COMPUTER PROGRAMMING

AND

FORMAL SYSTEMS

Edited by

P. BRAFFORT

Centre de Traitement de l'Information Scientifique, Euratom (Ispra)

and

D. HIRSCHBERG

IBM Belgium and Université Libre de Bruxelles

1963

NORTH-HOLLAND PUBLISHING COMPANY
AMSTERDAM

PREFACE

This book is a product of two seminars held in the IBM World Trade European Education Center of Blaricum (Holland) in 1961, the first of which was dedicated to a general survey of non-numerical applications of computers whereas the second was more specifically concerned with some aspects of the theory of formal systems. Professor E. W. Beth, who took part in both seminars, was kind enough to sponsor the publication of the proceedings in the "Studies of Logic and Foundations of Mathematics". Rather than publishing everything, a choice was made on the basis of relevance to the subject matter described in the title of this volume. Most authors preferred moreover to present a revised version of their contributions.

Symbol manipulation plays an important role both in the theory of formal systems and in computer programming and one would therefore expect some important relationships to exist between these domains. It may therefore seem surprising that specialists in the two fields have only recently become interested in one another's techniques. This situation is probably due to an original difference in motivation and to a phaseshift in time.

Electronic computers made their appearance about fifteen years ago, and have up to now been used essentially to solve problems in numerical mathematics and to process commercial data, two activities with a rather well-established methodology. Moreover, although a computer is indeed a general purpose symbol manipulator, advanced linguistic techniques for prescribing or describing its behaviour are unnecessary as long as the structure of the problems does not differ too much from the arithmetical structure built into the hardware of the machine.

Research in formal logics, on the other hand, received its original impetus from the foundation problems of mathematics, which originated at the end of the XIXth century. During the last three decades the emphasis has shifted from the study of particular formal systems, capable of formalizing mathematical disciplines, to the investigation of the fundamental properties of formal systems in general, such as the existence or non-existence of decision procedures for certain questions. In these investigations practical feasibility has not been taken into account and the results are therefore not directly related to actual computing problems, as was pointed out by Professor Hao Wang.

The current interest in formal logics, manifested by certain computer programmers, is due to a desire to broaden the scope of computer usage beyond the numerical area. It has become customary to call "non-numerical" a class of computer applications, typical members of which are: language translation, information retrieval, game playing, pattern recognition and theorem proving. In addition to exhibiting a structure which cannot easily be reduced, by an appropriate coding, to the arithmetical structure, most of these applications are not amenable to standard decision procedures and some are extremely complex. Hence the need for new efficient algorithms and for "heuristics", i.e. shortcuts.

Some logicians are similarly interested in going beyond the "asymptotic" concept of decidability, which corresponds to a machine with unlimited storage capacity and computing time. On the basis of experiments in theorem proving on actual computers they want to assess the relative efficiencies of various proof procedures and to investigate the practical impact of classical results on decidability.

The concept of a computing mechanism occurred to the logicians many years before the advent of the electronic stored program machine, but the existence of real computers and of corresponding programming techniques has had a considerable influence on present day research in automata theory.

These are the various aspects of the relationships between computer programming and the theory of formal systems discussed in the following pages. We have attempted to group as much as possible articles according to similarities of interest; it is however clear that often more than one aspect has been considered by the authors so that our classification remains quite arbitrary.

THE EDITORS

CONTENTS

MECHANICAL MATHEMATICS AND INFERENTIAL ANALYSIS

HAO WANG

Cambridge, Mass.

1. General Speculations

If we compare calculating with proving, four differences strike the eye: (1) Calculations deal with numbers; proofs, with propositions. (2) Rules of calculation are generally more exact than rules of proof. (3) Procedures of calculation are usually terminating (decidable, recursive) or can be made so by fairly well-developed methods of approximation. Procedures of proof, however, are often nonterminating (undecidable or nonrecursive, though recursively enumerable), indeed incomplete in the case of number theory or set theory, and we do not have a clear conception of approximate methods in theorem-proving. (4) We possess efficient calculating procedures, while with proofs it frequently happens that even in a decidable theory, the decision method is not practically feasible. Although shortcuts are the exception in calculations, they seem to be the rule with proofs in so far as intuition, insight, experience, and other vague and not easily imitable principles are applied. Since the proof procedures are so complex or lengthy, we simply cannot manage unless we somehow discover peculiar connections in each particular case.

