J. Symbolic Computation (1989) 7, 207–274

Unification Theory

JÖRG H. SIEKMANN
Universität Kaiserslautern, FB Informatik
Postfach 3049
D-6750 Kaiserslautern
WEST GERMANY

ABSTRACT: Most knowledge based systems in artificial intelligence (AI), with a commitment to a symbolic representation, support one basic operation: "matching of descriptions". This operation, called unification in work on deduction, is the "addition-and-multiplication" of AI-systems and is consequently often supported by special purpose hardware or by a fast instruction set on most AI-machines. Unification theory provides the formal framework for investigations into the properties of this operation. This article surveys what is presently known in unification theory and records its early history.

1. Introduction

Überhaupt hat der Fortschritt das an sich, daß er viel größer ausschaut, als er wirklich ist.

J.N.Nestroy, 1859

Not least because of its numerous applications in artificial intelligence (AI) and computer science, the field of ***unification theory*** is currently witnessing intense activity. This field is concerned with problems of the following kind: Let f and g be binary functions, a and b constants, and x and y variables, and consider the two ***first order terms*** s and t built from these symbols as follows:

$$s = f(x\ g(a\ b)) \qquad t = f(g(y\ b)\ x)$$

The decision problem is whether or not there exist terms which can be substituted for the variables x and y in s and t so that the two terms thus obtained are identical. In this example g(a b) and a are two such terms and we write

$$\delta = \{x \leftarrow g(a\ b), y \leftarrow a\}$$

0747–7171/89/030207+68 $03.00/0

to represent this unifying substitution.
We say that δ is a ***unifier*** for s and t, since $\delta s = \delta t = f(g(a\ b)\ g(a\ b))$.

In addition to the above *decision problem* there is also the problem of finding a *unification algorithm*, which enumerates the unifiers for a given pair of terms. Such algorithms are at the very heart of present day computing, in fact they form part of the central processing unit of the "fifth generation computers" (ICOT, 1984) (note the difference in reference convention to the bibligraphy in section five and the references in section four: references to section four are preceded by a *). For efficiency reasons they are often implemented in silicon or at least supported by an abstract machine, such as the Warren Abstract Machine (*Gabriel et al., 1984), into whose instruction set the terms to be unified are compiled.
Consider a variation of the above problem, which arises when we assume that f is commutative:

$$\text{(C)} \quad f(x\ y) = f(y\ x)$$

Now δ is still a unifier for s and t. However $\sigma = \{y \leftarrow a\}$ is also a unifier for s and t since

$$\sigma s = f(x\ g(a\ b)) =_C f(g(a\ b)\ x) = \sigma t.$$

But σ is ***more general*** than δ, or put another way δ is an ***instance*** of σ,since it is obtained as the composition of the substitutions $\lambda \circ \sigma$ where $\lambda = \{x \leftarrow g(a\ b)\}$. Hence a unification algorithm only needs to compute σ.

In some cases there is a single and unique least upper bound in the lattice of unifiers, called the ***most general unifier*** or alternatively the principal unifier. For example for every pair of uninterpreted terms as above there is at most *one* general unifying substitution. Under commutativity however, there are pairs of terms which have more than one most general unifier, but they always have at most *finitely* many.

The problem becomes entirely different, when we assume that the function f is associative:

$$\text{(A)} \quad f(x\ f(y\ z)) = f(f(x\ y)\ z)$$

In this case δ is still a unifying substitution, but $\tau = \{x \leftarrow f(g(a\ b)\ g(a\ b)), y \leftarrow a\}$ is also a unifier, since

$$\tau\ s = f(f(g(a\ b)\ g(a\ b))\ g(a\ b)) =_A f(g(a\ b)\ f(g(a\ b)\ g(a\ b))) = \tau\ t.$$

But $\tau' = \{x \leftarrow f(g(a\ b)\ f(g(a\ b)\ g(a\ b))), y \leftarrow a\}$ is again a unifying substitution and it is not difficult to see, by an iteration of this process, that there are *infinitely* many unifiers, all of which are most general.

Finally, if we assume that both axioms (A) and (C) hold for f, the situation changes yet again and for any pair of terms there are at most *finitely* many most general unifiers under (AC).
The above examples as well as the many practical applications of unification theory (see section 1.1.) share a common problem, which in its most abstract form is as follows: Let $\mathcal{L}$ be a formal language with variables and two words s and t in that language. Then for a given binary relation $\approx$ defined in $\mathcal{L}$ find a substitution σ such that $\sigma s \approx \sigma t$ (provided of course that σs and σt are welldefined).
If the relation $\approx$ can be specified by a set E of equational axioms and if $\mathcal{L}$ is the language of first order terms, unification of s and t in E amounts to solving the equation $s = t$ in the variety defined by E. For example if E consists of the associative axiom and the idempotence axiom $f(x,x) = x$, and s and t are terms,then unification of s and t amounts to solving equations in free idempotent semigroups. A better known example may be the following: if E is an axiomatization of the natural numbers and s and t are appropriate terms, unification of s and t amounts to solving Diophantine equations.
The mathematical investigation of equation solving is a subject as old as mathematics itself and right from the beginning was very much at the heart of it. It dates back to Babylonian mathematics (about 2000 B.C.) and has dominated much of mathematical research ever since. Unification theory carries this activity on in a more abstract setting. Just as universal algebra abstracts from certain properties that pertain to specific algebras and investigates issues that are common to all of them, unification theory addresses problems, which are typical for equation solving as such. And just as traditional equation solving drew much of its impetus from its numerous applications (for example the, for the times, complicated procedure for deviding legacies in Babylonian times or the applications in physics in more modern times), unification theory derives its impetus from its numerous applications in AI and computer science.

Central to unification theory are the notions of a ***set of most general unifiers*** μU and the ***unification hierarchy*** based on the cardinality of μU. Both notions will be formally introduced in section 2, where we define a ***unification problem*** for an equational theory E. However for many practical applications unification is too general a concept, instead it is of interest to know for two given terms s and t if there exists a ***matcher*** μ (a one-side-unifier) such that $\mu(s)$ and t are equal in E. In other words, in a ***matching problem*** we are allowed to substitute into one term only (into s using the above convention) and we say s matches t with matcher μ.

1.1. APPLICATIONS

There is a wide variety of areas in AI and computer science where unification problems arise.

Databases

A deductive database (*Gallaire & Minker, 1978) does not store every fact explicitly. Instead it contains only certain facts, from which other facts can be deduced by some inference rule. Such inference rules (deduction rules) heavily rely on unification algorithms.
The user of a relational database (*Date, 1976) may logically *and* the properties he wants to retrieve or else he may be interested in the *natural join* of two stored relations. But *and* is an associative and commutative operation and the *natural join* obeys an associative axiom, which distributes over some other operation, hence both can be built into a unification algorithm (Snelting & Henhapl, 1985).

Information Retrieval

A patent office may store all known electric circuits (*Bryan & Carnog, 1966) or all recorded chemical compounds (*Sussenguth, 1965) as some graph structure, and the problem of checking whether a given circuit or compound already exists is an instance of a test for graph isomorphism (*Ullman, 1976; *Unger, 1964; *Corneil, 1968). More generally, if the nodes of such graphs are labelled with universally quantified variables ranging over subgraphs, then these problems are instances of a *graph matching problem* (*Rastall, 1969).

Computer Vision

It has become customary in this field to store the internal representation of external scenes as some net structure (*Ballard & Brown, 1982; *Winston, 1975). The problem to find a particular object represented in a given scene, is then also an instance of a graph matching problem (*Ballard & Brown, 1972; *Rastall, 1969). Here one of the main problems is to specify exactly what constitutes a successful match (since a test for endomorphism is too rigid for most applications): matching is carried out with respect to some distance function (or some metric), that is usually not formally defined, but depends on the application in mind.

Natural Language Processing

The processing of natural language by a computer (*Winograd, 1972; *Winograd, 1983; *Tennant, 1981) is often based on transformation rules, which for example translate the surface structure of the input sentence into a more appropriate form for internal representation within the computer.Inference rules are used to derive the semantics of an input sentence and to disambiguate it. The knowledge about the external world that a natural language processing system must have, is represented by some machine oriented descriptions and it is of paramount importance to detect if two descriptions describe the same object or fact.

Transformation rules, inference rules and the matching of descriptions are but a few places where unification theory is applied within this field.
The meaning of a natural language utterance has to be represented in some internal representation language, which in turn should have a well defined semantics. Recently developed formalisms such as situation semantics (for which see (*Barwise & Perry, 1983)) or discourse representation theory (*Kamp, 1981) no longer use elementary set theoretical operations for the manipulation of natural language utterances, but rely on one basic operation, namely unification with respect to certain constraints.
Also special functional grammars have been designed for parsing natural languages, called unification grammars (Shieber, 1986), that depend on one fundamental operation: feature unification (Ait-Kaci, 1984; Smolka & Ait-Kaci, 1987).

Expert Systems

An expert system is a computer program (*Brachmann & Schmolze, 1985), whose performance largely depends on its ability to represent and manipulate the knowledge within its field of expertise. Commonly this knowledge is represented in the form of production rules, such that if the preconditions of a production rule are fulfilled, its action part will be executed. Special languages such as OPS5 (*Forgy, 1981) and others have been developed for the implementation of such systems. In OPS5 the conditional part of a production rule is matched against the entries in the knowledge base and if the match succeeds, the preconditions are considered true and the rule will fire.
The efficiency of this matching process is of crucial importance and special techniques, e.g. the Rete-algorithm (*Forgy, 1982) and even hardware realisations (*Ramnarayan & Zimmermann, 1985), have been proposed which are similar to efficient implementations of the unification algorithm in logic programming languages .

Text Manipulation Languages

The fundamental mode of operation in programming languages like SNOBOL (*Farber et al., 1964) is to detect the occurrence of a substring within a larger string of characters (which may be a program or some other text), and there are methods known for doing this, which require less than linear time (*Boyer & Moore, 1977). If these strings contain the SNOBOL "don't-care"-variable, the occurrence problem is an instance of the string unification problem (Siekmann, 1975).

Planning Systems

Computerbased generation of plans for actions, such as a plan for a robot action or plans for appropriate language generation, is an important subfield of AI. The methods for finding a plan can be viewed as a deduction process. In a recent paper Z.Manna and R.Waldinger show, how a tableau-based inference system with an extended unification algorithm (for additional equations and equivalences) can be used to generate such plans (*Manna & Waldinger, 1986).

Pattern Directed Programming Languages

An important contribution to programming language design is the mechanism of pattern-directed invocation of procedures (*Böhm et al. 1977; *Hewitt, 1972; *Rulifson et al., 1972; *Beilken et al., 1982). Procedures are identified by patterns, instead of procedure identifiers as in traditional programming languages, and these patterns usually express goals to be achieved by executing the procedure. Incoming messages are tested for matching against the invocation patterns of procedures in a procedural data base, and a procedure is activated after a successful match between message and pattern is achieved. Here matching is carried out to find an appropriate procedure that helps to accomplish an intended goal and also for transmitting information to the invoked procedure. For these applications (often called demons, censors, agents, etc.) it is particularly desirable to have methods for the description and matching of objects in high level data structures such as strings, sets, multisets, lists and others.

A little reflection will show that for very expressive matching languages, as e.g. MATCHLESS in PLANNER (*Hewitt, 1972), the matching problem is undecidable. This presents a problem for the designer of such languages: on the one hand, very rich and expressive languages are desirable, since they form the basis for the invocation and deduction mechanism. On the other hand, drastic restrictions will be necessary, if matching algorithms are to be found. The question is just how severe do these restrictions have to be.

Knowledge Representation Languages

Based on frame-like techniques to structure and represent knowledge (*Minsky, 1975), special purpose programming languages such as KRL (*Bobrow & Winograd 1977) or KL-ONE (*Brachman & Schmolze, 1985) have been designed for this task. Apart from their respective commitment to the representation and structuring issue, they all support one central operation: "matching of descriptions" (*Brachmann & Levesque, 1985). In a sense unification theory relates to these new kinds of programming languages - and hence to knowledge based systems - as formal language theory relates to traditional programming languages.

Logic Programming Languages

The discovery of the close relationship between logical deduction and computation, which means that logic enjoys a role in computer science analogous to that of analysis in physics, is certainly one of the outstanding scientific achievements of the later part of this century.

However there is a more specific point to this, namely that predicate logic itself can be viewed as a programming language (*Kowalski, 1979) given a suitable machine to execute it. Predicate logic relates to a deduction system as for example LISP relates to EVAL. This insight opened up a new technological race for logic programming languages and appropriate machines on which to execute them. The Japanese coined the name "fifth generation computers" for such machines. The central computation performed in logic programming machinery is unification. In fact the unification algorithm - be it implemented in software or in silicon - is the central processing unit, the CPU, of these machines. Hence the speed of these machines is no longer expressed in MIPS (Millions of

Instructions Per Second) as for conventional machines, but in KLIPS (thousands of Logical Inferences Per Second), which is in effect a measure of the number of unifications performed per second.

Term Rewriting Systems

The manipulation of terms in equationally defined theories, traditionally called demodulation (*Wos et al., 1967), is based on matching and has always played an important role in deduction systems. If in addition the equations can be transformed into a confluent and finitely terminating rewriting system (*Huet & Oppen, 1980), they can be used to compute a unique normal form for any term. The test for confluence can be carried out by a procedure known as the Knuth-Bendix completion procedure (Knuth & Bendix, 1979), which uses a unification algorithm as its central component.

Certain equational axioms, such as associativity or commutativity, are notoriously difficult to handle using these systems. Therefore a given equational theory T can sometimes be separated into two constituent parts, $T = R \cup E$, such that only R needs to be transformed into a canonical rewriting system and E (the difficult equations) can be built into a special purpose unification algorithms (Peterson & Stickel, 1981).

Term rewriting systems are of considerable interest in computer science (Buchberger, 1987) and have now found a place in most computer science curricula, since they provide for a convenient computational treatment of equational logics. Not the least important among the many applications these systems have, is their foundational role in new programming languages which elegantly combine functional with logic programming. Term rewriting systems that operate on the word monoid are called Semi-Thue-Systems, for a survey see R.Book (*Book, 1985).

Computer Algebra

In computer algebra, matching and unification algorithms also play an important role. For example the integrand in a symbolic integration problem (*Moses, 1971) may be matched against certain patterns in order to detect the class of integration problems to which it belongs. A succesful match then triggers the appropriate action for its solution (which in turn may involve several quite complicated matching attempts (*Blair et al., 1971; *Fateman, 1971). Hence most computer algebra systems like REDUCE (Hearn, 1971), MACSYMA (Moses, 1974) or MATHLAB (*Manove et al., 1968) make extensive use of unification or matching algorithms.

Algebra

A famous decidability problem, which inspite of many attacks remained open for over twenty years, has been solved. The Monoid Problem (also called Löb's Problem in western countries, Markov's Problem in eastern countries and the String Unification Problem in the field of automated deduction (Hmelevskij, 1964; Hmelevskij, 1966; Hmelevskij, 1967; *Markov, 1954; Plotkin, 1972; Siekmann, 1975; Livesey & Siekmann, 1975), is the problem of deciding whether or not an equational system over a free semigroup possesses a solution. This problem has been

shown to be decidable (Makanin, 1977). The Monoid Problem has important practical applications inter alia for deduction systems (string unification (Siekmann, 1975) and second order monadic unification (Huet, 1976; Winterstein, 1976)), for formal language theory (the crossreference problem for van Wijngaarden Grammars (*van Wijngaarden, 1976) and for pattern directed invocation languages in AI as mentioned above.

Without surveying classical equation solving as such, one "unification problem" that should be mentioned is Hilbert's Tenth Problem (Davis, 1973), which is known to be undecidable (Matiyasevich, 1970). The problem is whether or not a given polynomial $P[x_1,x_2,...,x_n] = 0$ has an integer solution (a Diophantine solution). Although this problem was posed originally within the framework of traditional equation solving, unification theory has shed new light upon this problem (Siekmann & Szabo, 1986).

Semigroup theory (*Clifford & Preston, 1961; *Howie, 1976) is the field that traditionally poses the most important unification problems, i.e. those involving associativity. Although more established than unification theory is today, some interesting semigroup problems have been solved using techniques from unification theory and term rewriting systems (see e.g. (*Siekmann & Szabo, 1982; Lankford, 1980; Lankford, 1979; Baader, 1987)).

Deduction Systems

All present day deduction systems - whether they are based on resolution (Robinson, 1965) or not - have a unification algorithm for first or higher order terms as their essential component: it is the "addition and multiplication of deduction work".

For almost as long as attempts at proving theorems by machines have been made, it has been well known that certain equational axioms, if left unconstrained in the data base of a deduction system, may force it to go astray. In 1967 J.A.Robinson proposed that substantial progress ("a new plateau") could be achieved, by removing these troublesome axioms from the data base and building them directly into the inference rules of the deductive machinery. One technique that has become important for deduction systems, is to build these axioms, which often define common data structures, into the unification algorithm itself. G.Plotkin has shown in a pioneering paper (Plotkin, 1972), that a deduction system is refutation complete, whenever its extended unification procedure generates a set of unifiers satisfying the three conditions of completeness, correctness and minimality. These properties are now used to axiomatically define the set of most general unifiers.

Nonclassical Logics

Knowledge representation systems in AI are often based on nonclassical logics that model temporal information, modality, probability or beliefs more adequately than ordinary first order logic (*Brachmann & Levesque, 1985). As it turned out, the nonclassical aspect of a logic can often be accounted for using special terms and the mechanization of such logics amounts to finding appropriate unification algorithms. For example various forms of modal and temporal logics have been coded this way (*Wallen, 1987; *Nonnengart, 1987; *Ohlbach, 1987) and particular

unification algorithms for some standard modal logics (like S4,T etc) are reported in (Ohlbach, 1988).

It is the field of automated deduction systems (the series of Conferences of Automated Deduction, CADE), where unification problems first became of general importance and that has historically contributed most to unification theory.

1.2. EARLY HISTORY

Allowing for some exceptions we take 1976 as the (not entirely arbitrary) date before which work is considered early history, whereas later contributions are recorded under the heading 'Results' in sections 3.1. and 3.2.

The visionary thoughts about the nature of mathematics, symbols and human reasoning that Emil Post recorded in his diary and notes (partially published in (*Davis, 1965)) contain the first hint as early as the 1920s of the concept of a unification algorithm that computes a most general representative as opposed to all possible instantiations (p.370 in (*Davis, 1965)).

The first explicit account of a unification algorithm was given in J. Herbrand's thesis "Recherches sur la theorie de la demonstration" in 1930 (Herbrand, 1930), where he introduced three concepts with respect to the validity of formulas. He called them A, B and C. Concept B and C were the basis for the wellknown Herbrand Theorem, whereas concept A was by and large consigned to oblivion. In order to calculate that property A holds for a formula, he gave an algorithm which computes it. This was the first published unification algorithm and was based on a technique later rediscovered by A.Martelli and U.Montanari (Martelli & Montanari, 1976) and that is still much in use today.

Based on Herbrand's idea of a finite counterexample, i.e. that only a finite number of instantiations are necessary in order to show the unsatisfiability of a set of formulas, early theorem proving programs were developed, but it was not until 1960 when D.Prawitz (*Prawitz, 1960) suggested a way out of these "British Museum Techniques" as they were called later on, which was to compute a most general representative for the abundant number of instantiations that are possible otherwise. However, as his logic did not contain any function symbols, there was little in fact to compute. In 1963 M.Davis published (*Davis, 1963) a proof procedure that combined the virtues of Prawitz's procedure with those of the Davis-Putnam procedure. The implementation of this new proof procedure on an IBM 7090 at Bell Telephone Laboratories November 1962 used a unification algorithm to compute the "matings" and appears to be the first fully implemented unification algorithm in actual use.

It was not until 1965, however, when the seminal paper on the resolution principle by J.A.Robinson was published (Robinson, 1965), that a formal account of a unification algorithm

for first order terms, which computes a unique, single representative (i.e. the most general unifier) first appeared in print. This has been the most influential paper in this field and firmly established the concept of unification in automated deduction systems (including systems not based on resolution).

The work for this paper was done essentially in 1963 at Argonne National Laboratory, a time when another group headed by J.R. Guard at the Air Force Cambridge Lab developed a deduction system based on a Gentzen-style sequent logic that also incorporated a unification algorithm. The work was published in some internal reports (*Guard, 1964) and later in (*Guard & Oglesby, 1969). However, although their algorithm was correct and complete, this was not proved. They also suggested extensions of the algorithm to higher order logic as well as first order extensions to incorporate axioms like commutativity and associativity. The algorithms used for these latter extensions were heuristically motivated (reordering of terms, rebracketing etc.) and were incorrect and incomplete in general.

The basic unification algorithm was discovered again by D.Knuth and published in a paper (Knuth & Bendix, 1970) that became a classic in the field of term rewriting systems. In order to turn a given set of equations into a canonical rewriting system, a completion process is described that depends heavily on a unification algorithm, whose theoretical properties (computation of the most general unifier) were recognised and demonstrated.

In 1967 J.A.Robinson proposed to build certain troublesome axioms directly into the deductive machinery of an automated theorem prover and in 1972 G.Plotkin (Plotkin, 1972) showed how this can be done without losing completeness. From the point of view of unification theory this paper contained two major contributions: first the definition of a set of most general unifiers, which became (in particular through the work of G.Huet) a central notion of the field, and second the discovery that there are equational theories (e.g. the associativity axiom) which induce an infinite set of most general unifiers.

