INFORMATION AND COMPUTATION 102, 83–101 (1993)

The Undecidability of the Semi-unification Problem*

A. J. KFOURY

Department of Computer Science,
Boston University, Boston, Massachusetts 02215

AND

J. TIURYN AND P. URZYCZYN

Institute of Mathematics, University of Warsaw,
00-901 Warsaw, PKIN 9p, Poland

The Semi-Unification Problem (SUP) is a natural generalization of both first-order unification and matching. The problem arises in various branches of computer science and logic. Although several special cases of SUP are known to be decidable, the problem in general has been open for several years. We show that SUP in general is undecidable, by reducing what we call the "boundedness problem" of Turing machines to SUP. The undecidability of this boundedness problem is established by a technique developed in the mid-1960s to prove related results about Turing machines.

1. INTRODUCTION

Let Σ be a first-order signature, X a countably infinite set of variables, and $\mathcal{T}$ the set of all terms over Σ and X. An *instance* Γ of Semi-Unification is a finite set of inequalities,

$$\Gamma = \{t_1 \leqslant u_1, \ldots, t_n \leqslant u_n\},$$

where $t_i, u_i \in \mathcal{T}$. A *substitution* is a function $S: X \to \mathcal{T}$. Every substitution extends in a natural way to a Σ-homomorphism $S: \mathcal{T} \to \mathcal{T}$. A substitution S is a *solution* of the instance Γ iff there are substitutions $S_1, \ldots, S_n$ such that

$$S_1(S(t_1)) = S(u_1), \ldots, S_n(S(t_n)) = S(u_n). \tag{1}$$

* This work is partly supported by NSF Grant CCR-8901647 and by a grant of the Polish Ministry of National Education, No. RP.I.09. A preliminary version of this paper has been presented at the *22nd ACM Symposium on Theory of Computing, Baltimore, 1990*, pp. 468–476.

0890-5401/93 $5.00

The *Semi-Unification Problem* (henceforth abbreviated SUP) is the problem of deciding, for any instance Γ, whether Γ has a solution.

Problems in several branches of computer science and logic give rise to SUP, or to special cases of SUP, or to generalizations of SUP. For example:

(P1) A sufficient condition for a rewrite rule to be non-terminating can be formulated as an instance of SUP with exactly one inequality (see [9, 20]).

(P2) The type reconstruction problem for the polymorphic functional language **ML** can be shown to be polynomial time equivalent to a special case of SUP where every instance satisfies a certain acyclicity condition (see [12]).

(P3) The type reconstruction problem for extensions of **ML** that allow polymorphic recursion (see [14, 18]) have been shown in [3, 10] to be polynomial time reducible to the general form of SUP. A polynomial time reduction in the opposite direction has been established in [11]. The reader is referred to [13] for another related result.

(P4) According to P. Pudlak [19], the decidability of SUP would imply a strengthened version of a conjecture due to Kreisel, stated as follows: "There exists a computable function f such that if Peano Arithmetic proves $A(0^{(n)})$ in k steps for all $n \leqslant f(k, A)$, where $0^{(n)}$ is the n-th natural number, then it also proves $\forall x\, A(x)$." Apparently M. Baaz, again according to [19], uses SUP to settle a weaker version of Kreisel's conjecture.[1] Very similar, it appears, is a study (unpublished) undertaken some ten years ago, about the relative speed of proofs in Frege systems with substitution, where the question "Can you decide whether a wff is provable within a certain number of steps?" is also reduced to SUP in general [21].

(P5) In computational linguistics, researchers have considered feature algebras with a relation of subsumption (see [2]). Feature algebras can be viewed as a generalization of the term algebra, while the subsumption preorder defined there is a generalization of the term matching. The semi-unification problem for feature clauses, as stated in [2], becomes then a generalization of SUP.

The special case of SUP arising in relation to problem (P1), i.e., exactly one inequality, is decidable (see [9, 19]). It has been shown in [9] that this problem is decidable in polynomial time. It is not difficult to see that SUP for one inequality is equivalent to the additional requirement that all the substitutions $S_1, \ldots, S_n$ in (1) be identical. It has been shown in [19] that the decidability of the general case of SUP is equivalent to the

[1] This result, however, remains unpublished.

decidability of two inequalities. The method of proof presented in this paper yields directly the undecidability of SUP restricted to two inequalities.

The special case of SUP arising in relation to problem (P2) was recently shown in [12] to be DEXPTIME-complete. This yields tight bounds on the complexity of the type reconstruction problem for **ML**. The reader is referred to [7, 16] for another direct proof of the above mentioned result for **ML**. There are other special cases of SUP proven to be decidable. In [15] it is shown that SUP restricted to instances with only two variables is decidable. In [11] it is shown that SUP is decidable for instances consisting of systems Γ in which every variable occurring on the left side of any inequality occurs in that inequality on the left side exactly once (the so called left-linear SUP). It has been shown in [5] that this problem is PTIME-complete.

The main result of [2] is the undecidability of the problem of checking whether a given feature clause is satisfiable in a finite feature algebra. A corollary of that result that is that the problem of whether a given system of inequalities Γ has a solution in regular trees[2] is undecidable. There is a striking similarity between the method of proving the undecidability result in [2] and the two independent results of undecidability of finite implication problem for functional and inclusion dependencies (see [1, 17]). It has been suggested by P. Kanellakis [8] that there ought to be a common metatheory relating the semi-unification, functional, and inclusion dependencies in database theory, and feature algebras with subsumption. However, we do not pursue this possibility in this paper.

In this paper we show that if the signature Σ contains at least one function symbol of arity $\geqslant 2$, then SUP is undecidable.[3] We have to admit, to our embarrassment, that among the many erroneous claims announcing the decidability of SUP there was also ours [10]. One of the practical consequences of the result presented in this paper is that automatic type-reconstruction for functional programs in type disciplines that include polymorphic recursion is not generally possible (see [11]).[4]

The proof is organized as follows. In Section 2 we develop some metatheory that deals with semi-unification. We give here a semi-decision

[2] Regular trees are (possibly infinite) trees that have only finitely many non-isomorphic subtrees. One has to assume that the first-order signature Σ contains some constants since otherwise Γ always has a regular solution.

[3] The case when Σ contains only function symbols of arity $\leqslant 1$, besides being decidable, is not encountered in the problems mentioned above.

