4

THE RESOLUTION PRINCIPLE IN THEOREM-PROVING

DAVID LUCKHAM
DEPARTMENT OF MATHEMATICS
UNIVERSITY OF MANCHESTER

INTRODUCTION

The evidence is already in favour of the computer becoming an aid to research in branches of mathematics which do not involve numerical computation. Programs have now been constructed for performing certain symbolic operations in mathematics, for example algebraic simplification of equations and indefinite integration, and for solving some of the problems (which are certainly not in the class of numerical calculations) now occurring in theoretical physics and other areas. One may reasonably expect that sooner or later programs of this type will be incorporated in a 'question-answering' package. As the construction of multi-programming systems progresses, and 'on-line' use of the computer is accepted as normal, it will become more and more practicable for the mathematician to rely on the computer for answers to routine non-numerical questions. It is, therefore, of interest from a practical as well as a theoretical point of view to ask if this development can be taken a stage further by mechanising the processes of mathematical proof. In this way the computer could be used as a tool for establishing true mathematical statements or checking proofs for correctness. The problem is essentially to produce practicable programs for proving theorems which will answer some reasonably complicated questions before overstepping the bounds of available time and memory space. This is potentially one of the most rewarding areas of computer applications, but as many people already know, it is also one of the most frustrating.

Although some research into the possibility of constructing heuristic programs based on the psychological knowledge of human problem-solving, and the role that mathematical intuition plays therein, has been reported, for example, in Newell, Shaw & Simon (1957), at present there can be little

hope of constructing a heuristic program which would be able to choose successful proof lines in say, number theory as accurately as a human mathematician. It would seem that, in the first instance, one must look for simple-minded mechanical procedures for doing mathematics, and must not rely on the activities of the human mathematician for inspiration. A natural choice for a starting point is an abstract formalisation of mathematical proof such as the first-order predicate calculus (in the following discussion we shall assume a familiarity with a formulation of this system such as is given in Hilbert & Ackermann (1950); there are many different but equivalent formulations, and we shall refer to them all under the collective title 'first-order logic'). One might hope that successful theorem-proving programs for this theory would form a basis for obtaining mechanical proofs of theorems belonging to various areas of 'real' mathematics (e.g., group theory or number theory) since in principle one would then be able to prove theorems of the various axiomatic theories formalisable within first-order logic. The results of experiments with theorem-proving programs (see, for example, Wos, Carson & Robinson 1964, 1965) would seem to support this hope, at least in the case of elementary group theory, but it is already clear that the axioms of equality will require additional strategies over and above the basic first-order procedures. And in the case of number theory it may well happen that the first-order procedures are of no use at all. However, the logical first step is to see how far first-order procedures can be improved and extended. If the extensions are significant (so that non-trivial mathematical theorems are obtained) one can expect that heuristic search principles will be used, but these may rely more on features of the basic mechanical procedure than on psychology.

At the outset of our attempt to construct a theorem-proving program we must live with the further complication that first-order logic is undecidable—a result due to Church and Turing (see Davis 1965). Essentially this means that it is *not* possible to construct a computer program which will answer correctly for every statement of first-order logic the question whether or not the statement is a theorem. Under this restriction, the best that can be done is to construct a procedure which will produce a proof of any statement that is in fact a theorem (i.e., is provable), but will continue computing for ever in an attempt to find a proof in the case when it is given a non-theorem (and we ourselves will not be able to tell in general from the state of the computation at any stage whether the procedure will ever stop). We shall refer to such procedures as *proof procedures* or *partial decision procedures.*

One positive result is the existence of sub-classes of sentences of first-order logic which *are* decidable and contain some interesting theorems of mathematics (see, e.g., Ackermann 1954). It will be possible to extend a partial decision procedure so that if it is given a sentence belonging to a decidable sub-class, it will answer 'no' in case the sentence is a non-theorem.

A recent hopeful development is a formulation of first-order logic due to Robinson (1965). Robinson's system, which involves one rule of inference

called the *Resolution Principle*, is intended to provide a basic proof procedure upon which efficient theorem-proving programs can be built, and the results so far, as reported, for example, in Wos, Carson & Robinson (1964, 1965), have been encouraging. Our purpose here is to present and justify Robinson's system, and to indicate how one may hope to derive more powerful systems from it. We shall rely heavily throughout on Robinson (1965).

HERBRAND PROCEDURES

A basic algorithm for proof procedures was given in a fundamental paper by Jacques Herbrand (1930). During the past eight years procedures using Herbrand's theorem have been programmed (see Davis, Logemann & Loveland 1962, Davis & Putnam 1960, Gilmore 1960, McIlroy, Chinlund, Davis & Hinman 1964, Wos, Carson & Robinson 1964) and used to obtain proofs of fairly simple theorems in first-order logic and elementary algebra.†

Briefly, one may describe the workings of these programs (sometimes called refutation procedures since the method used to establish a theorem is to show that its negation is inconsistent) in the following way. Suppose sentence $\mathscr{F}$ is a proposed theorem of first-order logic. We consider $\sim\mathscr{F}$, and suppose this is in prenex form (all quantifiers preceding a quantifier-free matrix) and that the matrix is in conjunctive normal form; suppose, e.g.,

$$\sim\mathscr{F} = (\exists x_1)(\forall x_2)(\exists x_3)(\forall x_4)R(x_1, x_2, x_3, x_4)$$

