Chapter 1

Syntax and Semantics of Predicate Logic

Courses on logic generally start with propositional logic, which is more suitable for initial exposition due to its simplicity. However, the full power of logic in computer science only manifests itself with the use of predicate logic. Let us begin with an informal introduction that illustrates the basic aspects of predicate logic. We will return to a formal exposition in the following sections.

1.1 Introduction

Recall that in propositional logic, we described the world using *propositions* composed of *atomic propositions*—answers to yes/no questions about the world. In (first-order[1]) predicate logic the basic building blocks are *variables* representing *individuals*—indivisible objects from some set: e.g., natural numbers, vertices of a graph, or states of a microprocessor.

These individuals can have certain properties and relationships, which we call *predicates*, e.g., '$\mathrm{Leaf}(x)$' or '$\mathrm{Edge}(x,y)$' when talking about a graph, or '$x \leq y$' in natural numbers. In addition, individuals can be passed into functions, e.g., '$\mathrm{lowest\_common\_ancestor}(x,y)$' in a rooted tree, '$\mathrm{succ}(x)$' or '$x+y$' in natural numbers, and they can be *constants* with special meaning, e.g., 'root' in a rooted tree, '0' in natural numbers.

Atomic formulas describe a predicate (including *equality* predicate $=$) about variables or *terms* ('expressions' composed[2] of functions or constants). More complex statements (*formulas*) are built from atomic formulas using logical connectives and two *quantifiers*:

- $\forall x$ "for all individuals (represented by variable x)," and
- $\exists x$ "there exists an individual (represented by variable x)".

Let us look at an example: the statement *"Everyone who has a child is a parent."* could be formalized by the following formula:

$$(\forall x)((\exists y)\mathrm{child\_of}(y,x) \to \mathrm{is\_parent}(x))$$

[1]In second-order logic, we also have variables representing sets of individuals or even sets of n-tuples, i.e., relations on the set of individuals.

[2]Similarly to how we create arithmetic expressions.

where child_of(y, x) is a binary predicate expressing that the individual represented by variable y is a child of the individual represented by variable x, and is_parent(x) is a unary predicate (i.e., a 'property') expressing that the individual represented by x is a parent.

What about the validity of this formula? That depends on the specific *model* of the world/system we are interested in. A model is a (non-empty) set of objects together with a unary relation (i.e., a subset) *interpreting* the *unary relation symbol* is_parent and a binary relation interpreting the *binary relation symbol* child_of. These relations can generally be arbitrary, and it is easy to construct a model where the formula is not valid.[3] However, if we model, for example, all people in the world, and the relations have their natural meaning, then the formula will be valid.[4]

Let us look at another example, this time with function symbols and a constant symbol: "If $x_1 \leq y_1$ and $x_2 \leq y_2$, then $(y_1 \cdot y_2) - (x_1 \cdot x_2)$ is nonnegative." The resulting formula could look like this:

$$\varphi = (x_1 \leq y_1) \wedge (x_2 \leq y_2) \rightarrow ((y_1 \cdot y_2) + (-(x_1 \cdot x_2)) \geq 0)$$

We see two binary relation symbols $(\leq, \geq)$, a binary function symbol $+$, a unary function symbol $-$, and a constant symbol 0.

An example of a model in which the formula is valid is the set of natural numbers $\mathbb{N}$ with binary relations $\leq^{\mathbb{N}}, \geq^{\mathbb{N}}$, binary functions $+^{\mathbb{N}}, \cdot^{\mathbb{N}}$, unary function $-^{\mathbb{N}}$, and the constant $0^{\mathbb{N}} = 0$. However, if we similarly take the set of integers, the formula will no longer be valid.

Remark 1.1.1. We could understand the symbol $-$ as a binary operation, but it is usually introduced as unary. For the constant symbol 0, we use (as is customary) the same symbol as for the natural number 0. But note that in our model, this constant symbol could be interpreted as a different number, or our model might not consist of numbers at all!

There are no quantifiers in the formula (such formulas are called *open*), the variables x_1, x_2, y_1, y_2 are *free variables* of this formula (they are not *bound* by any quantifier), we write $\varphi(x_1, x_2, y_1, y_2)$. We understand the semantics of this formula in the same way as the formula

$$(\forall x_1)(\forall x_2)(\forall y_1)(\forall y_2)\varphi(x_1, x_2, y_1, y_2)$$

The expression $(y_1 \cdot y_2) + (-(x_1 \cdot x_2))$ is an example of a *term*, and the expressions $(x_1 \leq y_1)$, $(x_2 \leq y_2)$ and $((y_1 \cdot y_2) + (-(x_1 \cdot x_2)) \geq 0)$ are *atomic (sub)formulas*. What is the difference? Given a specific model and a specific *variable assignment* by individuals (elements) of this model, atomic formulas can be assigned a truth value. Therefore, they can be combined with logical connectives into more complex 'logical expressions', i.e., formulas. On the other hand, the 'result' of a term (under a given variable assignment) is some specific individual from the model.

We also note that in the formula φ, we used infix notation for the function symbols $+, \cdot$ and for the relations $\leq, \geq$, and similar conventions for parentheses as in propositional logic. Otherwise, we would write the formula φ as follows:

$$((\leq (x_1, y_1) \wedge \leq (x_2, y_2)) \rightarrow \leq (+(\cdot(y_1, y_2), -(\cdot(x_1, x_2))), 0))$$

[3]For example, take a single-element set $A = \{a\}$, and the relations child_of$^A = \{(a, a)\}$, parent$^A = \emptyset$; here the only object is its own child, but it is not a parent.

[4]When formalizing, we must be very careful not to add additional assumptions that may not hold in the modeled system. Here, for example, there is an implicit assumption that if someone has a child, they must be the child's parent.

Exercise 1.1. Find a suitable definition for the notion of a *tree of a formula* (generalizing the *tree of a proposition* from propositional logic). Draw the tree for the following formula from the above example: $(\forall x_1)(\forall x_2)(\forall y_1)(\forall y_2)\varphi(x_1, x_2, y_1, y_2)$.

Now, let us start by formalizing this notion of a "*model*", a so-called *structure*. The rest of the chapter follows the outline of the exposition on propositional logic: we will introduce the syntax, then the semantics, and finally the more advanced properties of formulas, theories, and structures. At the end, we will show a simple but very useful application of predicate logic, called *definability* of subsets and relations, which is the basis of *relational databases* (e.g., SQL), and once again look at the relationship between propositional and predicate logic.

1.2 Structures

First, we specify what *type* the given structure will be, i.e., what relations, functions (of which arities), and constants it will have, and what symbols we will use for them. This formal specification is sometimes called a *type*, but we will call it a *signature*.[5] Recall that we can consider *constants* as functions of arity 0 (i.e., functions without inputs).

Definition 1.2.1. A *signature* is a pair $\langle \mathcal{R}, \mathcal{F} \rangle$, where $\mathcal{R}, \mathcal{F}$ are disjoint sets of symbols (*relation* and *function*, the latter include *constant* symbols) with given arities (i.e., given by a function $\mathrm{ar}\colon \mathcal{R} \cup \mathcal{F} \to \mathbb{N}$) and not containing the symbol '=' (which is reserved for *equality*).

However, we will often write a signature just by listing the symbols, where their arity and whether they are relation or function symbols will be clear from the context. Here are a few examples of signatures:

- $\langle E \rangle$ the signature of *graphs*: E is a binary relation symbol (structures are directed graphs),
- $\langle \leq \rangle$ the signature of *partial orders*: the same as the signature of graphs, just a different symbol,[6]
- $\langle +, -, 0 \rangle$ the signature of *groups*: $+$ is a binary function, $-$ a unary function, 0 a constant symbol
- $\langle +, -, 0, \cdot, 1 \rangle$ the signature of *fields*: $\cdot$ is a binary function, 1 a constant symbol
- $\langle +, -, 0, \cdot, 1, \leq \rangle$ the signature of *ordered fields*: $\leq$ is a binary relation symbol,
- $\langle -, \wedge, \vee, \perp, \top \rangle$ the signature of *Boolean algebras*: $\wedge, \vee$ are binary function symbols, $\perp, \top$ are constant symbols,
- $\langle S, +, \cdot, 0, \leq \rangle$ the signature of *arithmetic*: S is a unary function symbol ('successor').

[5]You can think of a signature as similar to the definition of a *class* in OOP; structures then correspond to *objects* of this class (in the 'programming language' of set theory).

[6]Not every structure in this signature is a partial order; for that, it needs to satisfy the corresponding *axioms*.

In addition to common symbols for relations, functions, and constants (familiar e.g. from arithmetic), we typically use $P, Q, R, \dots$ for relation symbols, $f, g, h, \dots$ for function symbols, and $c, d, a, b, \dots$ for constant symbols.

A *structure* of a given signature is obtained by taking some non-empty *domain* and choosing *interpretations* (also called *realizations*) of all relation and function symbols (and constants) on that domain, i.e., specific relations or functions of the appropriate arities. (In the case of a constant symbol, its interpretation is a chosen element from the domain.)[7]

Example 1.2.2. The formal definition of a *structure* is given below; first, let us show a few examples:

- A structure in the empty signature $\langle\ \rangle$ is any non-empty set.[8] (It does not have to be finite, not even countable!)
- A structure in the signature of graphs is $\mathcal{G} = \langle V, E\rangle$, where $V \neq \emptyset$ and $E \subseteq V^2$, called a *directed graph.*
  - If E is irreflexive and symmetric, it is a *simple* graph (i.e., undirected, without loops).
  - If E is reflexive, transitive, and antisymmetric, it is a *partial order.*
  - If E is reflexive, transitive, and symmetric, it is an *equivalence relation.*
- Structures in the signature of partial orders are the same as in the signature of graphs, differing only by the symbol used. (Thus, not every structure in the signature of partial orders is a partial order!)
- Structures in the signature of groups are, for example, the following *groups*:
  - $\underline{\mathbb{Z}_n} = \langle \mathbb{Z}_n, +, -, 0\rangle$, the *additive group of integers modulo n* (operations are modulo n).[9]
  - $\mathcal{S}_n = \langle \mathrm{Sym}_n, \circ, ^{-1}, \mathrm{id}\rangle$ is the *symmetric group* (the group of all permutations) on n elements.
  - $\underline{\mathbb{Q}^*} = \langle \mathbb{Q} \setminus \{0\}, \cdot, ^{-1}, 1\rangle$ is the *multiplicative group of (non-zero) rational numbers.* Note that the interpretation of the *symbol* 0 is the *number* 1.

