Stanford Verification Group
Report No. 15

Computer Science Department
Report No. STAN-CS-80-785

January 1980

EQUATIONS AND REWRITE RULES
A Survey

by

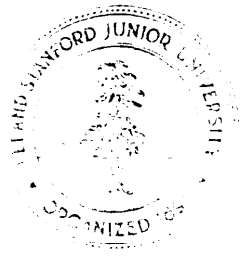
Gerard Huet and Derek C. Oppen

Research sponsored **by**

National Science Foundation

Air Force Office of Scientific Research

COMPUTER SCIENCE DEPARTMENT
Stanford University



Stanford Verification Group
Report No. 15

'January 1980

Computer Science Department
Report No. STAN-CS-80-786

EQUATIONS AND REWRITE RULES
A Survey

by

Gerard Huet and Derek C. Oppen

ABSTRACT

Equations occur frequently in mathematics, logic and computer science. in this paper, we survey the main results concerning equations, and the methods avilable for reasoning about them and computing with them. The survey is self-contained and unified, using traditional abstract algebra.

Reasoning about equations may involve deciding if an equation follows from a given set of equations (axioms), or if an equation is true in a given theory. When used in this manner, equations state properties that hold between objects. Equations may also be used as definitions; this use is well known in computer science: programs written in applicative languages, abstract interpreter definitions, and algebraic data type definitions are clearly of this nature. When these equations are regarded as oriented "rewrite rules", we may actually use them to compute.

In addition to covering these topics, we discuss the problem of "solving*' equations (the "unification" problem), the problem of proving termination of sets of rewrite rules, and the decidability and complexity of word problems and of combinations of equat i onal theories. We restrict ourselves to first-order equations, and do not treat equations which define non-terminating computations or recent work on rewrite rules applied to equational congruence classes.

This research was supported by the National Science Foundation under Contracts NSF MCS78-02835 and NSF MCS79-04012,, and Air Force Office of Scientific Research under Contract AFOSR-S752. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of Stanford University, or any agency of the U.S. Government.

Equations and Rewrite Rules

A Survey

Gérard Huet
IRIA and **SRI** International

Derek C. **Oppen**
Stanford University

Table of Contents

Addressee: Computer Science Laboratory, SRI International, 333 Ravenswood Ave., Menlo Park, Ca. 94025. Computer Systems Laboratory, Computer Science Department, Stanford University, Stanford, Ca. 94305. **An earlier version of this paper was presented at the Conference on Formal Language Theory, Santa Barbara, December 1979, and will appear in "Formal Languages: Pcrepcctivce and Open Problems", cd. Ron Book, Academic Press, 1980. Preparation of this paper was supported in part by the National Science Foundation under grants MCS78-02835 and MCS79-04012, and in part by the Air Force Office of Scientific Research under contract AFOSR-8752.**

1. Introduction

Equations — formulas of the form $M = N$ — occur frequently in mathematics, logic and computer science. In this paper, we survey in a uniform fashion the main results concerning equations, and the methods available for reasoning about them and computing with them.

Reasoning about equations may involve deciding if an equation "follows" or is a consequence of a given set of equations (axioms), or if an equation is "true" in a given theory.

Consider the following set of axioms for group theory:

$$x + 0 = x$$
$$x + -x = 0$$
$$(x + y) + z = x + (y + z).$$

We might wish to know if another equation $0 + x = x$ is a consequence of these axioms, that is, is true in all groups. This is an example of "reasoning" about equations. As is well known, this equation can be shown to follow from the axioms solely by equational reasoning, that is, substituting arbitrary terms for variables and replacing "equals by equals".

When used in this manner, equations state properties that hold between objects.

Equations may also be used as ***definitions.*** For instance, a set of equations may define a particular group (we then speak of the group presented by the equations). The use of equations as definitions is well known in computer science since programs written in applicative languages, abstract interpreter definitions, and algebraic data type definitions are clearly of this nature. When these equations are regarded as oriented "rewrite rules", we may actually use them to compute. For instance, the following equations define the function ***Append***:

$$Append(Null, x) = x$$
$$Append\ (Cons(x, y), z) = Cons(x, Append\ (y, z)).$$

We can use this definition to compute, say, $Append(Cons(A, Cons(B, Null)), Cons(C, Null))$ into $Cons(A, Cons(B, Cons(C, Null)))$.

Computing with equations and reasoning about equations are closely related. For instance, one method for reasoning about equations consists in "compiling" them into rewrite rules and using them to reduce expressions to canonical form. We shall show later how this method applies to the group axioms above, and how

$0 + x = x$ can be proved by reducing both sides of the equality to the canonical form x.

However, reasoning about equations considered as definitions requires more than just equational reasoning. For instance, it is well known that *Append* is associative, but proving the theorem requires some kind of induction.

It is useful to start with a clear understanding of the various semantics one can associate with equations and with rewrite rules. We shall therefore first give various semantic definitions, and then consider their associated proof theories (if any). We shall develop these notions in the framework of traditional abstract algebra.

The survey attempts to be fairly comprehensive. However, for lack of space, we restrict ourselves to first-order equations. Further, we do not treat equations which define non-terminating computations; for them a more sophisticated model theory is required (continuous domains or algebras). Nor do we treat recent work on rewrite rules applied to equational congruence classes (we give an outline of this research in section 14).

2. Sorted **Algebras**

We assume we are given a set $\mathcal{S}$ of *sorts,* which are names for the various domains under consideration. We deal here only with simply sorted theories, in which these domains are disjoint.

Example: $\mathcal{S}^0 =$ {integer, boolean}.

We now define a language Θ of types, obtained by composing sorts. We restrict ourselves to first-order theories, and assume that a type is just a finite sequence of sorts: $\Theta = \mathcal{S}^+$. We write $t = s_1 \times s_2 \times \text{-} \times s_n \rightarrow$ s instead of $t = \langle s_1, s_2, \ldots, s_n, \text{s}\rangle$. A type denotes the type of operators (that is, mappings) with n arguments of sorts $s_1, s_2, \ldots, s_n$ and which return a value of sort s. When $n = 0$, we write $t =$ s; in this case, t denotes a constant of sort s.

A signature over Y consists of a set Σ of operators, together with a typing function $\tau{:}\Sigma \rightarrow \Theta$. For ease of notation, we assume that τ is given, and we use Σ to denote the signature.

If Σ is a signature over Y, we say that sort s in $\mathcal{S}$ is strict in Σ if and only if there exists an F in Σ such that $\tau(F) =$ s or $\tau(F) = s_1 \times s_2 \times \ldots \times s_n \rightarrow$ s and s_i is strict in Σ for each i. We say that the signature Σ is sensible if and only if, for every F in Σ such that $\tau(F) = s_1 \times s_2 \times \ldots \times s_n \rightarrow$ s, if s is strict then so are all the s_i. We assume from now on that all signatures are sensible; this restriction is technical and will be motivated later. (Note that every signature can be extended to a sensible one by adding constant symbols.)

Example: Given $\mathcal{Y}^0$ above, consider the signature $\Sigma^0 = \{O, S, +, TRUE, FALSE, \sharp\}$, with

$$\tau(0) = \text{integer}$$
$$T(S) = \text{integer} \to \text{integer}$$
$$\tau(+) = \text{integer} \times \text{integer} \to \text{integer}$$
$$\tau(TRUE) = \tau(FALSE) = \text{boolean}$$
$$\tau(\sharp) = \text{integer} \times \text{integer} \to \text{boolean}.$$

Note that every sort in $\mathcal{Y}^0$ is strict in Σ^0.

A C-algebra or algebra is a pair $\langle \mathcal{A}, 4 \rangle$ such that $\mathcal{A}$ is an Y-indexed family of sets and' $\mathcal{F}$ is a C-indexed family of functions (the fundamental operations of the algebra), with:

$\mathcal{F}_F \in \mathcal{A}_s$ if $\tau(F) =$ s, and
$\mathcal{F}_F \in \mathcal{A}_{s_1} \times \mathcal{A}_{s_2} \times \ldots \times \mathcal{A}_{s_n} \to \mathcal{A}_s$ if $\tau(F) = s_1 \times s_2 \times \ldots \times s_n \to s$.

$\mathcal{A}_s$ is called the carrier or universe of sort s in the algebra. We shall usually denote an algebra by its family of carriers if no ambiguity arises. For instance, we talk about the algebra $\mathcal{A}$ above, and use F to denote the corresponding operator $\mathcal{F}_F$. Note that $\mathcal{A}_s \neq \emptyset$ for any strict sort s.

Example: $\mathcal{A}^0$ is a CO-algebra, with $\mathcal{A}^0_{integer} = \mathcal{N}$, $\mathcal{A}^0_{boolean} =$ {true, false}, 0 is the integer 0, S is the successor function, $TRUE$ = **true**, $FALSE$ = **false**, $\sharp(m, n)$ is true if $m \neq n$ and false otherwise.

We now define special types of algebras, whose carriers will consist of finite ordered trees, **labelled** with operators from Σ. These (free) C-algebras will be helpful in defining terms.

With every operator F in Σ we associate a tree consisting of one node **labelled** F. We now define recursively the family of sets of trees $\mathcal{T}(\Sigma)_s$, for s $\in$ Y. If T(C) = s, then the tree with one node **labelled** C and no successors is an element of $\mathcal{T}(\Sigma)_s$. If $\tau(F) = s_1 \times s_2 \times \ldots \times s_n \to$ s and $M_i \in \mathcal{T}(\Sigma_i)_{s_i}$ for all $1 \leq i \leq n$, then the tree with one node **labelled** F and n successors $M_1, \ldots, M_n$ is an element of $\mathcal{T}(\Sigma)_s$. By abuse of notation, we will denote these terms by strings in the usual manner, as in, for example, C or $F(M_1, \ldots, M_n)$.

The corresponding algebra $\mathcal{T}(\Sigma)$ is called the initial algebra. We call $\mathcal{T}(\Sigma)_s$ the set of ***ground terms*** of sort s. For instance, $0 + S(0)$ is a ground term of sort integer, over Σ^0 above. Ground terms are called ***words*** in algebra and abstract syntax ***trees*** in computer science. Logicians use them implicitly by considering concrete syntax (strings) defined by a non-ambiguous grammar.

Note that $\mathcal{T}(\Sigma)_s \neq \emptyset$ if and only if s is strict.

Let us now consider an Y-indexed family of sets $\mathcal{V}$. Elements of $\mathcal{V}_s$ are called variables of sort s. We assume that the variables of distinct sorts are distinct and that no variable is a member of Σ. We denote by $\Sigma \cup \mathcal{V}$ the signature composed of Σ plus every $\mathcal{V}_s$ considered as a set of constants of sort s. The corresponding algebra $\mathcal{T}(\Sigma \cup \mathcal{V})$ (when restricted to a C-algebra) is called the **free** C-algebra generated by $\mathcal{V}$, and its carrier $\mathcal{T}(\Sigma \cup \mathcal{V})_s$ is the set of ***terms*** of sort s. This terminology is consistent with standard algebra books such as Cohn (65) where, in the case of only one sort, $\mathcal{T}(\Sigma \cup \text{Or})$ would be denoted $W_\Sigma(\mathcal{V})$. Compare also with $\mathcal{T}_{\Sigma(\mathcal{V})}$ in Goguen (79) and $M(\Sigma, \mathcal{V})$ in Nivat (75).

A completely formal development of term structures would require a careful definition of tree domains, and terms as mappings from these domains to $\Sigma \cup \mathcal{V}$ respecting the types. We shall not do this here, but rather will informally use ***occurrences*** — sequences of integers denoting an access path in a term, à la Dewey decimal notation. For M in $\mathcal{T}(\Sigma \cup \mathcal{V})$, we denote by $O(M)$ the set of such occurrences of M, and, for u in $O(M)$, we use M/u to denote the ***subterm*** of M at ***occurrence*** u. For example, with $M = 0 + S(O)$, we have $O(M) = \{A, 1, 2, 2.1\}$, and $M/2 =$ S(0). We use $\mathcal{V}(M)$ to denote the set of variables occurring in M; that is, $x \in \mathcal{V}(M)$ if and only if $x \in \mathcal{V}$ and there exists a u in $O(M)$ such that $M/u = x$. Finally we define $M[u \leftarrow N]$ as the term M, in which we have replaced the **subterm** at occurrence u by N. (Occurrences were proposed by Gorn(67) and were subsequently used by Brainerd (68), Rosen (69) and Huet(77).)

We assume in the following that mappings between C-algebras are **sort-preserving**, and write $h: \mathcal{A} \to \mathcal{B}$ for a Y-indexed family of mappings $h_s: \mathcal{A}_s \to \mathcal{B}_s$.

Finally, if $\mathcal{A}$ and $\mathcal{B}$ are two algebras, we say that $h: \mathcal{A} \to \mathcal{B}$ is a E-morphism (or just morphism) if and only if, for all F in Σ, with $\tau(F) = s_1 \times s_2 \times \ldots \times s_n \to s$, we have $h_s(F_\mathcal{A}(a_1, \ldots a_n)) = F_\mathcal{B}(h_{s_1}(a_1), \ldots h_{s_n}(a_n))$.

Theorem. (The universal property for the free C-algebras) Let $\mathcal{A}$ be any Σ-algebra. Any **$\mathcal{A}$-assigment** $\nu: \mathcal{V} \to \mathcal{A}$ can be extended in a unique way to a Σ-morphism from $\mathcal{T}(\Sigma \cup \mathcal{V})$ to $\mathcal{A}$.

We use ν to denote both the assignment and its extension.

Examples:

1. The *trivial* algebra is ***Sort***, where ***Sort***, = {s} and $F(s_1, \ldots, s_n) =$ s when $\tau(F) = s_1 \times s_2 \times \ldots \times s_n \to$ s. The assignment τ defined by $\tau(x) =$ s if and only if $x \in \mathcal{V}_s$ extends in a unique way from $\mathcal{T}(\Sigma \cup \mathcal{V})$ to ***Sort***. For instance, over Σ^0 above, with $n \in \mathcal{V}_{integer}$, we have $\tau(S(0) + n) =$ integer.

2. **Σ-endomorphisms** from $\mathcal{T}(\Sigma \cup \mathcal{V})$ to itself are called substitutions. We shall generally be interested in the effect of such a substitution σ on a finite number of terms, and therefore **assume** that a(x) = x except on a finite set of variables $\mathcal{D}(\sigma)$

which we call the domain of σ by abuse of notation. Such substitutions can then be represented by the finite set of pairs $\{\langle x, \sigma(x)\rangle \mid x \in \text{a(a)}\}$. We say that σ is ***ground*** if and only if $\mathcal{V}(\sigma(x)) = \emptyset$ for any x in $\mathfrak{D}(\sigma)$.

One more definition is needed for the next section.

If $\mathcal{A}$ is an algebra, a sort-preserving relation $\sim$ on elements of $\mathcal{A}$ is called a C-congruence over $\mathcal{A}$ if and only if:

$$(\forall F \in \Sigma)\,(\forall a_1, b_1, \ldots, a_n, b_n \in \mathcal{A})$$
$$a_1 \sim b_1 \,\&\, \ldots \,\&\, a_n \sim b_n \Rightarrow F(a_1, \ldots, a_n) \sim F(b_1, \ldots, b_n).$$

3. **Equations** and Varieties

A E-equation or equation is a pair $M = \text{N}$ where M and N are members of $\mathcal{T}(\Sigma \cup \mathcal{V})_s$, that is, are formed from variables ranging over the elements of the universe and from symbols denoting the fundamental operations of the algebra. In equations, variables are (implicitly) universally quantified.

Let $\mathcal{A}$ be an algebra and $M = \text{N}$ be an equation. We say that $M = \text{N}$ is ***valid*** in $\mathcal{A}$ or that $\mathcal{A}$ is a model of $M = \text{N}$, and we write $\mathcal{A} \models M = \text{N}$, if and only if for every A-assignment $\nu: \mathcal{V}(M) \cup \text{Y(N)} \rightarrow \mathcal{A}$ we have $v(M) = \nu(N)$; that is, M and N denote the same element of the carrier A, no matter how the variables of M and N are interpreted as elements of A. $\mathcal{A}$ is a model of a set of equations if every A-assignment validates every equation in the set.

Examples:

$$\mathcal{A}^0 \models x + y = y + x$$
$$\mathcal{A}^0 \not\models 0 = \text{S(x)}$$
$$\textit{Sort} \models M = \text{N for every } M \text{ and N of the same sort.}$$

It is apparent that the relation $=_{\mathcal{A}}$ on $\mathcal{T}(\Sigma \cup \mathcal{V})_s$, defined by $M =_{\mathcal{A}} N$ if and only if $\mathcal{A} \models M = \text{N}$, is a C-congruence, provided Σ is a sensible signature (or else provided that every carrier in the algebra is non-empty).

Now a set of C-equations may be thought of as defining a family of algebras, or some unique (up to an isomorphism) algebra. The variety or equational class $\mathcal{M}\text{b}(\Sigma, \mathfrak{E})$ or simply $\mathcal{M}\text{b}(\mathfrak{E})$ is the class of all models of 8. That is, $\mathcal{A} \in \mathcal{M}\text{b}(\mathfrak{E})$ if and only if $\mathcal{A} \models E$ for every E in $\mathfrak{E}$.

Note that $\mathcal{M}\text{b}(\mathfrak{E})$ is never empty, since ***Sort*** is always in it.