First, each existentially quantified variable is replaced by a Skolem function of the universally quantified variables governing it (a 0-placed function symbol is interpreted as a constant) and the universal quantifiers are dropped; this gives us

$$\mathscr{S} = R(a, x_2, f(x_2), x_4)$$

(Clearly $\sim\mathscr{F}$ is satisfiable (i.e., has a model) if and only if $\mathscr{S}$ is satisfiable, since $\mathscr{S}$ logically implies $\sim\mathscr{F}$, and if $\sim\mathscr{F}$ is true in some universe U, then there are functions over U satisfying $\mathscr{S}$.) Note that $\mathscr{S}$ is a finite conjunction of *clauses*, each clause being a disjunction of atomic formulae and negations of atomic formulae. Next, consider the set of terms, H, obtained from $\mathscr{S}$ by (i) if a is a constant (i.e., a 0-place function symbol) in $\mathscr{S}$ then a is in H; (ii) if $a_1, a_2, \ldots, a_n$ are terms in H, and f is an n-place function symbol in $\mathscr{S}$, then $f(a_1, a_2, \ldots, a_n)$ is in H. (If $\mathscr{S}$ has no constants, an arbitrary constant, a, is included in H.) We shall call H the Herbrand domain of $\mathscr{S}$. The program starts by generating substitution instances of $\mathscr{S}$ obtained by substituting terms from H for the variables in $\mathscr{S}$. For example

x_2	x_4	$R(a, x_2, fx_2, x_4)$
a	a	$R(a, a, fa, a)$
a	fa	$R(a, a, fa, fa)$
fa	a	$R(a, fa, ffa, a)$
a	ffa	$R(a, a, fa, ffa)$
...	...	...

† First-order procedures based on natural deduction systems have also been programmed, notably by H. Wang (1960b), and are discussed in this volume by R. Popplestone.

where x_2, x_4 are replaced in all possible ways by the terms $a, fa, ffa, \ldots$ in H, each variable being replaced at all of its occurrences by the same term. The set of all substitution instances of $\mathscr{S}$ obtainable by replacing each variable in $\mathscr{S}$ by a term from H, is called the Herbrand expansion of $\mathscr{S}$. At certain intervals the program tests the conjunction of the instances so far generated for inconsistency. If this conjunction is consistent, it generates further instances and tests the larger conjunction, and continues in this way until an inconsistent set of instances is found. That this will happen precisely in the case that $\mathscr{S}$ is unsatisfiable (and hence $\mathscr{F}$ is a theorem) follows from Herbrand's theorem (a simple proof is given by Quine 1955).

In most of the earlier proof procedures referred to above, the accent was understandably on improving methods of testing propositional conjunctions for inconsistency. However, the usual method of generating the instances of $\mathscr{S}$ produces all instances involving terms of a given level (or depth of nesting of function symbols) before terms of the next higher level appear. It is a simple matter to show that the number of instances of $\mathscr{S}$ increases exponentially as a function of the maximum level of terms occurring, and as a result, some elementary theorems are beyond the capacity of such procedures (see Robinson 1963). The problem therefore shifts from testing propositional forms to finding more astute ways of generating an inconsistent set of instances of $\mathscr{S}$. One proposal, due to Prawitz and Davis, was to generate instances of the clauses of $\mathscr{S}$ only if those instances were 'linked' in the following sense. A set of clauses is *linked* if whenever a literal L occurs in a clause, the complementary literal $\mathscr{S}L$ occurs in another clause. This is a necessary (but not sufficient) condition for a conjunction of a set of clauses to be inconsistent while the conjunction of any subset is consistent. Certainly a program which generates linked sets of instances stands a better chance of finding an inconsistent set before running out of memory space, and this chance will be greatly improved if good criteria can be found for deciding which of the possible linked sets to try first. A second suggestion is Robinson's Resolution Principle. It would seem that the basic resolution procedure is about as efficient as a procedure which generates linked instances of $\mathscr{S}$. However, the resolution procedure has a very concise formulation which makes the task of finding good preference criteria much easier, and this gives it a definite advantage over methods which involve generating the Herbrand expansion explicitly (see, for example, McIlroy, Chinlund, Davis & Hinman 1964, Wos, Carson & Robinson 1964, 1965).

THE RESOLUTION PRINCIPLE OF J. A. ROBINSON

In the usual formulations of first-order logic, the concept of a mathematical proof is formalised as a finite sequence of well-formed formulae (w.f.f.), each member of the sequence being related to earlier members by one of a specified set of transformations (on w.f.f.) called rules of inference; sometimes a certain set of w.f.f. is designated as 'starting formulae' (axioms). The

theorems are the end-formulae of proof sequences. Herbrand's theorem implies that it is possible to specify the transformations so that, in the case that the starting formula is the negation of a first-order theorem, a contradictory pair of formulae will appear in a proof sequence. Thus equivalent systems of first-order logic may be constructed using 'refutation rules'. A theorem is 'proved' in such a system by refuting its negation. Further, in order to prove a theorem T of a finitely axiomatisable theory with axioms $A_1, A_2, \ldots, A_n$ (expressible in first-order logic) the implication

$$(A_1 \,\&\, A_2 \,\&\, \ldots \,\&\, A_n) \rightarrow T$$

is established by refuting the conjunctive form,

$$A_1 \,\&\, A_2 \,\&\, \ldots \,\&\, A_n \,\&\sim T$$

Robinson's system is of the refutation type. We give a formulation of this system and show that it is both consistent and complete.

On the basis of the remarks in the previous section (p. 49) about Herbrand's expansion, it is sufficient to restrict the formal language so that the formulae are in Skolem free variable conjunctive form. We adopt the following formalism:

We take as primitive symbols of the language:

(i) **Variables:** $u, v, w, x, y, z, u_1, v_1, w_1, x_1, y_1, z_1, \ldots$

(ii) **Function symbols** for each degree n:
$f^n, g^n, h^n, f_1^n, g_1^n, h_1^n, \ldots$

(iii) **Predicate symbols** for each degree n:
$P^n, Q^n, R^n, P_1^n, Q_1^n, R_1^n, \ldots$

(iv) **Logical connectives** $\sim$ (not), & (and), $\vee$ (or).

