (\varphi_1 \wedge \varphi_2)$, *then* $ast(\varphi) =$



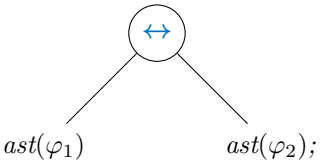
4. *if* $\varphi = (\varphi_1 \vee \varphi_2)$, *then* $ast(\varphi) =$



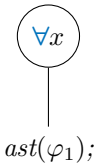
5. *if* $\varphi = (\varphi_1 \rightarrow \varphi_2)$, *then* $ast(\varphi) =$

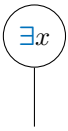


6. *if* $\varphi = (\varphi_1 \leftrightarrow \varphi_2)$, *then* $ast(\varphi) =$



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



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

Exercise 15. *Compute the abstract syntax tree of formulas shown in Example 13.*

3.5 The Brackets

The brackets (and) are used to mark the order of carrying out the logical operations (and, or, not, etc.). Next, we will drop certain extra brackets, just like in the case of propositional logic: if a formula can be interpreted as an abstract syntax in two or more ways, we will use brackets to fix the desired tree.

For example, $\varphi_1 \vee \varphi_2 \wedge \varphi_3$ could be understood as $((\varphi_1 \vee \varphi_2) \wedge \varphi_3)$ or as $(\varphi_1 \vee (\varphi_2 \wedge \varphi_3))$. In order to save brackets, we establish the following priority order of logical connectives:

$$\neg, \wedge, \vee, \rightarrow, \leftrightarrow, \forall, \exists,$$

where $\neg$ has the highest priority and $\exists$ has the lowest priority. When we are not 100% sure, it is better to use extra brackets.

Because of the order of priority for logical connectives, $\varphi_1 \vee \varphi_2 \wedge \varphi_3$ shall always be understood as $(\varphi_1 \vee (\varphi_2 \wedge \varphi_3))$ (because $\wedge$ has priority over $\vee$). As an anology, it works the same way as in arithmetic: $1 + 2 * 3$ will be understood as $1 + (2 \times 3)$, because $\times$ has priority over $+$ ($\times$ is similar to $\wedge$ and $+$ to $\vee$).

Exercise 16. *Write the formulas in Example 13 using the minimal number of brackets.*

In Section 3.12 we will discuss more about the interaction between the cuantifiers and the other logical connectives. We will see that we have some extra rules besides the above priorities.

3.6 Model english sentences as formulas in $\mathbb{FOL}$

Next, we will explain the signature used to model in first-order logic the statments *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 $\mathtt{H}$ for the first predicate and the predicate symbol $\mathtt{M}$ for the second predicate. We also have one constant in the text: Socrates. We choose the functional symbol $\mathtt{s}$ (of arity 0) for this constant. Therefore, to model the statements above, we shall work in the signature

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

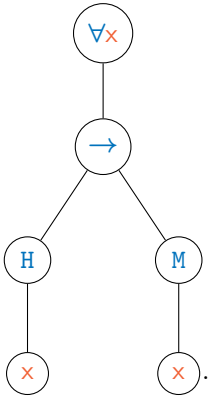
where $\mathtt{H}$ and $\mathtt{M}$ are predicate symbols of arity $ar(\mathtt{H}) = ar(\mathtt{M}) = 1$, and $\mathtt{s}$ is a functional symbol of arity $ar(\mathtt{s}) = 0$.

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

$$(\forall \mathtt{x}.(\mathtt{H}(\mathtt{x}) \to \mathtt{M}(\mathtt{x}))),$$

whose abstract syntax tree is:

$$ast\Big((\forall \mathtt{x}.(\mathtt{H}(\mathtt{x}) \to \mathtt{M}(\mathtt{x})))\Big) =$$



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

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

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

Anticipanting a little bit (we shall discuss the semantics of first-order formulae in the next lecture), all tree formulae discussed in this section, $(\forall x.(H(x) \to M(x)))$, $H(s)$ and respectively $M(s)$, 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 $M(s)$ is necessarily true in *any* structure in which the formulae $(\forall x.(H(x) \to M(x)))$ and respectively $H(s)$ are true, not just in the structure S above.

