INFORMATION AND CONTROL 38, 170–178 (1978)

The Undecidability of the Third Order Dyadic Unification Problem

LEWIS D. BAXTER

Department of Computer Science, York University, Downsview, Ontario, Canada

The unification problem for expressions of third order dyadic logic is proved to be undecidable by a reduction of the problem of solving Diophantine equations. This refines an earlier result on the undecidability of the third order unification problem.

INTRODUCTION

The unification problem of logic is to determine, given two expressions e_1 and e_2 containing variables, whether or not there exists a substitution for these variables by expressions which, applied to e_1 and e_2, makes them equal.

The unification problem arises from computational logic, which is concerned with the automation of logical inferences, and is well motivated in Robinson (1965), Pietrzykowski and Jensen (1972), and Huet (1975).

In first order logic an algorithm of linear time complexity was discovered by Paterson and Wegman (1976) to solve the unification problem. The decidability of the unification problem in second order logic remains unresolved (Winterstein, 1976). Huet (1973) and Lucchesi (1972) independently showed that the unification problem in third order logic is undecidable by a reduction of the Post correspondence problem. Here, we refine their result by restricting the arity (number of arguments) in the expressions to be at most two.

The results of Huet and Lucchesi imply the existence of $k \geqslant 4$ such that the third order k-adic unification problem is undecidable. This follows from the existence of $k \geqslant 4$ such that the Post correspondence problem with k pairs of words is undecidable and from their construction of an instance of the third order k-adic unification problem from an instance of the Post correspondence problem with k pairs of words.

Here, we prove the undecidability of the third order dyadic unification problem by a reduction of the problem of solving Diophantine equations. The undecidability of this follows from Matijasevic's solution to Hilbert's Tenth Problem (Matijasevic, 1970).

Our result can be used to refine the undecidability result of solving equations in type theory (Andrews, 1974).

0019-9958/78/0382-0170$02.00/0
Copyright © 1978 by Academic Press, Inc.
All rights of reproduction in any form reserved.

In Section 1 we describe the language of expressions and the unification problem. In Section 2 we present the Diophantine problem of finding integer solutions to multinomial equations. In Section 3 we prove our result by showing the equivalence between the third order dyadic unification problem and the Diophantine problem.

1. Language

We briefly describe the notion of types, expressions, and substitutions. A detailed presentation is found in Huet (1975) or Baxter (1976). We shall also define the unification problem.

1.1. *Types*

Each expression of our language has a unique *type* which indicates its position in a functional hierarchy.

The set T of types is defined inductively from a fixed set T_0 of *basic types* by

(1) $t \in T_0$ implies $t \in T$;

(2) $t_1, \ldots, t_n \in T$ $(n \geqslant 1)$ and $t_0 \in T_0$ imply $(t_1, \ldots, t_n \rightarrow t_0) \in T$.

The *order* of a type is defined as a mapping from T to the set of integers by:

$$order[t] = \begin{cases} 1 & \text{if } t \in T_0, \\ 1 + max\{order[t_i] \mid 1 \leqslant i \leqslant n\} & \text{if } t = (t_1, \ldots, t_n \rightarrow t_0). \end{cases}$$

The *arity* of a type is defined as a mapping from T to the set of integers by:

$$arity[t] = \begin{cases} 0 & \text{if } t \in T_0, \\ n & \text{if } t = (t_1, \ldots, t_n \rightarrow t_0). \end{cases}$$

1.2. *Expressions*

Our expressions are based on the normal form of λ-calculus expressions of Church (1940).

Expressions are constructed from *symbols*, which are either *variables* or *constants*, and from the punctuation marks: λ, $\cdot$ ( ). Associated with each symbol s is some type, written $type[s]$. We extend order and arity to symbols by

$$order[s] = order[type[s]],$$

$$arity[s] = arity[type[s]].$$

Informally, the order of a symbol indicates whether it is an individual, function, functional,..., and the arity of a symbol indicates the number of arguments it has.

We now define expressions and their types by induction on the number of occurrences of symbols.