Auxiliary symbols ',', '(', ')'.

We shall practise the habit of omitting the superscripts denoting degree whenever this does not confuse the issue at hand.

From among the strings composed from the above symbols, certain of them are designated as the well-formed (or meaningful) expressions of the language. These are:

(i) **Terms.** The set of terms is given by the following inductive definition:
(a) a variable is a term,
(b) if $a_1, a_2, \ldots, a_n$ are terms then $f_i^n(a_1, \ldots, a_n)$ is a term.

(ii) **Atomic formulae.** The expression $P(a_1 \ldots a_n)$ where P is a predicate symbol of degree n, and $a_1, \ldots, a_n$ are terms, is an atomic formula.

(iii) **Literals.** A literal is an atomic formula or the negation of an atomic formula.

(iv) **Clauses.** A clause is a finite disjunction of literals (possibly empty).

(v) **Conjunctive forms.** A conjunctive form is a finite conjunction of clauses.

At certain points in what follows, it will be convenient to think of a clause as being the set of the literals occurring in it, and a conjunctive form as being a set of clauses.

We assume that the well-formed expressions and sequences of well-formed expressions have a *lexical ordering*:

(*a*) the primitive symbols are ordered according to type (variables, functions, predicates, connectives) and then within each type according to superscript, then subscript, and finally by alphabetical order;

(*b*) well-formed expressions are ordered first by length, and then two expressions of equal length are placed in the order of their *i*th pair of symbols if they differ first at the *i*th position. Similarly sequences of well-formed expressions are ordered by length and then by the order of first members which differ.

For example, x comes before y in the lexical ordering, ffx before ffy, ffy before fgx, and so on. (We shall often omit the auxiliary symbols when this does not cause confusion.) Indeed given any two well-formed expressions (or sequences of such) we can decide their lexical order.

Our transformation rule depends on the basic operation of substituting terms for variables in well-formed expressions. We therefore introduce the following definitions:

If the variable v is replaced at each of its occurrences in the well-formed expression C by the term t, the resulting well-formed expression is called a *substitution instance* of C, and is denoted by $C\{(t, v)\}$. Similarly, the result of *simultaneously* replacing all occurrences of different variables $v_1, v_2, \ldots, v_n$ in C by terms $t_1, t_2, \ldots, t_n$ respectively, is denoted by $C\{(t_1, v_1), \ldots, (t_n, v_n)\}$.

Definition. A *substitution* σ is a set of ordered pairs, $\sigma = \{(t_1, v_1), \ldots, (t_n, v_n)\}$, the first element of each pair being a term and the second element a variable, such that $v_i \neq v_j$ and $t_i \neq v_i$. The result of applying σ to C, $C\sigma$, is the substitution instance $C\{(t_1, v_1), \ldots, (t_n, v_n)\}$.

Examples

Let $C = P(x, z, y) \vee P(e, f(u, y), f(a, b))$,

$\sigma = \{(e, x), (f(u, y), z), (f(a, b), y)\}$,

and $\tau = (\{f(a, b), y)\}$;

then $C\sigma = P(e, f(u, y), f(a, b)) \vee P(e, f(u, f(a, b)), f(a, b))$,

and $(C\sigma)\tau = P(e, f(u, f(a, b)), f(a, b))$

The condition, $t_i \neq v_i$ ensures that two substitutions are set-theoretically equal if and only if they have the same effect on all well-formed expressions.

The notation $P(\mathcal{S})$ denotes the set of instances obtained by applying to a set $\mathcal{S}$ of well-formed expressions all substitutions with terms belonging to a set P of terms.

First we introduce a special case of the transformation rule which applies only to expressions containing no variables; the motivation for the general rule will then be clear. Following Robinson, we call expressions containing no variables *ground* expressions. Ground terms are either 0-degree function symbols (constants) or else terms of the form $f^{(n)}(a_1, \ldots, a_n)$ where $a_1, \ldots, a_n$ are ground terms; a literal or clause is a ground expression if all of its terms are ground terms. If $\mathcal{S}$ is a conjunctive form, the Herbrand expansion of

$\mathscr{S}$ is the set $H(\mathscr{S})$ of ground instances of the clauses of $\mathscr{S}$, and this will be infinite if $\mathscr{S}$ is not a ground expression and contains at least one n-placed function symbol, $n>0$. Herbrand's theorem may be stated in the following form:

Herbrand's Theorem. *If $\mathscr{S}$ is a finite conjunction of clauses, and H is its Herbrand domain, then $\mathscr{S}$ is unsatisfiable if and only if some finite subset of $H(\mathscr{S})$ is inconsistent.*

Thus in order to test the unsatisfiability of a conjunctive form $\mathscr{S}$, it is (in theory) only necessary to test the inconsistency of a finite conjunction of the ground clauses in $H(\mathscr{S})$. One method of testing the inconsistency of a conjunction of propositional clauses (or ground clauses) is the well-known method of complementary literal elimination which Robinson calls *Ground Resolution.*

Definition. A *ground resolvent* of two ground clauses C, D is a third ground clause E obtained as follows: if there is a complementary pair of literals, L in C and $\sim L$ in D, then E is the disjunction of the set of literals,

$$(C-L) \cup (D-\sim L).$$

Notice that any pair of clauses has only finitely many resolvents, and that a resolvent of two clauses contains only literals already in the starting clauses. Hence, if one starts with a finite set of ground clauses and iterates the procedure of adding to the set of clauses so far obtained all resolvents of those clauses, the set obtained after some finite number of steps will be closed under the operation of taking resolvents.