[4] This is the problem that led us to semi-unification in the first place. For us, type-reconstruction means the ability to effectively determine whether an untyped program phrase is typable and, if it is, to construct a type for it—and this is generally not possible in the presence of polymorphic recursion.

procedure for solvability of semi-unification. It is presented as a reduction system that consists in reducing two kinds of redexes, transforming in this way instances of SUP into other instances of SUP. It is proved (Lemma 2) that the initial instance of SUP has a solution iff it can be reduced to an instance containing no redexes. Another important concept introduced in this section is that of a *path equation*. Path equations express possible relationships between various parts of a solution, should it exist. A natural formal proof system for deriving path equations is given here too. We also prove in Section 2 the *principality property* for SUP (Proposition 3). This says that every instance Γ of SUP that has a solution, has a minimal solution, i.e., a solution such that every other solution is a substitution instance of that one. This result is not used in the proof of the main theorem. It is stated here since it is an easy consequence of the results obtained in this section. The principality property has been independently established by M. Baaz (see [19] for reference to Baaz's unpublished work) and by F. Henglein [4].

In Section 3, which plays a purely technical role in our proof, we introduce an auxiliary kind of a Turing machine that we call *Intercell Turing Machine* (ITM). Its main characteristic feature which differentiates it from other models of Turing machines is that its read/write head is always located between cells, enabling it to scan either the contents of the left or of the right cell. This model proves useful in the next section where we combine path equations with ITM's. The main result of Section 3 is the undecidability of the boundedness problem for ITM's. A machine M is bounded if there is a constant k such that for every instantaneous description[5] (ID) of M the number of different ID's reachable by M from this one is at most k. A direct proof of the undecidability of the boundedness problem is difficult since "most" of the machines are unbounded. This problem is closely related to the problem of *immortality* for Turing machines. M is immortal if there is an ID of M from which M diverges. P. H. Hooper [6] in his 1965 Ph. D. Thesis proved that the immortality problem for Turing machines is undecidable.[6] We use his result and show first that the boundednes problem for ordinary deterministic Turing machines is undecidable too (Corollary 6). We then establish (Lemma 9) the undecidability of the boundedness problem for ITM's.

Section 4, the last part of this paper, contains the main reduction step in the proof of the undecidability. In this section we put together the path equations and computations of ITM's. More precisely, we show how computations of a symmetric ITM can be simulated by derivable path

[5] For technical reasons we allow an instantaneous description to be an infinite word.

[6] The preliminary version of the present paper contained in its Appendix a sketch of Hooper's proof since we were not aware of a published version of it. After we learned of the JSL publication [6], we decided to omit the Appendix.

equations of a constructed instance of SUP (Lemma 10). This yields a reduction of the boundedness problem for ITM's to SUP (Lemma 11), thereby proving the main result of this paper.

2. Semi-Unification

Let Σ be a first-order signature consisting of exactly one binary function symbol. Let X be a countably infinite set of variables, and $\mathcal{T}$ the set of all terms over Σ and X. For reasons of convenience, the single binary function symbol in Σ is $\rightarrow$, used in infix notation, with the usual convention that $\alpha \rightarrow \alpha' \rightarrow \alpha''$ stands for $(\alpha \rightarrow (\alpha' \rightarrow \alpha''))$. We reserve early lower-case Greek letters to denote members of X.

A *substitution* is a function $S: X \rightarrow \mathcal{T}$. Every subsitution extends in a natural way to a Σ-homomorphism $S: \mathcal{T} \rightarrow \mathcal{T}$.

We write an instance Γ of the Semi-Unification Problem (SUP) as a finite set of inequalities, $\Gamma = \{t^1 \leqslant u^1, ..., t^n \leqslant u^n\}$, where $t^i, u^i \in \mathcal{T}$. A substitution S is a *solution* of instance Γ iff there are substitutions $S_1, ..., S_n$ such that

$$S_1(S(t^1)) = S(u^1), ..., S_n(S(t^n)) = S(u^n). \tag{2}$$

SUP is the problem of deciding, for any instance Γ, whether Γ has a solution. We refer to a solution of Γ by writing either S or the whole vector $(S, S_1, ..., S_n)$, depending on whether it is clear from the context what the S_i's are.

Let $\Gamma = \{t^1 \leqslant u^1, ..., t^n \leqslant u^n\}$ be an instance of SUP. We always assume that Γ comes with a fixed order in which the inequalities are written. To stress that the order in which the inequalities are listed is fixed, we explicitly write the number of an inequality as an index of that inequality. Thus Γ is now written as

$$t^1 \leqslant_1 u^1, ..., t^n \leqslant_n u^n. \tag{3}$$

We extend the set X of initially given variables to another set $\bar{X}$ whose members are each of the form α_w, where $\alpha \in X$ and $w \in \{1, ..., n\}^*$. Let $\bar{\mathcal{T}}$ be the set of all finite terms over Σ and $\bar{X}$. We identify α_ε with α, where ε is the empty string, so that $X \subseteq \bar{X}$ and $\mathcal{T} \subseteq \bar{\mathcal{T}}$. Variables in $\bar{X}$ are denoted by ξ and η, and terms in $\bar{\mathcal{T}}$ by t and u. Functions $S: \bar{X} \rightarrow \bar{\mathcal{T}}$ are also called subsitutions.

The intended relationship between variables in the initial X and variables in the extension $\bar{X}$ is as follows. Suppose $\alpha \in X$ and let $w = j_1 j_2 \cdots j_p$, where $j_1, ..., j_p$, where $j_1, ..., j_p \in \{1, ..., n\}$. If $(S, S_1, ..., S_n)$ is a solution of instance

Γ, we can think of α as representing the term $S(\alpha)$. Variable α_w then represents the term obtained by applying to $S(\alpha)$ the substitutions $S_{j_1}, S_{j_2}, \ldots, S_{j_p}$ (in that order). This explanation motivates the following definition. Given an $(n+1)$-tuple $(S, S_1, \ldots, S_n)$ of substitutions from X to $\mathscr{T}$, it induces a substitution $\bar{S}$ from $\bar{X}$ to $\mathscr{T}$ (and therefore from $\bar{X}$ to $\bar{\mathscr{T}}$ also), defined inductively by

$$\bar{S}(\alpha_\varepsilon) = S(\alpha) \tag{4}$$

and

$$\bar{S}(\alpha_{wi}) = S_i(\bar{S}(\alpha_w)), \tag{5}$$

for all $\alpha \in X$, $w \in \{1, \ldots, n\}^*$, and $i \in \{1, \ldots, n\}$.

