2 Technical Preliminaries

In this chapter we briefly review some basics of first-order predicate logic. As we mentioned above, in what follows we only assume familiarity with material tha tis covered in a typical basic introductory logic course or an introductory textbook such as (Halbach 2010). The expert reader may safely skip this chapter. In the respective chapters we define our languages in some detail. The present chapter only serves the purpose of preparing a reader with little background in logic to understand the terminology used later, as different terminologies are used in different textbooks. Moreover, we discuss some topics that the reader might not have seen. For instance, we will make heavy use of function symbols in our formal languages. Function symbols, however, are often not covered in introductory logic books. Thus we introduce them here and explain them in some detail.

Abbreviations

We will use some abbreviations. 'iff' is short for 'if and only if'. Occasionally we use '$\Rightarrow$' for 'if then'. The double arrow does not belong to any object language; it is just an abbreviation in our metalanguage.

2.1 Languages of First-Order Predicate Logic

All languages we consider have an infinite stock of variables: $v_0, v_1, v_2, v_3, \ldots$ Some also contain function symbols. Each function symbol has an arity assigned, which is some natural number 0, 1,...

All variables are terms. If f is a function symbol of arity n and $t_1, \ldots, t_n$ are terms, then $f(t_1 \ldots t_n)$ is a term. For some binary function symbols, that is, function symbols f of arity 2 we make an exception and say that $(t_1 f t_2)$ is a term, that is, we write the function symbol between terms and use brackets around this expression. In mathematics the symbols for addition and multiplication are binary

function symbols. They correspond to the English phrases 'the sum of … and …' and 'the product of … and …'. In mathematics we write + and × between terms, for instance, $a+b$ rather than $+ab$. It is also clear that when we use binary function symbols brackets are required: $(a-b)+c$ and $a-(b+c)$, for instance, are clearly not equivalent.

Unary function symbols correspond to expressions such as 'the father of' in English. The expression 'the father of' has arity 1. It takes a term, for instance, 'Alfred Tarski' and yields a new term 'the father of Alfred Tarski'. An example of a binary function expression is 'the border between … and …'

We also allow function symbols with arity 0. They are individual constants. Since individual constants are 0-place function symbols, they need to be combined with 0 many terms to yield a term, that is, they are terms by themselves.

How many function symbol of a specific arity is in a language will be specified for each language. A language may have no function symbols at all or only individual constants, that is, function symbols of arity 0. All the languages we consider will have only finitely many function symbols.

In our languages we also have predicate symbols. As is the case with function symbols, each of the languages we consider will feature only a finite number of predicate symbols. Predicate symbols also come with an arity. If P is a predicate symbol with arity n and $t_1, \ldots, t_n$ are terms, then $Pt_1 \ldots t_n$ is an atomic formula. An exception is the binary predicate symbol = for identity, which is written between terms. We use = as a predicate symbol in our formal languages, but also as the usual identity symbol in the language we use. The identity symbol will be a predicate symbol in all of the languages we consider.

As in the case of function symbols, the predicate symbols will be specified for every language we discuss. In many elementary logic textbooks such as (Halbach 2010) a single language with infinitely many predicate symbols of arbitrary arity is considered. This ascertains that we never run out of predicate symbols when formalizing argument in natural language. But once we consider a specific set of assumptions, that is, a theory, this will often be formulated with very few predicate symbols. Set theory, for instance, in which all (or perhaps almost all) of mathematics can be carried out, has only one binary predicate symbol.

All atomic formulæ are formulæ. If φ, ψ are formulæ and x is a variable, then $\neg\varphi$, $(\varphi \to \psi)$, and $\forall x\varphi$ are formulæ.We use $(\varphi \wedge \psi)$ as a abbreviation for $\neg(\varphi \to \neg\psi)$, $(\varphi \vee \psi)$ as an abbreviation for $(\neg\varphi \to \psi)$, $(\varphi \leftrightarrow \psi)$ as an abbreviation for $\neg((\varphi \to \psi) \to \neg(\psi \to \varphi))$, and finally $\exists x\varphi$ as an abbreviation for $\neg\forall x\neg\varphi$. These

abbreviations are metalinguistic abbreviations. That is, we do not introduce new symbols $\wedge$, $\vee$, $\leftrightarrow$, and $\exists$ into our formal object language; rather, we use in our informal metalanguage $\wedge$, $\vee$, $\leftrightarrow$, and $\exists$ in order to save some space and make this book more readable.