Notation. If $\mathscr{S}$ is a set of ground clauses, $R(\mathscr{S})$ denotes the union of $\mathscr{S}$ and all ground resolvents of pairs of members of $\mathscr{S}$;

$$R^{n+1}(\mathscr{S}) = R(R^n(\mathscr{S})).$$

Now the conjunction of a set of ground clauses is consistent if it is possible to choose a ground literal from each of the clauses without being forced to choose a complementary pair of literals (i.e., a pair L, $\sim L$). Otherwise the conjunction is contradictory (truth-functionally inconsistent). We shall call a set of ground literals which does *not* contain a complementary pair a model. Thus we may say that a conjunction of ground clauses is consistent if and only if there is a model containing a member of each clause. Under this definition the empty clause, ϕ, is inconsistent, and so is any conjunction containing ϕ.

Theorem 1 (Robinson). *If $\mathscr{S}$ is a finite conjunction of ground clauses, then $\mathscr{S}$ is inconsistent if and only if $\phi \in R^n(\mathscr{S})$ for some $n \geq 0$.*

Proof. If E is a resolvent of clauses C and D, then any model containing a literal from each of C and D also contains a literal from E (since E is obtained by eliminating exactly *one* complementary pair of literals). Hence a model for $\mathscr{S}$ is also a model for $R^n(\mathscr{S})$, $n=1, 2, \ldots$. Therefore, if for some n, $\phi \in R^n(\mathscr{S})$, there is no model for $\mathscr{S}$.

On the other hand, if ϕ does not belong to $R^n(\mathcal{S})$ for any n, it is possible to construct a model for $\mathcal{S}$ as follows.

Let $L_1, L_2, \ldots, L_k$ be the atomic formulae which occur in $\mathcal{S}$ or whose negations occur in $\mathcal{S}$. Let m be such that $R^{m+1}(\mathcal{S}) = R^m(\mathcal{S})$, and let C be any clause in $R^m(\mathcal{S})$. Then define a sequence $M_1, M_2, \ldots, M_k$ of literals such that each M_t is either L_t or $\sim L_t$ according to the rule,

$$M_{t+1} = \begin{cases} L_{t+1} & \text{if no clause } C \text{ has all of its literals in the set} \\ & \{\sim M_1, \sim M_2, \ldots, \sim M_t, \sim L_{t+1}\} \\ \sim L_{t+1} & \text{otherwise} \end{cases}$$

Let M be the model $\{M_1, M_2, \ldots, M_k\}$.

Now assume that no clause is contained in $\{\sim M_1, \ldots, \sim M_t\}$ but that $C \subseteq \{\sim M_1, \ldots, \sim M_{t+1}\}$. Then $\sim M_{t+1} \in C$. By the construction $M_{t+1} = \sim L_{t+1}$ (hence $L_{t+1} \in C$) and this implies that there is a clause $D \subseteq \{\sim M_1, \ldots, \sim M_t, \sim L_{t+1}\}$. Under the assumption above, $\sim L_{t+1} \in D$; but then the resolvent $(C - L_{t+1}) \cup (D - \sim L_{t+1})$ belongs to $R^m(\mathcal{S})$ and is contained in $\{\sim M_1, \ldots, \sim M_t\}$, contradiction.

Hence if no clause is contained in $\{\sim M_1, \ldots, \sim M_t\}$, the same is true of $\{\sim M_1, \ldots, \sim M_{t+1}\}$. But a similar argument shows that no clause is contained in $\{\sim M_1\}$ since $\phi \in R^m(\mathcal{S})$. Therefore by induction M contains a literal from each clause in $R^m(\mathcal{S})$. Q.E.D.

The set $\mathcal{S}$ of ground clauses is therefore inconsistent if and only if the empty clause ϕ belongs to the set of ground clauses, $R^n(\mathcal{S})$, where $R^{n+1}(\mathcal{S}) = R^n(\mathcal{S})$; to test for inconsistency, one starts with $\mathcal{S}$ and generates $R(\mathcal{S})$, $R^2(\mathcal{S})$, . . . until either the empty clause or closure is obtained.

As was mentioned in the previous section (p. 49), the problem with a procedure of the above type, depending on testing subsets of $H(\mathcal{S})$, is not the time taken to test a subset (in the case when $\mathcal{S}$ is itself relatively simple) but that if $H(\mathcal{S})$ is generated according to the increasing complexity of the ground terms in the instances, a prohibitively large number of instances may have to be generated before an inconsistent conjunction arises. If $\mathcal{S}$ is simple, then a small inconsistent sub-conjunction can be found. The problem therefore is to avoid generating the consistent instances. One way to do this is to *predict* which instances of two clauses will contain complementary literals (below we call this a 'clash') using the information we have at hand about the way in which the terms of the Herbrand domain are constructed. We have in mind an argument of the following sort. Suppose the clauses of $\mathcal{S}$ are

(1) $P(x, e, x)$
(2) $\sim P(y, z, v) \vee \sim P(y, v, w) \vee P(e, z, w)$
(3) $P(a, f(u, v), e)$
(4) $\sim P(e, f(f(b, c), a), a)$

where a, b, c, e are constants.

Consider any instance of (1); this will 'clash' with any instance of (2) in

which v is replaced by e, and y and w are replaced by the term which replaces x in (1), for such an instance will have the form

$$\sim P(x, z, e) \vee \sim P(x, e, x) \vee P(e, z, x).$$

Thus there will be a family of resolvents of instances of (1) and (2) of the form

(5) $\sim P(x, z, e) \vee P(e, z, x)$.

Some of these clauses will 'clash' with instances of (3); namely those instances of (5) of the form

$$\sim P(a, f(u, v), e) \vee P(e, f(u, v), a),$$

the resolvents being

(6) $P(e, f(u, v), a)$. This family contains the clause

(7) $P(e, f(f(b, c), a), a)$ which clearly contradicts (4). Thus large enough segments of $H(\mathscr{S})$ are contradictory; if $H(\mathscr{S})$ is generated by the usual method, the first contradictory segment would contain between 4^6 and 20^6 clauses.

