Chapter 1

Preliminaries

1.1 Logic Formulas

When describing some state of affairs in the real world we often use *declarative*[1] sentences like:

(i) "Every mother loves her children"

(ii) "Mary is a mother and Tom is Mary's child"

By applying some general rules of reasoning such descriptions can be used to draw new conclusions. For example, knowing (i) and (ii) it is possible to conclude that:

(iii) "Mary loves Tom"

A closer inspection reveals that (i) and (ii) describe some *universe* of persons and some *relations* between these individuals — like "... is a mother", "... is a child of ..." or the relation "... loves ..." — which may or may not hold between the persons.[2] This example reflects the principal idea of *logic programming* — to describe possibly infinite relations on objects and to apply the programming system in order to draw conclusions like (iii).

For a computer to deal with sentences like (i)–(iii) the *syntax* of the sentences must be precisely defined. What is even more important, the rules of reasoning — like the one

[1] The notion of declarative sentence has its roots in linguistics. A declarative sentence is a complete expression of natural language which is either true or false, as opposed to e.g. imperative or interrogative sentences (commands and questions). Only declarative sentences can be expressed in predicate logic.

[2] Some people would probably argue that "being a mother" is not a relation but rather a property. However, for the sake of uniformity properties will be called relations and so will statements which relate more than two objects (like "... is the sum of ... and ...").

which permits inferring (*iii*) from (*i*) and (*ii*) — must be carefully formalized. Such problems have been studied in the field of mathematical logic. This chapter surveys basic logical concepts that are used later on in the book to relate logic programming and logic. (For basic set theoretic notions see Appendix B.)

The first concept considered is that of *logic formulas* which provide a formalized syntax for writing sentences like (*i*)–(*iii*). Such sentences refer to *individuals* in some *world* and to *relations* between those individuals. Therefore the starting point is an assumption about the alphabet of the language. It must include:

- symbols for denoting individuals (e.g. the symbol *tom* may be used to denote the person Tom of our example). Such symbols will be called *constants*;
- symbols for denoting relations (*loves*, *mother*, *child_of*). Such symbols are called *predicate symbols*.

Every predicate symbol has an associated natural number, called its arity. The relation named by an n-ary predicate symbol is a set of n-tuples of individuals; in the example above the predicate symbol *loves* denotes a set of pairs of persons, including the pair Mary and Tom, denoted by the constants *mary* and *tom*.

With the alphabet of constants, predicate symbols and some auxiliary characters, sentences of natural language like "Mary loves Tom" can be formalized as formulas like $loves(mary, tom)$.

The formal language should also provide the possibility of expressing sentences like (*i*) which refers to *all* elements of the described "world". This sentence says that "for all individuals X and Y, if X is a mother and Y is a child of X then X loves Y". For this purpose, the language of logic introduces the symbol of *universal quantifier* "$\forall$" (to be read "for every" or "for all") and the alphabet of *variables*. A variable is a symbol that refers to an unspecified individual, like X and Y above. Now the sentences (*i*)–(*iii*) can be formalized accordingly:

$$\forall X\,(\forall Y\,((mother(X) \wedge child\_of(Y, X)) \supset loves(X, Y))) \tag{1}$$

$$mother(mary) \wedge child\_of(tom, mary) \tag{2}$$

$$loves(mary, tom) \tag{3}$$

The symbols "$\wedge$" and "$\supset$" are examples of *logical connectives* which are used to combine logic formulas — "$\wedge$" reads "and" and is called *conjunction* whereas "$\supset$" is called *implication* and corresponds to the "if-then" construction above. Parentheses are used to disambiguate the language.

Another connective which will be used frequently is that for expressing negation. It is denoted by "$\neg$" (with reading "not"). For example the sentence "Tom does not love Mary" can be formalized as the formula:

$$\neg loves(tom, mary)$$

In what follows the symbol "$\exists$" is also sometimes used. It is called the *existential quantifier* and reads "there exists". The existential quantifier makes it possible to express the fact that, in the world under consideration, there exists at least one individual

which is in a certain relation with some other individuals. For example the sentence "Mary has a child" can be formalized as the formula:

$$\exists X \; child\_of(X, mary)$$

On occasion the logical connectives "$\vee$" and "$\leftrightarrow$" are used. They formalize the connectives "or" and "if and only if" ("iff").