For every $v \in \{1, \ldots, n\}^*$, we have a natural homomorphism $(\ )_v : \bar{\mathscr{T}} \to \bar{\mathscr{T}}$, defined on variables in $\bar{X}$ by

$$(\alpha_w)_v = \alpha_{wv}.$$

For an arbitrary term t, we also define the *left* and *right* subterms of t, denoted $L(t)$ and $R(t)$. More precisely, if t is a variable then $L(t)$ and $R(t)$ are undefined; otherwise we set

$$L(t^1 \to t^2) = t^1$$
$$R(t^1 \to t^2) = t^2.$$

If $\Pi \in \{L, R\}^*$, say $\Pi = x_1 x_2 \cdots x_p$, the notation $\Pi(t)$ means $x_1(x_2(\cdots(x_p(t))\cdots))$. For an arbitrary $\Pi \in \{L, R\}^*$, the subterm $\Pi(t)$ is defined provided Π (read from right to left) is a path (from the root to an internal node or to a leaf node) in the binary tree representation of t. As now defined, $\Pi(\ )$ is a partial function; it will be convenient later to turn $\Pi(\ )$ into a total function.

In the procedure below and in Lemma 1, we allow instances of SUP to involve variables in $\bar{X}$. A solution of such an instance is now a substitution from $\bar{X}$ to $\bar{\mathscr{T}}$. For clarity, let us remark the following: If Γ is an instance which mentions variables only in the initial X, and $(S, S_1, \ldots, S_n)$ is a solution of Γ (S defined only on X), then $\bar{S} : \bar{X} \to \bar{\mathscr{T}}$, defined according to the rules (4) and (5), is also a solution of Γ, which may be identified with S.

We now define a procedure which halts on input Γ iff Γ has a solution; moreover, if and when it halts, it constructs a solution for Γ. The procedure consists in repeatedly reducing *redexes*, which can be of two kinds, and it halts when no further reduction is possible.

• (*Redex I reduction*) Let $\xi \in \bar{X}$ and let $t' \notin \bar{X}$ be a term with the property that there is a path $\Pi \in \{L, R\}^*$ and $i \in \{1, ..., n\}$ such that if $t \leqslant_i u$ is the ith inequality of Γ, then

$$\Pi(t) = t' \qquad \text{and} \qquad \Pi(u) = \xi.$$

The pair of terms $(\xi, (t')_i)$ is called a *redex I*. The result of reducing this redex consists in substituting $(t')_i$ for all occurrences of ξ throughout Γ.

• (*Redex II reduction*) Let $\xi \in \bar{X}$ and $u' \in \bar{\mathscr{T}}$ have the property that $\xi \neq u'$ and there are paths $\Pi, \Delta, \Sigma \in \{L, R\}^*$ and $i \in \{1, ..., n\}$ such that if $t \leqslant_i u$ is the ith inequality in Γ, then

$$\Pi(t) = \Delta(t) \in \bar{X} \qquad \text{and} \qquad \Sigma\Pi(u) = \xi \qquad \text{and} \qquad \Sigma\Delta(u) = u'.$$

Such a pair (ξ, u') is called a *redex II*. The result of reducing this redex consists in substituting u' for all occurrences of ξ throughout Γ.

Our next goal is to establish the correctness of the above procedure. For this purpose we need some more definitions and notation. Each redex (ξ, t) determines a substitution $R: \bar{X} \to \bar{\mathscr{T}}$ such that $R(\xi) = t$ and $R(\eta) = \eta$, for $\eta \neq \xi$. Suppose that Γ' is an instance obtained from Γ by a sequence of reductions, and that $R_1, ..., R_n$ are the corresponding substitutions. If $\alpha \in X$ then we say that the term $t = R_n \cdots R_1(\alpha)$ is *assigned* to α in Γ'. Note that t is always a subterm of one of the terms occurring in Γ'.

Let now Γ be an instance with arbitrary variables in $\bar{X}$. By $\mathscr{S}_\Gamma$ we denote the set of all substitutions $S: \bar{X} \to \bar{\mathscr{T}}$ that are solutions of Γ. Each set $\mathscr{S}_\Gamma$ is quasi-ordered by the following *instantiation quasi-order*:

$$S' \sqsubseteq_\Gamma S'' \qquad \text{iff} \qquad \text{there is a substitution } P: \bar{X} \to \bar{\mathscr{T}} \text{ such that } P(S'(\xi)) = S''(\xi) \text{ holds for every } \xi \text{ occurring in } \Gamma.$$

A solution $(S, S_1, ..., S_n)$ of Γ is said to be *coherent* if for every $\eta \in \bar{X}$, $S(\eta_i) = S_i(S(\eta))$ holds. Note that if Γ mentions variables only in the initial X, and $S: X \to \mathscr{T}$ is a solution of Γ, then its extension (in the sense of the rules (4) and (5)) $\bar{S}: \bar{X} \to \bar{\mathscr{T}}$ is a coherent solution of Γ. If S and S' are substitutions, we also write SS' for their composition $S(S'(\ ))$.

LEMMA 1. *Let Γ be an instance of SUP with variables in $\bar{X}$, and let Γ' be obtained from Γ by reduction of a single redex (I or II). Let R be the substitution associated with this reduction. Then $\hat{R}: \mathscr{S}_{\Gamma'} \to \mathscr{S}_\Gamma$ is $\sqsubseteq$-monotone, where $\hat{R}(S) = SR$. Thus, if Γ' has a solution, then Γ has a solution. Moreover, if S is a coherent solution of Γ, then S is also a solution of Γ' and $\hat{R}(S) = S$.*

Proof. Let (ξ, t) be the redex which is being reduced. We have $\Gamma' = \Gamma[t/\xi]$. Clearly if $(S', S'_1, ..., S'_n)$ is a solution of Γ', then $(S'R, S'_1, ..., S'_n)$ is a solution of Γ. Thus $\hat{R}$ is well defined.

Let $S', S'' \in \mathscr{S}_{\Gamma'}$. If $S' \sqsubseteq_{\Gamma'} S''$, then $S'R \sqsubseteq_{\Gamma} S''R$ also holds (it suffices to check that $P(S'(t)) = S''(t)$, for an appropriate substitution P). Thus, $\hat{R}$ is $\sqsubseteq$-monotone.

It remains to prove the second part of the conclusion of Lemma 1. Let $(S, S_1, ..., S_n)$ be a coherent solution of Γ. Since R is the identity on all variables $\neq \xi$, then in order to establish $SR = S$, it suffices to prove that

$$S(\xi) = S(t). \tag{6}$$

If (ξ, t) is a redex I, i.e., $t = (t')_i$, where $i \in \{1, ..., n\}$ and t' is a subterm of a term in Γ, then because S is a solution, it must be that $S_i(S(t')) = S(\xi)$. By the coherence of S we obtain $S(t) = S((t')_i) = S_i(S(t')) = S(\xi)$, and therefore (6) holds. If (ξ, t) is a redex II, then clearly $S(\xi) = S(t)$ and (6) holds again.

