Disunification: a Survey

Hubert COMON*

June 1991

Abstract

Solving an equation in an algebra of terms is known as unification. Solving more complex formulas combining equations and involving in particular negation is called *disunification.* With such a broad definition, many works fall into the scope of disunification. The goal of this paper is to survey these works and bring them together in a same framework.

Résumé

On appelle habituellement (algorithme d') *unification* un algorithme de résolution d'une équation dans une algèbre de termes. La résolution de formules plus complexes, comportant en particulier des négations, est appelée ici *disunification.* Avec une définition aussi étendue, de nombreux travaux peuvent être considérés comme portant sur la disunification. L'objet de cet article de synthèse est de rassembler tous ces travaux dans un même formalisme.

*Laboratoire de Recherche en Informatique, Bât. 490, Université de Paris-Sud, 91405 ORSAY cedex, France. E-mail: comon@lri.lri.fr

Contents

1 Syntax **5**
- 1.1 Basic Definitions . . . 5
- 1.2 Equational Formulas . . . 5
- 1.3 Syntactic Equivalence of qfef . . . 6
- 1.4 Elementary Formulas . . . 6
- 1.5 Particular Cases . . . 8
  - 1.5.1 Unification problems . . . 8
  - 1.5.2 Word problems . . . 8
  - 1.5.3 Equations and disequations in PROLOG II . . . 9
  - 1.5.4 Complement problems . . . 9
- 1.6 Related Work Using a Similar Formalism . . . 11
- 1.7 Extensions . . . 11
  - 1.7.1 Many-Sorted Signatures . . . 11
  - 1.7.2 Order-Sorted Signatures . . . 11
  - 1.7.3 Introduction of Inequalities . . . 12
  - 1.7.4 Arbitrary first order formulas . . . 12
- 1.8 Constraint Languages . . . 12

2 Semantics **14**
- 2.1 F-algebras . . . 14
- 2.2 Solutions of an equational formula . . . 14
- 2.3 Examples . . . 15
  - 2.3.1 $\mathcal{A} = T(F)$ or $\mathcal{A} = T(F, X)$ . . . 15
  - 2.3.2 $\mathcal{A} = RT(F)$ . . . 15
  - 2.3.3 $\mathcal{A} = IT(F)$ . . . 15
  - 2.3.4 $T(F)$-solutions in $A = NF$ . . . 15
  - 2.3.5 $\mathcal{A} = T(F)/_{=_E}$ . . . 15
  - 2.3.6 Order-Sorted Signatures . . . 16
  - 2.3.7 Classes of algebras . . . 16
  - 2.3.8 Handling inequalities . . . 16

3 Solved Forms **18**
- 3.1 Solved Forms for Unification . . . 18
- 3.2 Basic formulas . . . 19
- 3.3 Solved forms for complement problems . . . 20
- 3.4 Other solved forms . . . 20
- 3.5 Equational formulas in equational theories . . . 21
- 3.6 Summary . . . 22

4 Axioms and Rules **24**
- 4.1 Definitions . . . 24
- 4.2 Finite trees axioms . . . 25
- 4.3 Rational Trees axioms . . . 25
- 4.4 Syntactic theories axioms . . . 26
- 4.5 Rules for finite trees . . . 28

5 Termination and Completeness **34**
- 5.1 Definitions . . . 34
- 5.2 Completeness and termination results for finite trees . . . 34
- 5.3 Completeness and termination results for rational and infinite trees . . . 36
- 5.4 Completeness and termination results for quotients of finite trees . . . 36
  - 5.4.1 Quasi-free theories . . . 36
  - 5.4.2 Compact theories . . . 36
  - 5.4.3 Equational theories with a decidable unification problem . . . 37
  - 5.4.4 Summary of the equational case . . . 37
- 5.5 Completeness with respect to Unification problems . . . 37
- 5.6 Other completeness results . . . 38

6 Irreducible solutions **39**

7 Ongoing Research **41**

Introduction

Solving an equation $s = t$ in a term algebra is known as *unification*. In this survey we are interested in solving more general formulas combining equations and involving in particular negation. An algorithm solving such formulas will be called a *disunification* algorithm.

With such a broad (and, up to now, fuzzy) definition, many different works can be seen as special cases of disunification. From the logical point of view, the existence of an algorithm solving an arbitrary first order formula whose only predicate symbol is equality (let's call it an *equational formula*) in an algebra $\mathcal{A}$ shows the decidability of the theory of $\mathcal{A}$ and provides a complete axiomatization of $\mathcal{A}$. The work of logicians such as Mal'cev [Mal71] thus falls into the scope of disunification. There are also completely different approaches: solving certain kinds of equational formulas comes up when compiling functional programs, when dealing with negation in logic programming or when automating inductive proofs in equational specifications. Let us give few examples showing how the problem of solving equational formulas arises in such situations.

Terms are often used to denote values independently of their actual representation. For example, $\{0, s(0), s(s(0)), \ldots\}$ can denote the set of non-negative integers. Then, terms with variables such as $s(x)$ are used to denote (possibly infinite) sets of values: $s(x)$ stands for $\{s(0), s(s(0)), \ldots\}$. Such denotations are used in programming languages. For example, assuming that $+$ and $*$ are already defined, the factorial function may be defined by the two equations

$$\left\{ \begin{array}{rcl} fact(0) & = & 1 \\ fact(x+1) & = & (x+1) * fact(x). \end{array} \right.$$

Another example is logic programs. For example, the predicate EVEN may be defined by the two Horn clauses

$$\left\{ \begin{array}{rcl} \mathrm{EVEN}(0) & \leftarrow & \\ \mathrm{EVEN}(s(s(x))) & \leftarrow & \mathrm{EVEN}(x). \end{array} \right.$$

Then, proving properties of programs as well as computing in a high level programming language which actually uses such notations will involve symbolic manipulation of terms with variables. A famous example is *unification* of two terms: the intersection of the sets denoted by two terms s and t which share no variable is the set denoted by u, if u is a most general common instance of s and t. To unify s and t means to find such a u or, equivalently, to solve the equation $s = t$.

However, it turns out that terms alone present a lack of expressiveness. In particular, the complement of a set of terms denoted by a term with variables cannot always be represented by a finite set of terms (see [LM87]). For example, it is not possible to find a finite set of terms whose instances are all terms that are not instances of $f(x, x)$.

This lack of expressiveness was encountered in many situations involving symbolic manipulations on terms. In order to overcome this difficulty, researchers had to study more general formulas involving equations and other logical connectives. Such formulas are particular cases of equational formulas. Because of the introduction of negation, solving such formulas was coined "disunification."

To "solve" an equational formula means to turn it into an equivalent set of simpler (in a sense which has to be defined) formulas called *solved forms*. For example, in the unification case, the formula to be solved is an equation $s = t$ and the solved forms will be expressions $x_1 = t_1 \wedge \ldots \wedge x_n = t_n$ which actually denote a most general unifier of $s = t$: a most general instance of s and t is obtained by replacing in (say) s the x_i's with the t_i's.

Another example is the famous problem of *sufficient completeness*:

Example 1 Assume we want to define addition on integers modulo 2 by the following rules:

$$\left\{ \begin{array}{rcl} s(s(0)) & \rightarrow & 0 \\ x+0 & \rightarrow & x \\ 0+x & \rightarrow & x \\ x+x & \rightarrow & 0 \end{array} \right.$$

An important question is indeed to decide whether $+$ is indeed completely defined by these rules. By "completely defined" we mean that every expression constructed on $0, s, +$ can effectively be reduced to a term that does not contain any occurrence of $+$. Introduced by Guttag and Horning [GH78], this problem is called "sufficient completeness." It is related to correctness in algebraic specifications and to automated proofs by induction.

For solving such problems, a simple idea is to look for a counter example: $+$ is not completely defined if we can find two terms x_1 and x_2 built on $0, s, +$ that are irreducible (i.e., no rule applies to them) and such that no rule applies to $x_1 + x_2$ itself. This last statement can be reformulated as follows: $x_1 + x_2$ is different from every left hand side of a rule whatever values the variables of the left hand sides take. This problem can be stated: find x_1, x_2 such that

$$\forall x\ (x_1 + x_2 \neq s(s(0)) \wedge x_1 + x_2 \neq x + 0 \wedge x_1 + x_2 \neq 0 + x \wedge x_1 + x_2 \neq x + x) \tag{1}$$

Hence, + is completely defined iff (1) has no irreducible solution. This is an equational formula.

As we have seen in the above example, equational formulas may involve quantifiers and we may be interested in irreducible solutions. This slightly differs from the unification case. Actually, depending on the application at hand, syntaxes as well as semantics may differ. For example, the syntax may allow quantification [KL87, CL89, Mah88a] or not [Col84, Bür88]. The semantics may consider one particular algebra (e.g. the Herbrand Universe [LM87] or the algebra of rational trees [Col84] or irreducible trees [Com89]) or a class of algebras [Mal71, MSK90].

The definition of a solved form may also differ, depending on the application at hand. Indeed, we may be interested in deciding the validity (then solved forms are just "true" and "false") or in interpretations in which the formula holds or even in problems for which solved forms are of a special kind.

Finally, syntax, semantics and solved forms specify what disunification is supposed to do but not how to achieve it. Of course, the algorithmic part will again depend on the syntax, the semantics and the solved forms.

Nevertheless, it can always be stated as follows:

- First, find a set of (first order) axioms $\mathcal{T}$ that hold in the model(s) we consider. ($\mathcal{T}$ has to be large enough for achieving the further steps).
- Secondly, design a set of rules $\mathcal{R}$ (for the transformation of equational formulas) that can be proved to be correct using $\mathcal{T}$.[1]
- Third, design a control C on $\mathcal{R}$ such that the application of the rules satisfying C terminates and such that irreducible formulas are in solved form. Then, a disunification algorithm is given by any deterministic control C' strengthening C.
- Finally, if the syntax allows arbitrary equational formulas and if the validity of a solved form is trivially decidable (this should be required in the definition of a solved form), then termination of $\mathcal{R}$ implies the decidability of the first order theory of $\mathcal{T}$. If, moreover, the solved forms of any closed equational formula is either "true" or "false," this shows the completeness of the theory.

A simple example for the first three steps is (syntactic) unification in finite trees: as shown e.g. in the survey on unification (this volume, chapter 8) or in [LMM86], it simply consists in a strategy for the use of the axioms for syntactic equality recalled in Figures 1, 2.

This paper aims at giving an overview of the main results on disunification as well as their applications. Due to the multiplicity of the different frameworks, I choose to organize the paper by first reviewing in section 1 the several syntaxes and in section 2 the several semantics that one can find in the literature. In section 3 I review possible solved forms. In each section, I aim at giving general concepts capturing the different approaches. Therefore, each section will start with definitions and then give a list of examples showing how to express well known problems using these definitions.

Then, following the above remarks we propose axiomatizations of various models in which disunification has to take place and give an example of a possible control (section 4). When defining the sets of axioms, I mention in passing known results on the theory they define. In fact, these results can be derived from the termination and completeness results of section 5: this is the central part of the paper. After a short introduction to the problem of finding irreducible solutions in section 6, I conclude by a list of open questions.

[1] It is possible to skip the first step and directly prove the correctness of $\mathcal{R}$ in the models we consider. However, the first step is necessary if we want to have a feed-back on the axiomatization of the models.

Equality axioms

Reflexivity

$$\forall s,\ s = s$$

Symmetry

$$\forall s, t,\ s = t \Leftrightarrow t = s$$

Transitivity

$$\forall s, t, u,\ s = t \wedge s = u \Rightarrow t = u$$

Compatibility

$$\forall t_1, \ldots, t_n, u_1, \ldots, u_n,\ \ t_1 = u_1 \wedge \ldots \wedge t_n = u_n \ \Rightarrow\ f(t_1, \ldots, t_n) = f(u_1, \ldots, u_n)$$

Figure 1: Equality axioms

Finite Trees Axioms

Decomposition

$$\forall t_1, \ldots, t_n, u_1, \ldots, u_n,\ f(t_1, \ldots, t_n) = f(u_1, \ldots, u_n) \Rightarrow t_1 = u_1 \wedge \ldots \wedge t_n = u_n$$

Clash

$$\forall t_1, \ldots, t_n, u_1, \ldots, u_m,\ g(t_1, \ldots, t_n) \neq f(u_1, \ldots, u_m) \qquad \text{If } f \neq g$$

Occur Check

$$\forall s, t,\ s \neq t[s] \qquad \text{If } t[s] \text{ is not identical to } s$$

Figure 2: Finite Trees axioms

1 Syntax

We recall in the first paragraph some well known definitions. The reader familiar with terms and substitutions can skip it. For a detailed exposition, see [DJ90].

1.1 Basic Definitions

F will denote a non-empty set of function symbols (with typical elements $f, g, h, \ldots, a, b \ldots$). To each function symbol f is associated a non-negative integer n called its *arity*. X will denote an infinite set of symbols, disjoint from F (with typical elements $x, y, z, \ldots$) called *variables*. Variables are considered as having arity 0.

A *position* is an element of $\mathbf{N}^*$, the monoid of finite sequences of non-negative integers with ϵ as neutral element and concatenation $\cdot$ as the binary composition. $\mathbf{P} \subseteq \mathbf{N}^*$ is *prefix closed* iff, for all $p \in \mathbf{P}$, $p = q \cdot r$ implies $q \in \mathbf{P}$. The *successors* (or *sons*) of a position p in a set of positions $\mathbf{P}$ are the positions q such that $q = p \cdot i \in \mathbf{P}$ for some integer i.

A *term* (or finite tree) over F and X consists of a finite non-empty prefix-closed set of positions $Pos(t)$ together with a mapping t from $Pos(t)$ into $F \cup X$ such that, if f has arity n, then $t(p) = f$ implies that the successors of p in $Pos(t)$ are $p \cdot 1, \ldots, p \cdot n$.

$T(F, X)$ is the set of terms constructed on X and F (with typical elements $s, t, u, v \ldots$). Equality on trees will be denoted by $\equiv$. A *ground term* is a term in $T(F, \emptyset)$. This set is also written $T(F)$.

A *substitution* σ is a mapping from X into $T(F, X)$ which is the identity but on a finite subset of X called the domain of σ. A substitution whose domain is $\{x_1, \ldots, x_n\}$ and such that, for all i, $x_i\sigma \equiv t_i$ will be denoted by $\{x_1 \to t_1; \ldots; x_n \to t_n\}$. A substitution can be extended in a standard way to the whole set of terms.

The subterm of t at position $p \in Pos(t)$ is denoted t/p. The replacement by a term t in a term u at position $p \in Pos(u)$ is written $u[t]_p$. Sometimes the subscript p is dropped when it is not relevant; in order to express that t is a subterm of u, we write $u[t]$ and call $u[\ ]$ a *context*. Such a context is *non-trivial* if $u[t] \not\equiv t$.

1.2 Equational Formulas

An *equation* is an unordered pair of terms s, t denoted by $s = t$. Let E be a (finite) set of equations. The congruence $=_E$ generated by E is the smallest equivalence relation on $T(F, X)$ such that, for every substitution σ, every context $t[\ ]$, and every equation $u = v \in E$, $t[u\sigma] =_E t[v\sigma]$. The *equational theory* of E is the set of equations $s = t$ such that $s =_E t$. (Equivalently, this is the set of equations that can be inferred from E and the set of equality axioms in Figure 1).

