Computability and Logic

Fifth Edition

GEORGE S. BOOLOS

JOHN P. BURGESS
Princeton University

RICHARD C. JEFFREY

CAMBRIDGE
UNIVERSITY PRESS

CAMBRIDGE UNIVERSITY PRESS
Cambridge, New York, Melbourne, Madrid, Cape Town, Singapore, São Paulo

Cambridge University Press
The Edinburgh Building, Cambridge CB2 8RU, UK
Published in the United States of America by Cambridge University Press, New York

www.cambridge.org
Information on this title: www.cambridge.org/9780521877527

© George S. Boolos, John P. Burgess, Richard C. Jeffrey 1974, 1980, 1990, 2002, 2007

This publication is in copyright. Subject to statutory exception and to the provision of relevant collective licensing agreements, no reproduction of any part may take place without the written permission of Cambridge University Press.

First published in print format 2007

ISBN-13 978-0-511-36668-0 eBook (EBL)
ISBN-10 0-511-36668-X eBook (EBL)

ISBN-13 978-0-521-87752-7 hardback
ISBN-10 0-521-87752-0 hardback

ISBN-13 978-0-521-70146-4 paperback
ISBN-10 0-521-70146-5 paperback

Cambridge University Press has no responsibility for the persistence or accuracy of urls for external or third-party internet websites referred to in this publication, and does not guarantee that any content on such websites is, or will remain, accurate or appropriate.

9

A Précis of First-Order Logic: Syntax

This chapter and the next contain a summary of material, mainly definitions, needed for later chapters, of a kind that can be found expounded more fully and at a more relaxed pace in introductory-level logic textbooks. Section 9.1 gives an overview of the two groups of notions from logical theory that will be of most concern: notions pertaining to formulas *and* sentences*, and notions pertaining to* truth under an interpretation*. The former group of notions, called* syntactic*, will be further studied in section 9.2, and the latter group, called* semantic*, in the next chapter.*

9.1 First-Order Logic

Logic has traditionally been concerned with relations among statements, and with properties of statements, that hold by virtue of 'form' alone, regardless of 'content'. For instance, consider the following argument:

(1) A mother or father of a person is an ancestor of that person.
(2) An ancestor of an ancestor of a person is an ancestor of that person.
(3) Sarah is the mother of Isaac, and Isaac is the father of Jacob.
(4) Therefore, Sarah is an ancestor of Jacob.

Logic teaches that the premisses (1)–(3) (*logically*) *imply* or have as a (*logical*) *consequence* the conclusion (4), because in any argument of the same form, if the premisses are true, then the conclusion is true. An example of another argument of the same form would be the following:

(5) A square or cube of a number is a power of that number.
(6) A power of a power of a number is a power of that number.
(7) Sixty-four is the cube of four and four is the square of two.
(8) Therefore, sixty-four is a power of two.

Modern logic represents the forms of statements by certain algebraic-looking symbolic expressions called *formulas*, involving special signs. The special signs we are going to be using are shown in Table 9-1.

Table 9-1. *Logical symbols*

$\sim$	Negation	'not . . . '
&	Conjunction	'. . . and . . . '
$\vee$	Disjunction	'. . . or . . . '
$\rightarrow$	Conditional	'if . . . then . . . '
$\leftrightarrow$	Biconditional	'. . . if and only if . . . '
$\forall x, \forall y, \forall z, \ldots$	Universal quantification	'for every x', 'for every y', 'for every z', . . .
$\exists x, \exists y, \exists z, \ldots$	Existential quantification	'for some x', 'for some y', 'for some z', . . .

In this symbolism, the form shared by the arguments (1)–(4) and (5)–(8) above might be represented as follows:

(9) $\forall x \forall y((\mathbf{P}yx \vee \mathbf{Q}yx) \rightarrow \mathbf{R}yx)$
(10) $\forall x \forall y(\exists z(\mathbf{R}yz \;\&\; \mathbf{R}zx) \rightarrow \mathbf{R}yx)$
(11) **Pab** & **Qbc**
(12) **Rac**

Content is put back into the forms by providing an *interpretation*. Specifying an interpretation involves specifying what sorts of things the xs and ys and zs are supposed to stand for, which of these things **a** and **b** and **c** are supposed to stand for, and which relations among these things **P** and **Q** and **R** are supposed to stand for. One interpretation would let the xs and ys and zs stand for (human) persons, **a** and **b** and **c** for the persons Sarah and Isaac and Jacob, and **P** and **Q** and **R** for the relations among persons of mother to child, father to child, and ancestor to descendent, respectively. With this interpretation, (9) and (10) would amount to the following more stilted versions of (1) and (2):

(13) For any person x and any person y, if either y is the mother of x or y is the father of x, then y is an ancestor of x.
(14) For any person x and any person y, if there is a person z such that y is an ancestor of z and z is an ancestor of x, then y is an ancestor of x.

(11) and (12) would amount to (3) and (4).

