HERBRAND'S FUNDAMENTAL THEOREM AND THE BEGINNING OF LOGIC PROGRAMMING*

FRANCINE ABELES

Department of Mathematics & Computer Science
Kean College, SCNJ
Union, NJ 07083, USA

INTRODUCTION.

In this paper we argue that in his 1930 thesis, Jacques Herbrand developed the concept of unification central to automatic theorem proving and logic programming, not peripherally as has been suggested, but as an element essential to the proof of his Fundamental Theorem, hereafter abbreviated, FT.

Excellent surveys of Herbrand's work can be found by Goldfarb in [Herbrand *1971*, 1–20], in Jean van Heijenoort [*1967*, *525–529*; *1968*; *1986*], and Irving Anellis [*1991*; *1992*]. Herbrand set out to develop a unified approach to proof theory. His method of investigation, involving the notions of completeness, consistency, and decidability, was directed toward answering the question: what finite sense can generally be ascribed to the truth property of a formula with quantifiers, particularly the existential quantifier, in an infinite universe? The existential quantifier, interpreted as standing for a choice function, posed the main difficulty. Since it is not generally replaceable by a computable function, it is not always possible to constructively instantiate these quantified variables.

The major influence on Herbrand's development of unification came from Russell and Whitehead's *Principia Mathematica*, hereafter abbreviated, PM. Herbrand, like Hilbert, used PM as the example that classical mathematics can be codified and presented as a formal system. His key concept of a normal identity and what he refers to as Property *A*, are derived from a method used by Russell and Whitehead to construct quantificational logic. This method appears in Herbrand's 1928 paper, "On Proof Theory" [Herbrand, 29–32]. As Goldfarb [Herbrand, 4] writes,

* The author is indebted to the editor, Irving Anellis, for his critical reading of this manuscript and his constructive suggestions which have enormously improved the manuscript.

> Herbrand applied the approach of the Hilbert school to pure quantification theory; his goal was to analyze quantificational provability in terms of truth-functional validity.... Herbrand's solution to this problem [of instantiating quantified variables] arises from two ways of connecting quantified formulas with quantifier-free ones. The first is an extension of Russell and Whitehead's construction of quantified logic in PM, and rests on a procedure for obtaining the proof of a formula *P* containing quantifiers from that of a quantifier-free tautology (that is, truth-functionally valid formula) related to *P* in a canonical way.

The second way, constructing a finite model, is based on the work of Lōwenheim and Skolem on quantifer-free formulas. It is important to the development of Herbrand's FT, but does not directly bear on unification.

DEFINITIONS.

Before proceeding further, we need definitions of the following terms: A *clause* is a finite disjunction of literals or the empty clause. A *literal* is a predicate followed by a list of terms or arguments. A *term* is a constant, variable or function symbol followed by 0...*n* terms denoting the degree of the function. Terms of degree 0 are *constants.* Any well-formed-formula (WFF) in the first order predicate logic can be converted to *clausal form* which can be thought of as an extension of conjunctive normal form. A literal having no variables is a *ground literal,* and a clause whose members are only ground literals is a *ground clause.* If we associate with the set of clauses *S*, the set of all terms containing only function symbols *F* in *S* of degree 0, otherwise the function symbols $\{a\} \cup F$, where *a* is a constant, we have the *Herbrand Universe of S*. Herbrand's FT implies that a set *S* of clauses is unsatisfiable (i.e. inconsistent or self-contradictory) if and only if there exists a finite subset *P* of the Herbrand Universe of *S* that is truth-functionally unsatisfiable. Van Heijenoort [*1992*, 247–248] provides an illustration of the expansion method Herbrand employed to obtain sententially valid quantifier-free formulas from satisfiable quantified formulas.

UNIFICATION AND RESOLUTION.

In 1963 J. Alan Robinson invented resolution, a single inference method for first order logic. Resolution, first announced by Robinson in 1963 in an abstract in the *Journal of Symbolic Logic*, was developed in his 1965 article. Resolution is a refutation method operating on clauses containing function symbols, universally quantified variables and constants. Quantified variables can be handled in a deductive system if there are tests of unifiability that act like tests of equality. The essence of the resolution method is that it searches for local evidence of unsatisfiability in the form of a pair of clauses, one containing a literal and the other its complement (negation). Herbrand's FT permits clauses with variables to stand for all their ground instances, so that resolving two of these clauses requires resolving all their ground instances. The simulation of ground resolution is the essence of unification. In automated deductive systems, a unification algorithm is the basis of resolution.