The notion of a free and bound occurrences is defined in the usual way: In an atomic formula all occurrences of variables are free. All occurrences of variables that are free in φ and ψ are also free in $\neg\varphi$ and $(\varphi \to \psi)$. All occurrences of a variable y in φ are also free in $\forall x\varphi$ iff y is not x. formulæ without any free occurrence of a variable are sentences.

We use rules for omitting brackets. Our formulæ are always the formulæ with all the brackets. If we omit brackets, we obtain abbreviations of formulæ. The rules for omitting brackets are specific to the particular occurrence of a formula; so they should be formulated for occurrences of formulæ. For instance we can omit the outer brackets in an occurrence of a formula $(\varphi \to \psi)$, if $(\psi \to \psi)$ does not occur within another formula. That is we can drop the 'outermost' set of brackets. The expression $(\varphi \wedge \psi \wedge \chi)$ is short for $((\varphi \wedge \psi) \wedge \chi))$. A similar rule applies to $\vee$. Of course, here we are already suing the abbreviations $\wedge$ and $\vee$. Moreover, in abbreviated formulæ with $\wedge$ and $\vee$, $\wedge$ and $\vee$ bind more strongly than $\to$ and $\leftrightarrow$. Thus $\varphi \wedge \psi \to \chi$, for instance, is short for $((\varphi \wedge \psi) \to \chi)$.

Above we have already used x and y as metavariables for variables in the object language. This means that x could be any of the variables v_0, v_1, v_2,... As in the case of the metavariables for formulæ, we employed metavariables for variables on order to make our definitions sufficiently general. For instance, we stipulated that $\forall x\varphi$ is a formula if x is a variable and φ is a formula. It would not suffice to say that $\forall \mathsf{v}_0\varphi$ is a formula if φ is, because this would not imply that $\forall \mathsf{v}_1\varphi$ is a formula. In what follows we continue to use x, y, and so on as metavariables for variables.

Writing 'v_0', 'v_1', and so on makes the notation somewhat cluttered. To avoid the indices we write x for v_0, y for v_1, and z for v_2. So sans-serif letters stand for specific variables, while letters in italics are metavariables for variables. In logic it usually does not matter which variable is used, as long as the variables employed in a formula are pairwise distinct. But in a theory of expressions and proving results such as the diagonal theorem, we have to be very specific and will use sans-serif letters, that is, specific variables.

2.2 Logical Calculi

There are numerous way to generate the logically valid sentences of these languages. In many textbooks such as (Halbach 2010) some variant of the system of Natural Deduction is used, while tableaux systems are used in others. There are also sequent and axiomatic systems, and calculi less likely to be seen in introductory texts for philosophers. For most parts of this book it does not matter which logical calculus is used. However, the reader should be familiar with at least one such calculus. When we prove a sentence of our formal language, we do not provide a proof in some specific calculus, but we outline the crucial steps that should allow the reader versed in a particular calculus to convert our informal proofs into a fully formal proof in his or her preferred logical calculus.

For many logical calculi an infinite stock of individual constants is needed for the quantifiers rules. However, we consider languages with only finitely many constants (or 0-place function symbols). We can simply add constants for the sake of proofs, so that these constants never appear in the premisses and the conclusion of the proof, but only in the intermediate steps. Alternatively, free variables can be used instead of the constants.

For the identity symbol = the chosen logical calculus should contain suitable rules and/or axioms. For function symbols usually additional specific axioms or rules are not required, as long as the rules for identity apply. However, compared to calculi for languages without function symbols, the rules and axioms must cover not only free variables and/or constants as terms but also complex terms involving function symbols. In section 2.3 we show that function symbols are dispensable.