  All these structures *satisfy the axioms of group theory*, but we can easily find other structures that do not satisfy these axioms and are therefore not groups. For example, if we change the interpretation of the symbol $+$ in the structure $\mathbb{Z}_n$ to the function $\cdot$ (modulo n).
- Structures $\underline{\mathbb{Q}} = \langle \mathbb{Q}, +, -, 0, \cdot, 1, \leq\rangle$ and $\underline{\mathbb{Z}} = \langle \mathbb{Z}, +, -, 0, \cdot, 1, \leq\rangle$, with the standard operations and the standard order relation, are in the signature of ordered fields (but only the first one is an ordered field).

[7]It does not matter what specific symbols we use in the signature; we can interpret them arbitrarily. For example, having the symbol $+$ does not mean that its interpretation must have anything to do with addition (other than being a binary function).

[8]As we will see in the definition below, formally, it is the triple $\langle A, \emptyset, \emptyset\rangle$, but we will ignore this distinction.

[9]Here, $\underline{\mathbb{Z}_n}$ denotes the structure, while $\mathbb{Z}_n = \{0, 1, \dots, n-1\}$ denotes only its domain. Often, this distinction is not made, and the symbol $\mathbb{Z}_n$ is used for both the whole structure and its domain. Similarly, $+, -, 0$ are both symbols and their interpretations. This is a common abuse of notation; it is crucial to always be aware of the meaning of the symbol in the given context.

- $\underline{\mathcal{P}(X)} = \langle \mathcal{P}(X), \bar{}, \cap, \cup, \emptyset, X\rangle$, the so-called *power set algebra* over a set X, is a structure in the signature of Boolean algebras. (It is a *Boolean algebra*, as long as $X \neq \emptyset$.)
- $\underline{\mathbb{N}} = \langle \mathbb{N}, S, +, \cdot, 0, \leq\rangle$, where $S(x) = x + 1$, and other symbols are interpreted in the standard way, is the *standard model of arithmetic.*

Definition 1.2.3 (Structure)**.** A *structure in the signature* $\langle \mathcal{R}, \mathcal{F}\rangle$ is a triple $\mathcal{A} = \langle A, \mathcal{R}^{\mathcal{A}}, \mathcal{F}^{\mathcal{A}}\rangle$, where

- A is a non-empty set, called the *domain* (also *universe*),
- $\mathcal{R}^{\mathcal{A}} = \{R^{\mathcal{A}} \mid R \in \mathcal{R}\}$ where $R^{\mathcal{A}} \subseteq A^{\mathrm{ar}(R)}$ is the *interpretation* of the relation symbol R,
- $\mathcal{F}^{\mathcal{A}} = \{f^{\mathcal{A}} \mid f \in \mathcal{F}\}$ where $f^{\mathcal{A}} \colon A^{\mathrm{ar}(f)} \to A$ is the *interpretation* of the function symbol f (in particular, for a constant symbol $c \in \mathcal{F}$, we have $c^{\mathcal{A}} \in A$).

Exercise 1.2*.* Consider the signature of n *constants* $\langle c_1, c_2, \ldots, c_n\rangle$. What do structures in this signature look like? Describe, for example, all structures with at most five elements in the signature of three constants. (The interpretations of constants do not have to be different!) And what about the case of the signature of *countably many constants* $\langle c_1, c_2, \ldots\rangle = \langle c_i \mid i \in \mathbb{N}\rangle$?

1.3 Syntax

In this section, we introduce the syntax of predicate (first-order) logic. Compare what the syntax has in common with, and how it differs from, the syntax of propositional logic.

1.3.1 Language

When specifying a language, we first determine what type of structures we want to describe, i.e., we specify the *signature.* Additionally, we include the information on whether the language is *with equality* or not, i.e., whether we can use the symbol '=' in formulas to express the equality (identity) of elements in the domain of structures.[10] The language includes the following:

- countably many *variables* $x_0, x_1, x_2, \ldots$ (but we also write $x, y, z, \ldots$; the set of all variables is denoted Var),
- *relation*, *function*, and *constant symbols* from the signature, and the symbol $=$ if the language is with equality,
- *universal* and *existential quantifiers* $(\forall x), (\exists x)$ for each variable $x \in \mathrm{Var}$,[11]
- symbols for logical connectives $\neg, \wedge, \vee, \to, \leftrightarrow$ and parentheses $(,)$.

[10]In most applications, we will use languages with equality. However, in some special areas, it is useful to not have equality in the language. For example, if we deal with very fast models of computation: finding out which variables are equal requires finding the transitive closure of equality predicates given by formulas, which is already a relatively computationally demanding problem.

[11]A quantifier is understood as a single symbol, so $(\forall x)$ *does not include* the variable x. Sometimes the symbols $\forall_x, \exists_x$ are used instead.

Similarly to the symbol $\square$ representing any binary logical connective, we will sometimes write (Qx) for the quantifier $(\forall x)$ or $(\exists x)$.

Symbols from the signature, and $=$, are called *non-logical*, and the other symbols are *logical.* The language must contain at least one relation symbol (either equality, or in the signature).[12]

Thus, we specify the language by giving a signature and the information 'with equality' (or 'without equality'). For example:

- The language $L = \langle\rangle$ with equality is the language of *pure equality*,
- the language $L = \langle c_0, c_1, c_2, \dots \rangle$ with equality is the language of *countably many constants*,
- the language of *order* is $\langle \leq \rangle$ with equality,
- the language of *graph theory* is $\langle E \rangle$ with equality,
- the languages of *group theory, field theory, ordered field theory, Boolean algebras, arithmetic* are languages with equality corresponding to the signatures from Example **??**.

1.3.2 Terms

Terms are syntactic 'expressions' composed of variables, constant symbols, and function symbols.

Definition 1.3.1 (Terms)**.** *Terms* of the language L are finite strings defined inductively:

- each variable and each constant symbol from L is a term,
- if f is a function symbol from L of arity n and $t_1, \dots, t_n$ are terms, then the string $f(t_1, t_2, \dots, t_n)$ is also a term.

The set of all *terms* of the language L is denoted Term_L.

When writing terms containing a binary function symbol, we can use *infix* notation, e.g., $(t_1 + t_2)$ means $+(t_1, t_2)$. Parentheses are sometimes omitted if the structure of the term ('operator precedence') is clear.

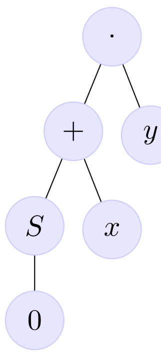
A *subterm* is a substring of a term that is itself a term (it is either the whole term or it appeared as some t_i in the construction of the term).

If a term does not contain a variable, we call it *ground* (also *constant*), for example, $((S(0) + S(0)) \cdot S(S(0)))$ is a ground term in the language of arithmetic.[13]

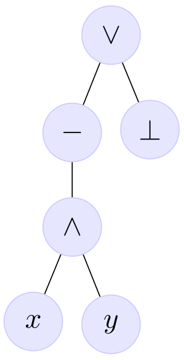
The *tree of the term t*, denoted $\mathrm{Tree}(t)$, is defined similarly to a tree of a proposition: leaves are labeled by variables or constant symbols, inner nodes by function symbols whose arity matches the number of children.

[12]Otherwise, we would not be able to build any 'statements' (*formulas*) in the language, see below.

[13]Note that terms are purely syntactic; we can only use symbols from the language, not elements of the structure, so $(1 + 1) \cdot 2$ is *not* a term in the language of arithmetic! (However, we could *define* new constant symbols $1, 2$ as abbreviations for $S(0)$ and $S(S(0))$ and *extend* our language, see Section **??**.)



(a) $(S(0) + x) \cdot y$ in the language of arithmetic



(b) $-(x \wedge y) \vee \bot$ in the language of Boolean algebras

Figure 1.1: Term tree

Example 1.3.2. Let us draw the trees of the terms (a) $(S(0)+x)\cdot y$ in the language of arithmetic, (b) $-(x \wedge y) \vee \bot$ in the language of Boolean algebras. Here, $\wedge, \vee$ are not logical connectives from the language but non-logical symbols from the signature of Boolean algebras (although we use the same symbols)! Terms in this language can be understood as propositional formulas (with constants for falsity and truth), see Section **??**. Figure **??** shows the trees of these terms.

It is not hard to guess what the *semantics* of terms will be. Given a specific structure, a term corresponds to a function on its domain: the input is an assignment of the variables to elements of the domain, the constant and function symbols are replaced by their interpretations, and the output is the value (element of the domain) at the root. More formally, this will be covered in Section **??**.

1.3.3 Formulas

Terms cannot be assigned a truth value in any sense; for that, we need a *predicate* (a relation symbol or equality) that talks about the 'relationship' between terms: in a specific structure with a specific variable assignment to elements of the domain, this relationship is either satisfied or not.

The simplest *formulas* are *atomic formulas.* We then build all formulas from them using logical connectives and quantifiers.

Definition 1.3.3 (Atomic Formula)**.** An *atomic formula* of the language L is a string $R(t_1, \dots, t_n)$, where R is an n-ary relation symbol from L (including $=$ if it is a language with equality) and $t_i \in \mathrm{Term}_L$.

For binary relation symbols, we often use infix notation, e.g., the atomic formula $\leq (x, y)$ is written as $x \leq y$, and (if the language has equality) instead of $= (t_1, t_2)$, we will write $t_1 = t_2$.

Example 1.3.4. Here are some examples of atomic formulas:

- $R(f(f(x)), c, f(d))$ where R is a ternary relation symbol, f a unary function symbol, c, d are constant symbols,
- $(x \cdot x) + (y \cdot y) \leq (x + y) \cdot (x + y)$ in the language of ordered fields,

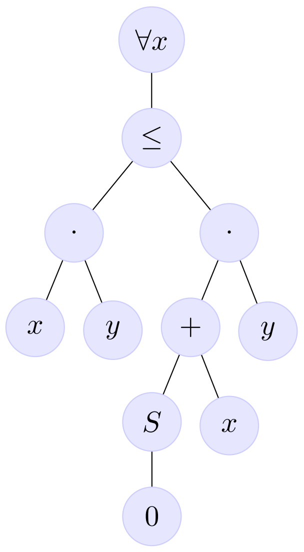


Figure 1.2: Tree of the formula $(\forall x)(x \cdot y \leq (S(0) + x) \cdot y)$

- $x \cdot y \leq (S(0) + x) \cdot y$ in the language of arithmetic,
- $-(x \wedge y) \vee \bot = \bot$ in the language of Boolean algebras

Definition 1.3.5 (Formula)**.** *Formulas* of the language L are finite strings defined inductively:

- each atomic formula of the language L is a formula,
- if φ is a formula, then $(\neg\varphi)$ is also a formula,
- if φ, ψ are formulas, then $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \rightarrow \psi)$, and $(\varphi \leftrightarrow \psi)$ are also formulas,
- if φ is a formula and x a variable, then $((\forall x)\varphi)$ and $((\exists x)\varphi)$ are also formulas.