Such arguments involve applying a succession of substitutions of terms (not necessarily ground terms) for variables, each substitution being chosen so that the result of applying it to two clauses is that some of the literals in the first clause become equal (or collapse) to a literal L and some of the literals in the second clause collapse to $\sim L$. In order to formalise these arguments, we need to introduce three simple concepts: (i) the operation of composition of substitutions, (ii) the concept of a 'unifying' substitution, and (iii) the concept of a simplest substitution having a given property.

Definition. Let θ, ψ be substitutions.

The *composition* of θ and ψ (denoted by $\theta\psi$) is the substitution defined as follows:

Let $\theta = \{(t_1, u_1), \ldots, (t_n, u_n)\}$,
$\theta' = \{(t_1\psi, u_1), \ldots, (t_n\psi, u_n)\}$, such that $t_i\psi \neq u_i$,

and let ψ' consist of those pairs of ψ whose second elements do not occur among the variables $u_1, \ldots, u_n$ of θ, say

$$\psi' = \{(s_1, v_1), \ldots, (s_k, v_k)\}.$$

Then

$$\theta\psi =_{df} \theta' \cup \psi'.$$

Remarks. It is clear from this definition that the result of applying θ and ψ successively to a well-formed expression C is the same as applying $\theta\psi$, i.e.,

$$(C\theta)\psi = C\theta\psi.$$

Further, composition of substitutions is associative,

$$(\theta\sigma)\psi = \theta(\sigma\psi).$$

Definition. Let $\mathscr{L} = \{L_1, L_2, \ldots, L_n\}$ be a set of literals. $\mathscr{L}$ is *unified* (Robinson's terminology) by σ if

$$L_1\sigma = L_2\sigma = \ldots = L_n\sigma$$

Remark. If σ unifies L, so does $\sigma\tau$ where τ is any substitution.

Definition. θ is a *simplest substitution* satisfying a condition C if for any σ satisfying C there is a λ such that

$$\sigma = \theta\lambda$$

Remark. Any pair of simplest substitutions (satisfying a given condition) differ by a change of variables. For, if $\theta = \sigma\lambda$ and $\sigma = \theta\psi$, it is easy to see that θ and σ must be of the form

$$\theta = \{(s_1, u_1), \ldots, (s_n, u_n)\} \text{ and}$$
$$\sigma = \{(s_1\psi, u_1), \ldots, (s_n\psi, u_n), (t_1, v_1) \ldots (t_m, v_m)\}$$

where $s_i = s_i\psi\lambda$ or $s_i\psi = u_i$, and $t_i\lambda = v_i$, which can only happen if ψ substitutes variables for variables.

In many cases there will be no simplest substitution satisfying the given condition. However, we shall show that if a set $\mathscr{L}$ of literals is unifiable, then there is a simplest substitution unifying $\mathscr{L}$ and, moreover, that there is an algorithm for finding such a substitution. In the definition below we need to pick one of the simplest substitutions unifying a set of literals; it does not matter which one, but that only one is chosen. We could for example choose the earliest one in the lexical ordering of sequences of well-formed expressions by considering the pairs in a substitution to be arranged as a sequence in the order of the second elements. We shall therefore assume that we have a method for choosing a unique simplest substitution unifying a given unifiable set of literals. It is also necessary to fix upon a method of changing the variables in a pair of clauses so that no variable occurs in both; it is not essential that this be the method given below.

We now extend the concept of a resolvent of two clauses to the case where the clauses are not necessarily ground clauses.

Definition. A resolvent of two clauses C_1 and C_2 is a third clause D obtained as follows:

(i) If $v_1, \ldots, v_m$ are the variables of C_2, and the highest variable of C_1 in the lexical order is u_k, let $\theta = \{(u_{k+1}, v_1), \ldots, (u_{k+m}, v_m)\}$. (None of the variables in $C_2\theta$ occurs in C_1.)

(ii) If there is a pair of sets of literals, $\mathscr{L} = \{L_1, \ldots, L_k\}$ and $\mathscr{M} = \{M_1, \ldots, M_n\}$ such that $\mathscr{L} \subseteq C_1$, and $\mathscr{M} \subseteq C_2$ and the set $\{L_1, \ldots, L_k, \sim M_1\theta, \ldots, \sim M_n\theta\}$ is unifiable, let σ_0 be the chosen simplest unifying substitution so that $\mathscr{L}\sigma_0$ and $\mathscr{M}\theta\sigma_0$ are complementary literals; then D is the disjunction of the literals,

$$(C_1 - \mathscr{L})\sigma_0 \cup (C_2 - \mathscr{M})\theta\sigma_0$$

Remarks. (i) Any pair of clauses have (at most) a finite number of resolvents because there are only finitely many pairs of sets of literals, $\mathscr{L}, \mathscr{M}$, and for each pair at most one substitution, σ_0. (ii) The resolvents of a pair of ground clauses are the ground resolvents.

The transformation rule or rule of inference for our system of logic, which Robinson calls the Resolution Principle, is the following:

Resolution Principle. From any two clauses C_1 and C_2, infer a resolvent of C_1 and C_2.

Finally, a 'proof' or, more accurately, a refutation of a conjunctive form $\mathscr{S}$ is a sequence $C_1, C_2, \ldots, C_n$ of clauses such that each clause either belongs to $\mathscr{S}$ or is inferred from two earlier clauses by a resolution, and $C_n = \phi$.

Our system of logic is now completely specified. We prove that it is consistent and complete, i.e., $\mathscr{S}$ is unsatisfiable if and only if there is a refutation of $\mathscr{S}$.