So far individuals have been represented only by constants. However it is often the case that in the world under consideration, some "individuals" are "composed objects". For instance, in some world it may be necessary to discuss relations between families as well as relations between persons. In this case it would be desirable to refer to a given family by a construction composed of the constants identifying the members of the family (actually what is needed is a *function* that constructs a family from its members). The language of logic offers means of solving this problem. It is assumed that its alphabet contains symbols called *functors* that represent functions over object domains. Every functor has assigned a natural number called its arity, which determines the number of arguments of the function. The constants can be seen as 0-ary functors. Assume now that there is a ternary[3] functor *family*, a binary functor *child* and a constant *none*. The family consisting of the parents Bill and Mary and children Tom and Alice can now be represented by the construction:

$$family(bill, mary, child(tom, child(alice, none)))$$

Such a construction is called a *compound term*.

The above informal discussion based on examples of simple declarative sentences gives motivation for introducing basic constructs of the language of symbolic logic. The kind of logic used here is called *predicate logic*. Next a formal definition of this language is given. For the moment we specify only the form of allowed sentences, while the meaning of the language will be discussed separately. Thus the definition covers only the *syntax* of the language separated from its *semantics*.

From the syntactic point of view logic formulas are finite sequences of symbols such as variables, functors and predicate symbols. There are infinitely many of them and therefore the symbols are usually represented by finite strings of primitive characters. The representation employed in this book usually conforms to that specified in the ISO standard of the programming language Prolog (1995). Thus, the *alphabet* of the language of predicate logic consists of the following classes of symbols:

- *variables* which will be written as alphanumeric identifiers beginning with capital letters (sometimes subscriped). Examples of variables are $X, Xs, Y, X_7, \ldots$;
- *constants* which are numerals or alphanumeric identifiers beginning with lower-case letters. Examples of constants are $x, alf, none, 17, \ldots$;
- *functors* which are alphanumeric identifiers beginning with lower-case letters and with an associated arity > 0. To emphasize the arity n of a functor f it is sometimes written in the form f/n;

[3] Usually the terms *nullary*, *unary*, *binary* and *ternary* are used instead of 0-ary, 1-ary, 2-ary and 3-ary.

- *predicate symbols* which are usually alphanumeric identifiers starting with lowercase letters and with an associated arity ≥ 0. The notation p/n is used also for predicate symbols;
- *logical connectives* which are $\wedge$ (conjunction), $\neg$ (negation), $\leftrightarrow$ (logical equivalence), $\supset$ (implication) and $\vee$ (disjunction);
- *quantifiers* — $\forall$ (universal) and $\exists$ (existential);
- *auxiliary* symbols like parentheses and commas.

No syntactic distinction will be imposed between constants, functors and predicate symbols. However, as a notational convention we use $a, b, c, \ldots$ (with or without adornments) to denote constants and $X, Y, Z, \ldots$ to denote variables. Functors are denoted $f, g, h, \ldots$ and $p, q, r, \ldots$ are used to denote predicate symbols. Constants are sometimes viewed as nullary functors. Notice also that the sets of functors and predicate symbols may contain identical identifiers with different arities.

Sentences of natural language consist of words where objects of the described world are represented by nouns. In the formalized language of predicate logic objects will be represented by strings called *terms* whose syntax is defined as follows:

Definition 1.1 (Terms) The set $\mathcal{T}$ of *terms* over a given alphabet $\mathcal{A}$ is the smallest set such that:

- any constant in $\mathcal{A}$ is in $\mathcal{T}$;
- any variable in $\mathcal{A}$ is in $\mathcal{T}$;
- if f/n is a functor in $\mathcal{A}$ and $t_1, \ldots, t_n \in \mathcal{T}$ then $f(t_1, \ldots, t_n) \in \mathcal{T}$.

∎

In this book terms are typically denoted by s and t.

In natural language only certain combinations of words are meaningful sentences. The counterpart of sentences in predicate logic are special constructs built from terms. These are called *formulas* or well-formed formulas (*wff*) and their syntax is defined as follows:

Definition 1.2 (Formulas) Let $\mathcal{T}$ be the set of terms over the alphabet $\mathcal{A}$. The set $\mathcal{F}$ of *wff* (with respect to $\mathcal{A}$) is the smallest set such that:

- if p/n is a predicate symbol in $\mathcal{A}$ and $t_1, \ldots, t_n \in \mathcal{T}$ then $p(t_1, \ldots, t_n) \in \mathcal{F}$;
- if F and $G \in \mathcal{F}$ then so are $(\neg F)$, $(F \wedge G)$, $(F \vee G)$, $(F \supset G)$ and $(F \leftrightarrow G)$;
- if $F \in \mathcal{F}$ and X is a variable in $\mathcal{A}$ then $(\forall X F)$ and $(\exists X F) \in \mathcal{F}$.

∎

Formulas of the form $p(t_1, \ldots, t_n)$ are called *atomic formulas* (or simply *atoms*).