A *subformula* is a substring that is itself a formula. The *tree of a formula*, denoted $\text{Tree}(\varphi)$, is defined as follows: the tree of an atomic formula $\varphi = R(t_1, \ldots, t_n)$ has the relation symbol R at the root, to which we append the trees $\text{Tree}(t_i)$. If φ is not atomic, the tree is constructed similarly to a propositional formula tree.[14] When writing formulas, we use similar conventions as in propositional logic, with quantifiers having the same precedence as $\neg$ (higher than other logical connectives). Therefore, instead of $((\forall x)\varphi)$ we can write $(\forall x)\varphi$.[15]

Example 1.3.6. An example of a formula in the language of arithmetic is $(\forall x)(x \cdot y \leq (S(0) + x) \cdot y)$. Its tree is shown in Figure **??**.

Free and Bound Variables

The meaning of a formula[16] may or may not depend on the variables that appear in it: compare $x \leq 0$ and $(\exists x)(x \leq 0)$ (and how about $x \leq 0 \vee (\exists x)(x \leq 0)$?). We now clarify this concept and introduce the necessary terminology.

[14] Quantifiers, like negations, have a single child.

[15] Sometimes parentheses are not written in quantifiers, e.g., $\forall x\varphi$, but we will always include them, for better readability.

[16] More precisely, its *truth value*, formally defined below in Section **??**.

An *occurrence* of a variable x in a formula φ means a leaf in Tree(φ) labeled x. [17] An occurrence is *bound* if it is a part of some subformula (subtree) starting with (Qx). If an occurrence is not bound, it is *free*. A variable is *free* in φ if it has a free occurrence in φ, and *bound* in φ if it has a bound occurrence in φ. The notation $\varphi(x_1, \ldots, x_n)$ means that $x_1, \ldots, x_n$ are all the free variables in the formula φ.

Example 1.3.7. A variable can be both free and bound, e.g., in the formula $\varphi = (\forall x)(\exists y)(x \leq y) \vee x \leq z$, the first occurrence of x is bound and the second occurrence is free. (Draw the formula tree!) The variable y is bound (its only occurrence is bound) and z is free. Thus, we can write $\varphi(x, z)$.

Remark 1.3.8. As we will see below, the meaning (*truth value*) of a formula depends only on the assignment of free variables. Variables in quantifiers, along with their corresponding bound occurrences, can be renamed (we have to be careful though, see below).

Open and Closed Formulas

We often talk about the following two important properties of formulas:

Definition 1.3.9 (Open and Closed Formula)**.** A formula is *open* if it contains no quantifier, and *closed* (or a *sentence*) if it has no free variable.

Example 1.3.10. Here are a few examples:

- the formula $x + y \leq 0$ is open,
- the formula $(\forall x)(\forall y)(x + y \leq 0)$ is closed (i.e., it is a sentence),
- the formula $(\forall x)(x + y \leq 0)$ is neither open nor closed,
- the formula $(0 + 1 = 1) \wedge (1 + 1 = 0)$ is both open and closed.

Every atomic formula is open, and open formulas are just combinations of atomic formulas using logical connectives. A formula can be both open and closed if all its terms are constant. A formula is closed if and only if it has no free variable.[18]

Remark 1.3.11. As we will see later, the *truth value* of a formula depends only on the assignment of its free variables. In particular, a sentence has a truth value of 0 or 1 in a given structure (independently of variable assignments). This is why sentences play an important role in logic.

1.3.4 Instances and Variants

As we have seen, one variable can appear in different 'roles' in a formula. This is a very similar principle to programming, where one identifier can mean different variables in a program (either local or global). The term *instance* can be understood as 'substituting' (a term) into a (global) variable (or better, 'replacing' a variable with some expression that computes it), and the term *variant* as 'renaming' a (local) variable. Consider, for example, the formula $\varphi(x)$:

$$P(x) \wedge (\forall x)(Q(x) \wedge (\exists x)R(x))$$

[17] Thus, the variable x does *not occur* in the symbol for the quantifier (Qx).

[18] It is not true that a formula is open if it has no bound variable, see the formula $(\forall x)0 = 1$.

The first occurrence of the variable x is free, the second is bound by the quantifier $(\forall x)$, and the third is bound by $(\exists x)$. If we 'substitute' the term $t = 1 + 1$ for the variable x, we get an *instance* of the formula φ, denoted $\varphi(x/t)$:

$$P(1+1) \land (\forall x)(Q(x) \land (\exists x)R(x))$$

We can also rename the quantifiers in the formula, thus obtaining a *variant* of the formula φ, e.g.:

$$P(x) \land (\forall y)(Q(y) \land (\exists z)R(z))$$

How do we know when and how we can do this to preserve the meaning, i.e., so that the instance is a *consequence* of φ, and the variant is *equivalent* to φ? This is what we now want to formalize.

Instances

If we *substitute* a term t for a free variable x in a formula φ, we require that the resulting formula 'says' about t 'the same' as φ says about x.

Example 1.3.12. For example, the formula $\varphi(x) = (\exists y)(x + y = 1)$ says about x that 'there exists $1 - x$'. The term $t = 1$ can be substituted because $\varphi(x/t) = (\exists y)(1 + y = 1)$ says 'there exists 1-1'. But the term $t = y$ cannot be substituted because $(\exists y)(y + y = 1)$ says '1 is divisible by 2'. The problem is that the term $t = y$ contains the variable y, which will now be bound by the quantifier $(\exists y)$. We must avoid such situations.

Definition 1.3.13 (Substitutability and Instance)**.** A term t is *substitutable* for a variable x in a formula φ if, after simultaneously replacing all free occurrences of x in φ with t, no new bound occurrence of a variable from t arises in φ. In that case, the resulting formula is called an *instance* of φ obtained by substituting t for x, denoted $\varphi(x/t)$.

Remark 1.3.14. Note that a term t is *not* substitutable for x in φ if and only if x has a free occurrence in some subformula φ of the form $(Qy)\psi$ and the variable y appears in t. In particular, ground terms are always substitutable.

Variants

If we need to substitute a term t into a formula φ, we can always do so if we first rename all quantified variables to entirely new ones (i.e., ones that do not appear in φ or t), and then substitute t into the resulting *variant* of the formula φ.

Definition 1.3.15 (Variant)**.** If a formula φ has a subformula of the form $(Qx)\psi$ and y is a variable such that

- y is substitutable for x in ψ, and
- y has no free occurrence in ψ,

then replacing the subformula $(Qx)\psi$ with $(Qy)\psi(x/y)$ results in a *variant* of the formula φ in the subformula $(Qx)\psi$. The result of successive variations in multiple subformulas is also called a *variant.*

Note that the requirement for the variable y in the definition of a variant is always satisfied if y does not appear in the formula φ.

Example 1.3.16. Consider the formula $\varphi = (\exists x)(\forall y)(x \leq y)$. Then:

- $(\exists y)(\forall y)(y \leq y)$ is not a variant of φ because y is not substitutable for x in $\psi = (\forall y)(x \leq y)$,
- $(\exists x)(\forall x)(x \leq x)$ is not a variant of φ because x has a free occurrence in the subformula $\psi = (x \leq y)$,
- $(\exists u)(\forall v)(u \leq v)$ is a variant of φ.

This concludes the exposition on syntax; next is semantics.

1.4 Semantics

Before we delve into a more formal exposition, let us briefly summarize the semantics as we have already hinted in previous sections:

- Models are structures of the given signature,
- A formula is valid in a structure if it holds under every assignment of free variables to elements from the domain,
- The values of terms are evaluated according to their trees, where symbols are replaced by their interpretations (relations, functions, and constants from the domain),
- From the values of terms, we obtain the truth values of atomic formulas: is the resulting n-tuple in the relation?
- The values of complex formulas are also evaluated according to their tree, where $(\forall x)$ acts as 'conjunction over all elements' and $(\exists y)$ acts as 'disjunction over all elements' of the structure's domain.

Now more formally:

1.4.1 Models of the Language

Definition 1.4.1 (Model of the Language)**.** A *model of the language* L, or an L*-structure*, is any structure in the signature of the language L. The *class of all models* of the language is denoted M_L.

Remark 1.4.2. The definition does not care whether the language is with or without equality. And why can't we talk about the *set* of all models M_L, why do we have to say *class*? Because the domain of a structure can be any non-empty set, and the 'set of all sets' does not exist; it is a classic example of a so-called proper class. A class is the *'collection'* of all sets satisfying a given property (describable in the *language of set theory*).

Example 1.4.3. Among the models of the language of order $L = \langle \leq \rangle$ are the following structures: $\langle \mathbb{N}, \leq \rangle$, $\langle \mathbb{Q}, > \rangle$, any directed graph $G = \langle V, E \rangle$, $\langle \mathcal{P}(X), \subseteq \rangle$. But also, for instance, $\langle \mathbb{C}, R^{\mathbb{C}} \rangle$ where $(z_1, z_2) \in R^{\mathbb{C}}$ if and only if $|z_1| = |z_2|$ or $\langle \{0, 1\}, \emptyset \rangle$, which are *not* partial orders.

1.4.2 Value of a term

Consider a term t in the language $L = \langle \mathcal{R}, \mathcal{F} \rangle$ (with or without equality), and an L-structure $\mathcal{A} = \langle A, \mathcal{R}^{\mathcal{A}}, \mathcal{F}^{\mathcal{A}} \rangle$. A *variable assignment* in the set A is any function $e : \mathrm{Var} \to A$.

Definition 1.4.4 (Value of a term)**.** The *value of the term t in the structure $\mathcal{A}$ under the assignment e*, denoted $t^{\mathcal{A}}[e]$, is defined inductively:

- $x^{\mathcal{A}}[e] = e(x)$ for a variable $x \in \mathrm{Var}$,
- $c^{\mathcal{A}}[e] = c^{\mathcal{A}}$ for a constant symbol $c \in \mathcal{F}$, and
- if $t = f(t_1, \ldots, t_n)$ is a compound term where $f \in \mathcal{F}$, then:

$$t^{\mathcal{A}}[e] = f^{\mathcal{A}}(t_1^{\mathcal{A}}[e], \ldots, t_n^{\mathcal{A}}[e])$$

Remark 1.4.5. Note that the value of a term depends only on the assignment of the variables appearing in it. In particular, if t is a ground term, its value does not depend on the assignment. In general, each term t represents a *term function* $f_t^{\mathcal{A}} \colon A^k \to A$, where k is the number of variables in t, and ground terms correspond to constant functions.