Example: Let $\mathfrak{E}^0 = \{x + 0 = x, x + S(y) = S(x + \text{y})\}$. Define the Σ^0-algebra $\mathcal{A}^1$ by: $\mathcal{A}^1_{integer} = \mathcal{N}_{blue} \cup \mathcal{N}_{red}$, $\mathcal{A}^1_{\text{boolean}} = \{\textbf{true,false}\}$, $0 = 0_{blue}$, $\text{S} = Succ_{blue} \cup Succ_{red}$, $+ = \oplus$, with $n_{blue} \oplus m_{blue} = n_{blue} \oplus m_{red} = (n+m)_{blue}$, $n_{red} \oplus m_{blue} = n_{red} \oplus m_{red} = (n + m)_{red}$, $TRUE = \textbf{true}$, $FALSE = \text{f\&e}$, and

#(m, n) is **false** if m and n unpainted are the same and true otherwise. Then $\mathcal{A}^0$ and $\mathcal{A}^1$ are both models of $\mathfrak{E}^0$. However $\mathcal{A}^0$ and $\mathcal{A}^1$ are not isomorphic, and $x + y =$ y + x is not valid in A?

The validity ***problem*** in a class $\mathfrak{C}$ of algebras is: given an equation E over $\mathcal{T}(\Sigma \cup \mathcal{V})_s$, decide whether or not $\mathcal{A} \models E$ for every algebra $\mathcal{A}$ in $\mathfrak{C}$. If so, we write $\mathfrak{C} \models E$. When $\mathfrak{C} = \{\mathcal{A}\}$ and E is a ground equation, then the validity problem is called the word ***problem*** for A.

For instance, we shall consider the validity problem in the variety defined by $\mathfrak{E}$: $\mathcal{M}_0(\mathfrak{E}) \models E$. There is an obvious relationship between this notion and the traditional notion of validity in first-order equational logic. The only difference is that $\mathcal{M}_0(\mathfrak{E})$ may contain more than the usual first-order models of $\mathfrak{E}$, since we do not require the carriers of our algebras to be non-empty. $\mathcal{T}(\Sigma)_s$ coincides with the **Herbrand** universe of Σ of sort s whenever it is non-empty.

As in first-order logic, where semantic notions have an equivalent syntactic characterization through the completeness theorem, we have a proof theory corresponding to validity in a variety.

From now on, we use $\mathcal{T}$ instead of $\mathcal{T}(\Sigma \cup \mathcal{V})$ and $\mathfrak{G}$ instead of $\mathcal{T}(\Sigma)$ when there is no ambiguity.

4. Proof Theory

The ***equality*** $=_{\mathfrak{E}}$ generated by a set 8 of equations is the finest C-congruence over $\mathcal{T}$ containing all pairs $\langle \sigma(M), \sigma(N) \rangle$ for $M =$ N in $\mathfrak{E}$ and σ an arbitrary substitution. In equational logic, $=_{\mathfrak{E}}$ is called an equational theory, and $\mathfrak{E}$ a base or a set of axioms for the theory.

The following is the well-known completeness theorem of Birkhoff (35):

Theorem. $\mathcal{M}_0(\mathfrak{E}) \models M =$ N if and only if $M =_{\mathfrak{E}} N$.

This theorem tells us that an equation $M =$ N is true in the variety defined by 8 if and only if it can be obtained from the equations in 8 by substitutions and by replacing equals by equals. That is, two terms are equivalent if and only if they can be shown equivalent in a finite number of proof steps. Whenever 8 is a recursive set of axioms, this gives us a semi-decision procedure for the validity problem in the variety. (The theorem assumes that $\mathcal{V}_s$ is a denumerable set, for every s in Y. However, in certain varieties, it is enough to consider finite sets of variables.)

Now we need the notions of satisfiability and consistency.

We say that $M =$ N is satisfiable in $\mathcal{A}$ if and only if there is an A-assignment $\nu: \mathcal{V}(M) \cup \mathcal{V}(N) \to \mathcal{A}$ such that $\nu(M) =$ u(N). Equation $M =$ N is satisfiable (in

the variety of $\mathfrak{E}$) if and only if there is some algebra $\mathcal{A} \in \mathcal{M}(\mathfrak{E})$ such that $M = N$ is satisfiable in $\mathcal{A}$.

Thus far, the satisfiability problem is vacuous: ***Sort*** validates any equation. However, one may be interested in knowing whether there exist non-trivial algebras validating some equations. In order to do this, we assume we have the distinguished sort boolean, and distinguished constants ***TRUE*** and ***FALSE*** of sort boolean. We call such signatures signatures ***with booleans.*** For example, Σ^0 is a signature with booleans.

Let Σ be a signature with booleans. The set of equations $\mathfrak{E}$ is consistent if and only if ***TRUE*** $\neq_{\mathfrak{E}}$ ***FALSE.*** That is, if and only if 8 admits at least one model whose carrier of sort boolean is the genuine truth-values set {true, false}.

This condition refines in a nice way the usual notion in non-sorted equational logic, which defines $\mathfrak{E}$ inconsistent if and only if x $=_{\mathfrak{E}}$ y (recall that all variables are implicitly universally quantified). As before, inconsistency is recursively enumerable if $\mathfrak{E}$ is recursive.

Examples: $\mathfrak{E}^0$ is consistent, because ***TRUE*** = ***FALSE*** is not valid in $\mathcal{A}^0$ (nor in $\mathcal{A}^1$ for that matter).

The equation x $+ S(S(y)) = S(x +$ (y $+$ S(0))) is valid in $\mathcal{M}(\mathfrak{E}^0)$. Here is a **proof:** $x+S(S(y)) =_{\mathfrak{E}^0} x+S(S(y+0)) =_{\mathfrak{E}^0} x+S(y+S(0)) =_{\mathfrak{E}^0} S(x+(y+S(0)))$. Similarly, $\mathcal{M}(\mathfrak{E}^0) \models 0 +$ S(0) $=$ S(0). More generally $\mathcal{M}(\mathfrak{E}^0) \models 0 + M = M$ for every M of the form S(S(. . .S(O). . .)). But $\mathcal{M}(\mathfrak{E}^0) \not\models 0 + x =$ x since this equation is not valid in $\mathcal{A}^1$. In the same way, the commutativity of $+$ is not valid in $\mathcal{M}(\mathfrak{E}^0)$, even though every ground instance of it is. We would like somehow to be able to restrict the notion of validity to the "standard model" $\mathcal{A}^0$. We consider this in the next section.

5. Initial Algebras and the Word Problem

The initial algebra $\mathfrak{J}(\Sigma, \mathfrak{E})$ or simply J(8) generated by a set of equations $\mathfrak{E}$ is defined as the quotient of $\mathfrak{G}$ by the congruence $=_{\mathfrak{E}}$ restricted to ground terms. We leave it to the reader to check that this is indeed a C-algebra. (It is an initial object in the category whose objects are the C-models of $\mathfrak{E}$ and the morphisms **the Σ-morphisms;** that is, there is a unique C-morphism from J(8) to any algebra of $\mathcal{M}(\mathfrak{E})$. See Goguen-Thatcher-Wagner (76) for details.)

Theorem. J(8) $\models M =$ N if and only if for every ground substitution $\sigma: \mathcal{V} \to \mathfrak{G}$, we have $\sigma(M) =_{\mathfrak{E}} u(N)$.

In other words, the validity problem over J(8) is the same as the validity problem in the G-variety of all its ground instances, and this captures the intuitive

notion of standard model mentioned above. However, there is unfortunately no simple proof theory associated with this semantic notion, unlike the case for $\mathcal{M}(\mathcal{E})$ above. The situation is analogous to that in first-order **Peano** arithmetic on the one hand and the standard model $\mathcal{N}$ on the other.

Note that when M and N are ground terms, that is, for the word problem in the initial algebra, equational reasoning is complete. In other cases, $\mathcal{I}(\mathcal{E}) \models M = N$ may be solved using induction **schemas,** for example on the structure of $\mathcal{G}$, but in general no induction schema will be strong enough to solve the validity problem in the initial algebra. The next section will give conditions under which this problem reduces to a problem of inconsistency.

Remarks: The set of equations valid in $\mathcal{I}(\mathcal{E})$ is called $Induce(Th(\mathcal{E}))$ in Burstall-Goguen (77). $\mathcal{G}$ is $\mathcal{I}(\emptyset)$ and $\mathcal{T}$ is $\mathcal{I}(\Sigma \cup \mathcal{V}, \emptyset)$ restricted as a C-algebra.

Example: $\mathcal{I}(\mathcal{E}^0)_{integer}$ is $\mathcal{N}$. However, $\mathcal{I}(\mathcal{E}^0)$ is not $\mathcal{A}^0$ since its boolean carrier is more complicated — for instance, $0 \neq 0 \neq_{\mathcal{E}^0} FALSE$., We can prove $\mathcal{I}(\mathcal{E}^0) \models 0 + x = x$ by induction on the structure of $\mathcal{G}^0_{integer}$, which corresponds here to standard integer induction. Similarly, $\mathcal{I}(\mathcal{E}^0) \models x + y = y + x$ may be proved by a double induction argument.

Let us now briefly indicate the connection between our definitions and the word problem over finitely presented algebras.

We assume given a signature Σ and a variety defined by a set of equations $\mathcal{E}$. A ***presentation*** over this variety consists of a set $\mathcal{C}$ of constants (the generators) and a set $\mathcal{R}$ of ground equations (the ***defining*** relations) over $\mathcal{T}(\Sigma \cup \mathcal{C})$.

The algebra ***presented*** by $\langle \mathcal{C}, \mathcal{R} \rangle$ is defined as $\mathcal{I}(\Sigma \cup \mathcal{C}, 6 \cup \mathcal{R})$. It is a $(\Sigma \cup \mathcal{C})$-algebra, and therefore also a C-algebra. The word problem for this presentation is as above — given a (ground) equation E over $\mathcal{T}(\Sigma \cup \mathcal{C})$, to determine whether or not $\mathcal{I}(\Sigma \cup \mathcal{C}, \mathcal{E} \cup \mathcal{R}) \models E$. The presentation is said to be finite if $\mathcal{C}$ and $\mathcal{R}$ are, recursive if $\mathcal{C}$ and $\mathcal{R}$ are recursively enumerable.

The ***uniform word problem*** for the variety defined by axioms 8 is, given a finite presentation $\langle \mathcal{C}, \mathcal{R} \rangle$ and an equation E over $\mathcal{T}(\Sigma \cup \mathcal{C})$, whether or not $\mathcal{I}(\Sigma \cup \mathcal{C}, \mathcal{E} \cup \mathcal{R}) \models E$.

6. Unification

The equational systems we are describing in this survey manipulate the term structure $\mathcal{T}(\Sigma \cup \mathcal{V})$ or, more generally, $\mathcal{T}(\Sigma \cup \mathcal{V}, \mathcal{E})$, defined as $\mathcal{I}(\Sigma \cup \mathcal{V}, \mathcal{E})$ restricted to a C-algebra.

The unification problem is the satisfiability problem in this algebra. More precisely, let M and N be two terms in $\mathcal{T}(\Sigma \cup \mathcal{V})$. We say that M and N are &unifiable if and only if there exists a substitution $\sigma: \mathcal{V} \rightarrow \mathcal{T}(\Sigma \cup \mathcal{V})$ such that

$\sigma(M) =_{\mathcal{E}} u(N)$. We write $\mathcal{U}_{\mathcal{E}}(M, N)$ for the set of such substitutions, which we call **$\mathcal{E}$-unifiers.**

For instance, when $\mathcal{E}$ consists of **Peano's** axioms, the unification problem is precisely Hilbert's tenth problem, and so is undecidable (Matiyasevich 70). But if we keep only addition, the unification problem over integers without multiplication is decidable since it can be expressed as a formula of Presburger arithmetic, a decidable theory (Presburger 29). It has been announced by **Szabó** (79) that the following set has an undecidable unification problem:

$$x \times (y + z) = x \times y + x \times z$$
$$(x + y) \times z = x \times z + y \times z$$
$$x + (y + z) = (x + y) + z.$$

We are generally interested in finding not only whether two terms are unifiable, but also the set of all their unifiers. Of course we are not interested in the unifiers that can be obtained from others by composition — we are only interested in a basis from which we can generate all the solutions to a unification problem. This can best be explained by an ordering on substitutions, which is itself the extension of the instantiation ordering on terms. This is the purpose of the following definitions.

The instantiation or ***subsumption*** preorder $\preceq_{\mathcal{E}}$ is defined over $\mathcal{T}(\Sigma \cup \mathcal{V})$ by:

$M \preceq_{\mathcal{E}}$ N if and only $\exists \sigma : \mathcal{V} \to \mathcal{T}(\Sigma \cup \mathcal{V})$ N $=_{\mathcal{E}} u(M)$.

We now want to extend $\preceq_{\mathcal{E}}$ to substitutions. First let us recall that $\mathcal{D}(\sigma) = \{x \in \mathcal{V} \mid \sigma(x) \neq x\}$. We also use the notation J(u) to denote $\bigcup_{x \in \mathcal{D}(\sigma)} \mathcal{V}(\sigma(x))$, that is, the set of variables introduced by σ. Let V be any finite subset of $\mathcal{V}$. We define a preorder $\preceq_{\mathcal{E}}(V)$ on substitutions by:

$\sigma \preceq_{\mathcal{E}} u'(V)$ if and only if $(\exists \rho)(\forall x \in V)\ u'(x) =_{\mathcal{E}} \rho(\sigma(x))$.

Now let M and N be two terms of the same sort, and $V = \mathcal{V}(M) \cup \mathcal{V}(N)$. Let W be any finite set of variables containing V, that is, $V \subseteq W \subset \mathcal{V}$. We say that a set of substitutions S is a ***complete*** set of G-unifiers of M and N away from W if **and** only if

1. (Vu $\in$ S) $\mathcal{D}(\sigma) \subseteq V$ and J(u) $\cap W = \emptyset$
2. $S \subseteq \mathcal{U}_{\mathcal{E}}(M, N)$
3. $(\forall \sigma \in \mathcal{U}_{\mathcal{E}}(M, N))\ (\exists \rho \in S)\ \rho \preceq_{\mathcal{E}} \sigma\ (V)$.

The first condition is technical, the second concerns soundness, and the third completeness. It can easily be shown that complete sets of unifiers always exist (take all B-unifiers of M and N verifying 1).

The set S is said to be ***minimal*** if and only if it satisfies the additional condition

4. $(\forall \sigma, \sigma' \in S)\, \sigma \neq \sigma' \supset \sigma \not\preceq_{\mathcal{E}} \sigma'\ (V)$.

Minimal complete sets of unifiers do not always exist. When they do, they are unique, up to the equivalence generated by $\preceq_{\mathcal{E}}$ *(V).*

Our definitions are consistent with the ones in Huet (76). Plotkin's definition of a complete set of unifiers, in **Plotkin (72),** is similar to our definition of a minimal CSU, modulo minor technical details. The necessity of considering a set W comes from the fact that terms M and N may come from a larger context, containing variables not in V, and we do not want to mix these with the "new" variables introduced by a unifier.

Examples:

1. When $\mathcal{E} = \emptyset$, it is well known that there exists a most general (that is, minimum) unifier. Algorithms to compute this unifier have been proposed by **Herbrand** in 1930 in his thesis, Robinson **(65), Baxter(73), Huet(76),** Martelli and Montanari (79). All these algorithms are non-linear; a linear algorithm is given in Paterson and Wegman (79). The current state of the art on implementations is discussed in Martelli and Montanari(79). The equivalence $\equiv$ associated with $\preceq_{\emptyset}$ is variable renaming, and the corresponding ordering on $\mathcal{T}(\Sigma \cup \mathcal{V})/\equiv$ completed with a maximum element defines a complete lattice. Unification corresponds to finding the least upper bound in this lattice. These issues are discussed in **Plotkin(70), Reynolds(70)** and Huet(76). Unification is one of the basic algorithms used in computational logic, and plays a central role in the inference rules of resolution (Robinson 65) and paramodulation (Robinson and Wos 69). We shall use it in the superposition algorithm in the next section.

2. Let $\mathcal{E}$ consist solely of the associativity axiom for some function symbol $\times$: $\mathcal{E} = \{(x \times y) \times z = x \times (y \times z)\}$. In this case there always exist minimal complete sets of unifiers, although the sets may be infinite. For instance with $M = x \times A$ and $N = A \times x$, with A some constant, $S = \{\{x \leftarrow A \times (A \times (\cdots \times (A \times A) \cdot \cdot))\}\} \cup \{\{x \leftarrow A\}\}$ is a minimal complete set of unifiers of M and N away from any W. An algorithm for generating such complete sets of unifiers is given in **Plotkin** (72). The unification problem reduces to the string equation problem, recently shown decidable by Makanin (79).

3. When $\mathcal{E}$ consists of the commutativity and associativity axioms for some function symbol $\times$: $\mathcal{E} = \{(x \times y) \times z = x \times (y \times z), x \times y = y \times x\}$ and $\Sigma = \{\times\}$, then the unification problem corresponds to solving equations in free **abelian** semigroups. There always exists a finite minimal complete set of unifiers. **Stickel(75)** gives an algorithm to generate this set. His algorithm is correct over

a more general problem: 8 may consist of commutative and associative laws for several function symbols, and Σ may contain other function symbols. But it remains an open problem to prove the termination of his algorithm when some of these other function symbols are of arity greater than 0.