An *equational formula* is an arbitrary first order formula, built on the logical connectives $\wedge$, $\vee$ and $\neg$ and on the quantifiers $\forall$ and $\exists$, whose atoms are either equations or the special symbol $\top$. We choose these logical connectives (instead of, e.g. exclusive or and $\wedge$) in order to have a natural statement of the equality axioms given in Figure 1. An expression $\neg(s = t)$ is also written $s \neq t$ and called a *disequation*. In the same way, $\neg\top$ is written $\bot$. Equational formulas may be viewed as terms in an algebra $\mathcal{EF}$. As usual, binary connectives are used in infix notation.

For any syntactic object $\mathcal{O}$ (a term, a set of terms, an equation, a formula,etc...), $Var(\mathcal{O})$ will denote the set of (free) variables in $\mathcal{O}$.

A *quantifier free equational formula* (qfef for short) is a formula built on the logical connectives $\wedge$, $\vee$, $\neg$, $\top$ and whose atoms are equations.

1.3 Syntactic Equivalence of qfef

As usual, we do not want to write e.g. parentheses around $\phi_1 \wedge \phi_2$ in an expression $\phi_1 \wedge \phi_2 \wedge \phi_3$. It is indeed possible to drop them because $\wedge$ turns out to be associative. So, we define a syntactical equivalence $\sim$ on qfefs which captures such identities.

What should $\sim$ include ? Of course, two equivalent formulas (under $\sim$) must have the same models ($\sim$ has to be embedded into the semantic equivalence $\approx$ which will be defined later). $\sim$ has also, at least, to be decidable since we want to deal with representatives modulo $\sim$.

We give in Figure 3 a set of identities and a set of rules for normalizing qfef. The identities consist in associativity, commutativity and symmetry axioms. Let $\sim_I$ be the congruence they define on qfef. The rules define a rewrite system modulo the identities[2]. Let $\to_\mathcal{E}$ be the corresponding reduction relation. The rewrite system is convergent (modulo $\sim_I$). Let $\sim$ be the reflexive, symmetric, transitive closure of $\to_\mathcal{E}$. Then $\phi_1 \sim \phi_2$ iff their two normal forms w.r.t $\to_\mathcal{E}$ are equal modulo $\sim_I$.

[2]We refer to [DJ90] for definitions related to rewrite systems.

Identities

$$
\begin{array}{rcl}
s = t & \sim_I & t = s \\
\phi_1 \wedge \phi_2 & \sim_I & \phi_2 \wedge \phi_1 \\
\phi_1 \wedge (\phi_2 \wedge \phi_3) & \sim_I & (\phi_1 \wedge \phi_2) \wedge \phi_3 \\
\phi_1 \vee \phi_2 & \sim_I & \phi_2 \vee \phi_1 \\
\phi_1 \vee (\phi_2 \vee \phi_3) & \sim_I & (\phi_1 \vee \phi_2) \vee \phi_3
\end{array}
$$

Rewrite Rules

$$
\begin{array}{rcl}
s = s & \to & \top \\
\neg(s = t) & \to & s \neq t \\
s \neq s & \to & \bot \\
(a \wedge b) \vee c & \to & (a \vee c) \wedge (b \vee c) \\
a \wedge a & \to & a \\
a \vee a & \to & a \\
a \wedge \top & \to & a \\
a \vee \top & \to & \top \\
a \wedge \bot & \to & \bot \\
a \vee \bot & \to & a \\
\neg \top & \to & \bot \\
\neg \bot & \to & \top \\
\neg\neg a & \to & a \\
\neg(a \wedge b) & \to & \neg a \vee \neg b \\
\neg(a \vee b) & \to & \neg a \wedge \neg b \\
a \wedge (a \vee b) & \to & a
\end{array}
$$

Figure 3: qfef normalization

It should be noted that, by definition, $s = t \sim t = s$ which reflects the commutativity of $=$ stated in the definition of an equation. We also include $s = s \sim \top$ and $s \neq s \sim \bot$ in the syntactic equivalence. Such equations (respectively disequations) are called *trivial*.

Finally, note that rules of Figure 3 transform a qfef into a kind of conjunctive normal form. We could choose disjunctive normal forms as well : this choice is arbitrary.

1.4 Elementary Formulas

Syntactic equivalence is slightly extended in order to handle formulas involving quantifiers: $\exists x \exists y : \phi \sim \exists y \exists x : \phi$, for example. Actually, an expression $\exists x_1 \ldots \exists x_n \phi$ is written $\exists \vec{x} : \phi$, where $\vec{x}$ is the set of x_is. (This notation assumes the properties of finite sets; e.g., $\forall x \forall y \forall x \phi$ denotes the same formula as $\forall y \forall x \phi$.) Unless explicitly stated, $\vec{x}$ is assumed to denote a non-empty set.

It is possible to extend $\sim$ to equational formulas and to incorporate some (other) quantifier rules. However, this would lead to complicated rules. Actually, we will first only consider a special case of equational formulas called *elementary formulas* which contain at most two nested quantifiers. The general case of equational formulas will follow from this particular one.

An elementary formula is an equational formula having the following form:

$$\bigvee_{j \in J} \exists \vec{z_j}\ \forall \vec{y_j}\ \phi_j$$

where ϕ_i is a qfef, J is a finite set of subscripts and $\vec{z_j}, \vec{y_j}$ are (possibly empty) sets of variables. Note that elementary formulas may contain free variables.

The relation $\to_{\mathcal{E}}$ is extended to elementary formulas, preserving the compatibility with formula constructors. In the following, we only need to reduce an elementary formula to an equivalent elementary formula. Therefore, it is not necessary to enrich $\sim$ by adding other rules.

1.5 Particular Cases

Here we refer to particular problems whose syntax is restricted to a subset of equational formulas. This does not mean that the results for such problems must be particular cases of general results. For the moment, we only compare the syntax.

1.5.1 Unification problems

A *unification problem* is an equational formula without quantifiers nor negation. A unification problem is (equivalent to) a finite disjunction

$$\bigvee_{j \in J} \exists \vec{z_j} : (t_{j,1} = u_{j,1} \wedge \ldots \wedge t_{j,n_j} = u_{j,n_j})$$

Example 2 The formula

$$(x_1 = a \wedge x_2 = a) \vee (\exists z : x_1 = a + z + z \wedge x_2 = a + z)$$

is a unification problem. (It denotes a complete and minimal set of unifiers of the two terms $x_1 + a$ and $x_2 + x_2$ modulo associativity and commutativity of $+$.)

Note that unification usually consists in solving *one* equation. Therefore, unification is a particular case of solving unification problems.

1.5.2 Word problems

A *word problem* is a formula $\forall \vec{x} : s = t$ where $\vec{x} = Var(s,t)$. Such problems have been widely studied for many years (see e.g. [DJ90] for a survey). Word problems are, of course, particular cases of equational formulas.

Example 3 The formula

$$\forall x, y : I(x * y) = I(y) * I(x)$$

is a word problem. (It holds in the theory of groups $(G, *, e)$ where I is the inverse function).

The previous examples are, in some sense, degenerated ones since they do not involve one of the main feature we are interested in : negation. However, it is useful to keep in mind that any general disunification algorithm will include algorithms for unification and word problems.

1.5.3 Equations and disequations in PROLOG II

In PROLOG II [Col84], systems of equations and disequations are used to express restrictions on the applicability of the rules. They also enable the representation of other kinds of answers of the logic programs. In this case, the formulas are conjunctions of equations and of disjunctions of disequations:

$$\bigwedge_{i=1}^{n} t_i = u_i \wedge \bigwedge_{i=1}^{m} (\bigvee_{j=1}^{m_i} s_{i,j} \neq v_{i,j})$$

Such formulas are again (elementary) equational formulas : they are a particular case of quantifier free equational formulas.

1.5.4 Complement problems

Terms with variables often denote the set of all their ground instances. For example, assuming that each ground term represents a fact, a term with variables will represent a (possibly infinite) set of facts. (See for example [LM87]). Another example is the representation of infinite sets of states in FP2 [SJ89]. A third example is the "definition by pattern" in functional languages [Tho86]. A fourth example is logic programming when only Herbrand models are considered. Finally, complement problems were also introduced for the study of some properties of algebraic specifications [Thi84, Com86]. In all these examples, some operations on finite sets of terms with variables, including the complement, are required:

- In the first example, the set of terms to complement is the set of "counter-examples" which are facts (possibly infinitely many) known to be false.

- In the language FP2, term rewriting systems are used for the representation of infinite transition systems. Then, the complement of the left hand sides gives all states from which it is not possible to perform a transition. Therefore, computing the complement of a finite set of terms is used in the temporal analysis of communicating processes.

- In order to compile efficiently a function defined by patterns, we want to take into account the (assumed) sequential application of the rules. More precisely, for compiling the second rule, we know that the first one cannot be applied. Hence, we have some knowledge about the ground terms on which the second rule should be tried : they are in the complement of the first left hand side. This is effectively used in, for example [Sch88a]. Another kind of term complements is used in [Lav87, Lav88, PS90] in order to produce a non-ambiguous set of patterns .

- There are basically two kinds of applications of complement problems in logic programming. Both rely on the fact that the declarative semantics of a logic program is its least Herbrand model. First, it is possible to statically deduce from the program some knowledge about the negation of the defined predicate. For example, if the predicate EVEN is defined by

$$\left\{ \begin{array}{rcl} \mathrm{EVEN}(0) & \leftarrow & \\ \mathrm{EVEN}(s(s(x))) & \leftarrow & \mathrm{EVEN}(x) \end{array} \right.$$

  It is possible to (automatically) deduce a definition of ODD (an approximation, exact in this case, of ¬EVEN)

$$\left\{ \begin{array}{rcl} \mathrm{ODD}(s(0)) & \leftarrow & \\ \mathrm{ODD}(s(s(x))) & \leftarrow & \mathrm{ODD}(x) \end{array} \right.$$

  As shown, for example, in [BMPT90, Com88b] this automatic computation relies on the complement of the left hand sides defining EVEN. Another use of complement problems is developed by Kunen [Kun87a]. He uses them to represent infinite sets of answers to negative queries when dealing with negation as failure.

For these reasons, we want to be able to compute a suitable representation of all the ground instances of a given finite set of terms which are not instances of another given finite set of terms. Let $[\![t]\!]$ denote the set of all ground instances of the term t. $[\![t_1,\ldots,t_n]\!]$ is then the union of the sets $[\![t_i]\!]$. The problem is to compute $[\![t_1,\ldots,t_n]\!] - [\![u_1,\ldots,u_m]\!]$ for any terms $t_1,\ldots,t_n,u_1,\ldots,u_m$. Without loosing generality, it is possible to assume $n=1$ and $\vec{z} = Var(t_1)$ disjoint from $\vec{y} = Var(u_1,\ldots,u_n)$. Then the problem can be expressed using the following formula:

$$\exists\vec{z},\forall\vec{y} : x = t \wedge \bigwedge_{i=1}^{m} x \neq u_i$$

This is again a particular case of (elementary) equational formulas.

Unfortunately, as shown in [LM87], it is not always possible to express the solutions of such a formula only by means of sets $[\![t]\!]$ (i.e. without negation). In other words, the class of finite unions $[\![t_1]\!] \cup \ldots \cup [\![t_n]\!]$ is not closed under complement. That is the reason why *constrained terms* have been introduced in [Com88b]. A constrained term is a pair $t \,\&\, d$, where t is a term and d is a conjunction of non-trivial disequations whose variables are those of t. Then $[\![t \,\&\, d]\!]$ is the set of all ground instances of t which satisfy the constraint d. Now, the class of finite union of constrained terms is closed under complement. The complement of a constrained term $s \,\&\, d$ into a constrained term $t \,\&\, \delta$ is expressed by the following equational formula:

$$\exists\vec{z}\,\forall\vec{y} \;:\; x = t \wedge \delta \wedge (x \neq s \vee \neg d)$$

where $\vec{y} = Var(s)$, $\vec{z} = Var(t)$ and x is a free variable.

Therefore, we can define a complement problem as being equivalent to a formula

$$\exists\vec{z}\forall\vec{y} :\; x = t \wedge \delta \wedge (x \neq s_1 \vee \neg d_1) \wedge \ldots \wedge (x \neq s_m \vee \neg d_m)$$

where $\vec{z} \cap \vec{y} = \emptyset$, $\vec{z} = Var(t)$, $\vec{y} = Var(s_1,\ldots,s_m)$, x is free, $Var(\delta) \subseteq Var(t)$, $Var(d_i) \subseteq Var(s_i)$. More complicated complement problems, related to complete sets of positions are studied in [Com88b]. The *projection* of complement problems is also studied in [Nic87].

In example 1, we were also faced to a complement problem. Indeed, the formula

$$\forall x \;:\; x_1 + x_2 \neq x + 0 \wedge x_1 + x_2 \neq 0 + x \wedge x_1 + x_2 \neq x + x$$

expresses the complement of $[\![x+0]\!] \cup [\![0+x]\!] \cup [\![x+x]\!] \cup [\![s(s(0))]\!]$ (i.e., the left hand sides of the rules) into $[\![x_1+x_2]\!]$.

Some complement problems are also studied in [PS90] with a slightly different definition. In their framework, a constant ($\bullet$) plays a special role: its intended meaning is a non-terminating evaluation (or a non-evaluated term). Compiling (in an optimal way) pattern matching, taking care of possible looping evaluations, requires to solve another kind of complement problems.;
$t' \diamond t$ now represents all instances $t'\sigma$ of t' (in $T(F \cup \{\bullet\}, X)$) such that $t'\sigma \not\sqsubseteq t$ where $\sqsubseteq$ extends the subsumption quasi-ordering to terms which may contain non-evaluated subterms (represented by $\bullet$). Note that such complement problems cannot be expressed using pure equational formulas. Simplifications of such complement problems as well as properties of the corresponding constrained terms are studied in [PS90].

Classical complement problems are also studied and used by L. Puel in [Pue87]. Here, the purpose is to decompose terms into patterns of a given (unavoidable) set in order to compare them using a pattern path ordering.

1.6 Related Work Using a Similar Formalism

Solving equational formulas is closely related to the study of decidable theories. However, we do not intend to survey the decidable theories (instead, see e.g. [Rab77]). Actually, we will only consider syntactic interpretations of our formulas, i.e. interpretations in algebras of trees.

In this framework, Mal'cev studied in his landmark paper [Mal71] classes of (term) algebras that are axiomatizable. Kirchner and Lescanne [KL87] followed by Comon and Lescanne [CL89], Kunen [Kun87b, Kun87a] and Maher [Mah88a] studied general equational formulas.

1.7 Extensions

1.7.1 Many-Sorted Signatures

It is easy to extend the definitions to *many-sorted signatures.* All results presented here can be extended to this case as shown in [CL89, Mah88a]. In this survey, we preferred to avoid unessential problems by restricting our attention to one-sorted signatures.

