LECTURES IN LOGIC AND SET THEORY

Volume 1: Mathematical Logic

GEORGE TOURLAKIS
York University

CAMBRIDGE
UNIVERSITY PRESS

PUBLISHED BY THE PRESS SYNDICATE OF THE UNIVERSITY OF CAMBRIDGE
The Pitt Building, Trumpington Street, Cambridge, United Kingdom

CAMBRIDGE UNIVERSITY PRESS
The Edinburgh Building, Cambridge CB2 2RU, UK
40 West 20th Street, New York, NY 10011-4211, USA
477 Williamstown Road, Port Melbourne, VIC 3207, Australia
Ruiz de Alarcón 13, 28014 Madrid, Spain
Dock House, The Waterfront, Cape Town 8001, South Africa

http://www.cambridge.org

First published 2003

Printed in the United Kingdom at the University Press, Cambridge

Typeface Times 10/13 pt. *System* LaTeX 2_ε [TB]

A catalog record for this book is available from the British Library.

Library of Congress Cataloging in Publication Data

Tourlakis, George J.
Lectures in logic and set theory / George Tourlakis.
p. cm. – (Cambridge studies in advanced mathematics)
Includes bibliographical references and index.
Contents: v. 1. Mathematical logic – v. 2. Set theory.
ISBN 0-521-75373-2 (v. 1) – ISBN 0-521-75374-0 (v. 2)
1. Logic, Symbolic and mathematical. 2. Set theory. I. Title. II. Series.
QA9.2 .T68 2003
511.3 – dc21 2002073308

ISBN 0 521 75373 2 hardback

Contents

I

Basic Logic

Logic is the science of reasoning. Mathematical logic applies to mathematical reasoning – the art and science of writing down *deductions*. This volume is about the *form*, *meaning*, *use*, and *limitations* of logical deductions, also called *proofs*. While the user of mathematical logic will practise the various proof techniques with a view of applying them in everyday mathematical practice, the student of the subject will also want to know about the power and limitations of the deductive apparatus. We will find that there are some inherent limitations in the quest to discover truth by purely *formal* – that is, *syntactic* – techniques. In the process we will also discover a close affinity between formal proofs and *computations* that persists all the way up to and including issues of limitations: Not only is there a remarkable similarity between the types of respective limitations (computations vs. uncomputable functions, and proofs vs. unprovable, but "true", sentences), but, in a way, you cannot have one type of limitation without having the other.

The modern use of the term mathematical logic encompasses (at least) the areas of proof theory (it studies the structure, properties, and limitations of proofs), model theory (it studies the interplay between syntax and meaning – or semantics – by looking at the algebraic structures where formal languages are interpreted), recursion theory (or computability, which studies the properties and limitations of algorithmic processes), and set theory. The fact that the last-mentioned will totally occupy our attention in volume 2 is reflected in the prominence of the term in the title of these lectures. It also reflects a tendency, even today, to think of set theory as a branch in its own right, rather than as an "area" under a wider umbrella.

Volume 1 is a brief study of the other three areas of logic[†] mentioned above. This is the point where an author usually apologizes for what has been omitted, blaming space or scope (or competence) limitations. Let me start by outlining what is included: "Standard" phenomena such as *completeness*, *compactness* and its startling application to analysis, *incompleteness* or *unprovability* (including a *complete* proof of the *second incompleteness theorem*), and a fair amount of *recursion theory* are thoroughly discussed. Recursion theory, or *computability*, is of interest to a wide range of audiences, including students with main areas of study such as computer science, philosophy, and, of course, mathematical logic. It studies among other things the phenomenon of uncomputability, which is closely related to that of unprovability, as we see in Section I.9.

Among the topics that I have deliberately left out are certain algebraic techniques in model theory (such as the method of *ultrapowers*), formal interpretations of one theory into another,[‡] the introduction of "other" logics (*modal*, *higher order*, *intuitionistic*, etc.), and several topics in recursion theory (oracle computability, Turing reducibility, recursive operators, degrees, Post's theorem in the arithmetic hierarchy, the analytic hierarchy, etc.) – but then, the decision to stop writing within 300 or so pages was firm. On the other hand, the topics included here form a synergistic whole in that I have (largely) included at every stage material that is prerequisite to what follows. The absence of a section on *propositional calculus* is deliberate, as it does not in my opinion further the understanding of logic in any substantial way, while it delays one's plunging into what really matters. To compensate, I include all tautologies as "propositional" (or Boolean) logical axioms and present a mini-course on propositional calculus in the exercises of this chapter (I.26–I.41, pp. 193–195), including the completeness and compactness of the calculus.

It is inevitable that the language of sets intrudes in this chapter (as it indeed does in all mathematics) and, more importantly, some of the results of (informal) set theory are needed here (especially in our proofs of the completeness and compactness *meta*theorems). Conversely, *formal* set theory of volume 2 needs some of the results developed here. This "chicken or egg" phenomenon is often called "bootstrapping" (not to be confused with "circularity" – which it is not[§]), the term suggesting one pulling oneself up by one's bootstraps.[¶]

[†] I trust that the reader will not object to my dropping the qualifier "mathematical" from now on.

[‡] Although this topic is included in volume 2 (Chapter I), since it is employed in the relative consistency techniques applied there.

[§] Only informal, or naïve, set theory notation and results are needed in Chapter I at the meta-level, i.e, outside the formal system that logic is.