Undoubtedly, it is such differences that have discouraged responsible scientists from embarking on the enterprise of mechanizing significant portions of the activity of mathematical research. The writer, however, feels that the nature and the dimension of the difficulties have been misrepresented through uncontrolled speculation and exaggerated because of a lack of appreciation of the combined capabilities of mathematical logic and calculating machines.

Of the four differences, the first is taken care of either by quoting Gödel representations of expressions or by recalling the familiar fact that alphabetic information can be handled on numerical (digital) machines. The second difference has largely been removed by the achievements of mathematical logic in formalization during the past eighty years or so. Item (3) is not a difference that is essential to the task of proving

theorems by machine. The immediate concern is not so much theoretical possibility as practical feasibility. Quite often a particular question in an undecidable domain is settled more easily than one in a decidable region, even mechanically. We do not and cannot set out to settle all questions of a given domain, decidable or not, when, as is usually the case, the domain includes infinitely many particular questions. In addition, it is not widely realized how large the decidable subdomains of an undecidable domain (e.g., the predicate calculus) are. Moreover, even in an undecidable area, the question of finding a proof for a proposition known to be a theorem, or formalizing a sketch into a detailed proof, is decidable theoretically. The state of affairs arising from the Gödel incompleteness is even less relevant to the sort of work envisaged here. The purpose here is at most to prove mathematical theorems of the usual kind, e.g., as exemplified by treatises on number theory, yet not a single "garden-variety" theorem of number theory has been found unprovable in the current axiom system of number theory. The concept of approximate proofs, though undeniably of a kind other than approximations in numerical calculations, is not incapable of more exact formulation in terms of, say, sketches of and gradual improvements toward a correct proof.

The difference (4) is perhaps the most fundamental. It is, however, easy to exaggerate the degree of complexity which is necessary, partly because abstract estimates are hardly realistic, partly because so far little attention has been paid to the question of choosing more efficient alternative procedures. The problem of introducing intuition and experience into machines is a bit slippery. Suffice it to say for the moment, however, that we have not realized that much of our basic strategies in searching for proofs is mechanizable, because we had little reason to be articulate on such matters until large, fast machines became available. We are in fact faced with a challenge to devise methods of buying originality with plodding, now that we are in possession of servants which are such persistent plodders. In the more advanced areas of mathematics, we are not likely to succeed in making the machine imitate the man entirely. Instead of being discouraged by this, however, one should view it as a forceful reason for experimenting with mechanical mathematics. The human inability to command precisely any great mass of details sets an intrinsic limitation on the kind of thing that is done in mathematics and the manner in which it is done. The superiority of machines in this respect indicates that machines, while following the broad outline of

paths drawn up by people, might yield surprising new results by making many new turns which man is not accustomed to taking.

The attempt to mechanize, as much as possible, mathematical thinking opens up a large area of research. In my opinion, the theoretical core of this area is a new branch of applied logic which may be called inferential analysis, characterized by an emphasis on the explicitless and practical feasibility of methods of inference. This discipline enjoys a measure of autonomy not shared by numerical analysis which, for example, does not deal with logical operations on the choice and arrangement of numerical methods. It is believed that the development of mechanical mathematics will influence pedagogical and research methods in mathematics, as well as affect certain epistemological questions of a specifically mathematical coloring.

The governing general principle is: what can be formulated exactly can be put on a machine, subject to the practical limitations on the manageable dimension and complexity. Repetitions are a good indication of the suitability of a mechanical treatment. Thus, for example, much of the activity of teaching mathematics is tedious and requires patience. If no interaction between pupil and teacher were necessary, televisions, or sometimes just gramophones, would be sufficient to replace teachers. As it is, these ready-made conveniences are only used as partial substitutes but, even so, teaching has already begun to enjoy to a certain extent the advantages of mass production. However, interesting problems of mechanical mathematics arise only when we come to tasks which call for an active agent to give answers, advices, and criticisms, the simplest being the correction of exercises and examination papers. Psychologically, the pupil has many reasons for preferring a patient machine teacher when the problem is, as in the majority of situations, a matter of drill rather than inspiration. The result may be that human teachers will employ mechanical devices as teaching assistants.