M.Stickel presented special unification algorithms for associativity, commutativity and their combination in his thesis (Stickel, 1975; Stickel 1977), this work was essentially motivated by the matching problem in pattern invocated programming languages as already descibed above.

The work of G.Plotkin was taken up in my own thesis (Siekmann, 1978), which described several unification algorithms for the axioms of associativity, commutativity and idempotence and their combinations. This thesis also suggested that unification theory, at that stage a collection of special purpose algorithms, was worthy of study as a field in its own right and as an important branch of theoretical AI, centering around the unification hierarchy, a concept which was first introduced here along with some preliminary results concerning it.

While these developments were taking place in first order unification theory, there was also important work going on in higher order unification around the same time. Based on the theorem proving system of J.R.Guard and his associates mentioned above, W.F.Gould (Gould, 1966) investigated the most general common instance of two higher order terms and discovered that there are infinitely ascending chains of most general unifiers (i.e. a minimal set of most general

unifiers does not exist for ω-order logics).
Influenced by P.Andrews, whose work was seminal for higher order deduction systems (Andrews, 1971), G.P.Huet developed a "constrained resolution method" (Huet, 1972) for higher order theorem proving, based on an ω-order unification algorithm. This work was then further developed in his "thèse d'état" in 1976 (Huet, 1976), which became of fundamental importance in shaping the field of first and higher order unification theory as it is known today.

2. Notions and Notation

Unification Theory rests upon the notational conventions of Universal Algebra (see e.g. *Grätzer, 1979; *Burris, 1981) and of Computational Logic (see e.g. (*Loveland, 1978; *Huet & Oppen, 1980; *Buchberger, 1987), which we shall briefly review in the following paragraphs.
Given a set S with elements σ, δ, τ, ... and a ***quasi ordering*** $\leq$ on S, we say that two elements $\sigma,\tau \in S$ are ***equivalent***, $\sigma \equiv \tau$, iff $\sigma \leq \tau$ and $\tau \leq \sigma$. A subset $U \subseteq S$ is called an (upper) ***segment*** or a ***filter*** of S, iff for $\sigma \in S$ and $\tau \in U$ and $\tau \leq \sigma$ we have $\sigma \in U$. We say U is generated by a set cU iff U consists exactly of those elements of S that are greater than some elements of cU, i.e. $cU \subseteq U$ and $\forall \tau \in U$ there exists $\sigma \in cU$ with $\sigma \leq \tau$. A minimal generating set μU is called a ***base*** of U or the ***μ-set*** of U if it is a generating set for U with the following additional property: $\forall\ \sigma,\tau \in \mu U$: $\sigma \leq \tau$ implies $\sigma = \tau$. A segment does not necessarily have a base, but if the bases exist they are all equivalent.
We are interested in the existence, uniqueness and cardinality of such μ-sets in the more specific context of unification .

2.1. COMPUTATIONAL LOGIC

Our starting point is the familiar concept of an ***algebra*** as a pair $(\mathbb{A}, \mathbb{F})$, where $\mathbb{A}$ is the ***carrier*** and $\mathbb{F}$ is a family of ***operators*** (the ***signatur***) given with their arities.
For $\mathbb{F}$ and a denumerable set of variables $\mathbb{V}$, we define $\mathbb{T}$, the set of first order terms, over $\mathbb{F}$ and $\mathbb{V}$, as the least set with (i) $\mathbb{V} \subseteq \mathbb{T}$, and if arity(f) = 0 for $f \in \mathbb{F}$ then $f \in \mathbb{T}$ and
(ii) if $t_1, ..., t_n \in \mathbb{T}$ and arity(f) = n then $f(t_1 ... t_n) \in \mathbb{T}$.
Let $\mathbb{V}(t)$ be the variables occurring in term t, a term t is ***ground*** if $\mathbb{V}(t) = \emptyset$. The algebra with carrier $\mathbb{T}$ and with operators corresponding to the term constructors of $\mathbb{F}$ is the absolutely free (term) algebra, i.e. it just gives an algebraic structure to $\mathbb{T}$. If the carrier is the set of ground terms it is called the initial algebra (*Goguen & Thatcher, 1977) or Herbrand Universe (*Loveland, 1978).
A ***substitution*** $\sigma: \mathbb{T} \rightarrow \mathbb{T}$ is an endomorphism on the term algebra $\mathbb{T}$, which is identical almost everywhere on $\mathbb{V}$ and hence can be represented as a finite set of variable-term pairs:

$$\sigma = \{x_1 \leftarrow t_1, ..., x_n \leftarrow t_n\}.$$

Equational Logic

Although unification theory is not restricted to equationally defined theories, most results have been obtained within this frame.

An ***equation*** is a pair of terms, usually written as $s = t$. Given a set of equations E and a single equation $s = t$, we denote by $E \models s = t$ that $s = t$ is true in every model of E ($s = t$ is a modeltheoretical consequence of E). An ***equational theory*** T is a set of equations with $T \models s=t$ iff $(s = t) \in T$, i.e. T consists of all its consequences. For a given set of equations E, the least equational theory T(E) is the finest congruence on the term algebra containing all pairs $\sigma s = \sigma t$, for all equations in E and all substitutions σ (the substitution invariant congruence generated by E). We say s and t are ***E-equal,*** abbreviated as $s =_E t$, iff the terms s and t are in this congruence. E is a ***presentation*** of the congruence $=_E$ or an ***axiomatization*** of the equational theory T(E). Usually we say 'theory E' and mean the equational theory T(E) axiomatized by E.

Obviously the axiomatization for an equational theory is not unique. A theory that has a finite axiomatization is called ***finitely generated***, otherwise it is infinitely generated. E-equality is not decidable in general; however in unification theory we are usually only interested in equational theories with a decidable word problem. Another natural restriction is that we consider only consistent theories, i.e. theories, which do not collapse into a single equivalence class. A theory is ***consistent*** if for all $v,w \in V$: $v =_E w$ implies $v = w$.

A standard set of inference rules for equational logic is the following:

$s = s$

if $s = t$ then $t = s$

if $r = s$ and $s = t$ then $r = t$

if $s_i = t_i$, $1 \le i \le n$, then $f(s_1,s_2, ...,s_n) = f(t_1,t_2, ... ,t_n)$

if $s = t$ then $\sigma s = \sigma t$ for all substitutions σ.

An equation $s = t$ can be *derived* or *proved* from an axiomatization E, $E \vdash s = t$, if it can be obtained in finitely many steps from E using the above rules. G.Birkhoff gave the first completeness proof for this derivation system (*Birkhoff, 1935):

Theorem : $E \vdash s = t$ iff $E \models s = t$

For a survey on classical equational logic see e.g.(*Tarski, 1968; *Taylor, 1979); sequences of replacement are used in (*McNulty, 1976).

Term Rewriting Systems

Since neither $\vdash$ nor $\models$ are particularly convenient for a computational treatment of $=_E$, two computer oriented techniques for equational axioms called paramodulation (*Wos & Robinson, 1973) and demodulation (*Wos & Robinson, 1967) are extensively used in the field of automated deduction. Suppose the equational theory is actually presented as

$E=\{l_1=r_1, l_2=r_2,..., l_n=r_n\}$, with the assumption that the r_i are in some sense smaller than the l_i. A term s is said to be ***demodulated*** to t, if there is a subterm s´ in s and a pair $l_i = r_i$ in E such that $s' = \mu l_i$ for some substitution μ and term t is obtained from s by replacement of s´ by μr_i.
A term s is said to be ***paramodulated*** to t, if there is a subterm s´ in s and a pair $l_i = r_i$ in E such that $\sigma l_i = \sigma s'$ for a substitution σ; term t is obtained from σs by replacement of $\sigma s'$ by σr_i. Note that this is only a special case of paramodulation, in the context of full predicate logic a little extra machinery is required (*Loveland, 1978). The problem is of course how to find a presentation, such that the righthand side of the equations is smaller than the lefthand side. This problem has been addressed in a paper by D.Knuth (Knuth & Bendix, 1970), which is now a classic in this field. The essential observation is that it is often possible for a given set of equations E to find an equivalent set in the sense of the definition below, which is directed from left to right $R_E = \{l_1 \Rightarrow r_1, l_2 \Rightarrow r_2, ..., l_n \Rightarrow r_n\}$ with $Var(r_i) \subseteq Var(l_i)$ such that the r_i are smaller than the l_i. This is called a ***term rewriting system*** (TRS). A TRS can be used to define a reduction relation on terms by: $s \mapsto_R t$ if s can be demodulated to t using R. If there are no infinite sequences $s_1 \mapsto s_2 \mapsto ...$ the relation $\mapsto_R$ is said to be terminating or ***Noetherian***. The relation $\mapsto_R$ is called ***confluent*** if for every r, s, t with $r \mapsto_R s$ and $r \mapsto_R t$ there exists a term u such that $s \overset{*}{\mapsto} u$ and $t \overset{*}{\mapsto} u$. A confluent, Noetherian relation is called ***canonical*** ; similarily a TRS is called canonical, if the relation it is based upon is canonical. Canonical TRS´s are an important basis for a computational treatment of equational logic, since they define a unique normal form $||t||$ for every term t given by: $t \overset{*}{\mapsto} ||t||$ and there does not exist a term s with $||t|| \mapsto s$. $||t||$ exists because of the finite termination property and it is unique because of confluence. The TRS R_E is ***equivalent*** to E if: $s =_E t$ iff $||s|| = ||t||$. A terminating term rewriting system can sometimes be completed to a canonical system with the Knuth-Bendix completion procedure (Knuth & Bendix, 1970). Because of the great importance of TRS for computer science, there is intensive research now on methods of how to obtain a canonical TRS from a given set of equations (see (*Huet & Oppen, 1980; *Buchberger, 1987) for two classical surveys). Rewrite systems in the word monoid are known as Semi-Thue Systems (*Book, 1985).

Similar to the above rewrite relation $\mapsto_R$, we can define a relation $\nmapsto_R$, often called ***narrowing*** (Hullot, 1980) , such that $s \nmapsto_R t$ holds, if s can be paramodulated to t. This relation is of particular importance for universal unification algorithms (see section 3.2.2.).

2.2. UNIFICATION THEORY

A ***substitution*** σ: $\mathbb{T} \rightarrow \mathbb{T}$ is an endomorphism on the term algebra $\mathbb{T}$, which is identical almost everywhere on V and hence can be represented as a finite set of pairs $\sigma = \{x_1 \leftarrow t_1,..., x_n \leftarrow t_n\}$. The restriction $\sigma|_V$ of a substitution to a set of variables V is defined as $\sigma|_V x = \sigma x$ if $x \in V$ and $\sigma|_V x = x$ otherwise. SUB is the set of substitutions on $\mathbb{T}$ and ε the identity. The application of a substitution σ to a term t is written as σt. The composition of substitutions is defined as the usual

composition of mappings: $(\sigma \circ \tau)t = \sigma(\tau\, t)$ for $t \in \mathbb{T}$. Hence SUB is a substitution monoid, it is the set of finitely representable endomorphisms on the term algebra $\mathbb{T}$: $\varepsilon \in$ SUB and if $\sigma, \tau \in$ SUB then $\sigma\circ\tau \in$ SUB (identity and composition); if $c \in \mathbb{F}_0$, $f(t_1,...,t_n) \in \mathbb{T}$ then $\sigma c = c$ and $\sigma f(t_1,...,t_n) = f(\sigma t_1,...,\sigma t_n)$ (homomorphism); $\text{card}(\{v \in V: \sigma v \neq v\}) < \infty$ for $\sigma \in$ SUB.
The domain of a substitution is $\mathbb{T}$; by a slight abuse of language we define the special 'domain' (the 'codomain') of a substitution σ as the set of variables actually moved by σ (the terms introduced by σ):

$\text{DOM}\sigma = \{x \in V: \sigma x \neq x\}$	(domain of σ)
$\text{COD}\sigma = \{\sigma x : x \in \text{DOM}\sigma\}$	(codomain of σ)
$\text{VCOD}\sigma = \mathbb{V}(\text{COD}\sigma)$	(variables of codomain of σ)

If $\text{VCOD}\sigma = \varnothing$ then σ is a ***ground substitution.*** A substitution ρ is called a ***renaming substitution*** iff $\text{COD}\rho \subseteq V$ and $\rho x = \rho y$ implies $x = y$ for $x,y \in \text{DOM}\rho$. A ***permutation*** is a bijective renaming substitution.
Given a set of variables $W \subseteq \mathbb{V}$, E-equality in $\mathbb{T}$ is extended to the set of substitutions SUB by :

$$\sigma =_E \tau\ [W] \quad \text{iff} \quad \forall x \in W \quad \sigma x =_E \tau x$$

We say σ and τ are ***E-equal on W*** or the restrictions $\sigma|_W$ and $\tau|_W$ are E-equal.
A term s is an ***E-instance*** of t (or t is ***more general*** than s), $t \leq_E s$, iff there exist $\lambda \in$ SUB with $\lambda t =_E s$; s is ***E-equivalent*** to t, $s \equiv_E t$, iff $s \leq_E t$ and $s \geq_E t$. These notions are extended to substitutions by : A substitution τ is ***more general*** than σ on W (or σ is an ***E-instance*** of τ on W):

$$\tau \leq_E \sigma\ [W] \quad \text{iff} \quad \exists\, \lambda \in \text{SUB with } \lambda\tau =_E \sigma\ [W].$$

Two substitutions σ, τ are ***E-equivalent*** on W :

$$\sigma \equiv_E \tau\ [W] \quad \text{iff} \quad \sigma \leq_E \tau\ [W] \text{ and } \tau \leq_E \sigma\ [W].$$

Given two terms s, t in $\mathbb{T}$ and an equational theory E, an ***E-unification problem*** is denoted as $\langle s=t\rangle_E$. Note that a unification problem is not only characterized by the equational theory E, but also by the signature out of which s and t are built. In particular the type of a unification problem, as defined below, depends on both E and $\mathbb{T}$.
The problem $\langle s = t\rangle_E$ is ***E-unifiable*** iff there exists a substitution $\sigma \in$ SUB such that $\sigma s =_E \sigma t$, σ is called an ***E-unifier*** of s and t. The set of all E-unifiers of s and t is written $U_E(s,t)$, which is a left ideal in the substitution monoid SUB, since $U_E =_E \text{SUB}\circ U_E$ [W]. In particular U_E is a filter or an (upper) segment of SUB, since if $\sigma \in$ SUB and $\tau \in U_E$ and $\tau \leq_E \sigma$ then $\sigma \in U_E$.
Without loss of generality we assume the unifiers of s and t to be idempotent, i.e. $\sigma\bullet\sigma = \sigma$, since if not, we can always find equivalent ones which are. For a given unification problem $\langle s = t\rangle_E$, it would be of little avail to compute the whole set of unifiers $U_E(s, t)$, which is always recursively enumerable for a decidable theory E, but instead smaller sets useful in representing U_E.

Therefore we define a generating set of U_E , called $cU_E(s,t)$ the ***complete set of unifiers of s and t on*** $W = V(s,t)$, as:

(i) $cU_E \subseteq U_E$ (correctness)

(ii) $\forall\, \delta \in U_E\ \exists\, \sigma \in cU_E:\quad \sigma \leq_E \delta\ [W]$ (completeness)

The base $\mu U_E(s,t)$, called the ***set of most general unifiers*** , is defined as the μ-set of $U_E(s,t)$ with respect to $\leq_E [W]$:

(iii) $\forall\, \sigma,\tau \in \mu U_E(s,t):$ if $\sigma \leq_E \tau\ [W]$ then $\sigma = \tau$. (minimality)

A set of substitutions $S \subseteq SUB$ is said to be ***separated on W away from Z*** , with $W \subset Z$, iff the following two conditions are satisfied:

- $DOM\sigma = W$ for all $\sigma \in S$
- $VCOD\sigma \cap Z = \emptyset$ for all $\sigma \in S$.

For substitutions σ separated on W we have in paticular $DOM\sigma \cap VCOD\sigma = \emptyset$, which is equivalent to the idempotence of σ. This property is often technically useful and we usually require μU_E to be separated on $W = V(s,t)$ away from some $Z \supset W$. The set μU_E does not always exist (Fages & Huet, 1983; Schmidt-Schauß, 1986; Baader, 1986); if it does then it is not unique. However it is unique up to the equivalence $\equiv_E$ (see for example (Fages & Huet, 1983)) and hence it is sufficient to compute just one μU_E as a representative of the equivalence class $[\mu U_E]_{\equiv E}$.
A possible reason for the non-existence of minimal sets of unifiers is that the quasi order $\leq_E [W]$ on U_E is not well-founded. Obviously if it is well-founded (i.e. every strictly decreasing chain in U_E is finite), a minimal subset will always exist. More generally, if every decreasing chain of unifiers -including infinite ones - has a lower bound in U_E , then U_E has a μ-set. Although sufficient, this condition is not necessary for the existence of minimal sets of E-unifiers.
The above definitions are given for a unification problem that consists just of one equation, but unfortunately we have the following theorem : there is a theory E, where all single unification problems (as defined above) have minimal sets of unifiers, but for a finite set of problems this is not the case, the minimal set of E-unifiers does not even exist (Bürckert et al., 1986). For that reason we extend the definition of a unification problem to a finite system of equations $\Gamma = \{s_i = t_i : 1 \leq i \leq n\}$. Γ is called an ***E-unification problem*** or an equation system and is then denoted as:

$$<s_i = t_i : 1 \leq i \leq n>_E$$

A substitution σ is an ***E-unifier of*** Γ, or a solution of Γ, iff $\sigma s_i =_E \sigma t_i$, for $1 \leq i \leq n$. The set of E-unifiers is denoted accordingly as $U_E(\Gamma)$, similarily $cU_E(\Gamma)$ for a complete set of unifiers and $\mu U_E(\Gamma)$ for a minimal one.

Based on the cardinality of μU, we can classify unification problems and equational theories according to the following ***unification hierarchy,*** which turned out to be a backbone of unification theory. A *unification problem* Γ for an equational theory E is of type:

(i)	***unitary***	if $\mu U_E(\Gamma)$ exists and has at most one element
(ii)	***finitary***	if $\mu U_E(\Gamma)$ exists and is finite
(ii)	***infinitary***	if $\mu U_E(\Gamma)$ exists and is infinite
(iv)	***nullary*** (or ***zero***)	if $\mu U_E(\Gamma)$ does not exist.

Similarily we say an *equational theory* E is unitary (is finitary) if for all Γ, $\mu U_E(\Gamma)$ is unitary (is finitary), and E is infinitary (is nullary) if there exists some Γ such that $\mu U_E(\Gamma)$ is infinitary (is nullary).The unitary, finitary, infinitary and nullary classes of equational theories are $\mathcal{U}_1$, $\mathcal{U}_\omega$, $\mathcal{U}_\infty$ and $\mathcal{U}_0$ respectively.
We say that $\mathcal{U} = \mathcal{U}_1 \cup \mathcal{U}_\omega \cup \mathcal{U}_\infty$ the class of unitary, finitary and infinitary theories is ***μ-based,*** whereas $\mathcal{U}_0$ is not μ-based.
A ***unification algorithm*** for a given theory E is an algorithm that takes a set of equations Γ as input and generates some subset of $U_E(\Gamma)$. A ***complete*** unification algorithm generates a complete set $cU_E(\Gamma)$ and a ***minimal*** unification algorithm generates a base $\mu U_E(\Gamma)$. An important task of the field is to find minimal unification algorithms for a given theory, however for many applications the notion of a minimal algorithm is not strong enough, since it does not imply that the algorithm terminates even for a finite μU_E . On the other hand for a finitary theory the minimality requirement is often too strong, since an algorithm which generates a superset of μU may be far more efficient than a minimal one and hence sometimes preferable.
For that reason we say a unification algorithm is ***type conformal*** if it generates a set Ψ with:

(i) $\mu U \subseteq \Psi \subseteq cU$, i.e. Ψ is a complete set of unifiers.
(ii) If E is finitary then Ψ is finite and the algorithm terminates.
(iii) If E is infinitary then $\Psi \equiv_E \mu U$, i.e. Ψ is a μ-base.

The aim of Unification Theory is to give an answer to the following three mayor problems:

PROBLEM ONE:	For a given equational theory E, is it decidable whether two terms are unifiable in E ?
PROBLEM TWO:	Given an equational theory E, what is its unification type ?
PROBLEM THREE:	For a given μ-based equational theory E find an (efficient) unification algorithm that enumerates μU_E; respectively find an algorithm that is type conformal.

3. Results

The development of unification theory into a scientific field of its own was hallmarked by the slow emergence of a general theory, that addresses the above mentioned problems in a rather general setting. It was motivated inter alia by the comparatively late realization, that unification is equation solving in varieties, however the abstract nature of the theories under investigation as well as the computeroriented approach account for its destinct syntactic flavor.
Typical questions that are asked in this field are: How and under what conditions can unification algorithms be combined? Why is the combination of a finitary and an infinitary theory sometimes finitary and sometimes infinitary? How is a unification problem influenced by the choice of its signature, in particular when order sorted signatures are taken into account? Is it possible to find a Universal Unification Algorithm (similar to a Universal Turing Machine), which takes as input a pair of terms *and* an equational theory? What is the exact relationship between matching and unification? Is it possible to develop a general theory that classifies equational theories with respect to the unification hierarchy?
For this and other reasons this section is divided into two main subsections: special results and results from the general theory.