A different interpretation would let the xs and ys and zs stand for (natural) numbers, **a** and **b** and **c** for the numbers sixty-four and four and two, and **P** and **Q** and **R** for the relations of the cube or the square or a power of a number to that number, respectively. With this interpretation, (9)–(12) would amount to (5)–(8). We say that (9)–(11) imply (12) because in *any* interpretation in which (9)–(11) come out true, (12) comes out true.

Our goal in this chapter will be to make the notions of formula and interpretation rigorous and precise. In seeking the degree of clarity and explicitness that will be needed for our later work, the first notion we need is a division of the symbols that may occur in formulas into two sorts: *logical* and *nonlogical*. The logical symbols are the logical operators we listed above, the *connective symbols* (the tilde $\sim$, the ampersand &, the wedge $\vee$, the arrow $\rightarrow$, the double arrow $\leftrightarrow$), the *quantifier symbols* (the inverted ay $\forall$, the reversed ee $\exists$), plus the *variables* $x, y, z, \ldots$ that go with the quantifiers, plus left and right parentheses and commas for punctuation.

The nonlogical symbols are to begin with of two sorts: *constants* or *individual symbols*, and *predicates* or *relation symbols*. Each predicate comes with a fixed positive number of *places*. (It is possible to consider zero-place predicates, called *sentence letters*, but we have no need for them here.) As we were using them above, **a** and **b** and **c** were constants, and **P** and **Q** and **R** were two-place predicates.

Especially though not exclusively when dealing with mathematical material, some further apparatus is often necessary or useful. Hence we often include one more logical symbol, a special two-place predicate, the *identity symbol* or equals sign $=$, for '... is (the very same thing as) ...'. To repeat, the equals sign, though a two-place predicate, is counted as a logical symbol, but it is the only exception: all other predicates count as nonlogical symbols. Also, we often include one more category of nonlogical symbols, called *function symbols*. Each function symbol comes with a fixed number of *places*. (Occasionally, constants are regarded as zero-place function symbols, though usually we don't so regard them.)

We conscript the word 'language' to mean an enumerable set of nonlogical symbols. A special case is the *empty language* $L_\varnothing$, which is just the empty set under another name, with no nonlogical symbols. Here is another important case.

9.1 Example (The language of arithmetic). One language that will be of especial interest to us in later chapters is called the *language of arithmetic*, L^*. Its nonlogical symbols are the constant zero $\mathbf{0}$, the two-place predicate less-than $<$, the one-place function symbol successor $'$, and the two-place function symbols addition $+$ and multiplication $\cdot$.

Intuitively, *formulas* are just the sequences of symbols that correspond to grammatically well-formed sentences of English. Those that, like (9)–(12) above, correspond to English sentences that make a complete statement capable of being true or false are called *closed* formulas. Those that, like $(\mathbf{P}yz \vee \mathbf{Q}yx)$, correspond to English sentences involving unidentified xs and ys and zs that would have to be identified before the sentences could be said to be true or false, are called *open* formulas.

The *terms* are sequences of symbols, such as $\mathbf{0}$ or $\mathbf{0}+\mathbf{0}$ or x or x'', that correspond to grammatically well-formed phrases of English of the kind that grammarians call 'singular noun phrases'. The *closed* terms are the ones that involve no variables, and the *open* terms are the ones that involve variables whose values would have to be specified before the term as a whole could be said to have a denotation. When no function symbols are present, the only closed terms are constants, and the only open terms are variables. When function symbols are present, the closed terms also include such expressions as $\mathbf{0}+\mathbf{0}$, and the open terms such expressions as x''.

The formulas and terms of a given language are simply the ones all of whose nonlogical symbols belong to that language. Since languages are enumerable and each formula of a language is a finite string of symbols from the language plus variables and logical symbols, the set of formulas is enumerable, too. (One might at first guess that the empty language would have no formulas, but at least when identity is present, in fact it has infinitely many, among them $\forall x\, x = x$, $\forall y\, y = y$, $\forall z\, z = z$, and so on.)

An *interpretation* $\mathcal{M}$ for a language L consists of two components. On the one hand, there is a nonempty set $|\mathcal{M}|$ called the *domain* or *universe of discourse* of the interpretation, the set of things $\mathcal{M}$ interprets the language to be talking about. When we say 'for every x' or 'for some x', what we mean, according to interpretation $\mathcal{M}$, is 'for every x in $|\mathcal{M}|$' or 'there exists an x in $|\mathcal{M}|$'. On the other hand, there is for each nonlogical symbol a *denotation* assigned to it. For a constant c, the denotation $c^{\mathcal{M}}$ is to be some individual in the domain $|\mathcal{M}|$. For an n-place nonlogical predicate R, the denotation $R^{\mathcal{M}}$ is to be some n-place relation on $|\mathcal{M}|$ (which is officially just a set of n-tuples of elements of $|\mathcal{M}|$, a one-place relation being simply a subset of $|\mathcal{M}|$).