In order to adopt a syntax similar to that of Prolog, formulas in the form $(F \supset G)$ are instead written in the form $(G \leftarrow F)$. To simplify the notation parentheses will be removed whenever possible. To avoid ambiguity it will be assumed that the connectives

have a binding-order where $\neg$, $\forall$ and $\exists$ bind stronger than $\vee$, which in turn binds stronger than $\wedge$ followed by $\supset$ (i.e. $\leftarrow$) and finally $\leftrightarrow$. Thus $(a \leftarrow ((\neg b) \wedge c))$ will be simplified into $a \leftarrow \neg b \wedge c$. Sometimes binary functors and predicate symbols are written in infix notation (e.g. $2 \leq 3$).

Let F be a formula. An occurrence of the variable X in F is said to be *bound* either if the occurrence follows directly after a quantifier or if it appears inside the subformula which follows directly after "$\forall X$" or "$\exists X$". Otherwise the occurrence is said to be *free*. A formula with no free occurrences of variables is said to be *closed*. A formula/term which contains no variables is called *ground*.

Let $X_1, \ldots, X_n$ be all variables that occur free in a formula F. The closed formula of the form $\forall X_1(\ldots(\forall X_n\, F)\ldots)$ is called the *universal closure* of F and is denoted $\forall F$. Similarly, $\exists F$ is called the *existential closure* of F and denotes the formula F closed under existential quantification.

1.2 Semantics of Formulas

The previous section introduced the language of formulas as a formalization of a class of declarative statements of natural language. Such sentences refer to some "world" and may be true or false in this world. The meaning of a logic formula is also defined relative to an "abstract world" called an (algebraic) *structure* and is also either true or false. In other words, to define the meaning of formulas, a formal connection between the language and a structure must be established. This section discusses the notions underlying this idea.

As stated above declarative statements refer to individuals, and concern relations and functions on individuals. Thus the mathematical abstraction of the "world", called a structure, is a nonempty set of individuals (called the *domain*) with a number of relations and functions defined on this domain. For example the structure referred to by the sentences (*i*)–(*iii*) may be an abstraction of the world shown in Figure 1.1. Its domain consists of three individuals — Mary, John and Tom. Moreover, three relations will be considered on this set: a unary relation, "... is a mother", and two binary relations, "... is a child of ..." and "... loves ...". For the sake of simplicity it is assumed that there are no functions in the structure.

The building blocks of the language of formulas are constants, functors and predicate symbols. The link between the language and the structure is established as follows:

Definition 1.3 (Interpretation) An interpretation $\Im$ of an alphabet $\mathcal{A}$ is a nonempty domain $\mathcal{D}$ (sometimes denoted $|\Im|$) and a mapping that associates:

- each constant $c \in \mathcal{A}$ with an element $c_\Im \in \mathcal{D}$;
- each n-ary functor $f \in \mathcal{A}$ with a function $f_\Im : \mathcal{D}^n \to \mathcal{D}$;
- each n-ary predicate symbol $p \in \mathcal{A}$ with a relation $p_\Im \subseteq \underbrace{\mathcal{D} \times \cdots \times \mathcal{D}}_{n}$.

∎

The interpretation of constants, functors and predicate symbols provides a basis for assigning truth values to formulas of the language. The meaning of a formula will be

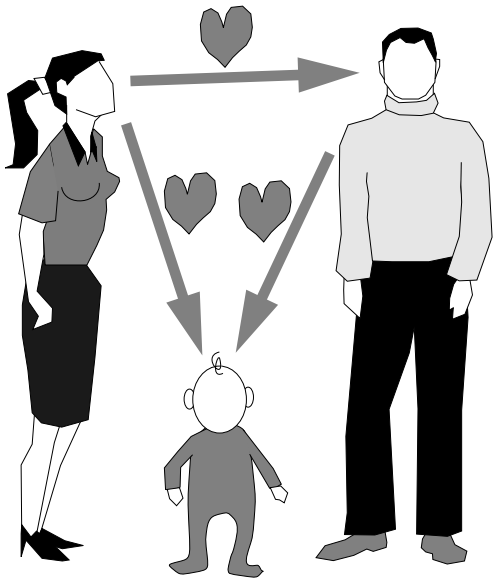

Figure 1.1: A family structure

defined as a function on meanings of its components. First the meaning of terms will be defined since they are components of formulas. Since terms may contain variables the auxiliary notion of *valuation* is needed. A valuation φ is a mapping from variables of the alphabet to the domain of an interpretation. Thus, it is a function which assigns objects of an interpretation to variables of the language. By the notation $\varphi[X \mapsto t]$ we denote the valuation which is identical to φ except that $\varphi[X \mapsto t]$ maps X to t.