3.1 THE SPECIAL THEORY

> *"... a general comparative study necessarily presupposes some previous separate study, comparison being impossible without knowledge."*
>
> *N.Whitehead, 1898*

This paragraph is divided into six parts giving separate accounts of first and higher order unification, of unification in sorted logics, unification in programming languages, of unification grammars and of some complexity results.

3.1.1 FIRST ORDER UNIFICATION

Unification in the Absolutely Free Termalgebra.

The historical experience with the early deduction systems clearly revealed that "the unification computation occurs at the very heart of most deduction systems. It is the addition and multiplication of deduction work. There is accordingly a very strong incentive to design the last possible ounce of efficiency into a unification program. The incentive is very much the same as that for seeking maximally efficient realizations of the elementary arithmetic operations in numerical computing - and the problem is every bit as interesting" (Robinson, p.64, 1971).
A first and influential paper in this direction was published in 1971 by J.A.Robinson (Robinson, 1971), who proposed a table-driven implementation technique that derived its strength from an

ingenious manipulation of pointer structures, which is - with some improvements - still at the heart of many current techniques.The manipulation of pointers, instead of the objects themselves, was also proposed by R.Boyer and J.S.Moore and became known as structure sharing (*Moore, 1973).

The final race for the fastest algorithm however started in 1973 with a proposal by L.D.Baxter (Baxter, 1973), that was further improved by M.Venturini-Zilli (Venturini-Zilli, 1975) in 1975, by G.P.Huet (Huet,1976) in 1976 and by A.Martelli and U.Montanari (Martelli & Montanari, 1979) in 1979, who presented an almost linear algorithm. It is a well-known fact that the original unification algorithm is exponential in worst case. The first linear unification algorithm was found in 1977 by M.Paterson and W.Wegman and finally published in (Paterson & Wegman, 1978). They used a particular data structure, directed acyclic graphs (dags), to represent the terms. Linearity is achieved by moving an additional pointer structure through these dags.
Although this result appeared to settle the problem once and for all, the issue was taken up again, when it became apparent that maintaining the dags and the pointer structure can be expensive and for most practical cases (i.e. short and usually not deeply nested terms) too inefficient.
A recent improvement was published by D.Kapur, M.S.Krishnamoorthy and P.Narendran (Kapur et al., 1982), other improvements or specific implementation techniques are published among others in (Bidoit & Corbin, 1983; Escalada & Ghallab, 1987). A comparison of several algorithms in terms of empirical findings was carried out by G.Winterstein (Winterstein, 1977).

Unification in Equational Theories.

The following table summarizes most of the results that have been obtained for unification problems in special equational theories E. The special theories consist of combinations of the following equations:

A:	$f(f(x,y), z) = f(x, f(y,z))$	U:	$1 * x = x * 1 = x$
FPAG:	Finitely Presented Abelian Group	QG:	Quasi-Groups
AG:	Abelian Groups	H10:	Hilbert'sTenthProblem
ABS:	Signed Binary Trees	BR:	Boolean Rings
D_R:	$f(x, g(y,z)) = g(f(x,y), f(x,z))$	C:	$f(x,y) = f(y,x)$
D_L:	$f(g(x,y), z) = g(f(x,z), f(y,z))$		
H:	$\varphi(x \circ y) = \varphi(x) \circ \varphi(y)$	I:	$f(x,x) = x$
T:	$f(g(x,y), g(y,z)) = f(g(x,y), g(x,z))$		
C_L:	$f(f(x,y), z) = f(f(x,z), y)$	MINUS:	$-(-x) = x$; $-(x * y) = (-y)*(-x)$
C_R	$f(x, f(y,z)) = f(y, f(x,z))$	FH:	$1*x = x$, $q(x*y) = q(y)$

The column under A_E indicates whether or not a type conformal algorithm is known.

Theory	Type of E	Unification decidable	A_E	References
Ø	$\mathcal{U}_1$	Yes	Yes	(Herbrand, 1930; Robinson, 1965, 1971; Knuth & Bendix, 1970; Guard, 1964; Prawitz, 1960; Baxter, 1973; Huet, 1976; Martelli & Montanari, 1979; Paterson & Wegmann, 1978; Kapur et al., 1982)
A	$\mathcal{U}_\infty$	Yes	Yes	(Hmelevskij, 1967; Plotkin, 1972; Siekmann, 1975; Livesey & Siekmann, 1975; Makanin, 1977)
C	$\mathcal{U}_\omega$	Yes	Yes	(Herold, 1987; Kirchner, 1985; Siekmann, 1976)
I	$\mathcal{U}_\omega$	Yes	Yes	(Raulefs & Siekmann, 1978; Hullot, 1980; Herold, 1986)
A+C	$\mathcal{U}_\omega$	Yes	Yes	(Stickel, 1981; Livesey & Siekmann, 1976; Hullot, 1979; Fages, 1983; Huet, 1978; Herold & Siekmann, 1986; Büttner, 1985)
A+I	$\mathcal{U}_\omega$	Yes	?	(Siekmann & Szabo, 1982; Schmidt-Schauß, 1986; Baader, 1986)
C+I	$\mathcal{U}_\omega$	Yes	Yes	(Raulefs & Siekmann, 1978; Jouannaud et al., 1983)
A+C+I	$\mathcal{U}_\omega$	Yes	Yes	(Livesey & Siekmann, 1976; Büttner, 1986)
D	$\mathcal{U}_\infty$	?	Yes	(Szabo ,1982; Arnborg & Tidén, 1985; Mzali, 1986; Szabo & Unvericht, 1978)
D+A	$\mathcal{U}_\infty$	No	Yes	(Szabo, 1982; Siekmann & Szabo, 1986)
D+C	$\mathcal{U}_\infty$	?	Yes	(Szabo, 1982)
D+A+C	$\mathcal{U}_\infty$	No	Yes	(Szabo, 1982)
D+A+I	?	Yes	?	(Szabo, 1982)
H	$\mathcal{U}_1$	Yes	Yes	(Vogel, 1978)
T	$\mathcal{U}_\omega$	Yes	Yes	(Kirchner, 1985)
T+C	$\mathcal{U}_\omega$	Yes	Yes	(Kirchner, 1985)
T+C+C	$\mathcal{U}_\omega$	Yes	Yes	(Kirchner, 1985)
$C_{R,L}$	$\mathcal{U}_\omega$	Yes	Yes	(Jeanrond, 1980)
QG	$\mathcal{U}_\omega$	Yes	Yes	(Hullot, 1980)
AG	$\mathcal{U}_\omega$	Yes	Yes	(Lankford, 1979; Lankford et al., 1984)
H10	?	No	?	(Matiyasevitch, 1970; Davis, 1973)
FPAG	$\mathcal{U}_\omega$	Yes	Yes	(Lankford, 1980; Kandry-Rody et al., 1985)
FH	$\mathcal{U}_0$	Yes	?	(Fages & Huet, 1983)
MINUS	$\mathcal{U}_\infty/\mathcal{U}_\omega$	Yes	Yes	(Kirchner, 1985)
ABS	$\mathcal{U}_\infty/\mathcal{U}_\omega$	Yes	Yes	(Kirchner, 1982)
BR	$\mathcal{U}_1$	Yes	Yes	(Martin & Nipkow, 1986, 1987; Büttner & Simonis, 1986)
D_l+A+U_r			No	(Arnborg & Tidén, 1985)
D_l,D_r			Yes	(Arnborg & Tidén, 1985)
U			Yes	(Arnborg & Tidén, 1985)

Except for Hilbert's Tenth Problem, we have not included the classical work on equation solving in common structures such as rings and fields, which is well known. Let us comment on a few entries in the above table:

The **Robinson Unification Problem**, i.e. unification in the absolutely free algebra of terms or unification under the empty theory Ø, has attracted most attention so far and was already discussed in the previous paragraph.

Unification under Associativity is the famous monoid problem quoted in section 1.1. G.Plotkin gave the first unification algorithm for this theory (Plotkin, 1972) and used it to demonstrate the existence of infinitary equational theories. Completeness, correctness and minimality proofs were presented in (Siekmann, 1978), more recently in (Jaffar, 1985). J.Hmelevskij (Hmelevskij, 1967) and others worked on the decidability problem, which was finally positively settled by G.S.Makanin (Makanin, 1977).

Unification under Commutativity has a trivial solution, which is however insufficient for practical applications. In particular minimality presents a hard problem; type conformal algorithms are presented in (Siekmann, 1976; Herold, 1987; Kirchner, 1985). The main interest in this theory derives from its being finitary, which is in contrast for example to the infinitary theory of associativity. A nice characterization of this difference is possible in terms of the universal unification algorithm to be discussed below. However, a deeper theoretical explanation, of why two apparently rather similar theories belong to entirely different unification classes, is still an open research problem.

Terms under **Associativity and Commutativity** closely resemble the datastructure multisets (sets which may contain multiple occurrences of the same element), which is used in the matching of patterns (pattern directed invocation) in many programming languages of Artificial Intelligence. This pattern matching problem for multisets (often called **bags** in the AI-literature) was investigated by M.Stickel in (Stickel, 1975; Stickel, 1977), who observed that it can be reduced to the problem of solving homogeneous linear diophantine equations over positive integers, with the additional proviso that only positive linear combinations of the solution set are admissible. His results were finally published in (Stickel, 1981).
Building upon the work of G.Plotkin (Plotkin, 1972), M.Livesey and J.Siekmann (Livesey & Siekmann, 1976) investigated these axioms also, since they so frequently occur in applications of automated theorem proving. Independently of M.Stickel they observed the close relationship between the AC-unification problem and solving linear diophantine equations and proposed a reduction to inhomogeneous linear diophantine equations.
However an important problem remained open: the extension of the AC-unification algorithm to the whole class of first order terms turned out to be more difficult than anticipated. The suggestions for such an extension in (Stickel, 1976) as well as the sketch of an extension in (Livesey & Siekmann, 1976) were missing a crucial point, namely that the subformulas of a term to be AC-unified can have more symbols, than the original term. Hence the termination of the extended AC-unification procedure became a major problem, which remained open for many years. It was finally positively solved by F. Fages (Fages, 1984), who invented a particular complexity measure for this purpose. G.P.Huet (Huet, 1978), A.Fortenbacher (Fortenbacher, 1983), D.Lankford (Lankford, 1985) and W.Büttner (Büttner, 1985) give efficient algorithms to solve homogeneous linear equations, where only positive linear combinations are admissible. Such an algorithm, originally investigated in (*Gordan, 1873), is an important component of every AC-unification algorithm. A comparison of

the algorithms of G.P.Huet and A.Fortenbacher and an extension of these algorithms to the case of inhomogeneous equations can be found in (Guckenbiehl & Herold, 1985).
J.M.Hullot (Hullot, 1980), F.Fages (Fages, 1984) and A.Fortenbacher (Fortenbacher, 1983; Fortenbacher, 1985) discuss computational improvements of the original Stickel-algorithm.
Recently another approach to AC-unification based on the decomposition technique of A.Martelli and U.Montanari was proposed by C.Kirchner (Kirchner, 1985; Kirchner, 1987)
G.E.Peterson and M.E.Stickel (Peterson & Stickel, 1981) present a generalisation of the Knuth-Bendix completion algorithm based inter alia on AC-unification. The practical advantage of a special purpose AC-unification algorithm is particularily well demonstrated for term rewriting systems in (Stickel, 1984).
Apart from interest in a practical and fast algorithm, which computes the set of unifiers, there is the main theoretical observation that the set of most general unifiers is always *finite* for AC-unification problems. This fact was independently discovered in (Stickel, 1985; Livesey & Siekmann, 1976). However, since the set of most general unifiers corresponds to the set of nonnegative solutions of certain linear diophantine equations, the finiteness of the μ-set of unifiers follows immediately from a theorem of Dickson (*Dickson, 1913).
Two recent papers by A.Herold, J.Siekmann (Herold & Siekmann, 1986) and W.Büttner (Büttner, 1985) improved on the original work of (Livesey & Siekmann, 1976). In (Herold & Siekmann, 1986) an extension of the algorithm to the whole class of first order terms is presented using a modification of the Fages-complexity measure in the proof of termination.
Since the axioms of associativity and commutativity so frequently occur in practice, the AC-unification algorithm has become just as important for most applications as the original Robinson algorithm for free terms. However there are still annoying efficiency problems and substantial progress is still to be expected (see for example (Bürckert et al, 1988)).

Unification under **Distributivity and Associativity** provides a point in case that the combination of two infinitary theories is an infinitary theory. Is this always the case? The (D+A)-Unification Problem is of theoretical interest with respect to Hilbert's Tenth Problem, which is the problem of Diophantine solvability of polynomial equations. An axiomatization of Hilbert's Tenth Problem would involve the axioms (A) and (D) plus additional axioms for integers, multiplication, etc. Calling the union of these axioms H10, Y.Matiyasevich's celebrated result (Matiyasevich, 1970) shows in fact the undecidability of the H10-unification problem. Now the undecidability of the (D+A)-Unification Problem demonstrates that all Hilbert axioms in H10 can be eliminated except for (D) and (A) and the problem still remains undecidable. Since A-unification is known to be decidable, the race is open as to whether or not (A) can be eliminated as well, such that (D) on its own presents an undecidable unification problem. More generally it is an interesting and natural question for an undecidable unification problem to ask for its "minimal undecidable substructure". Whatever the result may be, the (D+A)-problem already highlights the advantage of the abstract nature of unification theory in contrast to the traditional point of view, with its reliance on intuitively given entities (like integers) and structures (like polynomials).

An important recent discovery is that unification in **Boolean Rings** is unitary (Büttner & Simonis, 1986; Martin & Nipkow, 1986; Martin & Nipkow, 1987) which is likely to speed up a new technology race: Boolean rings are a common datastructure in computer science, e.g. they can be used advantageously to describe logical circuits or to build sets as data structure into logic programming languages.The fact that the unification of this data structure is *unitary* holds great practical potential in particular for programming languages, however the combination with free function symbols is unsettled.

It is important to realize that the results recorded in the above table do not always hold for the whole class of first order terms, but mostly only for some subset. The extension to the whole class of terms (assuming the empty theory for every function symbol that is not part of the known unification result) is nothing but a special case of the ***Combination Problem*** of theories. From the above table we already have:

A	infinitary,	I	finitary	and	A+I	nullary,	i.e. $\infty + \omega = 0$
D	infinitary,	A	infinitary	and	D+A	infinitary,	i.e. $\infty + \infty = \infty$
D	infinitary,	C	finitary	and	D+C	infinitary,	i.e. $\infty + \omega = \infty$
A	infinitary,	C	finitary	and	A+C	finitary,	i.e. $\infty + \omega = \omega$
C	finitary,	I	finitary	and	C+I	finitary,	i.e. $\omega + \omega = \omega$
H	unitary,	A	infinitary	and	H+A	infinitary,	i.e. $1 + \infty = \infty$
D_L	unitary,	C	finitary	and	D_L+C	infinitary,	i.e. $1 + \omega = \infty$
D_L	unitary,	D_R	unitary	and	$D_L+D_R = D$	infinitary,	i.e. $1 + 1 = \infty$

Here we assume that for example (C) and (A) hold for the same function symbol and the combination of these axioms is denoted as (C+A). But what happens if (C) and (A) hold for two different function symbols, say (C) for f and (A) for g? The known results for these combination problems are recorded in section 3.2.1.

Summarizing we notice that unification algorithms for different theories appear on first sight to be based on entirely different techniques. They provide the experimental laboratory for the general unification theory and it is paramount to obtain a much larger experimental test bed than is currently known .

Disunification

Given a unification problem $<s = t>$, we are interested in all unifiers,i.e. all substitutions σ such that $\sigma s = \sigma t$. Given a *disunification problem* $<s \neq t>$, we are interested in inequality, i.e. we are interested in all substitutions σ such that $\sigma s \neq \sigma t$.

Such problems are relevant for logic programming, sufficient completeness of algebraic specifications and 'inductionless induction' and have been investigated by A.Colmerauer (Colmerauer, 1984) and H.Comon (Comon, 1986) and C.Kirchner and P.Lescanne (Kirchner & Lescanne, 1987). We say a disequation is satisfied iff $\forall \sigma.\ \sigma s \neq \sigma t$, i.e. s=t is not unifiable. A substitution σ unifies the disequation $s \neq t$ iff $\forall \delta.\delta\sigma s \neq \delta\sigma t$. The problems with disequations are:

(i) To find a disunification algorithm for a solution of the disunification problem of uninterpreted terms, (ii) How to represent the set of disunifiers. For example the problem $<x \neq b>$, where x is a variable and b a constant, has infinitely many solutions that can not be represented by a single most general idempotent unifier (disunifier). But "x not b" *is* an intuitively satisfactory representation. For that reason it has become customary to represent the solutions in 'solved form',that is as variable/term-pairs of the form $x = t$ or $x \neq t$. Using this more expressive representation $<x \neq b>$ is now unitary.

The open research problems in this area are the extension of disunification of uniterpreted terms to E-disunification problems.

3.1.2 UNIFICATION IN ORDER SORTED LOGICS

Most programming languages are typed,i.e. usually a variable declaration ensures that the variable ranges over integers, reals, lists or such like. Similarily most practical applications of predicate logic utilize some sorted variant. For example we like to write formulas like

$$\forall\ x{:}\mathrm{REAL},\ \exists y{:}\mathrm{COMPLEX}.\ y^2 = x$$

and treat them formally as an abbreviation for

$$\forall\ x.\mathrm{real}(x) \Rightarrow \exists y.\mathrm{complex}(y) \wedge y^2 = x,$$

since the explicit representation of sorts as unary predicates has many practical disadvantages. Hence the sort information should be "built-in". Sorting (or typing) terms also provides a way of building taxonomical knowledge into the logic.

The idea is to represent the sort (or taxonomical) hierarchy separately and also to provide an algorithm,which computes the sort of every term. For example a variable x of sort REAL stands for real numbers and can only be instantiated by a term t that also represents a real number or a number of a lower type in the sort hierarchy.This restricted instantiation has to be taken into account by an extended unification algorithm, which exploits the given information and computes a set of well-sorted unifiers for two terms. The remarkable increase in efficiency of a deduction system based on sorted unification is due to the fact that two syntactically unifiable terms may not be sort-unifiable and hence many redundant deduction steps can be avoided (Walther, 1983).

There are different kinds of sorted signatures with respect to their expressiveness. The simplest version requires that the sort structure is flat, i.e. the domain is just partitioned into subdomains that do not have any subsorts. Such sort structures are called many-sorted and are often used in algebraic specifications and also for term rewriting systems. Unification with proper sort-hierarchies, but restricted to one assignment f: $S_1 \times S_2 \times .... \times S_n \rightarrow S$ for every function symbol, is called order-sorted unification and was first investigated by Ch.Walther (Walther, 1983) and A.G.Cohn (Cohn, 1987), although the idea to build sorts into the logic is older (Herbrand, 1930; *Oberschelp, 1962). Signatures as considered by Ch.Walther in (Walther, 1985) ensure that there is a single and unique most general unifier for two terms, if the sort structure is a semilattice. Otherwise see (Walther, 1986). When more than one sort assignment per function symbol (i.e.

polymorphism) is allowed, there may be more than one but at most finitely many most general unifiers (Schmidt-Schauß, 1985; Schmidt-Schauß, 1987). Signatures with a sort-hierarchy and multiple sort assignment per function symbol are useful for automated reasoning systems, algebraic specifications and functional and logic programming languages. If in addition, not only function assignments, but also term declarations are used to specify the sort of a term as proposed in (*Goguen, 1978; *Wadge, 1982) then unification may become undecidable and the set of most general unifiers may be infinite (Schmidt-Schauß, 1985).
The combination of sorted signatures with equational theories was also first investigated by M.Schmidt-Schauss,who showed that, with some restrictions, the unification algorithms for an unsorted equational theory can be used to solve unification problems in the sorted equational theory. A most recent account of order sorted unification with term declarations is (Schmidt-Schauß, 1987), which also contains a complete bibliography of the work on sorted unification.

3.1.3 UNIFICATION IN LOGIC PROGRAMMING LANGUAGES

The close relationship between logic and computation (*Hoare & Shepherdson, 1985) and the fact that predicate logic itself can be viewed as a programming language, was already discussed in section 1.1.
There are some specific problems for logic programming languages however: Terms like f(x,g(x)) and f(y,y) are not unifiable in the classical sense: although both terms are "standardized apart" (i.e. have different variables), once the first arguments of f are unified the second arguments share the same variable in y and g(y) and the socalled "occur-in-check" reports failure. In order to avoid this (expensive) checking two approaches are possible: either to admit infinite terms (Colmerauer, 1982; Mukai, 1983; Martelli & Rossi, 1984) or else to accept the occasional error as for example in most PROLOG implementations (*Clocksin & Mellish, 1981).
Since unification is *the* central operation of logic programming languages, more elaborate schemes have been designed for speed up. Most prominent is currently the WARREN-Machine (*Warren, 1983; *Gabriel et al., 1984), which consists of an abstract set of machine instructions into which a logic programming language can be compiled. This set constitutes an abstract machine and each instruction can then either be supported by actual hardware or else by some sequence of microcode instructions of a more or less conventional machine. Using these techniques current LIPS-rates (number of unifications per second) are around 100 K LIPS and estimated to be in the order of 10^4 to 10^6 K LIPS in about ten years (*Lusk & Overbeek, 1984; *Gabriel et al., 1984).