For example, for the language L_G with constants **a** and **b** and **c** and two-place predicates **P** and **Q** and **R**, the genealogical interpretation $\mathcal{G}$ of L_G indicated above would now be described by saying that the domain $|\mathcal{G}|$ is the set of all persons, $\mathbf{a}^{\mathcal{G}}$ is Sarah, $\mathbf{b}^{\mathcal{G}}$ is Isaac, $\mathbf{c}^{\mathcal{G}}$ is Jacob, $\mathbf{P}^{\mathcal{G}}$ is set of ordered pairs of persons where the first is the mother of the second, and analogously for $\mathbf{Q}^{\mathcal{G}}$ and $\mathbf{R}^{\mathcal{G}}$. Under this interpretation, the open formula $\exists z(\mathbf{P}yz \;\&\; \mathbf{Q}zx)$ amounts to 'y is the paternal grandmother of x', while $\exists z(\mathbf{Q}yz \;\&\; \mathbf{P}zx)$ amounts to 'y is the maternal grandfather of x'. The closed formula $\sim\exists x\ \mathbf{P}xx$ amounts to 'no one is her own mother', which is true, while $\exists x\ \mathbf{Q}xx$ amounts to 'someone is his own father', which is false.

When the identity symbol is present, it is *not* treated like the other, nonlogical predicates: one is *not* free to assign it an arbitrary two-place relation on the domain as its denotation; rather, its denotation must be the genuine identity relation on that domain, the relation each thing bears to itself and to nothing else. When function symbols are present, for an n-place function symbol f, the denotation $f^{\mathcal{M}}$ is an n-argument function from $|\mathcal{M}|$ to $|\mathcal{M}|$.

9.2 Example (The standard interpretation of the language of arithmetic). One interpretation that will be of especial interest to us in later chapters is called the *standard interpretation* $\mathcal{N}^*$ of the language of the language of arithmetic L^*. Its domain $|\mathcal{N}^*|$ is the set of natural numbers; the denotation $\mathbf{0}^{\mathcal{N}^*}$ of the cipher $\mathbf{0}$ is the number zero; the denotation $<^{\mathcal{N}^*}$ of the less-than sign is the usual strict less-than order relation; the denotation $'^{\mathcal{N}^*}$ of the accent is the successor function, which takes each number to the next larger number; and the denotations $+^{\mathcal{N}^*}$ and $\cdot^{\mathcal{N}^*}$ of the plus sign and times sign are the usual addition and multiplication functions. Then such an open term as $x \cdot y$ would stand for the product of x and y, whatever they are; while such a closed term as $\mathbf{0}''$ would stand for the successor of the successor of zero, which is to say the successor of one, which is to say two. And such a closed formula as

(15) $\forall x \forall y(x \cdot y = \mathbf{0}'' \rightarrow (x = \mathbf{0}'' \vee y = \mathbf{0}''))$

would stand for 'for every x and every y, if the product of x and y is two, then either x is two or y is two' or 'a product is two only if one of the factors is two'. This happens to be true (given that our domain consists of natural numbers, with no negatives or fractions). Other closed formulas that come out true on this interpretation include the following:

(16) $\forall x \exists y(x < y \& \sim\exists z(x < z \& z < y))$

(17) $\forall x(x < x' \;\&\; \sim\exists z(x < z \& z < x'))$.

Here (16) says that for any number x there is a *next larger* number, and (17) that x' is precisely this next larger number.

(For the empty language $L_\varnothing$, there are no nonlogical symbols to be assigned denotations, but an interpretation must still specify a domain, and that specification makes a difference as to truth for closed formulas involving =. For instance, $\exists x \exists y \sim x = y$ will be true if the domain has at least two distinct elements, but false if it has only one.)

Closed formulas, which are also called *sentences*, have *truth values*, true or false, when supplied with an interpretation. But they may have different truth values under different interpretations. For our original example (9)–(12), on the genealogical interpretation we have since named $\mathcal{G}$ (and equally on the alternative arithmetical interpretation that we have left nameless) all four sentences came out true. But alternative interpretations are possible. For instance, if we kept everything else the same as in the genealogical interpretation, but took **R** to denote the relation of descendant to ancestor rather than vice versa, (10) and (11) would remain true, but (9) and (12) would become false: descendants of descendants are descendants, but parents and grandparents are not descendants. Various other combinations are possible. What one will *not* find is any interpretation that makes (9)–(11) all true, but (12) false. Precisely that, to repeat, is what is meant by saying that (9)–(11) *imply* (12).