Because $SR = S$, it immediately follows that S is a solution of Γ'. This completes the proof of Lemma 1. ∎

As a corollary of the above lemma we get the following result.

LEMMA 2. *Let Γ be an instance of SUP with all variables in the initial X.*

1. *Γ has a solution iff the above procedure, when started on instance Γ, halts producing an instance Γ' without redexes.*

2. *If Γ has no solution, then the above procedure will keep producing instances Γ' which assign arbitrarily large terms to variables in X.*

Proof. Let us first observe that if the procedure does not terminate, then we infinitely often perform a substitution, where we substitute a non-variable term for a variable. Indeed, if a redex (ξ, t) is such that t is a variable then (ξ, t) must be a redex II, whose reduction decreases by one the number of variables in the instance of SUP where (ξ, t) occurs. Hence, if the procedure does not terminate then the reduction of (ξ, t) must be eventually followed by the reduction of another redex (ξ', t') where t' is not a variable. Hence also, if the procedure does not terminate, it assigns arbitrarily large terms to variables in X. Thus part 2 of the lemma follows from part 1.

Now, if Γ has a solution S in $\mathscr{T}$, then we uniquely extend it to a coherent solution $\bar{S}$. Let us assume that the procedure does not terminate. Let $R_1, ..., R_n, ...$ be the substitutions associated with the successive

reductions performed (in this order) by the procedure. It follows from Lemma 1 and its proof that for every $n \geqslant 1$,

$$\bar{S} = \bar{S}R_n \cdots R_1. \tag{7}$$

But this is a contradiction, because arbitrarily large terms are assigned to variables in X. Hence, $\bar{S}$ cannot be a solution, so that S cannot be either.

On the other hand, if the procedure terminates producing Γ' without redexes, then the identity is a solution of Γ', by the definition of what redexes I and II are. (It does not matter at this point whether the identity is a coherent solution.) Let $S^* = R_n \cdots R_1$, where $R_1, ..., R_n$ are the substitutions applied (in this order) to obtain Γ' from Γ. By Lemma 1, S^* is a solution of Γ. This completes the proof of part 1 of the lemma. ∎

An easy consequence of the above two results is the existence of a principal solution. A solution S of Γ is *principal* iff $S \sqsubseteq_\Gamma S'$, for every $S' \in \mathscr{S}_\Gamma$.

PROPOSITION 3. *Let Γ be an instance of SUP with all variables in the initial X. If Γ has a solution, then it has a principal solution.*

Proof. Let us assume that Γ has a solution, say S. By Lemma 2, the procedure terminates with an instance Γ' which has no redexes. The substitution S^* defined at the end of the proof of Lemma 2 is a solution of Γ, while the identity id is a solution of Γ'. We claim that $S^* \sqsubseteq_\Gamma S$. Indeed, if $\bar{S} : \bar{X} \to \bar{\mathscr{T}}$ is the coherent extension of S, then $\bar{S}$ is a solution of Γ', by Lemma 1. Since $\mathrm{id} \sqsubseteq_{\Gamma'} \bar{S}$, by $\sqsubseteq$-monotonicity of the induced transformation (cf. Lemma 1), $S^* \sqsubseteq_\Gamma \bar{S}$, and of course $S^* \sqsubseteq_\Gamma S$. Since S was arbitrary, S^* is principal. ∎

For technical reasons we introduce one more kind of variable. An *auxiliary variable* is an expression of the form ξ^Δ, where ξ is an arbitrary variable in $\bar{X}$, and $\Delta \in \{L, R\}^*$. We identify ξ^ε with ξ, and agree that $L(\xi^\Delta) = \xi^{L\Delta}$ and $R(\xi^\Delta) = \xi^{R\Delta}$ for all $\Delta \in \{L, R\}^*$. With the introduction of auxiliary variables, for every $\Pi \in \{L, R\}^*$, $\Pi(\ )$ is now conveniently extended to a total function on the set of terms.

For our purposes it is easier to work with partial information about the sought solution. Thus we introduce the following definition. A *path equation* is an expression of the form

$$\Pi\xi = \Delta\eta, \tag{8}$$

where $\Pi, \Delta \in \{L, R\}^*$, and $\xi, \eta \in \bar{X}$. We say that the path equation (8) is *satisfied* by a vector $(S, S_1, ..., S_n)$ iff $\Pi(\bar{S}(\xi)) = \Delta(\bar{S}(\eta))$.

Let $t = u$ be an equation between two terms in $\mathscr{T}$, and suppose that for some $\Pi, \Delta \in \{L, R\}^*$, and some $\xi, \eta \in \bar{X}$ we have $\xi = \Pi(t)$ and $\eta = \Delta\Pi(u)$ (or conversely, $\xi = \Pi(u)$ and $\eta = \Delta\Pi(t)$). Then the path equation

$$\eta = \Delta\xi$$

is said to be *induced* by "$t = u$". We say that a path equation is induced by an inequality "$t \leqslant_i u$", iff it is induced by the equation "$(t)_i = u$".

We need a proof system for deriving the path equations from a given instance Γ. In what follows, $\alpha, \beta, \gamma \in X$, u, v, $w \in \{1, ..., n\}^*$, and Π, Δ, $\Sigma \in \{L, R\}^*$.

Axioms. All path equations induced by inequalities in Γ and all identity equations $\alpha = \alpha$.

Rules of Inference:

$$\text{(transitivity)} \qquad \frac{\Pi\alpha_w = \Delta\beta_v,\ \Delta\beta_v = \Sigma\gamma_u}{\Pi\alpha_w = \Sigma\gamma_u}$$

$$\text{(symmetry)} \qquad \frac{\Pi\alpha_w = \Delta\beta_v}{\Delta\beta_v = \Pi\alpha_w}$$

$$\text{(instantiation)} \qquad \frac{\Pi\alpha_w = \Delta\beta_v}{\Pi\alpha_{wu} = \Delta\beta_{vu}}$$

$$\text{(subterm)} \qquad \frac{\Pi\alpha_w = \Delta\beta_v}{\Sigma\Pi\alpha_w = \Sigma\Delta\beta_v}$$

The next result establishes a relationship between path equations that are derivable in the above system and those induced by inequalities produced by the above procedure. In part 2 of Lemma 4, what it means for a term t to be "assigned to a variable $\alpha \in X$ in Γ'" was defined earlier, after the definition of "redex I" and "redex II." A path equation is "derivable from Γ" if it is derivable from the path equations induced by the inequalities in Γ.