In a similar fashion, since in mathematical research there is also a great deal of mechanizable tedious work, mechanical devices may be used to aid individual mathematicians. In this connection, in view of the fact that specific mathematical discoveries are made essentially once and for all, there are less of exact repetitions, but more of the problem of using mechanical devices flexibly by, for example, designing and combining programs on general purpose computers. In order to use machines either to aid research or to aid teaching, the results, methods, and spirit of formalization in mathematical logic are to play an essential role.

The advance of mechanical mathematics may also affect some of our concepts in the philosophy of mathematics. We get, not only in theory, but even in practice, an objective criterion of mathematical rigor in mechanical terms. The range of feasible mathematical methods will be extended so that the theoretically feasible and the practically feasible begin to converge, and we have a more realistic guidance to the improvement of feasibility. As we understand more fully the range of mechanical mathematics, we get a clearer view of the relation between complexity and conceptual difficulty in mathematics, since we would probably wish to say that mechanizable pieces, even when highly complex, are conceptually easy. When alternative proofs are fully mechanizable, we obtain also a quantitative measure of the simplicity of mathematical proofs, to supplement our vaguer but richer intuitive concept of simplicity. With the increasing power to formalize and mechanize, we are freed from tedious details and can more effectively survey the content and conceptual core of a mathematical proof.

2. The Central Role of Logic

In theory all mathematical arguments can be formalized in elementary logic (quantification theory, predicate calculus). If we add equality and the quantifiers "for all x" and "for some y" to the propositional connectives "and", "if", "or", "not", etc., we obtain the predicate calculus, in which, as logicians will know, every usual mathematical discipline can be so formulated that each theorem T in the latter becomes one in the former when the relevant mathematical axioms A are added as premises. That is to say, if T is the theorem in the mathematical discipline, then "if A, then T" is a theorem of logic. From this fact it is clear that in order to prove mathematical theorems by machines a major step is to deal with theorems of the predicate calculus.

One may question the advantage of thus handling mathematics, on the ground that the peculiar mathematical content of each individual branch is lost when the disciplines are thus uniformly incorporated into the predicate calculus by formalization and abstraction. Now it is indeed true that we must add special methods for each special mathematical discipline. An adequate treatment of the predicate calculus is, however, of dominant importance, and for each discipline the basic additional special methods required are fairly uniform. For number theory, the essential new feature is largely concentrated in mathematical induction

as a method of proof and of definition; for set theory, in the axiom of comprehension, i.e., the axiom specifying all the conditions which define sets. So there is the problem of choosing the formula to make induction on, or of choosing the condition for defining a set. While it seems doubtful that there is any uniform efficient mechanical method for making such selections, there are often quite feasible partial methods. For example, for making such selections in number theory the obvious uninspired method of trying the desired conclusion of one or another of its clauses as the induction formula should suffice in many cases. It would seem that, once a feasible way of doing logic is given, fairly simple additional methods could carry us quite some way into special mathematical disciplines.

Since most of us learned Euclid and number theory without worrying about the predicate calculus, it might seem that the natural course is to bypass logic and go directly to mathematics. But in fact such an approach is ill-advised, so long as the aim is to prove more and harder theorems rather than merely to re-enact the history of mathematical thinking. What is natural for people need not be natural for the machine. If logic is not treated in an explicit and systematic way, constant subsequent additions of *ad hoc* devices keep slowing our progress toward interesting theorems, while multiplying the sources of possible confusion. In general, a vast machinery specifically designed to obtain a few easy theorems is wasteful; results obtained from whatever approaches should be measured against the generality and economy of the machinery used. Foundations, furthermore, should be scaled to large future superstructures. It is our conviction that to treat logic only by the way would score very poorly by both criteria.