Example 1.4.6. Here are two examples:

- The value of the term $-(x \vee \bot) \wedge y$ in the Boolean algebra $\underline{\mathcal{P}(\{0, 1, 2\})}$ under the assignment e where $e(x) = \{0, 1\}$ and $e(y) = \{1, 2\}$ is $\{2\}$.
- The value of the term $x + 1$ in the structure $\mathcal{N} = \langle \mathbb{N}, \cdot, 3 \rangle$ of the language $L = \langle +, 1 \rangle$ under the assignment e where $e(x) = 2$ is $(x + 1)^{\mathcal{N}}[e] = 6$.

1.4.3 Truth Value of a Formula

We are now ready to define the *truth value*. Locally, we introduce the notation TVal for it.

Definition 1.4.7 (Truth Value)**.** Given a formula φ in the language L, a structure $\mathcal{A} \in \mathrm{M}_L$, and a variable assignment $e : \mathrm{Var} \to A$. The *truth value of φ in $\mathcal{A}$ under the assignment e,* $\mathrm{TVal}^{\mathcal{A}}(\varphi)[e]$, is defined inductively according to the structure of the formula:

For an atomic formula $\varphi = R(t_1, \ldots, t_n)$, we have

$$\mathrm{TVal}^{\mathcal{A}}(\varphi)[e] = \begin{cases} 1 & \text{if } (t_1^{\mathcal{A}}[e], \ldots, t_n^{\mathcal{A}}[e]) \in R^{\mathcal{A}}, \\ 0 & \text{otherwise.} \end{cases}$$

In particular, if φ is of the form $t_1 = t_2$, then $\mathrm{TVal}^{\mathcal{A}}(\varphi)[e] = 1$ if and only if $(t_1^{\mathcal{A}}[e], t_2^{\mathcal{A}}[e]) \in =^{\mathcal{A}}$, where $=^{\mathcal{A}}$ is the identity on A, i.e., if and only if $t_1^{\mathcal{A}}[e] = t_2^{\mathcal{A}}[e]$ (both sides of the equality are the same element $a \in A$).

The truth value of a negation is defined as follows:

$$\mathrm{TVal}^{\mathcal{A}}(\neg\varphi)[e] = f_{\neg}(\mathrm{TVal}^{\mathcal{A}}(\varphi)[e]) = 1 - \mathrm{TVal}^{\mathcal{A}}(\varphi)[e]$$

Similarly, for binary logical connectives, if φ, ψ are formulas and $\square \in \{\wedge, \vee, \to, \leftrightarrow\}$, then:

$$\mathrm{TVal}^{\mathcal{A}}(\varphi \mathbin{\square} \psi)[e] = f_{\square}(\mathrm{TVal}^{\mathcal{A}}(\varphi)[e], \mathrm{TVal}^{\mathcal{A}}(\psi)[e])$$

It remains to define the truth value for quantifiers, i.e., formulas of the form $(Qx)\varphi$. We will need the following notation: If in the assignment $e : \mathrm{Var} \to A$ we change the value for the variable x to a, the resulting assignment is written as $e(x/a)$. Thus, $e(x/a)(x) = a$. The truth value for $(Qx)\varphi$ is defined as follows:

$$\mathrm{TVal}^{\mathcal{A}}((\forall x)\varphi)[e] = \min_{a\in A}(\mathrm{TVal}^{\mathcal{A}}(\varphi)[e(x/a)])$$
$$\mathrm{TVal}^{\mathcal{A}}((\exists x)\varphi)[e] = \max_{a\in A}(\mathrm{TVal}^{\mathcal{A}}(\varphi)[e(x/a)])$$

Thus, under the assignment e, we set the value of the variable x successively to all elements $a \in A$ and require that the truth value is 1 always (in the case of $\forall$) or at least once (in the case of $\exists$).[19]

Remark 1.4.8. The truth value depends only on the assignment of free variables. In particular, if φ is a sentence, then its truth value does not depend on the assignment.

Example 1.4.9. Consider the ordered field $\underline{\mathbb{Q}}$. Then:

- $\mathrm{TVal}^{\underline{\mathbb{Q}}}(x \le 1 \land \neg(x \le 0))[e] = 1$ if and only if $e(x) \in (0, 1]$,
- $\mathrm{TVal}^{\underline{\mathbb{Q}}}((\forall x)(x \cdot y = y))[e] = 1$ if and only if $e(y) = 0$,
- $\mathrm{TVal}^{\underline{\mathbb{Q}}}((\exists x)(x \le 0 \land \neg x = 0))[e] = 1$ for any assignment e (it is a sentence), but
- $\mathrm{TVal}^{\mathcal{A}}((\exists x)(x \le 0 \land \neg x = 0))[e] = 0$ (for any e), if $\mathcal{A} = \langle \mathbb{N}, +, -, 0, \cdot, 1, \le\rangle$ with the standard operations and order.

1.4.4 Validity

Based on the truth value, we can now define the key notion of semantics, *validity.*

Definition 1.4.10 (Validity in a Structure)**.** Given a formula φ and a structure $\mathcal{A}$ (in the same language).

- If e is an assignment and $\mathrm{TVal}^{\mathcal{A}}(\varphi)[e] = 1$, we say that φ *is valid in* $\mathcal{A}$ *under the assignment* e, and write $\mathcal{A} \models \varphi[e]$. (Otherwise, we say that φ *is not valid in* $\mathcal{A}$ *under the assignment* e, and write $\mathcal{A} \not\models \varphi[e]$.)
- If φ is valid in $\mathcal{A}$ under every assignment $e : \mathrm{Var} \to A$, we say that φ *is valid (true) in* $\mathcal{A}$, and write $\mathcal{A} \models \varphi$.
- If $\mathcal{A} \models \neg\varphi$, i.e., φ is not valid in $\mathcal{A}$ under any assignment (for every e we have $\mathcal{A} \not\models \varphi[e]$), then φ *is contradictory in* $\mathcal{A}$.[20]

Let us summarize some simple properties, first concerning validity under an assignment. Let $\mathcal{A}$ be a structure, φ, ψ formulas, and e a variable assignment.

- $\mathcal{A} \models \neg\varphi[e]$ if and only if $\mathcal{A} \not\models \varphi[e]$,
- $\mathcal{A} \models (\varphi \land \psi)[e]$ if and only if $\mathcal{A} \models \varphi[e]$ and $\mathcal{A} \models \psi[e]$,

[19]Recall that $f_\land(x, y) = \min(x, y)$ and $f_\lor(x, y) = \max(x, y)$. Thus, quantifiers play the role of 'conjunction' ($\forall$) or 'disjunction' ($\exists$) over all elements of the structure.

[20]Note that *contradictory* is not the same as *not valid*! This only holds for sentences.

- $\mathcal{A} \models (\varphi \vee \psi)[e]$ if and only if $\mathcal{A} \models \varphi[e]$ or $\mathcal{A} \models \psi[e]$,
- $\mathcal{A} \models (\varphi \to \psi)[e]$ if and only if: if $\mathcal{A} \models \varphi[e]$ then $\mathcal{A} \models \psi[e]$,
- $\mathcal{A} \models (\varphi \leftrightarrow \psi)[e]$ if and only if: $\mathcal{A} \models \varphi[e]$ if and only if $\mathcal{A} \models \psi[e]$,
- $\mathcal{A} \models (\forall x)\varphi[e]$ if and only if $\mathcal{A} \models \varphi[e(x/a)]$ for all $a \in A$,
- $\mathcal{A} \models (\exists x)\varphi[e]$ if and only if $\mathcal{A} \models \varphi[e(x/a)]$ for some $a \in A$.
- If a term t is substitutable for the variable x in the formula φ, then
$$\mathcal{A} \models \varphi(x/t)[e] \text{ if and only if } \mathcal{A} \models \varphi[e(x/a)] \text{ for } a = t^{\mathcal{A}}[e].$$
- If ψ is a variant of φ, then $\mathcal{A} \models \varphi[e]$ if and only if $\mathcal{A} \models \psi[e]$.

Exercise 1.3. Prove all the listed properties of validity under an assignment in detail.

And what about the notion of truth (validity) in a structure?

- If $\mathcal{A} \models \varphi$, then $\mathcal{A} \not\models \neg\varphi$. If φ is a sentence, then the converse implication also holds (i.e., it is 'if and only if').
- $\mathcal{A} \models \varphi \wedge \psi$ if and only if $\mathcal{A} \models \varphi$ and $\mathcal{A} \models \psi$,
- If $\mathcal{A} \models \varphi$ or $\mathcal{A} \models \psi$, then $\mathcal{A} \models \varphi \vee \psi$. If φ is a sentence, then the converse implication also holds (i.e., it is 'if and only if').
- $\mathcal{A} \models \varphi$ if and only if $\mathcal{A} \models (\forall x)\varphi$.

The *general closure* of a formula $\varphi(x_1, \ldots, x_n)$ (i.e., $x_1, \ldots, x_n$ are all the free variables of the formula φ) is the sentence $(\forall x_1) \cdots (\forall x_n)\varphi$. From the last point, it follows that a formula is valid in a structure if and only if its general closure is valid in it.

Exercise 1.4. Prove all the listed properties of validity in a structure in detail.

Exercise 1.5. Give an example of a structure $\mathcal{A}$ and a formula φ such that $\mathcal{A} \not\models \varphi$ and yet $\mathcal{A} \not\models \neg\varphi$.

Exercise 1.6. Give an example of a structure $\mathcal{A}$ and formulas φ, ψ such that $\mathcal{A} \models \varphi \vee \psi$ but $\mathcal{A} \not\models \varphi$ and $\mathcal{A} \not\models \psi$.

1.5 Properties of Theories

Based on the notion of *validity*, we will build semantic terminology similar to that in propositional logic. A *theory* of a language L is any set T of L-formulas, whose elements are called *axioms*. A *model* of the theory T is an L-structure in which all axioms of the theory T are valid, i.e., $\mathcal{A} \models \varphi$ for all $\varphi \in T$, which we denote by $\mathcal{A} \models T$. The *class of models*[21] of the theory T is:
$$\mathrm{M}_L(T) = \{\mathcal{A} \in \mathrm{M}_L \mid \mathcal{A} \models T\}$$
As in propositional logic, we will often omit the language L when it is clear from the context and write $M(\varphi_1, \ldots, \varphi_n)$ instead of $M(\{\varphi_1, \ldots, \varphi_n\})$ and $M(T, \varphi)$ instead of $M(T \cup \{\varphi\})$.