LEMMA 4. *Let Γ be an instance of SUP with all varibles in the initial X.*

1. *If a path equation is derivable then it is satisfied by every solution of Γ. In particular, if "$\alpha_w = \Pi\beta_v$" is derivable, and $(S, S_1, ..., S_n)$ is a solution of Γ then $\bar{S}(\beta_v)$ is of depth at least equal to $|\Pi|$.*

2. *Let Γ' be obtained from Γ by reducing some number of redexes (I or II), and let t be the term assigned to $\alpha \in X$ in Γ'. If a path equation is induced by the equation $\alpha = t$ then it is derivable from Γ.*

Proof. Part 1 of the lemma is proved by a routine induction on the length of a derivation of a path equation. Part 2 is a straightforward consequence of the following claim:

Let Γ'' be an instance obtained from Γ' by reducing a single redex (I or II), and let t' and t'' be the terms assigned to a variable $\alpha \in X$ in Γ' and Γ'', respectively. All path equations induced by Γ'' and the equation $\alpha = t''$ are derivable from Γ' and the equation $\alpha = t'$.

We prove that all path equations induced by Γ'' are derivable from Γ', leaving the other part of the claim to the reader (similar proof). Consider a path equation, say $\xi_1 = \Pi\xi_2$, induced by Γ'', i.e., there is an inequality $t'' \leqslant_i u''$ of Γ'' such that $\xi_1 = \Pi\xi_2$ is induced by the equation $(t'')_i = u''$. Under these conditions, it is easy to see that there are variables η_1 and η_2, and $\Delta, \Sigma_1, \Sigma_2 \in \{L, R\}^*$, such that

(a) the path equation $\xi_1 = \Sigma_1\eta_1$ is induced by Γ',

(b) the path equation $\xi_2 = \Sigma_2\eta_2$ is induced by Γ',

(c) the path equation $\eta_1 = \Delta\eta_2$ (or, respectively, the path equation $\eta_2 = \Delta\eta_1$) is induced by the equation $(t')_i = u'$, where $t' \leqslant_i u'$ is the ith inequality of Γ',

(d) $\Sigma_1\Delta = \Pi\Sigma_2$ (or, respectively, $\Sigma_1 = \Pi\Sigma_2\Delta$).

The variable η_1 appears in $(t')_i$, and η_2 in u', or vice-versa. Because Γ'' is obtained from Γ' by reducing a single redex, not both $\Sigma_1 \neq \varepsilon$ and $\Sigma_2 \neq \varepsilon$, except possibly in case $\eta_1 = \eta_2$. If both of η_1 and η_2 appear in Γ'', then $\eta_1 = \xi_1$ and $\eta_2 = \xi_2$ (and $\Sigma_1 = \Sigma_2 = \Delta = \varepsilon$), so that the path equation $\xi_1 = \Pi\xi_2$ is also induced by (and therefore derivable from) Γ'. To conclude the proof, suppose that one of these two variables, say η_1, does not appear in Γ''. Hence, Γ'' is obtained from Γ' by substituting some term u^* for all occurrences of η_1 throughout Γ', with variable ξ_1 appearing in u^*. It must then be that the equation $\eta_1 = u^*$ is induced by Γ', from which we can also verify assertions (a) and (b) above. Using the rules of inference (transitivity, symmetry and subterm) and the equality in (d), it now readily follows that $\xi_1 = \Pi\xi_2$ is derivable from the path equations in (a), (b), and (c); i.e., it is derivable from Γ'. ∎

3. Mortal and Bounded Turing Machines

Let M be a deterministic Turing Machine with set of states Q and tape alphabet A. We assume that M has one tape, infinite in both directions, and that tape cells are numbered by integers. An *instantaneous description*

(ID) of M is understood here as a triple $\langle \alpha, m, f \rangle$, where $f: \mathscr{Z} \to A$ ($\mathscr{Z}$ stands for the set of all integers) is the tape contents, $\alpha \in Q$ is the current state, and $m \in \mathscr{Z}$ is the position of the head.

This definition slightly differs from the usual one. Ordinarily, it is assumed that $f(k)$ is equal to a blank symbol, for almost all k. Such ID's will be called *finite*, and we will also use more conventional notation $\langle w_1, \alpha, m, w_2 \rangle$ for finite ID's: here the tape contents is represented by $w_1 w_2 \in A^*$, and the tape head is assumed to scan the first symbol in w_2. For the purpose of this section we need however to consider arbitrary, possibly infinite ID's.[7] It should be stressed that the properties of Turing Machines we discuss below have nothing to do with any *initial* states: we could just as well assume that our machines have no distinguished initial states at all. In particular, it does not matter whether a given ID is reachable from an initial one or not.

A deterministic Turing Machine M is said to be *bounded* iff there is a positive integer k such that, if C is an arbitrary ID of M, then the number of different ID's reachable by M from C is at most k. Put differently, M is bounded iff there is l, such that no sequence of moves may cause the tape head to go more than l cells to the right or to the left from an initial position. Clearly, it suffices to require the above property to hold for all finite ID's because if it holds in this case then the number of reachable configurations for all ID's is also bounded.

A related property is the following: M is *mortal* if it halts when started from an arbitrary (possibly infinite) ID; otherwise M is called *immortal.* Note that now a restriction to finite ID's would result in a different property.

The *boundedness* (resp. *immortality*) *problem* is to decide, for a given deterministic Turing Machine M, whether M is bounded (resp. mortal) or not. The following theorem, due to Philip K. Hooper ([6]), and its corollary play a crucial role in the proof of the undecidability of SUP.

THEOREM 5. *The immortality problem for deterministic Turing machines is undecidable.*

COROLLARY 6. *The boundedness problem for deterministic Turing machines is undecidable.*

Proof. In fact, the original proof in [6] carries over without substantial changes to an undecidability proof for the boundedness problem (and also

[7] Another difference is that we number the tape cells: this is chosen to distinguish between ID's that are identical except that one is obtained from another by a left or right shift. Such ID's would be identified by the usual definition, the ambiguity of which was pointed to the authors by Rohit Parikh.

for the immortality problem restricted to finite ID's). For the sake of completeness we provide a direct proof that Theorem 5 implies the undecidability of boundedness. We first show that mortality implies boundedness.

Let M be a deterministic Turing machine, and let $C_1, C_2, \ldots$ be an arbitrary (finite or infinite) sequence of ID's of M, such that each C_{i+1} is obtained from C_i in one step. Let $C_i = \langle \alpha_i, m_i, f_i \rangle$ and let $b_i = f_i(m_i)$, the currently scanned tape symbol. The sequence of pairs (α_i, b_i), obtained in such a way, is called a *computation history* of M. Consider now the set of all finite computation histories ordered by the prefix relation. This is obviously a tree with a finite fan-out, with the empty history as the root. We claim that each infinite branch of our tree is an infinite computation history.