[¶] I am told that Baron Münchhausen was the first one to apply this technique, with success.

This is a good place to outline how our story will unfold: First, our objective is to *formalize* the rules of reasoning in general – as these apply to all mathematics – and develop their properties. In particular, we will study the interaction between formalized rules and their "intended meaning" (semantics), as well as the limitations of these formalized rules: That is, how good (= potent) are they for capturing the informal notions of truth?

Secondly, once we have acquired these tools of formalized reasoning, we start behaving (mostly[†]) as *users* of formal logic so that we can discover important theorems of two important mathematical theories: *Peano arithmetic* (Chapter II) and *set theory* (volume 2).

By *formalization* (of logic) we understand the faithful representation or simulation of the "reasoning processes" of mathematics *in general* (*pure* logic), or of a particular mathematical theory (*applied* logic: e.g., Peano arithmetic), within an activity that – in principle – is driven exclusively by the *form* or syntax of mathematical statements, totally ignoring their meaning.

We build, describe, and study the properties of this *artificial replica* of the reasoning processes – the formal theory – within "everyday mathematics" (also called "informal" or "real" mathematics), using the usual abundance of mathematical symbolism, notions, and techniques available to us, augmented by the descriptive power of English (or Greek, or French, or German, or Russian, or . . . , as particular circumstances or geography might dictate). This milieu within which we build, pursue, and study our theories is often called the *metatheory*, or more generally, *metamathematics*. The language we speak while at it, this mélange of mathematics and "natural language", is the *metalanguage*.

Formalization turns mathematical theories into mathematical objects that we can study. For example, such study may include interesting questions such as "is the continuum hypothesis provable from the axioms of set theory?" or "can we prove the consistency of (axiomatic) Peano arithmetic within Peano arithmetic?"[‡] This is analogous to building a "model airplane", a replica of the real thing, with a view of studying *through the replica* the properties, power, and limitations of the real thing.

But one can also use the formal theory to *generate theorems*, i.e., discover "truths" in the real domain by simply "running" the simulation that this theory-replica is.[§] Running the simulation "by hand" (rather than using the program

[†] Some tasks in Chapter II of this volume, and some others in volume 2, will be to treat the "theory" at hand as an object of study rather than using it, as a machine, to crank out theorems.

[‡] By the way, the answer to both these questions is "no" (Cohen (1963) for the first, Gödel (1938) for the second).

[§] The analogy implied in the terminology "running the simulation" is apt. For formal theories such as set theory and Peano arithmetic we can build within real mathematics a so-called "provability

of the previous footnote) means that you are acting as a "user" of the formal system, a formalist, proving theorems through it. It turns out that once you get the hang of it, it is easier and safer to reason formally than to do so informally. The latter mode often mixes syntax and semantics (meaning), and there is always the danger that the "user" may assign incorrect (i.e., convenient, but not *general*) meanings to the symbols that he[†] manipulates, a phenomenon that has distressed many a mathematics or computer science instructor.

"Formalism for the user" is hardly a revolutionary slogan. It was advocated by Hilbert, the founder of formalism, partly as a means of – as he believed[‡] – formulating mathematical theories in a manner that allows one to check them (i.e., run "diagnostic tests" on them) for freedom from contradiction,[§] but also as *the right way to "do" mathematics*. By this proposal he hoped to salvage mathematics itself, which, Hilbert felt, was about to be destroyed by the Brouwer school of intuitionist thought. In a way, his program could bridge the gap between the classical and the intuitionist camps, and there is some evidence that Heyting (an influential intuitionist and contemporary of Hilbert) thought that such a *rapprochement* was possible. After all, since meaning is irrelevant to a formalist, then all that he is doing (in a proof) is shuffling finite sequences of symbols, never having to handle or argue about infinite objects – a good thing, as far as an intuitionist is concerned.[¶]

predicate", that is, a relation $P(y, x)$ which is true of two natural numbers y and x just in case y *codes* a proof of the formula *coded* by x. It turns out that $P(y, x)$ has so simple a structure that it is programmable, say in the C programming language. But then we can write a program (also in C) as follows: "Systematically generate all the pairs of numbers (y, x). For each pair generated, if $P(y, x)$ holds, then print the formula coded by x". Letting this process run for ever, we obtain a listing of *all* the theorems of Peano arithmetic or set theory! This fact does not induce any insomnia in mathematicians, since this is an extremely impractical way to obtain theorems. By the way, we will see in Chapter II that either set theory or Peano arithmetic is sufficiently strong to *formally* express a provability predicate, and this leads to the incompletableness phenomenon.

[†] In this volume, the terms "he", "his", "him", and their derivatives are by definition gender-neutral.

[‡] This belief was unfounded, as Gödel's incompleteness theorems showed.

[§] Hilbert's *metatheory* – that is, the "world" or "lab" *outside the theory*, where the replica is actually manufactured – was *finitary*. Thus – Hilbert advocated – all this theory building and theory checking ought to be effected by *finitary means*. This ingredient of his "program" was consistent with peaceful coexistence with the intuitionists. And, alas, this ingredient was the one that – as some writers put it – destroyed Hilbert's program to found mathematics on his version of formalism. Gödel's incompleteness theorems showed that a finitary metatheory is not up to the task.