1.7.2 Order-Sorted Signatures

Equational problems were also studied for *order-sorted signatures.* Order-sorted signatures (as defined in e.g. [GM87, SNMG87]) generalize many-sorted signatures, allowing a function symbol to have finitely many profiles and the sorts to be ordered (interpreted as inclusion between the corresponding domains). Order-sorted equational problems are investigated in [Com88b, Com90b]. Let us precise a little bit more what we call an equational formula in this framework.

Given a finite set of sorts S, the set of *sort expressions* $\mathcal{SE}$ is defined as the least set satisfying:

- a sort is a sort expression
- if $\underline{s}_1, \ldots, \underline{s}_n$ are sort expressions, then every propositional formula over the propositional variables $\underline{s}_1, \ldots, \underline{s}_n$ is a sort expression
- if $\underline{s}_1, \ldots, \underline{s}_n$ are sort expressions, and $f \in F$, then $f(\underline{s}_1, \ldots, \underline{s}_n)$ is a sort expression

For every sort expression $\underline{s}$, "$\in \underline{s}$" is a unary predicate symbol used in postfix notation. Then, an equational formula is a first order formula over the predicate symbols $=$ and $\in \underline{s}$. (Note that there is in general infinitely many such predicate symbols).

1.7.3 Introduction of Inequalities

Also interesting is the introduction of the symbol $\geq$ (inequations). Formulas involving such symbols are studied in [Ven87, LM89, Com90c, JO91, Tul91] and many others. $\geq$ differs from $=$ in that, given an F-algebra, it has several distinct possible interpretations, as we shall discuss in the next section.

1.7.4 Arbitrary first order formulas

The case of formulas involving arbitrary predicate symbols is investigated in [Com88b, Lug89]. However, results can be derived only for a particular subclass of formulas. Nevertheless, this is sufficient for deriving a complete deduction procedure for logic programs where negative goals and general programs are allowed [Lug89]. It is also shown in [Lug89] how to use these results for the transformation of logic programs.

1.8 Constraint Languages

A *constraints system* is a logic which enjoys two particular properties:

- Only one model is considered
- There is a decision algorithm for the satisfiability problem in this logic.

The constraints domain may be integers (Presburger arithmetic), polynomials over real numbers, linear inequalities over rational numbers, ... , but they may also be *symbolic constraints*. Symbolic constraints are formulas interpreted in a fixed "syntactic" model, i.e. an algebra of terms. That is where disunification comes in: each time we have a disunification algorithm, we may derive a constraints system which meets the above two requirements. A very popular constraints system is conjunction of equations, interpreted in the Herbrand Universe: it is convenient to see unification as constraints solving in logic programming since this avoids applying substitutions (see also chapter 8). But, as we are going to see, it is also possible to use equational formulas as a constraints system.

Adding constraints systems may enhance both expressiveness and efficiency of logic programs while keeping safe semantical foundations and a simple operational semantics. "Logic program" should be understood here in a broad sense: a logic program may be a (constrained) set of clauses [JL87] or a (constrained) set of equations [KK89], or any other set of formulas in some logic. Enhancement of expressiveness is obvious, but some particular applications are shown in [KK89]. For example, it is possible to deal with AC function symbols during Knuth-Bendix completion in cases which are untractable without constraints.

Enhancement of efficiency stems from the ability to express some control or *strategy*: it is possible to use constraints in order to prevent the application of some rules. A typical example of such kind of use of symbolic constraints is the introduction of inequalities, expressing an ordered strategy. Another reason why adding constraints may enhance efficiency is the use of some dedicated algorithms for solving constraints which would have been solved using the general rules otherwise. (See e.g. [DSH87, Col88].)

2 Semantics

2.1 F-algebras

(see [DJ90] for complete definitions)

An F-algebra $\mathcal{A}$ (or a *model* of F) consists in a non-empty set, denoted $D_{\mathcal{A}}$, together with a family $\{f_{\mathcal{A}} \mid f \in F\}$ of mappings such that, if f has arity n, then $f_{\mathcal{A}}$ is a mapping from $D_{\mathcal{A}}^n$ into $D_{\mathcal{A}}$. For clarity, the equality on a set $\mathcal{A}$ will be denoted $=_{\mathcal{A}}$. Given an F-algebra $\mathcal{A}$, an $\mathcal{A}$-*assignment* is a mapping σ from a set of variables $Dom(\sigma) \subseteq X$ into $\mathcal{A}$. Such a mapping can be extended in a unique way into an homomorphism from $T(F,X)$ into $\mathcal{A}$ which is again denoted by σ.

When there is no ambiguity on the mappings $f_{\mathcal{A}}$, their definition will be omitted. $T(F)$ and $T(F,X)$ are examples of F-algebras. If $=_E$ is a congruence relation generated by a finite set of equational axioms E, $T(F)/_{=_E}$ and $T(F,X)/_{=_E}$ are also F-algebras. $IT(F)$, the set of (possibly) *infinite trees* over F is also an F-algebra. An infinite tree is defined like a finite one, except that its set of positions may be infinite. $RT(F)$, the set of *rational trees* over F is the set of infinite trees having only finitely many distinct subtrees. It is again an F-algebra.

2.2 Solutions of an equational formula

An equational formula is assumed to be given together with a finite set of *unknowns* $\mathcal{U} \subseteq X$. This set is understood as the set of variables with respect to which a formula has to be solved. $\mathcal{U}$ is always assumed to contain the free variables of the formula we consider. As we will see, the transformations of formulas will never add free variables. On the opposite, some free variables may disappear during the transformations. Therefore, $\mathcal{U}$ may be larger than the set of free variables. However, since $\mathcal{U}$ does not change during transformations, it is generally not relevant and will sometimes be omitted.

Definition 1 (Solutions) *Let $\mathcal{A}$ be an F-algebra. For any equational formula ϕ, $\mathcal{S}(\phi,\mathcal{A},\mathcal{U})$, called the set of $\mathcal{A}$-*solutions *is the set of $\mathcal{A}$-assignments that satisfies*[3]*:*

$$\begin{array}{rcl}
\mathcal{S}(\top,\mathcal{A},\mathcal{U}) & = & \{\sigma \mid Dom(\sigma) = \mathcal{U}\} \\
\mathcal{S}(s=t,\mathcal{A},\mathcal{U}) & = & \{\sigma \mid s\sigma =_{\mathcal{A}} t\sigma \text{ and } Dom(\sigma) = \mathcal{U}\} \\
\mathcal{S}(\phi_1 \wedge \phi_2,\mathcal{A},\mathcal{U}) & = & \mathcal{S}(\phi_1,\mathcal{A},\mathcal{U}) \cap \mathcal{S}(\phi_2,\mathcal{A},\mathcal{U}) \\
\mathcal{S}(\exists x\ \phi,\mathcal{A},\mathcal{U}) & = & \{\sigma_{|\mathcal{U}-\{x\}} \mid \sigma \in \mathcal{S}(\phi,\mathcal{A},\mathcal{U}\cup\{x\})\} \\
\mathcal{S}(\neg\phi,\mathcal{A},\mathcal{U}) & = & \mathcal{S}(\top,\mathcal{A},\mathcal{U}) - \mathcal{S}(\phi,\mathcal{A},\mathcal{U})
\end{array}$$

The definition for other logical connectives can be derived from the above ones. We write $\phi \approx_{\mathcal{A}} \psi$ when $\mathcal{S}(\phi,\mathcal{A},\mathcal{U}) = \mathcal{S}(\psi,\mathcal{A},\mathcal{U})$ for some set $\mathcal{U}$ containing the free variables of ϕ and ψ (or, equivalently, for *any* set $\mathcal{U}$ containing the free variables of ϕ and ψ). The subscript $\mathcal{A}$ is omitted when the equivalence holds for every algebra $\mathcal{A}$. It is easy to see that $\phi \approx \psi$ if $\phi \sim \psi$.

When $\phi \approx_{\mathcal{A}} \top$ we also write $\mathcal{A} \models \phi$. When this relation holds for any algebra $\mathcal{A}$ in a class of algebras $\mathcal{K}$, we write $\mathcal{K} \models \phi$. Finally, when $\mathcal{K}$ is the class of algebras $\mathcal{A}$ such that $\mathcal{A} \models \mathcal{T}$ for a given set of equational formulas $\mathcal{T}$, we write $\mathcal{T} \models \phi$ instead of $\mathcal{K} \models \phi$.

Let A be a subset of the F-algebra $\mathcal{A}$. Definition 2.1 can be extended to *solutions in* A: a solution in A is a $\mathcal{A}$-solution such that every variable in $\mathcal{U}$ is mapped to an element of A.

2.3 Examples

2.3.1 $\mathcal{A} = T(F)$ or $\mathcal{A} = T(F,X)$

This is the interpretation used for complement problems (for example [Sch87, Sch88b], [Thi84], [BMPT90], [LM87], [Com86] and others). This interpretation is also considered in e.g [Ven87], [CL89], [Mah88a], [Bür88], [Kun87b, Kun87a, Nic87].

It appears that the main point for such interpretations is to know whether F is finite or infinite: the rules for solving equational formulas will be different in the two cases. In particular, as shown in e.g. [Mah88a] (see also section 4), the axiomatizations of $T(F)$ are different in the two cases.

The interpretation $\mathcal{A} = T(F,X)$ is nothing but a particular case of $\mathcal{A} = T(F)$ since it is sufficient to consider all (free) variables as new constants. Note however that the set of function symbols then becomes infinite. Therefore, we will no longer consider the case $\mathcal{A} = T(F,X)$.

[3] $\sigma_{|V}$ denotes the restriction of σ to $V \subseteq X$.Bounded variables in a formula ϕ are also assumed to be distinct from variables in $\mathcal{U}$.

2.3.2 $\mathcal{A} = RT(F)$

In the implementation of logic programming, for efficiency reasons, it is better to delay the so-called "occur-check rule" as long as possible. This is the reason why the underlying semantics of Colmerauer's systems of equations and disequations [Col84] involves the algebra of rational trees $RT(F)$.

Equational problems with interpretations in rational trees were studied in [Mah88a, MS90].

2.3.3 $\mathcal{A} = IT(F)$

The interpretation of equational formulas in the algebra of infinite (and not necessarily rational) trees was also studied in [Mah88a] with the conclusion that a formula holds in $IT(F)$ iff it holds in $RT(F)$.

2.3.4 $T(F)$-solutions in $A = NF$

Let NF be the set of terms in $T(F)$ which are irreducible with respect to a given term rewriting system $\mathcal{R}$. As we have seen in example 1, the problem arises to find the solutions in NF of a complement problem. Such interpretations are studied in [Com89, Com88b, Com88a].

2.3.5 $\mathcal{A} = T(F)/_{=_E}$

Let E be a finite set of (implicitly universally quantified) equational axioms. The problem of E-unifying two terms s and t consists in finding all substitutions σ such that $s\sigma =_E t\sigma$. This is a generalization of syntactic unification: syntactic unification corresponds to the case $E = \emptyset$. Of course, general E-unification is undecidable. Some complete strategies for this problem where proposed ([GS87] for example. See also the survey on unification in this volume). However, E-unification is decidable for some special theories such as AC-theories where E consists of a set of commutativity and associativity axioms for some functions symbols in F [Sti81] (see also the survey on unification in this volume).

Since such cases are of practical interest, the problem arises to study equational problems in the algebra $T(F)/_{=_E}$ (or $T(F,X)/_{=_E}$). Indeed, finding all substitutions σ such that $s\sigma =_E t\sigma$ can be restated as finding all substitutions σ that are solutions of $s = t$ in $T(F,X)/_{=_E}$.

The study of equational formulas in such interpretations is at its start. HJ. Bürckert and M. Fernandez study purely existential formulas in such interpretations ([Bür88, BB89, Fer91]) while H. Comon studies equational formulas for particular sets of axioms [Com88b, Com91a].

2.3.6 Order-Sorted Signatures

Here, we consider equational formulas as they are defined in section 1.7.2. Therefore, we have to define the semantics of sorts expressions and the semantics of the predicate symbols $\in \underline{s}$. The model is defined by an order-sorted signature, i.e. a finite bottom-up tree automaton whose states are in the set of sorts S. Then a sort expression $\underline{s}$ denotes a regular tree language $[\![\underline{s}]\!]$ in an obvious way. A predicate symbol $\in \underline{s}$ is interpreted as the regular tree language $[\![\underline{s}]\!]$.

2.3.7 Classes of algebras

A different problem is to decide whether an equational formula holds in a class of algebras. For example, Mohan & al. [MSK90] consider the problem of deciding whether $\phi \models \psi$ where ϕ and ψ are purely universally quantified equational formulas (i.e. qfef whose variables are all universally quantified). They indeed try to generalize equational reasoning and completeness results in equational logic to this logic involving disequalities.

Mal'cev in [Mal71] and Maher in [Mah88a] study the validity of an arbitrary formula in some restricted classes of algebras: the *locally free* algebras. Mal'cev also proposed some extensions to classes of permutative algebras.

Nelson and Open in [NO80] show the decidability of satisfiability for the purely existential equational formulas.

2.3.8 Handling inequalities

Considering general equational problems which may contain inequalities requires an interpretation for the ordering $\geq$. Although interesting, we do not want to consider special sets of function symbols F. We restrict our attention to "syntactic" interpretations i.e. to algebras of trees or their quotients without any further assumption on F than e.g. finiteness. Interpretations on naturals, integers, real numbers, real closed fields, and so on require specialized techniques and are out of the scope of this paper.

Even when the domain under consideration is the algebra of finite trees, many interpretations of inequality can be considered. For example, inequality is interpreted as the subterm relation in [Ven87, Tul91]. It is interpreted as the matching quasi-ordering in [KK89] and as a recursive path ordering[4] in [Com90a, JO91]. Each interpretation is motivated by some kind of application.

[4] The recursive path ordering is defined in e.g. [Der87]. To each total ordering on F is associated a total recursive path ordering on $T(F)$. Such orderings are used in all implementations of Term Rewriting as well as in Ordered strategies in Automated Deduction. (see e.g. [HR87]).

3 Solved Forms

Solved forms are particular classes of formulas on which the problem to be solved is trivially decidable. As a first example, let us recall what can be used as a solved form in the unification case.

3.1 Solved Forms for Unification

Assume that the problem is a single equation $s = t$ and that we want to solve the problem in finite trees ($\mathcal{A} = T(F)$). Then, a solved form is either $\perp$, $\top$ or a conjunction of equations:

$$x_1 = t_1 \wedge \ldots \wedge x_n = t_n$$

where the x_i's occur only once[5]. This defines an idempotent substitution $\sigma = \{x_1 \to t_1; \ldots; x_n \to t_n\}$ such that $s\sigma \equiv t\sigma$ and $s\theta \equiv t\theta \Rightarrow \exists\phi, \theta = \phi \circ \sigma$. σ is usually called a *most general unifier* of s and t. This is the reason why such solved forms play a central role: every solution can be easily obtained from the solved form.