As before, let $R(\mathscr{S})$ denote the set of all clauses in $\mathscr{S}$ and all resolvents of pairs of clauses in $\mathscr{S}$; $R^{n+1}(\mathscr{S}) = R(R^n(\mathscr{S}))$. There is a refutation of $\mathscr{S}$ if and only if $\phi \in R^n(\mathscr{S})$ for some n.

Remark. If σ_0 is a simplest substitution unifying a set $\mathscr{L}$ of literals, then all of the function symbols of the terms of σ_0 occur in the terms of $\mathscr{L}$. This will be proved later.

Theorem 2. *If $\phi \in R^n(\mathscr{S})$ for some n, then $\mathscr{S}$ is unsatisfiable.*

Proof. Let $\mathscr{R}$ denote resolution in the extended sense that in the definition of resolvent above, $\mathscr{L}$ or $\mathscr{M}$ may be empty. Thus $R(\mathscr{S}) \subseteq \mathscr{R}(\mathscr{S})$. This in turn imposes a corresponding trivial extension on ground resolution in which additional resolvents of the form $A \cup B$ or $A \cup (B - L)$ are introduced. These are already redundant and it is easy to see in the case when $\mathscr{S}$ is a set of ground clauses, that if $\phi \in \mathscr{R}^n(\mathscr{S})$ then $\phi \in R^n(\mathscr{S})$. We make this extension only to facilitate the following proof.†

Let H be the Herbrand domain of $\mathscr{S}$.

First we prove $H(\mathscr{R}(\mathscr{S})) \subseteq \mathscr{R}(H(\mathscr{S}))$. (1)

If $C \in \mathscr{S}$ then clearly $H(C) \subseteq \mathscr{R}(H(\mathscr{S}))$.

Suppose C is a resolvent (in the extended sense).

Let $C = (A - \mathscr{L})\sigma_0 \cup (B - \mathscr{M})\theta\sigma_0$ where $A, B \in \mathscr{S}$ and either $\mathscr{L}\sigma_0$, $\mathscr{M}\theta\sigma_0$ are complementary literals or at least one of $\mathscr{L}, \mathscr{M}$ is empty.

For any substitution τ,

$$\begin{aligned} C\tau &= (A - \mathscr{L})\sigma_0\tau \cup (B - \mathscr{M})\theta\sigma_0\tau \\ &= (A\sigma_0\tau - \mathscr{L}'\sigma_0\tau) \cup (B\theta\sigma_0\tau - \mathscr{M}'\theta\sigma_0\tau) \end{aligned}$$

where $\mathscr{L}' = \mathscr{L}$ or $\mathscr{L}' = \phi$ and $\mathscr{M}' = \mathscr{M}$ or $\mathscr{M}' = \phi$ depending on whether $\mathscr{L}$ contains all literals L in A, such that $L\sigma_0\tau = \mathscr{L}\sigma_0\tau$ and $\mathscr{M}$ contains all literals M in B such that $M\theta\sigma_0\tau = \mathscr{M}\theta\sigma_0\tau$.

Let τ be a substitution from H such that $C\tau \in H(\mathscr{R}(\mathscr{S}))$. Then there is a τ' such that $\tau \subseteq \tau'$, $C\tau = C\tau'$, and $A\sigma_0\tau'$, $B\theta\sigma_0\tau' \in H(\mathscr{S})$. (Essentially, τ' substitutes constants from H for any variables remaining in the terms of $A\sigma_0\tau$ and $B\theta\sigma_0\tau$; by the remark above, $\sigma_0\tau'$ and $\theta\sigma_0\tau'$ are also substitutions from H.)

Either $\mathscr{L}'\sigma_0\tau' = \sim\mathscr{M}'\theta\sigma_0\tau'$ or one of $\mathscr{L}'$, $\mathscr{M}'$ is empty.

Therefore $C\tau \in \mathscr{R}(H(\mathscr{S}))$.

† Resolution does in fact generate such redundant clauses, e.g., if $\mathscr{S}$ is the two-clause set, $\{(P(x, x), P(x, a)), (\sim P(x, x), \sim P(x, a))\}$, then $H = \{a\}$ and $R(H(\mathscr{S})) = \{(P(a, a)), (\sim P(a, a)), \emptyset\}$ whereas $H(R(\mathscr{S})) = \{(P(a, a)), (\sim P(a, a)), (P(a, a), \sim P(a, a)), \emptyset\}$.

By the above remark, the function symbols in $\mathscr{R}^n(\mathscr{S})$ already occur in $\mathscr{S}$ and hence H is also the Herbrand domain of $\mathscr{R}^n(\mathscr{S})$.

If we assume $H(\mathscr{R}^n(\mathscr{S})) \subseteq \mathscr{R}^n(H(\mathscr{S}))$,

then $H(\mathscr{R}^{n+1}(\mathscr{S})) \subseteq \mathscr{R}(H(\mathscr{R}^n(\mathscr{S})))$ by (1),

$$\subseteq \mathscr{R}(\mathscr{R}^n(H(\mathscr{S}))).$$

Therefore $H(\mathscr{R}^n(\mathscr{S})) \subseteq \mathscr{R}^n(H(\mathscr{S}))$ for all n, by induction.

If $\phi \epsilon R^n(\mathscr{S})$ then $\phi \epsilon H(R^n(\mathscr{S}))$; by the above result $\phi \epsilon \mathscr{R}^n(H(\mathscr{S}))$ and hence $\phi \epsilon R^n(H(\mathscr{S}))$. This implies that there is a finite subset $Q \subseteq H(\mathscr{S})$ containing at most 2^n clauses such that $\phi \in R^n(Q)$. Q is inconsistent (Theorem 1), and therefore $\mathscr{S}$ is unsatisfiable (Herbrand's Theorem). Q.E.D.