9.3 Example (Alternative interpretations of the language of arithmetic). For the language of arithmetic, there is an alternative interpretation $\mathcal{Q}$ in which the domain is the nonnegative rational numbers, but the denotation of **0** is still zero, the denotation of $'$ is still the function that adds one to a number, the denotations of + and · are the usual addition and multiplication operations, and the denotation of < is still the less-than relation among the numbers in question. On this interpretation, (16) and (17) above are both false (because there are lots of rational numbers between x and any larger y in general, and lots of rational numbers between x and x plus one in particular). There is another alternative interpretation $\mathcal{P}$ in which the domain consists of the nonnegative half integers 0, $^1/_2$, 1, $1^1/_2$, 2, $2^1/_2$, 3, and so on, but the denotation of **0** is still zero, the denotation of $'$ is still the function that adds one to a number, the denotation of + is still the usual addition operation, and the denotation of < is still the less-than relation among the numbers in question. (Multiplication cannot be interpreted in the usual way, since a product of two half integers is not in general a half integer, but for purposes of this example it does not matter how multiplication is interpreted.) On this interpretation, (16) would be true (because there is no half integer between x and $y = x$ plus one-half), but (17) would be false (because there is a half integer between x and x plus one, namely x plus one-half). What you *won't* find is an interpretation that makes (17) true but (16) false. And again, that is what it means to say that (16) is a consequence of (17).

The explanations given so far provide part of the precision and rigor that will be needed in our later work, but only part. For they still rely on an intuitive understanding of what it is to be a sentence of a language, and what it is for a sentence be true in an interpretation. There are two reasons why we want to avoid this reliance on intuition. The first is that when we come to apply our work on computability to logic, we are going to want the notion of sentence to be so precisely defined that a *machine* could tell whether or not a given string of symbols is a sentence. The second is that the notion of truth was historically under a certain cloud of suspicion, owing to the occurrence of certain contradictions, euphemistically called 'paradoxes', such as the ancient *Epimenides* or *liar* paradox: If I say, 'what I am now saying is not true', is what I am saying true? We are therefore going to want to give, for sentences of the kind of formal language we are considering, a definition of truth just as rigorous as the definition of any other notion in mathematics, making the notion of truth, as applied to the kind of formal language we are considering, as respectable as any other mathematical notion.

The next section will be devoted to giving precise and rigorous definitions of the notions of formula and sentence, and more generally to giving definitions of notions pertaining to *syntax*, that is, pertaining to the internal structure of formulas. The next chapter will be devoted to giving the definition of truth, and more generally to giving definitions of notions pertaining to *semantics*, that is, pertaining to the external interpretation of formulas.

9.2 Syntax

Officially we think of ourselves as working for each $k > 0$ with a fixed denumerable stock of k-place predicates:

$$\begin{array}{cccc} A_0^1 & A_1^1 & A_2^1 & \cdots \\ A_0^2 & A_1^2 & A_2^2 & \cdots \\ A_0^3 & A_1^3 & A_2^3 & \cdots \\ \vdots & \vdots & \vdots & \end{array}$$

and with a fixed denumerable stock of constants:

$$f_0^0 \qquad f_1^0 \qquad f_2^0 \qquad \ldots.$$

When function symbols are being used, we are also going to want for each $k > 0$ a fixed denumerable stock of k-place function symbols:

$$\begin{array}{cccc} f_0^1 & f_1^1 & f_2^1 & \cdots \\ f_0^2 & f_1^2 & f_2^2 & \cdots \\ f_0^3 & f_1^3 & f_2^3 & \cdots \\ \vdots & \vdots & \vdots & . \end{array}$$

Any language will be a subset of this fixed stock. (In some contexts in later chapters where we are working with a language L we will want to be able to assume that there are infinitely many constants available that have not been used in L. This is no real difficulty, even if L itself needs to contain infinitely many constants, since we can either add the new constants to our basic stock, or assume that L used only every other constant of our original stock to begin with.)

We also work with a fixed denumerable stock of variables:

$$v_0 \qquad v_1 \qquad v_2 \qquad \ldots.$$

Thus the more or less traditional **0** and < and ′ and + and · we have been writing—and in practice, are going to continue to write—are in principle to be thought of as merely nicknames for f_0^0 and A_0^2 and f_0^1 and f_0^2 and f_1^2; while even writing x and y and z rather than v_i and v_j and v_k, we are using nicknames, too.

The official definition of the notion of formula begins by defining the notion of an *atomic formula*, which will be given first for the case where identity and function symbols are absent, then for the case where they are present. (If sentence letters were admitted, they would count as atomic formulas, too; but, as we have said, we generally are not going to admit them.) If identity and function symbols are absent, then an *atomic formula* is simply a string of symbols $R(t_1, \ldots, t_n)$ consisting of a predicate, followed by a left parenthesis, followed by n constants or variables, where n is the number of places of the predicate, with commas separating the successive terms, all followed by a right parenthesis. Further, if F is a formula, then so is its *negation* $\sim F$, consisting of a tilde followed by F. Also, if F and G are formulas, then so is their *conjunction* $(F \,\&\, G)$, consisting of a left parenthesis, followed by F, which is called the *left* or *first conjunct*, followed by the ampersand, followed by G, which is called the *right* or *second conjunct*, followed by a right parenthesis. Similarly for disjunction. Also, if F is a formula and x is a variable, the *universal quantification* $\forall xF$ is a formula, consisting of an inverted ay, followed by x, followed by F. Similarly for existential quantification.