3.7 Model arithmetic sentences as formulas in $\mathbb{FOL}$

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 Σ:

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

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

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

Two of the possible Σ-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).

Anticipanding 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 evalute the truth value of the formula, but also on the values of the variables $\mathtt{x}, \mathtt{y}, \mathtt{z}$. Because the variables $\mathtt{x}, \mathtt{y}, \mathtt{z}$ 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 $\mathtt{x}, \mathtt{y}, \mathtt{z}$ that make the formula true (e.g. the values $1, 2, 3$ for $\mathtt{x}, \mathtt{y}$, and respectively $\mathtt{z}$).

3.8 The Variables of a Formula

By $vars(\varphi)$ we denote the variables of the formula φ. For example, we shall have $vars\big((\forall \mathtt{z}.(\mathtt{P}(\mathtt{x}, \mathtt{y})))\big) = \{\mathtt{x}, \mathtt{y}, \mathtt{z}\}$. We next define the function $vars : \mathbb{FOL} \to 2^{\mathcal{X}}$.

First of all, we define a function $vars : \mathcal{T} \to 2^{\mathcal{X}}$ as being the function that associates to a term (from the set $\mathcal{T}$) the set of variables occurring in that term. All following definitions are defined inductively, as the corresponding syntactic definitions are. We call them recursive definitions and we do not explicitly differentiate the base case and the inductive cases. We recall that the set $2^{\mathcal{X}}$ is the set of all subsets of $\mathcal{X}$.

Definition 17. *The function $vars : \mathcal{T} \to 2^{\mathcal{X}}$ is defined recursively as follows:*

1. $vars(c) = \emptyset$ *(if $c \in \mathcal{F}_0$ is a constant symbol);*
2. $vars(x) = \{x\}$ *(if $x \in \mathcal{X}$ is a variable);*
3. $vars(f(t_1, \ldots, t_n)) = \bigcup_{i \in \{1,\ldots,n\}} vars(t_i)$.

We can now define the homonymous (extended) function $vars : \mathbb{FOL} \to 2^{\mathcal{X}}$, which associates to any first-order formula the set of variables of the formula:

Definition 18. *The function $vars : \mathbb{FOL} \to 2^{\mathcal{X}}$ is define recursively as follows:*

1. $vars(P(t_1, \ldots, t_n)) = \bigcup_{i \in \{1,\ldots,n\}} vars(t_i)$;

2. $vars(\neg\varphi) = vars(\varphi)$;
3. $vars((\varphi_1 \wedge \varphi_2)) = vars(\varphi_1) \cup vars(\varphi_2)$;
4. $vars((\varphi_1 \vee \varphi_2)) = vars(\varphi_1) \cup vars(\varphi_2)$;
5. $vars((\varphi_1 \rightarrow \varphi_2)) = vars(\varphi_1) \cup vars(\varphi_2)$;
6. $vars((\varphi_1 \leftrightarrow \varphi_2)) = vars(\varphi_1) \cup vars(\varphi_2)$;
7. $vars\big((\exists x.\varphi)\big) = vars(\varphi) \cup \{x\}$;
8. $vars\big((\forall x.\varphi)\big) = vars(\varphi) \cup \{x\}$.

Observe that the variable x is added to the set of variables even if it appears only as next to the quantifier $\exists$ or $\forall$.