Definition 1.4 (Semantics of terms) Let $\Im$ be an interpretation, φ a valuation and t a term. Then the *meaning* $\varphi_\Im(t)$ of t is an element in $|\Im|$ defined as follows:

- if t is a constant c then $\varphi_\Im(t) := c_\Im$;
- if t is a variable X then $\varphi_\Im(t) := \varphi(X)$;
- if t is of the form $f(t_1, \ldots, t_n)$, then $\varphi_\Im(t) := f_\Im(\varphi_\Im(t_1), \ldots, \varphi_\Im(t_n))$.

∎

Notice that the meaning of a compound term is obtained by applying the function denoted by its main functor to the meanings of its principal subterms, which are obtained by recursive application of this definition.

Example 1.5 Consider a language which includes the constant *zero*, the unary functor s and the binary functor *plus*. Assume that the domain of $\Im$ is the set of the natural numbers ($\mathbb{N}$) and that:

$$zero_\Im \quad := \quad 0$$

$$\begin{aligned} s_{\Im}(x) &:= 1 + x \\ plus_{\Im}(x, y) &:= x + y \end{aligned}$$

That is, *zero* denotes the *natural number* 0, s denotes the successor function and *plus* denotes the addition function. For the interpretation $\Im$ and a valuation φ such that $\varphi(X) := 0$ the meaning of the term $plus(s(zero), X)$ is obtained as follows:

$$\begin{aligned} \varphi_{\Im}(plus(s(zero), X)) &= \varphi_{\Im}(s(zero)) + \varphi_{\Im}(X) \\ &= (1 + \varphi_{\Im}(zero)) + \varphi(X) \\ &= (1 + 0) + 0 \\ &= 1 \end{aligned}$$

∎

The meaning of a formula is a truth value. The meaning depends on the components of the formula which are either (sub-) formulas or terms. As a consequence the meanings of formulas also rely on valuations. In the following definition the notation $\Im \models_\varphi Q$ is used as a shorthand for the statement "Q is true with respect to $\Im$ and φ" and $\Im \not\models_\varphi Q$ is to be read "Q is false w.r.t. $\Im$ and φ".

Definition 1.6 (Semantics of wff's) Let $\Im$ be an interpretation, φ a valuation and Q a formula. The meaning of Q w.r.t. $\Im$ and φ is defined as follows:

- $\Im \models_\varphi p(t_1, \ldots, t_n)$ iff $\langle \varphi_{\Im}(t_1), \ldots, \varphi_{\Im}(t_n) \rangle \in p_{\Im}$;
- $\Im \models_\varphi (\neg F)$ iff $\Im \not\models_\varphi F$;
- $\Im \models_\varphi (F \wedge G)$ iff $\Im \models_\varphi F$ and $\Im \models_\varphi G$;
- $\Im \models_\varphi (F \vee G)$ iff $\Im \models_\varphi F$ or $\Im \models_\varphi G$ (or both);
- $\Im \models_\varphi (F \supset G)$ iff $\Im \models_\varphi G$ whenever $\Im \models_\varphi F$;
- $\Im \models_\varphi (F \leftrightarrow G)$ iff $\Im \models_\varphi (F \supset G)$ and $\Im \models_\varphi (G \supset F)$;
- $\Im \models_\varphi (\forall X F)$ iff $\Im \models_{\varphi[X \mapsto t]} F$ for every $t \in |\Im|$;
- $\Im \models_\varphi (\exists X F)$ iff $\Im \models_{\varphi[X \mapsto t]} F$ for some $t \in |\Im|$.

∎

The semantics of formulas as defined above relies on the auxiliary concept of valuation that associates variables of the formula with elements of the domain of the interpretation. It is easy to see that the truth value of a closed formula depends only on the interpretation. It is therefore common practice in logic programming to consider all formulas as being implicitly universally quantified. That is, whenever there are free occurrences of variables in a formula its universal closure is considered instead. Since the valuation is of no importance for closed formulas it will be omitted when considering the meaning of such formulas.

Example 1.7 Consider Example 1.5 again. Assume that the language contains also a unary predicate symbol p and that:

$$p_{\Im} := \{\langle 1\rangle, \langle 3\rangle, \langle 5\rangle, \langle 7\rangle, \ldots\}$$

Then the meaning of the formula $p(zero) \wedge p(s(zero))$ in the interpretation $\Im$ is determined as follows:

$$\begin{aligned} \Im \models p(zero) \wedge p(s(zero)) \quad &\text{iff} \quad \Im \models p(zero) \text{ and } \Im \models p(s(zero)) \\ &\text{iff} \quad \langle \varphi_{\Im}(zero)\rangle \in p_{\Im} \text{ and } \langle \varphi_{\Im}(s(zero))\rangle \in p_{\Im} \\ &\text{iff} \quad \langle \varphi_{\Im}(zero)\rangle \in p_{\Im} \text{ and } \langle 1 + \varphi_{\Im}(zero)\rangle \in p_{\Im} \\ &\text{iff} \quad \langle 0\rangle \in p_{\Im} \text{ and } \langle 1\rangle \in p_{\Im} \end{aligned}$$