[¶] True, a formalist applies classical logic, while an intuitionist applies a different logic where, for example, double negation is not removable. Yet, unlike a Platonist, a Hilbert-style formalist does not believe – or he does not have to disclose to his intuitionist friends that he might believe – that infinite sets exist *in the metatheory*, as his tools are just finite symbol sequences. To appreciate the tension here, consider this anecdote: It is said that when Kronecker – the father of intuitionism – was informed of Lindemann's proof (1882) that π is transcendental, while he granted that this was an interesting result, he also dismissed it, suggesting that "π" – whose decimal expansion is, of

In support of the "formalism for the user" position we must definitely mention the premier paradigm, Bourbaki's monumental work (1966a), which is a formalization of a huge chunk of mathematics, including set theory, algebra, topology, and theory of integration. This work is strictly for the *user* of mathematics, not for the metamathematician who *studies* formal theories. Yet, it is fully formalized, true to the spirit of Hilbert, and it comes in a self-contained package, including a "Chapter 0" on formal logic.

More recently, the proposal to employ formal reasoning as a tool has been gaining support in a number of computer science undergraduate curricula, where logic and discrete mathematics are taught in a formalized setting, starting with a rigorous course in the two logical calculi (propositional and predicate), emphasizing the point of view of the *user* of logic (and mathematics) – hence with an attendant emphasis on "calculating" (i.e., writing and annotating formal) proofs. Pioneering works in this domain are the undergraduate text (1994) and the paper (1995) of Gries and Schneider.

I.1. First Order Languages

In the most abstract (therefore simplest) manner of describing it, a *formalized mathematical theory* consists of the following sets of things: A set of basic or primitive symbols, $\mathscr{S}$, used to build *symbol sequences* (also called strings, or expressions, or words) "over $\mathscr{S}$". A set of strings, **Wff**, over $\mathscr{S}$, called the *formulas* of the theory. Finally, a *subset* of **Wff**, called **Thm**, the set of *theorems* of the theory.†

Well, this is the *extension* of a theory, that is, the explicit set of objects in it. How is a theory "given"?

In most cases of interest to the mathematician it is given by $\mathscr{S}$ and two sets of simple *rules*: formula-building rules and theorem-building rules. Rules from the first set allow us to build, or *generate*, **Wff** from $\mathscr{S}$. The rules of the second set generate **Thm** from **Wff**. In short (e.g., Bourbaki (1966b)), *a theory consists of an alphabet of primitive symbols, some* rules *used to generate the "language of the theory"* (*meaning, essentially,* **Wff**) *from these symbols, and some* additional *rules used to generate the theorems.* We expand on this below:

course, infinite but not periodic – "does not exist" (see Wilder (1963, p. 193)). We are not to propound the tenets of intuitionism here, but it is fair to state that infinite sets *are* possible in intuitionistic mathematics as this has later evolved in the hands of Brouwer and his Amsterdam "school". However, such sets must be (like all sets of intuitionistic mathematics) *finitely generated* – just as our formal languages and the set of theorems are (the latter *provided* our axioms are too) – in a sense that may be familiar to some readers who have had a course in "automata and language theory". See Wilder (1963, p. 234)

† For a less abstract, but more detailed view of theories see p. 38.

I.1.1 Remark . What is a "rule"? We run the danger of becoming circular or too pedantic if we overdefine this notion. Intuitively, the rules we have in mind are string manipulation rules, that is, "black boxes" (or functions) that receive string inputs and respond with string outputs. For example, a well-known theorem-building rule receives as input a formula and a variable, and returns (essentially) the string composed of the symbol $\forall$, immediately followed by the variable and, in turn, immediately followed by the formula.[†] □

(1) First off, the (*first order*) *formal language*, L, where the theory is "spoken",[‡] is a triple ($\mathscr{V}$, **Term**, **Wff**), that is, it has three important components, each of them a set.

$\mathscr{V}$ is the *alphabet* or vocabulary of the language. It is the collection of the *basic* syntactic "bricks" (symbols) that we use to form *expressions* that are *terms* (members of **Term**) or *formulas* (members of **Wff**). We will ensure that the processes that build terms or formulas, using the basic building blocks in $\mathscr{V}$, are intuitively *algorithmic* or "mechanical".

Terms will formally codify "objects", while formulas will formally codify "statements" about objects.

(2) *Reasoning* in the theory will be the process of discovering *true statements* about objects – that is, *theorems*. This discovery journey begins with certain formulas which codify statements that we take for granted (i.e., we accept without "proof" as "basic truths"). Such formulas are the *axioms*. There are two types of axioms:

Special or *nonlogical* axioms are to describe specific aspects of any specific theory that we might be building. For example, "$x + 1 \neq 0$" is a special axiom that contributes towards the characterization of number theory over the natural numbers, $\mathbb{N}$.

The other kind of axiom will be found in *all* theories. It is the kind that is "universally valid", that is, *not* theory-specific (for example, "$x = x$" is such a "universal truth"). For that reason this type of axiom will be called *logical*.

(3) Finally, we will need *rules* for reasoning, actually called *rules of inference*. These are rules that allow us to deduce, or derive, a true statement from other statements that we have already established as being true.[§] These rules will be chosen to be oblivious to meaning, being only concerned with

[†] This rule is usually called "generalization".

[‡] We will soon say what makes a language "first order".

[§] The generous use of the term "true" here is only meant for motivation. "Provable" or "deducible" (formula), or "theorem", will be the technically precise terminology that we will soon define to replace the term "true statement".

form. They will apply to statement "configurations" of certain *recognizable forms* and will produce (derive) new statements of some *corresponding recognizable forms* (See Remark I.1.1).

