Survey Document for a CP 2001 Tutorial

Complexity of Constraint Satisfaction Problems

Nadia Creignou
LIM (CNRS, FRE 2246),
Université de la Méditerranée
Marseille
France
Nadia.Creignou@lim.univ-mrs.fr

Miki Hermann
LORIA (CNRS, UMR 7503)
BP 239
54506 Vandœuvre-lès-Nancy
France
Miki.Hermann@loria.fr

Reinhard Pichler
Technische Universität Wien
and
Siemens AG Austria
Wien
Austria
reini@logic.at

Remark: This document is an overview, from a computational complexity standpoint, of several constraint satisfaction problems. We assume that the reader is familiar with the usual notions and definitions used in the complexity theory, like polynomial time, NP-complete or coNP-complete problems. We also use some necessary notions from counting complexity, like the classes FP or #P, or from approximation theory, that became largely known in the recent years. Therefore we do not recall the definitions of these notions in this document. However, a reader who is not familiar with these concepts can find more information in the book [Pap94] or in the survey [Joh90].

1 Complexity of Constraint Satisfaction Problems on Boolean and Finite Domains

Constraint satisfaction problems occur commonly in practice, optimization and in settings arising from artificial intelligence. This section is devoted to complexity results for such problems on Boolean and finite domains. The question of identifying restrictions to the general problem that are sufficient to ensure tractability is important for both a practical and a theoretical viewpoint and has been extensively studied. Such restrictions may either involve the structure of the constraints or they may involve the "nature" of the constraints. Here we take the second approach and we study the complexity of deciding satisfiability of a given constraint satisfaction problem as a function of the "nature" of the constraints.

We are interested in an infinite class of constraint satisfaction problems. Informally speaking, a problem in this class is characterized by a finite collection of finitely specified constraint templates, say $\mathcal{F}$. An instance of such a problem, called the $\mathsf{CSP}(\mathcal{F})$ problem, consists of n variables and m constraints applied to various subsets of the given variables such that each constraint is drawn from

the collection $\mathcal{F}$. The computational objective is to determine whether there is an assignment to the input variables which satisfies all the given constraints.

Example 1.1

- An instance of 2-SAT is a collection of Boolean binary constraints on n variables. Each clause is a constraint that rules out certain assignments to the variables.
- An instance of 3-colorability is a graph which can be seen as a collection of binary constraints ($\neq$) over the finite domain $D = \{0, 1, 2\}$.

We are interested in studying the complexity of answering the above question for every problem CSP($\mathcal{F}$). In view of the two examples above it becomes clear that this class captures polynomially solvable problems as well as intractable (NP-complete) ones. There are at least two different motivations to do such a study. On the one hand a complete complexity classification for an infinite family of problems is important for the design of good algorithms since it delineates the boundary between tractable and intractable cases. We will see that the classifications obtained highlight some problems as central when it comes to propose an efficient algorithm. On the other hand, constraint satisfaction problems are an excellent testbed for abstracting some global inferences about the nature of computation and may provide very useful hints at the ultimate goal of complexity theory: identify what renders some problems hard whereas some others seemingly very similar are easy.

1.1 Constraint satisfaction problems over Boolean and finite domains

As we have seen above two examples of constraint satisfaction problems are (1) 2-SAT and (2) 3-colorability. The difference between these two problems is the nature of the underlying constraints. In order to specify a computational problem in term of its underlying constraint structure, one needs a finite specification of the set of the constraints. In order to achieve this objective, we distinguish constraints from their applications. For example there are $\Omega(n^2)$ different clauses of length 2, when applied to n Boolean variables. However it is clear that the underlying template only needs to include all the different constraints on 2 variables; and the rest can be achieved by specifying to which ordered subset of variables is a basic constraint applied. This distinction between constraints and their applications is formalized in [CKS01] and reproduced next. Once we formalize this distinction, we present the class of constraint satisfaction problems that we will study.

Throughout this section D denotes a finite domain of cardinality r, $r \geq 2$, $D = \{0, 1, \ldots, r-1\}$.

Definition 1.2 [Constraint] A constraint is a Boolean function $f\colon D^k \longrightarrow \{0, 1\}$, where k is a non-negative integer called the arity of f.