4. Various other theories have been studied: commutativity (Siekmann 78); idempotence (Raulefs and Siekmann 78); wmmutativity, associativity and/or identity and/or idempotence (Stickel 76; Livesey and Siekmann 77). We shall see in section 12 a general method to generate a complete set of unifiers in a wide class of theories, which will include group theory. An abelian group theory unification algorithm is given by Lankford(79a). However, his algorithm does not always terminate when extra function symbols are allowed (Stickel, private communication).

Unification has been studied for higher-order languages. An w-order unification algorithm is given in Huet(75,76) and third-order unification is shown undecidable by Huet(73). Monadic second-order unification is decidable, as it is equivalent to associative unification, but it has recently been shown by Goldfarb(79) that polyadic second order unification is undecidable.

One-way unification (when unification is permitted in only one of the terms) is called matching. We say σ is an $\mathcal{E}$-*match* of M and N if and only if $\sigma(M) =_{\mathcal{E}} N$. A complete set of $\mathcal{E}$-*matches of* M and N away *from* W is defined as for unification, except that in 1 we replace $\mathcal{D}(\sigma) \subseteq V$ by $\mathcal{D}(\sigma) \subseteq \mathcal{V}(M) - \mathcal{V}(N)$. Such a set always exists when $\mathcal{V}(M) \cap Y(N) = \emptyset$.

7. Term Rewriting Systems

We are now ready to develop one of the main paradigms of computing with equations — using them as rewrite rules over terms. This paradigm is the basis for, on one hand, decision procedures based on canonical forms and, on the other hand, the construction of abstract interpreters for directed equations considered as a programming language.

A *term* rewriting system (over Σ) is a set of directed equations $\mathfrak{R} = \{\lambda_i \to \rho_i \mid i \in \mathrm{I}\}$ such that, for all $\lambda \to \rho$ in $\mathfrak{R}$, $\mathcal{V}(\rho) \subseteq \mathcal{V}(\lambda)$.

The reduction relation $\to_{\mathfrak{R}}$ associated with $\mathfrak{R}$ is the finest relation over $\mathcal{T}$ containing $\mathfrak{R}$ and closed by substitution and replacement. That is,

$$M \to_{\mathfrak{R}} \mathrm{N} \Rightarrow \sigma(M) \to_{\mathfrak{R}} \mathrm{u(N)} \tag{1}$$

$$M \to_{\mathfrak{R}} N \Rightarrow P[u \leftarrow M] \to_{\mathfrak{R}} P[u \leftarrow N]. \tag{2}$$

Equivalently, $\rightarrow_{\mathfrak{R}}$ is the finest relation containing all pairs $\sigma(\lambda), \sigma(\rho)$ such that $\lambda \rightarrow \rho \in \mathfrak{R}$ and closed under replacement (2).

Example: With $\mathfrak{R} = \{A \rightarrow B, F(B,x) \rightarrow G(x,x)\}$ *we* have $F(A,A) \rightarrow_{\mathfrak{R}} F(B,A) \rightarrow_{\mathfrak{R}} G(A,A)$.

From now on, we shall use $\rightarrow$ for $\rightarrow_{\mathfrak{R}}$. We use the standard notation $\rightarrow^+$ for the transitive closure of $\rightarrow$, $\rightarrow^*$ for its transitive-reflexive closure, and $\leftrightarrow$ for its symmetric closure. Note that $\leftrightarrow^*$ is the same as the $\mathfrak{R}$ equality $=_{\mathfrak{R}}$, when $\mathfrak{R}$ is considered a set of equations. Conversely, any set of equations $\mathfrak{E}$ can be transformed into a term rewriting system $\mathfrak{R}$ over an extended signature Σ' in such a way that $\leftrightarrow^*$ is a conservative extension of $=_{\mathfrak{E}}$. (Then, for all M, N in $\mathcal{T}(\Sigma \cup \mathcal{V})_s$, $M =_{\mathfrak{E}}$ N if and only if $M \leftrightarrow^*$ N.) The construction is as follows: for every $M =$ N in $\mathfrak{E}$, choose nondeterministically one of the following:

1. if $\mathcal{V}(M) \subseteq \mathcal{V}(N)$, put N $\rightarrow M$ in $\mathfrak{R}$
2. if Y(N) $\subseteq T(M)$, put $M \rightarrow N$ in $\mathfrak{R}$
3. with $\{x_1, \ldots, x_n\} = \mathcal{V}(M) \cap V(N)$, introduce in Σ' a new operator H of the appropriate type, and put in $\mathfrak{R}$ the two rules $M \rightarrow P$ and N $\rightarrow P$ with $P = H(x_1, \ldots, x_n)$. (Note that this third possibility may force certain sorts to be strict in Σ' when they were not in Σ; it may then be necessary to add extra constants to Σ' for it to be sensible. Also certain C-algebras which were models may not be extendible as Σ'-algebras because the corresponding carriers are empty.)

The fundamental difference between equations and term rewriting rules is that equations denote equality (which is symmetric) whereas term rewriting systems treat equations directionally, as one-way replacements. Further, the only substitutions required for term rewriting rules are the ones found by pattern matching. The completeness of using rewrite rules to make deductions equationally is expressed by the following Church-Rosser property:

$\mathfrak{R}$ is Church-Rosser if and only if, for all M and N, $M =_{\mathfrak{R}}$ N if and only if there exists a P such that $M \rightarrow^* P$ and N $\rightarrow^*$ P.

An equivalent characterization is "confluence". $\mathfrak{R}$ is confluent if and only if for all $M, N,$ and P, $P \rightarrow^* M$ and $P \rightarrow^*$ N implies there is some Q such that $M \rightarrow^*$ Q and N $\rightarrow^*$ Q.

We say that M is ***irreducible*** or ***in normal*** form (relative to $\mathfrak{R}$) if and only if there is no N such that $M \rightarrow$ N; that is, no subterm of M is an instance of some lefthand side of a rule in $\mathfrak{R}$. We say that N is a $\mathfrak{R}$-*normal* form of M if and only if $M \rightarrow^*$ N and N is a normal form relative to $\mathfrak{R}$.

When $\mathfrak{R}$ is Church-Rosser the normal form of a term is unique, when it exists. A sufficient condition for the existence of such a canonical form is the termination of all rewritings:

$\mathfrak{R}$ is **nœtherian** or finitely terminating if and only if for no M is there an infinite chain of reductions issuing from $M: M = M_1 \rightarrow M_2 \rightarrow \ldots$.

Let $M\downarrow$ be any normal form obtained from M by an arbitrary sequence of reductions.

When $\mathfrak{R}$ is a finite set of rules which is confluent and **nœtherian,** the equational theory $=_{\mathfrak{R}}$ is decidable, since now $M =_{\mathfrak{R}} N$ if and only if $M\downarrow = N\downarrow$. Actually, confluent and **nœtherian** rewrite rules provide a rather general method for solving the validity problem in decidable varieties.

The property of confluence is undecidable for arbitrary term rewriting systems, since a confluence test **could** be used to decide the equivalence, for instance, of recursive program **schemas.** The decidability of confluence for ground term rewriting systems is open. We shall now show that confluence is decidable for **nœtherian** systems.

The first step in the construction is showing that, for ncetherian systems, confluence is equivalent to "local wnfiuence". $\mathfrak{R}$ is locally ***confluent*** if and only if for all M, N, and P, $P \rightarrow M$ and $P \rightarrow$ N implies there is some Q such that $M \rightarrow^*$ Q and N $\rightarrow^*$ Q.

Newman Theorem. Let $\mathfrak{R}$ be a **nœtherian** term rewriting system. $\mathfrak{R}$ is confluent if and only if it is locally confluent.

This theorem was first proved by Newman (42); a simple **proof** appears in Huet (77). **V**arious equivalent or related "diamond lemmas" have been proposed, for instance by Hindley (69); Knuth and Bendix (70); Nivat (70); **Aho, Sethi** and Ullman (72); and **Lankford** (75).

The second step consists in showing that local confluence of (finite) term rewriting systems is decidable. We show this now.

Superposition Algorithm.

Let $\lambda \rightarrow \rho$ and $\lambda' \rightarrow \rho'$ be two rules in $\mathfrak{R}$. We assume we have renamed variables appropriately, so that λ and λ' share no variables. Assume u is a **non-**variable occurrence in λ such that λ/u and λ' are unifiable, with minimal unifier σ (we assume of course that λ/u and λ' are of the same sort). We then say that the pair $\langle \sigma(\lambda[u \leftarrow \rho']), \sigma(\rho) \rangle$ of terms is critical in $\mathfrak{R}$.

Example: Consider $F(x, G(x, H(y))) \rightarrow K(x,$ y) and $G(A, z) \rightarrow L(z)$. We can superpose the first rule at occurrence 2 with the second one using the minimal unifier $\{x \leftarrow A, z \leftarrow H(y)\}$, obtaining the critical pair $\langle F(A, L(H(y))), K(A, y) \rangle$.

If $\mathfrak{R}$ is finite, there are only finitely many such critical pairs. They can be effectively wmputed using the standard unification algorithm.

Knuth-Bendix Theorem. $\mathfrak{R}$ is locally confluent if and only if $P\downarrow = Q\downarrow$ for every critical pair (P, Q) of $\mathfrak{R}$.

The original idea for this theorem is due to Knuth and Bendix (70), who combined it with Newman's theorem (see also Lankford (75)). The version of the theorem given above appears in Huet (77) and does not require termination.

Combining the Knuth-Bendix theorem and Newman's theorem gives us a decision procedure for the confluence of ncetherian term rewriting systems with a finite number of rules. When such a system $\mathfrak{R}$ satisfies the critical pair condition, it defines a canonical form for the corresponding equational theory $=_{\mathfrak{R}}$. We then say that $\mathfrak{R}$ is a complete or canonical term rewriting system.

There exist other sufficient criteria for the confluence of term rewriting systems that do not assume that every sequence of reductions terminates. Most of them rely on a sufficient condition for confluence: "strong confluence". $\mathfrak{R}$ is *strongly confluent* if and only if for all M, N, and P, $P \rightarrow M$ and $P \rightarrow$ N implies there is some Q such that $M \rightarrow^{\epsilon}$ Q and N $\rightarrow^{\epsilon}$ Q, where $\rightarrow^{\epsilon}$ denotes the reflexive closure of $\rightarrow$.

Let us call term M *linear* if and only if every variable in M occurs only once. When $\mathfrak{R}$ is such that, for every $\lambda \rightarrow \rho$ in $\mathfrak{R}$, λ and ρ are linear, then $\mathfrak{R}$ is strongly confluent if for every critical pair $(P,$ Q) of $\mathfrak{R}$, there exists an R such that $P \rightarrow^{\epsilon} R$ and Q $\rightarrow^{\epsilon} R$ (Huet 77). However the condition of linearity of ρ is not very natural. A more useful result requires only linearity of left hand sides of rules. Let us denote by $\#\!\!\rightarrow$ the parallel reduction by rules of $\mathfrak{R}$ at disjoint occurrences.

Theorem. (Huet 77) If, for every $\lambda \rightarrow \rho$, λ is linear, and for every critical pair $(P,$ Q) we have $P \#\!\!\rightarrow$ Q, then the relation $\#\!\!\rightarrow$ is strongly confluent; and therefore $\mathfrak{R}$ is confluent.

This theorem extends the main theorem in (Rosen 73) which applies only to ground systems. It shows in particular that left-linear term rewriting systems with no critical pairs are confluent. Such recursive definitions are therefore determinate. They extend both the usual recursive schemes and the primitive recursive-like definitions.

8. Termination

In order to simplify the discussions and notation, we assume fmm now on that we are dealing with only one sort. We assume that Σ is finite, and that $\mathfrak{G} \neq \emptyset$.

The main problem with the Knuth-Bendix confluence test is the proof of termination. Recall that $\mathfrak{R}$ is ncetherian if and only if there is no term M such that there exists an infinite sequence $M = M_1 \rightarrow M_2 \rightarrow \cdots$. Proving $\mathfrak{R}$ ncetherian

is crucial, since without it local confluence does not tell us anything. For instance, the system $\mathfrak{R} = \{A \to B, A \to C, B \to A, B \to D\}$ is locally wnfluent, even though A has two distinct normal forms C and D.

It is undecidable whether an arbitrary term rewriting system is ncetherian. See Huet and **Lankford** (78)where it is shown however that this problem is decidable for gmund systems.

We first remark that $\mathfrak{R}$ is **nœtherian** if and only if there is no ground term M having an infinite sequence of reductions (recall that we are assuming $\mathfrak{G}$ is nonempty). That is, $\mathfrak{R}$ is ncetherian if and only if the relation $\to_{\mathfrak{R}}^{+}$ defines a **well-founded** partial ordering on $\mathfrak{G}$. This suggests the following general method for **proving $\mathfrak{R}$ nœtherian.**

Let $\succ$ be any partial ordering relation (that is, transitive and irreflexive) on $\mathfrak{G}$ such that:

(1) $\succ$ is well-founded; that is, there is no infinite sequence of ground terms $M_1 \succ M_2 \succ \cdots$

(2) $M \to N \Rightarrow M \succ N$

Then (obviously) $\mathfrak{R}$ is ncetherian. Conversely, if $\mathfrak{R}$ is ncetherian, then the relation $\to^{+}$ satisfies (1) and (2).

Various refinements of this general scheme have been proposed. Let us present a few.

A. Well-founded Mapping Method

Let $\mathfrak{D}$ be any set given with a well-founded partial ordering $\succ_{\mathfrak{D}}$. The method consists in exhibiting a mapping $\varphi: \mathfrak{G} \to \mathfrak{D}$, and defining the relation $\succ$ by:

$M \succ N$ if and only if $\varphi(M) \succ_{\mathfrak{D}} \varphi(N)$.

Pmperty (1) is obviously satisfied. The method then consists in showing (2). Conversely, if $\mathfrak{R}$ is **nœtherian,** consider $\mathfrak{G}$ for $\mathfrak{D}$, $\to^{+}$ for $\succ_{\mathfrak{D}}$, and the identity for φ. Also, if $\to$ is finitely branching (this is the case for instance whenever $\mathfrak{R}$ is finite), we may always show $\mathfrak{R}$ **nœtherian** by the method above, with $\mathfrak{D}$ being the set of natural numbers $\mathcal{N}$ with their usual ordering $>$, by defining $\varphi(M)$ to be the length of the longest sequence of reductions issuing from M (hint: use **König's** lemma).

However, the real crux of the matter comes with proving (2), and this is not very convenient, since it quantifies over all possible reductions $M \to N$. An important refinement consists in remarking that (2) is implied by (3) and (4) below:

(3) $M \succ N \Rightarrow F(\vec{P}, M, \vec{Q}) \succ F(\vec{P}, N, \vec{Q})$ for every operator F in Σ and sequences of ground terms $\vec{P}$ and $\vec{Q}$.

(4) $\sigma(\lambda) \succ \sigma(\rho)$ for every ground substitution σ and rule $\lambda \to \rho$ in $\mathfrak{R}$.

Condition (3) means that the relation $\succ$ is wmpatible with the C-algebra structure. This suggests refining the mapping method above into the following.

B. **Increasing Interpretation** Method

We take $\mathfrak{D}$ to be a C-algebra, and φ to be the unique C-homomorphism from $\mathfrak{G}$ to $\mathfrak{D}$. Condition (3) is implied by the corresponding property in algebra $\mathfrak{D}$; that is, every operator of Σ corresponds in $\mathfrak{D}$ to a mapping strictly increasing in every argument:

(5) $(\forall x, y \in \mathfrak{D})(\forall \vec{u}, \vec{v} \in \vec{\mathfrak{D}})\, x \succ_{\mathfrak{D}} y \Rightarrow F(\vec{u}, x, \vec{v}) \succ_{\mathfrak{D}} F(\vec{u}, y, \vec{v})$.

Given such an increasing interpretation of Σ, all we have to show is (4), or its stronger analogue in $\mathfrak{D}$; that is:

(6) $(\forall \lambda \to \rho \in \mathfrak{R})\, \mathfrak{D} \models \lambda \succ_{\mathfrak{D}} \rho$.

Actually, (4) and (6) suggest rather different methods of proof: (4) suggests using structural induction in $\mathfrak{G}$, whereas we must resort to **nœtherian** induction in $\mathfrak{D}$ to **prove** the stmnger (6).

The increasing interpretation method has been proposed by Manna and Ness (70), and by **Lankford** (75) in the special case of polynomial interpretations over the integers.

Example: The ten **group** reductions shown in the Appendix can be shown to terminate using the following increasing interpretation over the set of integers greater than 1:

$$x + y = x \times (1 + 2 \times y)$$
$$I(x) = x^2$$
$$0 = 2.$$

However, the proof of (6) is not straightforward, since it involves showing for instance $(\forall x, y \in \mathcal{N})\, x > 1$ and $y > 1$ implies $x^2(1+2y)^2 > y^2(1+2x^2)$. Actually sentences of the form of (6) are undecidable in general, for polynomial integer interpretations. This precludes practical use of this method, even assuming the human user guesses the right interpretation.