An *expression* is of the form

$$\lambda u_1 \cdots u_m \cdot s(e_1, \ldots, e_n),$$

where $u_1, \ldots, u_m$ are $m \geqslant 0$ distinct variables (formal parameters), s is a symbol of arity n, and $e_1, \ldots, e_n$ are $n \geqslant 0$ expressions (arguments) and where for some basic type t_0

$$type[s] = \begin{cases} t_0 & \text{if } n = 0, \\ (type[e_1], \ldots, type[e_n] \to t_0) & \text{if } n \geqslant 1. \end{cases}$$

The type of the expression is then defined to be

$$type[\lambda u_1 \cdots u_m \cdot s(e_1, \ldots, e_n)] = \begin{cases} t_0 & \text{if } m = 0, \\ (type[u_1], \ldots, type[u_m] \to t_0) & \text{if } m \geqslant 1. \end{cases}$$

We will often abbreviate expressions; for example, $F(x)$ abbreviates $\lambda \cdot F(\lambda \cdot x(\,))$, and if $type[\psi] = ((\alpha \to \beta), \gamma \to \delta)$ then ψ abbreviates $\lambda fu \cdot \psi(\lambda v \cdot f(v), u)$ where $type[f] = (\alpha \to \beta)$, $type[u] = \gamma$, and $type[v] = \alpha$. Note that ψ is not an expression of our language but merely an abbreviation. Contrast this with the case of languages of λ-calculus which use η-conversion. There the expression $\lambda fu \cdot \psi(\lambda v \cdot f(v), u)$ can be converted to the expression $\lambda fu \cdot \psi(f, u)$ which, in turn, can be converted to the expression ψ. We assume familiarity with the notion of free and bound occurrences of variables in expressions. Expressions are defined to be *equal* if they can be made identical by appropriate changes of bound variables.

1.3. *Substitutions*

A *substitution* is a finite set of ordered pairs:

$$\{(v_1, e_1), \ldots, (v_n, e_n)\},$$

where the v_i's are distinct variables (to which the substitution *pertains*) and the e_i's are expressions such that $type[v_i] = type[e_i]$ for $i = 1, \ldots, n$. This substitution is written suggestively as $\{v_1 \leftarrow e_1, \ldots, v_n \leftarrow e_n\}$.

Informally, the *application* of a substitution σ to an expression e, written $\sigma(e)$, is found by simultaneously replacing the free variables in e to which σ pertains, by the corresponding expressions. We must be careful to ensure that there is no conflict of bound variables. For example, if

$$e = \lambda xyz \cdot f(g(A), f(g(x), g(y), x), H(z))$$

and

$$\sigma = \{f \leftarrow \lambda xyz \cdot y, g \leftarrow \lambda u \cdot F(u, x, w), z \leftarrow A\}$$

then

$$\sigma(e) = \lambda x'y'z' \cdot F(y', x, w).$$

The notion of application can be formally defined in several ways. Huet (1975) defines the application of $\{v_1 \leftarrow e_1, \ldots, v_n \leftarrow e_n\}$ to e as the normal form of $[\lambda v_1 \cdots v_n \cdot e](e_1, \ldots, e_n)$. This is found by applying the two rules of λ-conversion: α-conversion which renames bound variables and β-reduction which replaces formal parameters by actual arguments. Baxter (1976) uses the following inductive definition of $\sigma(e)$. Let V be the set of variables which pertain to σ or which occur free in some e^* where $v^* \leftarrow e^*$ belongs to σ for some v^*. Rename the bound variables occurring in e to obtain an expression e' which is equal to e but in which no variable in V is bound: $e' = \lambda u_1 \cdots u_m \cdot s(e_1, \ldots, e_n)$. If the symbol s is some u_i $(i = 1, \ldots, m)$ or if s is a constant or if σ does not pertain to s, then define $\sigma(e)$ to be $\lambda u_1 \cdots u_m \cdot s(\sigma(e_1), \ldots, \sigma(e_n))$. Otherwise let $s \leftarrow \lambda v_1 \cdots v_n \cdot E$ belong to σ and define $\sigma(e)$ as $\lambda u_1 \cdots u_m \cdot \sigma'(E)$ where $\sigma' = \{v_1 \leftarrow \sigma(e_1), \ldots, v_n \leftarrow \sigma(e_n)\}$. The basis of the inductive definition is the application of the empty substitution ϵ to an expression e and is defined by $\epsilon(e) = e$.