Definition 1.3 [Constraint application] Given a constraint $f\colon D^k \longrightarrow \{0, 1\}$ and $(i_1, \ldots, i_k)$, the pair $\langle f, (i_1, \ldots, i_k)\rangle)$ is referred to as an application of the constraint f to $x_{i_1}, \ldots, x_{i_k}$.

Definition 1.4 [Satisfiable constraint application] Every assignment $\phi\colon \{x_1, \ldots, x_n\} \longrightarrow D$ naturally extends itself to any constraint application $C = \langle f, (i_1, \ldots, i_k)\rangle$, we have

$$\phi(C) \quad := \quad f(\phi(x_{i_1}), \ldots, \phi(x_{i_k})).$$

An assignment ϕ satisfies a constraint application C if $\phi(C) = 1$.

Throughout this section, $\mathcal{F}$ will denote a finite set of constraints over the domain D.

Definition 1.5 [$\mathcal{F}$-collection of constraint applications] Let $\mathcal{F}$ be a finite collection of constraints. A constraint application of the form $\langle f, (i_1, \ldots, i_k)\rangle$ where $f \in \mathcal{F}$ is referred to as an $\mathcal{F}$-constraint application. A collection of m such constraints applications is called an $\mathcal{F}$-collection of m constraint applications. Such a collection is satisfiable if there exists an assignment which satisfies all of its constraint applications.

Definition 1.6 [Constraint Satisfaction Problem ($\mathsf{CSP}(\mathcal{F})$)] The *constraint satisfaction problem* $\mathsf{CSP}(\mathcal{F})$ is to decide whether there exists an assignment that satisfies a given $\mathcal{F}$-collection of constraint applications.

Example 1.7

- The classical 2-SAT problem is the same as $\mathsf{CSP}(\{f_0, f_1, f_2\})$, where $f_i\colon \{0,1\}^2 \longrightarrow \{0,1\}$ for $i = 0, \ldots, 2$ and
  $f_0(x, y) = 0$ if and only if $(x, y) = (0, 0)$,
  $f_1(x, y) = 0$ if and only if $(x, y) = (1, 0)$,
  $f_2(x, y) = 0$ if and only if $(x, y) = (1, 1)$.
- The 3-colorability problem can be seen as $\mathsf{CSP}(\{g\})$

  where $g\colon \{0,1,2\}^2 \longrightarrow \{0,1\}$ and $g^{-1}(1) = \{(0,1), (0,2), (1,0), (1,2), (2,0), (2,1)\}$. Each edge of the graph given in input consists of the constraint g applied to its two endpoints.

1.2 Complexity of Boolean Constraint Satisfaction Problems

A significant amount of research effort was oriented towards studying the constraint satisfaction problems on Boolean domain (which are nothing else but generalized satisfiability problems, see [CKS01] for a uniform survey). There is now a growing body of complexity classification results for problems derived from Boolean constraint satisfaction. Schaefer [Sch78] began this line of research in 1978. In seek of a didactical presentation we will focus our attention on Schaefer's remarkable result and few of its extensions in giving the gist of the proof.

Throughout this subsection $\mathsf{CSP}(\mathcal{F})$ problems will be denoted by $\mathsf{SAT}(\mathcal{F})$ in order to remind the reader that we are dealing with Boolean domains.

Schaefer was interested in the complexity of deciding whether an instance of $\mathsf{SAT}(\mathcal{F})$ is satisfiable for every problem $\mathsf{SAT}(\mathcal{F})$. His study led to a strikingly simple answer: every problem in this class is either in P or NP-complete. This result is surprising and unexpected for several reasons. First prior to this result NP-completeness was established on a problem by problem basis. Schaefer's result gives a uniform proof to establish NP-completeness for an infinite collection of problems. Furthermore, rarely in complexity theory one comes across an infinite class of problems where every problem belongs to a finite collection of computational equivalence classes.

Let us now give some definitions in order to state his result.

Definition 1.8 A constraint f is said to be

0-valid if $f(0, \ldots, 0) = 1$.

1-valid if $f(1, \ldots, 1) = 1$.

weakly positive (weakly negative) if f is expressible as a CNF-formula having at most one negated (unnegated[1]) variable in each clause.

bijunctive if f is expressible as a 2CNF-formula.

affine if f is expressible as a system of linear equations over GF(2); that is, it is equivalent to a system of linear equations of the forms $v_1 \oplus \cdots \oplus v_n = 0$ and $v_1 \oplus \cdots \oplus v_n = 1$, where $\oplus$ denotes the exclusive or connective.