Robinson [*1992*, 43–44] writes,

> Dag Prawitz [*1960*] had also forcefully advocated the use of the process which we now call unification..., he apparently had independently rediscovered unification in the late 1950's. He apparently did not realize it had already been introduced by Herbrand in his thesis of 1930 (albeit only in a brief and rather obscure passage).... I was immediately very impressed by the significance of this idea. It is essentially the idea underlying Herbrand's 'Property *A* Method' developed in the same thesis.

In a recent article, Jean-Pierre Jouannaud and Claude Kirchner give their view of the connections between Herbrand's and Robinson's work. They write [Jouannaud and Kirchner, 257],

> Solving equations on first-order terms emerged with Herbrand's work on proof theory and was coined unification by Alan Robinson.... Invented by Alan Robinson, resolution was the first really effective mechanization of first-order logic.... Unification bridges the conceptual gap between ground and non-ground atoms [i.e. literals] by computing a representative of all their common instances. Alan Robinson gave the first algorithm ever for computing such a representative, calling it a most general unifier [MGU], and showed its uniqueness (up to an equivalence).

UNIFICATION ALGORITHMS.

Herbrand never used the name unification algorithm for his equation solving process. His algorithmic method transforms a solvable equation set, one that has a solution of the form $x_j = t_j, \ldots, x_k = t_k$, where the x_j are variables that do not occur on the right hand side of any equation and the t_j are terms, into an equivalent solved form equation set. For any unsolvable equation set the algorithm halts with failure. The definition of a set of equations in solved form is given by [Martelli and Montanari, 261] as satisfying the conditions: (1) the equations are $x_j = t_j$, $j = 1, \ldots, k$ and (2) every variable which is the left member of some equation occurs only there.

Herbrand's algorithm appears as follows [Herbrand, 148]:

Now, to find an appropriate set of associated equations is easy, if such a set exits; it suffices, for each system of equations between arguments, to proceed by recursion, using one of the following procedures, which simplify the system of equations to be satisfied.

(1) If one of the equations to be satisfied equates a restricted variable x to an individual, either this individual contains x ⟦or some other restricted variable⟧, and then the equation cannot be satisfied, or else the individual does not contain x ⟦or any other restricted variable, or any general variable that is not superior to x⟧, and then the equation will be one of the associated equations that we are looking for; in the other equations to be satisfied we replace x by the individual;

(2) If one of the equations to be satisfied equates a general variable to an individual that is not a restricted variable, the equation cannot be satisfied;

(3) If one of the equations to be satisfied equates $f_1(\phi_1, \phi_2, ..., \phi_n)$ to $f_2(\psi_1, \psi_2, ..., \psi_m)$, either the elementary functions f_1 and f_2 are different, and then the equation cannot be satisfied, or they are the same, and then we turn to those equations that equate the ϕ_i to the ψ_i.

Therefore, if we successively consider each prenex form of P, we shall be able, after a finite and determinate number of steps, to decide whether the proposition P is a normal identity.

Similarly, given a proposition P and any scheme, we can test whether a proposition P' derived from P by the scheme is a normal identity, hence whether the scheme permits us to show that P has property A.

In 1982, Alberto Martelli and Ugo Montanari described the unification problem in first-order logic in this way: Find the simplest substitution, i.e. the assignment of some term to each variable, for two given terms containing some variables. This substitution, if it exists, is a MGU. They present the unification problem as the solution of a set of equations, and follow it by an example [Martelli and Montanari, 261–262]:

A set of equations is said to be *in solved form* iff it satisfies the following conditions:

(1) the equations are $x_j = t_j, j = 1, ..., k$;
(2) every variable which is the left member of some equation occurs only there.

A set of equations in solved form has the obvious unifier

$$\vartheta = \{(t_1, x_1), (t_2, x_2),...,(t_k, x_k)\}$$

If there is any other unifier, it can be obtained as

$$\sigma = \{(t_{1_\alpha}, x_1), (t_{2_\alpha}, x_2), \ldots, (t_{k_\alpha}, x_k)\} \cup \alpha$$

where α is any substitution instance which does not rewrite variables $x_1, \ldots, x_k$. Thus ϑ is called a *most general unifier* (*mgu*).

The following nondeterministic algorithm shows how a set of equations can be transformed into an equivalent set of equations in solved form.

Algorithm 1

Given a set of equations, repeatedly perform any of the following transformations. If no transformation applies, stop with success.

(a) Select any equation of the form

$$t = x$$

where t is not a variable and x is a variable, and rewrite it as

$$x = t$$

(b) Select any equation of the form

$$x = x$$

where x is a variable, and erase it.