Combining Logical and Functional Programming

Universal unification algorithms as presented in section 3.2.2. are the basis of an interesting new approach to programming languages that combines the virtues of functional programming with logic programming. The idea is to have logic with equality (*Goguen & Meseguer, 1986;

Dershowitz & Plaisted, 1986; Fribourg, 1985) as a programming language and to use the predicates (i.e. the nonequality relations) for the standard logic programming aspects. The functional programming aspect is taken care of by an appropriate term rewriting system that computes the values of terms and handles the equality relation. In other words the equations are used in just the same way as they are used in the ***narrowing*** algorithms (see section 3.2.2.) and interest is in finding equational classes and narrowing strategies, such that it can be done efficiently.

For example B.Fribourg (Fribourg, 1985) discusses normalized innermost narrowing, whereas narrowing for nonterminating rewriting systems based on lazy unification (Bürckert, 1987) (lazy functional programming) was investigated by J.H.You and P.A.Subrahmanyam (You & Subrahmanyam, 1986) and S.Hölldobler (Hölldobler, 1987). A recent improvement was published by W.Nutt, R. Réty and G.Smolka (Nutt et al., 1987).

Alternatives to narrowing are presented among others by A.Martelli, C. Moiso and G.Rossi (Martelli et al., 1987) and also by J.Gallier and S.Raatz (Gallier & Raatz, 1986).

An interesting recent development called ***feature unification*** was motivated by unification grammars and knowledge representation schemes.The aim is to build socalled feature terms, an important datastructure that is used in AI to represent taxonomical knowledge as well as certain grammars,into a logic programming language (Smolka & Ait-Kaci, 1987) (see section 3.1.4. below).

The field of logic programming was in many ways influential in the development of unification theory, not the least important influence is the view that a logic program is in fact a special purpose unification algorithm that computes its answer values as appropriate bindings of the output variables, i.e. as ´most general unifiers´. This view that originated with the question answering systems (*Rulifson et al., 1972) of the sixties can be captured by the slogan that "relational programming is unification".

Unification Chips

Anticipating the upcoming technological demand for ultrafast unification, there were early attempts to "compile the unification algorithm into silicon"; for example there was a special unification processor called the SUM (Robinson, 1985).

Similarly, if the Warren instruction set is directly supported by suitable hardware, this can be viewed as a unification machine.Current experiments use a pipeline of unification processors or else try to marry the Warren machine with a (set of) special unification processor(s).

3.1.4 UNIFICATION-BASED GRAMMARS

Recently developed grammar formalisms for natural languages such as Categorical Grammar, Head Grammars, Lexical Functional Grammars, Functional Unification Grammars and Definite Clause Grammars rely on a feature/value system to represent the linguistic information about some

sentence (Shieber, 1986). These approaches to grammar formalisms have been called ***unification-based,*** since they employ unification as a central operation to manipulate the feature/value structures.

The central idea is the following: Linguistic information such as "the number (of a pronoun) is singular and its person feature has the value third" can be expressed in a ***feature structure*** as:

$$\begin{bmatrix} \text{number}: & \text{singular} \\ \text{person}: & \text{third} \end{bmatrix}$$

Here 'number' and 'person' are ***features*** and 'singular' and 'third' are their respective ***values***. The feature values may themselves be structured as for example in:

$$\begin{bmatrix} \text{cat}: & \text{NP} \\ \text{agreement}: & \begin{bmatrix} \text{number}: & \text{singular} \\ \text{person}: & \text{third} \end{bmatrix} \end{bmatrix}$$

There is a natural partial order on such feature structures called ***subsumption*** , which is based on their information content: a feature structure D_1 subsumes a feature structure D_2, $D_1 \leq D_2$, if D_1 contains a subset of the information in D_2. Since there is now a partial order, which was the basic concept underlying the formal framework of unification theory as presented above, a unification based formalism can be developed for these grammars as well: two feature structures D_1 and D_2 can be unified, if there exists a feature structure D that contains the information of both D_1 and D_2, i.e. if $D \leq D_2$ and $D \leq D_1$.

Feature structures as introduced above are not only useful for the representation of grammatical and linguistic knowledge, but can in fact be used for the representation of any knowledge (as they are nothing but nested records with a particular interpretation). This is a particularily useful datastructure, when it is combined with an appropriate inheritance hierarchy (*Touretzky, 1987) as used in semantic networks, frames and some programming languages like SMALLTALK. ***Feature unification*** is then an operation that, given two feature terms A and B, computes a feature term C denoting the intersection of the denotations of A and B (Smolka & Ait-Kaci, 1987).

Feature terms and inheritance have attracted widespread interest recently. LOGIN (Ait-Kaci & Nasr, 1986) is an extension of PROLOG, where ordinary terms are replaced by some special feature terms, called ψ-terms, and ordinary unification is replaced by ψ-unification. K.Mukai's language CIL (Mukai, 1985) bears many similarities with LOGIN. L.Cardelli (*Cardelli, 1984) gives a semantics of higher order feature types and inheritance in the framework of functional programming and denotational semantics.Recent work by W.C.Rounds and R.Kasper (*Rounds & Kasper, 1986) gives an automata-theoretic formalization of feature terms. Actual feature unification algorithms have been reported by G.Smolka, H.Ait-Kaci and R.Nasr in (Ait-Kaci, 1984; Ait-Kaci & Nasr, 1986; Smolka & Ait-Kaci, 1987).

3.1.5 HIGHER ORDER UNIFICATION

Higher order logics, most prominently represented in the work of A.Church (*Church, 1940), have provided a logical basis for many deduction systems: just as mathematics is more conveniently based on some higher order calculus, many automated reasoning systems exploit the expressiveness of higher order constructs.

An early higher order deduction system was built under the guidance of J.R.Guard, W.F.Gould developped its ω-order unification algorithm and presented it in his thesis (Gould, 1966). The potential advantages of a higher order deduction system and its mechanization was also discussed by J.A.Robinson in (*Robinson, 1969). Since the number of unifiers of ω-order terms can be prolific, G.P.Huet developed an ω-order unification algorithm and a deduction system based on a "constraint resolution" method (Huet, 1972), whose characteristic is to postpone the computation of unifiers as long as possible (lazy unification). The unification algorithm was further elaborated in (Huet, 1975) and finally in (Huet, 1976).

Independently of C.L.Lucchesi (Lucchesi, 1972) G.P.Huet (Huet, 1973) discovered that ω-order unification is undecidable for $\omega \geq 3$; later D.Baxter (Baxter, 1978) showed the same result with a different proof technique, and D. Goldfarb (Goldfarb, 1981) showed, that in fact ω-order unification is undecidable for $\omega \geq 2$, thus providing yet another characterization of the gulf between first and higher order logics.

P.Andrews work (Andrews, 1971) on higher order deduction systems was most influential in this area, a most recent account of his work and that of his students D.A.Miller,E.L.Cohen and F.Pfenning is given in (Andrews, 1984)

Another unification algorithm for ω-order terms was developped by D.Jensen and T.Pietrzykowski and reported in (Jensen & Pietrzykowski, 1973, 1976; Pietrzykowski, 1971).

3.1.6 COMPLEXITY RESULTS

In this section, which is taken from D.Kapur and P.Narendran's survey paper (Kapur & Narendran, 1987) we give results obtained in studying the complexity of matching and unification problems. Both matching and unification problems for first-order terms built solely from uniterpreted function symbols have been known to be linear in the sum of the sizes of the input terms (Paterson & Wegman, 1978). When function symbols have properties such as associativity, idempotency, etc., both problems turn out to be much harder, in fact intractable in most cases.

In the following table, symbols are used to stand for theories. The associated axiom(s) with each of the symbols is given below. For example, the symbol A implies that some of the function symbols in the terms under consideration are associative.

A: f(x,f(y,z)) = f(f(x,y),z)
C: f(x,y) = f(y,x)
I: f(x,x) = x
U: f(x,1) = x
D: f(x,g(y,z)) = g(f(x,y),f(x,z))

When more than one symbol is used to stand for a theory, it means that the axioms corresponding to each of the symbols are conjuncted. For example, ACI, stands for the theory in which some function symbols appearing in the theory are assumed to be associative, commutative and idempotent. AC matching is an NP complete problem even if each variable in the pattern is restricted to have only at most two occurrences. AC1 stands for the theory in which function symbols may be associative-commutative and terms under consideration for unification and matching have unique occurences of each variable.

The set matching problem is defined as the problem of checking, given a set of patterns (sp) and a set of subjects (ss), whether there exists a substitution σ such that the set of terms obtained by applying σ on sp is the same as the set ss. Similarly, a set unification problem is defined as the problem of checking, given two sets of terms st and ss, whether there exists a substitution σ such that the set of terms obtained after applying σ on st is the same as the set of terms obtained after applying σ on ss. Bag matching and bag unification are defined analogously except that bags of terms (instead of sets of terms) are considered, i.e., number of occurrences of a term also becomes relevant.

As the table indicates, in most cases, both matching and unification problems turn out to be of the same order of complexity. The complexity does not seem to grow even when additional properties of function symbols are assumed in some cases.

It also appears that for linear terms (terms in which every variable appears uniquely), both matching and unification problems are easier than for nonlinear terms (for matching, only the pattern has to be linear). This perhaps suggest that one of the main sources of complexity is the nonlinearity of terms.

There is one anomaly in this table, which is with respect to associative matching and unification. As the table states, associative matching is NP-complete, whereas associative unification (solvability of word equations over free semigroups) is only known to be decidable. The only complexity result known about associative unification is that it is primitive-recursive. A better upper bound is not known.

In the table, results are also given for unification problems over finitely presented algebras. In a finitely presented algebra, the presentation consists of a finite set of generators, a finite set of relations expressed using generators and the operator symbols of the algebra. Variables are not allowed in the relations. Terms under consideration for unification are „elementary terms", i.e., they can have variables but they do not have any uninterpreted function symbols. For example, FPAG is a finite presentation of abelian groups generated by a finite set of generators with a finite set of relations expressed in terms of generators and the operators of abelian groups. FPBR stands

for finitely presented boolean rings. FPCSG stands for finitely presented commutative semigroups. FPA stands for arbitrary finitely presented algebras. If a finitely presented algebra does not have any relation, it is said to be freely generated. FCSG stands for finitely generated free commutative semigroups; FCSGI stands for finitely generated free commutative semigroups with idempotency; similarly, FCMI stands for finitely generated free commutative monoids with idempotency. FBR stands for finitely generated free boolean rings. SR is a theory presented by a finite complete (canonical) term rewriting system in which for each rule, the right-hand-side is either a ground term or a subterm of the left-hand-side.

Table: Complexity of Matching and Unification Problems

E	Matching		Unification	
Φ	linear		linear	(Paterson & Wegmann, 1978)
U	NP-complete	(Arnborg & Tidén, 1985)	NP-complete	(Arnborg & Tidén, 1985)
I	NP-complete	(Kapur & Narendran, 1987)	NP-complete	(Kapur & Narendran, 1987)
C	NP-complete	(Benanav et al., 1985)	NP-complete	(Set79)
A	NP-complete	(Benanav et al., 1985)	decidable	(Makanin, 1977)
CU	NP-complete	(Kapur & Narendran, 1987)	NP-complete	(Kapur & Narendran, 1987)
CI	NP-hard	(Kapur & Narendran, 1986)	NP-hard	(Kapur & Narendran, 1986)
AU	NP-complete	(Kapur & Narendran, 1987)	decidable	(Makanin, 1977)
AI	NP-hard	(Kapur & Narendran, 1986)	NP-hard	(Kapur & Narendran, 1986)
AC	NP-complete	(Kapur & Narendran, 1986)	NP-complete	(Kapur & Narendran, 1986)
ACU	NP-complete	(Kapur & Narendran, 1986)	NP-complete	(Kapur & Narendran, 1986)
ACI	NP-complete	(Kapur & Narendran, 1986)	NP-complete	(Kapur & Narendran, 1986)
D	NP-hard	(Arnborg & Tidén, 1985)	NP-hard	(Arnborg & Tidén, 1985)
DU	NP-hard	(Arnborg & Tidén, 1985)	NP-hard	(Arnborg & Tidén, 1985)
Set	NP-complete	(Kapur & Narendran, 1986)	NP-complete	(Kapur & Narendran, 1986)
Bag	NP-complete	(Kapur & Narendran, 1987)	NP-complete	(Kapur & Narendran, 1987)
AC1	P	(Benanav et al., 1985)	P	(Kapur & Narendran, 1986)
FPCSG	decidable	(Kapur & Narendran, 1987)	decidable	(Kapur & Narendran, 1987)
FCSG	NP-complete	(Kapur & Narendran, 1986)	NP-complete	(Kapur & Narendran, 1986)
FCSGI	P	(Kapur & Narendran, 1987)	P	(Kapur & Narendran, 1987)
FCMI	P	(Kapur & Narendran, 1987)	P	(Kapur & Narendran, 1987)
FPAG	P	(Kapur et al., 1985)	P	(Kapur et al., 1985)
FBR	NP-complete	(Kapur et al., 1985)	NP-complete	(Kapur et al., 1985)
FPBR	NP-hard	(Kapur et al., 1985)	NP-hard	(Kapur et al., 1985)
FPA	NP-complete	(Kozen, 1976)	NP-complete	(Kozen, 1976)
SR	NP-complete	(Kapur & Narendran, 1987)	NP-complete	(Kapur & Narendran, 1987)

3.2 THE GENERAL THEORY

" However to generalize, one needs experience ..."
G.Grätzer, 1968

This section is devided into four parts: an account of the state of the art of the combination problem, the relationship between matching and unification, an account of current universal unification algorithms and of the classification of equational theories with respect to the unification hierarchy.

It seems premature to have a section on *foundations* as well, although there is currently interesting work in this direction. While this survey presents the 'traditional' point of view of unification theory, based on the terminology of universal algebra and computational logic and centering around the notions of the minimal set of unifiers μU and of the unification hierarchy, taking a quasi ordered set as its most basic concept, it is not at all clear if things are going to stay that way. For example there is recent work that argues that there should be a category theoretical foundations (Rydeheard & Burstall, 1985; Rydeheard & Burstall, 1986; Goguen, 1988). There is also currently work in my own research group with the aim of a more abstract, axiomatic basis with the potential advantage of a more structured view of the whole field. For example the proof of the uniqueness lemma for μU and others do not depend on the notion of unifiers at all, but only on a quasi order, hence could be generalized and shown in a more general framework.

Another direction of work, having foundational importance, questions the notion of a μ-set as the basic cornerstone of the field: empirical evidence from logic programming gives some weight to this view. The essential idea of this ***constraint-oriented*** framework for unification is the following: instead of representing the solutions of a given equation system (a unification problem) explicitely as a set of unifiers, some "solved form"of these equations themselves is taken as a representation of the solution. For example a given disunification problem is transformed step by step until a "solved form" is generated consisting solely of equations like $x = t$ or $x \neq t$, for $x \in \mathbb{V}$ and $t \in \mathbb{T}$. The interesting point is, that this view can be generalized: the equations do not need to be solved 'entirely', but only subject to certain constraints that arise naturally for example in a programming task. Among others, this has the advantage that sometimes a unification problem that is nonunitary according to the traditional definition, can have a unitary representation according to the new definition (Smolka & Ait-Kaci, 1987). This approach may well develop into an interesting alternative, since it provides a more expressive representation for the solutions as compared to idempotent unifiers. The problem is that there is as yet no satisfactory foundation for the notion of a "solved form"; for example a decidable unification problem would already count on its own as a representation of its solutions, given the present state of development.

3.2.1 COMBINATION OF UNIFICATION ALGORITHMS

Given a unification algorithm for an equational theory E_1 and another algorithm for a theory E_2: how can one obtain an algorithm for the theory $E = E_1 \cup E_2$?

There are two cases to be distinguished: In the first case, if the axioms in E_1 and E_2 involve the same function symbols, there is little hope for a general recipe that constructs a unification algorithm for E out of the separate algorithms for E_1 and E_2. For example if E_1 is the associativity axiom (A), a complete and minimal unification algorithm is known and the problem is *infinitary*. Suppose now E_2 is the commutativity axiom (C) for the same function symbol: again a type conformal algorithm is known and the problem is *finitary*. The A+C-problem is *finitary* and in particular the A+C-algorithm for the union of (A) and (C) is completely different from the separate (A) and (C) cases. As another point in case take E_1 to be the associative axiom as above, but let E_2 be the idempotent axiom.There is a type conformal algorithm for idempotence and the problem is *finitary*. However the combination of both, the A+I-unification problem is of type *nullary* .

This situation is to be expected in general: solving equations in an algebra defined by E_1 and E_2 respectively may have nothing to do with solving equations in the algebra defined by $E = E_1 \cup E_2$.

In the second case however, if E_1 and E_2 involve different function symbols, the situation is different and under certain precautions the separate algorithms for E_1 and E_2 can indeed be combined, just as decision procedures for different theories can sometimes be combined into a decision procedure for their union (*Nelson & Oppen, 1980). There are currently four approaches: Building upon the variable abstraction method of M.Stickel, that was already used for the extension of the AC-unification algorithm, K.Yelick (Yelick, 1985) and E.Tidén (Tidén, 1985) independently gave algorithms for a combination of finitary theories. The essential idea in these algorithms is that subterms, that do not belong to the theory of the top function symbol, are temporarily replaced by variables, such that one of the given algorithms is applicable.The subterms are taken care of in a recursive call, and the main problem is to show termination of the whole process. K.Yelick restricts her method to regular finitary collapse free theories, whereas E.Tidén presents his method for collapse free theories without the regularity restriction.

A second approach was developed by A.Herold (Herold, 1986), whose technique is a generalization of the constant abstraction method used for the AC-unification algorithm of M.Livesey and J.Siekmann. Again his technique is restricted to finitary and regular collapse free theories.

A third approach was taken by C.Kirchner (Kirchner, 1985), who tackles the problem by a decomposition of the terms to be unified, similar to the technique A.Martelli and U.Montanary used for their unification algorithm. Currently his combination only works for a more restrictive class than the regular finitary collapse free theories (however this can be generalized).

A most recent solution of the combination problem for arbitrary equational theories with socalled simple theories was announced by M.Schmidt-Schauss (Schmidt-Schauß, 1988), which is not restricted to collapse free theories.

3.2.2 UNIVERSAL UNIFICATION

As unification algorithms for different theories are usually based on entirely different techniques it would be interesting to have a universal unification algorithm for a whole class of theories: a ***universal unification algorithm*** for a class of theories $\mathbb{E}$, is an algorithm which takes as input a pair of terms (s, t) and a theory $E \in \mathbb{E}$ and generates a complete set of unifiers for $<s = t>_E$. In other words just as a Universal Turing Machine takes as its input the description of a special Turing Machine and its arguments, a universal unification algorithm accepts an (equational) theory E *and* two terms to be unified under E.

In a sense classical work on the mechanisation of deductive calculi constitutes a "universal unification algorithm": for example resolution is complete on pure equations (as long as the equality axioms are present) and hence is an undeterministic universal algorithm. Similarily paramodulation, E-resolution and the myriad of methods developed for equational reasoning (see (*Bläsius & Siekmann, 1988) for most references) can be seen as universal algorithms.

However in the sequel we shall just concentrate on the more specific techniques that have been proposed. There are currently two approaches:

Narrowing

To show the essential idea this class of universal algorithms is based upon, suppose $<s = t>_E$ is the unification problem and R is a canonical rewrite system for E. Let h be a "new" binary function symbol then h(s,t) is a term and we have:

There exists $\sigma \in$ SUB with $\sigma s =_E \sigma t$ iff there exist terms p, q and $\delta \in$ SUB such that $h(s,t) \overset{*}{\mapsto}_R h(p,q)$ with $\delta p = \delta q$,where $\mapsto_R$ is the narrowing relation as defined above.

A first move towards an application of this result is a proper organization of the narrowing steps $\mapsto_R$ into a tree, with the additional proviso that variables are never narrowed. Then we have: if h(p, q) is a node in the narrowing tree, such that p, q are Robinson-unifiable with σ, i.e. $\sigma p = \sigma q$ then $\delta = \sigma \circ \theta$ is a correct E-unifier for s and t, where θ is the combination of all the narrowing substitutions obtained along the path from h(s,t) to h(p,q). And vice versa, for every E-unifier τ for s and t there exists a node h(p,q) in the narrowing tree, such that p and q are Robinson-unifiable with σ and $\sigma \circ \theta \leq_E \tau$ (Hullot, 1980).

Of course the set of unifiers obtained with this tree is far too large to be of any practical interest and the work of D.Lankford (Lankford, 1979) and J.-M.Hullot (Hullot, 1980) based on (Fay, 1979), is concerned with prunig this tree while maintaining completeness. J.-M.Hullot (Hullot, 1980) shows the close correspondence between rewriting steps and narrowing, some recent literature on narrowing algorithms for logic programming is recorded in section 3.1.3.

Decomposition

An alternative approach towards a universal algorithm, developed by C. and H.Kirchner, is based on a generalisation of A.Martelli and U.Montanari's decomposition technique that was already used for the combination problem.For a given unification problem and an equational theory

E, the terms are fed into a cycle of three operations:

(i) decomposition

(ii) merging

(iii) mutation relative to E.