Of course, polynomial interpretations do not suffice in general, since they give a polynomial upper bound on the complexity of the computations by $\mathfrak{R}$, interpreted as a program computing over integers, whereas arbitrary recursive functions can be defined by term rewriting systems. This argument extends to primitive recursive complexity measures, as remarked by M. **Stickel.**

The increasing interpretation is completely general, however, since if $\mathfrak{R}$ is ncetherian then $\mathfrak{G}$ itself as a C-algebra defines an increasing interpretation, taking

$\rightarrow^+$ for $\succ_{\mathfrak{G}}$. But here we cannot restrict ourselves to total orderings, since for instance A and B must be unrelated in any increasing interpretation which shows the termination of $\mathfrak{R} = \{F(A) \rightarrow F(B), G(B) \rightarrow G(A)\}$. Therefore, even ordinal numbers are not general enough here!

Next let us show how condition (1) can be refined. We define in $\mathfrak{G}$ the ***homeomorphic*** embedding relation $\trianglelefteq$ by induction as follows:

$F(M_1, \ldots, M_n) \trianglelefteq N$ if and only if there exists in N a **subterm** $F(N_1, \ldots, N_n)$ and a permutation π of $(1, 2, \ldots, n)$ such that $M_i \trianglelefteq N_{\pi_i}$ for every i, $1 \leq i \leq n$.

Kruskal's Theorem. For every infinite sequence $M_1, M_2, \ldots$ of ground terms in $\mathfrak{G}$, there exist i and j, with $i < j$, such that $M_i \trianglelefteq M_j$.

See **Kruskal(60)** and Nash-Williams(63) for the proof of more general theorems (which do not require Σ to be finite). The theorem suggests replacing (1) by the (stronger) condition:

(7) $M \trianglelefteq N$ implies $\neg(M \succ N)$.

Finally, Dershowitz **(79a,b)** shows that (7) follows **from** (3) and:

(8) $F(\vec{P}, M, \vec{Q}) \succ M$.

All this justifies the following method.

C. Simplification Ordering Method

Let $\succ$ be any partial ordering on $\mathfrak{G}$ satisfying (3) and (8). (We then say that $\succ$ is a simplification ordering.) Then $\mathfrak{R}$ is **nœtherian** if (4) is satisfied.

The method was first proposed by Dershowitz **(79a,** 79b). An important **application** is the following.

First recall that a partial ordering $\succ$ on a set Y may be extended to a partial ordering $\gg$ on the set of multisets of $\mathfrak{Y}$, that is, of mappings $Y \rightarrow \mathcal{N}$, as follows: $S \gg S'$ if and only if $S \neq S' \;\&\; (\forall x \in \mathfrak{Y})\,[S'(x) > S(x) \Rightarrow (\exists y \succ x)\, S(y) > S'(y)]$.

Furthermore, the set of finite multisets of $\mathfrak{Y}$ is well founded by $\gg$ if and only if $\mathfrak{Y}$ is well-founded by $\succ$ (see Dershowitz and Manna (79)). The **multiset** ordering generalizes the lexicographic ordering, obtained when $\succ$ is a total ordering.

Next we define the permutation equivalence $\sim$ on $\mathfrak{G}$ by: $M \sim N$ if and only if $M = F(M_1, \ldots, M_n)$, $N = F(N_1, \ldots, N_n)$, for some F in Σ, and there exists a permutation π of $1, 2, \ldots, n$ such that $M_i \sim N_{\pi_i}$, $1 \leq i \leq n$.

We denote the permutation class $\tilde{M}$ of $M = F(M_1, \ldots, M_n)$ by $F\{\tilde{M}_1, \ldots, \tilde{M}_n\}$. Let $\succ_\Sigma$ be any partial ordering on the operators of Σ. We define the recursive path ordering $\succ$ on $\mathfrak{G}/\sim$ recursively as follows: $\tilde{M} = F\{\tilde{M}_1, \ldots, \tilde{M}_n\} \succ \tilde{N} = G\{\tilde{N}_1, \ldots, \tilde{N}_p\}$ if and only if:

(a) either $F = G$ and $\{\tilde{M}_1, \ldots, \tilde{M}_n\} \gg \{\tilde{N}_1, \ldots, \tilde{N}_p\}$

(b) or $F \succ_\Sigma G$ and $(\forall i \leq p)\, \tilde{M} \succ \tilde{N}_i$

(c) or (otherwise) $(\exists j \leq n)\, \tilde{M}_j \succeq \tilde{N}$, where $\tilde{M} \succeq \tilde{N}$ if and only if $\tilde{M} \succ \tilde{N}$ or $\tilde{M} := \tilde{N}$.

The recursive path ordering generalizes the orderings defined in Plaisted (78), Itturiaga (67), and Dershowitz and Manna (79). Now defining $M \succ N$ as $\tilde{M} \succ \tilde{N}$, we get:

Theorem. $\succ$ is a simplification ordering.

For a proof, see Dershowitz (79b) for a slightly more general formulation permitting infinite signatures that may contain varyadic operations.

This theorem gives a fairly general method for proving termination of $\mathfrak{R}$:

D. Recursive Path Ordering Method

Guess some partial ordering $\succ_\Sigma$ and show (4) using the corresponding simplification ordering.

Of course, this method does not give us yet a completely mechanical way of checking (4) because of the quantification on every ground substitution. What we need here is to be able to define recursively an ordering $\succ$ on ground terms that "lifts" to an order $\succ$ over general terms, in the sense that M $\succ$ N implies $\sigma(M) \succ$ a(N), for every ground substitution σ.

We are able to do this to a limited extent. First, the clause (c) in the definition above does not depend on the M_k's, for $k \neq$ j, and we may therefore permit non-ground terms in these positions. Secondly, properties (3) and (8), which hold here because of the theorem above, quantify over all P's and Q's and we may again permit non-ground terms there, extending the recursive path ordering to non-ground terms. Actually, (3) is directly implied by the algorithm above, and (8) can be incorporated easily in the algorithm as follows. We construct the recursive path ordering $\succ$ on possibly non-ground terms, that is on $\mathcal{T}/\sim$. (We extend $\sim$ to include z-z for every variable x.) We keep the three clauses (a), (b), and (c) above and we add one more clause:

(d) $\tilde{M} \succ$ x if and only if $M \notin \mathcal{V}$ and x $\in$ V(M).

It should be obvious to the reader that $M \succ$ N with the new definition implies that $\sigma(M) \succ \sigma(N)$ for every ground substitution σ with the old one (for (d), use a simple induction on M, and (8)). We call the new ordering the generalized recursive path *ordering*. All that precedes justifies the following:

E. **Generalized Recursive** Path Ordering Method

Let $\succ_\Sigma$ be some partial ordering on Σ, $\succ$ the generalized path ordering defined above. If for all $\lambda \to \rho$ in $\mathfrak{R}$ we have $\lambda \succ \rho$, then $\mathfrak{R}$ is **nœtherian.**

Example: (Disjunctive normal form)

$$\text{Let } \mathfrak{R} = \left\{ \begin{array}{l} \neg(\neg(x)) \to x, \\ \neg(x \vee y) \to \neg(x) \wedge \neg(y), \\ \neg(x \wedge Y) \to \neg(x) \vee \neg(y), \\ \mathbf{x} \wedge (\mathbf{y} \vee z) \to (x \wedge y) \vee (x \wedge z), \\ (\mathbf{y} \vee z) \wedge x \to (y \wedge x) \vee (z \wedge x). \end{array} \right|$$

We take $\neg \succ_\Sigma \wedge \succ_\Sigma \vee$. We leave it to the reader to show that for every $\lambda \to \rho$ in $\mathfrak{R}$ we have $\lambda \succ \rho$, establishing the termination of $\mathfrak{R}$. Note that what we have done here is to automate completely, inside the generalized recursive path algorithm, the proof exhibited by Dershowitz (79b).

Other definitions of recursive orderings on general terms that lift simplification orderings (like the surface ordering) are given in **Plaisted(78b).**

We conclude our discussion on simplification orderings by remarking that it has a semantic analogue, refining the increasing interpretation method, as follows.

F. **Homeomorphic** Interpretation Method

The method consists in choosing a C-algebra $\mathfrak{D}$ given with a partial ordering $\succ_\mathfrak{D}$ verifying the increasing condition (5) and the extra condition, for every operator F in Σ:

(9) $(\forall x \in \mathfrak{D})(\forall \vec{u}, \vec{v} \in \vec{\mathfrak{D}})\ F(\vec{u}, x, \vec{v}) \succ_\mathfrak{D} x.$

Now $\mathfrak{R}$ is **nœtherian** if (6) is satisfied. The justification of the method should be clear: taking φ to be the unique C-homomorphism from $\mathfrak{G}$ to $\mathfrak{D}$, and ordering $\mathfrak{G}$ by $M \succ$ N if and only if $p(M) \succ_\mathfrak{D} \varphi(N)$, we get (2) by (5) and (6). But now (1) is a consequence of just (7), which follows from (6) and (9). That is, we do not need here $\succ_\mathfrak{D}$ to define a well-founded ordering on $\mathfrak{D}$.

This method may well provide a new practical approach to proofs of termination of term rewriting systems, taking algebras for which condition (6) is decidable. For instance, **Dershowitz(79a)** proposes to use this method, taking for $\mathfrak{D}$ the reals with polynomial operators. He remarks that conditions (5), (9) and (6) are indeed decidable in this setting, using **Tarski's** decision method (**Tarski** 51). Furthermore, the existence of such polynomials of a bounded degree that decide the termination of $\mathfrak{R}$ is decidable. The practicality of this method depends of

course on the availability of implementations of such decision procedures (Cohen 69, Collins 75). The use of interpretations over the reals was first proposed by **Lankford** (75).

We conclude this section by a generalization of simplification orderings to pre-orderings, that is, transitive and reflexive relations. Let $\succeq$ be a pre-ordering on $\mathfrak{G}$ satisfying the simplification conditions:

(3') $M \succeq N \Rightarrow F(\vec{P}, M, \vec{Q}) \succeq F(\vec{P}, N, \vec{Q})$
(8') $F(\vec{P}, M, \vec{Q}) \succeq M$.

We then say that $\succeq$ is a simplification pre-ordering. Defining $M \succ N$ by $M \succeq N$ and not $N \succeq M$, we get:

G. Simplification **pre-ordering** method

Let $\succeq$ be a simplification pre-ordering. If (4) is satisfied, then $\mathfrak{R}$ is **nœtherian.**

The correctness of this method is shown in Dershowitz (79b) where it is stated that it may be used to prove the correctness of the recursive ordering described in Knuth and Bendix (70).

9. Compiling Canonical Forms

The Knuth-Bendix theorem of section 7 gives us conditions under which an equational theory admits a canonical form, as follows.

Let 8 be the (finite) set of equations given as basis (axioms) for the theory. If there exists a canonical term rewriting system $\mathfrak{R}$ such that we have (denoting by $\mathfrak{R}(M)$ the unique normal form of M obtained by applying to it an arbitrary number of reductions from $\mathfrak{R}$):

(1) $(\forall M = N \in \mathfrak{E})\, \mathfrak{R}(M) = \mathfrak{R}(N)$, and
(2) $(\forall \lambda \to \rho \in \mathfrak{R})\, \lambda =_{\mathfrak{E}} \rho$,

then $\mathfrak{R}(M)$ is a canonical form of M for the theory $=_{\mathfrak{E}}$, in the sense that

$(\forall M, N \in \mathfrak{T}(\Sigma \cup \mathcal{V}))\, M =_{\mathfrak{E}} N$ if and only if $\mathfrak{R}(M) = \mathfrak{R}(N)$.

Note that this gives us a decision procedure for the validity problem in $\mathfrak{M}(\mathfrak{E})$, and therefore, as a corollary, a decision procedure for the word problem in $\mathfrak{I}(\mathfrak{E})$.

We saw in section 7 how to check local confluence with critical pairs, and in section 8 how to check $\mathfrak{R}$ **nœtherian.** Let us now give an algorithm that attempts to generate $\mathfrak{R}$ satisfying (1) and (2) for a given $\mathfrak{E}$.

Completion Algorithm.

$\mathcal{E}_i$ is a set of equations, $\mathcal{R}_i$ a term rewriting system, $i \in \mathcal{N}$. Initially let $\mathcal{E}_0 = \mathcal{E}$, $\mathcal{R}_0 = \emptyset$ and $i = 0$.

1. If $\mathcal{E}_i = \emptyset$, stop with answer $\mathcal{R} = \mathcal{R}_i$. Otherwise, select $M = \mathrm{N}$ in $\mathcal{E}_i$, and let $M\downarrow$ and $N\downarrow$ be normal forms for M and N respectively, using the current system $\mathcal{R}_i$. If $M\downarrow = N\downarrow$ then let $\mathcal{E}_{i+1} = \mathcal{E}_i - \{M = \mathrm{N}\}$, $\mathcal{R}_{i+1} = \mathcal{R}_i$, increment i by 1 and go to 1. Otherwise go to 2.

2. Choose non-deterministically one of the following:

(a) if $\mathcal{V}(M\downarrow) \subset \mathcal{V}(N\downarrow)$ then let $\lambda = N\downarrow, \rho = M\downarrow$.

(b) if $\mathcal{V}(N\downarrow) \subset \mathcal{V}(M\downarrow)$ then let $\lambda = M\downarrow, \rho = N\downarrow$.

If neither (a) nor (b) applies, stop with failure.

Otherwise, let $\mathcal{R}'_i = \{\mathrm{h}' \to \rho' \in \mathcal{R}_i \mid \lambda'$ or ρ' contains an instance of λ as subterm$\}$. Then let $\mathcal{R}''_i = \mathcal{R}_i - \mathcal{R}'_i \cup \{\lambda \to \rho\}$ and go to 3.

3. If $\mathcal{R}''_i$ is not **nœtherian,** stop with failure. Otherwise, let $\mathcal{R}_{i+1} = \mathcal{R}''_i$, and $\mathcal{E}_{i+1} = (\mathcal{E}_i - \{M = N\}) \cup \{\lambda' = \rho' \mid \lambda' \to \rho' \in \mathcal{R}'_i\} \cup \{P = Q \mid \langle P, Q\rangle$ is a critical pair of $\mathcal{R}_{i+1}\}$. Increment i by 1 and go to 1.

The completion algorithm is due to Knuth and Bendix (70). Its correctness follows fmm the following facts: (i) At every iteration i, $=_{\mathcal{R}_i \cup \mathcal{E}_i}$ equals $=_{\mathcal{E}}$. (ii) Every $\mathcal{R}_i$ is **nœtherian,** and every $\lambda \to \rho$ in $\mathcal{R}_i$ is such that ρ is a normal form for $\mathcal{R}_i$ and λ is a normal form for $\mathcal{R}_i - \{\lambda \to \mathrm{p}\}$. (iii) If the algorithm stops with success, $\mathcal{R}$ is locally confluent, and therefore (1) and (2) above are satisfied.

Note that the algorithm is non-deterministic because of the various choices. It may stop with success, stop with failure, or loop forever. But we were careful in its formulation, in that the only possibility of non-termination is by the global looping of the outer iteration, and this can be prevented by setting a bound on i.

Various optimizations of the algorithm are possible. For instance, we need to recompute only the critical pairs of $\mathcal{R}_{i+1}$ which could not be obtained as critical pairs of $\mathcal{R}_i$. We omit the proof for lack of space.

There are various cases of failure. When neither (a) nor (b) applies at step 2, this suggests starting the process over again with an augmented signature Σ' and an augmented set of equations $\mathcal{E}'$ as follows. Let $\{\mathrm{xl}, \ldots, x_n\} = \mathcal{V}(M\downarrow) \cap \mathcal{V}(N\downarrow)$. Augment Σ with operator H of type $\tau(x_1) \times \ldots \times \tau(x_n) \to \tau(M)$, and augment $\mathcal{E}$ with the equation $H(x_1, \ldots, x_n) = M\downarrow$, for instance. As indicated above, some extra constants may be needed for the new signature to be sensible, in the **multi-**sorted case. But in any case, the new equational theory is a conservative extension of the old one; that is, for any terms P and Q over Σ, $P =_{\mathcal{E}'}$ Q if and only if $P =_{\mathcal{E}}$ Q. Intuitively, the new equations are just definitions of the new symbols. These definitions are acceptable, since $M\downarrow =_{\mathcal{E}} N\downarrow$ implies that in any C-model of $\mathcal{E}$ the value of $M\downarrow$ does not depend on the assignments to variables other than the

x_i's. When running the completion algorithm with $\mathfrak{E}'$ the situation will be different, since the new equation will give rise to a new reduction $M\downarrow \rightarrow H(x_1, \ldots, x_n)$ and when $M =$ N is selected case (a) will apply. The new system may indeed converge, as shown in some examples in Knuth and Bendix (70). This modification can be taken into account in the algorithm, by replacing the "stop with failure" at step 2 by the following:

Let $\mathfrak{E}_{i+1} = (\mathfrak{E}_i - \{M =$ N$\}) \cup \{M\downarrow = H(x_1, \ldots p\ x_n), N\downarrow = H(x_1, \ldots, x_n)\}$, $\mathfrak{R}_{i+1} = \mathfrak{R}_i$, increment i by 1 and go to 1.

The other case of failure comes from step 3, if we recognize $\mathfrak{R}$ as non-ncetherian. This may actually come from three different causes:

1. We choose the wrong orientation of some rule at step 2; that is, some other choices may lead to a successfully terminating computation.

