LOGIC: FORM AND FUNCTION

The Mechanization of Deductive Reasoning

at the University Press
EDINBURGH

Edinburgh University Press
22 George Square, Edinburgh
ISBN 0 85224 305 7
Printed in Great Britain by
R.& R.Clark Ltd
Edinburgh

Logic: Form and Function

Acknowledgements

Professor Kenneth Bowen, Dr Robert Kowalski, Mr Kevin Greene and the Rev. Hugh King-Smith (Brother Robert of the Society of St Francis) all read the manuscript of this book and made useful suggestions for which I am most grateful. To Professor F. L. Morris I owe a very special debt of thanks. I had the privileged advantage, throughout the making of this book, of his sage advice and his creative criticism and, in reading and correcting the proofs, of his meticulous perception: few authors are as fortunate, and this one hereby expresses his deep appreciation.

Contents

	Acknowledgements	vi
1.	Logic : Form and Content	1
2.	Formulas : Syntax and Intuitive Semantics	8
3.	Boolean Analysis of Sentences	35
4.	Infinite Finitary Trees and Boolean Compactness	61
5.	Semantic Analysis of Sentences and Terms	71
6.	Logical Consequence : Sequents and Proofs	92
7.	Logical Equivalence : Substitutivity and Variants	129
8.	Normal Forms of Sentences and Sequents	143
9.	Herbrand Models and Maps	163
10.	Quad Notation for Clausal Sequents	170
11.	Unification	182
12.	Resolution	199
13.	Resolution on the Computer	226
14.	Historical Notes	279
	Appendix	294
	References	300
	Index	303

1. Logic : Form and Content

Logic deals with what follows from what. It is the systematic study of the fundamental principles that underlie correct, "necessary" pieces of reasoning (or *true sequents*, as they are called later in the book), as these occur in *proofs*, *arguments*, *inferences*, and *deductions*. The correctness of a piece of reasoning, it is found, does not depend on what the reasoning is about (we can see that the conclusion *all epiphorins are turpy* follows from the premisses *all epiphorins are febrids* and *all febrids are turpy*, without understanding all the words) so much as on how the reasoning is done; on the pattern of relationships between the various constituent ideas rather than on the actual ideas themselves.

To get at the relevant aspects of such reasoning, logic must abstract its *form* from its *content*. We must disregard the epiphorins and the febrids and the turpiness and see the general truth that *all A are C* follows from *all A are B* and *all B are C* quite independently of the particular nature of the *A*, *B* and *C* that happen to occupy those places in the reasoning pattern.

If the content is irrelevant to the correctness of the inference of a conclusion from given premisses, it cannot matter whether the conclusion is true or false, nor whether the premisses are. For example, from the (false) premiss that *all prime numbers are odd* and the (true) premiss that *two is a prime number* the (false) conclusion that *two is odd* follows quite correctly. The form of the inference is: to infer *P is B* from *all A are B* and *P is A*; and this inference form is *valid*.

What does it mean to say that an inference – a piece of reasoning in which a sentence is inferred as conclusion from one or more sentences as premisses – is correct? What is being claimed when it is stated that the conclusion follows from the premisses?

It is not, as we have just seen, that the inference is correct because its conclusion is true: correct inferences can have false conclusions. Moreover, incorrect inferences can have true conclusions, as for example when we infer that *five is prime* from *five is odd* and *some primes are odd*: both premisses are true; the conclusion is true; the inference is nevertheless incorrect. It is incorrect by virtue of its

form: to infer *P is B* from *P is A* and *some A are B* is an *invalid* form of inference.

So we have an answer to the question: what makes an inference correct? The answer is: an inference is correct if its form is valid; incorrect if its form is invalid.

This naturally raises a further question: what does it mean to say that a form of inference is valid or invalid? Are not these just alternative words meaning correct or incorrect? It must be confessed that they are. The point is that we must ask (essentially) the same question about inference forms, rather than about inferences themselves. Correctness, or validity, is a property of inference forms rather than of individual inferences. Our answer above begs the question: it says that an inference is correct if its form is correct, and that it is incorrect if its form is incorrect!

However, we now have the question properly posed. Its answer is: an inference form is correct if there is no inference of that form whose premisses are true and whose conclusion is false.

Some people use the word *counterexample* as a convenient short means of expressing this: an inference form is correct if there is no counterexample to it. A counterexample to an inference form is just an inference, having that form, whose premisses are true but whose conclusion is false. Thus when we said that the inference form "to infer *P is B* from *P is A* and *some A are B*" is not correct, we meant that counterexamples to it exist. For instance, here is a counterexample to it: *nine is odd*, and *some primes are odd*, therefore *nine is prime*. Both premisses are true but the conclusion is false; and the form is that of the principle in question, which therefore has a counterexample, and so is not correct.