3. Some Possible Directions for Further Exploration

Results so far are too rudimentary to provide us with any decisive conclusions as to the dimension of the long-range effects of the pursuit of mechanical mathematics. Nevertheless, I shall venture a few comments drawn from my own restricted experience. (a) I have examined the theoretically undecidable domain of the predicate calculus and managed to make an IBM 704 prove all theorems (over 350) of Principia mathematica in this domain in less than 9 minutes; this suggests that we usually do not use the full power of strong mathematical methods and should not be prevented from trying to handle an area on account of

pessimistic abstract estimates of the more difficult cases in the region. (b) Care in the theoretical design of the procedures is essential and a certain amount of sophistication in mathematical logic is indispensable, because most existing methods are not immediately applicable on machines. (c) In particular, one often has to reformulate available methods or even invent fundamentally new ones; sometimes theoretically insignificant improvements could increase the speed or reduce the necessary storage by several orders of magnitude, for example, a device to try out certain "preferred" substitutions first. (d) Long-range planning and efforts to make results cumulative are necessary; *ad hoc* measures and desire for quick sensation should be avoided because otherwise the limit of diminishing return will be reached too soon; the correct course would increase reward per unit of work more and more quickly with greater and greater efforts. (e) While more can be done with larger machines, the design and choice of methods is, at least at the present stage, more crucial because we are far from having made full use of an IBM 704 or 7090 yet. (f) Distrust luck and do not, for example, use obscure methods with the hope that something wonderful might happen since we do not know what will happen; the chances of undesirable consequences are much bigger.

At the present stage, mechanical mathematics seems to be one of the areas in information processing which promise the highest reward for each unit of labor. Only accidental circumstances such as the lack of alliance of potential contributors in administration, programming, and logic have so far sabotaged more rapid developments. The laziest solution of this practical difficulty is for one to attack problems in isolation and hope that the pieces will miraculously fit together in due course. This is not the most satisfactory solution but perhaps the most feasible, given all the facts of competition, sales exaggeration, desire for liberty and independence. There are at least three groups of preliminary work necessary for genuine advances in the long run: a good common (idealized programming) language for crystallization, communication, and accumulation of results; a decent library of subroutines for simple algebraic and logical manipulations of symbols, as well as for simple basic proof and decision procedures; and a fairly sophisticated logical analysis of a number of specific mathematical proofs with a view to bringing out the details which have to be taken care of in order to thoroughly formalize and mechanize them.

It is of course not excluded that one would often run into blind

alleys. But I am confident major wastes can be avoided through careful planning and alert flexibility. With these provisos in mind, I now proceed to list a few possible directions which, in my opinion, are worthy of at least some preliminary exploration.

That proof procedures for elementary logic can be mechanized is familiar. In practice, however, were we slavishly to follow these procedures without further refinements, we should encounter a prohibitively expansive element. It is desirable to study underlying properties of such expansions in order to increase efficiency. In this way we are led to a closer study of reduction procedures and of decision procedures for special domains, as well as of proof procedures of more complex sorts. Such deeper considerations of elementary logic also provide us with a systematic approach to axiomatic theories viewed as applied predicate calculus. The insights thus obtained can complement our direct treatment of specific mathematical disciplines.

For the sake of a more concrete goal to guide the choice of theoretical questions, we may set ourselves the aim of programming machines to formalize and "discover" proofs in quantifier-free number theory and axiomatic set theory. These areas are chosen both because they are so central and because it seems desirable to isolate the two basically difficult mathematical concepts: functions and quantifiers. It is possible that the quantifier-free theory of positive integers, including arbitrary simple recursive definitions, can be handled mechanically with relative ease, and yield fairly interesting results. It is clear from works in the literature that this restricted domain of number theory is rather rich in content. It goes beyond logic in an essential way because of the availability of (quantifier-free) mathematical induction. On the other hand, in axiomatic set theory, the explicit use of functions can be postponed for quite a long time. Moreover, here certain general concepts often prove difficult; perhaps machines will more quickly excel in areas where people's intuitions are not strong. A case in point would be Quine's axiomatic system "New Foundations," which was obtained by relaxing certain syntactical restrictions of the theory of types.