2. The nœtherian test used at step 3 is not general enough; that is, $\mathfrak{R}''_i$ is indeed terminating but our test does not show it. Some more powerful test may be needed for successful completion.

3. Every choice at step 2 leads to a non-nœtherian $\mathfrak{R}''_i$. This "unrecoverable" failure occurs for instance whenever a symmetric axiom such as commutativity is encountered. This is obviously the main pitfall of the whole method: it does not apply to theories with such permutative axioms. (However, see Section 14.)

The appendix contains a complete run of an implementation (by Jean-Marie Hullot) of the completion algorithm for the group axioms. The Knuth-Bendix ordering is used there for checking termination.

10. Decidability and Complexity of Word Problems

Word problems have been extensively studied in logic; some sources of results on decidability and undecidability are Tarski (68), McNulty (76) and Evans (78).

For example, when $\mathfrak{E}$ is a standard set of axioms for group theory: $Group = \{x+0 = x, x + (-x) = 0, (x+y)+z = x+(y+z)\}$, then certain word problems (and therefore the uniform word problem) are unsolvable. The word problem in the free group with w generators, that is, $\mathfrak{I}(\Sigma \cup \mathcal{V}, Group)$, is solvable. Dehn's algorithm (Dehn 11, Greenlinger 60) gives an algorithm for the word problem in a family of presentations of groups.

The computational complexity of several decidable word problems has been studied. The uniform word problem for abelian semigroups is solvable (see Malcev (58) and Emilichev (58)); Cardoza, Lipton and Meyer (76) show that the pmblem is complete in exponential space under log-space transformability. See also Lipton and Zalcstein (75), and Meyer and Stockmeyer (73) for other results.

An extensively studied case is the uniform word problem for the case $6 = \emptyset$, that is, in the variety of all C-algebras. The problem is thus determining whether some gmund equation follows from a given set of ground equations by substitutivity of equality, and so is also the decision problem for the quantifier-free theory of equality with uninterpreted function symbols. An example is determining whether f(a) = a is a consequence of $\{f(f(f(a))) = a, f(f(f(f(f(a))))) = a\}$. **Ackermann** (54) showed that the problem was decidable, but did not give a practical algorithm.

The problem reduces to the problem of constructing the "congruence closure" of a relation on a graph. Let $G = (V, E)$ be a directed graph with **labelled** vertices, possibly with multiple edges. For a vertex v, let $\lambda(v)$ denote its label and $\delta(v)$ its outdegree, that is, the number of edges leaving v. The edges leaving a vertex are ordered. For $1 \leq i \leq$ S(v), let $v[i]$ denote the ith successor of v, that is, the vertex to which the ith edge of v points. A vertex u is a predecessor of v if $v = u[i]$ for some i. Since multiple edges are allowed, possibly $v[i] = v[j]$ for $i \neq$ j. Let n be the number of vertices of G and m the number of edges of G. We assume that $n = O(m)$.

Let $\sim$ be a relation on V. Two vertices u and v are congruent ***under*** $\sim$ if $\lambda(u) =$ A(v), s(u) $= \delta(v)$, and, for all i such that $1 \leq i \leq$ S(u), $u[i] \sim v[i]$. The relation $\sim$ is closed under congruences if, for all vertices u and v such that u and v are congruent under $\sim$, u-v. The congruence closure of $\sim$ is the finest equivalence relation that is closed under congruences and contains $\sim$.

The relation between the uniform word problem and congruence closure is given by the following theorem (see Shostak 78, or Nelson and Oppen 77):

Theorem. Let 6 be the set of defining equations of the algebra. Suppose that we wish to determine if $M = N$ follows from 6. Let G be a directed graph corresponding to the set of terms appearing in 6 and $M = N$, and let $\theta(P)$ be the vertex of G corresponding to the term P. Let $\sim$ be the congruence closure of $\{\langle \theta(U), \theta(V) \rangle \mid U = V \in 6)$. Then $M = N$ follows from 6 if and only if $O(M)$--$B(N)$.

Kozen (77) shows that the problem is in P (polynomial time), Nelson and Oppen (79) show that it admits a $O(n^2)$ solution, and Downey, **Sethi** and **Tarjan** (79) show that it admits a $O(n \log^2 n)$ solution. (The algorithms require linear space, and the times given apply whether the algorithms are online or offline.) It is still open whether the problem is solvable in linear time.

Canonical term rewriting systems may be used to solve word pmblems over algebras, in cases where the completion algorithm of section 7 terminates given the equations formed from the axioms of the variety, plus the presentation of the algebra.

Furthermore, the completion algorithm may be used to solve uniform word problems, if its termination can be established for all the presentations involved. This is the case, for instance, for the variety of all C-algebras, using a simple lexicographic ordering to show termination (Lankford, private communication). Here, the problems of superposing the **lefthand** sides and reducing terms both reduce to finding common subexpressions in the set of equations. The latter problem is the essential problem of congruence closure, and so can be implemented in worst-case time $O(n \log^2 n)$ and space O(n).

Greendlinger (60) gives conditions on the presentation of a group under which the algorithm of Dehn (11) solves the word problem. **Bücken** (79) shows that one can construct effectively, fmm the ten **group** reductions given in the Appendix and from the presentation $\mathfrak{P}$ of a group, a **nœtherian** term rewriting system $\mathfrak{R}$; then, under the same conditions as Greendlinger, one may decide the word problem as follows. Any relator (i.e. word equal to the identity in the group) reduces to the identity, using reductions from $\mathfrak{R}$ in any order; conversely, no other word reduces to the identity. We could say that $\mathfrak{R}$ is confluent on the relators. However, **non**-relators may possess several normal forms.

11. Separable Equational Theories

In this section, we assume Σ is a signature with booleans. Further, we assume that Σ contains an operator $\#$, with $\tau(\#) = s \times s \rightarrow$ boolean. We then say that 6 is ***s-separable*** if and only if

1. x#x $= FALSE \in \mathfrak{E}$ for $x \in \mathcal{V}_s$
2. for all M, N in $\mathcal{G}_s$, $M \neq_{\mathfrak{E}} N \Rightarrow M \# N =_{\mathfrak{E}} TRUE$.

Note that these conditions imply that, for all ground terms ***M*** and N of sort s, if M $=_{\mathfrak{E}}$ N then $M \# N =_{\mathfrak{E}} FALSE$, and otherwise $M \# N =_{\mathfrak{E}} TRUE$; and therefore that the word problem for sort s is decidable. For consistent s-separable theories, the validity problem for terms of sort s in the initial algebra is equivalent to a consistency problem, as follows.

Theorem. Let 6 be a consistent s-separable set of equations, and ***M, N*** be two terms of type ***s***. Then J(6) $\models M =$ N if and only if 6 $\cup$ {$M =$ N} is consistent.

Proof:

$\Rightarrow$ obvious since, if 6 is consistent, J(6) $\not\models$ ***TRUE*** $=$ ***FALSE.***

$\Leftarrow$ Assume 6' $=$ 6 $\cup$ {$M = N$} is consistent, and let σ be any ground substitution. We have $u(M) \# \sigma(N) =_{\mathfrak{E}'} FALSE$ by condition 1 of s-separability. Further, if $a(M) \neq_{\mathfrak{E}}$ u(N), then $a(M) \# u(N) =_{\mathfrak{E}} TRUE$ by condition 2, and

hence $u(M) \# u(N) =_{\mathfrak{E}'} TRUE$. This contradicts the consistency of 6'. Therefore, $\sigma(M) =_{\mathfrak{E}} \sigma(N)$, and thus J(C) $\models M = N$. ∎

More generally, let 6' be any set of equations $\{M_i = N_i \mid i \in I\}$, and let 6 be consistent and $\tau(M_i)$-**separable** for every i. Then J(6) $\models M_i = N_i$ for every i in I if and only if 6 ∪ 6' is consistent.

This theorem is inspired by **Guttag** and Horning (78) and Musser **(80),** and generalizes Goguen(79) (where a data type specification amounts to a theory s-separable for every sort s). It gives us a semi-decision procedure for the truth of sentences of the form $\exists \vec{x}\ M \neq N$ in J(6). This is not very surprising, since for instance in arithmetic this corresponds to checking finitely refutable statements. It **allows us** to limit induction to a once and for all proof of s-separability, and allows us to obtain proofs by pure equational reasoning that normally require induction. However, its practical importance seems to be limited, since consistency is not even recursively enumerable. It may be most useful for decidable theories, for which consistency is decidable too. We shall see below how it can be used in conjunction with the Knuth-Bendix extension algorithm.

Example: Let $\mathfrak{E}^1 = \mathfrak{E}^0 \cup \{x \# x = FALSE,\ S(x) \# S(y) = x \# y,\ 0 \# S(x) = TRUE,\ S(x) \# 0 = TRUE\}$. It is easy to check, by induction on $\mathcal{T}(\Sigma^0)$, that $\mathfrak{E}^1$ is integer-separable and consistent. Actually $\mathfrak{I}(\mathfrak{E}^1) = \mathcal{A}^0$. We will describe later how to show that $+$ is associative and commutative in $\mathfrak{I}(\mathfrak{E}^1)$, using the theorem above.

The theorem above is reminiscent of the traditional observation in first-order logic that a complete theory is maximally consistent. However, s-separable does not imply complete in the traditional sense. For instance, note that $\mathfrak{I}(\mathfrak{E}^1) \models 0 + x = x$. However $\mathcal{M}_0(\mathfrak{E}^1) \not\models 0 + x = x$, since, for instance, $\mathcal{A}^1 \in \mathcal{M}_{\Sigma^0}(\mathfrak{E}^1)$.

When the completion algorithm terminates, it may be used to decide the consistency of a theory, and may therefore be used for solving the validity problem in the initial algebra of certain separable equational theories. It must be emphasized however that this new use of the completion algorithm is different in spirit from the its use in generating a canonical form for an equational theory. Generating a canonical term rewriting system is similar to compilation: you run the algorithm only once for the theory in which you are interested, even if it is very costly. The future proofs will consist uniquely in reductions to normal forms, and are thus fairly efficient (at least intuitively). On the other hand, each proof in d(6) will require using the completion algorithm and its associated costs (such as pmofs of termination) even if 6 itself is already completed. However, disproofs may be obtained even when the completion algorithm does not terminate: as soon as $TRUE = FALSE$ is generated as a critical pair, we know that the theory is inconsistent.

We give examples of such proofs in the Appendix. In a simple theory of list structures we show the associativity of the ***Append*** function, defined recursively. Then we introduce the recursive definition of the ***Reverse*** function, and show that it verifies $Reverse(Reverse(x)) = x$. Finally, we introduce an iterative (that is, recursive only in a terminal position) version of ***Reverse,*** called ***Reviter,*** and show the equivalence of the two programs. Note that none of these proofs actually uses the inequality axioms, since no superposition with them is ever possible.

12. A **Meta-unification** Algorithm

In this section, we show how we can do unification in theories that admit a canonical term rewriting system. More precisely, let $\mathfrak{R}$ be a (finite) canonical term rewriting system obtained by running the completion algorithm on the set of equations 6. We describe here a **meta-unification** algorithm which, given any such $\mathfrak{R}$ and a finite set of variables W, generates a complete set of 6-unifiers away **from** W. This algorithm is due to Fay (79a), with improvements by Lankford. It combines in an elegant way ordinary unification and "narrowing", the process of effecting the minimum substitution to a normal form term so that the substituted term is not in normal form (Slagle 74).

More precisely, let M be in %-normal form, and V be a finite set of variables containing $Y(M)$. Let u be some non-variable occurrence in M such that the **subterm** M/u at u is unifiable with some **lefthand** side of a rule in $\mathfrak{R}$ using ordinary unification. In symbols, $M/u \notin \mathcal{V}$, and $\mathcal{U}_\emptyset(M/u, \lambda) \neq \emptyset$ with $\lambda \rightarrow \rho \in \mathfrak{R}$. Assume $\lambda \rightarrow \rho$ has been renamed away from V; that is, Y(h) $\cap V = \emptyset$. Let σ be the minimum unifier of M/u and λ, restricted to $v(M)$. That is, σ is the minimum substitution such that $\sigma(M/u)$ is an instance of λ. We say that σ is a narrowing substitution of M away from V, and we write $NS(M, V)$ for the (finite) set of such substitutions.

We are now ready to describe a (non-deterministic) 6-unification algorithm. $\mathfrak{R}$ is assumed to be a canonical term rewriting system for 6, and we write $\mathfrak{R}(M)$ for the %-normal form of term M.

6-unification algorithm.

The input consists of two terms M and N of the same type, and a finite set of variables W, with $\mathcal{V}(M) \cup$ Y(N) $\subseteq$ W. M_i and N_i are terms of the same sort, W_i is a set of variables, θ_i is a **substitution.** Variables σ and S denote respectively a substitution and a finite set of substitutions.

1. Initially let $M_0 = \mathfrak{R}(M), N_0 = \mathfrak{R}(N), W_0 = W, \theta_0 = \emptyset, i = 0.$

2. If $M_i = N_i$ then stop with answer θ_i. Otherwise let $S = NS(M_i, W_i) \cup NS(N_i, W_i)$, to which we add the minimum unifier of M_i and N_i if it exists. Select σ in S. Let $\theta_{i+1} = \sigma \circ \theta_i$. If for some x in W, $\theta_{i+1}(x)$ is not a normal form, stop with failure. Otherwise, let $M_{i+1} = \mathfrak{R}(\sigma(M_i))$, $N_{i+1} = \mathfrak{R}(\sigma(N_i))$, $W_{i+1} = W_i \cup \mathfrak{I}(\sigma)$, increment i by 1 and go to 2.

This algorithm can be viewed as non-deterministic, or equivalently as an algorithm enumerating a finitely-branching tree. This tree may be infinite, but if M and N are B-unifiable, then there will be some successfully terminating execution sequence, that is, there will be a success node at a finite level in the tree. Actually, a stronger result holds, as follows. Let $V = \mathcal{V}(M) \cup \mathcal{V}(N)$, and let $\mathcal{U}_{\mathcal{E}}(M, N, W)$ be the (possibly infinite) set of all answers θ_i, restricted to V; that is, $\mathcal{U}_{\mathcal{E}}(M, N, W) = \{\{\langle x, \theta_i(x)\rangle \mid x \in V\} \mid$ the algorithm stops with answer $\theta_i\}$.

Theorem. $\mathcal{U}_{\mathcal{E}}(M, N, W)$ is a complete set of $\mathcal{E}$-unifiers of M and N away from W.

For a proof, see Hullot(80) which expands Fay (79b).

Various optimizations of the algorithm are possible. In general, however, the algorithm will not enumerate a minimal set of unifiers, even when such a set exists. This is the case for associativity, for instance, for which Plotkin's algorithm (Plotkin 72) is preferred. It is an interesting open problem to refine of the algorithm above in such a way that it indeed generates a minimal set whenever possible. (The "obvious" solution, consisting of throwing away solutions subsumed by others, is not satisfactory, since the result may not be complete for theories which do not admit minimal complete sets of unifiers.)

This algorithm gave the first known solution to unification in group theory, using for $\mathfrak{R}$ the canonical set shown in the appendix.

13. Extensions and Combinations of Equational Theories

In previous sections we have discussed formalisms for describing and manipulating equational systems for different sorts of objects, and in particular have discussed methods for handling decision problems for particular equational theories. However, it is clear that in practice we want to "combine" equational theories, and in this section we discuss research presently being done on combinations of such theories. (See also Burstall and Goguen (79).)

It is also clear that in practice we wish to reason about formulas rather than just equations. We therefore wish to extend the language of equational calculi, in particular, to include boolean connectives and conditional expressions of the form if . . . then . . . else.

Consider the following axiomatization of the theory of arrays:

select(store($a, i,$ e), j) = if $i =$ j then e **else select(**a, j)
store(a, i, **select(**a, i**))** = a
store(store($a, i,$ e), j, f**)** =
  if $i =$ j then **store(**a, i, f**)** *else* **store(store(**a, j, f**),** i, e).

(The equality symbol appearing in the literal $i =$ j is equality in the theory of the array indices.) An obvious use of this axiomatization is to **prove** formulas such as $i \neq$ j $\supset$ **select(store(**$a, i,$ e), j) = **select(**a, j).

Consider the following (weak) axiomatization $\mathfrak{R}$ for the the reals under addition ("weak" because the axioms have other models, such as the integers under addition):

$x + 0 = x$
$x + (-x) = 0$
$(x + y) + z = x + (y + z)$
$x + y = y + x$
$x \not< x$
$x < y \vee y < x \vee x = y$
$x <$ y $\supset \neg(y < x)$
$x < y \wedge y < z \supset x < z$
$x < y \supset x + z < y + z$
$0 < 1$.

Using these two theories, we may wish to prove the theorem **select(store(**a, 0, e), 1) =**select(**a, 1); this is an example of a theorem in the "combination" of the two individual theories.

Some results are known on the decidability and complexity of combinations of theories such as these (Nelson and **Oppen(79a,79b), Oppen(79a,** 79b)). As we shall see, equations between variables play a crucial role.

We assume that we have several quantifier-free theories formalized in classical first-order logic with equality, extended to include the three-argument conditional if-then-else. The symbols =, A, $\vee$, $\neg$, $\supset$, **V,** 3 and if-then-else are common to all theories; we call them the logical symbols. Each theory is characterized in the usual way by its set of non-logical symbols and non-logical axioms.