I.1.2 Remark. We may think of axioms of either logical or nonlogical type as special cases of rules, that is, rules that receive *no* input in order to produce an output. In this manner item (2) above is subsumed by item (3), and thus we are faithful to our abstract definition of theory where axioms were not mentioned.

An example, outside mathematics, of an inputless rule is the rule invoked when you type **date** on your computer keyboard. This rule receives no input, and outputs on your screen the current date. □

We next look carefully into (first order) formal languages.

There are two parts in each first order alphabet. The first, the collection of the *logical symbols*, is *common to all first order languages* regardless of which theory is "spoken" in them. We describe this part immediately below.

Logical Symbols

LS.1. *Object or individual variables*. An *object variable* is any one *symbol* out of the non-ending sequence $v_0, v_1, v_2, \ldots$. *In practice* – whether we are using logic as a tool or as an object of study – we agree to be sloppy with notation and use, *generically*, x, y, z, u, v, w with or without subscripts or primes as *names* of object variables.† This is just a matter of notational convenience. We allow ourselves to write, say, z instead of, say, $v_{1200000000560000009}$. Object variables (intuitively) "vary over" (i.e., are allowed to take values that are) the objects that the theory studies (numbers, sets, atoms, lines, points, etc., as the case may be).

LS.2. *The Boolean or propositional connectives.* These are the symbols "$\neg$" and "$\vee$".‡ They are pronounced *not* and *or* respectively.

LS.3. *The existential quantifier*, that is, the symbol "$\exists$", pronounced *exists* or *for some*.

LS.4. *Brackets*, that is, "(" and ")".

LS.5. *The equality predicate*. This is the symbol "$=$", which we use to indicate that objects are "equal". It is pronounced *equals*.

† Conventions such as this one are essentially agreements – effected in the metatheory – on how to be sloppy and get away with it. They are offered in the interest of user-friendliness.

‡ The quotes are *not* part of the symbol. They serve to indicate clearly here, in particular in the case of "$\vee$", what is part of the symbol and what is not (the following period).

The logical symbols will have a fixed interpretation. In particular, "=" will always be expected to mean *equals*.

The theory-specific part of the alphabet is not fixed, but varies from theory to theory. For example, in set theory we just add the nonlogical (or special) symbols, $\in$ and U. The first is a special *predicate symbol* (or just predicate) of *arity* 2, the second is a predicate symbol of arity 1.[†]

In number theory we adopt instead the special symbols S (intended meaning: successor, or "$+1$" function), $+$, $\times$, 0, $<$, and (sometimes) a symbol for the exponentiation operation (function) a^b. The first three are *function symbols* of arities 1, 2, and 2 respectively. 0 is a *constant symbol*, $<$ a predicate of arity 2, and whatever symbol we might introduce to denote a^b would have arity 2.

The following list gives the general picture.

Nonlogical Symbols

NLS.1. A (possibly empty) set of symbols for *constants*. We normally use the metasymbols[‡] a, b, c, d, e, with or without subscripts or primes, to stand for constants unless we have in mind some alternative "standard" formal notation in specific theories (e.g., $\emptyset$, 0, ω).

NLS.2. A (possibly empty) set of symbols for *predicate symbols* or *relation symbols* for each possible "arity" $n > 0$. We normally use P, Q, R generically, with or without primes or subscripts, to stand for predicate symbols. Note that $=$ is in the logical camp. Also note that theory-specific formal symbols are possible for predicates, e.g., $<, \in$.

NLS.3. Finally, a (possibly empty) set of symbols for *functions* for each possible "arity" $n > 0$. We normally use f, g, h, generically, with or without primes or subscripts, to stand for function symbols. Note that theory-specific formal symbols are possible for functions, e.g., $+, \times$.

I.1.3 Remark. (1) We have the option of assuming that each of the *logical* symbols that we named in **LS.1**–**LS.5** have no further "structure" and that the symbols are, ontologically, *identical to their names*, that is, they are just these exact signs drawn on paper (or on any equivalent display medium).

In this case, changing the symbols, say, $\neg$ and $\exists$ to $\sim$ and **E** respectively results in a "different" logic, but one that is, trivially, "isomorphic" to the one

[†] "Arity" is a term mathematicians have made up. It is derived from "ary" of "un*ary*", "bin*ary*", etc. It denotes the number of arguments needed by a symbol according to the dictates of correct syntax. Function and predicate symbols need arguments.

[‡] *Meta*symbols are *informal* (i.e., outside the formal language) symbols that we use within "everyday" or "real" mathematics – the *meta*theory – in order to describe, as we are doing here, the formal language.

we are describing: Anything that we may do in, or say about, one logic trivially translates to an equivalent activity in, or utterance about, the other as long as we systematically carry out the translations of all occurrences of $\neg$ and $\exists$ to $\sim$ and **E** respectively (or vice versa).

An alternative point of view is that the symbol names are *not* the same as (identical with) the symbols they are naming. Thus, for example, "$\neg$" names the connective we pronounce **not**, but we do not know (or care) exactly what the nature of this connective is (we only care about how it behaves). Thus, the name "$\neg$" becomes just a typographical expedient and may be replaced by other names that name the same object, **not**.