Let $=_S$ be the strict equality on variables defined by a solved form S: if t_i is a variable, then $x_i =_S t_i$ and $=_S$ is the smallest equivalence relation satisfying this property. The following result can be found in e.g. [LMM86]

Theorem 1 *For every equation $s = t$ there is a system S in solved form such that $S \approx s = t$. Moreover, S is unique up to permutation of variables in the same equivalence class modulo $=_S$.*

In the case of rational trees, solved forms are slightly different since an equation $x = t[x]$ does have solutions. More generally, the following basic result shows the relationship between systems of equations and rational trees (see e.g. [Cou81]):

Theorem 2 *A system $x_1 = t_1 \wedge \ldots \wedge x_n = t_n$ where $x_1, \ldots, x_n$ are variables, $t_1, \ldots, t_n \in T(F, \{x_1, \ldots, x_n\})$ and, for every i, $x_i \not\equiv t_i$, has a unique $RT(F)$-solution.*

Therefore, in the case of rational trees a solved form should be defined as either $\top$, $\perp$ or a conjunction

$$x_1 = t_1 \wedge \ldots \wedge x_n = t_n$$

where $x_1, \ldots, x_n$ are distinct variables and, for every i, if t_i is a variable, then x_i occurs only once.

The following result is given by e.g. G. Huet [Hue76]:

Theorem 3 *For every equation $s = t$ there is a system S in solved form such that $s = t \approx_{RT(F)} S$.*

Note that we do not have a unicity result since the rational tree defined by $x = g(x)$ is also defined by $x = g(g(x))$. However, as in the finite tree case, every solution can be obtained from the $RT(F, X)$-assignment defined by the solved form : a solved form S defines a unique tuple of rational trees $(u_1, \ldots, u_n)$ over $RT(F, Var(s,t) - \{x_1, \ldots x_n\})$ and every $RT(F)$-solution of $s = t$ (w.r.t the unknowns $x_1, \ldots, x_n$) is an instance of $\{x_1 \to u_1; \ldots, x_n \to u_n\}$.

3.2 Basic formulas

The main properties of solved forms can be summarized as

Solvability a solved form has at least one solution

Simplicity every solution can be easily obtained from a solved form

Completeness every equational problem is equivalent (modulo $\approx_{\mathcal{A}}$) to a finite disjunction of solved forms

These three features were ensured in the unification case by the above theorems. It remains however an unprecise point : what does "easily obtained" mean ? Actually, the answer is subjective and it is possible to propose many kind of solved forms depending on it.

We choose the following definition (inspired by [Col84])

Definition 2 *A* basic formula *is either $\top$, $\perp$ or has the form*

$$\exists \vec{z}: \ x_1 = t_1 \wedge \ldots \wedge x_n = t_n \wedge z_1 \neq u_1 \wedge \ldots \wedge z_m \neq u_m$$

where $x_1, \ldots, x_n$ are free variables, $Var(z_1, \ldots, z_m, u_1, \ldots, u_m) \subseteq Var(x_1, \ldots, x_n, t_1, \ldots, t_n)$ and

[5] Other solved forms such as "DAG solved forms" are interesting. See the survey on unification in this volume

- *if $\mathcal{A} = RT(F)$, then, for every index i, $x_1 = t_1 \wedge \ldots \wedge x_n = t_n \wedge z_i = u_i$ is in unification solved form.*
- *if $\mathcal{A} = T(F)$, every variable x_i occurs only once and, for every index j, $z_j \not\equiv u_j$.*

For example, $\exists z : x = g(z) \wedge z \neq g(x)$ is a basic formula if $\mathcal{A} = RT(F)$ but is not a basic formula if $\mathcal{A} = T(F)$. (In the latter case, the problem is equivalent to $\exists z : x = g(z)$ which is a basic formula).

We claim that the requirements for solved forms (Solvability, simplicity and completeness) hold for basic formulas.

Solvability follows the so-called "independence of disequations" lemma proposed by Colmerauer in [Col84]. This lemma has been studied in some further papers. Among them, let me cite [LMM86], [Com88b] where the lemma is extended to the case of some equational theories, [LM89] where the lemma is extended to systems of inequalities and [LM90] where this lemma is set up in a general framework.

The independence of disequations lemma plays a central role not only for showing the existence of a solution but also in the transformation rules of the next sections.

Lemma 1 *Assume that A is an infinite subset of $IT(F)$. Let e be a conjunction of equations. A qfef $e \wedge s_1 \neq t_1 \wedge \ldots \wedge s_n \neq t_n$ has at least one $\mathcal{A}$-solution iff for each i, $e \wedge s_i \neq t_i$ has at least one solution in A.*

The proof of this lemma is elementary. It proceeds e.g. by induction on the number of variables occurring in the qfef.

Corollary 1 *Assume that A is an infinite subset of an F-algebra $\mathcal{A} \subseteq IT(F)$. Then every basic formula (with respect to $\mathcal{A}$) distinct from $\perp$ has at least one solution in A.*

Basic formulas also satisfy the completeness requirement. It is a Corollary of termination and completeness results of section 5.

Theorem 4 *Let $\mathcal{A}$ be one of the following : $T(F)$, $RT(F)$, $IT(F)$. Every equational formula is equivalent modulo $\approx_{\mathcal{A}}$ to a finite disjunction of basic formulas*

Actually, it is possible to prove a stronger result: it is possible to effectively compute the finite disjunction of basic formulas equivalent to a given equational formula (see theorems 5.2, 5.3). When combined with Corollary 3.6, this result has decidability consequences: the theories of $T(F)$, $RT(F)$, $IT(F)$, are recursive. These results will be re-stated when we give the precise set of axioms.

3.3 Solved forms for complement problems

In the case of complement problems, many algorithms were proposed (for example [LM87, Thi84, Com86, Kuc88, MS90]) corresponding to different kinds of solved forms. The main problem is to transform a complement problem P into a unification problem each time there exists a unification problem U such that $P \approx_{T(F)} U$. Lassez and Marriott give in [LM87] necessary and sufficient conditions under which the complement problem associated with $[\![t]\!] - [\![t_1, \ldots, t_n]\!]$ (call it $\mathcal{P}_{t;t_1,\ldots,t_n}$) can be turned into a unification problem. Let us give these conditions in details. $T(F)$ is assumed to be infinite and F to be finite.

A term u is *linear* if no variable occurs more than once in u. A term $t\theta$ is called *restricted* with respect to t if either there is a variable x in t such that $x\theta$ is not linear or there are two distinct variables x_1 and x_2 in t such that $x_1\theta$ and $x_2\theta$ share a variable. Let $\theta_1, \ldots \theta_n$ be respectively the most general unifiers of t and $t_1, \ldots, t_n$[6].Then, there is a unification problem U such that $\mathcal{P}_{t;t_1,\ldots,t_n} \approx_{T(F)} U$ iff there exists $t\phi_1, \ldots, t\phi_n$ such that $[\![t\phi_1, \ldots t\phi_n]\!] = [\![t\theta_1, \ldots, t\theta_n]\!]$ and, for each i, $t\phi_i$ is unrestricted with respect to t. Similar results are shown by Maher and Stuckey in the case of rational trees [MS90].

Let us show an application. A rewrite system is *constructor-based* if there is some subset C of F such that $T(C)$ is the set of irreducible ground terms (symbols in C are called *constructors*). A rewrite system is *left-linear* if every left hand side is linear. Two rewrite systems $\mathcal{R}_1$ and $\mathcal{R}_2$ are *equivalent* if, for every ground term s, either s has no normal form in either system or its normal forms in both systems are identical. The above results show that any ground-convergent constructor-based rewrite system is equivalent to a left linear ground convergent constructor-based rewrite system. This property is important since left-linear rewrite systems are much simpler and enjoy a better operational behaviour than non-linear ones and, on the other hand, constructor-based rewrite systems are often used in equational programming (e.g. [O'D86]).

Solved forms in this particular case are unification problems. The next step, discussed in section 5, is to show *how* to find such a unification problem.

[6] If t and t_i are not unifiable, then it is possible to remove t_i from the complement problem. So, it is possible to assume every t_i unifiable with t.

3.4 Other solved forms

In [Mah88a], solved forms are slightly different from our basic formulas. Let a *simple formula*[7] (in the case of finite trees) be an equational formula of the form:

$$\exists \vec{w}: \ x_1 = t_1 \wedge \ldots \wedge x_n = t_n$$

where $\vec{w} \subseteq Var(t_1, \ldots, t_n)$ and $x_1, \ldots, x_n$ are distinct variables occurring only once in the problem. Then a solved form in Maher's formalism is any boolean conjunction of simple formulas.

We didn't choose them as solved forms since the solvability requirement is less obvious for boolean combinations of simple formulas than for basic formulas.

Mal'cev in [Mal71] also makes use of some (other) kind of solved forms called *standard formulas*. However the definition of standard formulas is much more complicated than the definition of basic or simple formulas. We don't give it here. In our opinion, these solved forms are too complex and don't fulfill the simplicity requirement for solved forms.

HJ. Bürckert [Bür88, BB89] uses some other kind of solved forms called *substitutions with exceptions*. Such solved forms are formulas

$$\exists \vec{w} \forall \vec{y} : x_1 = t_1 \wedge \ldots \wedge x_n = t_n \wedge w_1 \neq u_1 \wedge \ldots \wedge w_m \neq u_m$$

where

- $x_1, \ldots, x_n$ are free variables
- $Var(t_1, \ldots, t_n) \cap \vec{y} = \emptyset$
- $w_1, \ldots, w_m \in \vec{w}$
- $Var(u_1, \ldots, u_n) = \vec{y}$.

Actually, such solved forms are nothing but conjunctions of complement problems. Since Bürckert starts from quantifier free formulas, he roughly applied the inverse transformation of complement problems into basic formulas. Nevertheless, there is an important difference : substitutions with exceptions are in a simplified form, so that they always have a solution when F is infinite: they fulfill the solvability condition.

Substitutions with exceptions and simple formulas can be easily related: a substitution with exceptions is $\phi \wedge \neg\psi_1 \wedge \ldots \wedge \neg\psi_n$ where $\phi, \psi_1, \ldots, \psi_n$ are simple formulas. On the other hand, boolean combination of simple formulas can be written as a finite set of substitutions with exceptions.

3.5 Equational formulas in equational theories

Lemma 3.5 actually shows that the solvability property holds (in the case of finite trees) for a set of formulas more general than basic ones. *Compact formulas* are defined like basic ones except that the z_i in definition 3.4 are allowed to be non variable terms. Then Corollary 3.6 could be re-stated with compact formulas in place of basic formulas.

Definition 3 *Let $\mathcal{A} = T(F)/_{=_E}$. A basic (respectively compact) formula is either $\top$, $\perp$ or has the form:*

$$\exists \vec{w} : x_1 = t_1 \wedge \ldots \wedge x_n = t_n \wedge z_1 \neq u_1 \wedge \ldots \wedge z_m \neq u_m$$

where

- $\vec{w} \subseteq Var(t_1, \ldots, t_n, z_1, \ldots, z_m, u_1, \ldots, u_m)$
- $x_1, \ldots, x_n$ *are variables occurring only once in the formula*
- *for every index i, $z_i \neq_E u_i$*
- *for every i, z_i is a variable (respectively z_i is a term)*

Of course, this definition corresponds to definition 3.4 when $E = \emptyset$.

Compact formulas are not interesting in the case of free algebras. However, they are useful when $\mathcal{A}$ is a quotient algebra $T(F)/_{=_E}$. Let us call a set of equations E *compact* if

- There is an algorithm which transforms each equation into an equivalent finite set of simple formulas

[7] Such formulas are called "basic formulas" in Maher's paper. We choose to give them another name in order to avoid confusion with definition 3.4.

	simple formulas	$\exists \vec{w} :\ x_1 = t_1 \wedge \ldots \wedge x_n = t_n$ where $\vec{w} \subseteq Var(t_1, \ldots, t_n)$ and the variables x_i occur only once.
$\mathcal{SF}_u$	unification formulas	Finite disjunctions of simple formulas
$\mathcal{SF}_b$	disjunctions of basic formulas in the case of finite trees	$\bigvee_{j \in J} (\exists \vec{w} :\ x_1 = t_1 \wedge \ldots x_n = t_n \wedge z_1 \neq u_1 \wedge \ldots \wedge z_m \neq u_m)$ where $x_1, \ldots, x_n$ are free variables occurring only once, for all i, z_i is a variable and $z_i \not\equiv u_i$
$\mathcal{SF}_{br}$	disjunctions of basic formulas in the case of rational trees	$\bigvee_{j \in J} (\exists \vec{w} :\ x_1 = t_1 \wedge \ldots x_n = t_n \wedge z_1 \neq u_1 \wedge \ldots \wedge z_m \neq u_m)$ where $x_1, \ldots, x_n$ are free variables not occurring in $z_1, \ldots, z_m, u_1, \ldots, u_m$, for all i, z_i is a variable and $z_i \not\equiv u_i$ and there is no variable cycle
$\mathcal{SF}_{be}$	disjunctions of basic formulas in the case of equational theories	$\bigvee_{j \in J} (\exists \vec{w} :\ x_1 = t_1 \wedge \ldots x_n = t_n \wedge z_1 \neq u_1 \wedge \ldots \wedge z_m \neq u_m)$ where $x_1, \ldots, x_n$ are free variables occurring only once, for all i, z_i is a variable and $z_i \neq_E u_i$
$\mathcal{SF}_c$	compact formulas	$\bigvee_{j \in J} (\exists \vec{w} :\ x_1 = t_1 \wedge \ldots x_n = t_n \wedge z_1 \neq u_1 \wedge \ldots \wedge z_m \neq u_m)$ where $x_1, \ldots, x_n$ are free variables occurring only once, for all i, $z_i \neq_E u_i$
$\mathcal{SF}_s$	boolean combination of simple formulas	
$\mathcal{SF}_{se}$	substitutions with exceptions	$\bigvee_{j \in J} \exists \vec{w_j}, \forall \vec{y_j} :\ x_1 = t_1 \wedge \ldots \wedge x_n = t_n \wedge z_1 \neq u_1 \wedge \ldots \wedge z_m \neq u_m)$ where (for each j) 1) $\exists \vec{w_j} : x_1 = t_1 \wedge \ldots \wedge x_n = t_n$ is a simple formula, 2) $z_1, \ldots, z_m$ are variables occurring in the t_i's, 3) $\vec{y_j} = Var(u_1, \ldots, u_m)$ is disjoint from $Var(x_1, \ldots, x_n, t_1, \ldots, t_n)$

Figure 4: Definitions of solved forms

- Every equation $s = t$ such that $Var(s,t) = \{x\}$ and such that $s \neq_E t$ has only finitely many solutions in $T(F)/_{=_E}$

Examples of compact sets of equations are AC axioms and quasi-free axioms. (The latter will be defined and studied later on).

Compact sets of equations are interesting since compact formulas fulfill the solvability condition in this case.

Proposition 1 [Com88b, Com91a] *Let E be a compact set of equations and A be an infinite subset of $T(F)/_{=_E}$. Then every compact formula distinct from $\perp$ has at least one solution in A.*

3.6 Summary

All definitions of solved forms together with their denotations are summarized in Figure 4.

4 Axioms and Rules

In the first sections, we explained what a disunification algorithm is supposed to do. In this section and the following, we are interested in how to do it. As claimed in the introduction, disunification algorithms will consist in a clever use of the axioms defining the algebras under consideration. Therefore, we first give the axioms for various kinds of term algebras and then show a control in the case of finite trees.

4.1 Definitions

In the following $\mathcal{K}$ is a class of F-algebra, possibly a single algebra $\mathcal{A}$.