Let us give some examples. Define the quantifier-free theory $\mathcal{A}$ of arrays to have **store** and select as non-logical symbols and the axioms given above. Define the quantifier-free theory $\mathfrak{R}$ of reals under addition to 0, 1, $+, <$ as non-logical symbols and axioms as given above. Define the theory $\mathcal{L}$ of list structure to have car, **cdr,** cons and atom as non-logical symbols and to have the following axiomatization:

$\mathbf{car}(\mathbf{cons}(x, y)) = x$
$\mathbf{cdr}(\mathbf{cons}(x, y)) = y$
$\neg\ \mathbf{atom}(\mathrm{x}) \supset \mathbf{cons}(\mathbf{car}(x), \mathbf{cdr}(x)) = \mathrm{x}$
$\neg\ \mathbf{atom}(\mathbf{cons}(x, \mathrm{y}))$.

Finally, define the theory 6 whose non-logical symbols are all uninterpreted function symbols. 6 has no axioms, so it is just the quantifier-free theory of equality.

A theory is stably infinite if, for all quantifier-free formulas F, if F has a model in the theory, then it has an infinite model. Note that any theory that has no finite models is stably infinite; therefore $\mathfrak{B}$ is stably infinite. It is not difficult to prove that 6, $\mathcal{L}$ and $\mathcal{A}$ are stably infinite.

Let $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_k$ be k theories with no common non-logical symbols. Their combination, denoted $\bigcup_i(\mathcal{T}_i)$, is the theory whose set of non-logical symbols is the union of the sets of non-logical symbols of the $\mathcal{T}_i$, and whose set of axioms is the union of the sets of axioms of the $\mathcal{T}_i$. We do not consider combining theories which share non-logical symbols. The following theorem is proved by Nelson and Oppen (79b).

Theorem. Let $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_k$ be k decidable and stably-infinite quantifier-free theories with no common non-logical symbols. Then $\bigcup_i(\mathcal{T}_i)$ is decidable.

The proof is constructive; a decision procedure for the union is described for determining the satisfiability of conjunctions of literals. The decision procedure "combines" the decision procedures for the individual theories as follows.

Suppose we have just two theories $\mathcal{S}$ and $\mathcal{T}$ and wish to determine if a quantifier-free conjunction F of literals in their combined language is satisfiable. We assume that $F \equiv F_{\mathcal{S}} \wedge F_{\mathcal{T}}$, where $F_{\mathcal{S}}$ is a conjunction of literals in the language of $\mathcal{S}$ and $F_{\mathcal{T}}$ is a conjunction of literals in the language of 9'. (Any formula can be put in this form by introducing new variables; for instance, corresponding to the above formula $\mathbf{select}(\mathbf{store}(a, \mathbf{0}, \mathbf{e}), 1) = \mathbf{select}(a, 1)$ is the equivalent formula $\mathbf{select}(\mathbf{store}(a, \mathrm{x}, \mathrm{e}), \mathrm{y}) = \mathbf{select}(a, \mathrm{y}) \wedge \mathrm{x} = 0 \wedge \mathrm{y} = 1$ in the required form.) The following algorithm determines whether F is satisfiable. The algorithm uses the variables $F_{\mathcal{S}}$ and $F_{\mathcal{T}}$ which contain conjunctions of literals.

Equality Propagation Procedure.

1. [Unsatisfiable?] If either $F_{\mathcal{S}}$ or $F_{\mathcal{T}}$ is unsatisfiable, then F is unsatisfiable.

2. [Propagate equalities.] If either $F_{\mathcal{S}}$ or $F_{\mathcal{T}}$ entails some equality between variables not entailed by the other, then add the equality as a new conjunct to the one that does not entail it. Go to step 1.

3. [Case split necessary?] If either $F_{\mathcal{S}}$ or $F_{\mathcal{T}}$ entails a disjunction $u_1 = v_1 \vee \ldots \vee u_k = v_k$ of equalities between variables, without entailing any of the equalities

alone, then apply the procedure recursively to the k formulas $F_{\mathcal{T}} \wedge F_{\mathcal{T}} \wedge u_1 = v_1, \ldots, F_{\mathcal{T}} \wedge F_{\mathcal{T}} \wedge u_k = v_k$. If any of these formulas are satisfiable, then F is satisfiable. Otherwise F is unsatisfiable.

4. If we reach this step, F is satisfiable.

The following theorems on the complexity of combinations of theories appear in **Oppen(79b).**

Theorem. Let $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_k$ be as before. If the satisfiability problems for each of the $\mathcal{T}_i$ is in NP, then the satisfiability problem for $\bigcup_i(\mathcal{T}_i)$ is in NP and hence NP-complete. Otherwise the complexity of the satisfiability problem is dominated by the maximum of the complexities of the satisfiability problems for the $\mathcal{T}_i$.

Corollary. The satisfiability problem for $\mathcal{R} \cup \mathcal{E} \cup \mathcal{A} \cup \mathcal{L}$, that is, the quantifier-free theory of reals, arrays, list structure and uninterpreted function symbols under +, $\leq$, store, select, cons, car and cdr is NP-complete.

However, under certain conditions, a result on deterministic time is possible. A formula is non-convex if it entails a disjunction of equalities between variables without entailing any of the equalities alone; otherwise it is convex. Define a theory to be convex if every conjunction of literals in the language of the theory is convex; otherwise if is non-convex.

Some of the theories considered above are convex, others non-convex. The theories of equality with uninterpreted function symbols and of list structure under car, **cdr** and cons are convex (Nelson and Oppen 79a). The theory of rationals under + and $\leq$ is convex: the solution set of a conjunction of linear inequalities is a convex set; the solution set of a disjunction of equalities is a finite union of hyperplanes; and a convex set cannot be contained in a finite union of hyperplanes unless it is contained in one of them. The theories of integers under addition and of integers under successor are non-convex. For instance, the formula $1 \leq x \leq 2 \wedge y = 1 \wedge z = 2$ entails the disjunction $x = y \vee x = z$ without entailing either equality alone. The theory of arrays is non-convex. For instance, **the formula** $x = \mathbf{select}(\mathbf{store}(a, i, e), j) \wedge y = \mathbf{select}(a, j)$ entails $i = j \vee x = y$. The theory of **the reals under multiplication** is **not convex; for example, xy = 0 A z = 0 entails the disjunction x = z V y = z.** The theory of sets is also non-convex; for example, consider {a, ***b, c}*** $\cap$ ***{c, d, ej*** $\neq \emptyset$.

Define the *DNF* satisfiability ***problem*** to be the problem of determining if a quantifier-free formula in disjunctive normal form is satisfiable.

Theorem. Let $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_k$ be decidable, convex, quantifier-free theories with no common non-logical symbols and with deterministic polynomial time DNF satisfiability problems. Then $\bigcup_i(\mathcal{T}_i)$ sb h as a deterministic polynomial time DNF satisfiability problem.

Thus, for instance, the theory $\mathfrak{T} \cup \mathfrak{S}$ has a polynomial time decision procedure for formulas in disjunctive normal form. This follows from Khachian (78) and Nelson and Oppen (79a).

The critical notion in these results on combinations of sorted-theories is that of equality between variables. In the equality-sharing decision procedure given above for combining decision procedures, the only information propagated between decision procedures is either an equality between variables or a disjunction of equalities between variables. And the complexity results show that the complexity of the combination is critically related to the "convexity" of the theories -- whether or not conjunctions of literals can entail disjunctions of equalities.

These results can therefore be looked at as reducing the decision problem for non-equational theories to simpler decision problems of equational theories.

14. Further Results

For lack of space, a number of recent results are not described here. Let us indicate briefly a few promising directions.

The Knuth-Bendix characterization of confluence for **nœtherian** term-rewriting systems may be extended to congruence classes of terms under certain conditions. This has been done for commutativity (Lankford and Ballantyne **77a),** for **commutativity** and associativity (Lankford and Ballantyne **77c,** Peterson and **Stickel 77),** and in the general case for certain left-linear term rewriting systems (Huet 77). The completion algorithm can be extended to these cases, generating canonical systems for **abelian** groups, **abelian** rings, distributive lattices. Degano and Sirovich (79) use a complete set of such rewrite rules to show the decidability of equivalence for a new class of primitive recursive functions. Ballantyne and **Lankford** (79) show that this extension of the completion algorithm solves the uniform word problem for finitely presented commutative semigroups — it is conjectured that these methods extend to **abelian** groups and rings.

When term rewriting systems are not **nœtherian,** the problem arises of how to compute the normal forms of terms which have one. This problem of order of evaluation has well-known solutions for ordinary recursive definitions (Vuillemin 74, Downey and **Sethi** 76, Raoult and Vuillemin 78, Berry and **Lévy 79),** but is harder for more general term rewritings. Huet and Lkvy (79) give a strong sequentiality criterion permitting efficient implementations of correct interpreters for term rewriting systems whose **lefthand** sides are linear, and which have no critical pairs. See also **Hoffmann** and O'Donnell (79) for related results.

15. Acknowledgments

We thank Ron Book, Bob Boyer, Nachum Dershowitz, Joe Goguen, Jean-Marie Hullot and Jean-Jacques Lévy for their many helpful comments. We especially thank Dallas Lankford for his lively and helpful correspondence.

16. Appendix

This appendix is an image of a computer session run on the Stanford Artificial Intelligence Laboratory KL-10. The program, developed by Jean-Marie Hullot at IRIA, is written in VLISP (developed at Université de Vincennes by Patrick Greussay and Jérôme Chailloux). The times given are for the program run interpretively.

We first give an example of the Knuth-Bendix completion algorithm compiling a canonical term rewriting system from the standard set of three axioms for groups. User input is preceded by a question mark. Comments are surrounded by square brackets.

```
? (kbini)
MODE ? auto
LIST OF OPERATORS ? (0 + I)
WEIGHT OF 0 ? 1
WEIGHT OF + ? 0
WEIGHT OF I ? 0
MINIMUM WEIGHT OF A PURE WORD ? 1
LIST OF INFIX OPERATORS ? (+)
= READY
= ; time = 6 ms ;
[Now all the weights are set up, so that the termination will be proved automatically, using the original Knuth-Bendix method.]
? (kb group)
```

R1 : O+X —>X	GIVEN
R2 : I(X)+X —>O	GIVEN
R3 : (X+Y)+Z —>X+(Y+Z)	GIVEN
R4 : I(X)+(X+Y) —>Y	FROM R3 AND R2
R5 : I(O)+X —>X	FROM R4 AND R1
R6 : I(I(X))+O —>X	FROM R4 AND R2

R7 : I(I(O))+X $\longrightarrow$X — FROM R5 AND R4

R8 : I(0) $\longrightarrow$O — FROM R7 AND R2
R5 DELETED
REWRITE RULES : R8 R1 FOR LEFT PART
R7 DELETED
REWRITE RULES : R8 R8 R1 FOR LEFT PART

R9 : I(I(I(X)))+X $\longrightarrow$O — FROM R6 AND R4

R10 : I(I(X))+Y ->X+Y — FROM R4 AND R.4
R6 REPLACED BY :
X+O = X
REWRITE RULES : R10 FOR LEFT PART
R9 DELETED
REWRITE RULES : R10 R2 FOR LEFT PART

R11 : X+O $\longrightarrow$X — FROM R6

R12 : I(I(X)) $\longrightarrow$X — FROM R11 AND R10
R10 DELETED
REWRITE RULES : R12 FOR LEFT PART

R13 : X+I(X) $\longrightarrow$O — FROM R12 AND R2

R14 : X+(I(X)+Y) $\longrightarrow$Y — FROM R12 AND R.4

R15 : X+(Y+I(X+Y)) $\longrightarrow$O — FROM R13 AND R3

R16 : X+I(Y+X) $\longrightarrow$I(Y) — FROM R15 AND R4
R15 DELETED
REWRITE RULES : R16 R13 FOR LEFT PART

R17 : I(X+Y) $\longrightarrow$I(Y)+I(X) — FROM R16 AND R4
R16 DELETED
REWRITE RULES : R17 R14 FOR LEFT PART

COMPLETE SET: CGROUP
R1 : O+X $\longrightarrow$X
R2 : I(X)+X $\longrightarrow$O
R3 : (X+Y)+Z $\longrightarrow$X+(Y+Z)
R4 : I(X)+(X+Y) $\longrightarrow$Y
R8 : I(0) ->o
R11 : X+O $\longrightarrow$X

R12 : I(I(X)) —>X
R13 : X+I(X) —>O
R14 : X+(I(X)+Y) —>Y
R17 : I(X+Y) —>I(Y)+I(X)

9-Dec-79 21:19:43
= E N D
= ; time = 4790 ms ;
[The file PROOFS contains a simple axiomatization of list structures, simple recursive definitions of list manipulation programs, and lemmas proving their correctness. LISP is the axiomatization of list structures; note that it is list-separable.]
? (library proofs)
= PROOFS
= ; time = 41 ms ;
? (kbini)
MODE ? free
LIST OF INFIX OPERATORS ?(#)
= READY
= ; time = 2 ms ;
? (kb lisp)

X#X = FALSE GIVEN
COMMAND ? y
R1 : X#X —>FALSE

CONS(X,Y)#NULL = TRUE GIVEN
COMMAND ? y
R2 : CONS(X,Y)#NULL —>TRUE

NULL#CONS(X,Y) = TRUE GIVEN
COMMAND ? y
R3 : NULL#CONS(X,Y) —>TRUE

CONS(X,Y)#CONS(Z,U) = IF(X#Z,TRUE,Y#U) GIVEN
COMMAND ? y
R4 : CONS(X,Y)#CONS(Z,U) —>IF(X#Z,TRUE,Y#U)

IF(TRUE,X,Y) = X GIVEN
COMMAND ? y
R5 : IF(TRUE,X,Y) —>X

IF(FALSE,X,Y) = Y GIVEN

COMMAND ? y
R6 : IF(FALSE,X,Y) ⟶ **Y**

APPEND(NULL,X) = X GIVEN
COMMAND ? y
R7 : **APPEND(NULL,X) ⟶ X**

APPEND(CONS(X,Y),Z) = CONS(X,APPEND(Y,Z)) GIVEN
COMMAND ? y
R8 : APPEND(CONS(X,Y),Z) ⟶ **CONS(X,APPEND(Y,Z))**

COMPLETE SET: CLISP
R1 : **X≠X ⟶ FALSE**
R2 : CONS(X,Y)#NULL ⟶ **TRUE**
R3 : **NULL≠CONS(X,Y) ⟶ TRUE**
R4 : CONS(X,Y)#CONS(Z,U) ⟶ **IF(X≠Z,TRUE,Y≠U)**
R5 : IF(TRUE,X,Y) ⟶ **X**
R6 : **IF(FALSE,X,Y) ⟶ Y**
R7 : APPEND(NULL,X) ⟶ **X**
R8 : APPEND(CONS(X,Y),Z) ⟶ **CONS(X,APPEND(Y,Z))**

9-Dec-79 22: **13:34**
= E N D
= ; time = 905 ms ;
[**Let's** prove that APPEND is associative. PROOF1 contains the corresponding lemma.]
? (provelemma proof 1)

APPEND(APPEND(X,Y),Z) = APPEND(X,APPEND(Y,Z)) GIVEN
COMMAND ? y
R9 : **APPEND(APPEND(X,Y),Z) ⟶ APPEND(X,APPEND(Y,Z))**

COMPLETE SET: **CPROOF1**
R1 : **X≠X ⟶ FALSE**
R2 : **CONS(X,Y)≠NULL ⟶ TRUE**
R3 : **NULL≠CONS(X,Y) ⟶ TRUE**
R4 : CONS(X,Y)#CONS(Z,U) ⟶ **IF(X≠Z,TRUE,Y≠U)**
R5 : IF(TRUE,X,Y) ⟶ **X**
R6 : IF(FALSE,X,Y) ->Y
R7 : APPEND(NULL,X) ⟶ **X**
R8 : APPEND(CONS(X,Y),Z) ⟶ **CONS(X,APPEND(Y,Z))**

R9 : APPEND(APPEND(X,Y),Z) —>APPEND(X,APPEND(Y,Z))

9-Dec-79 22: 16:09
= END
= ; time = 640 ms ;
[The resulting set is consistent, proving the lemma. Next we enrich our lisp theory with the recursive definition of the function REV, that reverses a list. We then prove that REV(REV(x))=x for every list x.]
? (kb clisp pmof2)
R1 : X#X —>FALSE
R2 : CONS(X,Y)#NULL —>TRUE
R3 : NULL#CONS(X,Y) —>TRUE
R4 : CONS(X,Y)#CONS(Z,U) —>IF(X#Z,TRUE,Y#U)
R5 : IF(TRUE,X,Y) —>X
R6 : IF(FALSE,X,Y) —>Y
R7 : APPEND(NULL,X) —>X
R8 : APPEND(CONS(X,Y),Z) —>CONS(X,APPEND(Y ,Z))

REV(NULL)= NULL GIVEN
COMMAND ? y
R9 : REV(NULL) —>NULL

REV(CONS(X,Y)) = APPEND(REV(Y),CONS(X,NULL)) GIVEN
COMMAND ? y
R10 : REV(CONS(X,Y)) —>APPEND(REV(Y),CONS(X,NULL))

REV(REV(X)) = X GIVEN
COMMAND ? y
R11 : REV(REV(X)) —>X

REV(APPEND(FtEV(X),CONS(Y,NULL))) = CONS(Y,X)
FROM R11 AND R10
COMMAND ? y
R12 : REV(APPEND(REV(X),CONS(Y,NULL))) —>CONS(Y,X)

