Chapter 2

First-Order Logic

2.1 Structures

2.1.A Relations and Functions

Definition 2.1.1 Let A be a *nonempty* set and let $n = 1, 2, \ldots$. An *n-ary relation* or *predicate* on A is a subset $R \subseteq A^n$, where $A^n = \{(a_1, \ldots, a_n) : a_1, \ldots, a_n \in A\}$ is the set of ordered n-tuples of elements of A. We call n the *arity* of R.

Examples 2.1.2

(i) $A = \mathbb{N}$ $(= \{0, 1, 2, \ldots\})$

$$R = \{n \in \mathbb{N} : n \text{ is even}\} \subseteq \mathbb{N} \qquad \text{(unary (1-ary))}$$
$$S = \{(m, n) \in \mathbb{N}^2 : m < n\} \subseteq \mathbb{N}^2 \qquad \text{(binary (2-ary))}$$
$$T = \{(m, n, p) \in \mathbb{N}^3 : m \equiv n \mod (p)\} \subseteq \mathbb{N}^3 \qquad \text{(ternary (3-ary))}$$

(ii) $A = \mathbb{R}$ (= the set of reals)

$$Q = \{x \in \mathbb{R} : x \text{ is rational}\} \subseteq \mathbb{R} \qquad \text{(unary)}$$
$$P = \{(x, y) \in \mathbb{R}^2 : y = x^2\} \subseteq \mathbb{R}^2 \qquad \text{(binary)}$$
$$U = \left\{ \begin{array}{l} (a_1, \ldots, a_n) \in \mathbb{R}^n : \\ a_1 x^{n-1} + a_2 x^{n-2} + \cdots + a_{n-1} x + a_n \\ \text{has } n - 1 \text{ real solutions.} \end{array} \right\} \qquad (n\text{-ary})$$
$$B = \{(x, y, z) : y \text{ is between } x \text{ and } z\} \qquad \text{(ternary)}$$

(iii) $A = V$, where $G = \langle V, E\rangle$ is a graph with set of vertices V and set of edges E.

$$E = \{(x, y) \in V^2 : \ x \text{ and } y \text{ are connected by an edge}\} \qquad \text{(binary)}$$

$$F = \left\{(x, y) \in V^2 : \begin{array}{l} x \text{ and } y \text{ are connected by a path} \\ x_0 = x, x_1, \ldots, x_{n-1}, x_n = y. \end{array}\right\} \qquad \text{(binary)}$$

Remark. If $R \subseteq A^n$ is an n-ary relation, we also view R as expressing a property of n-tuples $(a_1, \ldots, a_n) \in A^n$ and we often write

$$R(a_1, \ldots, a_n),$$

instead of

$$(a_1, \ldots, a_n) \in R.$$

Thus $R(a_1, \ldots, a_n)$ means that $(a_1, \ldots, a_n)$ has property R or satisfies R. Note that a 1-ary relation is a property of single elements of A, or alternately just the subset of A containing all elements with that property.

Example 2.1.3 For R, S, T as in examples 2.1.2 (i) above:

(i) $R(n)$ means "n is even"

(ii) $S(m, n)$ means "$m < n$"

(iii) $T(m, n, p)$ means "$m \equiv n \mod (p)$

Remark. Example 2.1.3 (ii) above says that S essentially *is* the familiar "less than" relation, so we can write "$<(a, b)$" or even just "$a < b$" instead of $S(a, b)$. The same holds true for other familiar binary relations.

Definition 2.1.4 Let again A be any nonempty set. A *n-ary function* or *operation* on A is a map

$$f : A^n \to A,$$

in other words, a rule for assigning to each element $(x_1, \ldots, x_n)$ of A^n (where each $x_i \in A$) an element $f(x_1, \ldots, x_n)$ of A, called the *value* of the function at $(x_1, \ldots, x_n)$. We call n the *arity* of f.

Examples 2.1.5

(i) $A = \mathbb{N}$

$$\begin{aligned} f(n) &= n^2 && \text{(unary)} \\ g(m,n) &= m+n, && \text{(binary)} \\ a(m,n,p) &= m+n \mod (p), && \text{(ternary)} \\ q(m) &= \text{number of distinct primes dividing } m && \text{(unary)} \end{aligned}$$

(ii) $A = \mathbb{R}$

$$f(a_1, \ldots, a_n) = a_1^2 + a_2^2 + \cdots + a_n^2 \qquad (n\text{-ary})$$

(iii) $A =$ all strings of symbols in an alphabet S (which can be an arbitrary set).

$$\begin{aligned} f(s,t) &= s\hat{}t \text{ (the concatenation of } s \text{ and } t) && \text{(binary)} \\ g(s,t,u) &= \text{the result of substituting the leftmost occurrence of } s \text{ in } u \text{ by } t, \text{ if } s \text{ occurs in } u; \text{ otherwise, } u. && \text{(ternary)} \end{aligned}$$

By convention we allow also the case $n = 0$. Since $A^0 = \{\emptyset\}$ (that is, the only 0-tuple is the empty one), a 0-ary function f is simply an element of A, i.e. a *constant* in A. For example, π is a 0-ary function in $\mathbb{R}$.

Remark. On any set A we always have a canonical binary relation, namely "=" (equality among elements of A).