Essentially, the first step decomposes the terms to be unified as far as possible into its subterms, the second step merges those variable/value pairs that eventually constitute the same mapping and only the third step is specific for a particular given theory. For a class of equational theories,called syntactic theories,there is a general method for this final mutation step (see (Kirchner, 1987) for an overview and (Kirchner, 1985) for details).

Right now it appears that narrowing is preferable for certain theories, whereas decomposition is good for others: for example the decomposition technique does not work for collapse axioms like idempotence, while narrowing does (Herold, 1986). On the other hand narrowing presupposes a canonical rewriting system for the equational theory, whereas decomposition does not.

Minimality

The set of unifiers U_E is recursively enumerable for any decidable theory E: just enumerate all substitutions and check if each one unifies the given terms, which is possible as E is decidable. Hence there is the important requirement that a universal unification algorithm should either generate a minimal set μU_E or at least should be type conformal. Since such a result is unattainable in general, there is the problem to find classes of theories, such that a universal unification algorithm is minimal (is type conformal) for every theory within this class. Ideally such a class should be large enough to contain most theories of practical interest, and still admit a correct, minimal and complete universal unification algorithm for this class. J.Siekmann and P.Szabo proposed a class of equational theories as a first step in this direction (called ACFM-theories) in (Siekmann & Szabo, 1981), A.Herold (Herold, 1982) gives an extension of this result, a more recent account is (Nutt & Schmidt-Schauß, 1988).

The next 700 Unification Algorithms.

These general results can often be applied in practice for the design of an actual unification algorithm. So far the development of a special purpose algorithm was more of an art than a science, since for a given theory there was no indication whatsoever, of how the algorithm might actually work.

Using a universal unification algorithm as a starting point, this task is now much easier: first isolate the crucial parts and possible sources of inefficiency in the universal algorithm and then extract a practical and efficient special solution. A collection of canonical theories (Hullot, 1980) is a valuable source for this purpose and has already been used to find the first unification algorithms for Abelian group theory and quasi group theory (Lankford, 1979; Hullot, 1980) as well as an improvement of the algorithm for idempotence (Herold, 1986).

3.2.3 MATCHING AND UNIFICATION

An ***E-matching problem*** $\langle s \gg t \rangle_E$ is the problem to find a substitution μ with $DOM(\mu) \subseteq \mathbb{V} \backslash \mathbb{V}(s)$ such that $s =_E \mu t$. We say t is E-matchable to s and call μ an ***E-matcher*** of t to s. The set of all E-matchers of t to s is denoted by $M_E(s \gg t)$. Note that there is a difference between the matching relation and the instance relation since $f(x) \geq x$ but $M(f(x) \gg x) = \emptyset$ because $\mathbb{V}(x) \backslash \mathbb{V}(f(x)) = \emptyset$. Again we are interested in finding generating sets or μ-sets for the set of matchers. Matchers are compared by $\geq_E [\mathbb{V}(s, t)]$ in the same way as in the definition of unification, and ***complete and minimal sets of E-matchers of t to s*** denoted by $cM_E(s \gg t)$ and $\mu M_E(s \gg t)$ are defined analogously. Again minimal sets of matchers may not exist (Fages & Huet, 1986). Analogous to the unification hierarchy we can classify equational theories in ***unitary matching*** ($E \in \mathcal{M}_1$), ***finitary matching*** ($E \in \mathcal{M}_\omega$), ***infinitary matching*** ($E \in \mathcal{M}_\infty$) and ***nullary matching*** ($E \in \mathcal{M}_0$) theories. The class of μ-based theories $\mathcal{M}_1 \cup \mathcal{M}_\omega \cup \mathcal{M}_\infty$ is abbreviated by $\mathcal{M}$.

An equational theory that is finitary with respect to unification is of course finitary matching, but not vice versa: for example stringunification is finitary matching, but infinitary with respect to unification. Hence what is the relationship between matching and unification? How are the two respective hierarchies related?

If we denote the set of substitutions with $DOM(\sigma) \subseteq W$ by $SUB_{|W}$, the set of all matchers is a left ideal in the monoid $SUB_{|W}$, i.e. $M_E(s \gg t) = SUB_{|W} \circ M_E(s \gg t)$, for $W = \mathbb{V} \backslash \mathbb{V}(s)$. An equivalent definition of generating sets and bases of E-matchers defines the instance relation only in this monoid $SUB_{|W}$:

$$\sigma \geq_E \tau\ [W] \text{ iff } \exists \lambda \in SUB_{|W} \text{ with } \sigma =_E \lambda\tau, \text{ where } W = \mathbb{V}(t) \backslash \mathbb{V}(s).$$

and completeness and minimality are defined with respect to the quasi-ordering $\geq_E [W]$. This amounts to the same, as if the variables of the instance term s are blocked (or considered as constants) and E-unification is then performed for s and t. This definition is equivalent to that of 'demi-unification' in (Huet, 1976) and to the matching definition in (Szabo, 1982).

In the literature there are more general definitions of matching, but they are not adequate for the definitions of minimal and complete sets of matchers. For example in the definition of completeness, matchers could be compared only on $W = \mathbb{V}(t) \backslash \mathbb{V}(s)$ instead of $W = \mathbb{V}(s,t)$, but then $cM_E(s \gg t)$ is not a generating set for $M_E(s \gg t)$: Consider the theory

$$E = \{f(f(f(x))) = f(f(x))\} \text{ and the matching problem } \langle f(f(y)) \gg f(x) \rangle_E.$$

There are two interesting matchers $\mu = \{x \leftarrow f(y)\}$ and $\tau = \{x \leftarrow f(f(y))\}$, but both, the matcher τ and the non-matching substitution $\tau' = \{x \leftarrow f(f(a))\}$, are E-instances of μ on $\mathbb{V}(t) \backslash \mathbb{V}(s) = \{x\}$, because $\tau \geq_E \mu\ [\{x\}]$ and $\tau' \geq_E \mu\ [\{x\}]$.

As mentioned above, there is a difference between the matching relation and the instance relation.

The 'instance relation problem' can be reduced to the matching problem, if all variables of the instance terms are renamed with new variables:

$$\{\sigma:\ \sigma \in \mathrm{SUB} \text{ and } s =_E \sigma t\} = \{\mu\rho:\ \mu \in M_E(s \gg \rho t)\},$$

where ρ is the renaming substitution $\{x \leftarrow v_x: x \in V(s)\}$ with pairwise differently new variables $v_x \in \mathbb{V} \setminus V(s, t)$. Another way to remove the difference between matching and instance relation is to drop the restriction that matchers are in $\mathrm{SUB}_{|W}$ and to use the matching definition of F.Fages and G.P.Huet (Fages & Huet, 1983). But there are similar difficulties: If the substitutions are compared on $W = V(s,t)$ then $\mu = \{x \leftarrow f(y)\}$ is not the only most general matcher of $\langle f(y) \gg x\rangle$ since $\mu' = \{x \leftarrow f(y), y \leftarrow z\}$ is a matcher but $\mu \not\leq \mu'[\{x, y\}]$. On the other hand, if the comparison is on $V(t) \setminus V(s)$ then the set $cM_E(s \gg t)$ does not generate the set of all matchers $M_E(s \gg t)$ (using the same counterexample as above).

Matching and unification as defined here are special cases of the general notion of a ***V-restricted*** unification problem, which is a unification problem where the unifier is allowed to move only the variables in the subset $V \subseteq \mathbb{V}$. Some general results are shown in (Bürckert, 1986; Bürckert, 1987) in particular it is shown how the most general restricted unifiers can be computed from the unrestricted most general unifiers .

3.2.4 UNIFICATION PROPERTIES OF EQUATIONAL THEORIES

The Unification Hierarchy

In the 1970's many unitary, finitary and infinitary equational theories were discovered. It was also wellknown that for higher order logics a minimal set of unifiers μU does not always exist: i.e. for certain problems there are infinitely descending chains of unifiers $\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq \ldots$ with no lower bound. Hence the natural problem, which was open for several years: are there first order equational theories with the same unpleasant feature?

G.P.Huet and F.Fages (Fages & Huet, 1983) demonstrated that unfortunately this is the case: they construct a special equational theory, which even admits a canonical rewriting system, and showed it is of type zero. Recently M.Schmidt-Schauss (Schmidt-Schauß, 1986) and A.Baader (Baader, 1986; Baader, 1987) showed independently that idempotent semigroups (called bands in semigroup theory) are of type nullary, thus opening up a whole class of natural and quite simple theories, all of which are of type zero.

We may also ask if the unification hierarchy is the finest possible structure or else is it possible to refine the hierarchy into subclasses? A natural candidate might be the class of finitary theories that could be decomposed into ***bounded*** theories. An equational theory E is bounded by $n \in \mathbb{N}$ if for every pair of terms s,t the cardinality of $\mu U_E(s,t)$ is less than n. While it is easy to find particular unification problems that are bounded by some n (for certain subclasses of terms) it is shown in (Book & Siekmann, 1986) that equational theories which are not unitary are unbounded. Hence this notion can *not* be used to refine the hierarchy.

In particular it can not be used to characterize the borderline between unitary and finitary theories nor (considering the limes) the borderline between finitary and infinitary theories. Both of these characterization problems are still major open research problems.
Another attempt to find characterization theorems is due to P.Szabo (Szabo, 1982) who defined a ***local testset*** for a class of equational theories, in analogy to the Ehrenfeucht constructions.
Let term(E):= {l, r : l=r ∈ E} be the set of terms in the axiomatization of E and let $\mathbb{I}$(E) be the set of instances of these terms:

$$\mathbb{I}(E) = \{\sigma t : t \in \text{term}(E), \sigma \in \text{SUB}\}.$$

Similarly we define $\mathbb{G}$(E) as the finite set of all generalizations of these terms:

$\mathbb{G}(E)=\{t: t' \in \text{term}(E)$ and t is obtained from t' by a replacement of some subterms by variables$\}$.

We assume equivalent terms to be discarded, i.e. $\mathbb{G}(E)/_{\equiv}$. With these two sets the ***characteristic set*** $\mathfrak{K}$(E) of an equational theory E is defined as:

$$\mathfrak{K}(E) = \mathbb{I}(E) \cup \mathbb{G}(E) \cup \text{term}(E)$$

and the finite ***local-characteristic set*** λ as:

$$\lambda(E) := \text{term}(E) \cup \mathbb{G}(E).$$

Let $\mathcal{E}$(E) be some first order property of an equational theory E. If the property $\mathcal{E}$ only holds for a subset of terms S ∈ $\mathbb{T}$, we write $\mathcal{E}(E)|_S$, and say $\mathcal{E}(E)$ is ***$\mathfrak{K}$-reducible*** iff $\mathcal{E}(E)|_{\mathfrak{K}(E)}$ implies $\mathcal{E}$(E). Similarly theory E is ***λ-reducible*** iff $\mathcal{E}(E)|_{\lambda(E)}$ implies $\mathcal{E}$(E). Then we have:

- The matching problem for admissible, canonical and regular theories is $\mathfrak{K}$-reducible.

This theorem greatly simplifies the test for finitary or infinitary matching, since we only have to show that it holds for matching problems on the subset of terms $\mathfrak{K}$(E) (Szabo, 1982). Another result in this respect is the λ-reducibility of unitary matching theories (Szabo, 1982):

- There is a test for regular theories to be unitary matching that is λ-reducible.

Theorems of this nature are of considerable practical importance since they allow for an easy classification of a given theory. For example in 1975 P.Hayes conjectured that unification of free terms may well be the only case with at most one most general unifier. The problem turned out to be more difficult than anticipated at the time: for example let T_{aa} = {f(a,a) = a} for any constant a, then T_{aa} is unitary. Also unification in Boolean rings is unitary. In order to show that a given

theory is unitary, it was customary to invent a special algorithm and then prove its completeness and correctness for example by structural induction (Robinson, 1965; Knuth & Bendix, 1970).
A more elegant method is descibed by G.Huet (Huet, 1976): factoring $\mathbb{T}$ by $\equiv$, he showed that $\mathbb{T}|_{\equiv}$ is a complete semi-lattice under $\leq$. Hence if two terms are unifiable there exists a common instance and hence there exists a least lower bound, which is the most general such instance: thus follows unification of free terms is unitary. However using a result of P.Szabo (Szabo, 1982):

- The unitary unification theories are $\mathfrak{K}$-reducible.

this is immediate: For free terms, i.e. the empty theory, $\mathfrak{K}(E)$ is empty hence every test set is empty. But then there does not exist a pair in $\mathfrak{K}(E)$ with more than one mgu, thus follows unification of uninterpreted terms is unitary.

Classes of Equational Theories

An equation $p = q$ with $\#(X, p) = \#(X, q)$ for every symbol $X \in \mathbf{V} \cup \mathbb{F}$,where $\#(X, p)$ is the number of occurrences of symbol X in p, is called a ***permutation equation***.. A generalization of this notion is regularity: an equation $p = q$ is called ***regular*** iff $\mathbf{V}(p) = \mathbf{V}(q)$.
An equation is a ***collapse equation*** iff it is of the form $x = t$, where t is a non-variable term and x is a variable. Collapse equations of the form $p(v_1,\ldots,v_i,\ldots,v_n) = v_i$ (for some i with $1 \leq i \leq n$) with pairwise different variables $v_1,\ldots,v_n$ are called ***projection equations***. An equation is called ***subterm collapsing*** iff one side of the equation is a proper subterm of the other. Of course in a consistent theory every collapse equation is subterm collapsing.
An equational theory E is called ***permutative*** iff every equation in E is a permutation equation, ***regular*** iff every equation in E is regular, and ***collapse free*** iff it does not contain any collapse equations. In (Yelick, 1985) collapse free theories are called 'confined' and in (Kirchner, 1985) theories that contain collapse equations are said to be 'potent'. Examples for permutative and regular theories are associativity or commutativity . The theory of idempotence is an example for a theory that is regular but not permutative and not collapse free. A theory E is ***almost collapse free*** iff for the leading function symbol of every collapse equation in E there is also a projection equation in E, with the same top-function. A theory E is called ***simple*** iff there is no subterm collapsing equation in E. Associativity and commutativity are examples for such theories.
An equational theory is said to be ***finite*** iff every equivalence class of the corresponding congruence is finite. An equational theory is ***Noetherian*** iff every strictly descending chain of substitutions is finite (i.e. in Noetherian theories the strict instance relation on substitutions is well-founded). Another class of equational theories defined by the congruence $=_E$ is the class of Ω–free theories: a theory is ***Ω–free*** iff $f(s_1 ,..., s_n) =_E f(t_1 ,..., t_n)$ implies $s_i =_E t_i$ for all $1 \leq i \leq n$ and all function symbols $f \in \mathbb{F}$ (Szabo, 1982).
None of the solutions of the combination problem for unification algorithms handles the class of collapse theories. A reason for the difficulty is the fact that in collapse free theories the equivalence class of a variable only contains that variable, which is no longer true if there are collapse equations

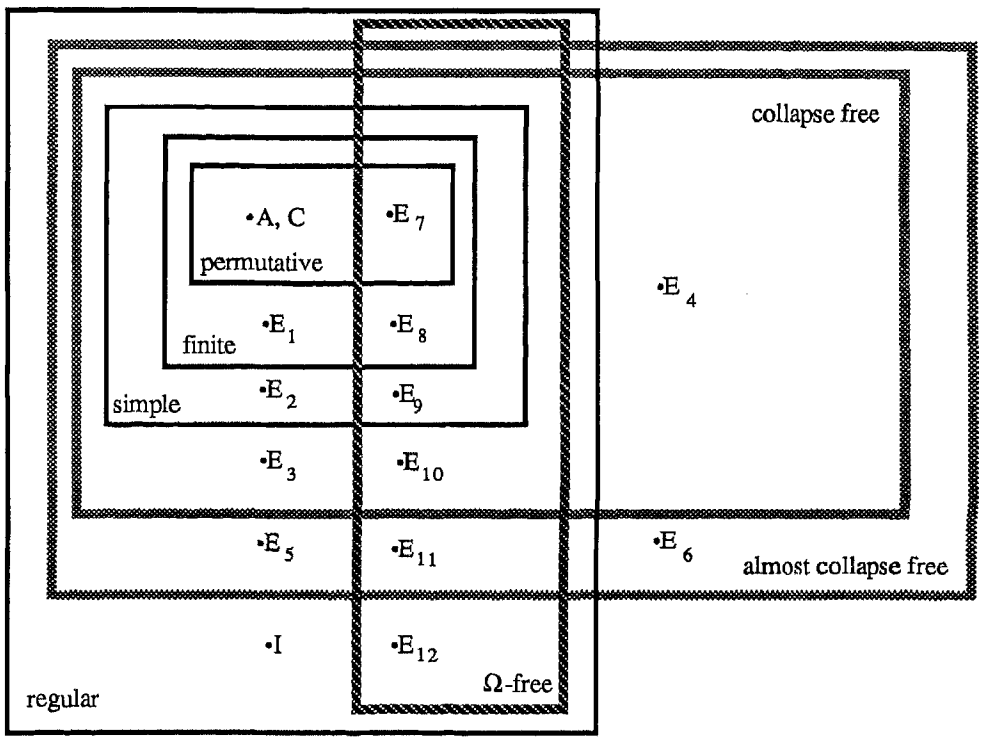
in the theory. One way to eliminate some collapse equations is described by H.J.Bürckert (Bürckert, 1986), who shows that every almost collapse free theory can be transformed into a collapse free theory with the same unification behaviour.
In regular theories variables cannot disappear and all the terms of an equivalence class have the same variables. An interesting consequence is that for every matching problem in a regular theory a minimal set of matchers exists (Szabo, 1982) (i.e. if E is a regular theory then $E \in \mathcal{M}$).
Regularity, collapse freeness, and permutativity of an equational theory can be characterized by examination of an arbitrary presentation: it is sufficient to show that some presentation is regular, collapse free or permutative.
The definition of permutative theories is due to D.S. Lankford and A.M. Ballantyne (Lankford & Ballantyne, 1977). They introduced these theories in order to extend the Knuth-Bendix completion procedure to commutative theories. Some authors (Szabo, 1982; Kirchner, 1985) defined permutative theories as finite theories, our definition is consistent with (Bürckert et al., 1987).
In finite theories the matching problems are always decidable and finitary matching (Szabo, 1982). Another important property of finite theories is that the set of most general unifiers always exists (Szabo, 1982), the reason is that finite theories are Noetherian. Obviously Noetherianness is sufficient for the existence of minimal sets of unifiers, but not necessary:

- Noetherian theories are in $\mathcal{U}$, however the theory of idempotence I is in $\mathcal{U}$ but not Noetherian.

Note, that there exist theories that are finite and hence Noetherian but infinitary unifying (e.g. associativity) and there are theories that are finite and hence Noetherian and finitary unifying (e.g. commutativity).
The class of simple theories plays a prominent role, since they admit a simple occurs-check: a variable x and a term t are E-unifiable iff x does not occur in t. Since finite theories are always simple, the simple theories are also orthogonal to the unification hierarchy.
A most interesting result for Ω–free theories has been shown by P.Szabó: Ω–free theories are regular and unitary matching and vice versa (Szabo, 1982). This result gives a nice algebraic characterization of a unification property and was the first result that links algebraic properties with the unification hierarchy (just as there is a correspondence between grammars and automata in the Chomsky-hierarchy in formal language theory). Ω–freeness of an equational theory is undecidable in general, but P.Szabó gave a sufficient condition for checking Ω–freeness: a theory E is Ω-free iff for all Robinson-unifiable terms p, $q \in \lambda(E)$ the Robinson unifier is the only most general E-unifier. Since $\lambda(E)$ is a finite set, this criterion yields a decision procedure for the Ω–freeness of E, if a complete unification algorithm for E is known. Ω–free theories are again orthogonal to the unification hierarchy, i.e., there exist Ω–free theories that are unitary, finitary and infinitary respectively (Szabo, 1982). There exists even a simple, unitary matching (hence Ω-free) theory of unification type nullary.

The Ω–free theories are also orthogonal to the other theories, i.e. there are examples of Ω–free theories being permutative, finite, simple, collapse free and regular respectively.

Up to now we introduced permutative, finite, simple, regular, collapse free, almost collapse free and Ω-free theories. These classes are arranged in the following diagram in an inclusion hierarchy with examples for each possibility and counterexamples to show the inclusions are strict. For example the theory E_9 is simple and Ω-free but not finite, hence it is regular, collapse free and almost collapse free. These results are due to H.J.Bürckert, A.Herold and M.Schmidt-Schauß (Bürckert et al., 1987).



$A := \{f(x\ f(y\ z)) = f(f(x\ y)\ z)\}$
$C := \{f(x\ y) = f(y\ x)\}$
$I := \{f(x\ x) = x\}$

$E_1 := \{f(a) = f(b)\}$
$E_2 := \{f(g(x)) = f(x)\}$
$E_3 := \{f(x\ f(y\ y)) = f(f(x\ x)\ y)\}$
$E_4 := \{x * 0 = 0\}$
$E_5 := \{f(a) = f(b),\ g(x) = x\}$
$E_6 := \{g(x\ y) = x\}$
$E_7 := \{f(g(a)) = g(f(a))\}$
$E_8 := \{f(a) = g(b)\}$
$E_9 := \{f(g(h(x))) = g(x)\}$
$E_{10} := \{f(a\ a) = a\}$
$E_{11} := \{f(g(x)) = x,\ f(x) = x\}$
$E_{12} := \{f(g(x)) = x,\ g(f(x)) = x\}$

Noetherian theories are more interesting with respect to the unification hierarchy. In order to ge some intuition for these theories, here are some relationships between the classes listed above anc the Noetherian theories:

- Every finite theory is Noetherian.
- There exists a finitary unifying theory that is not Noetherian.
- There exists a Noetherian, but not regular theory.