[21] Recall that we cannot say 'set'.

1.5.1 Validity in a Theory

If T is a theory in the language L and φ is an L-formula, then we say that φ is:

- *valid (true) in* T, denoted $T \models \varphi$, if $\mathcal{A} \models \varphi$ for all $\mathcal{A} \in \mathrm{M}(T)$ (in other words: $\mathrm{M}(T) \subseteq \mathrm{M}(\varphi)$),
- *contradictory in* T, if $T \models \neg\varphi$, i.e., if it is contradictory in every model of T (in other words: $\mathrm{M}(T) \cap \mathrm{M}(\varphi) = \emptyset$),
- *independent in* T, if it is neither valid in T nor contradictory in T.

If we have an empty theory $T = \emptyset$ (i.e., $\mathrm{M}(T) = \mathrm{M}_L$), then we omit the theory T, write $\models \varphi$, and say that φ *is (universally) valid, (logically) valid, is a tautology*; similarly for other notions.

A theory is *inconsistent* if the *contradiction* $\perp$ is valid in it; $\perp$ in predicate logic can be defined as $R(x_1, \ldots, x_n) \wedge \neg R(x_1, \ldots, x_n)$, where R is any (say, the first) relation symbol from the language or the equality symbol (recall: if the language has no relation symbol, it must be with equality). A theory is *inconsistent* if and only if every formula is valid in it, or equivalently, if and only if it has no model. Otherwise, we say that the theory is *consistent* (if the contradiction is not valid in it, equivalently, if it has at least one model).

Sentences valid in T are called *consequences* of T; the *set of all consequences* of T in the language L is:

$$\mathrm{Csq}_L(T) = \{\varphi \mid \varphi \text{ is a sentence and } T \models \varphi\}$$

Completeness in Predicate Logic

How about the notion of *completeness* of a theory?[22]

Definition 1.5.1. A theory T is *complete* if it is consistent and every *sentence* is either valid in T or contradictory in T.

However, we cannot say that a theory is complete if and only if it has a single model. If we have one model, we derive infinitely many different, but *isomorphic* models, i.e., differing only by the naming of the elements of the universe.[23] Even considering a single model 'up to isomorphism' would not be sufficient. The correct notion is called *elementary equivalence*:

Definition 1.5.2. Structures $\mathcal{A}, \mathcal{B}$ (in the same language) are *elementarily equivalent* if the same sentences are valid in both of them. We denote this by $\mathcal{A} \equiv \mathcal{B}$.

Example 1.5.3. An example of structures that are elementarily equivalent but not isomorphic are the ordered sets $\mathcal{A} = \langle \mathbb{Q}, \leq \rangle$ and $\mathcal{B} = \langle \mathbb{R}, \leq \rangle$. They are not isomorphic because $\mathbb{Q}$ is countable while $\mathbb{R}$ is uncountable, so there is not even a *bijection* between their universes. It is not difficult to show that for any sentence φ, $\mathcal{A} \models \varphi \Leftrightarrow \mathcal{B} \models \varphi$ holds: by induction on the structure of the formula φ, the only non-trivial case is the existential quantifier, and the key property is the *density* of both orders, i.e., the following property:

$$(x \leq y \wedge \neg x = y) \rightarrow (\exists z)(x \leq z \wedge z \leq y \wedge \neg x = z \wedge \neg y = z)$$

[22] Recall that a *propositional* theory is complete if it is consistent and every proposition is either valid in it or its negation is valid in it. Equivalently, it has exactly one model.

[23] Formally, the notion of *isomorphism* is defined later in the part on *model theory*, in Section **??**, but it is a generalization of the isomorphism you know from graph theory.

Observation 1.5.4. *A theory is complete if and only if it has exactly one model up to elementary equivalence.*

Validity by Unsatisfiability

The question of truth (validity) in a given theory can be reduced to the problem of the existence of a model:

Proposition 1.5.5 (On Unsatisfiability and Validity)**.** *If T is a theory and φ is a* sentence *(in the same language), then: $T \cup \{\neg\varphi\}$ has no model if and only if $T \models \varphi$.*

Proof. The following equivalences hold: $T \cup \{\neg\varphi\}$ has no model if and only if $\neg\varphi$ is not valid in any model of T, if and only if (since it is a sentence) φ is valid in every model of T. □

The assumption that φ is a sentence is necessary: consider the theory $T = \{P(c)\}$ and the formula $\varphi = P(x)$ (which is not a sentence). Then $\{P(c), \neg P(x)\}$ has no model, but $P(c) \not\models P(x)$. (Here, P is a unary relation symbol and c is a constant symbol.)

1.5.2 Examples of Theories

Here are some examples of important theories.

Theory of Graphs

The *theory of graphs* is a theory in the language $L = \langle E \rangle$ with equality, satisfying the axioms of *irreflexivity* and *symmetry*:

$$T_{\text{graph}} = \{\neg E(x,x), E(x,y) \to E(y,x)\}$$

The models of T_{graph} are structures $\mathcal{G} = \langle G, E^{\mathcal{G}} \rangle$, where $E^{\mathcal{G}}$ is a symmetric irreflexive relation, i.e., so-called *simple* graphs, where the edge $\{x, y\}$ is represented by the ordered pairs $(x, y), (y, x)$.

- The formula $\neg x = y \to E(x, y)$ holds in a graph if and only if the graph is a *clique* (a *complete* graph). The formula is thus independent in T_{graph}.
- The formula $(\exists y_1)(\exists y_2)(\neg y_1 = y_2 \wedge E(x, y_1) \wedge E(x, y_2) \wedge (\forall z)(E(x, z) \to z = y_1 \vee z = y_2))$ expresses that each vertex has degree exactly 2. It thus holds precisely in graphs that are disjoint unions of cycles and is independent in the theory T_{graph}.

Theory of Order

The *theory of order* is a theory in the language of order $L = \langle \leq \rangle$ with equality, whose axioms are:

$$\begin{aligned} T = \{&x \leq x, \\ &x \leq y \wedge y \leq x \to x = y, \\ &x \leq y \wedge y \leq z \to x \leq z\} \end{aligned}$$

These axioms are called *reflexivity*, *antisymmetry*, and *transitivity.* The models of T are L-structures $\langle S, \leq^S \rangle$, in which the axioms T are valid, so-called *(partially) ordered sets.* For example: $\mathcal{A} = \langle \mathbb{N}, \leq \rangle$, $\mathcal{B} = \langle \mathcal{P}(X), \subseteq \rangle$ for $X = \{0, 1, 2\}$.

- The formula $x \leq y \vee y \leq x$ (*linearity*) is valid in $\mathcal{A}$ but not in $\mathcal{B}$, because it is not valid, for example, under the assignment where $e(x) = \{0\}$, $e(y) = \{1\}$ (we write $\mathcal{B} \not\models \varphi[e]$). It is thus independent in T.
- The sentence $(\exists x)(\forall y)(y \leq x)$ (denote it by ψ) is valid in $\mathcal{B}$ and contradictory in $\mathcal{A}$, we write $\mathcal{B} \models \psi$, $\mathcal{A} \models \neg\psi$. It is thus also independent in T.
- The formula $(x \leq y \wedge y \leq z \wedge z \leq x) \to (x = y \wedge y = z)$ (denote it by χ) is valid in T, we write $T \models \chi$. The same applies to its *general closure* $(\forall x)(\forall y)(\forall z)\chi$.

Algebraic Theories

- The *theory of groups* is a theory in the language $L = \langle +, -, 0 \rangle$ with equality, whose axioms are:

$$\begin{aligned} T_1 = \{&x + (y + z) = (x + y) + z, \\ &0 + x = x,\ x + 0 = 0, \\ &x + (-x) = 0,\ (-x) + x = 0\} \end{aligned}$$

  These properties are called *associativity of* $+$, *neutrality of* 0 *with respect to* $+$, and $-x$ *is the inverse element of* x *(with respect to* $+$ *and* 0*).*

- The *theory of commutative groups* additionally contains the axiom $x + y = y + x$ (*commutativity of* $+$), thus:

$$T_2 = T_1 \cup \{x + y = y + x\}$$

- The *theory of rings* is in the language $L = \langle +, -, 0, \cdot, 1 \rangle$ with equality and adds the following axioms:

$$\begin{aligned} T_3 = T_2 \cup \{&1 \cdot x = x \cdot 1, \\ &x \cdot (y \cdot z) = (x \cdot y) \cdot z, \\ &x \cdot (y + z) = x \cdot y + x \cdot z, \\ &(x + y) \cdot z = x \cdot z + y \cdot z\} \end{aligned}$$

  These properties are called *neutrality of* 1 *with respect to* $\cdot$, *associativity of* $\cdot$, and *(left and right) distributivity of* $\cdot$ *with respect to* $+$.

- The *theory of commutative rings* additionally has the axiom of *commutativity of* $\cdot$, thus:

$$T_4 = T_3 \cup \{x \cdot y = y \cdot x\}$$

- The *theory of fields* is in the same language but additionally has the axioms of *existence of an inverse element with respect to* $\cdot$ and *non-triviality*:

$$T_5 = T_4 \cup \{\neg\, x = 0 \to (\exists y)(x \cdot y = 1), \neg\, 0 = 1\}$$

- The *theory of ordered fields* is in the language $\langle +, -, 0, \cdot, 1, \leq \rangle$ with equality, consisting of the axioms of the theory of fields, the theory of order along with the axiom of linearity, and the following axioms of *compatibility of the order*: $x \leq y \rightarrow (x + z \leq y + z)$ and $(0 \leq x \wedge 0 \leq y) \rightarrow 0 \leq x \cdot y$. (The models are thus fields with *linear (total)* order compatible with the field operations in this sense.)

1.6 Substructure, Expansion, Reduct

In this section, we will look at ways to create new structures from existing ones.