The actual language used to define expression, substitution, and application is not critical since our result easily extends to similar formulations.

1.4. *Unification*

We say that the substitution σ *unifies* the two expressions e_1 and e_2 if $\sigma(e_1) = \sigma(e_2)$. The *unification problem* is to determine, given two expressions e_1 and e_2, whether or not there exists a substitution σ which unifies e_1 and e_2. The *nth order unification problem* is obtained by restricting the order of all variables occurring in e_1 and e_2 to be at most n. The *nth order k-adic unification problem* is obtained by making the further restriction that the arity of all symbols occurring in e_1 and e_2 is at most k.

2. Diophantine Problem

2.1. *Multinomial*

We first define a *multinomial* (multivariable polynomial) in the variables $m_1, \ldots, m_N$ inductively by

(1) 1 is a multinomial;

(2) $m_1, \ldots, m_N$ are multinomials;

(3) if P and Q are multinomials, then $(P + Q)$ is a multinomial;

(4) if P and Q are multinomials, then $(P \times Q)$ is a multinomial.

The *Diophantine problem* is to determine whether or not a Diophantine equation has a solution in nonnegative integers, that is, to determine whether or not

$$\exists m_1 \cdots \exists m_N \; P[m_1, \ldots, m_N] = Q[m_1, \ldots, m_N],$$

where P and Q are multinomials in the variables $m_1, \ldots, m_N$. The Diophantine problem was first posed as Hilbert's Tenth Problem in 1900 and was found to be undecidable by Matijasevic (1970).

2.2. *Association*

In order to relate the Diophantine problem with our unification problem we will associate with any multinomial in the variables $m_1, \ldots, m_N$ an expression of our language.

These expressions will consist of the following symbols: the variables x, f, $\phi_1, \ldots, \phi_N$, and the constant L. Their types are given by

$$\begin{aligned} type[x] &= \alpha, \\ type[f] &= (\alpha \to \alpha), \\ type[L] &= (\alpha, \alpha \to \alpha), \\ type[\phi_i] &= ((\alpha \to \alpha), \alpha \to \alpha) \qquad \text{for} \quad i = 1, \ldots, N, \end{aligned}$$

where the set of basic types is $T_0 = \{\alpha\}$. Note that all symbols are of order at most three, and arity at most two. Each ϕ_i corresponds to m_i, and L is used to construct a list of expressions.

We will now define an expression E_P associated with any multinomial P. Note that the variable x occurs both free and bound in E_P and that the variable f occurs only free in E_P. Let $E_P[e', e'']$ denote the application of the substitution $\{f \leftarrow e', x \leftarrow e''\}$ to E_P. The definition of E_P is by induction on multinomials:

(1) $E_1 = f(x)$,

(2) $E_{m_i} = \phi_i(\lambda x \cdot f(x), x) \qquad \text{for} \quad i = 1, \ldots, N$,

(3) $E_{(P+Q)} = E_P[\lambda x \cdot f(x), E_Q]$,

(4) $E_{(P \times Q)} = E_P[\lambda x \cdot E_Q, x]$.

For example, let $P = ((m_2 \times (m_2 \times m_2)) + (m_1 \times m_2))$, then $E_P = \phi_2(\lambda x \cdot \phi_2(\lambda x \cdot \phi_2(\lambda x \cdot f(x), x), x), \phi_1(\lambda x \cdot \phi_2(\lambda x \cdot f(x), x), x))$.

3. Theorem

THEOREM. *The third order dyadic unification problem is undecidable.*

Before we prove this, we present a lemma which will enable us to make a correspondence between certain substitutions and integers.

DEFINITION 3.1. Let $f^m(e)$ denote the expression $f(f(\cdots f(e)\cdots))$ where f occurs m times, for $m \geqslant 0$. If P is a multinomial in $m_1, \ldots, m_N$ then we will abbreviate $f^{P[m_1,\ldots,m_N]}(e)$ by $f^P(e)$ where $P[m_1, \ldots, m_N]$ denotes the value of the multinomial for integer values $m_1, \ldots, m_N$.