For this, observe that, informally speaking, a history uniquely determines a fragment of initial tape (a subset of f_1) which has been visited during the computation, and all changes that have been made. If a chain of histories is given, then the appropriate fragments of initial tape (partial functions) must be consistent, and must also form a chain (with respect to inclusion). The union of these functions describes a (fragment of) an ID of M. If M is started on this ID, the corresponding history will be exactly our infinite branch.

If there are arbitrarily long histories, then, by König's Lemma, we would have an infinite branch, and thus an infinite history, corresponding to a non-terminating computation. Thus, if M is mortal then there is a bound for the length of an arbitrary history, and hence M must be bounded.

Suppose now that the boundedness problem is decidable. We show that the immortality problem must also be decidable. Let a machine M be given. In order to verify the mortality of M, we first ask if M is bounded. If not, then it must be immortal, by the above observation. Otherwise, each computation of M must be fully performed within a bounded part of the tape, and the number of different possible computations is essentially finite (modulo the unseen part of the tape). An exhaustive verification of all possible sequences of moves makes it possible to determine if the machine must always terminate, or if it can enter a loop. ▮

As we have already observed, the boundedness condition holds for all ID's, provided it holds for finite ID's. Thus, we do not need to consider infinite ID's anymore, as only finite ID's are reachable from a finite ID. From now on, "ID" will always mean "finite ID".

We shall reduce the boundedness problem to the semi-unification problem, which will imply the undecidability of semi-unification. First, however, we need some additional considerations of technical nature. The main reduction will be described in Section 4. A Turing Machine is usually

understood so that at each stage, the head of the machine scans a single tape cell, and it may move left, move right, or remain at that cell, depending on the internal state and the scanned symbol. It will be more convenient for us to assume that the machine head is always positioned *between* two tape cells, and it must always move right or left. Let us formulate this more precisely. An *Intercell Turing Machine* (ITM) is a triple $Y = \langle Q, A, T \rangle$, where

- Q is a finite set of *states*;
- A is a finite *tape alphabet*;
- $T \subseteq Q \times \{-1, +1\} \times A \times A \times Q$ is a *transition relation.*

An instantaneous description (ID) of Y takes the form $\langle w_1, \alpha, m, w_2 \rangle$, and is understood as in the case of conventional Turing Machines, except that the head is positioned *between* the $(m-1)$th and the mth cells of the tape. The *next move* relation $\vdash_Y$ on ID's of Y is defined as follows:

$$\langle w_1 a, \alpha, m, w_2 \rangle \vdash_Y \langle w_1, \beta, m-1, bw_2 \rangle, \quad \text{for} \quad \langle \alpha, -1, a, b, \beta \rangle \in T;$$

$$\langle w_1, \alpha, m, aw_2 \rangle \vdash_Y \langle w_1 b, \beta, m+1, w_2 \rangle, \quad \text{for} \quad \langle \alpha, +1, a, b, \beta \rangle \in T.$$

The notion of reachability for ID's and the notion of a *bounded* ITM are defined in the obvious way. An ITM is *deterministic* iff the following conditions hold:

- the set of states Q splits into two disjoint subsets Q_L and Q_R, so that $T \subseteq (Q_L \times \{-1\} \times A \times A \times Q) \cup (Q_R \times \{+1\} \times A \times A \times Q)$;
- T is a partial function from $Q \times \{-1, +1\} \times A$ to $A \times Q$.

Thus, for a deterministic ITM there is at most one move possible from any given ID, the direction of the move being determined by the internal state.

LEMMA 7. *Let M be a deterministic Turing Machine. One can effectively construct a deterministic intercell Turing Machine Y, such that M is bounded iff Y is bounded. In addition, one can require that the tape alphabet of Y consists of eactly two elements, including the blank symbol.*

Proof. Using the well-known methods, one can first obtain a deterministic Turing Machine M', such that M' is bounded iff M is bounded, and such that the tape alphabet of M' consists of two symbols only. Now, Y is constructed so that it simulates each move of M' by at most three moves, and does not use new tape symbols. It follows that the number of reachable ID's grows at most three times, and remains bounded. ∎

Let $Y = \langle Q, A, T \rangle$ be an ITM. The *symmetric closure* of Y is $Y_S = \langle Q, A, T_S \rangle$, where

$$T_S = T \cup \{(\alpha, -x, a, b, \beta) : (\beta, x, b, a, \alpha) \in T\}.$$

If $Y = Y_S$, then Y is called *symmetric*. Clearly, the symmetric closure of a deterministic machine, is no longer deterministic. However, it preserves boundedness.

LEMMA 8. *Let Y be a deterministic ITM. If Y is bounded then so is Y_S. (The converse is obvious.)*

Proof. Assume that Y is bounded. There exists a constant l such that, if C is an arbitrary ID of Y, and C' is reachable from C then the head positions in C and C' differ by at most l. Now suppose that Y_S is unbounded. Then there must exist a sequence $C_0, C_1, ..., C_p$ of ID's such that, for all $i = 0, ..., p$:

- $C_i = \langle w_i, \alpha_i, m_i, w_i' \rangle$;
- $C_i \vdash_{Y_S} C_{i+1}$, for $i < p$;
- $|m_0 - m_p| > 2l$.

One can assume without loss of generality that $m_p > m_0 + 2l$ (because of the symmetry) and that, $m_0 \leqslant m_i \leqslant m_p$, for all $i = 0, ..., p$ (otherwise, we choose an appropriate fragment of the sequence from the leftmost to the rightmost position of the tape head).

Clearly, for $i < p$ we have either $C_i \vdash_Y C_{i+1}$ or $C_{i+1} \vdash_Y C_i$. If it happens for some i that $C_i \vdash_Y C_{i+1}$ and $C_i \vdash_Y C_{i-1}$, then $C_{i+1} = C_{i-1}$. The sequence $C_0, ..., C_{i-1}, C_{i+2}, ..., C_p$ still describes a valid computation of Y_S with the tape head moved more than $2l$ cells to the right. Thus, if we choose our sequence to be a shortest possible, then there must be $i \in \{0, ..., p\}$, such that C_i is reachable in Y from both C_0 and C_p. The tape head in C_i is at the position m_i, between m_0 and m_p. Thus, $m_i - m_0$ or $m_p - m_i$ must be greater than l, and we see that Y is able to move its head by more than l cells. This contradicts our assumption. ∎

The following lemma puts together the sequence of reductions we defined so far, and will be used in the next section.