Remark. We have used the same notation $R(a_1, \ldots, a_n)$ to denote that the relation R is satisfied by the n-tuple $(a_1, \ldots, a_n)$ and $f(a_1, \ldots, a_n)$ to denote the value of the function f at the arguments $(a_1, \ldots, a_n)$. The context will tell us with which case we are dealing, so that this shouldn't cause any confusion.

2.1.B Structures

Definition 2.1.6 A *structure* consists of a nonempty set, called the *universe* of the structure, together with certain relations and functions on this set.

We write structures as

$$\mathcal{A} = \langle A; f, g, h, \ldots; R, S, T, \ldots \rangle,$$

where A is the universe, $f, g, h, \ldots$ are the functions of the structure, and $R, S, T, \ldots$ are the relations of the structure. Note that A can be finite or infinite (but not empty), and there can be any number (finite, infinite, or even zero) of functions, each with any arity $n \geq 0$, and any number (finite, infinite, or zero) of relations, each with any arity $n \geq 1$.

A structure codifies a context, of mathematical or other objects, which we study. The universe of the structure encompasses all the objects under consideration and the functions and relations of the structure are the basic or primitive operations and relations between these objects that we want to study.

Definition 2.1.7 The *signature* of $\mathcal{A}$ is

$$\langle \text{arity}(f), \text{arity}(g), \text{arity}(h), \ldots ; \text{arity}(R), \text{arity}(S), \text{arity}(T), \ldots \rangle.$$

Examples 2.1.8

(i) The *structure of (natural number) arithmetic*:

$$\mathcal{N} = \langle \mathbb{N}; 0, S, +, \cdot; < \rangle.$$

It has signature

$$\langle 0, 1, 2, 2; 2 \rangle.$$

(Here $S(n) = n + 1$ is the "successor" function.)

(ii) The *structure of the reals*:

$$\mathcal{R} = \langle \mathbb{R}; 0, 1, +, \cdot; \rangle.$$

It has signature

$$\langle 0, 0, 2, 2; \rangle.$$

This structure represents the basic algebra of real numbers. If we want to study something different, such as elementary trigonometry, a more appropriate structure might be $\langle \mathbb{R}; 0, 1, +, \cdot, \sin, \cos, \ldots \rangle$.

(iii) Let X be an arbitrary set (finite or infinite). Then

$$\langle P(X); \emptyset, \cap, \cup; \subseteq \rangle,$$

where $P(X)$ is the collection of all subsets of X, is a structure with signature

$$\langle 0, 2, 2; 2 \rangle.$$

(iv) Many algebraic structures are simply structures in the above sense, which satisfy certain properties (axioms). For example, a *group* is a structure of the form

$$\langle G; e, \cdot; \rangle$$

of signature $\langle 0, 2; \rangle$ satisfying the group laws.

(v) A *graph* is a structure of the form

$$\langle V; ; E \rangle$$

of signature $\langle ; 2 \rangle$ satisfying the usual conditions.

(vi) Let S be the set of records of employees in some company (containing, e.g., social security numbers, dates of birth, salary, etc.), let $N = \{0, 1, \ldots, n\}$ for n a large enough integer, and let

$$\mathcal{A} = \langle S \cup N; f, g, h; P, Q, R \rangle,$$

where

$$f(x) = \begin{cases} \text{the age of the employee with record } x & \text{if } x \in S \\ 0 & \text{otherwise.} \end{cases}$$

$$g(x) = \begin{cases} \text{the SSN of the employee with record } x & \text{if } x \in S \\ 0 & \text{otherwise.} \end{cases}$$

$$h(x) = \begin{cases} \text{the salary of the employee with record } x & \text{if } x \in S \\ 0 & \text{otherwise.} \end{cases}$$

$P(x)$ iff $x \in S$

$Q(x)$ iff $x \in N$

$R(x, y)$ iff x is a supervisor of y.

Then $\mathcal{A}$ is a structure of arity $\langle 1, 1, 1; 1, 1, 2 \rangle$. This example shows how we can include both the objects we want to study (here, the records) and some basic mathematical tools to use in reasoning about them (here, some natural numbers) in a single structure.

2.1.C Introduction to First-Order Languages

We now want to describe a formal language in which we can express statements about structures. We will give a series of examples showing how to develop such a language.

Example 2.1.9 Consider the statement:

"The square of an odd number is odd."

This is a statement about the structure of arithmetic,

$$\mathcal{N} = \langle \mathbb{N}; 0, S, +, \cdot; < \rangle.$$

We now introduce the following notation:

$x, y, z, \ldots$	variables representing arbitrary natural numbers;
$\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$	the usual propositional connectives;
$(,)$	parentheses;
$\exists$ (there exists),	the *existential* and
$\forall$ (for all)	*universal* quantifier;
$=$	symbol for equality;
$0, S, +, \cdot, <$	symbols for the operations and relations of $\mathcal{N}$.

In order to write this statement in a formal language using these symbols, we must first express it in a more precise form:

"For all natural numbers x, if x is odd, then $x \cdot x$ is odd."

We must further refine this by replacing "x is odd" with

"There exists a natural number y such that $x = 2 \cdot y + 1$."