A simpler characterization of a Noetherian theory appears to be the requirement that there are nc infinitely descending chains of terms, but this does not hold in general. The theory E := {g(h(x)) = x, f(h(x)) = f(x)} has no infinitely descending chains of terms, but the theory is not Noetherian However:

- If E is Ω-free, then E is Noetherian iff every descending chain of terms is finite.

Decidability Results

The ***class problem*** for a class of equational theories is the problem, whether a given equational theory belongs to this class. The uniform word problem for a class of equational theories is the problem to find an algorithm that decides the word problem for all equational theories in that class. Permutativity, regularity and collapse freeness of a theory are easily decidable by examining a presentation of the theory.

Theorem 4.1 in (Nelson & Oppen, 1980) shows that for finite Church-Rosser Semi-Thue-systems T the question "Does T admit any infinite congruence classes?" is undecidable. Hence the class problem for finite theories is undecidable.We also have:

- Almost collapse freeness, Ω-freeness and Noetherianness of a theory are undecidable .
- The uniform word problem for ground terms in simple theories is undecidable.
- The class problem for simple theories is undecidable.

Finally it is shown in (Bürckert et al., 1987), that it is undecidable where an equational theory resides in the unification hierarchy, by the asterisk in $\mathcal{U}_i$* and $\mathcal{M}_i$* we denote the interserction of $\mathcal{U}_i$ and $\mathcal{M}_i$ with the class of regular theories:

- It is undecidable whether an equational theory is in $\mathcal{U}_1$*, $\mathcal{U}_\omega$*, $\mathcal{U}_\infty$*, $\mathcal{U}_0$*, $\mathcal{M}_1$*, $\mathcal{M}_\omega$*, $\mathcal{M}_\infty$*.

Hence in general:

- The class problem for $\mathcal{U}_1$, $\mathcal{U}_\omega$, $\mathcal{U}_\infty$, $\mathcal{U}_0$, $\mathcal{M}_1$, $\mathcal{M}_\omega$, $\mathcal{M}_\infty$ is undecidable.

Note that $\mathcal{M}_0$* is empty (Szabo, 1982), nevertheless,we have:

- The class problem for $\mathcal{M}_0$ is undecidable.

Let us close by summarizing the results on the word problem in various classes:

- The uniform word problem in finite equational theories is decidable.
- The uniform word problem in simple theories is undecidable
- The uniform word problem for Ω–free theories is undecidable.

The most recent results about unification and equational classes are given by H.J.Bürckert, A.Herold and M.Schmidt-Schauss in (Bürckert et al., 1987), from which most of this section has been taken.

Although the comparative study of theories and classes of theories has uncovered interesting algebraic structures, this is without doubt nothing but the tip of an iceberg of still unknown results.

Acknowledgements: I had many helpful suggestions and hints to the literature from P.Andrews, M.Davis, C.Kirchner, M.McRobbie, J.P.Jouannaud and J.A.Robinson. I am grateful to D.Kapur and P.Narendran for a preliminary version of their survey on complexity results, upon which section 3.1.6. is based. I also like to acknowledge the active involvement and the contri- butions of the members of my research group on unification H.J.Bürckert, A.Herold, W.Nutt, G.Smolka and M.Schmidt-Schauß without whose help this survey would have been impossible.

4. References

This listing contains only those references, that do not directly contribute to unification theory, all others are contained in the unification bibliography below. Note the difference in reference convention between this listing and the bibliography.

Andrews, P. (1971): "Resolution in Type Theory", J. of Symbolic Logic, vol.36.

Andrews, P.(1984) D.Miller, E.Cohen, F.Pfenning: "Automating Higher Order Logic", in: Contemporary Mathematics, American Math Soc.

Ballard, D., Brown, Ch. (1982): "Computer Vision", Prentice Hall, New Jersy.

Barrow, Ambler, Burstall (1972): "Some techniques for recognizing Structures in Pictures", Frontiers of Pattern Recognition, Academic Press Inc.

Barwise, J., Perry, J. (1983): "Situations and Attitudes", Cambrige, MIT Press.

Beilken, Ch., Mattern, F., Spenke, M. (1982): "Entwurf und Implementierung von CSSA", vol.A-E, SEKI-Memo-82-03, University of Kaiserslautern.

Birkhoff, G. (1935): "On the Structure of Abstract Algebras", Proc. Cambridge Phil. Soc., vol.31.

Bläsius, K.H., Siekmann, J. (1988): "Partial Unification for Graph based Equational Reasoning", Proc. of 9th Intern. Conf. on Automated Deduction,Springer LNCS, vol . 310.

Blair, F. et al. (1971): "SCRATCHPAD/1: An interactive Facility for Symbolic Mathematics", Proc. of the 2nd Symposium on Symbolic Manipulation, Los Angeles.

Bobrow, D., Winograd, T. (1977): "An Overview of KRL", Cognitive Science, vol.1, no.1.
Buchberger, B. (1987): "History and Basic Features of the Critical-Pair Completion Procedure", Journal of Symbolic Computation, vol 3, no 1.

Böhm, H. P., Fischer, H. L., Raulefs, P. (1977): "CSSA: Language Concepts and Programming M., 1977

Book, R. (1985): "Thue Systems as Rewriting Systems", in: Proc. of Rewriting Techniques, Springer Lecture Notes in Comp. Sci., vol.202.

Boyer, R., Moore, J. S. (1977): "A Fast String Searching Algorithm", CACM vol.20, no.10.

Brachman, R., Levesque, H. (1985): "Readings in Knowledge Representation", Will. Kaufmann Inc.

Brachman, R., Schmolze, J. (1985): "An Overview of KL-ONE", Cognitive Science, vol.9, no.2.

Bryan, H., Carnog, J. (1966): "Search Methods used with Transistor Patent Applications", IEEE Spectrum 3, 2.

Buchanan, B., Shortliffe, R. (1985): "Rule Based Expert Systems", Addison Wesley.

Buchberger, B. (1987): "History and Basic Features of the Critical-Pair Completion Procedure", Journal of Symbolic Computation, vol. 3, no.1.

Burris, S., Sankappanavar, H. P. (1981): "A Course in Universal Algebra", Springer Verlag.

CADE see: Proceedings of the International Conference on Automated Deduction, Springer Lecture Notes in Computer Science, biannually,Springer Verlag.
Cardelli, L. (1984): "A Semantics of Multiple Inheritance", Proc. of Symposium on Semantics of Data Types, Springer LNCS, vol 173.
Church, A. (1940): "A Formulation of the Simple Theory of Types", Journal of Symbolic Logic, vol 5.
Clifford, A., Preston, G. (1961): "The Algebraic Theory of Semigroups", vol.I and vol. II.
Clocksin, W., Mellish, C.(1981): "Programming in PROLOG", Springer.
Corneil, D. G. (1968): "Graph Isomorphism", Ph.D., Dept. of Computer Science, University of Toronto.
Date, C. J. (1976): "An Introduction to Database Systems", Addison-Wesley Publ. Comp. Inc.
Davis, M. (1963): "Eliminating the Irrelevant from Mechanical Proofs", Symposia of Applied Math., vol.15, American Mathematical Society.
Davis, M. (1965): "The Undecidable", Hewlett, New York, Raven Press.
Dickson, L. (1913): "Finiteness of the Odd Perfect and Primitive Abundant Numbers", Amer. Journal of Maths, vol 35.
Farber, D. J., Griswald, R. E., Polonsky, I. P. (1964): "SNOBOL as String Manipulation Language", JACM, vol.11, no.2.
Fateman, R. (1971): "The User-Level Semantic Matching Capability in MACSYMA", Proc. of the 2nd Symposium on Symbolic Manipulation, Los Angeles.
Forgy, C. (1981): "OPS5 User Manual", Carnegie Mellon University, Techn. Report, CMU-CS-81-135.
Forgy, C. (1982): "Rete: A Fast Algorithm for the Many Pattern/Object Match Problem", J. of Art. Intelligence, vol.19, no.1.
Gabriel, J. (et al.) (1984): "A Tutorial on the Warrren Abstract Machine", Argonne National Lab., ANL-84-84.
Gallaire, H., Minker, J. (1978): "Logic and Databases", Plenum Press.
Goguen, J. A., Thatcher, J., Wagner, E., Wright, J. (1977): "Initial Algebra Semantics and Continuous Algebras", JACM, vol.24, no.1.
Goguen, J. A. (1978): "Order sorted algebra", Techn. Report No 14, UCLA, Comp. Sci. Dept.
Goguen, J. A., Meseguer, J. (1986): "Eqlog: Equality, Types and Generic Modules for Logic Programming" in: DeGroot, Lindstrom (eds), Logic Programming, Functions and Equations, Prentice Hall.
Gordan, P. (1873): "Über die Auflösung linearer Gleichungen mit reellen Coefficienten", Mathematische Annalen, 23-28.
Grätzer, G. (1979): "Universal Algebra", Springer Verlag.
Guard, J. R. (1964): "Automated Logic for Semi-Automated Mathematics", Scientific report no.1, Air Force Cambridge Research Labs., AD 602 710.
Guard, J. R., Oglesby, F.C., Benneth, J.H., Settle, L.G. (1969): "Semi-Automated Mathematics", JACM, vol.18, no.1
Hearn, A. (1971): "REDUCE2, A System and Language for Algebraic Manipulation", Proc. of the 2nd Symposium on Symbolic Manipulation, Los Angeles.
Hewitt, C. (1972): "Description and Theoretical Analysis of PLANNER, a Language for Proving Theorems and Manipulation Models in a Robot", Dept. of Mathematics, Ph.C. Thesis, MIT.
Hoare, C. R. A., Shepherdson, J. C. (eds.) (1985): "Mathematical Logic and Programming Languages", Prentice Hall.
Howie, J. (1976): "Introduction to Semigroup Theory", Academic Press.
Huet, G., Oppen, D. (1980): "Equations and Rewirte Rules", in: R.Book (ed.), Formal Language Theory: Perspectives and Open Problems, New York, Academic Press.
ICOT, (1984): "Fifth Generation Computer Systems", North Holland Publ. Comp.
Kamp, H. (1981): "A Theory of Truth and Semantic Representation", in: J.Groenendijk et. al.(eds), Formal Methods in the Study of Language, Amsterdam, Mathematical Centre.
Kowalski, R. (1979): "Logic for Problem Solving", North Holland.
Kozen, D. (1976): "Complexity of finitely presented algebras", Int. Report 76-294, Dept. of Comp. Sci, Cornell University.
Loveland, D. (1978): "Automated Theorem Proving", North Holland.

Manna, Z., Waldinger, R. (1986): "How to clear a block: Plan formation in situational logic", Proc. CADE, Springer LNCS, vol 230.
Manove, Bloom, Engelmann (1968): "Rational Functions in MATHLAB", IFIP Conf. on Symb. Manipulation, Pisa.
Markov, A. A. (1954): "Trudy Mat. Inst. Steklov", no.42, Izdat. Akad. Nauk SSSR, 1954, NR17, 1038.
McNulty, G. (1976): "The Decision Problem of Equational Bases of Algebras, Annals of Mathem. Logic, vol 10.
Minsky, M. (1975): "A Framework for Representing Knowledge" in: P.Winston: "The Psychology of Computer Vision", McGraw Hill.
Moore, J. S. (1973): "Computational Logic: Structure Sharing" Th.D. thesis, Univ. of Edinburgh.
Moses, J. (1971): "Symbolic Integration: The Stormy Decade", CACM 14, 8.
Moses, J. (1974): "MACSYMA - the fifth Year", Project MAC, MIT, Cambridge.
Nelson, G., Oppen, D. (1980): "Fast Decision Procedures based on Congruence Closure", JACM, p.356-364.
Nevins, A. (1974): "A Human Oriented Logic for ATP", JACM 21.
Oberschelp, A. (1962): "Untersuchungen zur mehrsortigen Quantorenlogik", Mathematische Annalen, vol 145, pp 297-333.
Ohlbach, H. J. (1988): "A Resolution Calculus for Modal Logics", (Ph.D. thesis), Universität Kaiserslautern.
Prawitz, D. (1960): "An Improved Proof Procedure", Theoria 26.
Ramnarayan, R., Zimmermann, G. (1985): "PESA, A Parallel Architecture for OPS5 Production Systems", 19th Annual Hawaii International Conference on Systems Sciences.
Rastall, J. (1969): "Graph Family Matching", University of Edinburgh, MIP-R-62.
Robinson, J. A. (1969): "Mechanizing Higher Order Logic", in: Machine Intelligence, vol 4, Edinburgh Univ. Press.
Rounds, W.C., Kasper, R. (1986): "A Complete Logical Calculus for Record Structures Representing Linguistic Information", Proc. of IEEE Symp. on Logic in Comp.Sci.
Rulifson, Derksen, Waldinger (1972): "QA4: A Procedural Calculus for Intuitive Reasoning", Stanford Univ., Nov.
Shostak, R. (1984): "Deciding Combinations of Theories", JACM, vol.31, no.1.
Springer Lecture Notes Comp. Sci., vol.87
Siekmann, J., Szabo, P. (1982): "A Noetherian and Complete Rewirte System for Idempotent Semigroups", Semigroup Forum, vol. 25.
Slagle, J. R. (1972): "ATP with built-in Theories including Equality, Partial Ordering and Sets", JACM 19, 120-135, 1972
Sussenguth, A. (1965): "A Graph-theoretical Algorithm for Matching Chemical Structures", J. Chem. Doc.5, 1.
Tarski, A. (1968): "Equational Logic and Equational Theories of Algebra", Schmidt et al (eds), Contributions to Mathematical Logic, North Holland.
Taylor, W. (1979): "Equational Logic", Houston J. of Math. , 5.
Tennant, H. (1981): "Natural Language Processing", Petrocelli Books.
Touretzky, D. S. (1987): "The mathematics of Inheritance Systems", M. Kaufman, Publishers, Los Altos, Research Notes in Art. Intelligence
Ullman, J. R. (1976): "An Algorithm for Subgraph Isomorphism", JACM, vol.23, no.1.
Unger, S. H. (1964): "GIT - Heuristic Program for Testing Pairs of Directed Line Graphs for Isomorphism", CACM, vol.7, no.1.
van Wijngaarden (et al.) (1976): "Revised Rep. on the Algorithmic Language ALGOL68", Springer Verlag, Berlin, Heidelberg, N.Y.
Wadge, W. (1982): "Classified Algebras", Internat. Report No 46, Univ. of Warwick, England.
Wallen, L. (1987): "Matrix Proof Methods for Modal Logics, Proc. of Intern. Joint Conf. on Art. Intelligence, Milan, Italy.
Warren, D. H. (1983): "An Abstract Prolog Instruction Set", SRI Technical Note 309, SRI-International.
Winograd, T. (1972): "Understanding Natural Language", Edinburgh Univ. Press.
Winograd, T. (1983): "Language as a Cognitive Process", vol.1, Addison Wesley.
Winston, P. (1975): "The Psychology of Computer Vision", McGraw Hill.

Wos, L., Robinson, G. A., Carson, D., Shalla, L. (1967): "The Concept of Demodulation in Theorem Proving", JACM, vol.14, no.4.

Wos, L., Robinson, G. A. (1973): "Maximal Models and Refutation Completeness: Semidecision Procedures in Automatic Theorem Proving", in: Word Problems (W.W.Boone, F.B. Cannonito, R.C.Lyndon, eds), North Holland.

5.BIBLIOGRAPHY

This bibliography on unification theory is maintained and updated by the members of my research group and is, to the best of our knowledge, a complete guide to the current (Fall 1987) literature.We would greatly appreciate any suggestions for missing entries as well as notification of future publications (including a copy of the paper for our unification library), as it is becoming increasingly difficult to keep track of the many sources of publication,in particular those that are not always publicly available.

Most papers were traditionally first presented at the conferences on automated deduction (CADE), the conferences on rewriting systems and their application(RTA), and at various conferences on logic programming. In 1986 there was a first workshop on Unification Theory that is now hold annually.

Andrews, P. (1971)
"Resolution in Type Theory".
J. of Symbolic Logic, vol 36.

Andrews, P., Miller, D., Cohen, E., Pfenning, E. (1984)
"Automating Higher Order Logic"
in: Contemparery Mathematics, American Math. Soc.

Aït-Kaci, H. (1984)
"A Lattice-Theoretic Approach to Computation Based on a Calculus of Partially Ordered Type Structures".
Ph.D. Thesis, Department of Computer and Information Science, University of Pennsylvania, Philadelphia.

Aït-Kaci, H. (1985)
"Solving Type Equations by Graph Rewriting".
Proc. of the 1st International Conference on Term Rewriting Techniques and Applications, Dijon, France, May 1985, Springer LNCS 202, 158-179.

Aït-Kaci, H. (1985)
"An algebraic Semantics Approach to the effective Resolution of Type Equations".
MCC Austin, Internal Report 1985, also in:
Journal of Theoretical Computer Science, vol. 45, pp. 293 - 351.

Aït-Kaci, H., Nasr, R., LOGIN (1986)
"A Logic Programming Language with Built-In Inheritance", Journal of Logic Programming, vol. 3.

Arnborg, A., Tiden, E. (1985)
"Unification Problems with One-Sided Distributivity".
Proc. of the 1st International Conference on Term Rewriting Techniques and Applications, Dijon, France, May 1985, Springer LNCS 202, 398-406.

Arnold, A., Nivat, M. (1980)
"The Metric Space of Infinite Trees. Algebraic and Topological Properties".
Annales Societatis Mathematicae Polonae, Series IV, Fundamenta Informaticae III, vol.4, 445-476.

Baader, F. (1986)
"Unification in Idempotent Semigroups is of Type Zero".
J. of Automated Reasoning, vol. 2, 283-286.

Baader, F. (1987)
"Unification in Varieties of Idempotent Semigroups".
Internal Report, Institut für Mathematische Maschinen und Datenverarbeitung, Universität Erlangen, 1987, also in: Semigroup Forum, vol. 36.

Baader, F. (1988)
"Unification in Commutative Theories",
Institut für Informatik 1, Universität Erlangen, Interner Bericht.

Baader, F., Büttner, W. (1988)
"Unification in Commutative, Idempotent Monoids",
to appear in J. of Theor. Comp. Sci.

Büttner, W., Estenfeld, K., Schneider, H. L. A., Schmid, R., Tiden, E. (1988)
"Symbolic Constraint Handling through Unification in Finite Algebras",
Internal Report, SIEMENS AG, Corporate Labs, München.

Bachmair, L., Plaisted, D. A. (1985)
"Termination Orderings for Associative Commutative Rewriting Systems".
J. of Symbolic Computation, vol. 1, 329-349.

Baxter, L. D. (1973)
"An Efficient Unification Algorithm".
Report CS-73-23, University of Waterloo, Ontario, Canada, Department of Analysis and Computer Science.

Baxter, L. D. (1977)
"The Complexity of Unification".
Internal Report CS-77-25, University of Waterloo, Ontario, Canada,Faculty of Mathematics.

Baxter, L. D. (1978)
"The Undecidability of the Third Order Dyadic Unification Problem".
Information and Control, vol. 38, no. 2.

Benanav, D., Kapur, D., Narendran, P. (1985)
"Complexity of Matching Problems".
Proc. of the 1st International Conference on Term Rewriting Techniques and Applications,Dijon, France, May 1985, Springer LNCS 202, 417-429.

Bidoit, M., Corbin, J. (1983)
"A Rehabilitation of Robinson's Unification Algorithm".
In R.E.A. Pavon (ed.), Information Processing 83, North Holland 1983, 909-914.

Bockmayr, A. (1986)
"A Note on a Canonical Theory with Undecidable Unification and Matching Problem".
Internal Report, Universität Karlsruhe, West Germany.

Book, R., Siekmann, J. (1986)
"On Unification: Equational Theories are not bounded".
J. of Symbolic Computation, vol. 2, 317-324.

Bosco, P. G., Giovannetti, G., Moiso, C. (1987)
"Refined Strategies for Semantic Unification".
Proc. of the International Joint Conference on Theory and Practice of Software Development, Pisa, Italy, March 1987, Springer LNCS 250, 276-290.

Bürckert, H.-J. (1984)
"Unification Algorithms".
PIPE-Working Paper, Fachbereich Informatik, Universität Kaiserslautern.

Bürckert, H.-J. (1986)
"Some Relationships between Unification, Restricted Unification, and Matching".
Proc. of the 8th Conference on Automated Deduction, Oxford, July 1986,
Springer LNCS 230, 514-524. Also SEKI-Report, SR-86-05, Fachbereich Informatik,Universität Kaiserslautern.

Bürckert,H.-J. (1986)
"Lazy Theory Unification in Prolog: An Extension of the Warren Abstract Machine".
Proc. of GWAI-86 and 2. Österreichische Artificial-Intelligence-Tagung, IFB 124, Springer, 277-289.