Theorem 1.9 *[Sch78] Given a constraint set $\mathcal{F}$, the problem* $\mathsf{SAT}(\mathcal{F})$ *is in* P *if $\mathcal{F}$ satisfies one of the conditions below, and* $\mathsf{SAT}(\mathcal{F})$ *is* NP*-complete, otherwise.*

1. *Every constraint in $\mathcal{F}$ is* 0*-valid (*1*-valid).*
2. *Every constraint in $\mathcal{F}$ is bijunctive.*
3. *Every constraint in $\mathcal{F}$ is weakly positive (weakly negative).*
4. *Every constraint in $\mathcal{F}$ is affine.*

Let us now analyze this result. A quick look to its statement leads to the following observation: easy problems are well-known! Indeed the central problems for the polynomial cases are the classical tractable satisfiability problems: 2-SAT, Horn SAT and deciding the consistency of a linear system over GF(2). For instance if every constraint in $\mathcal{F}$ is bijunctive then every collection of constraints applications can be expressed as a 2-CNF-formula, whose satisfiability can be decided in linear time [APT79]. On the other hand intractable cases need the most effort. If $\mathcal{F}$ does not satisfy any of the above conditions then Schaefer proved that one can "encode" the NP-hard constraint One-In-Three with $\mathcal{F}$-constraints (this "encoding", which has to preserve the satisfiability, is rigorously formalized in [CKS01] as *perfect implementation*). The proof is based on algebraic characterizations of the properties described above. For instance it is well-known that a constraint f is weakly negative (i.e. Horn) if and only it is closed under direct product, that is to say if and only for all $\vec{s}_1$ and $\vec{s}_2$ such that $f(\vec{s}_1) = 1$ and $f(\vec{s}_2) = 1$, the direct product $\vec{s}_1 \cap \vec{s}_2$ is also satisfying. Bijunctive and affine constraints are characterized by similar algebraic closure properties.

Schaefer's concise characterization allows us to determine whether for a given $\mathcal{F}$, $\mathsf{SAT}(\mathcal{F})$ is in P or is NP-complete. Then, a natural question is: what is the complexity of identifying tractable problems. In other words, how difficult is it to recognize that the problem specified by a given constraint set is indeed tractable. The question was settled in [CKS01].

Proposition 1.10 [CKS01] *Let $\mathcal{F}$ be a constraints set. Suppose that each constraint in $\mathcal{F}$ is specified by a CNF-formula. Then the problem of deciding whether* $\mathsf{SAT}(\mathcal{F})$ *is in* P *is* coNP*-hard.*

Many results extend Schaefer's study and explore different kinds of complexity classes restricted to Constraint Satisfaction Problems. Their authors have shown classification results for a variety of computational tasks where the goal of the computation varies while the instance remains the same. For instance we studied the counting version of the Boolean constraint satisfaction problems where the objective is to count the number of assignments satisfying all constraints. We obtained a dichotomy result FP/ #P (where #P is the counting counterpart of NP, see [Pap94, Chapter 18]).

[1] Such clauses are usually called Horn clauses.

Theorem 1.11 [CH96] *Given a constraint set $\mathcal{F}$, if every constraint in $\mathcal{F}$ is affine then the problem* #SAT($\mathcal{F}$) *is in* FP*, and* #SAT($\mathcal{F}$) *is* #P*-complete, otherwise.*

Let us compare this result to Schaefer's one. Most of the central tractable problems for the decision task become hard when it comes to counting. Indeed, while the decision problems 2-SAT and Horn-SAT are in P, the corresponding counting problems #2-SAT and #Horn-SAT are #P-complete (see [Val79]). Only counting the number of solutions of a linear system over a finite field is as easy as deciding its consistency via Gaussian elimination. So, once more this result reveals that easy problems are well-known. The #P-complete case is obtained by an "encoding" preserving the number of solutions (referred to as *faithful reduction* in [CKS01]) from the problems Positive-2-SAT [Val79] and Implicative-2-SAT [PB83] which are known to be #P-complete.

As a corollary observe that NP-completeness for a satisfiability problem implies the #P-completeness of the corresponding counting problem. this result confirms for an infinite and general class of problems the intuitive opinion that each NP-complete problem leads to a corresponding #P-complete counting problem.