And that is all: the definition of (*first-order*) *formula* is completed by saying that anything that is a (first-order) formula can be built up from atomic formulas in a sequence of finitely many steps—called a *formation sequence*—by applying negation, junctions, and quantifications to simpler formulas. (Until a much later chapter, where we consider what are called *second-order* formulas, ‘first-order’ will generally be omitted.)

Where identity is present, the atomic formulas will include ones of the kind $=(t_1, t_2)$. Where function symbols are present, we require a preliminary definition of terms. Variables and constants are *atomic* terms. If f is an n-place function symbol and $t_1, \ldots, t_n$ are terms, then $f(t_1, \ldots, t_n)$ is a term. And that is all: the definition of *term* is completed by stipulating that anything that is a term can be built up from atomic terms in a sequence of finitely many steps—called a *formation sequence*—by applying function symbols to simpler terms. Terms that contain variables are said to be *open*, while terms that do not are said to be *closed*. An atomic formula is now something of the type $R(t_1, \ldots, t_n)$ where the t_i may be any terms, not just constants or variables; but otherwise the definition of formula is unchanged.

Note that officially predicates are supposed to be written in front of the terms to which they apply, so writing $x < y$ rather than $<(x, y)$ is an unofficial colloquialism. We make use of several more such colloquialisms below. Thus we sometimes omit the parentheses around and commas separating terms in atomic formulas, and we generally write multiple conjunctions like $(A \,\&\, (B \,\&\, (C \,\&\, D)))$ simply as $(A \,\&\, B \,\&\, C \,\&\, D)$, and similarly for disjunctions, as well as sometimes omitting the outer parentheses on conjunctions and disjunctions $(F \,\&\, G)$ and $(F \vee G)$ when these stand alone rather than as parts of more complicated formulas. All this is slang, from the official point of view. Note that $\rightarrow$ and $\leftrightarrow$ have been left out of the official

Table 9-2. *Some terms of the language of arithmetic*

v_0	x
f_0^0	$\mathbf{0}$
$f_0^1(f_0^0)$	$\mathbf{1}$
$f_0^1(f_0^1(f_0^0))$	$\mathbf{2}$
$f_1^2(f_0^1(f_0^1(f_0^0)), v_0)$	$\mathbf{2} \cdot x$
$f_0^2(f_1^2(f_0^1(f_0^1(f_0^0)), v_0), f_1^2(f_0^1(f_0^1(f_0^0)), v_0))$	$\mathbf{2} \cdot x + \mathbf{2} \cdot x$

language entirely: $(F \to G)$ and $(F \leftrightarrow G)$ are to be considered unofficial abbreviations for $(\sim F \vee G)$ and $((\sim F \vee G)\ \&\ (\sim G \vee F))$. In connection with the language of arithmetic we allow ourselves two further such abbreviations, the bounded quantifiers $\forall y < x$ for $\forall y(y < x \to \ldots)$ and $\exists y < x$ for $\exists y(y < x\ \&\ \ldots)$.

Where identity is present, we also write $x = y$ and $x \neq y$ rather than $=(x, y)$ and $\sim=(x, y)$. Where function symbols are present, they also are supposed to be written in front of the terms to which they apply. So our writing x' rather than $'(x)$ and $x + y$ and $x \cdot y$ rather than $+(x, y)$ and $\cdot(x, y)$ is a colloquial departure from officialese. And if we adopt—as we do—the usual conventions of algebra that allow us to omit certain parenthesis, so that $x + y \cdot z$ is conventionally understood to mean $x + (y \cdot z)$ rather than $(x + y) \cdot z$ without our having to write the parentheses in explicitly, that is another such departure. And if we go further—as we do—and abbreviate $\mathbf{0}'$, $\mathbf{0}''$, $\mathbf{0}'''$, ..., as $\mathbf{1}$, $\mathbf{2}$, $\mathbf{3}$, ..., that is yet another departure.

Some terms of L^* in official and unofficial notation are shown in Table 9-2. The left column is a formation sequence for a fairly complex term.

Some formulas of L^* in official (or rather, semiofficial, since the the terms have been written colloquially) notation are shown in Table 9-3. The left column is a formation sequence for a fairly complex formula.

No one writing about anything, whether about family trees or natural numbers, will write in the official notation illustrated above (any more than anyone filling out a scholarship application or a tax return is going to do the necessary calculations in the rigid format established in our chapters on computability). The reader may well wonder why, if the official notation is so awkward, we don't just take the abbreviated