Let $\mathcal{T}$ be a set of equational formulas (axioms). $\mathcal{T}$ is said to be *correct* with respect to $\mathcal{K}$ if $\mathcal{K} \models \phi$ for every formula $\phi \in \mathcal{T}$.

The following definitions are derived from e.g. [Sho67].

A *sentence* is an equational formula without free variable. The *first order theory* of a set of sentences $\mathcal{T}$ is the set $Th(\mathcal{T})$ of all sentences ϕ such that $\mathcal{T} \models \phi$. Sometimes we will simply write "theory" instead of "first order theory". (This should not be confused with "equational theory" that only contains equations).

$\mathcal{T}$ is *complete* if, for every sentence ϕ, either $\mathcal{T} \models \phi$ or $\mathcal{T} \models \neg\phi$. A complete and recursively enumerable set of formulas has a decidable first order theory, i.e. $Th(\mathcal{T})$ is recursive.

The *first order theory* $Th(\mathcal{K})$ of a class of algebras $\mathcal{K}$ is the set of sentences ϕ such that $\mathcal{K} \models \phi$. An *axiomatization* of $\mathcal{K}$ is a set of axioms $\mathcal{T}$ such that $Th(\mathcal{T}) = Th(\mathcal{K})$. In particular, an axiomatization of $\mathcal{K}$ is correct with respect to $\mathcal{K}$.

Note that, when $\mathcal{K}$ is a single algebra $\mathcal{A}$, then any axiomatization $\mathcal{T}$ of $\mathcal{A}$ is complete.

Two F-algebras $\mathcal{A}$ and $\mathcal{A}'$ are *elementarily equivalent* if, for every sentence ϕ,

$$\mathcal{A} \models \phi \text{ iff } \mathcal{A}' \models \phi$$

When $\mathcal{T}$ is complete, then any two models of $\mathcal{T}$ are elementarily equivalent.

A *rule* R (or a set of rules $\mathcal{R}$) is the denotation of a rewrite system in the algebra $\mathcal{EF}$ (In general, there is no ambiguity on the rewrite system denoted by a rule). Given two equational formulas ϕ and ϕ', we write $\phi \mapsto_R \phi'$ (or simply $\phi \mapsto \phi'$ when there is no ambiguity) if there is some rewrite rule (denoted by R) such that ϕ reduces to ϕ' using this rewrite rule. A rule R is *correct* with respect to $\mathcal{K}$ if

$$\phi_1 \mapsto_R \phi_2 \quad \text{implies} \quad \phi_1 \approx_{\mathcal{K}} \phi_2$$

If $\mathcal{R}$ is a set of rules then $\mapsto_{\mathcal{R}}$ is the union of $\mapsto_R$ for $R \in \mathcal{R}$.

Let $\mathcal{T}$ be a set of correct axioms for some class of algebras $\mathcal{K}$. If it is possible to deduce $\phi \Leftrightarrow \phi'$ from $\mathcal{T}$ for every ϕ and ϕ' such that $\phi \mapsto_{\mathcal{R}} \phi'$, then every rule in $\mathcal{R}$ is correct with respect to $\mathcal{K}$.

We will follow the following method for deriving disunification algorithms : first, for each class of models $\mathcal{K}$ under consideration, design a set of axioms $\mathcal{T}$ that hold in $\mathcal{K}$. Then design a set of rules that can be proved to be correct using $\mathcal{T}$. Then (this is the subject of the next section) prove the termination of the rules. If every irreducible formula with respect to the set of rules is in solved form, then (by the solvability property), it is possible to decide whether a given sentence holds in the model(s). This proves in turn that $Th(\mathcal{T})$ is a recursive axiomatization of $\mathcal{K}$.

4.2 Finite trees axioms

When $\mathcal{A} = T(F)$ (or $T(F, X)$) the axioms were given in Figures 1 et 2: let E_{FT} be this set of axioms. It turns out that E_{FT} completely axiomatizes the algebra of finite trees over an infinite set of function symbols:

Theorem 5 [Kun87b] *E_{FT} is complete when F is infinite.*

This result is also proved in a different way in [Mah88a]. E_{FT} was also shown to be decidable in [Mal71][8].

A *locally free* algebra is an F-algebra such that every finitely generated subalgebra is free. Actually, the models of E_{FT} are the locally free algebras ([Mal71, Mah88a]). Therefore, theorem 4.1 shows that the class of locally free algebras is completely axiomatizable when F is infinite. Moreover, since $T(F)$ is a model of E_{FT}, we obtain the following Corollary:

Corollary 2 *E_{FT} is a complete axiomatization of $T(F)$ when F is infinite.*

[8] Mal'cev didn't make any assumption on F. On the other hand, it turns out that the theory is not complete when F is finite (E_{FT} is smaller in this case since there are only finitely many clash axioms). Therefore, it is not possible to derive theorem 4.1 from Malcev's results.

Rational Trees axioms

Equality axioms (figure 1)

Decomposition

$$\forall t_1, \ldots, t_n, u_1, \ldots, u_n,\ f(t_1, \ldots, t_n) = f(u_1, \ldots, u_n) \Rightarrow t_1 = u_1 \wedge \ldots \wedge t_n = u_n$$

Clash

$$\forall t_1, \ldots, t_n, u_1, \ldots, u_m,\ g(t_1, \ldots, t_m) \neq f(u_1, \ldots, u_n) \qquad \text{If } f \neq g$$

Unique solution

$$\exists x \forall z :\ x = t[x] \wedge (z = t[z] \Rightarrow z = x)$$

For every non-trivial context t.

Figure 5: Rational Trees axioms

When the alphabet F is finite, the problem is more difficult: E_{FT} is not complete since a formula such as $\forall y\ :\ x \neq 0 \wedge x \neq s(y)$ has no solution in $T(\{0, s\})$ but cannot be proved equivalent to $\perp$ using only E_{FT}. We actually need the *domain closure axiom*:

$$(DCA) \quad \forall x \bigvee_{f \in F} \exists \vec{w_f} : x = f(\vec{w_f})$$

where $\vec{w_f}$ is a sequence $w_1, \ldots, w_n$ of variables with $n = \text{arity}(f)$.

The set of axioms is now complete:

Theorem 6 [**Mah88a**] *$E_{FT} \cup \{DCA\}$ is a complete axiomatization of the algebra $T(F)$ over a finite alphabet F, provided that $T(F)$ contains at least two distinct elements.*

Again, theorems 4.1 and 4.3 can be proved using the results given in section 5. For, assume that $\mathcal{R}$ is a terminating set of rules such that $\phi \Leftrightarrow \phi'$ can be deduced from $E_{FT} \cup \{DCA\}$ whenever $\phi \mapsto_{\mathcal{R}} \phi'$. If any formula can be transformed into a basic formula using $\mathcal{R}$, then Theorem 4.3 follows from Corollary 3.6 (when $T(F)$ is infinite).

E_{FT} is not a finite set of axioms since Occur-check actually is a scheme of axioms which has infinitely many instances. (However, it is, of course, recursive). In fact, Mal'cev shows in [Mal71] that there is no finite complete axiomatization of the class of locally-free algebras and Maher [Mah88a] shows that there is no finite complete axiomatization of $T(F)$ (in either case F finite or not, provided that $T(F)$ is infinite).

4.3 Rational Trees axioms

Let E_{RT} be the axioms given in Figures 5 and 1.

Theorem 7 [**Mah88a**] *E_{RT} is a complete axiomatization of $RT(F)$ when F is infinite. $E_{RT} \cup \{DCA\}$ is a complete axiomatization of $RT(F)$ when F is finite and $RT(F) \neq T(F)$*

Since $IT(F)$ is also a model of E_{RT} we have the following corollary:

Corollary 3 *$IT(F)$ and $RT(F)$ are elementarily equivalent .*

Again, Unique Solution is a scheme of axioms that has infinitely many instances. But the set of instances is recursive.

4.4 Syntactic theories axioms

C. Kirchner introduced *syntactic theories* [Kir86] which are a class of equational theories for which it is possible to systematically derive unification algorithms (see also [DJ90]). In such theories it is possible to give axioms such as those of Figure 6 which can be used for "local" transformations of formulas. More precisely, it is possible to derive rules similar to decompositions and clashes. That is the reason why such theories are well suited for a study by transformation of formulas.

Mutations

$$\begin{array}{ll}\forall x_1,\dots,x_n,y_1,\dots,y_m & [f(x_1,\dots,x_n)=f(y_1,\dots,y_n)\iff \\ & (t_1=u_1\wedge\ldots\wedge t_n=u_n) \\ & \vee\exists\vec{x}: \bigvee_{f(v_1,\dots,v_n)=f(w_1,\dots,w_n)\in E}(\bigwedge_{i=1}^{n}(t_i=v_i\wedge u_i=w_i)) \\ & \vee\exists\vec{x}: \bigvee_{\substack{f(v_1,\dots,v_n)=x\in E\\ x\text{ variable}}}(x=f(t_1,\dots,t_n)\bigwedge_{i=1}^{n}u_i=v_i) \\ & \vee\exists\vec{x}: \bigvee_{\substack{f(v_1,\dots,v_n)=x\in E\\ x\text{ variable}}}(x=f(u_1,\dots,u_n)\bigwedge_{i=1}^{n}t_i=v_i)]\end{array}$$

For every $f\in F$. $\vec{x}$ is assume to contain all variables distinct from $x_1,\dots,x_n,y_1,\dots,y_n$.

$$\begin{array}{ll}\forall x_1,\dots,x_n,y_1,\dots,y_m & [f(x_1,\dots,x_n)=g(y_1,\dots,y_m)\iff \\ & \exists\vec{x}: \bigvee_{f(v_1,\dots,v_n)=g(w_1,\dots,w_m)\in E}(\bigwedge_{i=1}^{n}(t_i=v_i)\wedge\bigwedge_{i=1}^{m}u_i=w_i)) \\ & \exists\vec{x}: \bigvee_{\substack{g(v_1,\dots,v_m)=x\in E\\ x\text{ variable}}}(x=f(t_1,\dots,t_n)\bigwedge_{i=1}^{m}u_i=v_i) \\ & \exists\vec{x}: \bigvee_{\substack{f(v_1,\dots,v_n)=x\in E\\ x\text{ variable}}}(x=g(u_1,\dots,u_m)\bigwedge_{i=1}^{n}t_i=v_i)\end{array}$$

For all distinct function symbols $f,g\in F$. $\vec{x}$ is the set of variables which are not in $\{x_1,\dots,x_n,y_1,\dots,y_m\}$.

Figure 6: Mutation Axioms

Occur checks

$$\forall x\quad [f(t_1,\dots t_i[x],\dots,t_n)=x\iff\ \exists\vec{z}\bigvee_{f(v_1,\dots,v_n)\to w\in E_{c,i}}\bigwedge_{j=1}^{n}t_j=v_j\wedge x=w]$$

For each function symbol $f\in F$, each $i\le a(f)$, each terms $t_1,\dots,t_n$ and if $E_{c,i}$ is the subset of E containing all i-collapsing rules.

Figure 7: Axioms of quasi-strict theories

$$(P)\ \forall x_1,\ldots,x_n,y_1,\ldots,y_n,\ f(x_1,\ldots,x_n)=f(y_1,\ldots,y_n) \Leftrightarrow \bigvee_{\sigma\in\Pi_n}(x_1=y_{\sigma(1)}\wedge\ldots\wedge x_n=y_{\sigma(n)})$$

where Π_n is a subset of the n^{th} permutative group.

Figure 8: Permutative axioms in Mal'cev work

A finite set of equational axioms[9] E is *resolvent* if every proof of equality $s =_E t$ can be performed using at most once an equation of E at the root position. The axioms given in Figure 6 are correct in the class $\mathcal{K}_E$ of all models of E when E is resolvent. A *syntactic theory* is an equational theory (i.e. a set of equations) that can be generated by the rules of Figure 1 and a resolvent set of axioms. For example, a set of equations of the form

$$f(x_1,\ldots,x_n)=f(x_{\pi(1)},\ldots,x_{\pi(n)})$$

where π is a permutation of $\{1,\ldots,n\}$ is resolvent. Such a set of axioms will be called *permutative*. A typical example of permutative axioms are those of commutativity.

An associative axiom alone is resolvent. However associativity combined with commutativity leads to a non-resolvent set of axioms.

Resolvent presentations are useful because it is possible to derive the axioms of figure 6 which replace the decomposition and clash axioms which are no longer valid in the case of a quotient by a congruence. Similarly, we need hypotheses and rules for solving cyclic equations $s = t[s]$, thus replacing the occur-check axiom which is not valid in the quotient algebra $T(F)/_{=_E}$.

A rule $f(s_1,\ldots,s_n)\to t$ is called *i-collapsing* if, for every substitution σ, $|s_i\sigma|\geq|t\sigma|$. ($|\cdot|$ is the *size* of a term, i.e. the cardinal of its positions). Then, a finite set of equations is *quasi-strict* when, for every cyclic identity $t[u]_p =_E u$, there is a decomposition $p = q\cdot i\cdot r$ and an i-collapsing rule $l\to r\in E$ such that the proof of the cyclic equation can be performed by first using proof steps below the position q, then using $l\to r$ at position q (from left to right) then going on with any proof. (We refer to [Com91b] for more precise definitions).

Let E_{ST} be the set of axioms of Figures 1, 6 and 7. The results for finite trees can be extended to (some) syntactic theories: we say that E is *quasi-free* if it is resolvent, quasi-strict and if every equation in E has a depth 0 or 1. For example, commutativity (or permutativity) axioms fulfill this condition but not associativity. Also, equational theories which are defined by ground equations do have a conservative extension which is generated by a quasi-free set of axioms [Com91a]. There are however "few" equational theories satisfying these requirements.

Theorem 8 *E_{ST} is a complete axiomatization of $T(F)/_{=_E}$ when F is infinite and E is quasi-free. $E_{ST}\cup\{DCA\}$ is a complete axiomatization of $T(F)/_{=_E}$ if F is finite, $T(F)/_{=_E}$ is infinite and E is quasi-free.*

These results can be found in [Com91a, Com91b]. They are again consequences of results given in the next section.

Mal'cev [Mal71] shows a similar result in a particular case: let E_P be the axioms of Figure 8 and E^0_{FT} be the axioms in E_{FT} without the decomposition axioms. Then Mal'cev shows that $E_P\cup E^0_{FT}$ is decidable. (He calls "permutative locally free algebras" the models of this theory).

4.5 Rules for finite trees

In order to show how the rules are built and can actually be usued, we choose to give them in the case of finite trees over a finite alphabet. We also assume $T(F)$ to be infinite. In other models, the transformation rules are only slightly different.

In Figures 10, 12 and 13, each rule is given together with a (possibly empty) set of conditions, which are not always necessary for correctness. They are however all necessary for termination. This means that it is not possible to drop any condition at all without loosing the termination property. On the other hand, any additional condition will not impair termination. Such additional conditions are necessary in practice since the rules are non-deterministic.

The rules in Figures 10, 12 and 13 apply on elementary formulas. They result in elementary formulas.

The rules in Figure 11 are always valid. For simplicity we assume all formulas in normal form with respect to $\to_{\mathcal{E}}$. In other word, we assume $\to_{\mathcal{E}}$-normalization is applied before any other rule.