Notice that the concept of a counterexample introduces the concepts *true* and *false*. These are *semantic* ideas, having to do not only with the *meanings* of sentences and their constituent words and phrases, but also with the *facts*. On the other hand the form of a sentence and of its constituent sub-units is a *syntactic* idea, removed from any consideration of what the facts are or what the "irrelevant" words mean.

Although we are stressing the idea of abstracting away the "irrelevant" meanings, we must not give the impression that all meaning whatsoever is lost when we strip away content and leave behind only form. The "logical words" survive: **all, some, is (are), not, and, or, if ... then**, and so on. We represent and display logical forms as skeletons or schemata made up of various combinations of these logical words, linked together in the *patterns* found in working dis-

course, but with the "content words" removed and replaced by schematic letters or other symbols. The resulting "formulas" are simply notational devices for revealing, representing, portraying, exhibiting (whatever the term should be) the logical form of the sentences we started with. The transaction involved is better described as separating out the logical meaning from the nonlogical, and depicting the former with the help of special notations.

The exploitation of special notations in studying logical form has reached very high levels of perfection over the past century. The logician Quine says at the beginning of his well-known textbook *Methods of Logic*: "Logic is an old subject, and since 1879 it has been a great one." What happened in 1879? In that year, the German mathematician Gottlob Frege published a short booklet in which he set up a systematic notation for representing logical form. His system was essentially an abstract artificial language, designed with great care and deep insight, in which one could formulate propositions and proofs, with proper emphasis on their form.

Frege called his artificial language the *Begriffsschrift* – a German word he himself coined, to mean something like *the thought notation*, or *the concept language*. In more recent times it has come to be called the *predicate calculus*.

The great central core of modern logic – its mainstream – is the collection of ideas and facts which make up the system of the predicate calculus and its fundamental properties. The predicate calculus system is at one and the same time a notation intended for *use* in formulating and checking pieces of reasoning, and a repository or focus of all of the principal ideas and discoveries – some of them as deep and beautiful as any in the whole of mathematics – that constitute logic as a scientific theory. The chief purpose – or at any rate the chief satisfaction – of logic is the understanding it brings of the reasoning process as such – of the structure of "pure thought", as Frege put it in the title of his booklet: "BEGRIFFSSCHRIFT, a formalized language of pure thought, modelled upon that of arithmetic."

The "formalized language of arithmetic" upon which Frege modelled his predicate calculus is the one we all learn as children for representing numbers, and operations upon them, in both concrete and abstract ways. We learn the concrete algorithms of "counting", "addition", "multiplication", and so on, which help us to write such truths as

$$3 \times (17 + 4) = 63 \qquad (1)$$

and we acquire "algebraic" abstract general principles such as

$$a \times (b + c) = (a \times b) + (a \times c) \tag{2}$$

which allow us to represent the form of particular facts like (1). The great power of the system of symbolic representation, manipulation and computation provided by this familiar "formalized language" is available to all people with a standard elementary education. It is the basis for the more extensive, sophisticated, and less widely-known symbolic system of mathematical notation that is the common expressive medium and analytic instrument of the exact sciences.

Frege's predicate calculus, then, is "modelled upon" the notation that is designed to represent *numerical* form and structure. It is *only* modelled upon it: there is an analogy, and a deep one, but that is all. The predicate calculus is designed to represent *logical* form; to permit the construction and application of logical algorithms and analytical procedures. The way in which this representation is done, and what it is that is thereby represented, must be understood independently of the way that the numerical and algebraic symbolic systems work.

Yet there are certain concepts – those, one wants to say, that most nearly correspond to the "forms of pure thought" that Frege wanted to reveal and explain – that are not entirely unfamiliar to the student of the classic "arithmetical" symbolic formalisms. First among these notions is that of a *function*.

One is accustomed to considering, in numerical reasoning, such functions as are "defined" by or "given" by, e.g.

$$3x^2 + 2x + 4 \tag{3}$$

namely, by an *expression* that determines, when some thing (here, a number) x is given as "argument", another thing (here, again a number) as "value" or result. Thus the "expression" (3) defines a function F, which "yields" 9 when "applied to" the number 1, 20 when applied to 2,

$$F(1) = 9$$
$$F(2) = 20$$

and so on; since $3 \cdot (1)^2 + 2 \cdot (1) + 4 = 9$, and $3 \cdot (2)^2 + 2 \cdot (2) + 4 = 20$.