This point of view gives one flexibility in, for example, deciding how the variable symbols are "implemented". It often is convenient to think that the entire sequence of variable symbols was built from just two symbols, say, "v" and "|".[†] One way to do this is by saying that v_i is a name for the symbol sequence[‡]

$$\text{“}v\underbrace{|\ldots|}_{i\text{|'s}}\text{”}$$

Or, preferably – see (2) below – v_i might be a name for the symbol sequence

$$\text{“}v\underbrace{|\ldots|}_{i\text{|'s}}v\text{”}$$

Regardless of option, v_i and v_j will name distinct objects if $i \neq j$.

This is *not* the case for the *meta*variables ("abbreviated informal names") x, y, z, u, v, w. *Unless we say so explicitly otherwise, x and y* may *name the same formal variable, say,* v_{131}.

We will mostly abuse language and deliberately confuse names with the symbols they name. For example, we will say, e.g., "let v_{1007} *be* an object variable . . . " rather than "let v_{1007} *name* an object variable . . . ", thus *appearing* to favour option one.

(2) *Any two symbols* included in the alphabet are *distinct*. Moreover, if any of them are built from simpler "sub-symbols" – e.g., $v_0, v_1, v_2, \ldots$ might *really name* the strings vv, $v|v$, $v||v$, . . . – then none of them is a *substring* (or *subexpression*) of any other.[§]

[†] We intend these two symbols to be identical to their names. No philosophical or other purpose will be served by allowing "more indirection" here (such as "v names u, which actually names w, which actually is . . . ").

[‡] Not including the quotes.

[§] What we have stated under (2) are *requirements*, not metatheorems! That is, they are nothing of the sort that we can *prove* about our formal language within everyday mathematics.

(3) A formal language, just like a "natural" language (such as English or Greek), is "alive" and evolving. The particular type of evolution we have in mind is the one effected by *formal definitions*. Such definitions continually *add* nonlogical symbols to the language.†

Thus, when we say that, e.g., "$\in$ and U are the only nonlogical symbols of set theory", we are telling a small white lie. More accurately, we ought to have said that "$\in$ and U are the only 'primitive' nonlogical symbols of set theory", for we will add loads of other symbols such as $\cup$, ω, $\emptyset$, $\subset$, $\subseteq$.

This evolution affects the (formal) language of *any* theory, not just set theory. □

Wait a minute! If formal set theory is "the foundation of all mathematics", and if, ostensibly, this chapter on logic assists us to found set theory itself, then how come we are employing *natural numbers* like 1200000000560000009 as *subscripts* in the *names* of object variables? How is it permissible to already talk about "*sets* of symbols" when we are about to *found* a theory of *sets* formally? Surely we do not "have"‡ any of these "items" yet, do we?

First off, the presence of subscripts such as 1200000000560000009 in

$$v_{1200000000560000009}$$

is a non-issue. One way to interpret what has been said in the definition is to view the various v_i as abbreviated names of the real thing, the latter being strings that employ the symbols v and | as in Remark I.1.3. In this connection saying that v_i is "implemented" as

$$v\underbrace{|\ldots|}_{i\text{|'s}}v \tag{1}$$

especially the use of "i" above, is only illustrative, thus totally superfluous. We can say instead that strings of type (1) *are* the variables which we define as follows *without* the help of the "natural number i" (this is a variation of how this is done in Bourbaki (1966b) and Hermes (1973)):

An "|-calculation" forms a string like this: Write a "|".§ This is the "current string". Repeat a finite number of times: Add (i.e., concatenate) *one* | immediately to the right of the current string. Write this new string (it is now *the* current string).

† This phenomenon will be studied in some detail in what follows. By the way, any additions are made to the nonlogical side of the alphabet. All the logical symbols have been given, once and for all.

‡ "Do not have" in the sense of having not formally defined – or proved to exist – or both.

§ Without the quotes. These were placed to exclude the punctuation following.

Let us call any string that figures in some |-calculation a "|-string". A variable either *is* the string vv, or is obtained as the concatenation from left to right of v followed by an |-string, followed by v.

All we now need is the ability to generate as many as necessary distinct variables (this is the "non-ending sequence" part of the definition, p. 7): For any two variables we get a new one that is different from either one by forming the string "v, followed by the concatenation of the two |-parts, followed by v". Similarly if we had three, four, ... variables. By the way, two strings of | are distinct iff[†] both occur in the same |-calculation, one, but not both, as the last string.

Another, more direct way to interpret what was said about object variables on p. 7 is to take the definition literally, i.e., to suppose that it speaks about the ontology of the variables.[‡] Namely, the subscript is just a a string of meaningless symbols taken from the list below:

$$0, 1, 2, 3, 4, 5, 6, 7, 8, 9$$

Again we can pretend that we know nothing about natural numbers, and whenever, e.g., we want a variable other than either of v_{123} or v_{321}, we may offer either of v_{123321} or v_{321123} as such a *new* variable.

O.K., so we have *not* used natural numbers in the definition. But we did say "sets" and also "non-ending sequence", implying the presence of *infinite* sets!

As we have already noted, on one hand we have "real mathematics", and on the other hand we have syntactic *replicas* of theories – the formal theories – that we built *within real mathematics*. Having built a formal theory, we can then choose to *use* it (acting like formalists) to generate theorems, the latter being codified as symbol sequences (formulas). Thus, the assertion "axiomatic set theory is the foundation of all mathematics" is just a colloquialism proffered in the metatheory that means that "within axiomatic set theory we can construct the known sets of mathematics, such as the reals $\mathbb{R}$ and the complex numbers $\mathbb{C}$, and moreover we can simulate what we informally do whenever we are working in real or complex analysis, algebra, topology, theory of measure and integration, functional analysis, etc., etc."