When a sentence φ is provable in some fixed logical calculus, for instance, in Natural Deduction, from sentences in the set Γ of sentences, we write $\Gamma \vdash \varphi$. The sentences φ such that $\Gamma \vdash \varphi$ are the theorems of the theory Γ. The set Γ can be empty. In this case we write $\vdash \varphi$. Occasionally we will write $\Gamma \vdash \varphi$ for a formula φ containing free variables. We can take this to be shorthand for the claim that the universal closure of φ is provable from Γ. The universal closure of a formula is the result of prefixing universal quantifiers for all free variables in φ to φ. For instance, if $\varphi(v_0, v_3)$ has free occurrences of the variables v_0 and v_3 but of no other variables, then $\forall v_0 \forall v_3 \varphi(v_0, v_3)$ is the universal closure of $\varphi(v_0, v_3)$.

Sentence φ and ψ are logically equivalent iff $\vdash \varphi \leftrightarrow \psi$. A theory Γ is inconsistent iff $\Gamma \vdash \varphi$ for all sentences φ of the language. It is not hard to show that Γ is

inconsistent iff there is a sentence φ such that $\Gamma \vdash \varphi$ and $\Gamma \vdash \neg\varphi$.

We use the Deduction theorem without mentioning it explicitly, that is, we assume the following:

$$\Gamma \cup \{\varphi\} \vdash \psi \text{ iff } \Gamma \vdash \varphi \to \psi$$

The proof of the Deduction theorem varies from calculus to calculus. For Natural Deduction it is trivial: ψ is provable with the undischarged assumption φ iff $\varphi \to \psi$ is provable without φ as undischarged assumption. This follows from the introduction and elimination rules for the material conditional $\to$.

We will often talk about theories. A theory is just a set Γ of sentence. We say that a theory is given by certain axioms and rules iff exactly all sentences logically derivable from Γ can be obtained from these axioms and with these rules. Our theories are always classical first-order theories. They can always be obtained from a certain set of axioms and the axioms and/or rules of first-order predicate logic (depending on our chosen logical calculus).

2.3 Function Symbols

Since many introductory logic textbooks do not cover function symbols, we sketch some basics about function symbols.

Unary functions (not function *symbols*) can be understood as binary relations with certain properties. We write Rde to express that d and e stand in the relation R. If relations are conceived as sets of ordered pairs, this means that $\langle d, e\rangle \in R$ (see, for instance, Halbach 2010). A binary relation is a unary function on a set S iff is satisfies the following two properties:

1. For all $d \in S$ there is an e such that Rde.
2. If $d \in S$, Rde_1, and Rde_2, then $e_1 = e_2$.

We call 1. the existence condition and 2. the uniqueness condition. The relation 'd has e as a father' is a function on the set of persons. The two conditions are satisfied, because 1. every person has a father and, 2. every person has at most one father. Together the two conditions express that every person has exactly one father.

Binary functions are defined in an analogous way as special ternary relations: A ternary relation is a binary function on a set S iff the following two conditions are satisfied:

1. For all $d_1, d_2 \in S$ there is an e such that Rd_1d_2e.

2. If $d_1, d_2 \in S$, $Rd_1d_2e_1$, and $Rd_1d_2e_2$, then $e_1 = e_2$.

In mathematics addition is a binary function on the natural numbers. For any numbers d_1 and d_2 there is exactly one number that is the sum of d_1 and d_2. Similarly, if d_1 and d_2 are strings of symbols, then there is exactly one string of symbols that is the result of writing d_2 after d_1 or, in other words, of concatenating d_1 with d_2.

This is generalized in the obvious way to functions of arbitary arity: A $n+1$-ary relation is a n-ary function on a set S iff the following two conditions are satisfied:

1. For all $d_1, \ldots, d_n \in S$ there is an e such that $Rd_1 \ldots d_ne$.

2. If $d_1, \ldots, d_n \in S$, $Rd_1, \ldots, d_ne_1$, and $Rd_1, \ldots, d_ne_2$, then $e_1 = e_2$.

The interpretations or semantic values of an n-ary function symbol will be an n-ary function on the domain of the model (or structure).

When we formulate our theories about expressions later, we will use function symbols, for instance, for concatenation of expressions, that is, for the result appending an expression to an expression. We hope that it makes our formulæ easier to comprehend. We could dispense with the function symbols and use suitable relation symbols instead. How functions symbols can be eliminated using suitable predicate symbols is described in many textbooks, including (Boolos, Burgess, and Jeffrey 2007). Here we show how to eliminate function symbols or arity 1 or higher with predicate symbols. We do not eliminate individual constants, although this can be done as well.