LEMMA 3.2. $\sigma(\lambda x \cdot \phi(\lambda x \cdot x, x)) = \lambda x \cdot x$ *iff* $\exists m\,[\sigma(\phi) = \lambda fx \cdot f^m(x)]$.

Proof. (If): This is trivial since the substitution $\{\phi \leftarrow \lambda fx \cdot f^m(x)\}$ applied to $\lambda x \cdot \phi(\lambda x \cdot x, x)$ equals $\lambda x \cdot x$.

(Only if): Rather than define and then use the general method, given by Huet (1975), which searches for substitutions which unify expressions, we give a proof based on analyzing the symbols which occur in $\sigma(\phi)$.

Let $\phi \leftarrow \lambda fx \cdot e_0$ belong to σ where $e_0 = s_0(e_1, \ldots, e_n)$ and s_0 is a symbol. Then

$$\sigma(\lambda x \cdot \phi(\lambda x \cdot x, x)) = \lambda x \cdot \{f \leftarrow \lambda x \cdot x\}(s_0(e_1, \ldots, e_n)).$$

Now, s_0 cannot be a constant since the right side of this equation would be of the form $\lambda x \cdot s_0(\cdots)$ which cannot equal $\lambda x \cdot x$. Similarly s_0 cannot be a variable different from f or x. If s_0 equals x, then indeed $n = 0$ and the right side equals $\lambda x \cdot x$. If s_0 equals f, which has arity one, then the right side becomes $\lambda x \cdot \{f \leftarrow \lambda x \cdot x\}(e_1)$. By repeating this argument, we have

$$\phi \leftarrow \lambda fx \cdot f^m(x) \text{ belongs to } \sigma, \qquad \text{for some } m \geqslant 0.$$

3.3. *Proof of Theorem*

With the Diophantine problem

$$\exists m_1 \cdots \exists m_N\, P[m_1, \ldots, m_N] = Q[m_1, \ldots, m_N],$$

we associate the problem of unifying the two expressions

$$e_1 = \lambda fx \cdot L(\phi_1(\lambda x \cdot x, x), L(\phi_2(\lambda x \cdot x, x), \ldots, L(\phi_N(\lambda x \cdot x, x), E_P)\cdots)),$$
$$e_2 = \lambda fx \cdot L(x, L(x, \ldots, L(x, E_Q)\cdots)),$$

where E_P and E_Q are defined in Section 2.2. This is clearly a third order dyadic unification problem. We will now prove their equivalence by showing

$$\exists m_1 \cdots \exists m_N\, P[m_1, \ldots, m_N] = Q[m_1, \ldots, m_N] \quad \text{iff} \quad \exists \sigma[\sigma(e_1) = \sigma(e_2)].$$

Now

$$\exists \sigma\,[\sigma(e_1) = \sigma(e_2)]$$

iff

$$\exists \sigma\,[\forall i\, \sigma(\lambda x \cdot \phi_i(\lambda x \cdot x, x)) = \lambda x \cdot x$$

and

$$\phi(\lambda fx \cdot E_P) = \sigma(\lambda fx \cdot E_Q)] \qquad \text{(using simple properties of substitutions)}$$

iff

$$\exists\sigma\,[\forall i\,\exists m_i\,\sigma(\phi_i) = \lambda fx \cdot f^{m_i}(x) \text{ and } \sigma(\lambda fx \cdot E_P) = \sigma(\lambda fx \cdot E_Q)] \qquad \text{(by the lemma)}$$

iff

$$\exists m_1 \cdots \exists m_N\, \sigma^*(\lambda fx \cdot E_P) = \sigma^*(\lambda fx \cdot E_Q),$$

where

$$\sigma^* = \{\phi_1 \leftarrow \lambda fx \cdot f^{m_1}(x), \ldots, \phi_N \leftarrow \lambda fx \cdot f^{m_N}(x)\}.$$

We now show, by induction on multinomials, that for all integers $m_1, \ldots, m_N$ $\sigma^*(E_P) = f^P(x)$ where P is any multinomial in $m_1, \ldots, m_N$.

(1) $\sigma^*(E_1) = \sigma^*(f(x)) = f^1(x)$.