LEMMA 9. *It is undecidable whether a symmetric ITM over a two-element tape alphabet is bounded.*

4. From ITM to SUP

From now on, let $Y = \langle Q, A, T \rangle$ be a symmetric ITM. Our goal is to construct an instance Γ of SUP such that Γ has a solution iff Y is bounded. We assume that the tape alphabet A consists of only two elements, 0 and 1, with 0 used as the blank symbol. In order to simplify the construction to follow, we first slightly modify Y by adding new states as follows.

Let us consider all quintuples of T of the form $\langle \alpha, -1, x, i, \beta \rangle$, where $\alpha, \beta \in Q$, and $x, i \in \{0, 1\}$. (We choose only left moves because of the symmetry.) Let $S_1, ..., S_p, S_{p+1}, ..., S_r$ be all such quintuples and let $S_l = \langle \alpha^l, -1, x^l, i^l, \beta^l \rangle$, for $l \in \{1, ..., r\}$, where $i^l = 0$ if $l \leqslant p$, and $i^l = 1$, otherwise. Now, let $\gamma^l \notin Q$, for $l \in \{1, ..., r\}$, and let $Q' = Q \cup \{\gamma^1, ..., \gamma^r\}$. The set Q' is the set of states of a new machine $Y' = \langle Q', A, T' \rangle$, where T' is obtained from T by adding new quintuples of the form $\langle \alpha^l, -1, 1 - x^l, i^l, \gamma^l \rangle$ and $\langle \gamma^l, +1, i^l, 1 - x^l, \alpha^l \rangle$, for all $l \in \{1, ..., r\}$. The reason for this modification will soon become obvious, but now we would like to turn the reader's attention to the fact that the new states do not essentially affect the behaviour of our machine. Indeed, if $C_1 \vdash_{Y'} C_2 \vdash_{Y'} C_3$, and C_2 contains a state γ^l, then C_1 and C_3 must be identical ID's. The maximum number of ID's reachable in Y' from a given ID, may thus be at most twice as large as the corresponding bound for Y (provided it is finite). We conclude that Y' is bounded iff Y is bounded.

Now we can finalize our reduction by constructing an instance Γ of SUP. The variables of Γ will be exactly the states of Y'. The construction of Γ will be such that the path equations induced by inequalities in Γ correspond to possible moves of Y'. Each ID of Y' will be interpreted so that the tape contents to the left of the head represents a path in a tree (0 stands for "left", and 1 for "right"), while the word to the right of the head denotes a sequence of unknown substitutions.

Let $l \in \{1, ..., r\}$, and let S_l be as above. We define a term t^l to be

$$\beta^l \to \gamma^l, \qquad \text{if} \quad x^l = 0;$$
$$\gamma^l \to \beta^l, \qquad \text{if} \quad x^l = 1.$$

The inequalities of Γ are as follows:

$$t^1 \to t^2 \to \cdots \to t^p \leqslant_0 \alpha^1 \to \alpha^2 \to \cdots \to \alpha^p;$$
$$t^{p+1} \to t^{p+2} \to \cdots \to t^r \leqslant_1 \alpha^{p+1} \to \alpha^{p+2} \to \cdots \to \alpha^r.$$

Note that all path equations induced by the inequalities in Γ are of the form

$$\beta_i = L\alpha \qquad \text{or} \qquad \beta_i = R\alpha,$$

where $\alpha, \beta \in Q'$, and $i \in \{0, 1\}$. That is, informally, the left-hand sides of all inequalities are everywhere "deeper" than the corresponding right-hand sides by exactly one level. One observation that easily follows from this property is that all derivable path equations "$\Pi\alpha_w = \Sigma\beta_v$" satisfy $|\Pi| + |w| = |\Sigma| + |v|$. Let $\varphi: \{L, R\}^* \to \{0, 1\}^*$ be the isomorphism defined by $\varphi(L) = 0$ and $\varphi(R) = 1$. The following lemma states the crucial relationship between Y and Γ.

LEMMA 10. *Let $\Pi, \Sigma \in \{L, R\}^*$ and $w, v \in \{0, 1\}^*$ be such that $|\Pi| + |w| = |\Sigma| + |v|$. Let $p = |\Pi| - |\Sigma|$ and let $\alpha, \beta \in Q'$. Then the following conditions are equivalent:*

(1) *The path equation "$\Pi\alpha_w = \Sigma\beta_v$" is provable in the proof system of Section 2;*

(2) *For every $m \in \mathscr{N}$, the ID's $\langle \varphi(\Pi), \alpha, m, w \rangle$ and $\langle \varphi(\Sigma), \beta, m - p, v \rangle$ are reachable in Y' from each other.*

Proof. The implication (1) $\Rightarrow$ (2) follows by an easy induction on the lengths of proofs. An axiom of the form $\beta_i = L\alpha$ corresponds, by the construction of Γ, to a pair of mutually reachable ID's, namely $\langle \varepsilon, \beta, m, i \rangle$ and $\langle 0, \alpha, m + 1, \varepsilon \rangle$. (Here, ε denotes the empty word.) Similarly, an axiom $\beta_i = R\alpha$ corresponds to $\langle \varepsilon, \beta, m, i \rangle$ and $\langle 1, \alpha, m + 1, \varepsilon \rangle$. The induction steps are obvious.

The proof of (2) $\Rightarrow$ (1) is done by an induction on the number of steps needed to reach one of the ID's from the other. Indeed, every step of the machine corresponds to a provable path equation. For example, let $\langle \varphi(\Pi), \beta, m - 1, 0w \rangle$ be obtained from $\langle \varphi(\Pi R), \alpha, m, w \rangle$ as a result of executing $\langle \alpha, -1, 1, 0, \beta \rangle$. Then the path equation "$R\alpha = \beta_0$" is induced by Γ and, thus, is an axiom. By the subterm, symmetry and instantiation rules, we get $\Pi\beta_{0w} = \Pi R\alpha_w$. The remaining cases of the base step (configurations reachable in one move) are analogous. For the induction step, one uses the transitivity rule. ∎

Finally, we arrive at our main lemma:

LEMMA 11. *Y is bounded iff Γ has a solution.*

Proof. If Γ has no solution then, by Lemma 2, the procedure of Section 2 assigns arbitrarily large terms to some variable α. It follows from Lemma 4, that one can derive path equations of the form "$\Pi\alpha = \beta_v$", with arbitrarily long Π. Take $k > 0$ and let $|\Pi| = l > k + 1$.

By Lemma 10, we have $\langle \varepsilon, \beta, 0, v \rangle$ reachable from $\langle \Pi, \alpha, l, w \rangle$. But the head positions at these ID's differ at least by k. Since k was arbitrary, we conclude that Y', and therefore Y, is unbounded.