The function F defined by (3) above is applied to an argument a by the simple process of letting "x" denote a in the expression $3x^2 + 2x + 4$, and then "computing out" the resulting expression. This process is known as *evaluating* the expression $3x^2 + 2x + 4$ *at* a.

We can in this fashion define functions that apply to more than one argument, e.g. the function G defined by

$$3x^2 + 2xy + 4y^2 + 5 \tag{4}$$

which applies to pairs of numbers (a, b) and which we compute in the same way, by evaluating its "defining expression" $3x^2 + 2xy + 4y^2 + 5$ after letting x denote a and y denote b. Thus $G(1, 2)$ is

$$3 \cdot (1)^2 + 2 \cdot (1) \cdot (2) + 4 \cdot (2)^2 + 5 = 28$$

There is no reason why we cannot consider functions that yield things other than numbers when applied to numbers. For example, we can consider the "entities" *truth* and *falsehood* (known as the two *truthvalues*) as being yielded when we evaluate such expressions as

$$3x^2 + 12x + 4 \geqslant 0 \tag{5}$$

If x denotes 1 then (5) is true, i.e. "evaluates to truth"; while if x is -3 then (5) is false, i.e. "evaluates to falsehood". So we can look upon the function defined by (5) as one that yields truthvalues when applied to numbers. Similarly,

$$3x^2 + 2xy + 4y^2 + 5 < 21 \tag{6}$$

defines a function of *two* arguments which likewise yields truthvalues as results. Functions that yield truthvalues (and only truthvalues) as their results are called *relations*, or *predicates*.

The arguments of functions need not be numbers, either. For example, we can consider the expression

$$x \text{ is the sister of } y \tag{7}$$

as defining a function of two arguments, which are human beings, and that yields truth when applied to a pair (a, b) of humans such that a is b's sister, and falsehood when applied to other pairs of human beings. Indeed, the notion of function is entirely general, and neutral, as far as the nature of its arguments and results is concerned. We could, for instance, consider the function defined by

$$\text{the father of } x \tag{8}$$

which, when applied to a human being a, yields the human being who is a's father; and we can "nest" such functions just as we do with numerical ones, e.g.

$$x \text{ is the sister of (the father of } y) \tag{9}$$

Frege saw that the general, neutral concept of a function must be

made the principal one in his proposed system for symbolically representing the forms of "pure thought". The way in which a function can be defined, or represented, by giving an expression (as in the above examples) for its result as a computable or constructible combination of its arguments, is known as *abstraction*; and Frege's great insight was to see that the activity of "pure thought" he wanted to represent consisted of *acts of abstraction of functions from expressions*, interwoven with *acts of application of functions to arguments*, formulated as *acts of evaluation of expressions*. The interplay between abstraction, application and evaluation is the whole story of the thought that Frege wanted to analyse, explain, and represent.

The distinction (within the general framework of "thought as abstraction, application and evaluation" outlined above) between "pure" thought and the rest, is straightforward enough. Pure thought is not contrasted with impure thought – at least not by Frege, and not in this book – but with "applied thought", in much the same spirit as in the distinction between pure and applied mathematics, or pure and applied science. Certain functions, and certain kinds of result (i.e. the truthvalues) seem to occur in *all* forms of thought, whatever the particular subject matter. So these, along with the bare structure supplied by abstraction, application and evaluation as formative principles, are taken as the primitive ingredients of the "thought-writing" notation.

These functions and entities are the *logical* ones: *truth*, *falsehood*, *negation*, *conjunction*, *disjunction*, *universal* and *existential generalization*, and *exemplification*. We shall discuss these, and the notations for their representation, in the following chapter.

"Non-logical" functions and entities co-exist and interact with these logical ones in a completely integrated way: but for each special choice of subject-matter, and of a body of notions with which to organize that subject-matter conceptually, there is a particular "applied" system, or *calculus*. This "applied calculus" contains just the combination of general logical apparatus with the particular special notations deemed appropriate to the subject-matter of the applied calculus and to its particular special way of analysing that subject-matter.

Every "applied" calculus is an example of a particular way of conceptually organizing one's thoughts about some given "universe", or collection of "individuals", as subject-matter. If we then study how reasoning takes place within the framework of such an applied calculus, we are led to the underlying "pure" calculus, which directly depicts the "forms of pure thought" used to organize the subject-

matter, without actually introducing that subject-matter.

The first half of this book attempts to clarify the relationship between "pure" and "applied" forms of thought as embodied in the various pure and applied calculi of the symbolic system collectively known as "the" predicate calculus.