Bürckert, H.-J., Herold, A., Schmidt-Schauß, M. (1986/87)
"On Equational Theories, Unification and Decidability".
Proc. of the 2nd International Conference on Term Rewriting Techniques and Applications,Bordeaux, France, May 1987, Springer LNCS 256, 204-215.
Also SEKI Report, SR-86-19, Fachbereich Informatik, Universität Kaiserslautern, 1986.

Bürckert, H.-J. (1987)
"Matching: A Special Case of Unification".
Seki-Report SR-87-6, Fachbereich Informatik, Universität Kaiserslautern.

Bürckert, H.-J. et al. (1988)
"Opening the AC-Unification Race", Journal of Automated Reasoning, vol 4, pp. 465-474.

Büttner, W. (1985)
"Unification in the Datastructure Multisets".
J. of Automated Reasoning, vol. 2, 1986, 75-88. Also SEKI-Report MEMO SEKI-85-V-KL, Fachbereich Informatik, Universität Kaiserslautern.

Büttner, W. (1986)
"Unification in the Datastructure Sets".
Proc. of the 8th Conference on Automated Deduction, Oxford, July 1986,
Springer LNCS 230, 470-488.

Büttner, W., Simonis, H. (1986)
"Embedding Boolean Expressions into Logic Programming".
To appear in Journal of Automated Reasoning.
Also Internal Report, Siemens, München.

Büttner, W. (1988)
"Unification in Finite Algebras is Unitary (?)"
Internal Report, SIEMENS AG, Corporate Labs, München.

Buntine, W. (1986)
"A Theory of Equations, Inequations and Solutions for Logic Programming".
Internal Report , New South Wales Institute of Technology.

Chase, R. (1987)
"An Improvement to Bottom-Up Tree Pattern Matching".
Proc. of the ACM Symposium on *POPL'87*.

Cohn, A.G. (1987)
"A More Expressive Formulation of Many Sorted Logic",
Journal of Autom. Reasoning, vol 3, no 2.

Colmerauer, A. (1984)
"Equations and Inequations on Finite and Infinite Trees".
Proc. of the International Conference on Fifth Generation Computer Systems,
Tokyo 1984, North Holland, 85-99.

Colmerauer, A. (1982)
"Prolog and infinite Trees",
in: K. Clark, S.T. Tarnlund (eds), Logic Programming, Academic Press.

Comon, H. (1986)
"Sufficient Completeness, Term Rewriting Systems and 'Anti-Unification'".
Proc. of the 8th Conference on Automated Deduction, Oxford, July 1986,
Springer LNCS 230, 128-140.

Comon, H. (1987)
"About Disequations Simplification".
Internal Report RR 645-I-IMAG-55, Laboratoire d'Informatique Fondamentale et
d'Intelligence Artificielle, St. Martin d'Hères, France.

Cosmodakis, S. S. (1985)
"Equational Theories and Database Constraints".
Ph.D. Thesis, Massachusetts Institute of Technology.

Darscheid, J. (1980)
Unifikation von absorptiven Funktionen.Diplomarbeit, Universität Bonn.

Davis, M. (1973)
"Hilbert's Tenth Problem is Unsolvable".
American Mathematical Monthly, vol. 80.

Darlington, J. L. (1971)
"A Partial Mechanization of Second Order Logic",
Machine Intelligence, vol 6, Edinburgh University Press, pp. 91-100.

DeGroot, D. Lindstrom, G. (1986)
"Logic Programming: Functions, Relations and Equations".
Prentice Hall.

Dershowitz, N., Plaisted, D. A. (1986)
"Equational Programming".
Report No. UIUCDS-R-86-1265, UILU-ENG-86-1724, University of Illinois at Urbana-Champaign.

Dershowitz, N., Plaisted, D. A. (1985)
"Logic Programming cum Applicative Programming".
Proc. of the 1985 Symposium on Logic Programming, Boston, July 1985, 54-67.

Dincbas, M., Van Hentenrynck, P. (1985)
"Extended Unification Algorithms for the Integration of Functional Programming Languages into Logic Programming",
To appear in Journal of Logic Programming, 1986. Also Technical Report lp-7, ECRC, Muenchen.

Downey, P. J., Sethi, R., Tarjan, R. E. (1980)
"Variations on the Common Subexpression Problem".
J. of the ACM, vol. 27, No.4, 758-771.

Dwork, C., Kannelakis, P. C., Mitchell, J. C. (1984)
"On the sequential Nature of Unification".
J. of Logic Programming I, 1984, 35-40. Also Technical Report CS-83-26, Brown University.

Dwork, C., Kannelakis, P. C., Stockmeyer, L. (1986)
"Parallel Algorithms for Term Matching".
Proc. of the 8th Conference on Automated Deduction, Oxford, England, July 1986, Springer LNCS 230, 416-430.

Eder, E. (1985)
"Properties of Substitutions and Unifications".
J. of Symbolic Computation, vol. 1, 1985, no. 1, 31-46. Also Interner Bericht, ATP-17-II-83, Universität München.

van Emden, M., Yukawa, K. (1986)
"Equational Logic Programming".
Technical Report, CS-86-05, Faculty of Mathematics, University of Waterloo.

Escalada, G. Ghallab, M. (1987)
"A Practically Efficient and Almost Linear Unification Algorithm".
(Draft) Laboratoire d'Automatique et d'Analyse des Systemes, Toulouse, France.

Fages, F. (1984)
"Associative Commutative Unification".
Proc. of 7th International Conference on Automated Deduction, Napa, California, May 1984,Springer LNCS 170, 194-208.

Fages, F. (1983)
"Formes Canoniques dans les Algèbres Booléennes, et Application à la Démonstration Automatique en Logique de Premier Ordre".
Thèse de 3ème Cycle, Université Paris VI.

Fages, F., Huet, G. P. (1983)
"Complete Sets of Unifiers and Matchers in Equational Theories". Theoretical Computer Science 43, 1986, 189-200. Also Proc. of the Colloquium on Trees in Algebra and Programming 1983, Springer LNCS 159.

Farmer, W. M. (1984)
"Lenght of Proofs and Unification Theory".
Ph.D. Thesis, University of Wisconsin, Madison.

Farmer, W. M. (1986)
"A Unification Algorithm for Second-Order Monadic Terms".
MITRE-Report 253 MITRE-Corporation, Bedford, Massachusetts.

Fay, M. (1979)
"First Order Unification in an Equational Theory".
Proc. of the 4th Workshop on Automated Deduction, Austin, Texas 1979, 161-167.

Fortenbacher, A. (1983)
"Algebraische Unifikation".
Diplomarbeit, Universität Karslruhe.

Fortenbacher, A. (1985)
"An Algebraic Approach to Unification under Associativity and Commutativity".
Proc. of the 1st International Conference on Term Rewriting Techniques and Applications, Dijon, France 1985, Springer LNCS 202, 381-397.
Also Algebraische Unifikation. Diplomarbeit, Universität Karslruhe, 1983.

Franzen, M., Henschen, L. J. (1987)
"A New Approach to Universal Unification and its Application to AC-Unification",
Northwestern University, Dep. of Comp. Sci., Illinois, Internal Report.

Fribourg, L. (1982)
"Démonstration automatique: Réfutation par superposition de clauses équationelles".
Thèse de 3ème Cycle, Université Paris 7.

Fribourg, L. (1984)
"Oriented Equational Clauses as Programming Language",
J. of Logic Programming, 1984, nr. 2, 165-177.

Fribourg, L. (1984)
"A Narrowing Procedure with Constructors".
Proc. of the 7th International Conference on Automated Deduction, May 1984, Napa, California, Springer LNCS 170, 259-281.

Fribourg, L. (1985)
"SLOG: A Logic Programming Language Interpreter Based on Clausal Superposition and Rewriting".
Proc. of the 1985 Symposium on Logic Programming, Boston, July 1985, 172-184.

Gallier, J., Raatz, S. (1986)
"SLD-Resolution Methods for Horn Clauses with Equality Based on E-Unification".
Proc. of the 1986 Symposium on Logic Programming, Salt Lake City, 168-179.

Gallier, J. H., Snyder, W. (1987)
"A General Complete E-unification Procedure".
Proc. of the 2nd International Conference on Rewriting Techniques and Applications, Bordeaux, France, May 1987, Springer LNCS 256, 216-227.

Gallier, J. H., Raatz, S., Snyder, W. (1987)
"Theorem Proving Using Rigid E-Unification: Equational Matings".
Proc. of the Colloquium on the Resolution of Equations in Algebraic Structures, Lakeway, Texas, May 1987. Also Internal Report, Department of Computer and Information Science, University of Pennsylvania, Philadelphia.

Goguen, J. A. (1988)
"What is Unification? A Categorical View",
Comp. Sci. Lab., SRI International, SRI-CSL-88-2.

Goldfarb, D. (1981)
"The Undecidability of the Second Order Unification Problem". Theoretical Computer Science, vol. 13, 1981, 225-230.

Goossens, D. (1983)
"The Conceptual Calculus for Automatic Program Understanding".
Proc. of the 7th International Joint Conference on Artificial Intelligence, Vancouver, 992-997.

Gould, W. E. (1966)
"A Matching Procedure for ω-Order Logic".
Scientific Report no.4, Air Force Cambridge Research Laboratories, Bedford, Massachusetts.

Guard, J. R. (1964)
"Automated Logic for Semi-Automated Mathematics", Scientific Report no 1, Air Force Cambridge Research Lab, AD 602 710.

Guckenbiehl, T., Herold, A. (1985)
"Solving Diophantine Equations".
Internal Report, MEMO SEKI-85-IV-KL, Universität Kaiserslautern.

Haridi, S., Sahlin, D. (1984)
"Efficient Implementation of Unification of Cyclic Structures".
In J.A. Campbell (ed.), Implementations of Prolog, Ellis Horwood 1984, 234-249.

Heilbrunner, S., Hölldobler, S. (1986)
"The Undecidability of the Unification and Matching Problem for Canonical Theories".
Technical Report, University of the Federal Armed Forces, München.

Herbrand, J. (1930)
"Recherches sur la Théorie de la Démonstration", (thesis)
in: W. Goldfarb (ed), Logical Writings, Cambridge.

Herold, A. (1982)
"Universal Unification and a Class of Equational Theories".
Proc. of the 6th German Workshop on Artificial Intelligence 1982, Springer IFB 58, 177-190.

Herold, A. (1983)
"Some Basic Notions of First Order Unification Theory".
Internal Report MEMO-SEKI 15-83, Institut für Informartik I, Universität Karlsruhe.

Herold, A. (1986)
"Narrowing Techniques Applied to Idempotent Unification".
Internal Report SEKI-Report SR-86-16, Fachbereich Informatik, Universität Kaiserslautern.

Herold, A., Siekmann, J. (1985)
"Unification in Abelian Semigroups".
Internal Report MEMO-SEKI 85-III-KL, Fachbereich Informatik, Universität Kaiserslautern.
Journal of Automated Reasoning,vol. 3,pp 247-283,1987

Herold, A. (1986)
"Combination of Unification Algorithms".
Proc. of the 8th Conference on Automated Deduction, Oxford, July 1986,
Springer LNCS 230, 450-469.
Also MEMO-SEKI 86-VIII-KL, Universität Kaiserslautern.

Herold, A. (1986)
"Narrowing Techniques Applied to Idempotent Unification".
Internal Report Seki-Report SR-86-16, Fachbereich Informatik, Unviersität Kaiserslautern.

Herold, A. (1987)
"Combination of Unification Algorithms",
Dissertation, Universität Kaiserslautern.

Hmelevskij, J. I. (1964)
"The Solution of certain Systems of Word Equations".
Dokl. Akad. Nauk SSSR, vol. 156, 1964, 749ff = Soviet Math. Dokl., vol. 5, 724-729.

Hmelevskij, J. I. (1966)
"Word Equations without Coefficients".
Dokl. Akad. Nauk SSSR, vol. 171, 1966, 1047ff = Soviet Math. Dokl., vol.7, no. 6, 1966, 1611-1613.

Hmelevskij, J. I., (1967)
"Solution of Word Equations in three Unknowns".
Dokl. Akad. Nauk SSSR, vol. 177, 1967, no.5 = Soviet Math. Dokl., vol. 8, 1967, no.6, 1554-1556.

Hoddinott, P., Elcock, E. W. (1986)
"PROLOG: Subsumptions of Equality Axioms by the Homogeneous Form".
Proc. of the 1986 Symposium on Logic Programming, Salt Lake City, 115-126.

Hölldobler, S. (1987)
A Unification Algorithm for Confluent Theories.
Proc. of the 14th International Conference on Automata, Languages, and Programming, Karlsruhe, Germany 1987, Springer LNCS 267, 31-41.

Huet, G. P. (1972)
"Constrained Resolution: A Complete Method for Higher Order Logic".
Ph.D. Thesis, Case Western Reserve University.

Huet, G. P. (1973)
"The Undecidability of Unification in Third Order Logic".
Information and Control, vol. 22, no. 3, 1973, 257-267.

Huet, G. P. (1973)
"A Mechanisation for Type Theory".
Rapport de Recherche, IRIA Le Chesnay.

Huet, G. P. (1975)
"A Unification Algorithm for Typed λ-Calculus".
Theoretical Computer Science, vol. 1, 1975, 27-57. Also Proc. of the Symposium on λ-Calculus and Computer Science Theory, 1975, Rome, LNCS 37. Also Rapport de Recherche No. 23, INRIA Le Chesnay.

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

Huet, G. P. (1978)
"An Algorithm to Generate the Basis of Solutions to Homogenous Linear Diophantine Equations".
Information Processing Letters, vol. 7, no. 3, 144-147.

Hullot, J.-M. (1979)
"Associative Commutative Pattern Matching".
Proc. of the 6th International Joint Conference on Artificial Intelligence, Tokyo, 406-412.

Hullot, J.-M. (1980)
"Canonical Forms and Unification".
Proc. of the 5th Conference on Automated Deduction, Les Arcs, France, 1980, Springer LNCS 87, 318-334. Also Technical Report CSL-114, SRI-International

Hußmann, H. (1985)
"Unification in Conditional-Equational Theories".
Proc. of the EUROCAL '85, Springer LNCS 204, 1985, 543-553
Also Internal Report MIP-8502, Fakultät für Mathematik und Informatik, Universität Passau.

Iran, K. B., Shin, D. G. (1985)
"A many sorted resolution based on an extension of a first order language",
Proc. of 9th IJCAI, pp. 1175-1177.

Jaffar, J. (1984)
"A Decision Procedure for Theorems about Multisets".
Internal Report, Monash University, Clayton, Victoria, Australia.

Jaffar, J. (1985)
"Minimal and Complete Word Unification".
Internal Report , Monash University, Clayton, Victoria, Australia.

Jaffar, J. Lassez, J.-L. Maher, M. (1984)
"A Theory of Complete Logic Programming with Equality".
Proc. of the International Conference on Fifth Generation Computer Systems, Tokyo 1984, 175-184.

Jensen, D. Pietrzykowski, T. (1973)
"Mechanizing ω-Order Type Theory through Unification".
Internal Report CS-73-13, Department. of Applied Analysis and Computer Science, University of Waterloo, Ontario, Canada.

Josephson, A., Dershowitz, N. (1986)
"An Implemntation of Narrowing: The RITE Way".
Proc. of the 1986 Symposium on Logic Programming, Salt Lake City, 187-197.

Jouannaud, J. P., Kirchner, C., Kirchner, H., Picard, M. (1980)
"Signed Trees: An Algebraic Frame for Solving Equations on Binary Trees. Application to Lisp Program Synthesis".
Internal Report 80-R-30, Centre de Recherche en Informatique de Nancy, France.

Jouannaud, J. P., Kirchner, C., Kirchner, H. (1981)
"Algebraic Manipulations as a Unification and Matching Strategy for Linear Equations in Signed Binary Trees".
Proc. of the 7th International Joint Conference on Artificial Intelligence, Vancouver, 1016-1023.

Jouannaud, J. P., Kirchner, C., Kirchner, H. (1982)
"Incremental Unification in Equational Theories".
Proc. of the Allerton Conference on Communication, Control and Computation, University of Illinois, Urbana Champaign, 396-405.

Jouannaud, J. P., Kirchner, C., Kirchner, H. (1983)
"Incremental Construction of Unification Algorithms in Equational Theories",
Proc. of the 10th International Conference on Automata, Languages and Programming, Barcelona, Spain 1983, Springer LNCS 154, 361-373 .
Also Internal Report 82-R-047, Centre de Recherche en Informatique de Nancy, France.

Kandri-Rody, A., Kapur, D., Narendran, P. (1985)
"An Ideal-Theoretic Approach to Word Problems and Unification Problems over Finitely Presented Commutative Algebras".
Proc. of the 1st International Conference on Rewriting Techniques and Applications, Dijon, France, May 1985, Springer LNCS 202, 345-364.

Kaplan, S. (1984)
"Fair Conditional Term Rewriting Systems: Unification, Termination and Confluence".
Technical Report no. 194, Université de Paris-Sud, Centre d'Orsay, Laboratoire de Recherche en Informatique.

Kapur, D., Krishnamoorthy, M.S., Narendran, P. (1982)
"A New Linear Algorithm for Unification".
Internal Report, no. 82CRD-100, General Electric, New York.

Kapur, D., Narendran, P. (1986)
"NP-Completeness of the Set Unification and Matching Problems".
Proc. of the 8th International Conference on Automated Deduction, Oxford, July 1986, Springer LNCS 230, 489-495.

Kapur, D., Narendran, P. (1987)
"Matching, Unification and Complexity" to appear in Journal of Autom. Reasoning.

Karttunen, L., Kay, M. (1985)
"Structure Sharing with Binary Trees",
Proc. of 23rd annual Meeting, Assoc. Comp. Linguistics, Chicago, pp. 133-136.

Karttunen, L. (1985)
"Helsinki Unification Grammars".
Technical Report, SRI-International and CSLI, Menlo Park.

Kay, M. (1984)
"Functional Unification Grammars".
Proc. of COLING, Stanford, 1984.

Kirchner, C., Kirchner, H. (1981)
"Solving Equations in the Signed Trees Theory. Application to Program Derecursivation".
Internal Report 81-R-056, Centre de Recherche en Informatique de Nancy, France.

Kirchner, C., Kirchner, H. (1982)
"Contribution à la résolution d'équations dans les algèbres libres et les variétés équationelles d'algèbres".
Thèse de 3ème cycle, Université de Nancy I, France.

Kirchner, C., Kirchner, H. (1982)
Trois méthodes de résolution d'équations dans les algèbres libres et les variétés équationnelles d'algèbres".
Actes des journées Groplan, Greco Programmation, Berjerac.

Kirchner, C. (1984)
"A New Equational Unification Method: A Generalisation of Martelli-Montanari's Algorithm".
Proc.of the 7th International Conference on Automated Decuction, Napa, California, May 1984, Springer LNCS 170, 224-247.
Also Proc. of the NSF Workshop on the Rewrite Rule Laboratory, 1983.

Kirchner, C. (1984)
"Standardizations of Equational Unification: Abstract and Examples".
Internal Report 84-R-074, Centre de Recherche en Informatique, Nancy, France.

Kirchner, C. (1985)
"Méthodes et outils de conception systématique d'algorithmes d'unification dans les théories equationelles". Thèse d'État, Université de Nancy I, France.

Kirchner, C. (1986)
"Computing Unification Algorithms".
Conference on Logic in Computer Science, 206-216.

Kirchner, C., Lescanne, P. (1987)
"Solving Disequations".
Proc. of the 2nd IEEE Conference on Logic in Computer Science, Ithaca, New York.

Kirchner, C. (1987)
"Methods and Tools for Equational Unification".
Proc. of the Colloquium on the Resolution of Equations in Algebraic Structures, Lakeway, Texas, May .

Knuth, D. E., Bendix, P. B. (1970)
"Simple Word Problems in Universal Algebras".
In J.Leech (ed.), Computational Problems in Abstract Algebra, Pergamon Press, Oxford.

Kühner, S., Mathis, Ch., Raulefs, P., Siekmann, J. (1977)
"Unification of Idempotent Functions".
Proc. of the 5th International Joint Conference on Artificial Intelligence, MIT, Cambridge,Massachusetts.

Kurukawa, T. and ICOT Working Group WG.5 (1984)
"Several Aspects of Unification".
Internal Report TM-0046, ICOT, Tokyo.

Lankford, D. S. (1979)
"A Unification Algorithm for Abelian Group Theory".
Internal Report MTP-1, Louisiana Tech University, Ruston.

Lankford, D. S. (1979)
"Some New Approaches to the Theory and Applications of Conditional Term Rewriting Systems".
Internal Report MTP-6, Louisiana Tech University, Ruston.

Lankford, D. S. (1980)
"A New Complete FPA-Unification Algorithm".
Internal Report MTP-8, Louisiana Tech University, Ruston.

Lankford, D. S. (1985)
"New Non-Negative Integer Basis Algorithms for Linear Equations with Integer Coefficients".
Internal Report, Louisiana Tech University, Ruston.

Lankford, D. S., Butler, G., Brady, B.(1984)
"Abelian Group Unification Algorithms for elementary Terms",
in: Contemporary Mathematics, vol. 29.