Finally, since 1 and 2 are not constants in the structure of arithmetic, we must replace them with $S(0)$ and $S(S(0))$, respectively. We can now express the above statement in a formal language as:

$$\forall x[\exists y(x = S(S(0)) \cdot y + S(0)) \Rightarrow \exists z(x \cdot x = S(S(0)) \cdot z + S(0))]$$

We often use abbreviations, however, such as

$$S(0) = 1,\ S(S(0)) = 2,\ x \cdot x = x^2,\ u \cdot v = uv,$$

Using these abbreviations, our statement becomes more readable:

$$\forall x[\exists y(x = 2y + 1) \Rightarrow \exists z(x^2 = 2z + 1)].$$

Example 2.1.10 "Every non-empty subset of $\mathbb{N}$ has a least element".
The appropriate structure here is:

$$\langle \mathbb{N} \cup P(\mathbb{N});\ N, S, \in, < \rangle,$$

where

$$P(\mathbb{N}) = \{A : A \subseteq \mathbb{N}\} = \text{ the power set of } \mathbb{N}$$

$N(a)$ iff $a \in \mathbb{N}$
$S(a)$ iff $a \in P(\mathbb{N})$ (i.e., $a \subseteq \mathbb{N}$)
$a \in b$ iff $a \in \mathbb{N},\ b \in P(\mathbb{N}),$ and a is an element of b
$a < b$ iff $a, b \in \mathbb{N}$ and a is smaller than b

Using the same logical notation as before and symbols $N, S, \in, <$ for the relations of this structure we can express the previous statement as

$$\forall x[(S(x) \wedge \exists y(y \in x)) \Rightarrow \exists z(z \in x \wedge \forall w(w \in x \Rightarrow z = w \vee z < w))]$$

2.2 Syntax of First-Order Logic

We will now give a formal definition of the languages which were discussed by example in section 2.1.C.

2.2.A Symbols

Definition 2.2.1 A *first-order language* has the following set of symbols (alphabet):

(i) *Logical symbols*:

$x_1, x_2, x_3, \ldots$	(variables)
$\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$	(propositional connectives)
$\exists, \forall$	(quantifiers)
$(\ ,\ )$	(parentheses)
$=$	(equality symbol)

(ii) *Non-logical symbols*:

(a) For each $n \geq 0$ a set $\mathcal{F}_n$ (which may be empty) of symbols called *n-ary function symbols* (or function symbols of arity n). For $n = 0$, the symbols in $\mathcal{F}_0$ are called *constant* symbols.

(b) For each $n \geq 1$, a set $\mathcal{R}_n$ (which may be empty) of symbols called *n-ary relation symbols* (or relation symbols of arity n).

Remark. All these sets of symbols are of course assumed to be pairwise disjoint. The equality symbol $=$ is also a binary relation symbol, but plays a special role.

Remark. A first-order language is completely determined by its set $L = \bigcup_n \mathcal{F}_n \cup \bigcup_n \mathcal{R}_n$ of non-logical symbols (since the set of logical symbols is the same for all first-order languages).

Examples 2.2.2

(i) An example of a first-order language is $L = \{c, f, g; R\}$, where c is a constant symbol, f a unary function symbol, g a binary function symbol, and R a binary relation symbol.

(ii) A more meaningful example is the *language of arithmetic*:

$$L_{ar} = \{0, S, +, \cdot; <\}$$

where 0 is a constant symbol, S a unary function symbol, $+$ and $\cdot$ binary function symbols, and $<$ a binary relation symbol. This is the language that was used in example 2.1.9.

2.2.B Terms

We will now start defining the "grammatically correct" statements of a first-order language. We begin by defining strings called *terms*. Each term is intended to represent a single element of the structure. Terms are defined recursively as follows.

Definition 2.2.3 (Terms)

(i) Every variable symbol is a term.

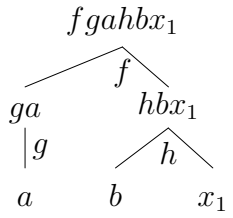
(ii) If $t_1, \ldots, t_n$ are terms and f is an n-ary function symbol, then $ft_1 \ldots t_n$ is a term. Note that if $n = 0$, i.e. f is a constant symbol, then f is a term all by itself.

Thus we use "Polish notation" to represent terms. (The weight of each n-ary function symbol is $1-n$ and that of constant symbols and variables is 1.)

Example 2.2.4 Suppose L contains the constant symbols a, b, the unary function symbol g and the binary function symbols f, h. Then the following is a term in this language

$$fgahbx_1.$$

This can be also represented by a parse tree as follows:



As in section 1.2.C, we have unique readability, in the sense that every term can be uniquely written as: x_i, a (constant symbol) or $ft_1 \dots t_n$, for uniquely determined $t_1, \dots, t_n$ terms and f an n-ary function symbol ($n \geq 1$).

To facilitate reading, we will add parentheses and adopt the following informal notation:

$$f(t_1, \dots, t_n) \text{ instead of } ft_1 \dots t_n$$

So we would write the term in the example above as

$$f(g(a), h(b, x_1)).$$

Also, for familiar binary function symbols f (like $+, \cdot$, etc) we will usually write

$$(tfs) \text{ instead of } fts,$$

and when there is no confusion we will even drop the parentheses and write tfs.

Finally, we often use $x, y, z, \dots$ for variables instead of $x_1, x_2, \dots$.

Example 2.2.5 In the language of arithmetic, the following are terms (using our informal notation, that is):

$(x + y) \cdot z$ (instead of $\cdot + xyz$)
$(x \cdot x) + (y \cdot y)$
$((x \cdot x) \cdot x + S(0) \cdot x) + S(S(0))$

In fact, all terms in this language are polynomials in several variables with coefficients in $\mathbb{N}$. We can view terms in arbitrary languages as "generalized polynomials."