While the ulterior aim is to use machines to aid mathematical research with the assistance of logic, machines can also be used to aid our theoretical research in logic at the present stage. Computers can be put to good use in the quantity production of concrete examples, which we constantly need as a means of clarifying our concepts and so expediting general theoretical results.

Already in the limited experience in the mechanizing of logical procedures, the machine outputs have from time to time brought out features of the procedures which one had not thought out clearly in advance. Such experiences have sufficed to persuade us that we would do well to experiment with computing machines even if it were only for purposes of theoretical logic.

Some other possible directions are: (1) Experiment with redoing school and college mathematics by machines; instruct a machine to compete with the average student by using its patience to compensate its lack of intuition; partial decision procedures in algebra, trigonometry, analytic geometry, the calculus; prove theorems in elementary geometry and algebra with extensive use of methods dealing with the underlying logic. (2) Try to combine numerical and inferential methods so that principles can be introduced for the machine to choose particular methods to apply according to the nature of the given problems; this aims at delegating to the machine as much as possible of the work which now requires a mathematical analyst. (R.W. Hamming is much interested in work along this direction.) (3) In fields like algebraic topology where often definitions are long but proofs are short, it is not unlikely that mechanized logical methods will prove to be of practical use in helping to sort out logical consequences of new concepts. (4) Fairly simple mathematical researches involving combinatorial considerations such as questions of completeness, independence, deducibility in the various systems of the propositional calculus can presumably be helped radically by a few suitably devised machine programs. (5) Use this type of work as data to guide us in the design of more realistic idealized programming languages.

With regard to the formulation of programming languages, it seems desirable not to pursue the task in isolation and then look for applications afterwards. One must not let the initial investment in a programming language control the choice of problems to be programmed, but frequent revisions of a fixed language require a prohibitive amount of energy and work which can easily prevent one from meeting new demands with an open mind.

A good compromise between rigidity and complete lack of organization would seem to be the isolation of necessary devices such as the designing of MACRO instructions at every stage, as is called for by specific but typical occasions. In this way, a body of quite well-organized data would gradually emerge as more programs are written. Attention to the accumulation of good MACRO instructions also brings

into the somewhat routine task of programming a theoretically more interesting element of formulating exactly concepts which are intuitively familiar.

4. Case Studies and Stock-of-Trade Systems

To analyze in detail specific mathematical proofs is clearly a useful preliminary step toward the mechanization of types of arguments. One might attempt to work out a few examples of such case studies drawn from number theory, geometry, and axiomatic set theory.

In number theory, one might compare quantifier and free variable proofs of the infinitude of primes and of the fundamental theorem of arithmetic, putting emphasis on recursive functions and mathematical induction. In geometry and axiomatic set theory, one might consider mildly difficult theorems which are proved with quantifiers but without functions. In each case, two types of problems can be conveniently separated: deriving very elementary properties such as the commutativity of addition from the basic axioms on the one hand, and organizing such elementary properties to obtain a basis for further derivations on the other. For the human being, the first type of problem is rather artificial and contra-intuitive. For example, the very early theorems in elementary geometry are more abstruse than the simple exercises about parallels, triangles, etc. A good organization of elementary properties plus an exact formulation of familiar methods of trying to find a proof would presumably yield in each discipline something similar to the principles obtained from what is often called the "heuristic approach". It is here proposed that such organizations be called stock-of-trade systems. However, despite terminological disputes, everybody probably agrees as to roughly what sort of thing is to be done, and the more relevant question is how good a result one gets. It is with regard to this last point that a patient study of special cases with ample use also of the stocks in trade of mathematical logic appears indispensible. For instance, even a formalization of Euclid's proof of the infinitude of primes contains a few surprises, and there are quite a number of theoretically interesting questions connected with the problem of proving the irrationality of $\sqrt{2}$ with no appeal to quantifiers.

We consider here only an example in axiomatic set theory derived from a paper of Hintikka [13].