Lankford, D. S., Musser, D. R. (1978)
"On Semideciding First Order Validity and Invalidity".
Draft, 1978

Lassez, J.-L., Maher, M. J., Marriott, K. (1986)
"Unification Revisited".
Internal Report RC 12394 (#55630) 12/16/86, IBM, Yorktown Heights.

Livesey, M., Siekmann, J. (1975)
"Termination and Decidability Results for Stringunification".
Internal Report Memo CSM-12, University of Essex.

Livesey, M., Siekmann, J. (1976)
"Unification of Sets and Multisets".
Internal Report MEMO SEKI-76-II, Institut für Informatik I, Universität Karlsruhe.

Livesey, M., Siekmann, J., Szabo, P., Unvericht, E. (1979)
"Unification Problems for Combinations of Associativity, Commutativity, Distributivity and Idempotence Axioms".
Proc. of the 4th Workshop on Automated Deduction, University of Texas at Austin.

Lucchesi, C. : (1972)
"The Undecidability of the Unification Problem for Third Order Languages".
Internal Report CSRR 2059, Deptartment of Applied Analysis and Computer Science, University of Waterloo Ontario, Canada.

Lyndon, R. C. (1960)
"Equations in Free Groups".
Transactions of the American Mathematical Society, vol. 96, 445-457.

Makanin, G. S. (1977)
"The Problem of Solvability of Equations in a Free Semigroup".
Math. USSR Sbornik, vol. 32, no.2, 129-198.

Makanin, G. S. (1983)
"Equations in a Free Group ".
Math. USSR Izvestiya, vol. 21, no. 3, 483-546.

Mannila, H., Ukkonen, E. (1986)
"On the Complexity of Unification Sequences".
Proc. of the 3rd International Conference on Logic Programming, London, July 1986,Springer LNCS 225, 122-133.

Martelli, A., Montanari, U. (1976)
"Unification in Linear Time and Space: A structured Presentation".
Technical Report B76-16, University of Pisa.

Martelli, A., Montanari, U. (1977)
"Theorem Proving with Structure Sharing and Efficient Unification".
Technical Report S 77-7, University of Pisa.

Martelli, A., Montanari, U. (1979)
"An Efficient Unification Algorithm".
ACM Transactions on Programming Languages and Systems, vol. 4, No. 2, 258-282.

Martelli, A., Rossi, G. F. (1984)
"Efficient Unification with Infinite Terms in Logic Programming".
Proc. of the International Conference on Fifth Generation Computer Systems, North Holland, 202-209.

Manna, Z., Waldinger, R. (1987)
"A Theory of Plans" in: Georgeff, Lansky (eds) Proc. of Workshop on Reasoning about Actions and Plans.

Martelli, A., Moiso, C., Rossi, G. F. (1986)
"An Algorithm for Unification in Equational Theories".
Proc. of the 1986 Symposium on Logic Programming, Salt Lake City, 180-186.

Martelli, A., Moiso, C., Rossi, G. F. (1987)
"Lazy Unification Algorithms for Canonical Rewrite Systems".
Proc. of the Colloquium on the Resolution of Equations in Algebraic Structures, Lakeway, Texas.

Martin, U. (1986)
"Unification in Boolean Rings and Unquantified Formulae of the First Order Predicate Calculus".
Internal Report, University of Manchester.

Martin, U., Nipkow, T. (1986)
"Unification in Boolean Rings".
Proc. of the 8th International Conference on Automated Deduction, July 1986, Oxford, Springer LNCS 230, 506-513, to apperar in: Journal of Autom. Reasoning.

Martin, U., Nipkow, T. (1987)
"Boolean Unification - A Survey", University of Manchester, Internal Report, 1987, to appear in: Journal of Symb. Computation.

Matiyasevich, Y. (1970)
"Diophantine Representation of Recursively Enumerable Predicates". Ak. Nauk USSR, Ser. Math.vol. 35, 1971, 3-30.

Miller, D., Nadatlur, G. (1986)
"Higher Order Logic Programming",
Proc. of the 3rd Intern. Conf. on Logic Programming, London.

Meseguer, J., Goguen, J., Smolka, G. (1987)
"Order-Sorted Unification".
Proc. of the Colloquium on the Resolution of Equations in Algebraic Structures, Lakeway, Texas, also: Report CSLI-87-86, Centre for the Study of Language and Information, Stanford University, 1987.

Mukai, K. (1983)
"A Unification Algorithm for Infinite Trees".
Proc. of the 8th International Joint Conference on Artificial Intelligence, Karlsruhe, 547-549.

Mukai, K. (1985)
"Unification over Complex Indeterminates in Prolog".
Technical Report TR-113, ICOT.

Murray, N. U. (1979)
"Linear and almost-linear Methods for the Unification of First Order Expressions", (Ph.D. thesis) School of Comp. Sci., Syracuse University.

Murray, N. U. (1980)
"Comparing Linear and Almost Linear Methods for Unification",
Dep. of Comp. Sci., Syracuse University, Internal Report.

Mzali, J. (1983)
"Algorithme de Filtrage Associatif Commutatif".
Inernal Report 83-R-60, Centre de Recherche en Informatique de Nancy, 1983.

Mzali, J. (1984)
"Filtrage Associatif, Commutatif ou Idempotent et ses Preuves".
Inernal Report 85-R-41, Centre de Recherche en Informatique de Nancy, 1984

Mzali, J. (1986)
"Matching with Distributivity".
Proc. of the 8th International Conference on Automated Deduction, July 1986, Oxford, Springer LNCS 230, 496-505.

Nipkow, T. (1987)
"Unification in Functionally Complete Algebras".
Internal Report, University of Manchester,Dep. of Comp. Sci, 1987

Nutt, W., Réty , P., Smolka, G. (1987)
"Basic Narrowing Revisited".
Internal Report Seki-Report SR-87-7, Fachbereich Informatik, Universität Kaiserslautern.

Nonnengart, A. (1987)
"A Resolution Calculus for Temporal Logics",
University of Kaiserslautern, SEKI-Report (forthcoming).

Ohlbach, H. J. (1985)
"Theory Unifcation in Abstract Clause Graphs".
Internal Report MEMO SEKI-85-I-KL, Fachbereich Informatik, Universität Kaiserslautern.

Ohlbach, H. J. (1988)
"A Resolution Calculus for Modal Logics",
University of Kaiserslautern, SEKI-Report (Ph.D.thesis).

Padawitz, P. (1987)
"Strategy-Controlled Reduction and Narrowing."
Proc. of the 2nd International Conference on Rewriting Techniques and Applicatons, Bordeaux, France, May 1987, Springer LNCS 256, 242-255.

Paterson, M., Wegman, M. (1978)
"Linear Unification".
J. of Computer and System Sciences, vol. 16, 158-167.

Pécuchet, J. P. (1982)
"Équations avec constantes et algorithme de Makanin".
Internal Report LITP 82-4, Université Paris 7.

Peltason, C. (1979)
"Unifikationsalgorithmen für schwach assoziative Funktionen".
Diplomarbeit, Universität Bonn.

Pereira, F.C.N. (1985)
"A structure sharing representation for unification-based grammar formalisms",
Proc. of 23rd Annual Meeting Assoc. Comp. Linguistics,
Chicago, pp.127-134.

Peterson, G., Stickel, M. (1981)
"Complete Sets of Reductions for Equational Theories with Complete Unification Algorithms".
J. of the ACM, vol. 28, no. 2, 1981, 233-264.
Also Technical Note 269, SRI International.

Pietrzykowski, T. (1971)
"A Complete Mechanization of Second Order Logic".
Research Report, CSSR 2038, Deptartment of Applied Analysis and Computer Science, University of Waterloo, Ontario, Canada.

Pietrzykowski, T., Jensen, D. (1976)
"Mechanizing ω-Order Type Theory through Unification",
J. of Theoretical Comp. Sci, vol 3, pp. 123-171.

Plaisted, D. A. (1984)
"The Occur-check Problem in Prolog".
Proc. of the 1984 International Symposium on Logic Programming, Atlantic City, 272-280.

Plotkin, G. (1972)
"Building in Equational Theories".
Machine Intelligence, vol.7, Edinburgh University Press, 73-90.

Prawitz, D. (1960)
"An Improved Proof Procedure",
Theoria 26.

Purdom, P., Brown, C. (1985)
"Fast Many -to-one Matching Algorithms".
Proc. of the 1st International Conference on Rewriting Techniques and Applications, Dijon, France, Springer LNCS 202, 407-416.

Ramesh, R., Ramakrishnan, I. V. (1987)
"Optimal Speedups for Parallel Pattern Matching in Trees".
Proceedings of the 2nd International Conference on Rewriting Techniques and Applications, Bordeaux, France, Springer LNCS 256, 274-285.

Raulefs, P., Siekmann, J. (1978)
"Unification of Idempotent Functions".
Internal Report MEMO SEKI-78-I, Universität Karlsruhe.

Raulefs, P., Siekmann, J., Szabo, P., Unvericht, E. (1979)
"A short Survey on the State of the Art in Matching and Unification Problems".
SIGSAM Bulletin, vol. 13, 1979, 14-20.

Reddy, U. S. (1985)
"Narrowing as the Operational Semantics of Functional Languages".
Proc. of the 1985 Symposium on Logic Programming, Boston, 138-151.

Rety, P., Kirchner, C., Kirchner, H., Lescanne, P. (1985)
"NARROWER: A New Algorithm for Unification and its Application to Logic Programming".
Proceedings of the 1st International Conference on Rewriting Techniques and Applications, Dijon, France, May 1985, Springer LNCS 202, 141-157.

Reynolds, J. (1970)
"Transformational Systems and the Algebraic Structure of Atomic Formulas",
Machine Intelligence, vol 5, Edinburgh University Press, pp. 135-152.

Réty, P. (1987)
"Improving Basic Narrowing Techniques".
Proc. of the 2nd International Conference on Rewriting Techniques and Applicatons, Bordeaux, France, Springer LNCS 256, 216-227.

Robinson, J. A. (1965)
"A Machine Oriented Logic Based on the Resolution Principle".
J. of the ACM, vol. 12, no. 1, 23-41.

Robinson, J. A. (1969)
"Mechanizing Higher Order Logic",
Machine Intelligence, vol 4, Edinburgh University Press, pp. 151-170.

Robinson, J. A. (1971)
"Computational Logic: The Unification Computation".
Machine Intelligence, vol. 6, Edinburgh University Press 1971, 63-72.

Robinson, P. (1985)
"The Sum: An AI Coprocessor", Byte, vol 10, no 6.

Robinson, P. J., Staples, J. (1986)
"Unification of Quantified Terms".
Internal Report 70, University of Queensland.

Rush, T., Coleman, D. (1986)
"Combining Pattern-Matching Algorithms".
Internal Report HPL-BRC-TR-86-036, Hewlett Packard, Bristol.

Rydeheard, D., Burstall, R. (1985)
"The Unification of Terms: A Category-Theoretic Algorithm".
Internal Report, University of Edinburgh.

Rydeheard, D., Burstall, R. (1986)
"A Categorical Unification Algorithm".
Proc. of the Summer Workshop on Category Theory and Computer Programming, Springer LNCS 240.

Sato, M., Sakurai, T. (1984)
"Qute: A Functional Language Based on Unification".
Proc. of the International Conference on Fifth Generation Computer Systems, Tokyo, North Holland, 157-167.

Schmidt-Schauß, M. (1985)
"Unification in a Many-Sorted Calculus with Declarations".
Proc. of the 9th German Workshop on Artificial Intelligence 1985, Springer IFB 118, 118-132.

Schmidt-Schauß, M. (1985)
"A many Sorted Calculus with Polymorphic Functions Based on Resolution and Paramodulation".
Proc. of the 9th International Joint Conference on Artificial Intelligence, Los Angeles, 1985, 1162-1168.Also Internal Report MEMO SEKI-85-II-KL, Fachbereich Informatik, Universität Kaiserslautern.

Schmidt-Schauß, M. (1985)
"Mechanical Generation of Sorts in Clause Sets".
Internal Report MEMO SEKI-85-VI-KL, Fachbereich Informatik, Universität Kaiserslautern.

Schmidt-Schauß, M. (1986)
"Unification under Associativity and Idempotence is of Type Nullary".
J. of Automated Reasoning, vol. 2, no. 3, 277-282.

Schmidt-Schauß, M. (1986)
"Unification Properties of Idempotent Semigroups".
Internal Report SEKI Report SR-86-07, Universität Kaiserslautern.

Schmidt-Schauß, M. (1986)
"Unification in Many-Sorted Equational Theories".
Proc. of the 8th International Conference on Automated Deduction, Oxford, Springer LNCS 230, 538-552.

Schmidt-Schauß, M. (1987)
"Unification in Permutative Equational Theories is Undecidable".
Internal Report Seki-Report SR-87-03, Fachbereich Informatik, Universität Kaiserslautern.

Schmidt-Schauß, M. (1987)
"Computational Aspects of an Order Sorted Logic with Term Declarations", (thesis) University of Kaiserslautern.

Schmidt-Schauß, M. (1987)
"Combination of Unification Algorithms for Equational Theories with Free Function Symbols", SEKI-Report , University of Kaiserslautern.

Shieber, S. (1986)
"An Introduction to Unification-Based Approaches to Grammar".
CSLI Lecture Notes, no. 4, CSLI, Stanford University.

Shieber, S., Karttunen, L., Pereira, F. (1984)
"Notes from the Unification Underground".
Technical Report SRI-International, TN327.

Shieber, S., Karttunen, L., Pereira, F. (1985)
"More Notes from the Unification Underground".
Technical Report SRI-International, TN361.

Shostak, R. (1977)
"On the Role of Unification in Mechanical Theorem Proving".
Acta Informatica, vol.7, 319-323.

Siekmann, J. (1975)
"String-Unification".
Internal Report Memo CSM-7, Essex University.

Siekmann, J. (1976)
"Unification of Commutative Terms".
Proc. of the International Symposium on Symbolic an Algebraic Manipulation, EUROSAM'79, Marseille, France, June 1979, Springer LNCS 72, 531-545.
Also Internal Report SEKI, Institut für Informatik I, Universität Karlsruhe, Germany.

Siekmann, J. (1978)
"Unification and Matching Problems".
Ph.D. Thesis, Essex University, Memo CSA-4-78.

Siekmann, J. (1984)
"Universal Unification".
Proc. of the 7th International Conference on Automated Deduction, Napa, California, Springer LNCS 170, 1-42.

Siekmann, J. (1986)
"Unification Theory".
Proc. of the 8th European Conference on Artificial Intelligence, Brighton, vol. 2, vi-xxv.

Siekmann, J., Szabo, P. (1981)
"Universal Unification and Regular ACFM Theories".
Proc. of the 7th International Joint Conference on Artificial Intelligence, Vancouver, 532-538.
Also Internal Report MEMO SEKI-81-III, Universität Karlsruhe.

Siekmann, J., Szabo, P. (1982)
"Universal Unification and a Classification of Equational Theories".
Proc. of the 6th Conference on Automated Deduction, New York, Springer LNCS 87, 369-389.
Also Internal Report MEMO SEKI-81-II, Universität Karlsruhe.

Siekmann, J., Szabo, P. (1982)
"Universal Unification".
Proc. of the 6th German Workshop on Artificial Intelligence, Bad Honnef, Germany 1982, Springer IFB 58, 177-190.

Siekmann, J., Szabo, P. (1986)
"The Undecidability of the D_A-Unification Problem".
Internal Report SEKI-Report, SR-86-19, Fachbereich Informatik, Universität Kaiserslautern, 1986, full paper appears in the Journal of Symbolic Logic (June 1989).

Simonis, H. (1986)
"Using Extended Prolog to Model Digital Circuits",
Technical Report, Europ. Comp. Research Centre, München.

Smolka, G., Aït-Kaci, H. (1987)
"Inheritance Hierarchies: Semantics and Unification".
MCC Technical Report no. AI-057-87, Microelectronics and Computer Technology Corporation, Austin, Texas.

Smolka, G., Nutt, W., Goguen, J., Meseguer, J. (1987)
"Order-Sorted Equational Computation".
Presented at the Colloquium on the Resolution of Equations in Algebraic Structures, Lakeway, Texas, to appear in Proceedings.

Snelting, G., Henhapl, W. (1985)
"Unification in Many Sorted Algebras as a Device for Incremental Semantic Analysis". Internal Report PU2R2/85, FB Informatik, Technische Universität Darmstadt.

Snyder, W., Gallier, J. H. (1988)
"Higher Order Unification Revisited: Complete Sets of Transformations",
Dep. of Comp. Sci., Univ. of Pennsylvania, Dep. Report.

Stickel, M. E. (1975)
"A Complete Unification Algorithm for Associative-Commutative Functions". Proc. of the 4th International Joint Conference on Artificial Intelligence, Tblisi, 71-82.

Stickel, M. E. (1977)
"Mechanical Theorem Proving and Artificial Intelligence Languages",
Ph.D. Thesis, Carnegie-Mellon University, Pittsburgh.

Stickel, M. E. (1981)
"A Unification Algorithm for Associative Commutative Functions". J. of the ACM, vol.28, no.3, 423-434.

Stickel, M. E. (1984)
"A Case Study of Theorem Proving by the KB-Method", Proc. of CADE, Springer LNCS, vol 170, Springer Verlag.

Szabo, P. (1982)
"Theory of First Order Unification".
Ph.D. Thesis, Universität Karlsruhe, (in German)

Szabo, P. (1979)
"Undecidability of the D_A-Unification Problems".
Proc. of the 3rd German Workshop on Artificial Intelligence.

Szabo, P., Unvericht, E. (1978)
"The Unification Problem for Distributive Terms".
Internal Report SEKI-78-05, Institut für Informatik I, Universität Karlsruhe, Germany .

Tidén, E. (1986)
"Unification in Combinations of Collapse-Free Theories with Disjoint Sets of Function Symbols".
Proc. of the 8th International Conference on Automated Deduction, Oxford, Springer LNCS 230, 431-449.
Also Internal Report, Royal Institute of Technology, Stockholm .

Tidén, E. (1986)
"First-Order Unification in Combinations of Equational Theories".
Ph D. Thesis, Stockholm.

Tidén, E. (1988)
"Symbolic Verification of Switch-Level Circuits using a Prolog Enhanced with Unification in Finite Algebras",
SIEMENS AG, Internal Report, Corporate Labs, München.

van Vaalen, J. (1975)
"An Extension of Unification to Substitutions with an Application to Automated Theorem Proving".
Proc. of the 4th International Joint Conference on Artificial Intelligence, Tbilisi, USSR, 77-82.

Vogel, E. (1978)
"Unifikation von Morphismen".
Diplomarbeit, Universität Karlsruhe.

Venturini-Zilli, M. (1975)
"Complexity of the Unification Algorithm for First Order Expressions". Calcolo vol. 12, no. 4, 361-371.

Walther, Ch. (1983)
"A Many-Sorted Calculus Based on Resolution and Paramodulation". Proc. of the 8th International Joint Conference on Artificial Intelligence, Karlsruhe, Germany, 1983, 882-891, full version reprinted in: Research Notes in Artificial Intelligence, M. Kaufmann Publishers, Los Altos.

Walther, Ch. (1985)
"Unification in many-sorted theories",
Proc. of 6th ECAI, pp. 593-602.

Walther, Ch. (1986)
"A Classification of Many Sorted Unification Problems".
Proc. of the 8th International Conference on Automated Deduction, Oxford, Springer LNCS 230, 525-537.
Also Internal Report 10/85, Institut für Informatik I, Universität Karlsruhe.

Walther, Ch. (1988)
"Many Sorted Unification",
J. of the ACM, vol 35, no 1, pp. 1-17.

Winterstein, G. (1976)
"Unification in Second Order Logic", Bericht No 3, University of Kaiserslautern.

Winterstein, G. (et al) (1977)
"A Comparison of Several Unification Algorithms", University of Kaiserslautern, Internal Report.

Yasuura, H. (1984)
"On Parallel Computational Complexity of Unification".
Proc. of the International Conference on Fifth Generation Computer Systems, Tokyo 1984, North Holland, 235-243.

Yelick, K. (1985)
"A Generalized Approach to Equational Unification".
Internal Report MIT/LCS/TR-344, Massachusetts Institute of Technology.

Yelick, K. (1985)
"Combining Unification Algorithms for Confined Regular Equational Theories".
Proc. of the 1st International Conference on Rewriting Techniques and Applications, Dijon, France, Springer LNCS 202, 365-380.

You, J.-H., Subrahmanyam, P. A. (1985)
"E-Unification Algorithms for a Class of Confluent Term Rewriting Systems".
Technical Report, Department of Computer Science, University of Alberta, Edmonton, Canada.

You, J.-H., Subrahmanyam, P. A. (1985)
"On Completeness of First-Order Unification inEquational Theories".
Technical Report, Department of Computer Science, University of Alberta, Edmonton, Canada.

You, J.-H., Subrahmanyam, P. A. (1986)
"A Class of Confluent Term Rewriting Systems and Unification".
J. of Automated Reasoning, Vol. 2, 391-418.

You, J.-H., Subrahmanyam, P. A. (1986)
"Limitations of Narrowing for E-Unification".
Technical Report, Department of Computer Science, University of Alberta, Edmonton Canada.

Zaionc, M. (1985)
"The Set of Unifiers in Typed λ-Calculus as Regular Expression".
Proc. of the 1st International Conference on Rewriting Techniques and Applications, Dijon, France, Springer LNCS 202, 431-440.