(c) Select any equation of the form

$$t' = t''$$

where t' and t'' are not variables. If the two root function symbols are different, stop with failure; otherwise, apply term reduction.

(d) Select any equation of the form

$$x = t$$

where x is a variable which occurs somewhere else in the set of equations and where $t \neq x$. If x occurs in t, then stop with failure; otherwise apply variable elimination.

As an example, let us consider the following set of equations:

$$\begin{aligned} g(x_2) &= x_1; \\ f(x_1, h(x_1),(x_2) &= f(g(x_3), x_4, x_3). \end{aligned}$$

By applying transformation (c) of Algorithm 1 to the second equation we get

$$\begin{aligned} g(x_2) &= x_1; \\ x_1 &= g(x_3); \\ h(x_1) &= x_4; \\ x_2 &= x_3. \end{aligned}$$

By applying transformation (d) to the second equation we get

$$\begin{aligned} g(x_2) &= g(x_3); \\ x_1 &= g(x_3); \\ h(g(x_3)) &= x_4; \\ x_2 &= x_3. \end{aligned}$$

We now apply transformation (c) to the first equation and transformation (a) to the third equation:

$$\begin{aligned} x_2 &= x_3; \\ x_1 &= g(x_3); \\ x_4 &= h(g(x_3)); \\ x_2 &= x_3. \end{aligned}$$

Finally, by applying transformation (d) to the first equation and transformation (b) to the last equation, we get the set of equations in solved form:

$$\begin{aligned} x_2 &= x_3; \\ x_1 &= g(x_3); \\ x_4 &= h(g(x_3)). \end{aligned}$$

Therefore, an mgu of the system is

$$\vartheta = \{(g(x_3), x_1), (x_3, x_2), (h(g(x_3)), x_4)\}.$$

The following theorem proves the correctness of Algorithm 1.

THEOREM 2.3. *Given a set of equations S,*

(i) Algorithm 1 *always terminates, no matter which choices are made.*

> *(ii) If Algorithm* 1 *terminates with failure, S has no unifier. If Algorithm* 1 *terminates with success, the set S has been transformed into an equivalent set in solved form.*

Like Jouannaud and Kirchner they credit unification to Robinson. "Unification was first introduced by Robinson as the central step of the inference rule called resolution" [Martelli and Montanari, 258].

In his Unification Theorem

> UNIFICATION THEOREM. *Let A be any finite nonempty set of well-formed expressions. If A is unifiable, then A is most generally unifiable with most general unifier* σ_A; *moreover, for any unifier* θ *of* A *there is a substitution* λ *such that* $\theta = \sigma_A\lambda$.
>
> PROOF. It will suffice to prove that under the hypotheses of the theorem the Unification Algorithm will terminate, when applied to A, at step 2; and that for each $k \geq 0$ until the Unification Algorithm so terminates, the equation $\theta = \sigma_k\lambda_k$ holds at step 2 for some substitution λ_k,

Robinson was the first to establish the correctness of the Unification Algorithm

> UNIFICATION ALGORITHM. The following process, applicable to any finite nonempty set A of well-formed expressions, is called the Unification Algorithm:
>
> *Step* 1. Set $\sigma_0 = \epsilon$ and $k = 0$, and go to step 2.
>
> *Step* 2. If $A\sigma_k$ is not a singleton, go to step 3. Otherwise, set $\sigma_A = \sigma_k$ and terminate.
>
> *Step* 3. Let V_k be the earliest, and U_k the next earliest, in the lexical ordering of the disagreement set B_k of $A\sigma_k$. If V_k is a variable, and does not occur in U_k, set $\sigma_{k+1} = \sigma_k \{U_k / V_k\}$, add 1 to k, and return to step 2. Otherwise, terminate.

and systematically relate a set of equations in solved form with a MGU [Robinson, *1965*, 32–34]. In both Robinson's and Martelli/Montanari's algorithms, when the set of equations in solved form is produced, its solution is a MGU of the system. To present all the necessary preliminaries for Robinson's algorithm is not feasible, but we can observe as Donald Loveland and Hao Wang did, that resolution depends on Gentzen's cut rule which, as Wang added, ensures the complexity of each step in a proof is not more complex than its conclusion [Loveland *1984*, 8; Wang *1970*, 228]. To understand how the algorithm works, consider two lists l_1, l_2 containing the actual parameters of the activation call and the formal parameters of the selected procedure. A third list t acts as a temporary, recording the assignments made when the contents of l_1 and l_2 are compared. When the algorithm is successful, the final state of t is a MGU for that step. If the algorithm fails, no unifier exists. At each step, two well-formed expressions (WFF's) are compared in one of the following ways determined by the form of the expressions: If both expressions are terms with the same principal function symbols, their arguments are added to l_1 and l_2 appropriately; if both expressions are terms with different principal function symbols, the algorithm fails. If either expression is a variable occurring in the other, the algorithm fails; if either expression is a variable not occurring in the other, the first expression is assigned the value of the second, thereby updating t. If either expression is a constant and the other is either a different constant or a term with function symbols, the algorithm fails.