If a theorem is proved in a system, even one with only a finite set

of axioms, it would seem that one major problem is to select the axioms needed for the proof. It stands to reason to expect that it would be easier for the machine to begin with the selected axioms. In so doing, we may lose some alternative proof which uses other axioms but that is something which we do not have to worry about yet. Moreover, it appears easier to select and prove intermediate lemmas and break up the whole proof into parts. In both cases, if we do not have the selection to begin with, it is not easy to decide whether it is advantageous to take all to begin with, or to add routines to select. In the long run, one would expect to use the latter alternative. But when the methods of selecting subproblems and branching out are as cumbersome as some existing crude attempts appear to be, it is not necessarily more efficient to use the selection alternative.

The example to be considered is of special interest because it lies in an area which has not been developed nearly as much as old subjects such as number theory. Consequently, we can draw very little from a cumulative intuition, and our advantages over machines are not great. Moreover, this area has been pursued with a considerable emphasis on formal arguments.

Let Hxy and $\exists z Gxyz$ be short for:

$$\exists z(z \neq x \wedge z \neq y \wedge Fzy \wedge Fyz).$$

(1) $u \neq v$.
(2) $y \neq a \supset (Fya \equiv Hay)$.
(3) $y \neq b \supset (Fyb \equiv \neg Hby)$.
(4) $y \neq c \supset (Fyc \equiv (y = a \vee y = b))$.
(5) $y \neq d \supset (Fyd \equiv y = c)$.

The assertion is that the conjunction of (1)–(5) is contradictory. More exactly, this says that the following formula is a theorem of the predicate calculus.

$$\text{(I)}\ \neg \exists u \exists v \exists a \exists b \exists c \exists d \forall y \forall z \exists w \exists x$$

$$\begin{aligned} &\{u \neq v \\ &\wedge [y \neq a \supset ((Fya \wedge Gayw) \vee (\neg Fya \wedge \neg Gayz))] \\ &\wedge [y \neq b \supset ((Fyb \wedge \neg Gbyz) \vee (\neg Fyb \wedge Gbyx))] \\ &\wedge [y \neq c \supset (Fyc \equiv (y = a \vee y = b))] \\ &\wedge [y \neq d \supset (Fyd \equiv y = c)]\}. \end{aligned}$$

If the system does not include $=$, then we have to treat $a = b$ as

an abbreviation for

$$\forall x(Fxa \equiv Fxb),$$

and add the axiom:

$$a = b \supset \forall y(Fay \equiv Fby).$$

This incidentally illustrates the fact that for mechanical mathematics it is in practice desirable to include $=$ to begin with. In that case, the formula is in one of the familiar decidable cases since it contains only two consecutive $\exists$'s (for validity). In terms of satisfiability, the part without the initial $\neg$ has no model and can be decided by the $\exists\forall_2\exists$ satisfiability case (see [23]).

On the whole, it seems easier to make machines do some of the formalizing work which logicians sometimes have to do. This may be viewed as an application of the principle "Charity begins at home." Some malicious soul might use this as evidence for his favorite view that logic is trivial, and he will be wrong for too many reasons which it is tiresome to elaborate.

In general, what is needed for mechanization is not just axiomatic systems with emphasis on economy and elegance but rather "stock-of-trade systems" and formalizations which are exact and yet remain as close to good common expositions as possible.

5. Some Theoretical Difficulties

In order to mechanize proof procedures of the predicate calculus, it seems natural to use Herbrand's Theorem. This has been suggested and carried out to varying degrees of completion by different people (see [21], [22], [10], [16], [6]). The crucial part contains the generation from a given formula of a sequence of propositional or Boolean conditions, and the testing of them. It is clear, both by theoretical estimates and from results obtained so far, that (i) doing the expansion and the testing both by a brute force approach is not feasible; (ii) even greatly speeding up the testing part is not adequate to dealing with fairly interesting cases because often we have to generate a large number of Boolean conditions.

Hence, a central theoretical problem is to find ways of selecting only useful terms from each sequence of Boolean conditions. This problem has been explored in a preliminary manner in [22] and [23]. One element is to develop decision procedures for subdomains of the predicate calculus.