In order to have a compact representation of the rules, we use the lexicographic conventions given in Figure 9. A variable x is *solved* in a formula $\phi\equiv\exists\vec{w}\forall\vec{y}:\ P$ (where P is a qfef) if $x\notin\vec{y}$ and $P\equiv x=t\wedge Q$ with $x\notin Var(t,Q)$.

[9] As before, $=$ is assumed to be commutative. Therefore, we make no difference between $s=t$ and $t=s$ for axioms in E.

$x, x', x_1, x_i, \ldots$	free variables
$y, y', y_1, y_i, \ldots$	universally quantified variables
$w, w', w_1, w_i, \ldots$	existentially quantified variables
$z, z', z_1, z_i, \ldots$	free or existentially quantified variable
$v, v', v_1, v_i, \ldots$	any variable
$t, t', t_1, t_i, \ldots$	terms without any occurrence of a universally quantified variable
$u, u', u_1, u_i, s, s', s_1, s_i, \ldots$	terms
$t[z], u[t], \phi[s = t], \ldots$	the subterm (or subformula) inside the brackets has at least one occurrence in the term outside the brackets
$d, d', d_1, d_i, \ldots$	disjunctions of equations and disequations
P	a qfef

Figure 9: Lexicographic conventions

For any term t, $size_u(t) = \sum_{p \in P_u(t)} |p|$ where $|p|$ is the size of the position p (i.e. the length of the word) and $P_u(t)$ is the set of positions p in t such that t/p is a universally quantified variable.

Let E_{eq} be the set of axioms in Figure 1. We start with correctness:

Proposition 2 *For every rule R given in Figure 10 $E_{eq} \models R$.*

Proposition 3 *For every rule R given in Figure 12, $E_{FT} \models R$.*

Proposition 4 *Assume that F is finite and that $T(F)$ is infinite. For every rule R given in Figure 13, $E_{FT} \cup \{DCA\} \models R$.*

Actually, all rules in Figure 13 but Explosion are consequences of E_{FT}. However, in the case F infinite, they can be (and must be) generalized. This is the reason why we didn't include them in Figure 12.

Let us show an example of transformation, using the rules in Figures 10, 11, 12, 13:

Example 4
$F = \{0, f, g\}$ where 0 is a constant, f is a binary function symbol and g is unary. Let $\mathcal{P}$ be the following elementary formula:

$$\mathcal{P} \equiv \ \exists w_1, \forall y_1, y_2 : \ f(f(x_1, y_1), f(y_2, y_2)) \neq f(f(g(y_1), x_2), x_2) \wedge f(w_1, x_1) \neq f(x_1, y_1)$$

We show a sequence of transformations of $\mathcal{P}$: first apply the decomposition rules

$$\mathcal{P} \mapsto^*_{D_2} \ \exists w_1, \forall y_1, y_2 : \ (x_1 \neq g(y_1) \vee y_1 \neq x_2 \vee x_2 \neq f(y_2, y_2)) \wedge (w_1 \neq x_1 \vee x_1 \neq y_1)$$

Now apply elimination of universally quantified variables:

$$\begin{array}{ll} \ldots & \mapsto^*_{UE_3} \ \exists w_1, \forall y_1, y_2 : (x_1 \neq g(x_2) \vee x_2 \neq f(y_2, y_2)) \wedge w_1 \neq x_1 \\ & \mapsto_{UE_1} \ \exists w_1, \forall y_2 : (x_1 \neq g(x_2) \vee x_2 \neq f(y_2, y_2)) \wedge w_1 \neq x_1 \end{array}$$

Now use the explosion rule, which leads to a disjunction of three problems:

$$\begin{array}{lll} \ldots \ \mapsto_E & & (\exists w_1, \forall y_2 : (x_1 \neq g(x_2) \vee x_2 \neq f(y_2, y_2)) \wedge w_1 \neq x_1 \wedge x_2 = 0) \\ & \vee & (\exists w_1, w_2, \forall y_2 : (x_1 \neq g(x_2) \vee x_2 \neq f(y_2, y_2)) \wedge w_1 \neq x_1 \wedge x_2 = g(w_2)) \\ & \vee & (\exists w_1, w_2, w_3, \forall y_2 : (x_1 \neq g(x_2) \vee x_2 \neq f(y_2, y_2)) \wedge w_1 \neq x_1 \wedge x_2 = f(w_2, w_3)) \end{array}$$

Applying Merging, Decompositions and Clashes yields:

$$\begin{array}{ll} \ldots \ \mapsto^*_{M_3, M_5, C_2, D_2} & (\exists w_1, \forall y_2 : \ w_1 \neq x_1 \wedge x_2 = 0) \\ & \vee (\exists w_1, w_2, \forall y_2 : \ w_1 \neq x_1 \wedge x_2 = g(w_2)) \\ & \vee (\exists w_1, w_2, w_3, \forall y_2 : \ (x_1 \neq g(x_2) \vee y_2 \neq w_2 \vee y_2 \neq w_3) \wedge w_1 \neq x_1 \wedge x_2 = f(w_2, w_3)) \end{array}$$

Replacement

$$
\begin{array}{llll}
(R_1) & z = t \wedge P[z] & \mapsto & z = t \wedge P[z]\{z \to t\} \\
(R_2) & z \neq t \vee P[z] & \mapsto & z \neq t \vee P[z]\{z \to t\}
\end{array}
$$

If $z \notin Var(t)$ and, if t is a variable, then it occurs in P. Moreover, (R_2) is not applied when (R_1) can be applied.

Universal quantifiers elimination

$$
\begin{array}{lllll}
(UE_1) & \forall \vec{y}, y : P & \mapsto & \forall \vec{y} : P & \text{if } y \notin Var(P) \\
(UE_2) & P \wedge y \neq u & \mapsto & \perp & \\
(UE_3) & y \neq u \vee d & \mapsto & d\{y \to u\} &
\end{array}
$$

If $y \notin Var(u)$.

Merging

$$
\begin{array}{llll}
(M_1) & z = t \wedge z = u & \mapsto & z = t \wedge t = u \\
(M_3) & z = t \wedge z \neq u & \mapsto & z = t \wedge t \neq u \\
(M_5) & z = t \wedge (z \neq u \vee R) & \mapsto & z = t \wedge (t \neq u \vee R) \\
(M_6) & z = t \wedge (z = u \vee R) & \mapsto & z = t \wedge (t = u \vee R)
\end{array}
$$

If t is not a variable and one of the following holds:

- $size(t) \leq size(u)$
- u contains a non-trivial occurrence of a universally quantified variable
- u is a solved variable.

$$
\begin{array}{llll}
(M_2) & v \neq s \vee v \neq u & \mapsto & v \neq s \vee s \neq u \\
(M_4) & v = s \vee v \neq u & \mapsto & s = u \vee v \neq u
\end{array}
$$

If s is not a variable and one of the following holds

- $size_u(s) \leq size_u(u) \neq 0$
- $size(s) \leq size(u)$ and s does not contain any occurrence of a universally quantified variable.

Existential quantifier elimination

$$
\begin{array}{llll}
(EE_1) & \exists w : P & \underset{\text{If } w \notin Var(P)}{\mapsto} & P \\
(EE_2) & \exists \vec{w}, w : w = t \wedge P & \underset{\text{If } w \notin Var(P, t)}{\mapsto} & \exists \vec{w} : P
\end{array}
$$

Figure 10: Equality rules

$\rightarrow_{\mathcal{E}}$-Normalization

$$\phi \mapsto \phi\downarrow_{\mathcal{E}}$$

Lifting Disjunctions

$$(DL) \quad \exists\vec{w}\,\forall\vec{y} : P \wedge (P_1 \vee P_2) \quad \mapsto \quad (\exists\vec{w}\,\forall\vec{y} : P \wedge P_1) \vee (\exists\vec{w}\,\forall\vec{y} : P \wedge P_2)$$

If $Var(P_1) \cap \vec{y} = \emptyset$ or $Var(P_2) \cap \vec{y} = \emptyset$.

Figure 11: Logical connectives rules

Conflict

$$\begin{array}{llll} (C_1) & f(t_1,\ldots,t_m) = g(u_1,\ldots,u_n) & \mapsto & \perp \\ (C_2) & f(t_1,\ldots,t_m) \neq g(u_1,\ldots,u_n) & \mapsto & \top \end{array}$$

If $f \neq g$

Decomposition

$$\begin{array}{llll} (D_1) & f(t_1,\ldots,t_m) = f(u_1,\ldots,u_m) & \mapsto & t_1 = u_1 \wedge \ldots \wedge t_m = u_m \\ (D_2) & f(t_1,\ldots,t_m) \neq f(u_1,\ldots,u_m) & \mapsto & t_1 \neq u_1 \vee \ldots \vee t_m \neq u_m \end{array}$$

$$(D_3) \quad f(t_1,\ldots,t_n) = f(u_1,\ldots,u_n) \vee d \quad \mapsto \quad (t_1 = u_1 \vee d) \wedge \ldots \wedge (t_n = u_n \vee d)$$

D_1 and D_3 are assumed to be applied only at a conjunction position. Moreover, (D_3) is applied only if one of the terms $t_1,\ldots,t_n,u_1,\ldots,u_n$ contains a universally quantified variable, or else d does not contain any universally quantified variable.

Occur Check

$$\begin{array}{llll} (O_1) & s = u[s] & \mapsto & \perp \\ (O_2) & s \neq u[s] & \mapsto & \top \end{array}$$

Figure 12: Rules for finite trees

Universal quantifiers elimination

$$(UE_5) \quad P \wedge (v_1 = u_1 \vee \ldots \vee v_n = u_n \vee d) \quad \mapsto \quad P \wedge d$$

If
1. for each i, $v_i = u_i$ contains at least one occurrence of a universally quantified variable,
2. d does not contain any universally quantified variable.

Existential quantifiers elimination

$$(EE_3) \quad \exists \vec{w} : (d_1 \vee z_1 \neq t_1) \wedge \ldots \wedge (d_n \vee z_n \neq t_n) \wedge P \quad \mapsto \quad \exists \vec{w} : P$$

If there exists a variable $w \in \vec{w} \cap Var(z_1, t_1) \cap \ldots \cap Var(z_n, t_n)$ which does not occur in P.

Explosion

$$(E) \quad \exists \vec{w_1} \, \forall \vec{y} : P \quad \mapsto \quad \bigvee_{f \in F} \exists \vec{w} \, \exists \vec{w_1} \, \forall \vec{y} : P \wedge z = f(\vec{w})$$

If $\vec{w} \cap Var(P) = \emptyset$, no other rule can be applied and there exists in P an equation or a disequation $z = u$ where u contains an occurrence of a universally quantified variable

Figure 13: Rules for finite trees over a finite set of function symbols

Eliminating the universally quantified variables and elimination of existentially quantified variables:

$$\begin{array}{lll} \ldots \quad \mapsto^*_{UE_3, UE_1, EE_1, EE_3} & & (x_2 = 0) \\ & \vee & (\exists w_2 : x_2 = g(w_2)) \\ & \vee & (\exists w_2, w_3 : (x_1 \neq g(x_2) \vee w_2 \neq w_3) \wedge x_2 = f(w_2, w_3)) \end{array}$$

After disjunctions lifting, the formula is actually a solved form. It is however still possible to apply a replacement rule:

$$\begin{array}{lll} \ldots \quad \mapsto^*_{DL, R_1} & & (x_2 = 0) \vee (\exists w_2 : x_2 = g(w_2)) \\ & \vee & (\exists w_2, w_3 : x_2 = f(w_2, w_3) \wedge x_1 \neq g(f(w_2, w_3))) \\ & \vee & (\exists w_2, w_3 : x_2 = f(w_2, w_3) \wedge w_2 \neq w_3) \end{array}$$

$\mathcal{EF}$	Every equational formula
qfef	Quantifier free equational formulas
elementary formulas $\mathcal{C}_e$	$\bigvee_{j\in J}(\exists \vec{w}_j, \forall \vec{y}_j : \phi)$ where ϕ is a qfef and $\vec{w}_j$, $\vec{y}_j$ are possibly empty sets of variables.
$\mathcal{C}_\exists$	$\exists \vec{w} : \phi$ where ϕ is a qfef
$\mathcal{C}_s$	$\exists \vec{w}, \forall \vec{y} : e \wedge d_1 \wedge \ldots \wedge d_n$ where e is a conjunction of equations and each d_i is a disjunction of disequations
$\mathcal{C}_c$	complement problems
$\mathcal{C}_u$	formulas that are equivalent (modulo $\approx_{T(F)}$) to unification formulas.
$\mathcal{C}_{su}$	$\mathcal{C}_s \cap \mathcal{C}_u$

Figure 14: Sets of equational formulas

5 Termination and Completeness

5.1 Definitions

Let $\mathcal{C}$ be a set of (equational) formulas (for example, complement problems, qfef, elementary formulas,...Their definitions and denotations are summarized in Figure 14) and $\mathcal{SF}$ be a set of solved forms (for example basic formulas, unification problems ,...). Let $\mathcal{K}$ be a class of F-algebras and $\mathcal{R}$ a set of (schemes of) rules which are correct in $\mathcal{K}$. We say that $\mathcal{R}$ is *complete* with respect to $\mathcal{SF}$ (and $\mathcal{K}$, $\mathcal{C}$ which are sometimes omitted) if for every formula ϕ in $\mathcal{C}$ there exists a formula ψ in $\mathcal{SF}$ such that $\phi \mapsto_{\mathcal{R}} \psi$. This implies in particular that every formula in $\mathcal{C}$ is equivalent (modulo $\approx_{\mathcal{K}}$) to a formula in $\mathcal{SF}$.

When the relation $\mapsto_{\mathcal{R}}$ terminates (i.e. there is no infinite chain $\mathcal{P}_1 \mapsto_{\mathcal{R}} \ldots \mapsto_{\mathcal{R}} \ldots$), then the completeness of $\mathcal{R}$ can be shown just by introspecting the irreducible problems: it is sufficient to show that any irreducible problem is in $\mathcal{SF}$.

5.2 Completeness and termination results for finite trees

Let $\mathcal{R}$ be the set of rules defined in Figures 10, 11, 12, 13. Then we have the expected result:

Theorem 9 [CL89] *$\mathcal{R}$ is correct, terminating and complete with respect to $\mathcal{C}_e$ and $\mathcal{SF}_b$ when $T(F)$ is infinite and F is finite.*

The proof of this result is carried out by exhibiting an interpretation of elementary formulas in a well-ordered set and showing that any application of a rule decreases the interpretation. This proves termination. Then, it suffices to show that the irreducible formulas in solved form: the completeness result follows.

Note that the termination result doesn't make any assumption on the way the rules in $\mathcal{R}$ are actually used. The termination property holds for every algorithm derived from $\mathcal{R}$ by strengthening the control.

Similar results can be obtained when F is infinite or when $T(F)$ is finite, but with different solved forms (see [Com88b, Mah88a]).

It is possible to generalize these results to arbitrary equational formulas, using a well known procedure for elimination of quantifiers. Let $\mathcal{R}'$ be the set of rules defined in Figure 15. In this Figure, the symbol $\oplus$ stands for disjoint union and the function α is the usual α-conversion. That is to say : some variables are renamed in order to get a disjoint union.