REV(APPEND(X,CONS(Y,NULL))) = CONS(Y,REV(X))
FROM R12 AND R11
COMMAND ? y
R13 : REV(APPEND(X,CONS(Y,NULL))) —>CONS(Y,REV(X))
R12 DELETED
REWRITE RULES : R13 Rll FOR LEFT PART

COMPLETE SET: CPROOF2
R1 : X#X —>FALSE
R2 : CONS(X,Y)#NULL —>TRUE
R3 : NULL#CONS(X,Y) —>TRUE
R4 : CONS(X,Y)#CONS(Z,U) —>IF(X#Z,TRUE,Y#U)
R5 : IF(TRUE,X,Y) ->X
R6 : IF(FALSE,X,Y) —>Y
R7 : APPEND(NULL,X) —>X
R8 : APPEND(CONS(X,Y),Z) —>CONS(X,APPEND(Y,Z))
R9 : REV(NULL) —>NULL
R10 : REV(CONS(X,Y)) —>APPEND(REV(Y),CONS(X,NULL))
R11 : REV(REV(X)) —>X
R13 : REV(APPEND(X,CONS(Y,NULL))) —>CONS(Y,REV(X))

9-Dec-79 22:21:57
= E N D
= ; time = 2272 ms ;
[Complete and consistent! That is, REV(REV(x))=x in LISP. Notice that the termination test has been left to the user, because the Knuth-Bendix ordering could not be used on rule R10. The orientation of each rule was given by the user after a COMMAND? ("y" means keep left to right.) This recursive REVERSE is not very efficient. Let's write an iterative version, called REVITER, and prove the equivalence of the two programs: REV(x)=REVITER(x,NULL). We need the associativity of APPEND, and the lemma REVITER(x,y)=APPEND(REV(x),y). Note: this crucial lemma is not assumed; it is proved as well.]
? (kb cproofl proof3)
R1 : X#X —>FALSE
R2 : CONS(X,Y)#NULL —>TRUE
R3 : NULL#CONS(X,Y) —>TRUE
R4 : CONS(X,Y)#CONS(Z,U) —>IF(X#Z,TRUE,Y#U)
R5 : IF(TRUE,X,Y) —>X
R6 : IF(FALSE,X,Y) —>Y
R7 : APPEND(NULL,X) —>X
R8 : APPEND(CONS(X,Y),Z) —>CONS(X,APPEND(Y,Z))
R9 : APPEND(APPEND(X,Y),Z) —>APPEND(X,APPEND(Y,Z))

REV(NULL)= NULL GIVEN
COMMAND ? y
R10 : REV(NULL) —>NULL

REV(CONS(X,Y)) = APPEND(REV(Y),CONS(X,NULL)) GIVEN
COMMAND ? y
R11 : REV(CONS(X,Y)) —>APPEND(REV(Y),CONS(X,NULL))

REVITER(NULL,X) = X GIVEN
COMMAND ? y
R12 : REVITER(NULL,X) ->X

REVITER(CONS(X,Y),Z) = REVITER(Y,CONS(X,Z)) GIVEN
COMMAND ? y
R13 : REVITER(CONS(X,Y),Z) —>REVITER(Y,CONS(X,Z))

APPEND(REV(X),Y) = REVITER(X,Y) GIVEN
COMMAND ? y
R14 : APPEND(REV(X),Y) —>REVITER(X,Y)
R11 REPLACED BY :
REV(CONS(X,Y)) = REVITER(Y,CONS(X,NULL))
REWRITE RULES : R14 FOR RIGHT PART

REV(CONS(X,Y)) = REVITER(Y,CONS(X,NULL)) FROM R11
COMMAND ? y
R15 : REV(CONS(X,Y)) —>REVITER(Y,CONS(X,NULL))

REV(X) = REVITER(X,NULL) GIVEN
COMMAND ? y
R16 : REV(X) —>REVITER(X,NULL)
R10 DELETED
REWRITE RULES : R16 R12 FOR LEFT PART
R14 REPLACED BY :
APPEND(REVITER(X,NULL),Y) = REVITER(X,Y)
REWRITE RULES : R16 FOR LEFT PART
R15 DELETED
REWRITE RULES : R16 R13 FOR LEFT PART

APPEND(REVITER(X,NULL),Y) = REVITER(X,Y) FROM R14
COMMAND ? y
R17 : APPEND(REVITER(X,NULL),Y) ->REVITER(X,Y)

APPEND(REVITER(X,Y),Z) = REVITER(X,APPEND(Y,Z))
FROM R17 AND R9
COMMAND ? y
R18 : APPEND(REVITER(X,Y),Z) ->REVITER(X,APPEND(Y,Z))

```
R17 DELETED
REWRITE RULES : R18 R7 FOR LEFT PART
-----------------------------------------------------------------
COMPLETE SET: CPROOF3
R1 : X#X ->FALSE
R2 : CONS(X,Y)#NULL ->TRUE
R3 : NULL#CONS(X,Y) ->TRUE
R4 : CONS(X,Y)#CONS(Z,U) ->IF(X#Z,TRUE,Y#U)
R5 : IF(TRUE,X,Y) ->X
R6 : IF(FALSE,X,Y) ->Y
R7 : APPEND(NULL,X) ->X
R8 : APPEND(CONS(X,Y),Z) ->CONS(X,APPEND(Y,Z))
R9 : APPEND(APPEND(X,Y),Z) ->APPEND(X,APPEND(Y,Z))
R12 : REVITER(NULL,X) ->X
R13 : REVITER(CONS(X,Y),Z) ->REVITER(Y,CONS(X,Z))
R16 : REV(X) ->REVITER(X,NULL)
R18 : APPEND(REVITER(X,Y),Z) ->REVITER(X,APPEND(Y,Z))
-----------------------------------------------------------------
9-Dec-79 22:36:34
= E N D
= ; time = 3358 ms ;
-----------------------------------------------------------------
```

17. References

1. Ackermann W., Solvable Cases of the Decision Problem. North-Holland, Amsterdam, 1954.
2. Aho A., Sethi R. and Ullman J., Code Optimization and Finite *Church-Rosser* Systems. in Proceedings of Courant Computer Science Symposium 5, Ed. Rustin R., Prentice Hall (1972).
3. Ashcroft E.A. and Wadge W.W., Lucid - A Formal System for Writing and Proving Programs. SIAM Journal on Computing 5,3 (1976).
4. Aubin R., Mechanizing Structural *Induction.* Ph.D. thesis, U. of Edinburgh, Edinburgh 1976.
5. Backus J., Programming Language Semantics and *Closed* Applicative Languages. ACM Symposium on Principles of Programming (1973), 71-86.
6. Backus J., Can Programming be Liberated from the von Neumann *Style?* A Functional Style and Its Algebra of Programs. CACM 21,8 (1978), 613-641.

7. Ballantyne A.M. and Lankford D.S., New Decision *Algorithms for Finitely Presented* Commutative Semigroups. Report MTP-4, Department of Mathematics, Louisiana Tech. U., May 1979.

8. Baxter L. D., Au *Efficient* Unification Algorithm. Technical Report CS-73-23, Dept. of Applied Analysis and Computer Science, U. of Waterloo, 1973.

9. Bergman G.M., The Diamond Lemma *for* Ring Theory. Advances in Math. 29 (1978), 178-218.

10. Berry G. and Lévy J.J., Minimal and Optimal Computations of *Recursive* Programs. JACM 26,1 (1979).

11. Berry G. and Lévy J.J., Letter to *the* Editor. Sigact News 11,1 (1979).

12. Birkhoff G., On the Structure of Abstract Algebras. Proc. Cambridge Phil. Soc. 31 (1935), 433-454.

13. Birkhoff G. and Lipson J.D., Heterogeneous *Algebras*. Journal of Combinatorial Theory 8 (1970), 115-133.

14. Boone W., The *Word Problem*. Ann. of Math. 2,70 (1959), 207–265.

15. Boyer R. and Moore J, Proving Theorems About LISP Functions. JACM 22 (1975), 129-144.

16. Boyer R. and Moore J, A Lemma *Driven* Automatic Theorem *Prover for Recursive* Function Theory. 5th International Joint Conference on Artificial Intelligence (1977), 511-519.

17. Boyer R. and Moore J, A Computational Logic. Academic Press, 1979.

18. Brainerd W.S., *Tree* Generating Regular *Systems*. Information and Control 14 (1969), 217-231.

19. Brand D., Darringer J. and Joyner J., *Completeness* of *Conditional Reductions*. Symposium on Automatic Deduction, 1979.

20. Brown T., A Structured Design *Method for* Specialized *Proof* Procedures. Ph.D. Thesis, California Inst. of Tech., Pasadena, California, 1975.

21. Bücken H., Reduction *Systems* and Small Cancellation *Theory*. Proc. Fourth Workshop on Automated Deduction, (1979), 53-59.

22. Burstall R.M., Proving *properties of Programs* by *Structural* Induction. Comp uter J. 12 (1969), 41-48.

23. Burstall R.M., Design Considerations *for* a *Functional Programming Language*. Infotech State of the Art Conference, Copenhagen, 1977.

24. Burstall R.M. and Darlington J., A Transformation System for Developing *Recursive* Programs. JACM 24 (1977), 44–67.

25. Burstall R.M. and Goguen J.A., Putting Theories Together to Make Specifications. 5th International Joint Conference on Artificial Intelligence (1977), 1045-1058.

28. Cadiou J.M., Recursive Definitions of Partial Functions and Their Computations. PhD Thesis, Computer Science Department, Stanford U., 1972.

27. Cardoza E., Lipton R., and Meyer A., Exponential Space Complete Problems for Petri Nets and Commutative Semigroups. Proceedings of the Eighth ACM Symposium on Theory of Computing, May 1976, 50-54.

28. Cargill T.A., Deterministic Operational Semantics for Lucid. Report CS-76-19, U. of Waterloo (1976).

29. Church A., The Calculi of Lambda-Conversion. Princeton U. Press, Princeton N. J. (1941).

30. Church A. and Rosser J.B., Some Properties of Conversion. Transactions of AMS 39 (1936), 472-482.

31. Cohen P. J., Decision Procedures for Real and *p-adic* Fields. Comm. Pure and Appl. Math. 22 (1969), 131-151.

32. Cohn P.M., Universal Algebra. Harper and Row, New York, 1965.

33. Collins G., Quantifier Elimination for Real *Closed* Fields by *Cylindrical* Algebraic Decomposition. Proc. 2nd GI Conference on Automata and Formal Languages, Kaiserslauten, 1975. Lecture Notes in Computer Science, Springer-Verlag, to appear.

34. Colmerauer A., *Les* grammaires de *métamorphose.* Rapport interne, U. de Marseille-Luminy, 1975.

35. Courcelle B., Infinite Trees in Normal Form and Recursive Equations Having a Unique Solution. Rapport 7906, U. de Bordeaux 1, UER de Mathématiques et Informatique, Février 1979. To appear Math. Systems Theory.

36. Curry H.B. and Feys R., Combinatory Logic Vol. *I.* North-Holland, Amsterdam, 1958.

37. Davis M., *Hilbert's* Tenth Problem is Unsolvable. Amer. Math. Monthly 80,3 (1973), 233-269.

38. Degano P. and Sirovich F., On Solving the Equivalence Problem for a Subclass of Primitive Recursive Functions. Note Scientifiche S-79-18, Istituto di Scienze dell'Informazione, Pisa, Giugno 1979.

39. Dehn M., *Uber unendliche diskontinuierliche* Gruppen. Math. Ann. 71 (1911), 116–144.

40. Dershowitz N., A Note on Simplification Orderings. Information Processing Letters 9,5 (1979), 212-215.

41. Dershowitz N., Orderings for Term-rewriting Systems. Proc. 20th Symposium on Foundations of Computer Science (1979), 123-131. To appear Theoretical Computer Science.

42. Dershowitz N. and Manna Z., Proving Termination with *Multiset* Orderings. CACM 22 (1979), 465-476.

43. Downey P.J. and Sethi R., Correct Computation Rules for Recursive Languages. SIAM J. on Computing 5,3 (1976), 378-401.

44, Downey P. and Sethi R., Assignment Commands with Array References. JACM (1978), volume 25, no. 4.

45. Downey P., Sethi R. and Tarjan R. E., Variations on the Common *Subexpression* Problem. To appear JACM, 1980.

46. Ehrig H. and Rosen B.K., Commutativity of *Independent* Transformations on Complex Objects. IBM Research Report RC 6251, 1976.

47. Ehrig H. and Rosen B.K., The Mathematics of Record Handling. Fourth International Colloquium on Automata, Languages and Programming, Turku, 1977.

48. Emelichev V. A., Commutative Semigroups with One Defining Relation. Shuya Gosudarstvennyi Pedagogicheskii Institut Uchenye Zapiski, vol. 6, 1958, 227-242.

49. Evans T., The Word Problem for Abstract Algebras. J. London Math. Soc. 26 (1951), 64–71.

50. Evans T., *Embeddability* and the *Word* Problem. J. London Math. soc. 28 (1953), 76–80.

51. Evans T., On *Multiplicative* Systems Defined by Generators and Relations *1.,* Normal Form Theorems. Proc. Cambridge Phil. Soc. 47 (1951), 637–649.

52. Evans T., Some Solvable Word Problems. Proc. Conf. on Decision Problems in Algebra, Oxford, 1976. To appear, North-Holland, Amsterdam.

53. Evans T., Word Problems. Bulletin of the AMS 84,5 (1978), 789–802.

54. Evans T., Mandelberg K. and Neff M.F., Embedding Algebras with Solvable Word Problems in Simple Algebras. Some *Boone-Higman* Type Theorems. Proc. Logic Colloq., U. of Bristol, 1973. North-Holland, Amsterdam, 1975, 259-277.

55. Fay M., First-order Unification *in* an Equational Theory. Master Thesis, U. of California at Santa Cruz. Tech. Report 78-5-002, May 1978.

56. Fay M., First-order Unification in an *Equational* Theory. 4th Workshop on Automated Deduction, Austin, Texas, Feb. 1979, 161-167.

57. Goguen J.A., Proving *Inductive Hypotheses* Without Induction *and Evaluating* Expressions *with* Non-terminating Rules. Unpublished manuscript, Oct. 1979.

58. Goguen J.A. and Tardo J.J., An *Introduction* to *OBJ,* a Language for Writing and Testing Formal Algebraic Specifications. Specifications of Reliable Sofware Conference, Boston, 1979.

59. Goguen J., Thatcher J., Wagner E. and Wright J., Abstract Data Types as *Initial* Algebras and Correctness of Data Representations. Conference on Computer Graphics, Pattern Recognition and Data Structure, May 1975, 89–93.

60. Goguen J.A., Thatcher J.W., Wagner E.G. and Wright J.B., *Initial* Algebras Semantics and Continuous Algebras. JACM 24 (1977), 68-95.

61. Goguen J.A., Thatcher J.W. and Wagner E.G., An Initial *Algebra* Approach to the Specification, Correctness, and Implementation of Abstract Data Types. "Current Trends in Programming Methodology", Vo14, Ed. Yeh R., Prentice-Hall (1978), 80-149.

62. Goldfarb W., The *Undecidability* of the Second-order Unification Problem. Unpublished manuscript, July 1979.

63. Gorn S., Handling the Growth by Definition of Mechanical *Languages.* Proc. Spring Joint Computer Conf. (1967), 213-224.

64. Gorn S., Explicit Definitions *and Linguistic Dominoes.* "Systems and Computer Science", Eds Hart J. and Takasu S., U. of Toronto Press (1967), 77-115.

65. Gorn S., On the Conclusive Validation of Symbol Manipulation Processes (How Do You Know *It* has To *Work?).* J. of the Franklin Institute, 296,6 (1973), 499-518.

66. 'Greendlinger M., Dehn's Algorithm for the Word Problem. Communications on Pure and Applied Mathematics, 13 (1960), 67-83.

67. Griesmer J.H. and Jenks R.D., SCRATCHPAD/I. An Interactive Facility for Symbolic Mathematics. Proceedings 2nd Symposium on Symbolic and Algebraic Manipulation, Ed. Petrick S., Los Angeles, March 1971.

68. Guard J.R., Oglesby F.C., Bennett J.H. and Settle L.G., Semi-automated Mathematics. JACM 16 (1969), 49-62.

69. Guttag J., The Specification and Application to Programming of Abstract Data Types. Ph. D. thesis, U. of Toronto, 1975.

70. Guttag J.V., Abstract Data Types and the Development of Data Structures. CACM 20 (1977), 397–404.

71. Guttag J.V., Notes on Type Abstraction. To appear, IEEE Transactions on Software Engineering.

72. Guttag J.V. and Horning J.J., The Algebraic Specification of Abstract Data Types. Acta Informatica 10 (1978), 27-52.

73. Guttag J.V., Horowitz E. and Musser D., The Design of Data Type Specifications. "Current Trends in Programming Methodology", Vo14, Data Structuring, Ed. Yeh R., Prentice Hall, 1978.

74. Guttag J.V., Horowitz E. and Musser D.R., Abstract Data Types and Software Validation. CACM 21 (1978), 1048–1064.

75. Hall P., Some Word Problems. J. London Math. Soc. 33 (1958), 482-496.

76. Henderson P. and Morris J.H. Jr., A Lazy Evaluator. Third ACM Conference on Principles of Programming Languages (1976), 95-103.