Now $\langle 1\rangle \in p_{\Im}$ but $\langle 0\rangle \notin p_{\Im}$ so the whole formula is false in $\Im$. ∎

Example 1.8 Consider the interpretation $\Im$ that assigns:

- the persons Tom, John and Mary of the structure in Figure 1.1 to the constants *tom*, *john* and *mary*;
- the relations "... is a mother", "... is a child of ..." and "... loves ..." of the structure in Figure 1.1 to the predicate symbols *mother*/1, *child_of*/2 and *loves*/2.

Using the definition above it is easy to show that the meaning of the formula:

$$\forall X\, \exists Y\, loves(X, Y)$$

is false in $\Im$ (since Tom does not love anyone), while the meaning of formula:

$$\exists X\, \forall Y\, \neg loves(Y, X)$$

is true in $\Im$ (since Mary is not loved by anyone). ∎

1.3 Models and Logical Consequence

The motivation for introducing the language of formulas was to give a tool for describing "worlds" — that is, algebraic structures. Given a set of closed formulas P and an interpretation $\Im$ it is natural to ask whether the formulas of P give a proper account of this world. This is the case if all formulas of P are true in $\Im$.

Definition 1.9 (Model) An interpretation $\Im$ is said to be a *model* of P iff every formula of P is true in $\Im$. ∎

Clearly P has infinitely many interpretations. However, it may happen that none of them is a model of P. A trivial example is any P that includes the formula $(F \wedge \neg F)$ where F is an arbitrary (closed) formula. Such sets of formulas are called *unsatisfiable*. When using formulas for describing "worlds" it is necessary to make sure that every description produced is *satisfiable* (that is, has at least one model), and in particular that the world being described is a model of P.

Generally, a satisfiable set of formulas has (infinitely) many models. This means that the formulas which properly describe a particular "world" of interest at the same time describe many other worlds.

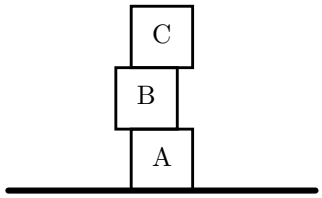

Figure 1.2: An alternative structure

Example 1.10 Figure 1.2 shows another structure which can be used as a model of the formulas (1) and (2) of Section 1.1 which were originally used to describe the world of Figure 1.1. In order for the structure to be a model the constants *tom*, *john* and *mary* are interpreted as the boxes 'A', 'B' and 'C' respectively — the predicate symbols *loves*, *child_of* and *mother* are interpreted as the relations "... is above ...", "... is below ..." and "... is on top". ∎

Our intention is to use the description of the world of interest to obtain more information about this world. This new information is to be represented by new formulas not explicitly included in the original description. An example is the formula (3) of Section 1.1 which is obtained from (1) and (2). In other words, for a given set P of formulas other formulas (say F) which are also true in the world described by P are searched for. Unfortunately, P itself has many models and does not uniquely identify the "intended model" which was described by P. Therefore it must be required that F is true in every model of P to guarantee that it is also true in the particular world of interest. This leads to the fundamental concept of *logical consequence*.

Definition 1.11 (Logical consequence) Let P be a set of closed formulas. A closed formula F is called a logical consequence of P (denoted $P \models F$) iff F is true in every model of P. ∎

Example 1.12 To illustrate this notion by an example it is shown that (3) is a logical consequence of (1) and (2). Let $\Im$ be an arbitrary interpretation. If $\Im$ is a model of (1) and (2) then:

$$\Im \models \forall X(\forall Y((mother(X) \wedge child\_of(Y,X)) \supset loves(X,Y))) \tag{4}$$

$$\Im \models mother(mary) \wedge child\_of(tom, mary) \tag{5}$$

For (4) to be true it is necessary that:

$$\Im \models_{\varphi} mother(X) \wedge child\_of(Y,X) \supset loves(X,Y) \tag{6}$$

for any valuation φ — specifically for $\varphi(X) = mary_{\Im}$ and $\varphi(Y) = tom_{\Im}$. However, since these individuals are denoted by the constants *mary* and *tom* it must also hold that:

$$\Im \models mother(mary) \wedge child\_of(tom, mary) \supset loves(mary, tom) \tag{7}$$

Finally, for this to hold it follows that $loves(mary, tom)$ must be true in $\Im$ (by Definition 1.6 and since (5) holds by assumption). Hence, any model of (1) and (2) is also a model of (3). ∎

This example shows that it may be rather difficult to prove that a formula is a logical consequence of a set of formulas. The reason is that one has to use the semantics of the language of formulas and to deal with all models of the formulas.