Substructure

The notion of *substructure* generalizes subgroups, subspaces of a vector space, and induced subgraphs of a graph: we select a subset B of the universe of the structure $\mathcal{A}$ and create on it a structure $\mathcal{B}$ of the same signature, which 'inherits' the relations, functions, and constants. To do this, we need the set B to be *closed* under all functions and contain all constants.[24]

Definition 1.6.1 (Substructure). Let $\mathcal{A} = \langle A, \mathcal{R}^{\mathcal{A}}, \mathcal{F}^{\mathcal{A}} \rangle$ be a structure in the signature $\langle \mathcal{R}, \mathcal{F} \rangle$. A structure $\mathcal{B} = \langle B, \mathcal{R}^{\mathcal{B}}, \mathcal{F}^{\mathcal{B}} \rangle$ is an *(induced) substructure of* $\mathcal{A}$, denoted $\mathcal{B} \subseteq \mathcal{A}$, if

- $\emptyset \neq B \subseteq A$,
- $R^{\mathcal{B}} = R^{\mathcal{A}} \cap B^{\mathrm{ar(R)}}$ for each relation symbol $R \in \mathcal{R}$,
- $f^{\mathcal{B}} = f^{\mathcal{A}} \cap (B^{\mathrm{ar(f)}} \times B)$ for each function symbol $f \in \mathcal{F}$ (i.e., the function $f^{\mathcal{B}}$ is the restriction of $f^{\mathcal{A}}$ to the set B, and also its outputs are all from B),
- in particular, for each constant symbol $c \in \mathcal{F}$ we have $c^{\mathcal{B}} = c^{\mathcal{A}} \in B$.

A set $C \subseteq A$ is *closed* under a function $f : A^n \rightarrow A$ if $f(x_1, \ldots, x_n) \in C$ for all $x_i \in C$. We have:

Observation 1.6.2. *A set $\emptyset \neq C \subseteq A$ is the universe of a substructure of the structure $\mathcal{A}$ if and only if C is closed under all functions of the structure $\mathcal{A}$ (including constants).*

In that case, we call this substructure the *restriction* of $\mathcal{A}$ to the set C, and denote it by $\mathcal{A} \restriction C$.

Example 1.6.3. $\underline{\mathbb{Z}} = \langle \mathbb{Z}, +, \cdot, 0 \rangle$ is a substructure of $\underline{\mathbb{Q}} = \langle \mathbb{Q}, +, \cdot, 0 \rangle$, we can write $\underline{\mathbb{Z}} = \underline{\mathbb{Q}} \restriction \mathbb{Z}$. The structure $\underline{\mathbb{N}} = \langle \mathbb{N}, +, \cdot, 0 \rangle$ is a substructure of both these structures, $\underline{\mathbb{N}} = \underline{\mathbb{Q}} \restriction \mathbb{N} = \underline{\mathbb{Z}} \restriction \mathbb{N}$.

Validity in Substructure

How is it with validity of formulas in a substructure? Here are some simple observations about *open* formulas.

Observation 1.6.4. *If $\mathcal{B} \subseteq \mathcal{A}$, then for any* open *formula φ and variable assignment $e\colon \mathrm{Var} \rightarrow B$, we have that: $\mathcal{B} \models \varphi[e]$ if and only if $\mathcal{A} \models \varphi[e]$.*

[24] Just as not every set of vectors is a subspace, it must contain the zero vector, contain all scalar multiples of each vector, and for any pair of vectors, contain their sum. In other words, only (non-empty) sets closed under *linear combinations* of vectors form subspaces.

Proof. For atomic formulas, this is obvious; further, it can be easily proved by induction on the structure of the formula. $\square$

Corollary 1.6.5. *An* open *formula is valid in the structure $\mathcal{A}$ if and only if it is valid in every substructure $\mathcal{B} \subseteq \mathcal{A}$.*

We say that a theory T is *open* if all its axioms are open formulas.

Corollary 1.6.6. *The models of an open theory are closed under substructures, i.e., every substructure of a model of an open theory is also a model of this theory.*

Example 1.6.7. The theory of graphs is open. Every substructure of a graph (a model of the theory of graphs) is also a graph, called an (induced) *subgraph.*[25] Similarly, for subgroups or Boolean subalgebras.

Example 1.6.8. The theory of fields is not open. As we will show later, it is not even *openly axiomatizable*, i.e., there is no equivalent open theory—there is no way to get rid of the quantifier in the axiom of the existence of an inverse element. The substructure of the field of real numbers $\mathbb{Q}$ on the set of all integers $\mathbb{Q} \restriction \mathbb{Z}$ is not a field. (It is a so-called *ring*, but non-zero elements other than $1, -1$ do not have a multiplicative inverse, e.g., the equation $2 \cdot x = 1$ has no solution in $\mathbb{Z}$).

Generated Substructure

What to do if we have a subset of the universe that is *not* closed under the functions of the structure? In that case, we consider the *closure* of this set under the functions.[26]

Definition 1.6.9. Let $\mathcal{A} = \langle A, \mathcal{R}^{\mathcal{A}}, \mathcal{F}^{\mathcal{A}} \rangle$ be a structure and a non-empty subset $X \subseteq A$. Let B be the smallest subset of A that contains the set X and is closed under all functions of the structure $\mathcal{A}$ (i.e., it also contains all constants). Then the substructure $\mathcal{A} \restriction B$ is said to be *generated* by the set X, and denoted by $\mathcal{A}\langle X \rangle$.

Example 1.6.10. Consider the structures $\underline{\mathbb{Q}} = \langle \mathbb{Q}, +, \cdot, 0 \rangle$, $\underline{\mathbb{Z}} = \langle \mathbb{Z}, +, \cdot, 0 \rangle$, and $\underline{\mathbb{N}} = \langle \mathbb{N}, +, \cdot, 0 \rangle$. Then $\underline{\mathbb{Q}}\langle\{1\}\rangle = \underline{\mathbb{N}}$, $\underline{\mathbb{Q}}\langle\{-1\}\rangle = \underline{\mathbb{Z}}$, and $\underline{\mathbb{Q}}\langle\{2\}\rangle$ is a substructure of $\underline{\mathbb{N}}$ on the set of all even numbers.

Example 1.6.11. If $\mathcal{A}$ has no functions (not even constants), e.g., if it is a graph or an order, then there is nothing to generate, and $\mathcal{A}\langle X \rangle = \mathcal{A} \restriction X$.

Expansion and Reduct

So far, we have constructed new structures by changing the universe. However, we can also keep the universe the same and add or remove relations, functions, and constants. The result of such an operation is called an *expansion* or a *reduct.* Note that this is a structure in a different signature.

[25] The notion of a *subgraph* in graph theory often means just $E^{\mathcal{B}} \subseteq E^{\mathcal{A}} \cap (B \times B)$, not $E^{\mathcal{B}} = E^{\mathcal{A}} \cap (B \times B)$. However, we will use the term *subgraph* in the stricter sense, as an induced subgraph.

[26] See the notion of *linear span* of a set of vectors.

Definition 1.6.12 (Expansion and Reduct)**.** Let $L \subseteq L'$ be languages, $\mathcal{A}$ an L-structure, and $\mathcal{A}'$ an L'-structure on the same domain $A = A'$. If the interpretation of each [relation, function, constant] symbol from L is the same [relation, function, constant] in both $\mathcal{A}$ and $\mathcal{A}'$, then we say that the structure $\mathcal{A}'$ is an *expansion* of the structure $\mathcal{A}$ to the language L' (also called an *L'-expansion*) and that the structure $\mathcal{A}$ is a *reduct* of the structure $\mathcal{A}'$ to the language L (also called an *L-reduct*).

Example 1.6.13. Consider the group of integers $\langle \mathbb{Z}, +, -, 0\rangle$. Then the structure $\langle \mathbb{Z}, +\rangle$ is its reduct, while the structure $\langle \mathbb{Z}, +, -, 0, \cdot, 1\rangle$ (the *ring* of integers) is its expansion.

Example 1.6.14. Consider a graph $\mathcal{G} = \langle G, E^{\mathcal{G}}\rangle$. Then the structure $\langle G, E^{G}, c_v^{\mathcal{G}}\rangle_{v\in G}$ in the language $\langle E, c_v\rangle_{v\in G}$, where $c_v^{\mathcal{G}} = v$ for all vertices $v \in G$, is an *expansion of $\mathcal{G}$ with names of elements (from the set G).*

1.6.1 Theorem on Constants

The *theorem on constants* states (informally) that satisfying a formula with a single free variable is equivalent to satisfying a sentence in which this free variable is replaced (substituted) by a *new* constant symbol (which is not bound by any axioms). The key fact is that this new symbol can be interpreted as any (hence each) element in each of the models. We will later use this trick in the tableau method.

Theorem 1.6.15 (Theorem on Constants)**.** *Let φ be a formula in the language L with free variables $x_1, \ldots, x_n$. Let L' be the extension of the language with new constant symbols $c_1, \ldots, c_n$ and let T' be the same theory as T but in the language L'. Then:*

$$T \models \varphi \text{ if and only if } T' \models \varphi(x_1/c_1, \ldots, x_n/c_n)$$

Proof. It is sufficient to prove the statement for one free variable x and one constant c; by induction, it can be easily extended to n constants.

First, suppose that φ holds in every model of the theory T. We want to show that $\varphi(x/c)$ holds in every model $\mathcal{A}'$ of the theory T'. So take such a model $\mathcal{A}'$ and any assignment $e\colon \mathrm{Var} \to A'$ and show that $\mathcal{A}' \models \varphi(x/c)[e]$.

Let $\mathcal{A}$ be the reduct of $\mathcal{A}'$ to the language L ('forget' the constant $c^{\mathcal{A}'}$). Note that $\mathcal{A}$ is a model of the theory T (the axioms of T are the same as T', they do not contain the symbol c) and hence φ holds in it. Since, by assumption, $\mathcal{A} \models \varphi[e']$ for *any* assignment e', it also holds for the assignment $e(x/c^{\mathcal{A}'})$ in which we evaluate the variable x as the interpretation of the constant symbol c in the structure $\mathcal{A}'$, so we have $\mathcal{A} \models \varphi[e(x/c^{\mathcal{A}'})]$. But this means that $\mathcal{A}' \models \varphi(x/c)[e]$, which is what we wanted to prove.

Conversely, suppose that $\varphi(x/c)$ holds in every model of the theory T' and show that φ holds in every model $\mathcal{A}$ of the theory T. So take such a model $\mathcal{A}$ and some assignment $e\colon \mathrm{Var} \to A$ and show that $\mathcal{A} \models \varphi[e]$.