Example 19. *Consider the formula φ:*

$$\Big(\big(\forall \mathtt{x}.(\mathtt{P}(\mathtt{x},\mathtt{y}) \wedge \exists \mathtt{y}.(\mathtt{P}(\mathtt{z},\mathtt{f}(\mathtt{x},\mathtt{y})) \wedge \mathtt{P}(\mathtt{x},\mathtt{y})))\big) \wedge \mathtt{P}(\mathtt{x},\mathtt{x})\Big).$$

We have that $vars(\varphi) = \{\mathtt{x},\mathtt{y},\mathtt{z}\}$.

Exercise 20. *Consider $\Sigma = (\{\mathtt{P},\mathtt{Q}\},\{\mathtt{f},\mathtt{i},\mathtt{a},\mathtt{b}\})$, wher $ar(\mathtt{P}) = ar(\mathtt{Q}) = 2$, $ar(\mathtt{f}) = 2$, $ar(\mathtt{i}) = 1$, $ar(\mathtt{a}) = ar(\mathtt{b}) = 0$. Compute $vars(\varphi)$ for each formula φ in the list below:*

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

3.9 The Scope of a Quantifier - Analogy with Programming Languages

In a programming language, we may declare several variables having the same name. For example, in C, we could have the following code:

```
/*  1:*/ int f()
/*  2:*/ {
/*  3:*/   int s = 0;
/*  4:*/   for (int x = 1; x <= 10; ++x) {
/*  5:*/     for (int y = 1; y <= 10; ++y) {
/*  6:*/       s += x * y * z;
/*  7:*/       for (int x = 1; x <= 10; ++x) {
/*  8:*/         s += x * y * z;
/*  9:*/       }
/* 10:*/     }
/* 11:*/   }
/* 12:*/   return s;
/* 13:*/ }
```

In the code fragment above, there are three variables, two of them having the same name, `x`. The scope of the variable `x` declared on line 4 is between the lines $4-11$, and the scope of the variable `x` declared on line 7 is the lines $7-9$. This way, any occurence of the name `x` between lines $7-9$ refers to the second declaration of the variable, while any occurrence of the name `x` between lines $4-11$ (except lines $7-9$) refers to the second declaration of `x`. For example, the occurrence of `x` on line 6 refers to the variable x declared on line 4. The occurrence of `x` on line 8 refers to the variable x declared on line 7.

The lines $4-11$ represent the scope of the first declaration of `x`, and the lines $7-9$ represent the scope of the second declaration of `x`. The variable z is a global variable.

This is similar to first-order formulae. For example, in the formula:

$$\Big(\forall x.\big(\forall y.(P(x,y) \wedge P(x,z) \wedge (\exists x.P(x,y)))\big)\Big),$$

the variable x is quantified twice (the first time universally, the second time existentially). A quantification of a variable is called *binding*. A *binding* is similar, from the point of view of the scope of the variable, to defining a variable in a programming language.

This way, the scope D_1 of the variable x that is quantified universally is $\big(\forall y.(P(x,y) \wedge P(x,z) \wedge (\exists x.P(x,y)))\big)$, while the scope D_2 of the variable x that is quantified existentially is $P(x,y)$:

$$\Big(\forall \mathtt{x}.\ \underbrace{\big(\forall \mathtt{y}.\big(\mathtt{P}(\mathtt{x},\mathtt{y}) \wedge \mathtt{P}(\mathtt{x},\mathtt{z}) \wedge (\exists \mathtt{x}.\ \overbrace{\mathtt{P}(\mathtt{x},\mathtt{y})}^{\mathtt{D_2}})\big)\big)}_{\mathtt{D_1}}\Big)$$

The occurrences of a variable in the scope of a quantifier that binds the variable are called *bound occurrences*, while the occurrences of a variable outside of the scope of any quantifier that binds the variable are called *free occurrences*.

3.10 Free and Bound Occurrences of Variables

In this section we formally define the notion of free/bound occurrence and of free/bound variable. The free occurrences of a variable in first-order logic are, as an analogy, similar to global variables in a programming language.

Recall the definition of the *syntax abstract tree* associated to a first order formula and consider that the sentence *"on the path to the root"* has a formal definition.