One possible way to prove $P \models F$ is to show that $\neg F$ is false in every model of P, or put alternatively, that the set of formulas $P \cup \{\neg F\}$ is unsatisfiable (has no model). The proof of the following proposition is left as an exercise.

Proposition 1.13 (Unsatisfiability) Let P be a set of closed formulas and F a closed formula. Then $P \models F$ iff $P \cup \{\neg F\}$ is unsatisfiable. ∎

It is often straightforward to show that a formula F is not a logical consequence of the set P of formulas. For this, it suffices to give a model of P which is not a model of F.

Example 1.14 Let P be the formulas:

$$\forall X(r(X) \supset (p(X) \vee q(X))) \tag{8}$$

$$r(a) \wedge r(b) \tag{9}$$

To prove that $p(a)$ is not a logical consequence of P it suffices to consider an interpretation $\Im$ where $|\Im|$ is the set consisting of the two persons "Adam" and "Eve" and where:

$a_\Im :=$ Adam
$b_\Im :=$ Eve
$p_\Im := \{\langle \text{Eve} \rangle\}$ % the property of being female
$q_\Im := \{\langle \text{Adam} \rangle\}$ % the property of being male
$r_\Im := \{\langle \text{Adam} \rangle, \langle \text{Eve} \rangle\}$ % the property of being a person

Clearly, (8) is true in $\Im$ since "any person is either female or male". Similarly (9) is true since "both Adam and Eve are persons". However, $p(a)$ is false in $\Im$ since Adam is not a female. ∎

Another important concept based on the semantics of formulas is the notion of *logical equivalence*.

Definition 1.15 (Logical equivalence) Two formulas F and G are said to be logically equivalent (denoted $F \equiv G$) iff F and G have the same truth value for all interpretations $\Im$ and valuations φ. ∎

Next a number of well-known facts concerning equivalences of formulas are given. Let F and G be arbitrary formulas and $H(X)$ a formula with zero or more free occurrences of X. Then:

$$\begin{array}{rcll}
\neg\neg F & \equiv & F & \\
F \supset G & \equiv & \neg F \vee G & \\
F \supset G & \equiv & \neg G \supset \neg F & \\
F \leftrightarrow G & \equiv & (F \supset G) \wedge (G \supset F) & \\
\neg(F \vee G) & \equiv & \neg F \wedge \neg G & \text{DeMorgan's law} \\
\neg(F \wedge G) & \equiv & \neg F \vee \neg G & \text{DeMorgan's law} \\
\neg\forall X H(X) & \equiv & \exists X \neg H(X) & \text{DeMorgan's law} \\
\neg\exists X H(X) & \equiv & \forall X \neg H(X) & \text{DeMorgan's law}
\end{array}$$

and if there are no free occurrences of X in F then:

$$\forall X(F \vee H(X)) \equiv F \vee \forall X H(X)$$

Proofs of these equivalences are left as an exercise to the reader.

1.4 Logical Inference

In Section 1.1 the sentence (*iii*) was obtained by reasoning about the sentences (*i*) and (*ii*). The language was then formalized and the sentences were expressed as the logical formulas (1), (2) and (3). With this formalization, reasoning can be seen as a process of manipulation of formulas, which from a given set of formulas, like (1) and (2), called the *premises*, produces a new formula called the *conclusion*, for instance (3). One of the objectives of the symbolic logic is to formalize "reasoning principles" as formal re-write rules that can be used to generate new formulas from given ones. These rules are called *inference rules*. It is required that the inference rules correspond to correct ways of reasoning — whenever the premises are true in any world under consideration, any conclusion obtained by application of an inference rule should also be true in this world. In other words it is required that the inference rules produce only logical consequences of the premises to which they can be applied. An inference rule satisfying this requirement is said to be *sound*.

Among well-known inference rules of predicate logic the following are frequently used:

- *Modus ponens* or elimination rule for implication: This rule says that whenever formulas of the form F and $(F \supset G)$ belong to or are concluded from a set of premises, G can be inferred. This rule is often presented as follows:

$$\frac{F \quad F \supset G}{G} \quad (\supset \mathrm{E})$$

- Elimination rule for universal quantifier: This rule says that whenever a formula of the form $(\forall X F)$ belongs to or is concluded from the premises a new formula can be concluded by replacing all free occurrences of X in F by some term t which is *free for* X (that is, all variables in t remain free when X is replaced by t: for details see e.g. van Dalen (1983) page 68). This rule is often presented as follows:

$$\frac{\forall X F(X)}{F(t)} \quad (\forall \mathrm{E})$$

- Introduction rule for conjunction: This rule states that if formulas F and G belong to or are concluded from the premises then the conclusion $F \wedge G$ can be inferred. This is often stated as follows:

$$\frac{F \quad G}{F \wedge G} \quad (\wedge \mathrm{I})$$

Soundness of these rules can be proved directly from the definition of the semantics of the language of formulas.