Let $\mathcal{A}'$ be the expansion of $\mathcal{A}$ to the language L', where the constant symbol c is interpreted as the element $c^{\mathcal{A}'} = e(x)$. Since, by assumption, $\mathcal{A}' \models \varphi(x/c)[e']$ for all assignments e', it also holds that $\mathcal{A}' \models \varphi(x/c)[e]$, which means that $\mathcal{A}' \models \varphi[e]$. (Since $e = e(x/c^{\mathcal{A}'})$ and $\mathcal{A}' \models \varphi(x/c)[e]$ if and only if $\mathcal{A}' \models \varphi[e(x/c^{\mathcal{A}'})]$, which is $\mathcal{A}' \models \varphi[e]$.) But the formula φ does not contain c (here we use that c is *new*), so we also have $\mathcal{A} \models \varphi[e]$. □

1.7 Extension of Theories

The notion of *extension* of a theory is defined similarly as in propositional logic:

Definition 1.7.1 (Extension of a Theory)**.** Let T be a theory in the language L.

- An *extension* of the theory T is any theory T' in the language $L' \supseteq L$ satisfying $\mathrm{Csq}_L(T) \subseteq \mathrm{Csq}_{L'}(T')$,
- it is a *simple extension* if $L' = L$,
- it is a *conservative extension* if $\mathrm{Csq}_L(T) = \mathrm{Csq}_L(T') = \mathrm{Csq}_{L'}(T') \cap \mathrm{Fm}_L$, where Fm_L denotes the set of all formulas in the language L.
- The theory T' (in the language L) is *equivalent* to the theory T if T' is an extension of T and T is an extension of T'.

Similar to propositional logic, for theories in the same language, the following semantic description of these notions holds:

Observation 1.7.2. *Let T, T' be theories in the language L. Then:*

- *T' is an extension of T if and only if $\mathrm{M}_L(T') \subseteq \mathrm{M}_L(T)$.*
- *T' is equivalent to T if and only if $\mathrm{M}_L(T') = \mathrm{M}_L(T)$.*

What about the case when the theory T' is in a larger language than T? Recall the situation in propositional logic, described in Observation **??**. We will formulate and prove an analogous statement: While in propositional logic we added values for new atomic propositions or forgot them, in predicate logic we will expand or reduce structures, i.e., add or forget interpretations of relation, function, and constant symbols. The principle of the statement remains the same.

Proposition 1.7.3. *Let $L \subseteq L'$ be languages, T a theory in the language L, and T' a theory in the language L'.*

(i) T' is an extension of the theory T if and only if the reduct of every model of T' to the language L is a model of T.

(ii) If T' is an extension of the theory T, and every model of T can be expanded to the language L' into some model of the theory T', then T' is a conservative extension of the theory T.

Remark 1.7.4. The reverse implication in part (ii) also holds, but the proof is not as simple as in propositional logic, so we will not present it. (The problem is how to obtain from a model of T that cannot be expanded to a model of T' an L-sentence that holds in T but not in T'.)

Proof. First, let us prove (i): Let $\mathcal{A}'$ be a model of the theory T' and denote by $\mathcal{A}$ its reduct to the language L. Since T' is an extension of the theory T, every axiom $\varphi \in T$ holds in T', and thus in $\mathcal{A}'$. But then $\mathcal{A} \models \varphi$ (since φ contains only symbols from the language L), hence $\mathcal{A}$ is a model of T.

Conversely, let φ be an L-sentence such that $T \models \varphi$. We want to show that $T' \models \varphi$. For any model $\mathcal{A}' \in \mathrm{M}_{L'}(T')$ we know that its L-reduct $\mathcal{A}$ is a model of T, hence $\mathcal{A} \models \varphi$. Therefore, $\mathcal{A}' \models \varphi$ (again, since φ is in the language L).

Now (ii): Take any L-sentence φ that holds in the theory T', and show that it also holds in T. Every model $\mathcal{A}$ of the theory T can be expanded to some model $\mathcal{A}'$ of the theory T'. We know that $\mathcal{A}' \models \varphi$, so $\mathcal{A} \models \varphi$. Thus, we have shown that $T \models \varphi$, i.e., it is a conservative extension. □

1.7.1 Extension by Definitions

Now we will show a special kind of conservative extension, namely the extension *by definitions* of new (relation, function, constant) symbols.

Definition of a Relation Symbol

The simplest case is defining a new relation symbol $R(x_1, \ldots, x_n)$. Any formula with n free variables $\psi(x_1, \ldots, x_n)$ can serve as the definition.

Example 1.7.5. First, let us provide some examples:

- Any theory in a language with equality can be extended with the binary relation symbol $\neq$, which is *defined* by the formula $\neg x_1 = x_2$. This means that we require $x_1 \neq x_2 \leftrightarrow \neg x_1 = x_2$.
- The theory of order can be extended with the symbol $<$ for strict order, which is *defined* by the formula $x_1 \leq x_2 \wedge \neg x_1 = x_2$. This means that we require $x_1 < x_2 \leftrightarrow x_1 \leq x_2 \wedge \neg x_1 = x_2$.
- In arithmetic, we can introduce the symbol $\leq$ using $x_1 \leq x_2 \leftrightarrow (\exists y)(x_1 + y = x_2)$.

Now, we give the definition:

Definition 1.7.6 (Definition of a Relation Symbol)**.** Let T be a theory and $\psi(x_1, \ldots, x_n)$ a formula in the language L. Let L' be the extension of the language L with a new n-ary relation symbol R. The *extension of the theory T by the definition of R by the formula ψ* is the L'-theory:

$$T' = T \cup \{R(x_1, \ldots, x_n) \ \leftrightarrow \ \psi(x_1, \ldots, x_n)\}$$

Note that every model of T can be *uniquely* expanded to a model of T'. From Proposition **??**, it follows immediately:

Corollary 1.7.7. *T' is a conservative extension of T.*

We will also show that the new symbol can be replaced in formulas by its definition, thus obtaining a (T'-equivalent) formula in the original language:

Proposition 1.7.8. *For every L'-formula φ', there exists an L-formula φ such that $T' \models \varphi' \leftrightarrow \varphi$.*

Proof. It is necessary to replace atomic subformulas with the new symbol R, i.e., of the form $R(t_1, \ldots, t_n)$. Such a subformula is replaced by the formula $\psi'(x_1/t_1, \ldots, x_n/t_n)$, where ψ' is a variant of ψ ensuring the substitutability of all terms, i.e., for example, renaming all bound variables of ψ to entirely new ones (not appearing in the formula φ'). □

Definition of a Function Symbol

We define a new function symbol similarly, but we must ensure that the definition provides a unique way to interpret the new symbol.

Example 1.7.9. Again, we start with examples:

- In the theory of groups, we can introduce a *binary* function symbol $-_b$ using $+$ and unary $-$ as follows:
$$x_1 -_b x_2 = y \;\leftrightarrow\; x_1 + (-x_2) = y$$
It is clear that for every x, y, there *exists* a *unique* z satisfying the definition.

- Consider the theory of *linear orders*, i.e., the theory of orders together with the linearity axiom $x \leq y \vee y \leq x$. Define the binary function symbol min as follows:
$$\min(x_1, x_2) = y \;\leftrightarrow\; y \leq x_1 \wedge y \leq x_2 \wedge (\forall z)(z \leq x_1 \wedge z \leq x_2 \rightarrow z \leq y)$$
Existence and uniqueness hold thanks to linearity. However, if we had only the theory of orders, such a formula would not be a good definition: in some models, $\min(x_1, x_2)$ for some elements would not exist, thus failing the required *existence.*

Definition 1.7.10 (Definition of a Function Symbol)**.** Let T be a theory and $\psi(x_1, \ldots, x_n, y)$ a formula in the language L. Let L' be the extension of the language L with a new n-ary function symbol f. Let the following hold in the theory T:

- *existence axiom* $(\exists y)\psi(x_1, \ldots, x_n, y)$,
- *uniqueness axiom* $\psi(x_1, \ldots, x_n, y) \wedge \psi(x_1, \ldots, x_n, z) \rightarrow y = z$.

Then the *extension of the theory T by the definition of f by the formula ψ* is the L'-theory:
$$T' = T \cup \{f(x_1, \ldots, x_n) = y \;\leftrightarrow\; \psi(x_1, \ldots, x_n, y)\}$$

The formula ψ thus defines in each model an $(n+1)$-ary relation, and by this relation, we require it to be a function, i.e., for each n-tuple of elements, there exists a unique way to extend it into an $(n+1)$-tuple that is an element of this relation. Note that if the defining formula ψ is of the form $t(x_1, \ldots, x_n) = y$, where $x_1, \ldots, x_n$ are variables of the L-term t, then the existence and uniqueness axioms always hold.

Again, it holds that every model of T can be *uniquely* expanded to a model of T', hence:

Corollary 1.7.11. *T' is a conservative extension of T.*

And an analogous statement about unwinding definitions holds:

Proposition 1.7.12. *For every L'-formula φ', there exists an L-formula φ such that $T' \models \varphi' \leftrightarrow \varphi$.*

Proof. It suffices to prove for a formula φ' with a single occurrence of the symbol f; if there are multiple occurrences, we apply the procedure inductively, in the case of nested occurrences in one term $f(\ldots f(\ldots)\ldots)$, we proceed from inner to outer.

Denote by φ^* the formula obtained from φ' by replacing the term $f(t_1, \ldots, t_n)$ with a *new* variable z. Construct the formula φ as follows:
$$(\exists z)(\varphi^* \wedge \psi'(x_1/t_1, \ldots, x_n/t_n, y/z))$$

where ψ' is a variant of ψ ensuring the substitutability of all terms.

Let $\mathcal{A}$ be a model of the theory T' and an assignment e. Denote $a = (f(t_1, \ldots, t_n))^{\mathcal{A}}[e]$. Due to existence and uniqueness, it holds:

$$\mathcal{A} \models \psi'(x_1/t_1, \ldots, x_n/t_n, y/z)[e] \text{ if and only if } e(z) = a$$

Thus, $\mathcal{A} \models \varphi[e]$ if and only if $\mathcal{A} \models \varphi^*[e(z/a)]$, if and only if $\mathcal{A} \models \varphi'[e]$. This holds for any assignment e, hence $\mathcal{A} \models \varphi' \leftrightarrow \varphi$ for every model T', thus $T' \models \varphi' \leftrightarrow \varphi$. □

Definition of a Constant Symbol

A constant symbol is a special case of a function symbol of arity 0. Thus, the same statements hold. The existence and uniqueness axioms are: $(\exists y)\psi(y)$ and $\psi(y) \wedge \psi(z) \to y = z$. The extension by definition of a constant symbol c by the formula $\psi(y)$ is the theory $T' = T \cup \{c = y \leftrightarrow \psi(y)\}$.