Definition 21. *A* free occurrence of a variable x in a formula φ *is a node in the tree of the formula* φ *labeled by* x *and having the property that there is no node labeled* $\forall x$ *or* $\exists x$ *on the path to the root.*

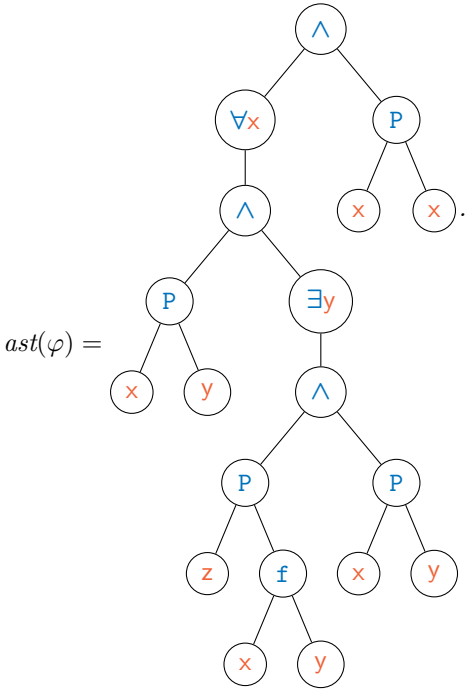
Definition 22. *A* bound occurrence of a variable x in a formula φ *is a node in the tree of the formula labeled by* x *and having the property that, on the path towards the root, there is at least a node labeled by* $\forall x$ *or by* $\exists x$*.*

The closest such node (labeled by $\forall x$ *or by* $\exists x$*) is the quantifier that* binds *that particular occurrence of the variable* x*.*

Example 23. *We next consider the formula*

$$\varphi = \Big(\Big(\forall \mathtt{x}.\big(\mathtt{P}(\mathtt{x},\mathtt{y}) \wedge \exists \mathtt{y}.\big(\mathtt{P}(\mathtt{z},\mathtt{f}(\mathtt{x},\mathtt{y})) \wedge \mathtt{P}(\mathtt{x},\mathtt{y})\big)\big)\Big) \wedge \mathtt{P}(\mathtt{x},\mathtt{x})\Big).$$

Its abstract syntax tree is:



In the formula φ above, the variable x has two free occurrences. The variable y has one free occurrence. The variable z has one free occurrence. All the free occurrences of the variables in the formula φ are underlined:

$$\varphi = \Big(\Big(\forall \mathtt{x}.\big(\mathtt{P}(\underline{\mathtt{x}},\underline{\mathtt{y}}) \wedge \exists \mathtt{y}.\big(\mathtt{P}(\underline{\mathtt{z}},\mathtt{f}(\underline{\mathtt{x}},\underline{\mathtt{y}})) \wedge \mathtt{P}(\underline{\mathtt{x}},\underline{\mathtt{y}})\big)\big)\Big) \wedge \mathtt{P}(\underline{\mathtt{x}},\underline{\mathtt{x}})\Big).$$

All the bound occurrences of the variables in the formula φ are underlined twice:

$$\varphi = \Big(\Big(\forall \mathtt{x}.\big(\mathtt{P}(\underline{\underline{\mathtt{x}}},\mathtt{y}) \wedge \exists \mathtt{y}.\big(\mathtt{P}(\mathtt{z},\mathtt{f}(\underline{\underline{\mathtt{x}}},\underline{\underline{\mathtt{y}}})) \wedge \mathtt{P}(\underline{\underline{\mathtt{x}}},\underline{\underline{\mathtt{y}}})\big)\big)\Big) \wedge \mathtt{P}(\mathtt{x},\mathtt{x})\Big).$$

Remark. *Some authors also consider that the nodes labeled by $\forall x$ or by $\exists x$ are bound occurrences of the variable x. In this course, we shall not consider the nodes $\forall x$ and respectively $\exists x$ as being occurrences of the variable x, but simply as binding sites for the variable x.*

3.11 Free Variables and Bound Variables

The set of variables that have at least one free occurrence in a formula φ is denoted $free(\varphi)$.