Another element is to use miniscope forms instead of prenex forms. A third element is to develop semidecision procedures whose range of application we do not know exactly in advance ([23], p. 30).

The decision procedures appear not to include the formulas which are of the most interest to us. More specifically, the decision procedures mostly deal with formulas in the prenex form, and when we derive a theorem from a few axioms, even though the theorem and the axioms are separately of simple forms, putting the implication (of the theorem by the axioms) in a prenex form quickly gets us outside the decidable subdomains. This suggests that we should try to extend the decision procedures to truth functions of formulas in the prenex form. Property C in Herbrand's dissertation [11] (see below) seems relevant to this question.

The semidecision procedure of [23] is not developed far enough in that the conditions under which we are to terminate the procedure are not specified explicitly. For example, if we encounter a periodic situation (a torus), we can naturally stop; but since the initial columns occupy a special place, we permit also cases while the initial columns and the others have two periods which can be fitted together. Closer examination is necessary in order to lay down broader stopping conditions.

The miniscope form defined in [22] is different from the one developed in Herbrand's dissertation because it permits the breaking up of a quantifier into several. While this permits more extensive reductions, it makes an elegant general treatment difficult. Hence, it seems desirable to return to Herbrand's treatment which is again connected intimately with his Property C and Property B.

Both for these reasons and for the additional reason that Herbrand's dissertation contains a wealth of relevant material which has been largely overlooked hitherto in the literature, we shall reproduce here in part lecture notes given at Oxford in the Michaelmas term of 1960 on Herbrand's dissertation, especially on the Properties B and C. His Property A also appears interesting and has been revived in Ackermann's book ([1], p. 93), but will not be discussed here because we do not understand fully its implications for mechanization.

6. Herbrand's Dissertation

6.1. Herbrand's System H. The primitive logical constants are $\neg$, $\vee$, $(+v)$ (or $\forall v$), $(-v)$ (or $\exists v$), with $\supset$, $\wedge$, $\equiv$ defined in the usual manner.

To avoid confusion, we shall use $p,q,r,\ldots,Fx,Gxy,\ldots$ as atomic

formulas and $X, Yx, \ldots$ as arbitrary formulas which may or may not be atomic. By a "tautology" we shall mean a formula that is always true according to the customary interpretations of truth-functional (Boolean) connectives, abstracting from an analysis of parts which contain quantifiers.

The system H contains six rules ([11], pp. 31–32).

RT. Rule of tautology. Every quantifier-free tautology is a theorem. (For example, $p \supset p$, although not $\forall x Fx \supset \forall x Fx$, falls under this.)

RI. Rules of inversion. Given a theorem X of H, we get another theorem of H if we replace within X a part which is of one of the following forms by its dual:

$$\begin{array}{ll} \neg\neg Y & Y \\ \neg(\pm v)Yv & (\mp v)\neg Yv \\ (\pm v)(Yv \vee Z) & (\pm v)Yv \vee Z \end{array}$$

Z not containing v.

RU. Rule of universal generalization. $Xxx \rightarrow +yXyy$ ("$\rightarrow$" for infer).

RE. Rule of existential generalization. $Xxx \rightarrow -yXxy$.

RC. Rule of contraction. $X \vee X \rightarrow X$.

RD. Rule of detachment (cut, modus ponens). $X, X \supset Y \rightarrow Y$.

The difference between RU and RE can be brought out by:

$$\begin{array}{l} x = x \rightarrow +y(y = y) \\ x = x \rightarrow -y(x = y) \\ x = x \not\rightarrow +y(x = y). \end{array}$$

The first important result is a direct proof of the following ([11], p. 36).

6.2. THEOREM 1. *Every tautology is a theorem of H; in other words, if we substitute quantifier expressions for parts in RT, we again get theorems of H.*

6.2.1. $X \vee \ldots \vee X \rightarrow X$.

6.3. Positive and Negative Parts. It is familiar that every formula can be brought into a prenex normal form with a matrix in the conjunctive (or disjunctive) normal form:

$$(\pm v_1)\ldots(\pm v_n)Xv_1\ldots v_n$$

such that X is in, say, a conjunctive normal form.