Table 9-3. *Some formulas of the language of arithmetic*

$A^2{}_0(x, \mathbf{0})$	$x < \mathbf{0}$
$A^2{}_0(x, \mathbf{1})$	$x < \mathbf{1}$
$A^2{}_0(x, \mathbf{2})$	$x < \mathbf{2}$
$A^2{}_0(x, \mathbf{3})$	$x < \mathbf{3}$
$\sim A^2{}_0(x, \mathbf{3}))$	$\sim x < \mathbf{3}$
$(= (x, \mathbf{1}) \vee = (x, \mathbf{2}))$	$x = \mathbf{1} \vee x = \mathbf{2}$
$(= (x, \mathbf{0}) \vee (= (x, \mathbf{1}) \vee = (x, \mathbf{2})))$	$x = \mathbf{0} \vee x = \mathbf{1} \vee x = \mathbf{2}$
$(\sim A^2{}_0(x, \mathbf{3}) \vee (= (x, \mathbf{0}) \vee (= (x, \mathbf{1}) \vee = (x, \mathbf{2}))))$	$x < \mathbf{3} \to (x = \mathbf{0} \vee x = \mathbf{1} \vee x = \mathbf{2})$
$\forall x((\sim A^2{}_0(x, \mathbf{3}) \vee (= (x, \mathbf{0}) \vee (= (x, \mathbf{1}) \vee = (x, \mathbf{2})))))$	$\forall x < \mathbf{3}(x = \mathbf{0} \vee x = \mathbf{1} \vee x = \mathbf{2})$

notation as the official one. The reason is that in proving things *about* the terms and formulas of a language, it is easiest if the language has a very rigid format (just as, in proving things *about* computability, it is easiest if the computations take place in a very rigid format). In writing examples of terms and formulas *in* the language, it is on the contrary easiest if the language has a very flexible format. The traditional strategy of logicians is to make the *official* language about which one proves theorems a very austere and rigid one, and to make the *unofficial* language in which one writes examples a very generous and flexible one. Of course, for the theorems proved about the austere idiom to be applicable to the generous idiom, one has to have confidence that all the abbreviations permitted by the latter but not the former *could in principle* be undone. But there is no need actually to undo them in practice.

The main method of proving theorems about terms and formulas in a language is called *induction on complexity*. We can prove that all formulas have a property by proving

Base Step: Atomic formulas have the property.

Induction Step: If a more complex formula is formed by applying a logical operator to a simpler formula or formulas, then, assuming (as *induction hypothesis*) that the simpler formula or formulas have the property, so does the more complex formula. The induction step will usually be divided into *cases*, according as the operator is $\sim$ or & or $\vee$ or $\forall$ or $\exists$.

Typically the proof will first be given for the situation where identity and function symbols are absent, then for the situation with identity present but function symbols absent, and then for the case with both identity and function symbols present. Identity typically requires very little extra work if any, but where function symbols are present, we generally need to prove some preliminary result about terms, which is also done by induction on complexity: we can prove that all terms have some property by proving that atomic terms have the property, and that if a more complex term is formed by applying a function symbol to simpler terms, then, assuming the simpler terms have the property, so does the more complex term.

The method of proof by induction on complexity is so important that we want to illustrate it now by very simple examples. The following lemma may tell us more than we want to know about punctuation, but is good practice.

9.4 Lemma (Parenthesis lemma). When formulas are written in official notation the following hold:

(a) Every formula ends in a right parenthesis.
(b) Every formula has equally many left and right parentheses.
(c) If a formula is divided into a left part and a right part, both nonempty, then there are at least as many left as right parentheses in the left part, and more if that part contains at least one parenthesis.

Proof: We give first the proof for (a). *Base step:* An atomic formula $R(t_1, \ldots, t_n)$ or $=(t_1, t_2)$ of course ends in a right parenthesis. *Induction step, negation case:* If F ends in a right parenthesis, then so does $\sim F$, since the only new symbol is at the beginning. *Induction step, junction case:* A conjunction $(F \,\&\, G)$ or disjunction $(F \vee G)$ of course ends in a right parenthesis. *Induction step, quantification case:* If

F ends in a right parenthesis, then so do $\forall xF$ or $\exists xF$, for the same reason as in the case of negation, namely, that the only new symbols are at the beginning.

In giving the proof for (b), we allow ourselves to be a little less rigid about the format. We consider first the case where function symbols are absent. First note that an atomic formula $R(t_1, \ldots, t_n)$ or $=(t_1, t_2)$ has equal numbers of left and right parentheses, namely, one of each. Then note that F has equal numbers of left and right parentheses, then so does $\sim F$, since there are no new parentheses. Then note that if F has m of each kind of parenthesis, and G has n of each, then $(F \mathbin{\&} G)$ has $m + n + 1$ of each, the only new ones being the outer ones. The proof for disjunction is the same as for conjunction, and the proofs for quantifications essentially the same as for negation.

If function symbols are present, we need the preliminary result that every term has equally many left and right parentheses. This is established by induction on complexity. An atomic term has equal numbers of left and right parentheses, namely zero of each. The nonatomic case resembles the conjunction case above: if s has m each of left and right parentheses, and t has n each, then $f(s, t)$ has $m + n + 1$ each; and similarly for $f(t_1, \ldots, t_k)$ for values of k other than two. Having this preliminary result, we must go back and reconsider the atomic case in the proof of (b). The argument now runs as follows: if s has m each of left and right parentheses, and t has n each, then $R(s, t)$ has $m + n + 1$ each, and similarly for $R(t_1, \ldots, t_k)$ for values of k other than two. No change is needed in the nonatomic cases of the proof of (b).