Simplification of elementary formulas

$$\phi \mapsto \phi\downarrow_{\mathcal{R}}$$

If ϕ is an elementary formula.

Handling negation

$$\begin{array}{rcl}
\neg\exists\vec{x} : \phi_0 & \mapsto & \forall\vec{x} : \neg\phi_0 \\
\forall\vec{y}\,\exists\vec{w} : \phi_0 & \mapsto & \neg\exists\vec{y}\,\forall\vec{w} : \neg\phi_0 \\
\neg\neg\phi & \mapsto & \phi \\
\neg\forall\vec{y}\,\exists\vec{w} : \phi_0 & \mapsto & \exists\vec{y}\,\forall\vec{w} : \neg\phi_0 \\
\neg\forall\vec{y} : \phi_0 & \mapsto & \exists\vec{y} : \neg\phi_0
\end{array}$$

where ϕ_0 is a qfef and $\vec{w}$ is not empty.

Quantifier Lifting rules

$$\begin{array}{rcl}
(\forall\vec{y} : \phi) \wedge (\forall\vec{y'} : \psi) & \mapsto & \forall\vec{y}\,\forall\vec{y'} : \phi \wedge \psi \\
(\exists\vec{w} : \phi) \vee (\exists\vec{w'} : \psi) & \mapsto & \exists\vec{w}\,\exists\vec{w'} : \phi \vee \psi \\
(\forall\vec{y} : \phi) \vee (\forall\vec{y'} : \psi) & \mapsto & \forall(\vec{y} \oplus \vec{y'}) : \phi \vee \alpha(\psi) \\
(\exists\vec{w} : \phi) \vee (\exists\vec{w'} : \psi) & \mapsto & \exists(\vec{w} \oplus \vec{w'}) : \phi \wedge \alpha(\psi)
\end{array}$$

Figure 15: Rules lifting the elementary formulas case to the equational formulas case

Theorem 10 *$\mathcal{R}'$ is correct, terminating and complete with respect to $\mathcal{EF}$ and $\mathcal{SF}_b$ when $\mathcal{A}$ is the set of finite trees over a finite set of function symbols.*

Similar results hold for the case where F infinite. More general sets of rules can also be provided, allowing to eliminate existential quantifiers as well as universal ones and dropping the requirement for the formula to be in prenex form [Com90b].

Other sets of rules (together with a different control) are given in [Mal71, Mah88a] together with their completeness and termination properties.

5.3 Completeness and termination results for rational and infinite trees

We do not give here the set of rules for rational (or infinite) trees, see [Com88b, Mah88a], but state termination and completeness result:

Theorem 11 *There is a set of rules that is correct, terminating and complete with respect to $\mathcal{EF}$ and $\mathcal{SF}_{br}$ when $\mathcal{A} = RT(F)$ and F finite.*

5.4 Completeness and termination results for quotients of finite trees

It is not possible to expect termination and completeness results when $\mathcal{A}$ is an arbitrary quotient of $T(F)$. This would indeed imply in particular the decidability of unification modulo any set of equational axioms which is known to be undecidable [Dav77]. It is not even possible to find a semi-decision procedure for the satisfiability of equational formulas in any quotient of $T(F)$ since the first order theory of $T(F)/_{=_E}$ is complete. Hence, such a semi-decision procedure would imply the decidability of theory (see section 4).

This is the reason why we need some additional hypotheses either on E or on the formulas we are interested in. We mention here three results. In all cases, it is assumed that unification modulo E is decidable. This is a reasonable assumption since unification simply consists in solving a single equation: this should be decidable before looking at more complex formulas.

The results detailed below are also summarized in Figure 16.

5.4.1 Quasi-free theories

Let E be a quasi-free set of axioms.

Algebras	$T(F)$		$RT(F)$		$IT(F)$	irreducible solutions
	F finite	F infinite	F finite	F infinite		
	[CL89, Mah88a]	[Kun87b, Mah88a]	[Mah88a, Com88b]	[Mah88a]	[Mah88a]	[Com89, Com88b]

Equational theory	free	quasi-free	compact	AC	with decidable unification
Equational Formulas:					
F finite	[CL89, Mah88a]	[Com88b]	undecidable	undecidable [Tre90]	undecidable
F infinite	[Mah88a, Kun87b]	[Com88b]	·	·	undecidable
Existential formulas:					
F finite	·	·	[Com88b]	[Com88b]	undecidable [Fer91]
F infinite	[Col84]	·	·	·	[Bür88]

Figure 16: Summary of the results for complete and terminating disunification

Theorem 12 [Com91a] *There is a set of rules that is correct, terminating and complete with respect to $\mathcal{EF}$ and $\mathcal{SF}_{be}$ when $\mathcal{A} = T(F)/_{=_E}$ and F is finite.*

5.4.2 Compact theories

E is a compact set of axioms.

Proposition 5 [Com88b, Com91a] *There is a set of rules that is correct terminating and complete with respect to $\mathcal{C}_\exists$ and $\mathcal{SF}_c$ when $\mathcal{A} = T(F)/_{=_E}$ and F finite.*

5.4.3 Equational theories with a decidable unification problem

HJ. Bürckert considers the existential fragment in the case of an infinite alphabet and for an arbitrary set of equations E provided that a (finitary) unification algorithm is known for the equational theory defined by E [Bür88, BB89]. Let $\mathcal{SF}_{se}$ be the set of "substitutions with exceptions" (see the section on solved forms).

Theorem 13 *There is a set of rules that is correct terminating and complete with respect to $\mathcal{C}_\exists$ and $\mathcal{SF}_{se}$ when $\mathcal{A} = T(F)/_{=_E}$ and F is infinite.*

But, surprisingly, it has been shown that this result fails when F is finite; in such a case $\mathcal{C}_\exists$ is not recursive [Fer91].

5.4.4 Summary of the equational case

We summarize the results on disunification in equational theories in Figure 16. Given an equational theory and a set $\mathcal{C}$ of equational formulas, we mention in the corresponding cell the decidability result: when a decision algorithm is known for the given equational theory and set of formulas, then we give pointers to references. We also mention undecidability results. The undecidability in the AC case is shown by coding Post's correspondance problem with a first order formula over the alphabet $F = \{+, f, a, b, \epsilon\}$ where $+$ is associative and commutative, f is binary, a, b are unary and ϵ is a constant [Tre90]. The undecidability of the existential fragment of $T(F)/_{=_E}$ for a theory E in which unification is decidable (and finitary) is shown by reducing the AC-ground reducibility problem which has been shown undecidable in [KNRZ91].

An empty cell in the array means that the result can be derived from the others in a straightforward way.

5.5 Completeness with respect to Unification problems

Let $\mathcal{SF}_u$ be the set of unification problems and $\mathcal{C}_u$ be the set of all equational formulas that are equivalent (modulo $\approx_{T(F)}$) to a problem in $\mathcal{SF}_u$. The problem is to design a set of rules which is correct terminating and complete with respect to $\mathcal{C}_u$ and $\mathcal{SF}_u$ (when F is finite). Such a problem has not yet been solved in the general case. It has been solved in [LM87] for the subset $\mathcal{C}_{ucp}$ of $\mathcal{C}_u$ consisting in complement problems which are equivalent to unification problems.

Let $\mathcal{C}_s$ be the set of elementary formulas whose conjunctive normal forms do not contain any equation inside the disjunctions; such formulas may be written

$$\exists \vec{w}, \forall \vec{y} :\ s_1 = t_1 \wedge \ldots \wedge s_n = t_n \wedge d_1 \wedge \ldots \wedge d_m$$

where each d_i is a disjunction of disequations.

Let $\mathcal{C}_{su} = \mathcal{C}_u \cap \mathcal{C}_s$. It is shown in [Com88b] that there is a correct and terminating set of rules which is complete w.r.t $\mathcal{C}_{su}$ and $\mathcal{SF}_u$.

The equivalence to unification problems in the case of rational trees is investigated in [MS90].

5.6 Other completeness results

We mention here other results related with disunification:

Nelson and Oppen in [NO80] show an algorithm for deciding the satisfiability of a formula in $\mathcal{C}_\exists$.

Venkataraman shows in [Ven87] that, introducing $>$, interpreted as the subterm ordering, leads to undecidability results : the validity in $T(F)$ of a generalized equational sentence is undecidable. By contrast, he shows that the existential fragment is decidable.

Interpreting $>$ as a recursive path ordering leads to similar results: the existential fragment is decidable [Com90c] and the first order theory of embedding[10] is undecidable [Tre90]. It is still open whether the first order theory of a recursive path ordering extending a total precedence on F is decidable (although we suspect undecidability).

The author shows in [Com90b] how to extend theorem 5.1 to the order-sorted case. More precisely, using the syntax defined in section 1.7.2 and the semantics given in section 2.3.6, there is a set of solved forms (which fulfill the general requirements for solved forms) and a terminating, correct and complete set of transformation rules [Com90b]. This shows the decidability, for every rational tree language, of some structure which describes this language.

Other results are given by e.g. [Lug89, Pug90] showing how some particular classes of formulas involving arbitrary predicate symbols can be turned into "solved forms" using techniques similar to those presented here.

[10] See e.g. [Der87] for a definition. The embedding ordering can be viewed as a recursive path ordering extending an empty precedence on F.

6 Irreducible solutions

As shown in example 1 it is sometimes useful to decide whether a given equational formula is satisfiable in the set NF of ground terms which are irreducible with respect to a given term rewriting system.

These problems (finding irreducible ground solutions) should not be confused with solving the formula in the algebra $\mathcal{A} = NF$ (or $\mathcal{A} = T(F)/_{=_E}$). For example, let

$$\mathcal{P} \equiv \forall y_1, y_2 : s(x) \neq y_1 + y_2$$

Every assignment of x to an irreducible ground term is a solution in NF whereas $\mathcal{P}$ has no $T(F)/_{=_E}$-solution. (with the signature and equations of the above example).

In order to find irreducible solutions, let us first remark that solutions in NF are solutions in $T(F)$. Therefore, a first step consists in approximating the set of solutions in NF by $T(F)$-solutions. A second step keeps the solutions which are indeed irreducible.

More precisely, the formula ϕ is first reduced to a $\approx_{T(F)}$-equivalent basic formula ψ. Since $NF \subseteq T(F)$, ϕ and ψ do have the same set of solutions in NF. Then, it is possible to compute a (conditional) tree grammar for the set of irreducible instances of such a basic formula (see [Com88b, Com88a]). Let us show briefly how it works on a very simple example:

Example 5 F consists of the four function symbols $0, s, p, +$. 0 is nullary, s, p are unary and $+$ is binary. The rewrite rules are:

$$\mathcal{R} = \left\{ \begin{array}{rcl} s(p(y)) & \rightarrow & y \\ p(s(y)) & \rightarrow & y \\ 0 + y & \rightarrow & y \\ s(y_1) + y_2 & \rightarrow & s(y_1 + y_2) \\ p(y_1) + y_2 & \rightarrow & p(y_1 + y_2) \end{array} \right.$$

Note that NF is $\{0, s(0), s^2(0), \ldots, p(0), p^2(0), \ldots\}$.

Let $\phi \equiv \forall y, y' : x \neq s(y) \wedge x \neq p(y) \wedge x \neq 0$. Applying the rules of section 4.5 yields the following:

$$\psi \equiv \exists w_1, w_2 : x = w_1 + w_2$$

The only problem is now to decide which ground terms $w_1 + w_2$ are indeed irreducible. Let $NF_{w_1+w_2}$ denote all irreducible ground instances of $w_1 + w_2$. This set is again approximated by solving the problem:

$$\forall y_1, y_2 : w_1 + w_2 \neq 0 + y_1 \wedge w_1 + w_2 \neq s(y_1) + y_2 \wedge w_1 + w_2 \neq p(y_1) + y_2$$

which states that $w_1 + w_2$ should not be reducible at the root : it is different from all left hand sides. We find an equivalent formula :

$$\exists w_3, w_4, w_5 : w_1 = w_3 + w_4 \wedge w_2 = w_5$$

Considering only irreducible solutions leads to the relation ($+$ denotes the symbol function, not the union of languages).

$$NF_{w_1+w_2} = NF_{w_3+w_4} + NF_{w_5}$$

Actually, since $NF_{w_1+w_2} = NF_{w_3+w_4}$, this is a fixed point equation which does not have any solution. Therefore $NF_{w_1+w_2} = \emptyset$ and ϕ does not have any irreducible ground solution.

Of course, this is a very simple example. However, as shown in [Com88b, Com88a] the method is general: it is always possible to compute a (conditional) tree grammar for the language of irreducible ground instances of a term t. Moreover, emptiness is decidable for such grammars.

7 Ongoing Research

Since the first draft of this survey, a lot of work has been done on disunification, solving some of the open questions. Disunification modulo associativity and commutativity has been shown undecidable [Tre90]. More precisely, [Tre90] shows the undecidability of the Σ_3 fragment of the theory of $T(F)/_{=_{AC}}$ when there is (at least) one AC function symbol, one binary function symbol and one constant. On the other hand, the theory is decidable when there are only constants and one AC function symbol because it can be reduced to Presburger's arithmetic (R. Treinen, private communication).

The existential fragment of the theory of lexicographic path ordering has been shown decidable [Com90c]. This result has been extended to other ordinal notations (in particular, recursive path orderings) [JO91]. Venkataraman's results on the theory of subterm ordering have been extended to rational trees [Tul91].

The set of transformation rules we gave in this survey for disunification in finite trees (with slight modifications) has been shown confluent [Del90], showing that $\approx_{T(F)}$ can be decided by rewriting the formulas until a solved form is reached.

Results similar to those of [LM87] have been shown in the case of rational trees [MS90]. Let us also mention some general work on equational problems in equational theories [BSS89] and a study of complete extensions of $E_{FT} \cup \{DCA\}$ [MT89].

Nevertheless, many open questions remain, and we conclude this survey by reviewing some of them.

The general study of disunification in equational theories (we may call it "universal disunification" because the set of axioms E is not precised) has received little attention. (The only attempt we know of is [BSS89].) Considering arbitrary first order formulas and arbitrary equational theories lead to the full undecidability of disunification. However, for some restricted kinds of formulas, universal disunification may become semi-decidable. There is a gap between the classes of formulas for which universal disunification is known undecidable and classes of formulas for which universal disunification is known semi-decidable. An open problem is to bridge this gap. [11]

On the other hand, it is possible to restrict the equational theories under consideration. And it is possible to restrict both formulas and equational theories. This has been investigated in some very restricted classes of theories only. There are some equational theories (e.g. boolean rings) for which word problems and unification problems are decidable. Is the first order theory of the corresponding quotient algebra decidable ? Or even if the first order theory is obvioulsy undecidable (e.g. for free abelian groups), is the existential fragment decidable? semi-decidable ? Even in the AC-case, it remains a little gap: we do not know any result when there is only one AC symbol, one unary symbol and constants. Also of interest for practical applications (e.g. compiling AC matching) is the solution of complement problems in case of AC function symbols. As complement problems form a subset of the Σ_2 fragment, this problem is still open.[12]

The completeness results with respect to unification problems that are given in this paper are not completely satisfactory since we would like to consider $\mathcal{C}_u$ alone instead of $\mathcal{C}_{su}$. We have proposed a set of rules,thought to be correct, terminating and complete with respect to $\mathcal{C}_u$ and $\mathcal{SF}_u$ but the proof has not yet been worked out.