(2) $\sigma^*(E_{m_i}) = \sigma^*(\phi_i(\lambda x \cdot f(x), x))$
$= f^{m_i}(x)$ (by applying σ^* to ϕ_i)
for all $i = 1, \ldots, N$.

(3) $\sigma^*(E_{(P+Q)}) = \sigma^*(E_P[\lambda x \cdot f(x), E_Q])$ (by definition)
$= f^P(\sigma^*(E_Q))$ (by hypothesis for E_P)
$= f^P(f^Q(x))$ (by hypothesis for E_Q)
$= f^{(P+Q)}(x)$.

(4) $\sigma^*(E_{(P\times Q)}) = \sigma^*(E_P[\lambda x \cdot E_Q, x])$ (by definition)
$= (\sigma^*(\lambda x \cdot E_Q))^P(x)$ (by hypothesis for E_P)
$= (f^Q)^P(x)$ (by hypothesis for E_Q)
$= f^{(P\times Q)}(x)$.

Hence $\exists\sigma\,[\sigma(e_1) = \sigma(e_2)]$ iff there exist $m_1, \ldots, m_N$ such that

$$\sigma^*(\lambda fx \cdot E_P) = \sigma^*(\lambda fx \cdot E_Q)$$

iff

$$\lambda fx \cdot \sigma^*(E_P) = \lambda fx \cdot \sigma^*(E_Q)$$

iff

$$\lambda fx \cdot f^P(x) = \lambda fx \cdot f^Q(x)$$

iff

$$P = Q.$$

Finally, since the Diophantine problem is undecidable, so also is the third order dyadic unification problem.

Conclusion

Having shown that the third order dyadic unification problem is undecidable, the decidability of the third order monadic case naturally arises. This problem is unresolved. Whereas we have shown that the substitutions involved in the third order dyadic problem can express multinomials, those involved in the third order monadic case appear to have the same expressiveness as in the second order monadic problem. This latter problem is also unresolved (Siekman, 1975; Winterstein, 1976).

A related problem is the third order instantiation problem. This is the third order unification problem in which one expression contains no free variables. Whereas the second order instantiation problem has been shown to be *NP*-complete by Baxter (1976) and has also been analyzed by Huet (1976), no results of third order are known.

Acknowledgments

I am grateful for the referee's constructive comments and criticisms.

Received: May 11, 1977; revised: October 11, 1977

References

Andrews, P. B. (1974), Provability in elementary type theory, *Z. Math. Logik Grundlagen Math.* **20**, 411–418.

Baxter, L. D. (1976), "The Complexity of Unification," Ph.D. Thesis, University of Waterloo.

Church, A. (1940), A formulation of the simple theory of types, *J. Symbolic Logic* **5**, 56–68.

Huet, G. P. (1973), The undecidability of unification in third order logic, *Inform. Contr.* **22**, 257–267.

Huet, G. P. (1975), A unification algorithm for typed λ-calculus, *Theor. Comput. Sci.* **1**, 27–57.

Huet, G. P. (1976), "Résolution d'équations dans des langages d'ordre 1, 2,..., ω," Thèse de Doctorat d'État, University of Paris.

Lucchesi, C. (1972), "The Undecidability of the Unification Problem for Third Order Language," Research Report CSRR 2059, Department of Computer Science, University of Waterloo.

Matijasevic, Y. (1970), Enumerable sets are diophantine, *Dokl. Akad. Nauk SSSR* **191**, 279–282.

Paterson, M. S., and Wegman, M. N. (1976), Linear unification, *in* "Proceedings of the Eighth Annual Association for Computing Machinery Symposium on Theory of Computing," pp. 181–186.

Pietrzykowski, T., and Jensen, D. (1972), A complete mechanization of ω-order logic *in*, "Proceedings of the Association for Computing Machinery National Conference 1972," Vol. 1, pp. 82–92.

ROBINSON, J. A. (1965), A machine-oriented logic based on the resolution principle, *J. Assoc. Comput. Mach.* **12**, 23–41.

SIEKMAN, J. (1975), String unification, Research Report CSM-7, Computing Centre, University of Essex.

WINTERSTEIN, G. (1976), "Unification in Second Order Logic," Research Report, Fachbereich Informatik, Universität Kaiserslautern.