Notation. If the variables occurring in a term t are among $x_1, \ldots, x_n$, we indicate this by writing $t(x_1, \ldots, x_n)$. Note that this does not mean that all of $x_1, \ldots, x_n$ occur in t. If $u_1, \ldots, u_n$ are terms, then

$$t[x_1/u_1, \ldots, x_n/u_n]$$

denotes the result of substituting each variable x_i in t by u_i. By induction on the construction of t, this is also a term.

Example 2.2.6 If

$$t(x, y) \text{ is } f(g(x), h(a, y)),$$

then

$$t[x/g(a), y/h(a, x))] \text{ is } f(g(g(a)), h(a, h(a, x)).$$

2.2.C Well-Formed Formulas

We can now define the statements in first-order languages, which we again call *(well-formed) formulas (wffs)*. We start with the simplest possible formulas.

Definition 2.2.7 An *atomic formula* is any string of the form

$$Rt_1 \ldots t_n$$

with R an n-ary relation symbol and $t_1, \ldots, t_n$ terms.

Example 2.2.8 Let $L = \{a, b, f, g, R\}$, a, b constant symbols, f a unary function symbol, g a binary function symbol, R a ternary relation symbol. The following are atomic formulas in this language:

$$Rfagxay, = gxyfa$$

Example 2.2.9 Let $L = \{<\}$, $<$ a binary relation symbol. Then the following is an atomic formula:

$$< xy.$$

Again we have unique readability: every atomic formula is of the form $Rt_1 \dots t_n$ for uniquely determined $R, t_1, \dots, t_n$.

And as we did with terms, we adopt various convenient informal notations. We write

$$R(t_1, \dots, t_n) \text{ instead of } Rt_1 \dots t_n.$$

For familiar binary relation symbols R (like $=, <$), we write

$$(tRs) \text{ instead of } Rts,$$

and when there is no chance of confusion we often omit the parentheses as well, writing tRs. So in example 2.2.8 we would write $R(f(a), g(x, a), y)$ and $(g(x, y) = f(a))$, and in example 2.2.9 we would write $(x < y)$ or even $x < y$.

Finally, we are ready to define:

Definition 2.2.10 (Well-Formed Formulas)

(i) Every atomic formula is a wff.

(ii) If A, B are wff, so are $\neg A, (A \wedge B), (A \vee B), (A \Rightarrow B), (A \Leftrightarrow B)$.

(iii) If A is a wff and x_i a variable, then $\exists x_i A,\ \forall x_i A$ are wff.

Examples 2.2.11

(i) Let $L = \{<\}$, $<$ a binary relation symbol. Then

$$\forall x \exists y (x < y)$$

is a wff, as is justified by the following parsing sequence:

$$(x < y),\ \exists y (x < y),\ \forall x \exists y (x < y).$$

(ii) In the same language,

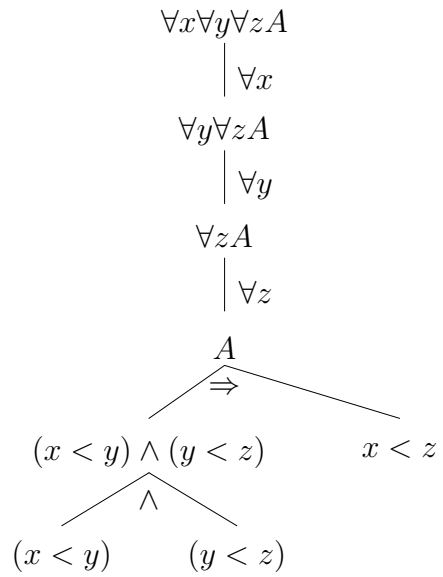
$$\forall x \forall y \forall z ((x < y) \wedge (y < z) \Rightarrow (x < z))$$

is a wff, as justified by the following parsing sequence:

$$(x < y),\ (y < z),\ (x < z),\ (x < y) \wedge (y < z),$$

$$\underbrace{((x < y) \wedge (y < z) \Rightarrow (x < z))}_{A},\ \forall z A,\ \forall y \forall z A,\ \forall x \forall y \forall z A.$$

We can also represent any wff by a parse tree. For (ii) this would be



(iii) $L = \{R, Q, S\}$, with R ternary and Q, S unary relation symbols. The following is a formula:

$$\exists x(\forall y R(x, y, z) \Rightarrow (\neg Q(x) \vee R(y))).$$

Yet again we have unique readability: every formula is in exactly one of the forms:

(i) atomic;

(ii) $\neg A,\ (A \wedge B),\ (A \vee B),\ (A \Rightarrow B),\ (A \Leftrightarrow B)$ for uniquely determined A, B;

(iii) $\exists x_i A,\ \forall x_i A$ for uniquely determined x_i, A.

Definition 2.2.12 A *subformula* of a formula A is any formula occurring in the parse tree of A.