Concerning (symbolic) ordering constraints, we still do not know whether the first order theory of a recursive path ordering extending a total precedence is decidable. (Treinen's result apply to any non-total precedence.) We do not know either whether the Σ_1 fragment is decidable in the case of a non-total pecedence.

Heintze and Jaffar study in [HJ90] *set constraints*. Such constraints are formulas $exp_1 \supseteq exp_2$ where exp_1, exp_2 are expressions involving usual operations on sets $(\cap, \cup, \neg)$ and also function application (from a fixed alphabet F) and projections. The formulas are interpreted in subsets of the Herbrand Universe. [HJ90] show the decidability of what they call *definite sets constraints* which are a fragment of set constraints. The decidability of general set constraints is still open.

The last question is the complexity of disunification; we didn't really give algorithms but rather some general techniques for deriving algorithms depending on the problem at hand. How complex can we expect the algorithms? Maher [Mah88b] showed that the validity in $T(F)$ of a boolean combination of simple formulas is NP-complete[13]. However, this is only a partial answer. First, this gives only a lower bound for the complexity of disunification in the general case. Second, this result only consider one interpretation and one kind of formulas. Finally, this doesn't provide a "best" control: what is the "best" use of the transformation rules? In every running implementation one has first to give a deterministic control for the application of the rules. The question of which control we should implement then arises. There are two actual implementations [Meh88, Del89] (to my knowledge these are the only one up to now). The implementations allow to make some experiments on the strategies but no general conclusion has been reached up to now.

[11] This problem has been recently addressed in [Fer91].
[12] Complement problems modulo AC have been recently solved in [KLP91].
[13] Similar results in the case of rational trees [MS90] have been proved since.

Acknowledgments

Many discussions were helpful to prepare this manuscript. I would like to thank in particular R. Caferra, G. Chaminade, J.-P. Jouannaud, C. Kirchner, J.-L. Lassez, A. Lux, M. Rodriguez-Artalejo and Ph. Schnoebelen.

References

[BB89] W. L. Buntine and H. J. Bürckert. On solving equations and disequations. SEKI Report SR-89-03, Univ. Kaiserslautern, January 1989.

[BMPT90] R. Barbuti, P. Mancarella, D. Pedreshi, and F. Turini. A transformation approach to negation in logic programming. *Journal of LOgic Programming.*, 8:201–228, 1990.

[BSS89] Hans-Jurgen Bürckert and Manfred Schmidt-Schauß. On the solvability of equational problems. SEKI Report SR-89-07, Univ. Kaiserslautern, 1989.

[Bür88] H. J. Bürckert. Solving disequations in equational theories. In *Proc. 9th Conf. on Automated Deduction, Argonne, LNCS 310*. Springer-Verlag, May 1988.

[CL89] Hubert Comon and Pierre Lescanne. Equational problems and disunification. *J. Symbolic Computation*, 7:371–425, 1989.

[Col84] A. Colmerauer. Equations and inequations on finite and infinite trees. In *FGCS'84 Proceedings*, pages 85–99, November 1984.

[Col88] A. Colmerauer. Une introduction à PROLOG III. Unpublished draft, 1988.

[Com86] Hubert Comon. Sufficient completeness, term rewriting systems and anti-unification. In *Proc. 8th Conf. on Automated Deduction, Oxford, LNCS 230*, pages 128–140. Springer-Verlag, July 1986.

[Com88a] Hubert Comon. An effective method for handling initial algebras. In *Proc. 1st Workshop on Algebraic and Logic Programming, Gaussig, LNCS 343*. Springer-Verlag, November 1988.

[Com88b] Hubert Comon. Unification et disunification: Théorie et applications. Thèse de Doctorat, Institut National Polytechnique de Grenoble, France, 1988.

[Com89] Hubert Comon. Inductive proofs by specifications transformation. In *Proc. 3rd Rewriting Techniques and Applications 89, Chapel Hill, LNCS 355*, pages 76–91. Springer-Verlag, April 1989.

[Com90a] H. Comon. Solving symbolic ordering constraints. *International Journal on Foundations of Computer Science*, 1(4), 1990.

[Com90b] Hubert Comon. Equational formulas in order-sorted algebras. In *Proc. ICALP, Warwick*. Springer-Verlag, July 1990.

[Com90c] Hubert Comon. Solving inequations in term algebras. In *Proc. 5th IEEE Symposium on Logic in Computer Science, Philadelphia*, June 1990.

[Com91a] Hubert Comon. Complete axiomatizations of some quotient term algebras. In *Proc. ICALP, Madrid*, July 1991. To appear.

[Com91b] Hubert Comon. Complete axiomatizations of some quotient term algebras. Research Report 650, LRI and CNRS, Univ. Paris-Sud, 1991.

[Cou81] B. Courcelle. Fundamental properties of infinite trees. In *Proc. Int. Summer School on Theoretical Foundations of Programming Methodology, Munich*, 1981.

[Dav77] Martin Davis. Unsolvable problems. In J. Barwise, editor, *Handbook of Mathematical Logic*, pages 567–594. North-Holland, 1977.

[Del89] Catherine Delor. *Transformation de problèmes équationnels.* Internal Report , Ecole Normale Supérieure, Paris (in french), 1989.

[Del90] C. Delor. Etude de la confluence d'un système de règles. Laboratoire d'Informatique de l'Ecole Normale Supérieure, Paris. Unpublished, 1990.

[Der87] Nachum Dershowitz. Termination of rewriting. *J. Symbolic Computation*, 3(1):69–115, February 1987.

[DJ90] Nachum Dershowitz and Jean-Pierre Jouannaud. Rewrite systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 243–309. North-Holland, 1990.

[DSH87] M. Dincbas, Helmut Simonis, and P. Van Hentenryck. Extending equation solving and constraint handling in logic programming. In *Proc. Colloquium on Resolution of Equations in Algebraic Structures*, Austin, Texas, 1987.

[Fer91] Maribel Fernandez. Equational disunification. Research Report 684, Laboratoire de Recherche en Informatique, Univ. Paris-Sud, July 1991.

[GH78] J. V. Guttag and J. J. Horning. The algebraic specification of abstract data types. *Acta Informatica*, 10:27–52, 1978.

[GM87] Joseph Goguen and José Meseguer. Order-sorted algebra I: Partial and overloaded operators, errors and inheritance. Draft, Computer Science Lab., SRI International, 1987.

[GS87] J. Gallier and W. Snyder. A general complete E-unification procedure. In *Proc. Rewriting Techniques and Applications 87, Bordeaux, LNCS 256.* Springer-Verlag, May 1987.

[HJ90] Nevin Heintze and Joxan Jaffar. A decision procedure for a class of set constraints. In *Proc. 5th IEEE Symp. Logic in Computer Science, Philadelphia*, June 1990.

[HR87] J. Hsiang and M. Rusinowitch. Proving refutational completeness of theorem proving strategies, Part I: The transfinite semantic tree method. Submitted to jacm, 1987.

[Hue76] G. Huet. *Résolution d'équations dans les langages d'ordre* $1, 2, \ldots \omega$. Thèse d'Etat, Univ. Paris 7, 1976.

[JL87] J. Jaffar and J.-L. Lassez. Constraint logic programming. In *Proc. 14th ACM Symp. Principles of Programming Languages, Munich*, 1987.

[JO91] Jean-Pierre Jouannaud and Mitsuhiro Okada. Satisfiability of systems of ordinal notations with the subterm property is decidable. In *Proc. ICALP 91, LNCS*, Madrid, 1991.

[Kir86] Claude Kirchner. Computing unification algorithms. In *Proc. 1st IEEE Symp. Logic in Computer Science, Cambridge, Mass.*, pages 206–216, 1986.

[KK89] Claude Kirchner and Hélène Kirchner. Constraint equational reasoning. In *Proc. ISSAC 89, Portland, Oregon*, 1989.

[KL87] Claude Kirchner and Pierre Lescanne. Solving disequations. In *Proc. 2nd IEEE Symp. Logic in Computer Science, Ithaca, NY*, pages 347–352, 1987.

[KLP91] E. Kounalis, D. Lugiez, and L. Pottier. Complement problems modulo associativity and commutativity. In *Proc. 16th Mathematical Foundations of Computer Science 90, Warsaw.* Springer-Verlag, 1991.

[KNRZ91] Deepak Kapur, Paliath Narendran, Daniel Rosenkratz, and Hantao Zang. Sufficient-completeness, ground-reducibility and their complexity. *Acta Informatica*, 28:311–350, 1991.

[Kuc88] G. A. Kucherov. A new quasi-reducibility testing algorithm and its applications to proofs by induction. In *Proc. 1st Workshop on Algebraic and Logic Programming, Gaussig, LNCS 343.* Springer-Verlag, November 1988.

[Kun87a] K. Kunen. Answer sets and negation as failure. In *Proc. 4th. Int. Conf. on Logic Programming, Melbourne*, pages 219–228, 1987.

[Kun87b] K. Kunen. Negation in logic programming. *J. Logic Programming*, 4:289–308, 1987.

[Lav87] A. Laville. Lazy pattern matching in the ML language. In *Proc. 7th Conf. Found. of Software Technology and Theoretical Computer Science, Pune, INDIA, LNCS 287*, December 1987.

[Lav88] A. Laville. *Evaluation Paresseuse des Filtrages avec Priorité. Application au langage ML.* Thèse de Doctorat, Univ. Paris 7, February 1988.

[LM87] J.-L. Lassez and K. G. Marriott. Explicit representation of terms defined by counter examples. *J. Automated Reasoning*, 3(3):1–17, September 1987.

[LM89] J.-L. Lassez and K. McAloon. Independence of negative constraints. In *Proc. TAPSOFT 89 (Vol. 1), Barcelona, LNCS 351.* Springer-Verlag, March 1989.

[LM90] Jean-Louis Lassez and Ken MacAloon. A constraint sequent calculus. In *Proc. 5th IEEE Symp. Logic in Computer Science, Philadelphia*, June 1990.

[LMM86] J.-L. Lassez, M. J. Maher, and K. G. Marriot. Unification revisited. In *Proc. Workshop on Found. of Logic and Functional Programming, Trento, LNCS 306*. Springer-Verlag, December 1986.

[Lug89] Denis Lugiez. A deduction procedure for first order programs. In G. Levi and M. Martelli, editors, *Proc. Int. Conf. on Logic Programming*, Lisbon, 1989. MIT Press.

[Mah88a] M. J. Maher. Complete axiomatizations of the algebras of finite, rational and infinite trees. In *Proc. 3rd IEEE Symp. Logic in Computer Science, Edinburgh*, pages 348–357, July 1988.

[Mah88b] M. J. Maher. Complexity of disunification, 1988. Private communication.

[Mal71] A. I. Mal'cev. Axiomatizable classes of locally free algebras of various types. In *The Metamathematics of Algebraic Systems. Collected Papers. 1936-1967*, pages 262–289. North-Holland, 1971.

[Meh88] M. Mehl. *Gleichungsdefinierte Probleme. Lösungsmethoden und Anwendungen bei algebraischen Spezifikationen*. Projektarbeit, Universität Kaiserslautern, 1988. In german.

[MS90] Michael J. Maher and Peter J. Stuckey. On inductive inference of cyclic structures. In F. Hoffman, editor, *Annals of Mathematics and Artificial Intelligence*, volume F. J.C. Baltzer Scientific Pub. Company, 1990. To appear.

[MSK90] C. Mohan, M. Srivas, and D. Kapur. Inference rules and proof procedures for inequations. *J. Logic Programming*, 9:75–104, 1990.

[MT89] G. Marongiu and S. Tulipani. Quantifiers elimination for complete extensions of clark equational theory and infinite terms. Unpublished Draft, 1989.

[Nic87] Peter Nickolas. The representation of answers to logical queries. Research Report 87/11, Department of Computing Science, University of Wollongong, Australia, December 1987.

[NO80] G. Nelson and D. C. Oppen. Fast decision procedures based on congruence closure. *Journal of the ACM*, 27:356–364, 1980.

[O'D86] M. J. O'Donnell. *Equational Logic as a Programming Language*. MIT Press, 1986.

[PS90] Laurence Puel and Ascander Suarez. Compiling pattern matching by term decomposition. In *Proc. Int. Conf. on LISP and Functional Programming*, 1990.

[Pue87] Laurence Puel. Bons préordres sur les arbres associés à des ensembles inévitables et preuves de terminaison des systèmes de réécriture. Thèse de Doctorat d'Etat, Université Paris VII, December 1987.

[Pug90] J-F. Puget. *Apprentissage et Programmation Logique*. PhD thesis, Université de Paris-Sud, Orsay, 1990.

[Rab77] M. Rabin. Decidable theories. In J. Barwise, editor, *Handbook of Mathematical Logic*, pages 595–629. North-Holland, 1977.

[Sch87] Ph. Schnoebelen. Rewriting techniques for the temporal analysis of communicating processes. In *Proc. PARLE 87, vol. II: Parallel Languages, Eindhoven, LNCS 259*, pages 402–419. Springer-Verlag, June 1987.

[Sch88a] Ph. Schnoebelen. Refined compilation of pattern-matching for functional languages. In *Proc. 1st Workshop on Algebraic and Logic Programming, Gaussig, LNCS 343*. Springer-Verlag, November 1988.

[Sch88b] Ph. Schnoebelen. Refined compilation of pattern-matching for functional languages. *Science of Computer Programming*, 11(2):133–159, December 1988.

[Sho67] J. R. Shoenfield. *Mathematical Logic*. Addison-Wesley, 1967.

[SJ89] Ph. Schnoebelen and Ph. Jorrand. Principles of FP2. Term algebras for specification of parallel machines. In J. W. de Bakker, editor, *Languages for Parallel Architectures: Design, Semantics, Implementation Models*, chapter 5, pages 223–273. Wiley, 1989.

[SNMG87] Gert Smolka, W. Nutt, José Meseguer, and Joseph Goguen. Order-sorted equational computation. In *Proc. Coll. on the Resolution of Equations in Algebraic Structures, Lakeway*, May 1987.

[Sti81] M. Stickel. A unification algorithm for associative-commutative functions. *Journal of the ACM*, 28(3):423–434, 1981.

[Thi84] J.-J. Thiel. Stop loosing sleep over incomplete specifications. In *Proc. 11th ACM Symp. Principles of Programming Languages, Salt Lake City*, 1984.

[Tho86] S. Thompson. Laws in Miranda. In *Proc. 86 ACM Conf. Lisp and Functional Programming, Cambridge, Mass.*, August 1986.

[Tre90] R. Treinen. A new method for undecidability proofs of first order theories. Tech. Report A-09/90, Universität des Saarladandes, Saarbrücken, May 1990.

[Tul91] Sauro Tulipani. Decidability of the existential theory of infinite terms with subterm relation. Unpublished Draft, October 1991.

[Ven87] K. N. Venkataraman. Decidability of the purely existential fragment of the theory of term algebras. *Journal of the ACM*, 34(2):492–510, 1987.