In this line of research Juban [Jub99] proved a dichotomy theorem for the Unique Satisfiability problem. Another closely related aspect is that of enumerating all solutions (without duplicate). Creignou and Hébrard refined Schaefer's result in identifying satisfiability problems for which all solutions can be generated in *polynomial delay* (see [JYP88]).

Theorem 1.12 [CH97] *Given a constraint set $\mathcal{F}$, the problem of generating all models for any given $\mathcal{F}$-collection of constraints has a polynomial space, polynomial delay algorithm if $\mathcal{F}$ satisfies one of the conditions below, and otherwise, no such algorithm exists unless* P=NP*.*

1. *Every constraint in $\mathcal{F}$ is weakly positive (weakly negative).*
2. *Every constraint in $\mathcal{F}$ is affine.*
3. *Every constraint in $\mathcal{F}$ is bijunctive.*

Another problem of interest is to evaluate a quantified collection of constraint applications. Such a quantified collection is of the form $\mathcal{Q}_1 x_1 \cdots \mathcal{Q}_n x_n \mathcal{C}$, where $\mathcal{C}$ is an $\mathcal{F}$-collection of constraints over the set of variables $\{x_1, \ldots, x_n\}$ and $\mathcal{Q}_i$ is either the quantifier "for all" or "exists", for $i = 1, \ldots, n$. The quantified satisfiability problem, QSAT($\mathcal{F}$) is to decide whether a given quantified $\mathcal{F}$-collection of constraints application is true. Quantified satisfiability problems form a natural subclass of PSPACE problems; and include some PSPACE-complete problems. There is also a dichotomy result for quantified constraint satisfaction problems (which was first stated in [Sch78] and independently proved in [CKS01] and in [Dal97]).

Theorem 1.13 *Given a constraint set $\mathcal{F}$, if $\mathcal{F}$ satisfies one of the following three conditions then* QSAT($\mathcal{F}$) *is in* P *otherwise it is* PSPACE*-complete*

1. *Every constraint in $\mathcal{F}$ is weakly positive (weakly negative).*
2. *Every constraint in $\mathcal{F}$ is affine.*
3. *Every constraint in $\mathcal{F}$ is bijunctive.*

Many other results deal with optimization problems. Creignou [Cre95] and, Khanna and Sudan [KS96] independently studied optimization problems where the objective is to maximize the number of satisfied constraints. They showed that every problem Max$\mathsf{SAT}(\mathcal{F})$ is either in P or APX-complete. In this result the central problem for the polynomial case is the s,t-Min-Cut problem, whereas the APX-complete case is obtained by an approximation-preserving encoding (referred to as *strict implementation* in [CKS01]) from the Max-Cut problem. Khanna *et al.* [KSW97, KSL97] studied other forms of commonly occurring optimization tasks in the constraint satisfiability setting and obtained classification results. Their results [KSW97, KSL97] are somewhat different from the others above in that the resulting classification theorems do not exhibit dichotomies but rather a partition into a larger but finite number of equivalence classes. Reith and Vollmer [RV00] studied the class of optimization problems where the objective is to find a lexicographically minimal (maximal) satisfying assignment.

Other people studied similar aspects of constraint satisfaction problems. Kavvadias and Sideri studied in [KS98] the inverse satisfiability problem where the goal is to find an $\mathcal{F}$-collection of constraint applications such that a given set of truth assignments constitutes its set of feasible solutions. Cadoli [Cad92] proved that testing for truth assignment minimality of a given propositional formula is coNP-complete, whereas Kirousis and Kolaitis [KK00, KK01] presented a dichotomy theorem for propositional circumscription. They investigated the class of decision problems Min-$\mathsf{SAT}(\mathcal{F})$ that ask whether a satisfying truth assignment for an $\mathcal{F}$-collection of constraint applications is minimal with respect to the coordinate-wise partial order. Their dichotomy theorem separates coNP-complete instances from instances in P. Durand, Hermann, and Kolaitis [DHK00] proved that counting the number of minimal truth assignments of a propositional formula is a #NP-complete problem. Böhler, Hemaspaandra, Reith and Vollmer considered the problem of determining whether two given $\mathcal{F}$-collections of constraint applications are equivalent in the sense that they possess the same set of satisfying assignments [BHRV01].