Once this formalism is understood. the reader can then easily entertain, and follow the development of, a proposition that has entranced students of logic for centuries: that deductive reasoning can be *mechanized* – literally, performed by a machine – just as many of the routine tasks of numerical computation can be and have been. It was not Frege's main motivation to help make the mechanization of deductive reasoning a practical possibility. He was very much aware, however, of the attempts of earlier thinkers such as Hobbes and Leibniz to argue that this could and must be done, and he well knew that his own work was an indispensable step.

The second half of this book traces the development of the theory of the predicate calculus as it is steered deliberately in the direction of this goal. The discussion culminates in a complete, detailed account of a working computer program for showing "what follows from what". The present author's hope is that the ideas underlying this computer program will be of interest even to those who have no prior experience of computers or of programming – we explain the necessary rudiments of the programming system used (LISP) so as to give the account a self-contained character.

Reasoning *is*, after all, a kind of computation: just as it has always been obvious that computation (as normally understood) is a kind of reasoning. The impressive rôle played by automatic computation in modern science and technology is made possible, ultimately, by the great *notations* of traditional mathematical analysis together with their supporting conceptual frameworks. The mechanization of "pure thought" itself – deductive reasoning as such – is, now that we have been given a notation for it, no longer just the dream of a Leibniz; it is now a reality. In the present author's opinion, mechanizing deductive reasoning is one of the most exciting and potentially fruitful areas of research that there has ever been.

Let us then begin by describing the elements of the predicate calculus system.

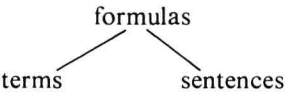
2. Formulas : Syntax and Intuitive Semantics

The predicate calculus has a simple, systematic basic *syntax*, whose principal feature is the characterization of the class of expressions that are its *formulas*. We shall denote this class by F.

The formulas are split into two subclasses: the class T of *terms*, and the class S of *sentences*. The general idea of the calculus is, roughly, that there is a set of things, which are "what the formulas are about". These things are called the *individuals*, and the set containing all of them is called the *universe of individuals*, or the *universe*, for short. The intended meaning of the terms is that they stand for ("denote") individuals. The sentences, on the other hand, are intended to express propositions about the individuals, which are either true or false. In the stylized semantics of the predicate calculus there is a special way of saying that a sentence expresses a *true* proposition; namely, one says that it stands for, or "denotes", *truth*. Similarly, one says of a sentence which expresses a *false* proposition that it stands for, or denotes, *falsehood*. Thus there is a set of two entities, called the *truthvalues*. These truthvalues, namely truth and falsehood, or **t** and **f** for short, are the counterparts, for the sentences, of the individuals. Sentences denote truthvalues; terms denote individuals.

That brief glimpse of the *semantics* of terms and sentences is offered as an aid to understanding the plan behind their *syntax*. We shall later on discuss the semantics much more fully.

So far, then, we have broken down the formulas into two classes, the terms and the sentences:



$$F = T + S$$

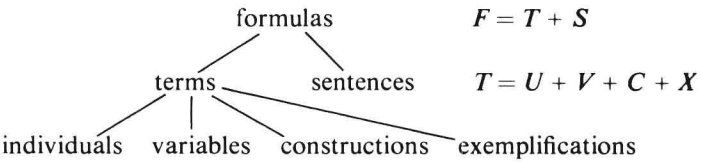
The class T of terms is divided into four classes: these are the *individuals*, the *variables*, the *constructions*, and the *exemplifications*. Let us consider these in order. First, the individuals.

We must emphasize the fact that individuals (i.e. elements of the universe of individuals) are not necessarily "symbolic" entities as usually understood by that notion: they are, rather, the things which

the symbolic entities represent. However, the usual notion of "symbol" is quite vague, and it seems to mean nothing more than "what formulas are ultimately made out of". In order to be reasonably clear here about our basic syntax we are taking an abstract attitude towards the formulas, by characterizing them as "constructs" of various kinds, without entering into an unnecessary commitment as to the nature of the basic building-blocks out of which the constructs are put together.

In this spirit we suppose that there are four sets: the universe U; the set V of *variables*; the set C of *constructions* and the set X of *exemplifications*. The universe U is the set of individuals. We assume nothing more about U than that it is a *non-empty* set: it may be finite, or infinite; and in the latter case its cardinal number may be arbitrarily large. (In this book, however, we shall not take advantage of this; we shall in fact be primarily interested in the case where U is finite or at most countably infinite.)

Next, the set V of variables. We assume that V is a countably infinite set, and that indeed there is some given enumeration of it (without repetitions) by reference to which we can unambiguously refer to the jth individual variable, where j is any positive integer.