In giving the proof for (c), we also first consider the case where function symbols are absent. First suppose an atomic formula $R(t_1, \ldots, t_n)$ or $=(t_1, t_2)$ is divided into a left part λ and a right part ρ, both nonempty. If λ is just R or $=$, it contains zero parentheses of each kind. Otherwise, λ contains the one and only left parenthesis and not the one and only right parenthesis. In either case, (c) holds. Next assume (c) holds for F, and suppose $\sim F$ is divided. If λ consists just of $\sim$, and ρ of all of F, then λ contains zero parentheses of each kind. Otherwise, λ is of the form $\sim\lambda_0$, where λ_0 is a left part of F, and ρ is the right part of F. By assumption, then λ_0 and hence λ has at least as many left as right parentheses, and more if it contains any parentheses at all. Thus in all cases, (c) holds for $\sim F$. Next assume (c) holds for F and G, and suppose $(F \mathbin{\&} G)$ is divided. The possible cases for the left part λ are:

Case1	Case 2	Case 3	Case 4	Case 5	Case 6
$($	$(\lambda_0$	$(F$	$(F \mathbin{\&}$	$(F \mathbin{\&} \lambda_1$	$(F \mathbin{\&} G$

where in case 2, λ_0 is a left part of F, and in case 5, λ_1 is a left part of G. In every case, the part of λ after the initial left parenthesis has at least as many left as right parentheses: obviously in case 1, by the assumption of (c) for F in case (2), by part (b) in case (3), and so on. So the whole left part λ has at least one more left than right parenthesis, and (c) holds for $(F \mathbin{\&} G)$. The proof for disjunction is the same as for conjunction, and the proofs for quantifications essentially the same as for negation. We leave the case where function symbols are present to the reader.

We conclude this section with the official definitions of four more important syntactic notions. First, we officially define a string of consecutive symbols within a given formula to be a *subformula* of the given formula if it is itself a formula. Where function symbols are present, we can similarly define a notion of *subterm*. We stop to note one result about subformulas.

9.5 Lemma (Unique readability lemma).

(a) The only subformula of an atomic formula $R(t_1, \ldots, t_n)$ or $=(t_1, t_2)$ is itself.
(b) The only subformulas of $\sim F$ are itself and the subformulas of F.
(c) The only subformulas of $(F \mathbin{\&} G)$ or $(F \vee G)$ are itself and the subformulas of F and G.
(d) The only subformulas of $\forall x F$ or $\exists x F$ are itself and the subformulas of F.

These assertions may seem obvious, but they only hold because we use enough parentheses. If we used none at all, the disjunction of $F \mathbin{\&} G$ with H, that is, $F \mathbin{\&} G \vee H$, would have the subformula $G \vee H$, which is neither the whole conjunction nor a subformula of either conjunct. Indeed, the whole formula would be the same as the conjunction of F with $G \vee H$, and we would have a serious ambiguity. A rigorous proof of the unique readability lemma requires the parenthesis lemma.

Proof: For (a), a subformula of $R(t_1, \ldots, t_n)$ or $=(t_1, t_2)$ must contain the initial predicate R or $=$, and so, if it is not the whole formula, it will be a left part of it. Being a formula, it must contain (and in fact end in) a parenthesis by 9.4(a), and so, if it is not the whole formula but only a left part, must contain an excess of left over right parentheses by 9.4(c), which is impossible for a formula by 9.4(b).

For (b), a subformula of $\sim F$ that is not a subformula of F must contain the initial negation sign $\sim$, and so, if it is not the whole formula $\sim F$, it will be a left part of it, and from this point the argument is essentially the same as in the atomic case (a).

For (c), we relegate the proof to the problems at the end of the chapter.

For (d), the argument is essentially the same as for (b).

Resuming our series of definitions, second, using the notion of subformula, we state the official definition of which occurrences of a variable x in a formula F are *free* and which are *bound*: an occurrence of variable x is bound if it is part of a subformula beginning $\forall x \ldots$ or $\exists x \ldots$, in which case the quantifier $\forall$ or $\exists$ in question is said to *bind* that occurrence of the variable x, and otherwise the occurrence of the variable x is free. As an example, in

$$x < y \mathbin{\&} \sim\exists z(x < z \mathbin{\&} z < y)$$

all the occurrences of x and y are free, and all the occurrences of z are bound; while in

$$\mathbf{F}x \rightarrow \forall x \mathbf{F}x$$

the first occurrence of x is free, and the other two occurrences of x are bound. [The difference between the role of a free variable x and the role of a bound variable u in a formula like $\forall u\ R(x, u)$ or $\exists u\ R(x, u)$ is not unlike the difference between the roles of x and of u in mathematical expressions like

$$\int_1^x \frac{\mathrm{du}}{u} \qquad \sum_{u=1}^{x} \frac{1}{u}$$

For some readers this analogy may be helpful, and those readers who do not find it so may ignore it.]