Theorem 3 (Robinson). *If $\mathscr{S}$ is unsatisfiable then $\phi \in R^n(\mathscr{S})$ for some n.*

Proof. We prove first that if $P \subseteq H$ then $R(P(\mathscr{S})) \subseteq P(R(\mathscr{S}))$. (1)

Suppose $C \in R(P(\mathscr{S}))$.

If $C \in P(\mathscr{S})$ then clearly $C \in P(R(\mathscr{S}))$.

If C is a resolvent, then

$$C = (A\alpha - L\alpha) \cup (B\beta - M\beta)$$

where A, $B \in \mathscr{S}$, $A\alpha$, $B\beta \in P(\mathscr{S})$, and $L\alpha$, $M\beta$ are complementary literals.

Let
$$\alpha = \{(s_1, u_1), \ldots, (s_k, u_k)\} \text{ and}$$
$$\beta = \{(t_1, v_1), \ldots, (t_m, v_m)\}$$

where $u_1, \ldots, u_k$ are all the variables of A and $v_1, \ldots, v_m$ are all the variables of B. Let θ be the change of variables, $\theta = \{(u_{k+1}, v_1), \ldots, (u_{k+m}, v_m)\}$ and let $\sigma = \{(s_1, u_1), \ldots, (s_k, u_k), (t_1, u_{k+1}), \ldots, (t_m, u_{k+m})\}$. Then $A\sigma = A\alpha$, $B\theta\sigma = B\beta$, and $C = (A - \mathscr{L})\sigma \cup (B - \mathscr{M})\theta\sigma$ where $\mathscr{L}\sigma = L\alpha$ and $\mathscr{M}\theta\sigma = M\alpha$. Furthermore, since σ unifies the set $\mathscr{L} \cup \sim\mathscr{M}\theta$ there is a chosen simplest unifying substitution σ_0 and a resolvent C' of A and B, where $C' = (A - \mathscr{L})\sigma_0 \cup (B - \mathscr{M})\theta\sigma_0$. Let $\sigma = \sigma_0\lambda$. Then $C = C'\lambda$. Thus $C \in P(R(\mathscr{S}))$.

The general form of (1) now follows by induction:

Assume $R^n(P(\mathscr{S})) \subseteq P(R^n(\mathscr{S}))$

Then
$$R^{n+1}(P(\mathscr{S})) = R(R^n(P(\mathscr{S})))$$
$$\subseteq R(P(R^n(\mathscr{S})))$$
$$\subseteq P(R^{n+1}(\mathscr{S})) \text{ by (1)}$$

If $\mathscr{S}$ is unsatisfiable, a finite subset of $H(\mathscr{S})$ is inconsistent. Suppose this is $P(\mathscr{S})$. Then $\phi \in R^n(P(\mathscr{S}))$ for some n (Theorem 1). Therefore $\phi \in P(R^n(\mathscr{S}))$.

This implies $\phi \in R^n(\mathscr{S})$ since substituting terms for variables cannot reduce a non-empty clause to the empty clause. Q.E.D.

Finally we must show that there is an algorithm for choosing a simplest substitution unifying a set of literals. An algorithm is given by Robinson (1965), and this seems to be as straightforward and efficient as one can hope for. This procedure starts with the empty substitution and builds up step by step a simplest σ_0 which unifies the set $\mathscr{L}$ of literals. If at the kth step

the substitution so far obtained is σ_k, and the literals $L_1\sigma_k, \ldots, L_n\sigma_k$ in $\mathscr{L}\sigma_k$ are not all equal, the procedure changes σ_k on the basis of a 'disagreement list' containing the first well-formed expression in each $L_i\sigma_k$ which needs to be changed. Consider each $L_i\sigma_k$ to be a sequence of symbols. Define the *disagreement set* of $\mathscr{L}\sigma_k$ to be the set of all well-formed sub-expressions of the literals in $\mathscr{L}\sigma_k$ which begin at the first symbol position at which not all the literals have the same symbol. This set is either $\mathscr{L}$ itself (in which case $\mathscr{L}$ is not unifiable) or else it contains a term or sub-term from each literal in $\mathscr{L}\sigma_k$ (e.g., the disagreement set of $\{P(x, f(x, g(y)), v), P(x, f(x, z), w)\}$ is $\{z, g(y)\}$).

Robinson's unification procedure is as follows:

(1) Set $\sigma_1 = \phi$ and go to (2).

(2) If the elements of $\mathscr{L}\sigma_k$ are all equal, set $\sigma_0 = \sigma_k$ and stop; otherwise go to (3).

(3) Let s_k, t_k be the two earliest expressions in the lexical ordering of the disagreement set of $\mathscr{L}\sigma_k$; if s_k is a variable and does not occur in t_k, set $\sigma_{k+1} = \sigma_k\{(t_k, s_k)\}$ and go to (2); otherwise stop.

Since $\mathscr{L}\sigma_{k+1}$ contains one less variable than $\mathscr{L}\sigma_k$, the procedure must stop in at most m steps if $\mathscr{L}$ has m variables. Also it is clear that if the procedure stops at stage (2), σ_0 is uniquely determined and its terms contain only function symbols occurring in $\mathscr{L}$. It remains for us to show that if $\mathscr{L}$ is unifiable then σ_0 is defined and is a simplest unifying substitution.