2.2.D Scope and Free Variables

Proposition 2.2.13 *Let A be a wff, Q be either $\exists$ or $\forall$, x a variable. For any occurrence of the string Qx in A we have $A = SQxBT$, where S, T are strings and B is a uniquely determined wff called the* scope *of this occurrence of Qx.*

Example 2.2.14 $P(x,y) \Rightarrow \forall x\,[\underbrace{\exists y\,\underbrace{R(x,y)}_{\text{scope}} \Rightarrow \forall x\,\underbrace{Q(x,y)}_{\text{scope}}}_{\textit{scope}}]$

Example 2.2.15 $\exists y\,\forall x\,[\underbrace{\underbrace{\exists y\,\underbrace{R(x,y)}_{\text{scope}}}_{\text{scope}} \Rightarrow Q(x,y)}_{\text{scope}}]$

Proof of proposition 2.2.13. It is easy to prove by induction on the construction of A that for any occurrence of Qx in A we have $A = SQxBT$ for some wff B and some strings S, T. Uniqueness follows from the fact that a proper nonempty initial segment of a wff is not a wff (see Assignment #6).$\dashv$

Definition 2.2.16 An occurrence of a variable x in a formula A is *bound* if it occurs in a substring of the form QxB, where B is the scope of x. Otherwise this occurrence is called *free*.

Example 2.2.17 In example 2.2.14 the first occurrences of x, y and the last occurrence of y are free. All the other occurrences are bound. In example 2.2.15, all occurrences are bound.

Definition 2.2.18 A variable x is *free* in a wff A if it has *at least one* free occurrence. It is *bound* if it has at least one bound occurence.

Note that a variable can be both free and bound. This is in contrast to a single *occurrence* of a variable, which is either one or the other.

Example 2.2.19 In example 2.2.14, x, y are both free and bound. In 2.2.15 x, y are bound but not free. In $R(x,y)$ they are free but not bound. In $\exists x R(x,y)$, x is bound and not free, and y is free but not bound.

Remark. This should be compared with a notation such as $\int(x^2+y)dx$. Here x is "bound" and y is "free."

Bound variables are sometimes called "dummy variables," since they can be replaced with a different variable (which does not appear elsewhere) without changing the meaning of the formula. That is, $\int(x^2+y)dx$ and $\int(t^2+y)dt$ are equivalent. And once we define equivalence of wffs, pairs such as $\forall x(x<y)$ and $\forall z(z<y)$ will be equivalent as well: see fact 2.3.17.

Notation. $A(x_1,\ldots,x_n)$ means that the *free variables* of A are *among* $x_1,\ldots,x_n$. So in example 2.2.14, we would write $A(x,y)$.

Definition 2.2.20 A *sentence* is a wff with no free variables.

Example 2.2.21 $\forall x\exists y(x<y)$ is a sentence, as is example 2.2.15, but $0<x$ is not, nor is example 2.2.14.

2.3 Semantics of First-Order Logic

Now we will connect the first-order languages to the structures we intended them to represent.

2.3.A Structures and Interpretations

Definition 2.3.1 Let $L=\bigcup_{n\geq 0}\mathcal{F}_n\cup\bigcup_{n\geq 1}\mathcal{R}_n$ be a first-order language. A *structure* $\mathcal{M}$ *for* L consists of:

(i) a nonempty set M (the universe of this structure)

(ii) an n-ary function $f^{\mathcal{M}}: M^n\to M$ for each $f\in\mathcal{F}_n$. (For $n=0$, i.e. $a\in\mathcal{F}_0$ is a constant symbol, $a^{\mathcal{M}}$ is simply an element of M.)

(iii) an m-ary relation $R^{\mathcal{M}}\subseteq M^m$ for each $R\in\mathcal{R}_m$.

We write this as

$$\mathcal{M}=\langle M,\{f^{\mathcal{M}}\}_{f\in\bigcup_{n\geq 0}\mathcal{F}_n},\{R^{\mathcal{M}}\}_{R\in\bigcup_{n\geq 1}\mathcal{R}_n}\rangle,$$

and call $f^{\mathcal{M}}$ the *interpretation* of f and $R^{\mathcal{M}}$ the *interpretation* of R in $\mathcal{M}$.

Example 2.3.2 If

$$L = \{ \quad \overbrace{a, f, g}^{\text{function symbols}} \quad ; \quad \overbrace{S, R}^{\text{relation symbols}} \quad \}$$

(a is a constant; f unary; g binary; S binary, R ternary), then a structure for L has the form

$$\mathcal{M} = \langle M; a^{\mathcal{M}}, f^{\mathcal{M}}, g^{\mathcal{M}}; S^{\mathcal{M}}, R^{\mathcal{M}} \rangle,$$

with $a^{\mathcal{M}} \in M,\ f^{\mathcal{M}} : M \to M, g^{\mathcal{M}} : M^2 \to M,\ S^{\mathcal{M}} \subseteq M^2,\ R^{\mathcal{M}} \subseteq M^3$.

Example 2.3.3 Let $L_{ar} = \{0, S, +, \cdot, <\}$ be the language of arithmetic. Then

$$\mathcal{N} = \langle N; 0, S, +, \cdot; < \rangle$$

is a structure for L_{ar} called the *standard structure* of this language. Another structure for this language is

$$\mathcal{A} = \langle A; 0^{\mathcal{A}}, S^{\mathcal{A}}, +^{\mathcal{A}}, \cdot^{\mathcal{A}}; <^{\mathcal{A}} \rangle,$$

where $A = \mathbb{R},\ 0^{\mathcal{A}} = \pi, S^{\mathcal{A}}(a) = e^a, \mathcal{A} +^{\mathcal{A}} b = a + b,\ a \cdot^{\mathcal{A}} b = a \cdot b,\ a <^{\mathcal{A}} b$ iff $b = \cos(a)$.