In general, any and all occurrences of variables in an atomic formula $R(t_1, \ldots, t_n)$ are free, since there are no quantifiers in the formula; the free occurrences of a variable in a negation $\sim F$ are just the free occurrences in F, since any subformula of $\sim F$ beginning $\forall x$ or $\exists x$ is a proper subformula of $\sim F$ and so a subformula of F; and similarly, the free occurrences of a variable in a junction $(F \,\&\, G)$ or $(F \vee G)$ are just those in F and G; and similarly, the free occurrences of a variable other than x in a quantification $\forall xF$ or $\exists xF$ are just those in F, while of course none of the occurrences of x in $\forall xF$ or $\exists xF$ is free.

Third, using the notion of free and bound occurrence of variables, we state the official definition of the notion of an *instance* of a formula. But before giving that definition, let us mention a convenient notational convention. When we write something like 'Let $F(x)$ be a formula', we are to be understood as meaning 'Let F be a formula in which no variables occur free except x'. That is, we indicate which variables occur free in the formula we are calling F by displaying them immediately after the name F we are using for that formula. Similarly, if we go on to write something like 'Let c be a constant, and consider $F(c)$', we are to be understood as meaning, 'Let c be a constant, and consider the result of substituting c for all free occurrences of x in the formula F'. That is, we indicate what substitution is to be made in the formula we are calling $F(x)$ by making that very substitution in the expression $F(x)$. Thus if $F(x)$ is $\forall y \sim y < x$, then $F(\mathbf{0})$ is $\forall y \sim y < \mathbf{0}$. Then the official definition of instance is just this: an *instance* of a formula $F(x)$ is any formula of form $F(t)$ for t a closed term. Similar notations apply where there is more than one free variable, and to terms as well as formulas.

Fourth and finally, again using the notion of free and bound occurrence of variables, we state the official definition of *sentence*: a formula is a sentence if no occurrence of any variable in it is free. A *subsentence* is a subformula that is a sentence.

Problems

9.1 Indicate the form of the following argument—traditionally called 'syllogism in Felapton'—using formulas:

(a) No centaurs are allowed to vote.

(b) All centaurs are intelligent beings.

(c) Therefore, some intelligent beings are not allowed to vote.

Do the premisses (a) and (b) in the preceding argument imply the conclusion (c)?

9.2 Consider (9)–(12) of at the beginning of the chapter, and give an alternative to the genealogical interpretation that makes (9) true, (10) false, (11) true, and (12) false.

9.3 Consider a language with a two-place predicate **P** and a one-place predicate **F**, and an interpretation in which the domain is the set of persons, the denotation of **P** is the relation of parent to child, and the denotation of **F** is the set of all female persons. What do the following amount to, in colloquial terms, under that interpretation?

(a) $\exists z \exists u \exists v (u \neq v \mathbin{\&} \mathbf{P}uy \mathbin{\&} \mathbf{P}vy \mathbin{\&} \mathbf{P}uz \mathbin{\&} \mathbf{P}vz \mathbin{\&} \mathbf{P}zx \mathbin{\&} {\sim}\mathbf{F}\,y)$

(b) $\exists z \exists u \exists v (u \neq v \mathbin{\&} \mathbf{P}ux \mathbin{\&} \mathbf{P}vx \mathbin{\&} \mathbf{P}uz \mathbin{\&} \mathbf{P}vz \mathbin{\&} \mathbf{P}zy \mathbin{\&} \mathbf{F}\,y)$

9.4 Officially, a *formation sequence* is a sequence of formulas in which each either is atomic, or is obtained by some earlier formula(s) in the sequence by negation, conjunction, disjunction, or universal or existential quantification. A formation sequence *for a formula* F is just a formation sequence whose last formula is F. Prove that in a formation sequence for a formula F, every subformula of F must appear.

9.5 Prove that every formula F has a formation sequence in which the *only* formulas that appear are subformulas of F, and the number of formulas that appear is no greater than the number of symbols in F.

9.6 Here is an outline of a proof that the only subformulas of $(F \mathbin{\&} G)$ are itself and the subformulas of F and of G. Suppose H is some other kind of subformula. If H does not contain the displayed ampersand, then H must be of one of the two forms:

(a) $(\lambda$ where λ is a left part of F, or

(b) $\rho)$ where ρ is a right part of G.

If H does contain the displayed ampersand, then some subformula of H (possibly H itself) is a conjunction $(A \mathbin{\&} B)$ where A and B are formulas and either

(c) $A = F$ and B is a left part λ of G,

(d) A is a right part ρ of F and $B = G$, or

(e) A is a right part ρ of F and B is a left part λ of G.

Show that (a) and (b) are impossible.

9.7 Continuing the preceding problem, show that (c)–(e) are all impossible.

9.8 Our definition allows the same variable to occur both bound and free in a formula, as in $P(x) \mathbin{\&} \forall x Q(x)$. How could we change the definition to prevent this?