There is no circularity here, but simply an empirical boastful observation *in the metatheory* of what our simulator can do. Moreover, our metatheory does

[†] If and only if.

[‡] Why not just say *exactly* what a definition is meant to say rather than leave it up to interpretation? One certainly could, as in Bourbaki (1966b), make the ontology of variables crystal-clear right in the definition. Instead, we have followed the custom of more recent writings and given the definition in a quasi-sloppy manner that leaves the ontology of variables as a matter for speculation. This gives one the excuse to write footnotes like this one and remarks like I.1.3.

have sets and all sorts of other mathematical objects. In principle we can use any among those towards building or discussing the simulator, the formal theory.

Thus, the question is not whether we can use sets, or natural numbers, in our definitions, but whether restrictions apply. For example, can we use infinite sets?

If we are Platonists, then we have available in the metatheory all sorts of sets, including infinite sets, in particular the set of all natural numbers. We can use any of these items, speak about them, etc., as we please, when we are describing or building the formal theory *within our metatheory*.

Now, if we are not Platonists, then our "real" mathematical world is much more restricted. In one extreme, we have *no* infinite sets.†

We can still manage to define our formal language! After all, the "non-ending" sequence of object variables $v_0, v_1, v_2, \ldots$ can be *finitely generated* in at least two different ways, as we have already seen. Thus we can explain (to a true formalist or finitist) that "non-ending sequence" was an unfortunate slip of the tongue, and that we really meant to give a *procedure* of how to generate on demand a *new* object variable, different from whatever ones we may already have.

Two parting comments are in order: One, we have been somewhat selective in the use of the term "metavariable". We have called x, x', y metavariables, but have implied that the v_i are formal variables, even if they are just *names* of formal objects such that we do not know or do not care what they look like. Well, strictly speaking the abbreviations v_i are also metavariables, but they are endowed with a property that the "generic" metavariables like x, y, z' do not have: Distinct v_i names denote distinct object variables (cf. I.1.3).

Two, we should clarify that a formal theory, when *used* (i.e., the simulator is being "run") is a *generator* of strings, *not* a decider or "parser". Thus, it can *generate* any of the following: variables (if these are given by procedures), formulas and terms (to be defined), or theorems (to be defined). *Decision* issues, no matter how trivial, the system is not built to handle. These belong to the metatheory. In particular, the theory does not see whatever numbers or strings (like 12005) may be hidden in a variable name (such as v_{12005}).

Examples of decision questions: Is this string a term or a formula or a variable (finitely generated as above)? All these questions are "easy". They are algorithmically decidable in the metatheory. Or, is this formula a theorem? This is

† A finitist – and don't forget that Hilbert-style metatheory was finitary, ostensibly for political reasons – will let you have as many integers as you like in one serving, as long as the serving is *finite*. If you ask for more, you can have more, but never the set of all integers or an infinite subset thereof.

algorithmically undecidable in the metatheory if it is a question about Peano arithmetic or set theory.

I.1.4 Definition (Terminology about Strings). A symbol sequence or *expression* (or *string*) that is formed by using symbols exclusively out of a given set[†] M is called *a string over the set, or alphabet, M*.

If A and B denote strings (say, over M), then the symbol $A * B$, or more simply AB, denotes the symbol sequence obtained by listing first the symbols of A in the given left to right sequence, immediately followed by the symbols of B in the given left to right sequence. We say that AB is (more properly, denotes or names) the *concatenation* of the strings A and B in that order.

We denote the fact that the strings (named) C and D are *identical sequences* (but we just say that they are *equal*) by writing $C \equiv D$. The symbol $\not\equiv$ denotes the negation of the string equality symbol $\equiv$. Thus, if # and ? are (we do mean "are") symbols from an alphabet, then

$$\#?? \equiv \#?? \quad \text{but} \quad \#? \not\equiv \#??$$

We can also employ $\equiv$ in contexts such as "let $A \equiv$ ##?", where we give the name A to the string ##?.[‡]

In this book the symbol $\equiv$ will be exclusively *used in the metatheory* for equality of strings over some set M.

The symbol λ normally denotes the *empty* string, and we postulate for it the following behaviour:

$$A \equiv A\lambda \equiv \lambda A \qquad \text{for all strings } A$$

We say that A *occurs in* B, or is a *substring of* B, iff there are strings C and D such that $B \equiv CAD$.

For example, "(" occurs four times in the (explicit) string "¬(()∨)((", at *positions* 2, 3, 7, 8. Each time this happens we have an *occurrence* of "(" in "¬(()∨)((".

If $C \equiv \lambda$, we say that A is a *prefix* of B. If moreover $D \not\equiv \lambda$, then we say that A is a *proper prefix* of B. □

[†] A set that supplies symbols to be used in building strings is not special. It is just a set. However, it often has a special name: "alphabet".

[‡] Punctuation such as "." is not part of the string. One often avoids such footnotes by enclosing strings that are explicitly written as symbol sequences inside quotes. For example, if A stands for the string #, one writes $A \equiv$ "#". Note that we must not write "A", unless we mean a string whose only symbol *is* A.