Given a structure $\mathcal{M} = \langle M, \dots \rangle$ for L, we can assign to each term $t(x_1, \dots, x_n)$ of L an n-ary function $t^{\mathcal{M}} : M^n \to M$ by recursion as follows:

(i) If t is a variable x_i, then $i \leq n$, and we let $t^{\mathcal{M}}(a_1, \dots, a_n) = a_i$.

(ii) If t is a constant c, then $t^{\mathcal{M}}(a_1, \dots, a_n) = c^{\mathcal{M}}$.

(iii) If $t = f(t_1, \dots, t_k)$, then

$$t^{\mathcal{M}}(a_1, \dots, a_n) = f^{\mathcal{M}}(t_1^{\mathcal{M}}(a_1, \dots, a_n), \dots, t_k^{\mathcal{M}}(a_1, \dots, a_n)).$$

Definition 2.3.4 We call $t^{\mathcal{M}}(a_1, \dots, a_n)$ the *evaluation* or *interpretation* of t at $a_1, \dots, a_n$ in $\mathcal{M}$.

Remark. If $t = t()$ has no variables, then $t^{\mathcal{M}}$ is simply an element of $\mathcal{M}$.

Remark. If $t = t(x_1, \dots, x_n)$, then also $t = t(x_1, \dots, x_n, x_{n+1}, \dots, x_k)$ for any $k > n$ and strictly speaking we have an infinite list of functions $t_n^{\mathcal{M}}$, one for each n such that all the variables of t are among $x_1, \dots, x_n$. Notice however that for $n < k$, $t_n^{\mathcal{M}}(a_1, \dots, a_n) = t_k^{\mathcal{M}}(a_1, \dots, a_n, a_{n+1}, \dots, a_k)$.

Example 2.3.5 Let $L_{ar} = \{0, S, +, \cdot, <\}$ and $\mathcal{N} = \langle \mathbb{N}; 0, S, +, \cdot; < \rangle$ its standard structure. Then if

$$t(x, y) = (x \cdot x + y) + S(S(0)),$$

we have

$$t^{\mathcal{N}} : \mathbb{N}^2 \to \mathbb{N},$$
$$t^{\mathcal{N}}(a, b) = a^2 + b + 2.$$

2.3.B Models and Validities

Now let L be a first-order language, A a wff in L, $\mathcal{M}$ a structure for L, and consider an assignment $x_i \mapsto a_i$ which associates with each variable x_i an element a_i of M (the universe of $\mathcal{M}$).

Definition 2.3.6 We say that A is *true* in $\mathcal{M}$ under this assignment, which we write

$$\mathcal{M}, a_1, a_2, \ldots \models A,$$

if the following recursive definition is satisfied:

(i) If A is atomic, say $A = R(t_1, \ldots, t_k)$, with $t_i = t_i(x_1, \ldots, x_n)$, then

$$\mathcal{M}, a_1, a_2, \ldots \models A \quad \text{iff} \quad R^{\mathcal{M}}(t_1^{\mathcal{M}}(a_1, \ldots, a_n), \ldots, t_k^{\mathcal{M}}(a_1, \ldots, a_n)).$$

Here we understand that if R is the equality symbol $=$, then $=^{\mathcal{M}}$ is the equality on M, so

$$\mathcal{M}, a_1, a_2, \ldots \models t_1 = t_2 \quad \text{iff} \quad t_1^{\mathcal{M}}(a_1, \ldots, a_n) = t_2^{\mathcal{M}}(a_1, \ldots, a_n).$$

(ii) $\mathcal{M}, a_1, a_2, \ldots \models \neg A$ iff it is not the case that $\mathcal{M}, a_1, a_2, \ldots \models A$ (i.e. $\mathcal{M}, a_1, a_2, \cdots \not\models A$)

$\mathcal{M}, a_1, a_2 \ldots \models A \wedge B$ iff $\mathcal{M}, a_1, a_2, \ldots \models A$ and $\mathcal{M}, a_1, a_2, \ldots \models B$

$\mathcal{M}, a_1, a_2 \ldots \models A \vee B$ iff $\mathcal{M}, a_1, a_2, \ldots \models A$ or $\mathcal{M}, a_1, a_2, \ldots \models B$

$\mathcal{M}, a_1, a_2 \ldots \models A \Rightarrow B$ iff $\mathcal{M}, a_1, a_2, \cdots \not\models A$ or $\mathcal{M}, a_1, a_2, \ldots \models B$

$\mathcal{M}, a_1, a_2 \ldots \models A \Leftrightarrow B$ iff either $\mathcal{M}, a_1, a_2 \ldots \models A, B$ or $\mathcal{M}, a_1, a_2 \cdots \not\models A, B$

(iii) $\mathcal{M}, a_1, a_2, \ldots, a_i, \ldots \models \exists x_i A$ iff for *some* $b_i \in M$,

$$\mathcal{M}, a_1, a_2, \ldots, a_{i-1}, b_i, a_{i+1}, \ldots \models A;$$

$\mathcal{M}, a_1, a_2, \ldots, a_i, \ldots \models \forall x_i A$ iff for *all* $b_i \in M$,

$$\mathcal{M}, a_1, a_2, \ldots, a_{i-1}, b_i, a_{i+1}, \ldots \models A.$$

It is easy to show by induction on the construction of A, that if

$$A(x_1, \ldots, x_n)$$

is given and a_i, b_i $(i = 1, 2, \ldots) \in M$ are such that $a_i = b_i$ for all $i \leq n$, then

$$\mathcal{M}, a_1, a_2, \ldots \models A \text{ iff } \mathcal{M}, b_1, b_2, \ldots \models A,$$

so we abbreviate

$$\mathcal{M}, a_1, a_2, \ldots \models A$$

by

$$\mathcal{M} \models A[a_1, \ldots, a_n]$$

or if we need to be more explicit

$$\mathcal{M} \models A[x_1 \mapsto a_1, x_2 \mapsto a_2, \ldots, x_n \mapsto a_n].$$

We read this as: $\mathcal{M}$ makes $A[a_1, \ldots, a_n]$ true or A is *satisfied* in $\mathcal{M}, a_1, \ldots, a_n$.