Their use can be illustrated by considering the example above. The premises are:

$$\forall X\ (\forall Y\ (\mathit{mother}(X) \wedge \mathit{child\_of}(Y, X) \supset \mathit{loves}(X, Y))) \tag{10}$$
$$\mathit{mother}(\mathit{mary}) \wedge \mathit{child\_of}(\mathit{tom}, \mathit{mary}) \tag{11}$$

Elimination of the universal quantifier in (10) yields:

$$\forall Y\ (\mathit{mother}(\mathit{mary}) \wedge \mathit{child\_of}(Y, \mathit{mary}) \supset \mathit{loves}(\mathit{mary}, Y)) \tag{12}$$

Elimination of the universal quantifier in (12) yields:

$$\mathit{mother}(\mathit{mary}) \wedge \mathit{child\_of}(\mathit{tom}, \mathit{mary}) \supset \mathit{loves}(\mathit{mary}, \mathit{tom}) \tag{13}$$

Finally *modus ponens* applied to (11) and (13) yields:

$$\mathit{loves}(\mathit{mary}, \mathit{tom}) \tag{14}$$

Thus the conclusion (14) has been produced in a formal way by application of the inference rules. The example illustrates the concept of *derivability*. As observed, (14) is obtained from (10) and (11) not directly, but in a number of inference steps, each of them adding a new formula to the initial set of premises. Any formula F that can be obtained in that way from a given set P of premises is said to be *derivable* from P. This is denoted by $P \vdash F$. If the inference rules are sound it follows that whenever $P \vdash F$, then $P \models F$. That is, whatever can be derived from P is also a logical consequence of P. An important question related to the use of inference rules is the problem of whether all logical consequences of an arbitrary set of premises P can also be derived from P. In this case the set of inference rules is said to be *complete*.

Definition 1.16 (Soundness and Completeness) A set of inference rules are said to be *sound* if, for every set of closed formulas P and every closed formula F, whenever $P \vdash F$ it holds that $P \models F$. The inference rules are *complete* if $P \vdash F$ whenever $P \models F$. ∎

A set of premises is said to be *inconsistent* if any formula can be derived from the set. Inconsistency is the proof-theoretic counterpart of unsatisfiability, and when the inference system is both sound and complete the two are frequently used as synonyms.

1.5 Substitutions

The chapter is concluded with a brief discussion on *substitutions* — a concept fundamental to forthcoming chapters. Formally a substitution is a mapping from variables of a given alphabet to terms in this alphabet. The following syntactic definition is often used instead:

Definition 1.17 (Substitutions) A *substitution* is a finite set of pairs of terms $\{X_1/t_1, \ldots, X_n/t_n\}$ where each t_i is a term and each X_i a variable such that $X_i \neq t_i$ and $X_i \neq X_j$ if $i \neq j$. The *empty substitution* is denoted ϵ. ∎

The *application* $X\theta$ of a substitution θ to a variable X is defined as follows:

$$X\theta \quad := \quad \begin{cases} t \text{ if } X/t \in \theta. \\ X \text{ otherwise} \end{cases}$$

In what follows let $Dom(\{X_1/t_1, \ldots, X_n/t_n\})$ denote the set $\{X_1, \ldots, X_n\}$. Also let $Range(\{X_1/t_1, \ldots, X_n/t_n\})$ be the set of all variables in $t_1, \ldots, t_n$. Thus, for variables not included in $Dom(\theta)$, θ behaves as the identity mapping. It is natural to extend the domain of substitutions to include also terms and formulas. In other words, it is possible to *apply* a substitution to an arbitrary term or formula in the following way:

Definition 1.18 (Application) Let θ be a substitution $\{X_1/t_1, \ldots, X_n/t_n\}$ and E a term or a formula. The application $E\theta$ of θ to E is the term/formula obtained by simultaneously replacing t_i for every free occurrence of X_i in E $(1 \leq i \leq n)$. $E\theta$ is called an *instance* of E. ∎

Example 1.19

$$\begin{aligned} p(f(X,Z), f(Y,a))\{X/a, Y/Z, W/b\} &= p(f(a,Z), f(Z,a)) \\ p(X,Y)\{X/f(Y), Y/b\} &= p(f(Y), b) \end{aligned}$$

∎

It is also possible to compose substitutions:

Definition 1.20 (Composition) Let θ and σ be two substitutions:

$$\begin{aligned} \theta &:= \{X_1/s_1, \ldots, X_m/s_m\} \\ \sigma &:= \{Y_1/t_1, \ldots, Y_n/t_n\} \end{aligned}$$

The *composition* $\theta\sigma$ of θ and σ is obtained from the set:

$$\{X_1/s_1\sigma, \ldots, X_m/s_m\sigma, Y_1/t_1, \ldots, Y_n/t_n\}$$

by removing all $X_i/s_i\sigma$ for which $X_i = s_i\sigma$ $(1 \leq i \leq m)$ and by removing those Y_j/t_j for which $Y_j \in \{X_1, \ldots, X_m\}$ $(1 \leq j \leq n)$. ∎

It is left as an exercise to prove that the above syntactic definition of composition actually coincides with function composition (see exercise 1.13).

Example 1.21

$$\{X/f(Z), Y/W\}\{X/a, Z/a, W/Y\} = \{X/f(a), Z/a, W/Y\}$$

∎

A kind of substitution that will be of special interest are the so-called idempotent substitutions:

Definition 1.22 (Idempotent substitution) A substitution θ is said to be *idempotent* iff $\theta = \theta\theta$. ∎

It can be shown that a substitution θ is *idempotent* iff $Dom(\theta) \cap Range(\theta) = \varnothing$. The proof of this is left as an exercise and so are the proofs of the following properties:

Proposition 1.23 (Properties of substitutions) Let θ, σ and γ be substitutions and let E be a term or a formula. Then:

- $E(\theta\sigma) = (E\theta)\sigma$
- $(\theta\sigma)\gamma = \theta(\sigma\gamma)$
- $\epsilon\theta = \theta\epsilon = \theta$

∎

Notice that composition of substitutions is not commutative as illustrated by the following example:

$$\{X/f(Y)\}\{Y/a\} = \{X/f(a), Y/a\} \neq \{Y/a\}\{X/f(Y)\} = \{Y/a, X/f(Y)\}$$

Exercises

1.1 Formalize the following sentences of natural language as formulas of predicate logic:

a) Every natural number has a successor.
b) Nothing is better than taking a nap.
c) There is no such thing as negative integers.
d) The names have been changed to protect the innocent.
e) Logic plays an important role in all areas of computer science.
f) The renter of a car pays the deductible in case of an accident.

1.2 Formalize the following sentences of natural language into predicate logic:

a) A bronze medal is better than nothing.
b) Nothing is better than a gold medal.
c) A bronze medal is better than a gold medal.

1.3 Prove Proposition 1.13.

1.4 Prove the equivalences in connection with Definition 1.15.

1.5 Let $F := \forall X\, \exists Y p(X, Y)$ and $G := \exists Y\, \forall X p(X, Y)$. State for each of the following four formulas whether it is satisfiable or not. If it is, give a model with the natural numbers as domain, if it is not, explain why.

$$(F \wedge G) \quad (F \wedge \neg G) \quad (\neg F \wedge \neg G) \quad (\neg F \wedge G)$$

1.6 Let F and G be closed formulas. Show that $F \equiv G$ iff $\{F\} \models G$ and $\{G\} \models F$.

1.7 Show that P is unsatisfiable iff there is some closed formula F such that $P \models F$ and $P \models \neg F$.

1.8 Show that the following three formulas are satisfiable only if the interpretation has an infinite domain

$$\forall X \neg p(X, X)$$
$$\forall X \forall Y \forall Z (p(X, Y) \wedge p(Y, Z) \supset p(X, Z))$$
$$\forall X \exists Y p(X, Y)$$

1.9 Let F be a formula and θ a substitution. Show that $\forall F \models \forall(F\theta)$.

1.10 Let P_1, P_2 and P_3 be sets of closed formulas. Redefine $\models$ in such a way that $P_1 \models P_2$ iff every formula in P_2 is a logical consequence of P_1. Then show that $\models$ is transitive — that is, if $P_1 \models P_2$ and $P_2 \models P_3$ then $P_1 \models P_3$.

1.11 Let P_1 and P_2 be sets of closed formulas. Show that if $P_1 \subseteq P_2$ and $P_1 \models F$ then $P_2 \models F$.

1.12 Prove Proposition 1.23.

1.13 Let θ and σ be substitutions. Show that the composition $\theta\sigma$ is equivalent to function composition of the mappings denoted by θ and σ.

1.14 Show that a substitution θ is idempotent iff $Dom(\theta) \cap Range(\theta) = \varnothing$.

1.15 Which of the following statements are true?

- if $\sigma\theta = \delta\theta$ then $\sigma = \delta$
- if $\theta\sigma = \theta\delta$ then $\sigma = \delta$
- if $\sigma = \delta$ then $\sigma\theta = \delta\theta$