Example 1.7.13. Let us show two examples:

- Any theory in the language of arithmetic can be extended by defining the constant symbol 1 with the formula $\psi(y)$ of the form $y = S(0)$, thus adding the axiom $1 = y \leftrightarrow y = S(0)$.
- Consider the theory of fields and a new symbol $\frac{1}{2}$, defined by the formula $y \cdot (1+1) = 1$, i.e., adding the axiom:
$$\frac{1}{2} = y \leftrightarrow y \cdot (1+1) = 1$$
This is not a correct extension by definition because the existence axiom does not hold. In the two-element field $\mathbb{Z}_2$ (and in every field *of characteristic 2*), the equation $y \cdot (1+1) = 1$ has no solution because $1 + 1 = 0$.

  However, if we take the theory of fields with characteristic not equal to 2, i.e., adding the axiom $\neg(1 + 1 = 0)$ to the theory of fields, it becomes a correct extension by definition. For example, in the field $\mathbb{Z}_3$, we have $\frac{1}{2}^{\mathbb{Z}_3} = 2$.

Extension by Definitions

If we have an L-theory T and an L'-theory T', we say that T' is an *extension* of T *by definitions* if it is obtained from T by a successive extension by definitions of relation and function (or possibly constant) symbols. The properties we proved about extensions by one symbol (whether relation or function) easily extend inductively to multiple symbols:

Corollary 1.7.14. *If T' is an extension of the theory T by definitions, then:*

- *Every model of the theory T can be uniquely expanded to a model of T'.*
- *T' is a conservative extension of T.*
- *For every L'-formula φ', there exists an L-formula φ such that $T' \models \varphi' \leftrightarrow \varphi$.*

Finally, let us show one more example, illustrating also the unwinding of definitions:

Example 1.7.15. In the theory $T = \{(\exists y)(x + y = 0), (x + y = 0) \wedge (x + z = 0) \rightarrow y = z\}$ of the language $L = \langle +, 0, \leq \rangle$ with equality, we can introduce $<$ and a unary function symbol $-$ by adding axioms:

$$\begin{aligned} -x = y &\leftrightarrow x + y = 0 \\ x < y &\leftrightarrow x \leq y \wedge \neg(x = y) \end{aligned}$$

The formula $-x < y$ (in the language $L' = \langle +, -, 0, \leq, < \rangle$ with equality) in this extension by definitions is equivalent to the following formula:

$$(\exists z)((z \leq y \wedge \neg(z = y)) \wedge x + z = 0)$$

1.8 Definability in Structures

A formula with one free variable x can be understood as a *property* of elements. In a given structure, such a formula *defines* the set of elements that satisfy this property, i.e., those for which the formula holds under an assignment e where $e(x) = a$. If we have a formula with two free variables, it defines a binary relation, etc. We now formalize this concept. Recall that the notation $\varphi(x_1, \ldots, x_n)$ means that $x_1, \ldots, x_n$ are precisely all the free variables of the formula φ.

Definition 1.8.1 (Definable Sets)**.** Let $\varphi(x_1, \ldots, x_n)$ be a formula and $\mathcal{A}$ a structure in the same language. The *set defined by the formula* $\varphi(x_1, \ldots, x_n)$ *in the structure* $\mathcal{A}$, denoted by $\varphi^{\mathcal{A}}(x_1, \ldots, x_n)$, is:

$$\varphi^{\mathcal{A}}(x_1, \ldots, x_n) = \{(a_1, \ldots, a_n) \in A^n \mid \mathcal{A} \models \varphi[e(x_1/a_1, \ldots, x_n/a_n)]\}$$

In short, we can also write $\varphi^{\mathcal{A}}(\bar{x}) = \{\bar{a} \in A^n \mid \mathcal{A} \models \varphi[e(\bar{x}/\bar{a})]\}$.

Example 1.8.2. Here are some examples:

- The formula $\neg(\exists y)E(x, y)$ defines the set of all *isolated* vertices in a given graph.
- Consider the field of real numbers $\underline{\mathbb{R}}$. The formula $(\exists y)(y \cdot y = x) \wedge \neg(x = 0)$ defines the set of all positive real numbers.
- The formula $x \leq y \wedge \neg(x = y)$ defines the relation of *strict ordering* $<^S$ in a given ordered set $\langle S, \leq^S \rangle$.

It is often useful to talk about properties of elements relative to other elements of a given structure. This cannot be expressed purely syntactically, but we can substitute elements of the structure as *parameters* for some of the free variables. The notation $\varphi(\bar{x}, \bar{y})$ means that the formula φ has free variables $x_1, \ldots, x_n, y_1, \ldots, y_k$ (for some n, k).

Definition 1.8.3. Let $\varphi(\bar{x}, \bar{y})$ be a formula, where $|\bar{x}| = n$ and $|\bar{y}| = k$, $\mathcal{A}$ a structure in the same language, and $\bar{b} \in A^k$. The *set defined by the formula* $\varphi(\bar{x}, \bar{y})$ *with parameters* $\bar{b}$ *in the structure* $\mathcal{A}$, denoted by $\varphi^{\mathcal{A},\bar{b}}(\bar{x}, \bar{y})$, is:

$$\varphi^{\mathcal{A},\bar{b}}(\bar{x}, \bar{y}) = \{\bar{a} \in A^n \mid \mathcal{A} \models \varphi[e(\bar{x}/\bar{a}, \bar{y}/\bar{b})]\}$$

For a structure $\mathcal{A}$ and a subset $B \subseteq A$, let $\mathrm{Df}^n(\mathcal{A}, B)$ denote the set of all sets definable in the structure $\mathcal{A}$ with parameters from B.

Example 1.8.4. For $\varphi(x,y) = E(x,y)$, $\varphi^{\mathcal{G},v}(x,y)$ is the set of all neighbors of vertex v.

Observation 1.8.5. *The set* $\mathrm{Df}^n(\mathcal{A}, B)$ *is closed under complement, intersection, and union, and contains* $\emptyset$ *and* A^n*. Therefore, it is a subalgebra of the power set algebra* $\mathcal{P}(A^n)$*.*

1.8.1 Database Queries

Definability finds natural application in relational databases, such as in the well-known query language SQL. A *relational database* consists of one or more *tables*, sometimes called *relations*, with rows of a table being *records* or *tuples.* Essentially, it is a structure in a purely relational language. Consider a database containing two tables, Program and Movies, as illustrated in Figure **??**.

cinema	title	time
Atlas	Forrest Gump	20:00
Lucerna	Forrest Gump	21:00
Lucerna	Philadelphia	18:30
⋮	⋮	⋮

title	director	actor
Forrest Gump	R. Zemeckis	T. Hanks
Philadelphia	J. Demme	T. Hanks
Batman Returns	T. Burton	M. Keaton
⋮	⋮	⋮

Figure 1.3: Tables Program and Movies

An SQL query in its simplest form (ignoring, for example, *aggregate functions*) is essentially a formula, and the result of the query is the set defined by this formula (with parameters). For instance, when and where can we see a movie starring Tom Hanks?

select Program.cinema, Program.time **from** Program, Movies **where**
Program.title = Movies.title **and** Movies.actor = 'T. Hanks'

The result will be the set $\varphi^{\text{Database,'T. Hanks'}}(x_{\text{cinema}}, x_{\text{time}}, y_{\text{actor}})$ defined in the structure Database $= \langle D, \text{Program}, \text{Movies}\rangle$, where $D = \{\text{'Atlas'}, \text{'Lucerna'}, \dots, \text{'M. Keaton'}\}$, with the parameter 'T. Hanks' by the following formula $\varphi(x_{\text{cinema}}, x_{\text{time}}, y_{\text{actor}})$:

$$(\exists y_{\text{title}})(\exists y_{\text{director}})(\text{Program}(x_{\text{cinema}}, y_{\text{title}}, x_{\text{time}}) \wedge \text{Movies}(y_{\text{title}}, y_{\text{director}}, y_{\text{actor}}))$$

1.9 Relationship Between Propositional and Predicate Logic

We will now demonstrate how propositional logic can be 'simulated' in predicate logic, in particular in the theory of Boolean algebras. First, we introduce the axioms of this theory:

Definition 1.9.1 (Boolean Algebras)**.** The *theory of Boolean algebras* is the theory of the language $L = \langle -, \wedge, \vee, \perp, \top\rangle$ with equality consisting of the following axioms:[27]

- *associativity* of $\wedge$ and $\vee$:

$$x \wedge (y \wedge z) = (x \wedge y) \wedge z$$
$$x \vee (y \vee z) = (x \vee y) \vee z$$

- *commutativity* of $\wedge$ and $\vee$:

$$x \wedge y = y \wedge x$$
$$x \vee y = y \vee x$$

[27]Note the *duality*: by swapping $\wedge$ with $\vee$ and $\perp$ with $\top$, we obtain the same axioms.

- *distributivity* of $\wedge$ over $\vee$ and $\vee$ over $\wedge$:
$$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$$
$$x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$$
- *absorption*:
$$x \wedge (x \vee y) = x$$
$$x \vee (x \wedge y) = x$$
- *complementation*:
$$x \wedge (-x) = \bot$$
$$x \vee (-x) = \top$$
- *non-triviality*:
$$\neg(\bot = \top)$$

The smallest model is the *2-element Boolean algebra* $\langle \{0,1\}, f_\neg, f_\wedge, f_\vee, 0, 1\rangle$. Finite Boolean algebras are (up to *isomorphism*) precisely $\langle \{0,1\}^n, f^n_\neg, f^n_\wedge, f^n_\vee, (0,\dots,0), (1,\dots,1)\rangle$, where f^n means that the function f is applied component-wise.[28]

Propositions can thus be understood as *Boolean terms* (and the constants $\bot, \top$ represent falsity and truth), with the truth value of a proposition under a given assignment of propositional variables being given by the value of the corresponding term in the 2-element Boolean algebra. Moreover, the *algebra of propositions* of a given propositional language or theory is a Boolean algebra (this holds even for infinite languages).

On the other hand, if we have an *open* formula φ (without equality), we can represent atomic propositions using propositional variables and obtain a proposition that holds if and only if φ holds. We will learn more about this direction in Chapter **??** (on resolution in predicate logic), where we will first eliminate quantifiers using the so-called *Skolemization*.

Propositional logic could also be introduced as a fragment of predicate logic if we allowed *nullary relations* (and nullary relation symbols in the language): $A^0 = \{\emptyset\}$, thus on any set there are precisely two nullary relations $R^A \subseteq A^0$: $R^A = \emptyset = 0$ and $R^A = \{\emptyset\} = \{0\} = 1$. However, we will not do that.

[28]These Boolean algebras are isomorphic to the *power set algebras* $\mathcal{P}(\{1,\dots,n\})$, the isomorphism being given by the bijection between subsets and their characteristic vectors.