Theorem 4 (Robinson). *If λ is any substitution unifying set $\mathscr{L}$ of literals, then the unification procedure applied to $\mathscr{L}$ stops at stage* (2) *and $\lambda = \sigma_0\tau$ for some τ.*

Proof. By induction on the number of steps taken by the procedure before it stops. Assume that at the kth step the procedure is at stage (2) and there is a τ_k such that $\lambda = \sigma_k\tau_k$. (This is clearly true at the first step since $\lambda = \phi\lambda$.) There are two cases. If σ_k unifies $\mathscr{L}$ then $\sigma_0 = \sigma_k$ and $\lambda = \sigma_0\tau_k$. Otherwise the disagreement set of $\mathscr{L}\sigma_k$ contains at least two expressions and these have different first symbols. Now $\sigma_k\tau_k$ unifies $\mathscr{L}$. Therefore τ_k unifies this disagreement set. But two expressions with different first symbols cannot be unified unless one of the expressions is a variable which does not occur in the other. Since variables are earliest in the lexical ordering, s_k and t_k satisfy the conditions of stage (3) so that the procedure does not stop there, and $\sigma_{k+1} = \sigma_k\{(t_k, s_k)\}$.

Now let τ_{k+1} be obtained from τ_k by deleting any pair containing s_k as a second element. $\tau_{k+1} = \tau_k - \{(s_k\tau_k, s_k)\}$.

Then
$$\begin{aligned}\tau_k &= \{(s_k\tau_k, s_k)\} \cup \tau_{k+1} \\ &= \{(t_k\tau_k, s_k)\} \cup \tau_{k+1}\end{aligned}$$

since τ_k unifies the disagreement set,

$$= \{(t_k\tau_{k+1}, s_k)\} \cup \tau_{k+1}$$

since s_k does not occur in t_k,

$$= \{(t_k, s_k)\}\tau_{k+1}$$

Therefore $\lambda = \sigma_{k+1}\tau_{k+1}$ and the procedure returns to stage (2). Q.E.D.

It is a simple matter to construct examples showing that the choice of a simplest substitution in the definition of resolution is crucial for the completeness of the system, though not for consistency. However it is likely that the definition may be restricted to 'pairwise' resolution in which the sets $\mathscr{L}$ and $\mathscr{M}$ each contain a single literal; the completeness proof should go through as before with minor complications.

As it stands, the basic resolution procedure of repeatedly computing resolvents, resolvents of resolvents, and so on, until the empty clause occurs, is still very inefficient. Some strategies for choosing more likely inferences first, and for cutting down redundant inferences, are easy to formulate. For example, one would naturally give preference to resolvents of 'short' clauses; the only guarantee that a resolvent is shorter than its parent clauses is that one of the parents is a unit clause. Again, some simple strategies will eliminate inferences from some of the consistent subsets of the starting set of clauses. Such strategies have been successfully used on simple algebraic theorems (see Wos, Carson & Robinson 1964, 1965), but it must be mentioned that in these cases the starting set contains two or three unit clauses. In more complex cases, more involved strategies and probably heuristic principles (which will destroy the completeness of the system) will have to be found.

One line of attack is to use the fact that if there is a deduction of ϕ from $\mathscr{S}$, then there is a deduction which denies a possible model M for $H(\mathscr{S})$ by using a clause satisfied by M as a parent for each sub-deduction. Using this, Robinson (to be published) has shown that in certain cases the procedure need only be applied to a subset of $R^n(\mathscr{S})$. It is not yet known how far such improvements to the basic resolution procedure can be taken and we can only state that much theoretical and experimental work needs to be done.

REFERENCES

Ackermann, W. (1954). *Solvable cases of the decision problem.* Amsterdam: North-Holland.

Chinlund, T. J., Davis, M., Hinman, P. G., & McIlroy, D. (1964). *Theorem-proving by matching.* Bell Laboratories. Spring 1964.

Davis, M., ed. (1965). *The undecidable.* New York: Raven.

Davis, M., Logemann, G. & Loveland, D. (1962). A machine program for theorem-proving. *Communs Ass. comput. Mach.*, **5**, 394-397.

Davis, M., & Putnam, H. (1960). A computing procedure for quantification theory. *J. Ass. comput. Mach.*, **7**, 201-215.

Gilmore, P. C. (1960). A proof method for quantification theory. *IBM Jl Res. Dev.*, **4**, 28-35.

Herbrand, J. (1930). Recherches sur la théorie de la demonstration. *Pr. Tow. nauk. warsz. (b).* **33.**

Hilbert, D., & Ackermann, W. (1950). *Principles of mathematical logic.* New York: Chelsea.

Newell, A., Shaw, J. C., & Simon, H. A. (1957). Empirical explorations of the logic theory machine. *Proc. west. jt Computer Conf.*, pp. 218-239.

Popplestone, R. J. (1966). Pp. 31-46 of this volume.

Prawitz, D. (1960). An improved proof procedure. *Theoria.*, **26**, 102-139.

Quine, W. V. O. (1955). A proof procedure for quantification theory. *J. of Sym. Logic*, **20**, 141-149.

Robinson, J. A. (1963). Theorem-proving on the computer. *J. Ass. comput. Mach.*, **10**, 163-174.

Robinson, J. A. (1965). A machine-oriented logic based on the resolution principle. *J. Ass. comput. Mach.*, **12**, 23-41.

Robinson, J. A. Automatic deduction with hyper-resolution. (To be published.)

Wang, H. (1960*a*). Towards mechanical mathematics. *IBM Jl Res. Dev.*, **4**, 2-22.

Wang, H. (1960*b*). Proving theorems by pattern recognition—I. *Communs Ass. comput. Mach.*, **3**, 220-234.

Wang, H. (1960*c*). Proving theorems by pattern recognition—II. *Bell Syst. tech. J.*, **40**, 1-41.

Wos, L., Carson, D. F., & Robinson, J. A. (1964). The unit preference strategy in theorem proving. *AFIPS Conference proceedings*—1964 *Fall Joint Computer Conference*, **26**, Pt 1, 615-621. Washington: Spartan.

Wos, L., Carson, D. F. & Robinson, G. A. (1965)., Efficiency and completeness of the set of support strategy in theorem proving. *J. Ass. comput. Mach.*, **12**, 536-541.