1.3 Complexity of Constraint Satisfaction Problems on non-Boolean finite domains

A complete complexity classification for all constraint satisfaction problems over arbitrary finite domains is an open problem and is a highly challenging task of much more than technical interest (see [FV98]).

There are some complete classification results when we restrict our attention to binary constraint satisfaction problems. First Hell and Nešetřil [HN90] obtained a dichotomy theorem for the H-coloring problem, in which the question is that of deciding whether there exists any homomorphism from a given graph G to the fixed graph H. They showed that the decision problem is in P if H has a loop or is bipartite; otherwise it is NP-complete. Dyer and Greenhill [DG00] considered the problem of exactly counting such homomorphisms and gave a similar complete characterization. They showed that counting is in FP if every connected component of H is an isolated vertex without loop, or a complete graph with all loops present or a complete unlooped bipartite graph; otherwise it is #P-complete. Cooper, Cohen and Jeavons [CCJ94] studied the complexity of $\mathsf{CSP}(\mathcal{F})$ when $\mathcal{F}$ is a set of binary constraints under the additional hypothesis that $\mathcal{F}$ is closed under two operations, domain restriction and label permutation. They proved that satisfiability can be decided in polynomial time if all constraints belong to a special class of constraints, called *0/1/all* or *implicational* constraint, and is NP-complete in all other cases. Their work was extended by Istrate [Ist97] who, under the same conditions, obtained a dichotomy classification for both

the corresponding counting problems (FP/#P-complete) and the optimization problems (P/APX-complete).

In a more general framework Jeavons, Cohen and Gyssens [JCG97] brought to the fore the link between the algebraic closure properties of the constraints and the complexity of the corresponding constraint satisfaction problem. It appears that when restricted to Boolean domain algebraic closure properties of the constraints (see [CKS01, Chapter 4, Section 4.4 and Chapter 6]) exactly characterizes the complexity of the corresponding constraint satisfaction problem. Jeavons, Cohen and Gyssens proved that any set of constraints that does not give rise to an NP-complete class of problems must satisfy a certain type of algebraic closure condition. Then, they investigated all the different possible forms of the algebraic closure property and established which of these are sufficient to ensure tractability. A number of tractable constraint classes have also been identified by Feder and Vardi [FV98]. In a highly nontrivial proof they pointed out that constraint satisfaction problems over non-Boolean domains are computationally equivalent to problems in "monotone monadic SNP", a syntactically restricted class of languages within NP which is, in some sense, the largest class within NP that may show dichotomy results.

Despite all these efforts the complexity classification of constraint satisfaction problems over finite domains is still incomplete.

2 Complexity of Equational Constraint Problems

Equational problems are first-order formulas with quantifier prefix $\exists^* \forall^*$, whose only predicate symbol is syntactic equality. They are an important tool in many areas of computer science.

In *automated deduction*, equational constraints can be used to restrict the set of ground instances of a clause. It is thus possible to define stronger redundancy criteria and hence, in general, more efficient theorem provers (cf. [CZ90, CP95b, CP95a]). In *automated model building*, equational problems can be used in several ways, e.g.: for model construction, for model representation, for the evaluation of clauses in a given model, etc (cf. [CZ90, FL96]). Complement problems are an important special case of equational problems with applications in *logic programming*, *functional programming*, *machine learning*, etc. (cf. [LM87, LMM91, Lug89, SM91]).

Equational problems over finite universes can be used to encode queries over *relational databases*. Finally, note that also the *constraint satisfaction problems* on Boolean and finite domains treated in Section 1 can be easily encoded as equational problems over finite universes. Let $D = \{a_1, \ldots, a_K\}$ be a finite domain and let $C = \{\langle f_1, (i_{11}, \ldots, i_{1k_1})\rangle, \ldots, \langle f_m, (i_{m1}, \ldots, i_{mk_m})\rangle\}$ denote a finite set of constraint applications over D, such that every f_j is a constraint of the form $f_j \colon D^{k_j} \longrightarrow \{0, 1\}$. Moreover, let $\vec{x}$ denote the set of all variables $x_{i_{\alpha\beta}}$ occurring in C. Then C is satisfiable, if and only if the equational problem

$$P \quad \equiv \quad (\exists \vec{x}) \quad \left[\bigwedge_{j=1}^{m} \bigvee_{(b_1,\ldots,b_{k_j}) \in (f_j)^{-1}(1)} (x_{i_{j1}} = b_1 \wedge \cdots \wedge x_{i_{jk_j}} = b_{k_j}) \right]$$

over the Herbrand universe $H = \{a_1, \ldots, a_K\}$ is satisfiable. A good overview of the wide range of applications of equational problems can be found in [CL89].