We have so far, then, a countable infinity of variables, and an unknown (but non-zero) number of individuals, among our terms. In writing formulas we shall follow accepted usage and represent variables by letters x, y, z, ..., a, b, c, etc., affixing numerical subscripts, or further letters, as in x_1, a_2, ab, etc., as is convenient. We do not insist on any standard choice of written representation.

The third kind of term, the constructions, comprise the set C. Each consists of an *operator* and an *operand*. The operator of a construction is a symbol, called a constructor (or, sometimes, a *function symbol*), which is taken from a countably infinite set of such symbols about which we shall say more in a moment. The operand of a construction is *a finite sequence of terms*. This finite sequence may be empty (in which case we say it is *the empty operand*) or it may contain one or more components. These are called the *immediate constituents* of the construction.

With each constructor there is associated a non-negative integer,

called the *arity* of the constructor. This number must be the same as the number of components in the operand (i.e. as the length of that operand, considered as a finite sequence).

The set C_k of constructors of arity k, $k \geqslant 0$, is assumed to be equipped with some given enumeration, without repetitions, with respect to which we can unambiguously refer to the jth individual operator of arity k, for all positive integers j. We thus suppose that the constructors of arity k, $k \geqslant 0$, are given as a countably infinite set C_k; so that we have an infinite set of constructors

$$C_0 + C_1 + \ldots$$
$$= \sum_j C_j$$

(Here, we are representing the *union* of sets by symbols for addition, as is usual in discussions of formal syntax when the sets are *disjoint*, i.e. have no common members.)

Now if we have a set A, we denote the set of finite sequences of length k, $k \geqslant 0$, each component of which is a member of A, by A^k. In particular A^0 denotes the set whose only member is the *empty* sequence ( )

$$A^0 = \{( )\}$$

while A^1 denotes the set of *unit sequences* of members of A, i.e., if A is $\{a_1, a_2, \ldots\}$ then

$$A^1 = \{(a_1), (a_2), \ldots, \quad\}$$

After these two, the rest are easy, We have

$$A^2 = \{(a_1, a_1), (a_1, a_2), (a_2, a_1), (a_2, a_2), \ldots \quad\}$$

and so on.

Now the "Cartesian" product $(A \times B)$ of two sets is the set of all sequences of length 2 ("pairs")

$$(a, b)$$

whose first component a is in A and whose second component b is in B. Note that A^2 is then the same set as $(A \times A)$.

We have said that a construction is a pair: the operator and the operand of the construction. We can now be more specific. The operator is always a *constructor* of arity $k \geqslant 0$, and, for that k, the operand is always an element of the set T^k. That is, the operand is a sequence of length k whose components are terms.

We can express this definition compactly by the "syntax equation"

$$C = (C_0 \times T^0) + (C_1 \times T^1) + \ldots + (C_k \times T^k) + \ldots$$

or, writing the "infinite sum" more neatly

$$C = \sum_{k=0}^{\infty} (C_k \times T^k)$$

For the purpose of writing down constructions systematically we can conveniently represent them as parenthesized lists, e.g.

$(f\,a\,b)$
$(g(h\,a)(k)(f\,a\,b)(h(h(h\,a))))$

In any such list, its initial component is its operator, and its remaining components, if any, are its immediate constituents and thus comprise its operand. Thus in the above we are supposing that a and b are variables, that f is an individual operator of arity 2, k is an individual operator of arity 0, h is an individual operator of arity 1, and g is an individual operator of arity 4.

Since the immediate constituents of a construction may (some or all of them) be themselves constructions, we can expect in general to meet with terms that have a *nested* structure. This nesting can be arbitrarily deep, but it is always *finite*. (It is, intuitively, the maximum number of nested pairs of parentheses in the written representation of the term.) The nesting depth, or *depth*, of a term may be more formally defined by the recursive rule:

(a) · the depth of a variable, or of an individual, is 0;
(b) the depth of a construction is 1 greater than the maximum of the depths of its immediate constituents;

provided that we interpret the second clause to mean, in the case of a construction with a 0-ary operator (and hence no immediate constituents), that the depth of the term is then 1. Thus the depth of a and b is 0; that of $(f\,a\,b)$ is 1; that of $(h\,a)$ and (k) is also 1; while that of $(g(h\,a)(k)(f\,a\,b)(h(h(h\,a))))$ is 4.

We think of constructions as examples of *applicative* expressions, so called because of the way in which they come to denote the entities they do denote. The general idea is that an applicative expression τ with operator ω and operand δ stands for, or denotes, the entity that results from *applying* a certain function Ω to a certain finite sequence Δ of entities. The function Ω is the one that the symbol ω denotes, while the finite sequence Δ is the one whose components are the entities denoted, in order, by the expressions that are the components of δ.