I.1.5 Definition (Terms). The set of *terms*, **Term**, is the *smallest* set of strings over the alphabet $\mathscr{V}$ with the following two properties:

(1) All of the items in **LS.1** or **NLS.1** (x, y, z, a, b, c, etc.) are included.
(2) If f is a function[†] of arity n and $t_1, t_2, \ldots, t_n$ are included, then so is the string "$f t_1 t_2 \ldots t_n$".

The symbols t, s, and u, with or without subscripts or primes, will denote arbitrary terms. Since we are using them in the *metalanguage* to "vary over" terms, we naturally call them metavariables. They also serve – as variables – towards the definition (this one) of the *syntax* of terms. For this reason they are also called *syntactic variables.* □

I.1.6 Remark. (1) We often abuse notation and write $f(t_1, \ldots, t_n)$ instead of $f t_1 \ldots t_n$.

(2) Definition I.1.5 is an *inductive definition*.[‡] It defines a more or less "complicated" term by assuming that we already know what "simpler" terms look like. This is a standard technique employed in real mathematics. We will have the opportunity to say more about such inductive definitions – and their appropriateness – in a  -comment later on.

(3) We relate this particular manner of defining terms to our working definition of a theory (given on p. 6 immediately before Remark I.1.1 in terms of "rules" of formation). Item (2) in I.1.5 essentially says that we build new terms (from old ones) by applying the following *general rule*: Pick an arbitrary function symbol, say f. This has a specific formation rule associated with it that, for the appropriate number, n, of an already existing ordered list of terms, $t_1, \ldots, t_n$, will build the new term consisting of f, immediately followed by the ordered list of the given terms.

To be specific, suppose we are working in the language of number theory. There is a function symbol $+$ available there. The rule associated with $+$ builds the new term $+ts$ for any prior obtained terms t and s. For example, $+v_1 v_{13}$ and $+v_{121} + v_1 v_{13}$ are well-formed terms. We normally write terms of number theory in "infix" notation,[§] i.e., $t + s$, $v_1 + v_{13}$ and $v_{121} + (v_1 + v_{13})$ (note the intrusion of brackets, to indicate sequencing in the application of $+$).

[†] We will omit from now on the qualification "symbol" from terminology such as "function symbol", "constant symbol", "predicate symbol".

[‡] Some mathematicians will absolutely insist that we call this a *recursive* definition and reserve the term "induction" for "induction *proofs*". This is seen to be unwarranted hair splitting if we consider that Bourbaki (1966b) calls *induction proofs* "*démonstrations par récurrence*". We will be less dogmatic: Either name is all right.

[§] Function symbol placed between the arguments.

A by-product of what we have just described is that *the arity of a function symbol f is whatever number of terms the associated rule will require as input.*

(4) A crucial word used in I.1.5 (which recurs in all inductive definitions) is "smallest". It means "least inclusive" (set). For example, we may easily think of a set of strings that satisfies both conditions of the above definition, but which is *not* "smallest" by virtue of having additional elements, such as the string "$\neg\neg($".

Pause. Why is "$\neg\neg($" *not* in the smallest set as defined above, and therefore not a term?

The reader may wish to ponder further on the import of the qualification "smallest" by considering the familiar (similar) example of $\mathbb{N}$, the set of natural numbers. The principle of induction in $\mathbb{N}$ ensures that this set is the *smallest* with the properties:

(i) 0 is included, and
(ii) if n is included, then so is $n + 1$.

By contrast, all of $\mathbb{Z}$ (set of integers), $\mathbb{Q}$ (set of rational numbers), $\mathbb{R}$ (set of real numbers) satisfy (i) and (ii), but they are clearly *not* the "smallest" such. □

I.1.7 Definition (Atomic Formulas). The set of *atomic formulas*, **Af**, contains precisely:

(1) The strings $t = s$ for every possible choice of terms t, s.
(2) The strings $Pt_1t_2 \ldots t_n$ for every possible choice of n-ary predicates P (for all choices of $n > 0$) and all possible choices of terms $t_1, t_2, \ldots, t_n$. □

We often abuse notation and write $P(t_1, \ldots, t_n)$ instead of $Pt_1 \ldots t_n$.

I.1.8 Definition (Well-Formed Formulas). The set of *well-formed formulas*, **Wff**, is the *smallest* set of strings or expressions over the alphabet $\mathscr{V}$ with the following properties:

(a) All the members of **Af** are included.
(b) If $\mathscr{A}$ and $\mathscr{B}$ denote strings (over $\mathscr{V}$) that are included, then $(\mathscr{A} \vee \mathscr{B})$ and $(\neg\mathscr{A})$ are also included.
(c) If $\mathscr{A}$ is[†] a string that is included and x is *any* object variable (*which may or may not occur (as a* substring*) in the string* $\mathscr{A}$), then the string $((\exists x)\mathscr{A})$ is also included. We say that $\mathscr{A}$ is the *scope* of $(\exists x)$. □

[†] Denotes!

I.1.9 Remark.