Now assume that Γ has a solution S. Let k be such that the depth of all terms $S(\alpha)$, for $\alpha \in Q'$ is at most k. Suppose that Y is unbounded. Then Y' is unbounded too and thus there are mutually reachable ID's of Y' such that the difference between the head positions is more than k. Take the sequence of moves that transforms one of the ID's into the other, and choose the leftmost and the rightmost position of the tape head during this sequence of moves. We have two ID's of the form: $\langle \varphi(\Sigma), \alpha, m, wv \rangle$ and $\langle \varphi(\Sigma\Pi), \beta, m+p, v \rangle$, reachable from each other so that the tape head never moves left past the mth cell and never moves right past the $(m+p)$th cell (here, $p = |\Pi| > k$). One can easily see that also $\langle \varepsilon, \alpha, m, w \rangle$ and $\langle \Pi, \beta, m+p, \varepsilon \rangle$ are reachable from each other. By Lemma 10, we have a provable path equation "$\alpha_w = \Pi\beta$", with $|\Pi| > k$. But, by Lemma 4, it follows that $S(\beta)$ must be of depth greater than k, a contradiction. ∎

Here is our main result:

THEOREM 12. *The semi-unification problem is undecidable.*

Proof. Let Y be a symmetric intercell Turing Machine. By Lemma 11, we can effectively construct an instance Γ of SUP, which has a solution iff Y is bounded. The conclusion now immediately follows from Lemma 9. ∎

Let us note that the instance Γ, constructed in the above proof has just 2 inequalities. Thus, our reduction provides yet another proof that the special case of SUP with 2 inequalities is as difficult as the general case (cf. [19]).

ACKNOWLEDGMENT

We were referred to Hooper's work by Albert Meyer. His help is greatly appreciated.

RECEIVED August 9, 1990; FINAL MANUSCRIPT RECEIVED January 4, 1991

REFERENCES

[1] CHANDRA, A. K., AND VARDI, M. Y. (1985), The implication problem for functional and inclusion dependencies is undecidable, *SIAM J. Comput.* **14**, No. 3, 761–677.

[2] DÖRRE, J., AND ROUNDS, W. (1990), On subsumption and semiunification in feature algebras, *in* "Proceedings, 5th IEEE Symposium Logic in Computer Science," pp. 300–310.

[3] HENGLEIN, F. (1988), Type inference and semi-unification, *in* "Proceedings, ACM Symposium LISP and Functional Programming," pp. 184–197.

[4] HENGLEIN, F. (1989), "Polymorphic Type Inference and Semi-unification," Technical Report 443, Computer Science Department, New York University.

[5] HENGLEIN, F. (1990), Fast left-linear semi-unification, *in* "Proceedings, International Conference on Computing and Information, May 23–26, 1990, Niagara Falls," pp. 32–36, Canadian Scholar's Press, Toronto.

[6] HOOPER, P. K. (1966), The undecidability of the Turing machine immortality problem, *J. Symbolic Logic* **31**, No. 2, 219–234.

[7] KANELLAKIS, P., AND MITCHELL, J. C. (1989), Polymorphic unification and ML typing, *in* "Proceedings, 16th Symposium on Principles of Programming Languages," pp. 105–115.

[8] KANELLAKIS, P. (1990), private communication.

[9] KAPUR, D., MUSSER, D., NARENDRAN, P., AND STILLMAN, J. (1988), Semi-unification, *in* "Proceedings of 8th Conference on Foundations of Software Technology and Theoretical Computer Science, Pun India" (Nori and Kumar, Eds.), pp. 435–454, Lecture Notes in Computer Science, Vol. 338, Springer-Verlag, Berlin/New York.

[10] KFOURY, A. J., TIURYN, J., AND URZYCZYN, P. (1988), A proper extension of ML with an effective type-assignment, *in* "Proceedings, 15th ACM Symposium Principles of Programming Languages," pp. 58–69.

[11] KFOURY, A. J., TIURYN, J., AND URZYCZYN, P. (1989), "Type-Checking in the Presence of Polymorphic Recursion," Boston University Research Report. Part of the results of this paper have been presented in Computational consequences and partial solutions of a generalized unification problem, *in* "Proceedings of IEEE 4th Symposium Logic in Computer Science," pp. 98–105 (1989).

[12] KFOURY, A. J., TIURYN, J., AND URZYCZYN, P. (1990), An analysis of ML typability, *in* "15th Colloquium on Trees in Algebra and Programming, CAAP'90" (Arnold, Ed.), pp. 206–220, Lecture Notes in Computer Science, Vol. 431, Springer-Verlag, Berlin/New York.

[13] KFOURY, A. J., AND TIURYN, J. (1990), Type reconstruction in finite-rank fragments of the second-order λ-calculus, *in* "Proceedings, 5th IEEE Symposium Logic in Computer Science," pp. 2–11.

[14] LEIß, H. (1987), On type inference for object-oriented programming languages, *in* "Proceedings, 1st Workshop on Computer Science Logic" (Börger, Büning, and Richter, Eds.), pp. 151–172, Lecture Notes in Computer Science, Vol. 329, Springer-Verlag, Berlin/New York.

[15] LEIß, H. (1990), Polymorphic recursion and semi-unification, *in* "Proceedings, of the 3rd Workshop on Computer Science Logic, Kaiserslautern, Germany, October 1989" (Börger *et al.* Eds.), pp. 211–224, Lecture Notes in Computer Science, Vol. 440, Springer-Verlag, Berlin/New York.

[16]. MAIRSON, H. G. (1990), Deciding ML typability is complete for deterministic exponential time, *in* "Proceedings, 16th ACM Symposium Principles of Programming Languages," pp. 382–401.

[17] MITCHELL, J. C. (1983), The implication problem for functional and inclusion dependencies, *Inform. and Control* **56**, 154–173.

[18] MYCROFT, A. (1984), Polymorphic type schemes and recursive definition, *in* "International Symposium on Programming" (Paul and Robinet, Eds.), pp. 217–228.

[19] PUDLÁK, P. (1988), On a unification problem related to Kreisel's conjecture, *Comment. Math. Univ. Carolin.* **29**, No. 3, 551–556.

[20] PURDOM, P. W. (1987), Detecting looping simplifications, *in* "Proceedings, 2nd Conference on Rewriting Techniques and Applications (RTA)" (Lescanne, Ed.), pp. 54–62, Lecture Notes in Computer Science, Vol. 250, Springer-Verlag, Berlin/New York.

[21] STATMAN, R. (1988), private communication.