In many of these applications, testing the satisfiability of an equational problem is even more important than actually computing the solutions. In this section, we present a survey of complexity results for this satisfiability problem, where we consider several restrictions on the equational

problems, namely: quantifier prefix $\exists^*$ versus $\exists^*\forall^*$, CNF versus DNF and, finally, interpretation of the formula over a finite universe versus an infinite universe.

2.1 Syntax and semantics of equational constraint problems

Equational problems are first-order formulas of the form $\exists\vec{w}\,\forall\vec{y}\,P(\vec{w},\vec{x},\vec{y})$, such that $P(\vec{w},\vec{x},\vec{y})$ is a quantifier-free formula with variables in $\vec{w}$, $\vec{x}$ and $\vec{y}$, where syntactic equality "=" is the only predicate symbol. A disequation $s \neq t$ is a short-hand notation for a negated equation $\neg(s = t)$. The trivially true problem is denoted by $\top$ and the trivially false one by $\bot$.

In this section, every equational problem P is considered over some fixed Herbrand universe H (or, equivalently, over some fixed finite signature Σ consisting of constant symbols and possibly function symbols). An interpretation over H is given through an H-ground substitution σ, whose domain coincides with the free variables of the equational problem. The trivial problem $\top$ evaluates to "true" in every interpretation. Likewise, $\bot$ always evaluates to "false". A single equation $s = t$ is validated by a ground substitution σ, if and only if $s\sigma$ and $t\sigma$ are syntactically identical. The connectives $\wedge$, $\vee$, $\neg$, $\exists$ and $\forall$ are interpreted as usual. A ground substitution σ which validates a problem P is called a *solution* of P. An equational problem is *satisfiable*, if and only if it has at least one solution.

As far as the satisfiability of an equational problem is concerned, there is no difference between free variables and existentially quantified ones. In particular, $\exists\vec{w}\,\forall\vec{y}\,P(\vec{w},\,\vec{x},\vec{y})$ is satisfiable, if and only if $\exists\vec{x}\,\exists\vec{w}\,\forall\vec{y}\,P(\vec{w},\,\vec{x},\vec{y})$ is. Without loss of generality we therefore only consider equational problems without free variables. In analogy with [CL89], universally quantified variables will be referred to as *parameters*.

In order to distinguish between syntactical identity and the equivalence of two equational problems, we use the notation "$\equiv$" and "$\approx$", respectively. We shall thus write $P \equiv Q$ to denote that the two equational problems P and Q are *syntactically identical*. If the equational problems P and Q have the same set of solutions, then they are *semantically equivalent*. In this case, we write $P \approx Q$.

Term tuples are used as a short-hand notation for a conjunction of equations or a disjunction of disequations, respectively. For term tuples $\vec{s} = (s_1, \ldots, s_k)$ and $\vec{t} = (t_1, \ldots, t_k)$, we shall abbreviate "$s_1 = t_1 \wedge \cdots \wedge s_k = t_k$" and "$s_1 \neq t_1 \vee \cdots \vee s_k \neq t_k$" to "$\vec{s} = \vec{t}$" and "$\vec{s} \neq \vec{t}$", respectively.

Example 2.1 Let $P \equiv (x_1 = a \wedge x_1 \neq x_2) \vee (x_2 \neq x_3 \wedge x_2 = b)$ be an equational problem over $H = \{a, b, c\}$. Then the following substitutions σ and τ are (examples of) solutions of P:
$\sigma = \{x_1 \leftarrow a, x_2 \leftarrow b, x_3 \leftarrow c\}$, $\tau = \{x_1 \leftarrow b, x_2 \leftarrow b, x_3 \leftarrow c\}$.

Let $Q \equiv (\exists y)(\forall z)[y = f(x) \wedge x \neq f(z)]$ be an equational problem over $H = \{a, f(a), f(f(a)), \ldots\}$. The only solution of Q is $\sigma = \{x \leftarrow a\}$.