(1) The above is yet another inductive definition. Its statement (*in the metalanguage*) is facilitated by the use of so-called *syntactic*, or meta, variables – $\mathscr{A}$ and $\mathscr{B}$ – used as *names* for *arbitrary* (indeterminate) formulas. In general, we will let calligraphic capital letters $\mathscr{A}, \mathscr{B}, \mathscr{C}, \mathscr{D}, \mathscr{E}, \mathscr{F}, \mathscr{G}$ (with or without primes or subscripts) be names for well-formed formulas, or just *formulas*, as we often say. The definition of **Wff** given above is standard. In particular, it permits well-formed formulas such as $((\exists x)((\exists x)x = 0))$ in the interest of making the formation rules "context-free".†

(2) The rules of syntax just given do not allow us to write things such as $\exists f$ or $\exists P$ where f and P are function and predicate symbols respectively. That quantification is deliberately restricted to act solely on object variables makes the language *first order*.

(3) We have already indicated in Remark I.1.6 where the arities (of function and predicate symbols) come from (Definitions I.1.5 and I.1.7 referred to them). These are numbers that are implicit ("hardwired") with the formation rules for terms and atomic formulas. Each function and each predicate symbol (e.g., $+, \times, \in, <$) has its own unique formation rule. This rule "knows" how many terms are needed (on the input side) in order to form a term or atomic formula. Therefore, since the theory, *in use*, applies rather than studies its formation rules, it is, in particular, ignorant of arities of symbols.

Now that this jurisdictional point has been made (cf. the concluding remarks about decision questions, on p. 12), we can consider an alternative way of making arities of symbols known (in the *metatheory*): Rather than embedding arities in the formation rules, we can hide them in the ontology of the symbols, not making them explicit in the name.

For example, a new symbol, say $*$, can be used to record arity. That is, we can think of a predicate (or function) symbol as consisting of two parts: an arity part and an "all the rest" part, the latter needed to render the symbol unique.‡ For example, $\in$ may be actually the name for the symbol "$\in^{**}$", where this latter name is identical to the symbol it denotes, or "what you see is what you get" – see Remark I.1.3(1) and (2), p. 8. The presence of the two asterisks declares the arity. Some people say this differently: They make available to the metatheory a "function", ar, from "the set of

† In some presentations, the formation rule in I.1.8(c) is "context-sensitive": It requires that x be *not* already quantified in $\mathscr{A}$.

‡ The reader may want to glimpse ahead, on p. 166, to see a possible implementation in the case of number theory.

all predicate symbols and functions" (of a given language) to the natural numbers, so that for any function symbol f or predicate symbol P, $ar(f)$ and $ar(P)$ yield the arities of f and P respectively.†

(4) *Abbreviations*

Abr1. The string $((\forall x)\mathscr{A})$ abbreviates the string "$(\neg((\exists x)(\neg\mathscr{A})))$". Thus, for any explicitly written formula $\mathscr{A}$, the former notation is informal (metamathematical), while the latter is formal (within the formal language). In particular, $\forall$ is a metalinguistic symbol. "$\forall x$" is the *universal quantifier*. $\mathscr{A}$ is its scope. The symbol $\forall$ is pronounced *for all*.

We also introduce – in the metalanguage – a number of additional Boolean connectives in order to abbreviate certain strings:

Abr2. (*Conjunction,* $\wedge$) $(\mathscr{A}\wedge\mathscr{B})$ stands for $(\neg((\neg\mathscr{A})\vee(\neg\mathscr{B})))$. The symbol $\wedge$ is pronounced *and*.

Abr3. (*Classical or material implication,* $\rightarrow$) $(\mathscr{A}\rightarrow\mathscr{B})$ stands for $((\neg\mathscr{A})\vee\mathscr{B})$. $(\mathscr{A}\rightarrow\mathscr{B})$ is pronounced *if* $\mathscr{A}$*, then* $\mathscr{B}$.

Abr4. (*Equivalence,* $\leftrightarrow$) $(\mathscr{A}\leftrightarrow\mathscr{B})$ stands for $((\mathscr{A}\rightarrow\mathscr{B})\wedge(\mathscr{B}\rightarrow\mathscr{A}))$.

Abr5. To minimize the use of brackets in the metanotation we adopt standard *priorities* of connectives: $\forall$, $\exists$, and $\neg$ have the highest, and then we have (in decreasing order of priority) $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$, and we agree not to use outermost brackets. All *associativities* are *right* – that is, if we write $\mathscr{A}\rightarrow\mathscr{B}\rightarrow\mathscr{C}$, then this is a (sloppy) counterpart for $(\mathscr{A}\rightarrow(\mathscr{B}\rightarrow\mathscr{C}))$.

(5) The language just defined, L, is *one-sorted*, that is, it has a single *sort* or *type* of object variable. Is this not inconvenient? After all, our set theory (volume 2 of these lectures) will have both *atoms* and *sets*. In other theories, e.g., geometry, one has points, lines, and planes. One would have hoped to have different "types" of variables, one for each.

Actually, to do this would amount to a totally unnecessary complication of syntax. We can (and will) get away with just *one* sort of object variable. For example, in set theory we will also introduce a 1-ary‡ predicate, U, whose job is to "test" an object for "sethood".§ Similar remedies are available to other theories. For example, geometry will manage with one sort of variable and unary predicates "Point", "Line", and "Plane".

† In mathematics we understand a function as a set of input–output pairs. One can "glue" the two parts of such pairs together, as in "$\in^{**}$" – where "$\in$" is the input part and "$**$" is the output part, the latter denoting "2" – etc. Thus, the two approaches are equivalent.

‡ More commonly called *unary*.

§ People writing about, or teaching, set theory have made this word up. Of course, one means by it the property of being a set.