HERBRAND'S FT.

Herbrand's test of when a proposition is a normal identity, a quantifier-free propositional identity obtained from certain provability rules, a stronger version of which he had discussed in his 1928 paper [Herbrand, 29–34], involves finding appropriate equations between arguments. He used his algorithm to obtain a simple form of those equations.

Herbrand developed property *A* to deal with propositions containing quantifiers. In Chapter 2 of his thesis, he proved that all quantified tautologies are normal identities (strong form), and implicitly that all quantified tautologies have a special case of property *A*. In his FT, he provided a constructive proof that a proposition is provable in his quantification theory if and only if it has property *A*. (A proposition has property *A* if there is a scheme — an array of signed letters and braces defined by recursion on the number of bound variables of the proposition, that generates a normal identity from that proposition.) Herbrand stated his FT this way: (1) If for some number p a proposition P has property B of order p, the proposition P is true. Once we know any such p, we can construct a proof of P. (2) If we have a true proposition P, together with a proof of P, we can, from this proof, derive a number p such that P has property B of order p [Herbrand, 168–169]. (A proposition expanded according to a certain set of rules [Herbrand, 153] that is free of both quantifiers and functions has property B.)

Anellis has provided a modern statement of Herbrand's FT: "For some formula F of classical quantification theory, an infinite sequence of quantifier-free formulas $F_1, F_2,\ldots$ can be effectively generated, for F provable in (any standard) quantification theory, if and only if there exists a k such that F is (sententially) valid; and a proof of F can be obtained from F_k" [Anellis *1991*, 75].

CONCLUSION.

A comparison of the unification algorithms presented here indicates that although their formalizations differ, conceptually they are the same. For example, the first two steps in Herbrand's algorithm correspond to item d in Martelli/Montanari's; Herbrand's step 3 is Martelli/Montanari's item e. The first two items in Martelli/Montanari have no counterpart in Herbrand because they are necessary only for machine computation.

I think Herbrand was hinting at a mechanical process, a computational procedure, that could be applied to a mathematical statement to answer the question of whether it was provable. Hao Wang understood the role of Herbrand's work linking provability and machine computation when he wrote,

> A fundamental result of Herbrand has the effect that any derivation of a theorem in a consistent axiom system corresponds to a truth-functional tautology of a form related to the statement of the theorem and the axioms of the system in a predetermined way. This and the possibility...of viewing axiom systems as proof-grinding machines can both be used to bring about the application of computing machines to the investigation of the question of derivability in general, and inconsistency (i.e., derivability of contradictions) in particular axiom systems. [Wang *1970*, 157]

Wolfgang Bibel [Bibel, 11–12] claims that Herbrand's FT, when applied as a constructive tool for providing an effective proof procedure, fails because it generates so much redundancy that it exhausts the available computational resources. (And the correction of an error Herbrand made in the proof of his FT results in an even larger number of terms than originally envisaged [Goldfarb, *1993*].) Bibel goes on to say that it is not surprising because Herbrand could not have thought of this application for his FT. But the fact that the FT produces redundancies does not invalidate the notion of a mechanical process. Robinson's resolution method, a mechanical process, also produces redundancies. Bibel notes that resolution can generate a large number of redundant new clauses and is even less advanced than the FT because it requires a formula to be in normal form, itself another source of redundancy.

In a paper published in 1931, "Sur la non-contradiction de l'arithmétique" [Herbrand, 282–298], Herbrand also hinted at the definition of a general recursive function. Herbrand and Gödel had exchanged letters — unpublished correspondence discussed by Gödel — in which Herbrand in 1931 had proposed a more general definition of a recursive, i.e. effectively calculable, function [Gödel, 70–71] Less that thirty years later, Gödel writes:

> [Herbrand] probably believed that such a proof [of the existence and unicity of a given recursive function] can be given only by exhibiting a computational

> procedure.... So I don't think there is any discrepancy between his two definitions as he meant them. What he failed to see (or to make clear) is that the computation, for *all* computable functions, proceeds by *exactly the same rules*. It is this fact that makes a precise definition of general recursiveness possible [Herbrand, 283–284].