Definition 2.3.7 In particular, if $A = A()$ is a sentence (has no free variables), we simply write

$$\mathcal{M} \models A,$$

and say that $\mathcal{M}$ *satisfies* A or A *is true in* $\mathcal{M}$. We also say that $\mathcal{M}$ *is a model of* A.

Example 2.3.8 Let $L = \{<\}$ ($<$ a binary relation symbol) and $\mathcal{M} = \langle \mathbb{N}, < \rangle$, and consider the following wffs:

(i) $A : x < y$. Then we have

$$\begin{aligned} &\mathcal{M} \models x < y[2, 3] && \text{(i.e. } 2 < 3) \\ &\mathcal{M} \models \neg x < y[2, 1] && \text{(i.e. } 2 \not< 1). \end{aligned}$$

(ii) $B : \exists x \forall y (x < y \vee x = y)$, a sentence. Then

$$\mathcal{M} \models \exists x \forall y (x < y \vee x = y). \tag{2.1}$$

By definition this means that there is $a \in \mathbb{N}$ so that $\mathcal{M} \models \forall y (x < y \vee x = y)[x \mapsto a]$, and again, by definition, this means that for all $b \in \mathbb{N}$,

$$\mathcal{M} \models (x < y \vee x = y)[x \mapsto a,\ y \mapsto b].$$

so equation (2.1) means that for some $a \in \mathbb{N}$, and all $b \in \mathbb{N}$, $a < b$ or $a = b$, i.e. $a \leq b$. Clearly $a = 0$ works.

If on the other hand we take $\mathcal{S} = \langle \mathbb{Z}, < \rangle$, a different model for L, then

$$\mathcal{S} \models \neg \exists x \forall y (x < y \vee x = y).$$

Example 2.3.9 Let $L_{ar} = \{0, S, +, \cdot, <\}$ be the language of arithmetic and $\mathcal{N} = \langle \mathbb{N}, 0, S, +, \cdot, < \}$ its standard structure. Then the following are true:

$$\mathcal{N} \models \forall x \forall y (x + S(y) = S(x + y))$$
$$\mathcal{N} \models \forall x \exists y_1 \exists y_2 \exists y_3 \exists y_4 (x = y_1 \cdot y_1 + y_2 \cdot y_2 + y_2 \cdot y_3 + y_4 \cdot y_4)$$

(parentheses omitted by association to the left.) The latter is because every natural number is the sum of four squares—*Lagrange's Theorem.*

$$\mathcal{N} \models \neg \forall x \exists y \exists z (x = y \cdot y + z \cdot z)$$

(since, e.g., 7 is not the sum of two squares).

$$\mathcal{N} \not\models \exists y (x = y \cdot y)\ [5].$$

However, take now the following structure for L_{ar}:

$$\mathcal{A} = \langle \mathbb{R}, 0, x \mapsto e^x, +, \cdot, < \rangle.$$

Then

$$\mathcal{A} \not\models \forall x \forall y (x + S(y) = S(x + y))$$

(because $a + e^b = e^{a+b}$ is not always true), and

$$\mathcal{A} \models \forall x \forall y (S(x + y) = S(x) \cdot S(y))$$

(but $\mathcal{N} \not\models \forall x \forall y (S(x + y) = S(x) \cdot S(y))$).

Definition 2.3.10 If S is a set of sentences, then a structure $\mathcal{M}$ *satisfies* or *models* S, in symbols

$$\mathcal{M} \models S,$$

if $\mathcal{M} \models A$ for all $A \in S$.

Definition 2.3.11 If S is a set of sentences and A is a sentence, then S *logically implies* A, in symbols

$$S \models A$$

if every model $\mathcal{M}$ of S is also a model of A.

Definition 2.3.12 If $S = \emptyset$ we simply write

$$\models A,$$

instead of $\emptyset \models A$, and we say that

$$A \text{ is } \textit{(logically) valid}$$

This simply means that A is true in every structure $\mathcal{M}$ of the language L.

Definition 2.3.13 Two sentences A, B are *logically equivalent* if $\{A\} \models B$ and $\{B\} \models A$, or $\models A \Leftrightarrow B$. We then write $A \equiv B$.

Definition 2.3.14 A set of sentences S is *satisfiable* if it has a model, i.e. there is a structure $\mathcal{M}$ with $\mathcal{M} \models S$.