Before describing the method of elimination in more detail, we give a sketch of the strategy. Assume we have a formula of the form $fab = c$ where a, b, and c are individual constants and f is a binary function symbol. We introduce a new ternary predicate symbol P that corresponds to the function symbol f. The sentence $fab = c$ will be replaced everywhere with $Pabc$. But $Pabc$ carries less information than fab, because $Pabc$ is consistent with $Pabd \wedge \neg c = d$, while $fab = c$ and $fab = d$ imply $c = d$ by the logical rules for identity. Thus we will have to add an additional assumption about P to our theory, namely $\forall x \forall y \forall v_2 \forall v_3 (Pxyv_2 \wedge Pxyv_3 \rightarrow v_2 = v_3)$. This expresses the uniqueness condition above. We also have to add an axiom corresponding to the existence condition.

We can prove $\forall x \exists y f x = y$ in our logical calculus, that is, in logic (with identity), because we have $\forall x f x = f x$ and thus, by existential weakening $\forall x \exists y f x = y$. However, the corresponding sentence $\forall x \exists y P x y$ with the predicate symbol is not logically true. Therefore we add $\forall x \exists y P x y$ as an additional axiom. After replacing the function symbol f with the predicate symbol P and adding the uniqueness and existence axioms to our theory, the new theory with the predicate symbol P and without the function symbol f will prove exactly the translations of the theorems of the original theory with f into our new theory with P.

Of course there remain some details that need to be filled in. In particular, function symbols may not only occur in formulæ of the form $f t_1 t_2 = t_3$ where t_1 and t_2 are constants or variables and t_3 is a term without an occurrence of f. Function symbols can be iterated. For instance $f x f x f x y$ is a term if f is a binary function symbol and x and y variables. Moreover the function symbol can occur not only in atomic formula with identity, but also in atomic formulæ built form other predicate symbols. We sketch the method of elimination only for a binary function symbol f. The method works for function symbols of another arity in an analogous way. If more than one function symbol is present in the language, they can be eliminated by the same method.

First we show that any formula φ in the language with the binary function symbol f is logically equivalent to a formula in which all occurrences of f in atomic formulæ are in atomic formulæ of the form $f t_1 t_2 = t_3$ where each of t_1, t_2, and t_3 is either a constant or a variable. This is shown by induction on the complexity of φ. Starting from the left of the formula there must be a first occurrence of f that is part of a term $f t_1 t_2$ with t_1 and t_2 a variable or constant that is not on the left hand side of an identity symbol (such an occurrence will be left alone). Replace $f t_1 t_2$ with the first variable x not occurring in φ to obtain $\varphi(x)$ and define φ' as $\exists x (f t_1 t_2 = x \wedge \varphi(x))$. φ' is logically equivalent to φ. Iterate this procedure, always using the first variable not already used. The resulting formula $\varphi'\cdots'$ is called φ^*. In φ^* there are only occurrences of f left that are in atomic formulæ of the form $f t_1 t_2 = \ldots$ with t_1 and t_2 variables or constants. Finally we define φ^P as the result of replacing all atomic subformulæ $f t_1 t_2 = t_3$ of φ^* with $P t_1 t_2 t_3$.

Let a theory Γ and a sentence φ in the language with f but without P be given. Let Γ^P be the result of replacing all sentences γ in Γ with γ^* and adding the two axioms

1. $\forall x \forall y \exists z P x y z$

2. $\forall x \forall y \forall v_2 \forall v_3 (Pxyv_2 \wedge Pxyv_3 \rightarrow v_2 = v_3)$

Then it is not hard to prove the following equivalence:

$$\Gamma \vdash \varphi \text{ iff } \Gamma^* \vdash \varphi^*$$

A detailed proof depends on the logical calculus chosen. We hope to have given enough detail that would allow the reader to carry out the proof for his or her chosen calculus.

Thus we can dispense with the use of f and use a predicate symbol instead. As the elimination procedure indicates, the formulation with the predicate symbol tends to produce less legible formulæ.