77. Hermann G., Die Frage der endlich vielen Schritte in der Theorie der Polynomideale, Math. Ann., Vol 95, 1926, 736738.

78. Hindley R., An Abstract Form of the Church-Rosser Theorem I. J. of Symbolic Logic 34,4 (1969), 545–560.

79. Hindley R., An Abstract Form of the Church-Rosser Theorem II: Applications. J. of Symbolic Logic 39,1 (1974), 1-21.

80. Hoffmann M. and O'Donnell M., Interpreter Generation Using Tree Pattern Matching. 6th ACM Conference on Principles of Programming Languages (1979).

81. Huet G., The Undecidability of Unification in Third Order Logic. Information and Control 22 (1973), 257-267.

82. Huet G., A Unification Algorithm for Typed Lambda Calculus. Theoretical Computer Science, 1,1 (1975), 27-57.

83. Huet G., Résolution d'équations dans des Zangages d'ordre $1, 2, \ldots, \omega$. Thèse d'Etat, Université de Paris VII, 1976.

84. Huet G., Confluent Reductions: Abstract properties and Applications to Term Rewriting Systems. 18th IEEE Symposium on Foundations of Computer Science (1977), 30-45.

85. Huet G., An Algorithm to Generate the Basis of Solutions *to* Homogenous **Linear** *Diophantine* Equations. Information Processing Letters **7,3 (1978),** 144-147.

86. Huef G. and Lang B., Proving and Applying Program *Transformations* Expressed *With* 2nd Order Patterns. **Acta Informatica** 11 **(1978),** 31-55.

87. Huet G. and **Lankford** D.S., On *the* Uniform Halting Problem *for* Term Rewriting Systems. Rapport Laboria 283, **IRIA,** Mars 1978.

88. Huet G. and **Lévy** J.J., *Call* by Need Computations in Non-AmbiguousLinear Term Rewriting Systems. Rapport Laboria 359, **IRIA, Août** 1979.

89. Hullot J.M., *Associative-Commutative* Pattern Matching. Fifth International Joint Conference on Artificial Intelligence, Tokyo, **1979.**

90. **Hullot** J.M., Canonical Forms and Unification. Unpublished manuscript, 1980.

91. Iturriaga R., Contributions *to* Mechanical Mathematics. Ph. D. thesis, Carnegie-Mellon University, 1967.

92. Jeanrond H.J., A Unique Termination Theorem *for* a Theory with *Generalised* Commutative Axioms. Unpublished manuscript, July 1979.

93. Kahn G. and **Plotkin** G., *Domaines concrets.* Rapport Laboria 336, IRIA, **Décembre** 1978.

94. Kamin S., Some Definitions for *Algebraic* Data Type Specifications. SIGPLAN Notices, Vol. 14, No. 3, March 1979.

95. Khachian, L., *Polynomial* Algorithm for Linear *Programming* Computing Center, Academy Sciences USSR, Moscow, 4 October 1978.

96. Kleene S.C., Introduction to Metamathematics. North-Holland, 1952.

97. Klop J.W., A Counter Example to the Church-Rosser Property for *Lambda*-Calculus With *Surjective* Pairing. Preprint 102, Dept. of Mathematics, U. of Utrecht, 1978.

98. Knuth D. and Bendix P., *Simple* **Word** Problems *in* Universal Algebras. "Comp **utational** Problems in Abstract Algebra". Ed. Leech J., Pergamon Press, 1970, '263-297.

99. Knuth D.E., Morris J. and Pratt V., Fast Pattern Matching *in* Strings. SIAM Journal on Computing, **6,2 (1977),** 323-350.

100. **König** J., *Einleitung* in die *Allgemeine Theorie* der *Algebraischen* Groszen, B. G. Teubner, Leipzig, 1903.

101. Kowalski R.A., Predicate Logic as Programming Language. **Proc.** IFIP 74, North Holland **(1974),** 569-574.

102. Kozen D., Complexity *of Finitely Presented* Algebras. Ninth ACM Symposium on Theory of Computing, May 1977, 164177.

103. Kozen D., *Finitely* Presented **Algebras** and *the Polynomial* Time *Hierarchy.* Report 77-303, Dept. of Computer Science, Cornell U., March 1977.

104. Kruskal J. B., *Well-quasi-ordering,* the Tree *Theorem* and *Vazsonyi's* Conjecture. Trans. Amer. Math. Soc. 95 (1960), 210-225.

105. **Lankford** D.S., *Canonical* Algebraic Simplification. Report ATP-25, Departments of Mathematics and Computer Sciences, University of Texas at Austin, May 1975.

106. **Lankford** D.S., Canonical *Inference.* Report ATP-32, Departments of Mathematics and Computer Sciences, University of Texas at Austin, Dec. 1975.

107. **Lankford** D.S., A Finite Termination Algorithm. Internal memo, Southwestern U., Georgetown, Texas, March 1976.

108. **Lankford** D.S., On *Deciding Word Problems by Rewrite Rules* Simplifiers. Unpublished Manuscript, Sept. 77.

109. **Lankford** D.S., A Unification Algorithm *for Abelian Group Theory.* Report MTP-1, Math. Dept., Louisiana Tech. U., Jan. 1979.

110. **Lankford** D.S., *Mechanical Theorem Proving in* Field Theory. Report MTP-2, Math. Dept., Louisiana Tech U., Jan. 1.979.

111. **Lankford** D.S., *On Proving Term Rewriting* Systems *are* **nœtherian.** Report MTP-3, Math. Dept., Louisiana Tech U., May 1979.

112. **Lankford** D.S. *Some* New Approaches to the *Theory* and Applications *of* Conditional Term Rewriting Systems. Report MTP-6, Math. Dept., Lousiana Tech. U., Aug. 1979.

113. **Lankford** D.S. and Ballantyne A.M., Decision Procedures *for Simple Equational Theories* With *Commutative Axioms: Complete Sets* of *Commutative Reductions.* Report ATP-35, Departments of Mathematics and Computer Sciences, U. of Texas at Austin, March 1977.

114. **Lankford** D.S. and Ballantyne A.M., Decision Procedures for *Simple Equational Theories With Permutative Axioms: Complete Sets of Permutative Reductions.* Report ATP-37, Departments of Mathematics and Computer Sciences, U. of Texas at Austin, April 1977.

115. **Lankford** D.S. and Ballantyne A.M., *Decision Procedures for Simple Equational Theories* With Commutative-Associative *Axioms: Complete* Sets of *Commutative-Associative* Reductions. Report ATP-39, Departments of Mathematics and Computer Sciences, U. of Texas at Austin, Aug. 1977.

116. Lankford D.S. and Ballantyne A.M., *The* Refutation Completeness of *Blocked* Permutative Narrowing and *Resolution.* Fourth Conference on Automated Deduction, Austin, Feb. 1979, 53-59.

117. Lankford D.S. and Musser D., On Semi-Deciding First Order *Validity* and Invalidity. Unpublished Manuscript, March 1978.

118. Lescanne P., Etude *algébrique et relationnelle* des types *abstraits et de leur* representation. Thèse d'Etat, U. de Nancy, Sept. 1979.

119. Lévy J.J., *Réductions correctes* et *optimales* dans *le lambda-calcul.* Thèse d'Etat, U. de Paris VII, Jan. 1978.

120. Lipton R. and Snyder L., *On the* Halting of *Tree* Replacement Systems. Conference on Theoretical Computer Science, U. of Waterloo, Aug. 1977, 43-46.

121. Lipton R. J. and Zalcstein Y., Word *Problems* Solvable in Logspace. Technical Report no. 48, Department of Computer Science, SUNY at StonyBrook, 1975.

122. Livesey M. and Siekmann J., Unification of Sets. Internal Report 3/76, Institut fur Informatik I, U. Karlsruhe, 1977.

123. Majster M., *Limits* of the Algebraic Specification of Abstract Data *Types.* SIGPLAN Notices, vol. 12, Oct. 1977.

124. Makanin G.S., *The Problem* of *Solvability* of *Equations in* a Free Semigroup. Akad. Nauk. SSSR, TOM 233,2 (1977).

125. Malcev A. I., On Homomorphisms of Finite Groups. Ivano Gosudarstvenni Pedagogicheski Institut Uchenye Zapiski, vol. 18, 1958, pp. 49-60.

126. Manna *Z.* and Ness S., On *the* Termination of *Markov Algorithms.* Third Hawaii International Conference on System Sciences, Jan. 1970, 789-792.

127. Martelli A. and Montanari U., *An Efficient* Unification *Algorithm.* Unpublished manuscript, 1979.

128. Matiyasevich Y., *Diophantine Representation of Recursively Enumerable Pred*icates. Proceedings of the Second Scandinavian Logic Symposium, North-Holland, 1970.

129. McCarthy J., *Recursive Fuctions of Symbolic Expressions* and Their *Computa*tions by *Machine,* Part *I.* CACM 3,4 (1960), 184-195.

130. McCarthy J., *A* Basis *For* a Mathematical *Theory* of Computation. Computer Programming and Formal Systems, Eds. Brafford P. and Hirschberg, North-Holland (1963), 33-70.

131. McNulty G., The Decision Problem for Equational Bases of Algebras. Annals of Mathematical Logic, 11 (1976), 193-259.

132. Meyer A. R. and Stockmeyer L., Word Problems Requiring Exponential Time. Fifth ACM Symposium on Theory of Computing, April 1973, 1-9.

133. Moses J., Algebraic *Simplification,* a Guide For the Perplexed. The Macsyma Papers, 1970, 32-54.

134. Musser D. L., A Data Type Verification System Based on Rewrite *Rules.* 6th Texas Conf. on Computing Systems, Austin, Nov. 1978.

135. Musser D. L., Convergent Sets of Rewrite Rules for Abstract Data Types. Unpublished Manuscript, Information Sciences Institute, Jan. 1979.

136. Musser D. L., On Proving Inductive Properties of Abstract Data Types. Seventh ACM Symposium on Principles of Programming Languages, Jan. 1980.

137. Musser D. L., Abstract Data Type Specification in the *AFFIRM* system. To appear, IEEE Transactions on Software Engineering.

138. Nash-Williams C. St. J. A., On Well-quasi-ordering Finite Trees. Proc. Cambridge Phil. Soc. 59 (1963), 833-835.

139. Nelson C. G. and Oppen D. C., Fast Decision *Algorithms Based* on Congruence *Closure.* Stanford CS Report No. STAN-CS-77-646, 1977. To appear JACM.

140. Nelson C. G. and Oppen D. C., Simplification by Cooperating Decision Procedures. ACM Transactions on Programming Languages and Systems 1,2 (1979), 245-257.

141. Nevins A., A Human Oriented Logic For Automatic Theorem Proving. JACM 21,4 (1974), 606–621.

142. Newman M.H.A., On Theories With a Combinatorial Definition of "Equivalence". Annals of Math. 43,2 (1942), 223-243.

143. Nivat M., Congruences *parfaites* et *quasi-parfaites.* Séminaire Dubreuil, 7, 1971-72. (Preliminary Version in Proc. Second Annual ACM Symposium on Theory of Computing, 1970, 221-225.)

144. Nivat M., On the Interpretation of *Recursive Polyadic* Program Schemes. Symposia Mathematica Vol. XV, Istituto Nazionale di Alta Matematica, Italy, 1975, 225-281.

145. O'Donnell M., Computing in Systems Described by Equations. Lecture Notes in Computer Science 58, Springer Verlag, 1977.

140. Oppen D. C., Reasoning About Recursively Defined Data Structures. Fifth ACM Symposium on Principles of Programming Languages, January 1978. To appear JACM.

147. Oppen D. C., *Convexity, Complexity,* and Combinations of Theories. Fourth Symposium on Automated Deduction, Austin, 1979. To appear Theoretical Computer Science.

148. Paterson M.S. and Wegman M.N., Linear Unification. J. of Computer and Systems Sciences 16 (1978), 158-167.

149. Peterson G.E. and Stickel M.E., Complete *Sets* of Reductions *for* Equational Theories With Complete Unification Algorithms. Tech. Report, Dept. of Computer Science, U. of Arizona, Tucson, Sept. 1977.

150. Plaisted D., Well-Founded Orderings *for Proving* Termination of Systems of *Rewrite* Rules. Dept. of Computer Science Report 78-932, U. of Illinois at Urbana- Champaign, July 1978.

151. Plaisted D., A *Recursively* Defined Ordering for *Proving* Termination of Term Rewriting Systems. Dept. of Computer Science Report 78-943, U. of Illinois at Urbana-Champaign, Sept. 1978.

152. Plotkin G., Lattice-Theoretic Properties of Subsumption. Memo MIP-R-77, U. of Edinburgh, 1970.

153. Plotkin G., Building-in Equational Theories. Machine Intelligence 7 (1972), 73-90.

154. Post E., Recursive Unsolvability of a Problem of Thue. J. Symbolic Logic 12(1947), l-11.

155. Presburger M., *Uber* die *Vollstandigkeit* eines gewissen Systems der *Arithmetik ganzer Zahlen,* in *welchem* die Addition *als einzige* Operation hervortritt. Comptes-Rendus du ler Congrès des Mathématiciens des Pays Slaves, 1929.

156. Raoult J.C. and Vuillemin J., Operational and Semantic Equivalence Between Recursive Programs. 10th ACM Symposium on Theory of Computing, San Diego, 1978.

157. Raulefs P. and Siekmann J., Unification *of Idempotent* Functions. Unpublished manuscript, 1978.

158. Reynolds J., Transformational *Systems* and the Algebraic *Structure* of Atomic Formulas. Machine Intelligence 5 (1970), 135-152.

159. Robinson G. A. and Wos L. T., *Paramodulation* and Theorem *Proving* in First-order Theories with Equality. Machine Intelligence 4, American Elsevier, 1969, 135-150.

160. Robinson J.A., A *Machine-Oriented* Logic Based on the *Resolution* Principle. JACM 12 (1965), 32-41.

161. Rosen B.K., *Subtree* Replacement Systems. PhD Thesis, Harvard U., 1971.

162. Rosen B.K., Tree-Manipulation Systems and Church-Rosser Theorems. JACM 20 (1973), 160–187.

163. Scott D., Outline of a Mathematical Theory of Computation. Monograph PRG-2, Oxford U. Press, 1970.

164. Seidenberg A., A *New Decision* Method *For Elementary* Algebraand Geometry. Ann. of Math., Ser. 2, 60 (1954), 365-374.

165. Sethi R., Testing for the Church-Rosser Property. JACM 21 (1974), 671-679. Erratum JACM 22 (1975), 424.

166. Shostak R., An Algorithm for Reasoning about Equality. CACM, 583-585, July 1978.

167. Siekmann J., Unification of Commutative Terms. Unpublished manuscript, 1978.

168. Siekmann J., Unification and Matching Problems. Ph. D. thesis, Memo CSM-478, University of Essex, 1978.

169. Slagle J.R., Automated Theorem-Proving for Theories with Simplifiers, *Commutativity* and Associativity. JACM 21 (1974), 622-642.

170 Staples J., Church-Rosser Theorems for Replacement Systems. Algebra and Logic, ed. Crossley J., Lecture Notes in Math., Springer Verlag 1975, 291-307.

171. Staples J., A Class of Replacement Systems with Simple *Optimality* Theory. To appear, Bull. of the Australian Math. Soc.

172. Staples J., Computation on Graph-like Expressions. Report 2/ 77, Math. and Computer Science dept., Queensland Institute of Technology, Brisbane, Australia, 1977.

173. Stickel M.E., A Compfete Unification Algorithm for Associative-Commutative Functions. 4th International Joint Conference on Artificial Intelligence, Tbilisi, 1975.

174. Stickel M.E., Unification Algorithms for Artificial *Intelligence* Languages. Ph. D. thesis, Carnegie-Mellon University, 1976.

175. Szabó P., The Undecidability of the D_A-*Unification* Problem. Unpublished manuscript, 1979.

176. Tarski A., A Decision Method for Elementary Algebra and Geometry. U. of California Press, Berkeley, 1951.

177. Tarski A., *Equational Logic.* Contributions to Mathematical Logic, ed. Schiitte et al, North-Holland, 1968.

178. Thatcher J., Wagner E. and Wright J., Data Type Specifications: *Parameterization* and the *Power* of Specification Techniques. Tenth ACM Symposium on Theory of Computing, May 1978.

179. Van Emden M.H. and Kowalski R.A., *The* Semantics of Predicate Logic as a Programming Language. JACM 23,4(1976), 733-742.

180. Vuillemin J., Correct and Optimal Implementation of Recursion in a Simple Programming Language. J. of Computer and System Sciences 9,3 (1974),332-354.

181. Wadsworth C.P., Semantics and *Pragmatics* of the X-calculus. PhD Thesis, Oxford U., 1971.

182. Wand M., First Order Identities as a Defining Language. Indiana U. Tech. Report 29, 1976.

183. Wand M., Final Algebra Semantics and Data Type Extensions. JCSS, vol. 19, no. 1, Aug. 1979.

184. Winker S., Dynamic *Demodulation.* Internal memo, Dept. of Computer Science, Northern Illinois U., 1975.

185. Zilles S., Data Algebra: A Specification Technique *for* Data Structures. Ph. D. thesis, MIT, 1978.