Herbrand's use of general recursive functions is noted by Goldfrab who identifies as a key element in Herbrand's proof of his FT,

> the elimination of the induction axiom schema...through the introduction of functions.... The functions are, in fact (general) recursive functions, and here is the first appearance of the notion of recursive (as opposed to primitive recursive function [Herbrand, 283].

The effects of Robinson's work on unification based resolution are profound. Programming resolution based proof procedures led to the birth of logic programming. Furthermore, in Robinson's own words [Robinson *1992*, 4],

> The earliest versions of the predicate calculus proof procedure were all based on *human-oriented* reasoning patterns — on types of inference which reflected formally the kind of 'small' reasoning steps which humans find comfortable.... Finally, in the early summer of 1963, I managed to devise a clausal logic with a single inference scheme...[producing] a rather inhuman but very effective new inference pattern, for which I proposed the name *resolution*. Resolution permits the taking of arbitrarily large inference steps which in general require very considerable computational effort to carry out.

Unification based resolution is essentially a machine-oriented reasoning pattern whose implications we have only begun to explore.

REFERENCES

ANELLIS, Irving H. 1991. *The Löwenheim-Skolem theorem, theories of quantification, and proof theory*, Thomas Drucker (editor), *Perspectives on the history of mathematical logic* (Boston/Basel/Berlin, Birkhäuser), 71–83.

—. 1992. *Jean van Heijenoort's contributions to proof theory and its history*, Modern Logic 3, 312–335.

BIBEL, Wolfgang. 1982. *Computationally improved versions of Herbrand's theorem*, in J. Stern (editor), *Proceedings of the Herbrand symposium, logic colloquium '81, Marseilles, France, July 1981* (Amsterdam/New York/Oxford, North-Holland, 1982), 11–28.

GÖDEL, Kurt. 1965. *On undecidable propositions of formal mathematical systems*, Martin Davis (editor), *The undecidable: Basic papers on undecidable propositions, unsolvable problems and computable functions* (Hewlett, N.Y., Raven), 41–73.

GOLDFARB, Warren. 1993. *Herbrand's error and Gödel's correction*, Modern Logic 3, 103–118.

HERBRAND, Jacques. 1971. *Jacques Herbrand: Logical writings* (Warren Goldfarb, editor), Cambridge, Harvard University Press.

JOUNNAUD, Jean-Pierre and Claude Kirchner. 1991. *Solving equations in abstract algebras: A rule-based survey of unification*, Jean-Louis Lassez and Gordon Plotkin (editors), *Computational logic: Essays in Honor of Alan Robinson* (Cambridge, MA/London, MIT Press), 257–321.

LOVELAND, Donald W. 1984. *Automated theorem proving: a quarter-century review*, W.W. Bledsoe & D.W. Loveland (editors), Contemporary Mathematics 29, 1–45.

MARTELLI, Alberto and Ugo MONTANARI. 1982. *An efficient unification algorithm*, ACM Transactions on Programming Languages and Systems 4, 258–282.

ROBINSON, J. Alan 1965. *A machine-oriented logic based on the resolution principle*, J. of the ACM 12, 23–41.

—. 1992. *Logic and logic programming*, Communications of the ACM 35, 40–65.

VAN HEIJENOORT, Jean. 1967. (editor), *From Frege to Gödel: A source book in mathematical logic, 1879 – 1931*, Cambridge, Mass., Harvard University Press.

—. 1968. *Préface*, J. van Heijenoort (editor), *Jacques Herbrand, Écrits logiques*. (Paris, Presses Universitaires de France), 1–12.

—. 1986. *Jacques Herbrand's work in logic and its historical context*, J. van Heijenoort, *Selected essays* (Naples, Bibliopolis, copyright 1985), 99–121. Revised English translation of *L'ouevre logique de Jacques Herbrand et son contexte historique*, in J. Stern (editor), *Proceedings of the Herbrand symposium, logic colloquium '81, Marseilles, France, July 1981* (Amsterdam/New York/Oxford, North-Holland, 1982), 57–85.

— 1992. *Historical development of modern logic*, Modern Logic 2, 242–255.

WANG, Hao. 1970. *Logic, computers, and sets* (New York, Chelsea Publishing Co.), 127–159, 224–268. Reprints of *A variant to Turing's theory of calculating machines*, Journal of the Association of Computing Machinery 4 (1957), 63–92, and *Toward mechanical mathematics*, IBM Journal 4 (1960), 2–22.