Definition 24. *The function* $free : \mathbb{FOL} \to 2^{\mathcal{X}}$ *is defined recursively as follows:*

1. $free(P(t_1, \ldots, t_n)) = vars(t_1) \cup \ldots \cup vars(t_n)$;
2. $free(\neg\varphi) = free(\varphi)$;
3. $free((\varphi_1 \wedge \varphi_2)) = free(\varphi_1) \cup free(\varphi_2)$;
4. $free((\varphi_1 \vee \varphi_2)) = free(\varphi_1) \cup free(\varphi_2)$;
5. $free((\varphi_1 \to \varphi_2)) = free(\varphi_1) \cup free(\varphi_2)$;
6. $free((\varphi_1 \leftrightarrow \varphi_2)) = free(\varphi_1) \cup free(\varphi_2)$;
7. $free\big((\forall x.\varphi)\big) = free(\varphi) \setminus \{x\}$;
8. $free\big((\exists x.\varphi)\big) = free(\varphi) \setminus \{x\}$.

Example 25. *Continuing the previous example, for the formula*

$$\varphi = \Big(\Big(\forall \mathtt{x}.\big(\mathtt{P}(\mathtt{x},\mathtt{y}) \wedge \exists \mathtt{y}.(\mathtt{P}(\mathtt{z},\mathtt{f}(\mathtt{x},\mathtt{y})) \wedge \mathtt{P}(\mathtt{x},\mathtt{y}))\big)\Big) \wedge \mathtt{P}(\mathtt{x},\mathtt{x})\Big),$$

we have that $free(\varphi) = \{\mathtt{x}, \mathtt{y}, \mathtt{z}\}$.

Exercise 26. *Compute* $vars(\varphi)$ *for each formula* φ *in Exercise 20.*

By $bound(\varphi)$ we denote the set of variables bound in a formula, that is the set of those variables x with the property that there exists in the formula at least one node labeled by $\forall x$ or by $\exists x$.

Definition 27. *The function* $bound : \mathbb{FOL} \to 2^{\mathcal{X}}$ *is defined recursively as follows:*

1. $bound(P(t_1, \ldots, t_n)) = \emptyset$;
2. $bound(\neg\varphi) = bound(\varphi)$;
3. $bound((\varphi_1 \wedge \varphi_2)) = bound(\varphi_1) \cup bound(\varphi_2)$;
4. $bound((\varphi_1 \vee \varphi_2)) = bound(\varphi_1) \cup bound(\varphi_2)$;

5. $bound((\varphi_1 \to \varphi_2)) = bound(\varphi_1) \cup bound(\varphi_2)$;
6. $bound((\varphi_1 \leftrightarrow \varphi_2)) = bound(\varphi_1) \cup bound(\varphi_2)$;
7. $bound((\forall x.\varphi)) = bound(\varphi) \cup \{x\}$;
8. $bound((\exists x.\varphi)) = bound(\varphi) \cup \{x\}$.

Exercise 28. *Compute $bound(\varphi)$ for each formula φ in Exercise 20.*

Exercise 29. *Compute $bound(\varphi)$, where*

$$\varphi = \Big(\Big(\forall \mathtt{x}.\big(\mathtt{P}(\mathtt{x},\mathtt{y}) \wedge \exists \mathtt{y}.\big(\mathtt{P}(\mathtt{z},\mathtt{f}(\mathtt{x},\mathtt{y})) \wedge \mathtt{P}(\mathtt{x},\mathtt{y})\big)\big)\Big) \wedge \mathtt{P}(\mathtt{x},\mathtt{x})\Big).$$

Definition 30 (Bound Variables of a Formula)**.** *The* bound variables of a formula φ *are the elements of* $bound(\varphi)$.