These notions can be also extended to arbitrary formulas (not necessarily sentences). A set of formulas S *logically* implies a formula A, in symbols $S \models A$, if for any structure $\mathcal{M}$ and each assignment $x_i \mapsto a_i \in M$ (= the universe of $\mathcal{M}$), if all the formulas in S are true, so is A. A formula A is *valid* iff $\models A$. If $A = A(x_1, \ldots, x_n)$, then it is easy to see that A is valid iff $\forall x_1 \forall x_2 \ldots \forall x_n A$ is valid. (We call $\forall x_1 \forall x_2 \ldots \forall x_n A$ the *universal closure* of A.) Again A, B are *equivalent*, in symbols $A \equiv B$, if $\models A \Leftrightarrow B$. Finally, a set of formulas S is *satisfiable* if there is a structure $\mathcal{M}$ *and an assignment* $x_i \mapsto a_i$ in $\mathcal{M}$ satisfying all the formulas in S.

Examples 2.3.15

(i) An *instance of a tautology* is any formula in first-order logic obtained from a tautology in propositional logic by substituting each propositional variable p_i by a formula A_i.

For instance,

$$A \vee \neg A$$
$$\neg(A \wedge B) \Leftrightarrow (\neg A \vee \neg B)$$

are instances of tautologies. Clearly, every instance of a tautology is valid.

(ii) A list of useful valid formulas is given in Appendix B.

(iii) In general, $\not\models \forall y \exists x A \Rightarrow \exists x \forall y A$.

For example, take $L = \{R\}$, R a binary relation symbol, and take A to be $R(x, y)$. Then we claim that

$$\not\models \forall y \exists x R(x, y) \Rightarrow \exists x \forall y R(x, y).$$

To see this we must find a structure $\mathcal{M}$ for L which fails to satisfy this sentence. Take

$$\mathcal{M} = \langle \mathbb{Z}, R^{\mathcal{M}} \rangle$$

where $R^{\mathcal{M}} = \leq$, the usual (non-strict) ordering on $\mathbb{Z}$. Then $\mathcal{M} \models \forall y \exists x R(x, y)$ (i.e. for any $b \in \mathbb{Z}$ there is $a \in \mathbb{Z}$ with $a \leq b$: that is, given any integer, there is an integer less than or equal to it). On the other hand $\mathcal{M} \not\models \exists x \forall y R(x, y)$ (i.e. it is not true that there is some $a \in \mathbb{Z}$ such that for any $b \in \mathbb{Z}$, $a \leq b$: that is, there is no least integer).

As another counterexample, take $L = \emptyset$ and let A be the formula $x = y$. If $\mathcal{M} = \langle M \rangle$ is any structure, with M having more than one element, then $\mathcal{M}$ does not satisfy this sentence.

(iv) In general, $\not\models (\forall x A \Rightarrow \forall x B) \Rightarrow \forall x (A \Rightarrow B)$.

Let $L = \{P, Q\}$, P, Q unary relation symbols. Let A be $P(x)$, B be $Q(x)$. Then $\not\models (\forall x P(x) \Rightarrow \forall x Q(x)) \Rightarrow \forall x (P(x) \Rightarrow Q(x))$. To see this, consider the structure $\mathcal{M} = \langle \mathbb{N}, P^{\mathcal{M}}, Q^{\mathcal{M}} \rangle$, where $P^{\mathcal{M}}(n)$ iff n is even, $Q^{\mathcal{M}}(n)$ iff n is odd. Then $\mathcal{M} \models \forall x P(x) \Rightarrow \forall x Q(x)$, simply because $\mathcal{M} \not\models \forall x P(x)$, but $\mathcal{M} \not\models \forall x (P(x) \Rightarrow Q(x))$ (because this would mean that $P^{\mathcal{M}} \subseteq Q^{\mathcal{M}}$).

(v) If A is a formula, B a subformula of A, $B' \equiv B$, and A' is obtained from A by substituting B by B' (not necessarily in all occurrences of B in A), then $A' \equiv A$. (This can be easily proved by induction on the construction of A.)

If A is a formula, x is a variable and t is a term, we denote by $A[x/t]$ the formula obtained by substituting every *free* occurrence of x in A by t.

Example 2.3.16

$$\begin{aligned} A &: \exists x R(x,y) \Rightarrow P(x,z) \\ t &: g(x,y) \\ A[x/t] &: \exists x R(x,y) \Rightarrow P(g(x,y),z). \end{aligned}$$

Fact 2.3.17 (Alphabetic change of bound variables) *If A is a formula, x is a variable and y is a variable* not occurring *in A, then*

$$\begin{aligned} \forall x A &\equiv \forall y A[x/y] \\ \exists x A &\equiv \exists y A[x/y]. \end{aligned}$$

Remark. The requirement that y not occur in A is crucial, otherwise the fact is not true. For example, we have

$$\forall x R(x,y) \equiv \forall z R(z,y)$$

but

$$\forall x R(x,y) \not\equiv \forall y R(y,y),$$

since in the latter case, y already occurs in the formula, so that after substitution the meaning becomes different. We will discuss this more in section 2.7.A.

Fact 2.3.18 *Every formula is equivalent to one using only $\neg, \wedge, \exists$ or only $\neg, \wedge, \forall$ (more generally only $S \cup \{\exists\}$ or only $S \cup \{\forall\}$, for any complete set of connectives S in propositional logic).*

This is because for any formula A, we have $\neg\forall x A \equiv \exists x \neg A$ (i.e. A is not true for all x iff it fails for a specific x). Equivalently, we also have $\forall x \neg A \equiv \neg\exists x A$ (i.e. A is false for all x iff it is not true for any x). Using these facts and corollary 1.6.6, we can eliminate all but the desired connectives and quantifier.