Definition 31 (Free Variables of a Formula)**.** *The* free variables of a formula φ *are the elements of the set* $free(\varphi)$.

Remark. *The sets $free(\varphi)$ and $bound(\varphi)$ could in general have common elements.*

Remark. *A variable can have several occurrences in a formula. Some of the occurrences could be free in the formula, while others could be bound.*

We must make the distinction between a free occurrence of a variable in a formula *and a* free variable of a formula*. The free occurrence is a node in the abstract syntax tree of the formula, while the free variable is an element of the set $\mathcal{X}$.*

We must also make the distinction between a bound occurrence of a variable in a formula *and a* bound variable of a formula*. The bound occurrence is a node in the abstract syntax tree, while the bound variable is an element of the set $\mathcal{X}$.*

3.12 The Scope of a Bound Variable and Brackets

Now that we have understood the scope of a bound variable, we may clarify a subtlety related to the order of priority of logical connectives. Up to this point, we have established that the order of priority is: $\neg, \wedge, \vee, \to, \leftrightarrow, \forall, \exists$. However, the quantifiers, $\forall$ and $\exists$, interact subtly with the other logical connectives, making parathesizing a formula more complicated.

Clarification: a formula should be paranthesized such that the scope of a bound variable extends as far to the right as possible.

For example, the formula:

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

should be paranthesized as follows:

$$\Big(\forall x.\big(P(x,x) \vee (\neg(\exists y.(P(x,y) \wedge P(x,x))))\big)\Big).$$

3.13 Exercises

Exercise 32. *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 33. *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 is a natural number,* $Int(u)$ = *"u is an integer number",* $Prime(u)$ = *"u is a prime" and* $Even(u)$ = *"u 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* Σ *so that the structure* S *is a* Σ *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 34. *Give examples of* 5 *terms over the signature in Exercise 33 (1.) and compute their abstract syntax tree.*

Exercise 35. *Give examples of* 5 *formulas over the signature in ExerciseExercise 33 (1.) and compute their abstract syntax tree.*

Exercise 36. *Compute the abstract syntax tree of the following formulae (hint: place brackets around subformulae, in the priority order of the logical connectives):*

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

Exercise 37. *Mark the free occurrences and respectively the bound occurrences of the variables in the formulae below:*

1. $\varphi_1 = (\forall \mathtt{x}.\mathtt{P}(\mathtt{x},\mathtt{x}) \wedge \mathtt{P}(\mathtt{x},\mathtt{y})) \wedge \mathtt{P}(\mathtt{x},\mathtt{z})$;
2. $\varphi_2 = (\forall \mathtt{x}.\mathtt{P}(\mathtt{f}(\mathtt{x},\mathtt{x}),\mathtt{i}(\mathtt{x})) \wedge \exists \mathtt{y}.(\mathtt{P}(\mathtt{x},\mathtt{y}) \wedge \mathtt{P}(\mathtt{x},\mathtt{z})))$.

Exercise 38. *Identify the scope of the quantifiers in the formulae φ_1 and φ_2 from Exercise 37.*

Exercise 39. *Compute the variables, the free variables and respectively the bound variables in the formulae φ_1 and φ_2 from Exercise 37.*

Exercise 40. *Let $A = \{\mathtt{p},\mathtt{q},\mathtt{r},\ldots\}$ be the set of propositional variables. We consider the signature $\Sigma_{\mathbb{PL}} = (A,\emptyset)$, where the propositional variables in A are predicate symbols of arity* 0.

1. *Prove that for any formula $\varphi \in \mathbb{PL}$ we have $\varphi \in \mathbb{FOL}$ (over signature $\Sigma_{\mathbb{PL}}$).*
2. *Prove that for any cuantifier free formula $\varphi \in \mathbb{FOL}$ over $\Sigma_{\mathbb{PL}}$, we have $\varphi \in \mathbb{PL}$.*