Now suppose that Q is interpreted over the universe $H = \{a, f(a), g(a), f(f(a)), f(g(a)), \ldots\}$ with signature $\Sigma = \{a, f, g\}$. Then Q has many more solutions, e.g.: $\tau_1 = \{x \leftarrow g(a)\}$, $\tau_2 = \{x \leftarrow g(f(a))\}$, etc.

Unification problems are equational problems without universal quantifiers and without negation. It is well known that the set S of solutions of a conjunction $P \equiv s_1 = t_1 \wedge \ldots \wedge s_n = t_n$ can be represented by a single substitution μ, which is called the *mgu* (= most general unifier) of P. For every solution σ of P, there exists a substitution η, such that σ is the composition of μ and η (which we denote by $\sigma = \mu\eta$). Recall that the *mgu* is unique up to variable renaming. Moreover, it can be decided efficiently whether the *mgu* exists (or, equivalently, whether P is satisfiable). Likewise, the actual computation of the *mgu* can be done efficiently (cf. [BS94, BS01, MM82]).

2.2 Transformation rules of Comon and Lescanne

In [CL89], a rule system is provided which terminates on every equational problem and which transforms the original problem into an equivalent one in the so-called "definition with constraints form", which is basically a purely existentially quantified equational formula in DNF. Below some of the rules of [CL89] are recalled, namely the replacement rules R_1, R_2, the cleaning rule CR_2, the universality of parameter rules U_2, U_4, U_5 and the explosion rule E. Note that many more rules of [CL89] (like the decomposition rule, the clash rule, the occur check, etc.), which are not mentioned explicitly here, are "hidden" in the unification steps.

(R_1) $z = t \wedge P \quad \rightarrow \quad z = t \wedge P(z \leftarrow t)$

(R_2) $z \neq t \vee P \quad \rightarrow \quad z \neq t \vee P(z \leftarrow t)$

(CR_2) $\exists(\vec{w}, w)(w = t \wedge P) \quad \rightarrow \quad (\exists \vec{w})P$
if $w \notin Var(P)$ and $w \notin Var(t)$

(U_2) $(\forall \vec{y})[P \wedge (y \neq t \vee R)] \quad \rightarrow \quad (\forall \vec{y})[P \wedge R(y \leftarrow t)]$
if $y \in \vec{y}$ and $y \notin Var(t)$.

(E) $(\forall \vec{y})P \quad \rightarrow \quad \bigvee_{f \in \Sigma} (\exists \vec{w})(\forall \vec{y})[P \wedge s = f(w_1, \ldots, w_{\alpha(f)})]$
if the following conditions hold:
1. Each f is a (constant or function) symbol from the signature Σ with arity $\alpha(f) \geq 0$,
2. the w_i's are fresh, pairwise distinct variables.

The following rule is only correct in case of an infinite universe :

(U_4) $(\forall \vec{y})[P \wedge (z_1 = u_1 \vee \cdots \vee z_n = u_n \vee R)] \quad \rightarrow \quad (\forall \vec{y})[P \wedge R]$
if the following conditions hold:
1. Every z_i is a variable syntactically different from u_i,
2. every equation $z_i = u_i$ contains at least one parameter from $\vec{y}$,
3. R contains no parameter from $\vec{y}$.

The following rule can only be applied in case of a finite universe.

(U_5) $(\forall \vec{y})[P \wedge Q] \quad \rightarrow \quad (\forall \vec{y})[P \wedge Q(y \leftarrow a_1) \wedge \cdots \wedge Q(y \leftarrow a_K)]$
if the universe H is of the form $H = \{a_1, \ldots, a_K\}$.

Rule system 2.1: Comon and Lescanne

The correctness of the rule R_1 is obvious. The rule R_2 follows from the equivalence $[A \vee B] \approx [(A \wedge \neg B) \vee B]$, which holds for any logical formulas A and B. The correctness of the U_2-rule essentially follows from the R_2-rule and the unsatisfiability of the disequation $(\forall \vec{y})(y \neq t)$ over any nontrivial universe.

The explosion rule E (and, analogously, the U_5-rule) is sometimes also referred to as the *domain closure axiom*. Its idea is the following: Let H be the Herbrand universe of terms over some finite signature Σ. Then every ground term $t \in H$ has one of the symbols in Σ as its leading symbol. Hence the formula $\bigvee_{f \in \Sigma} (\exists \vec{w})[s = f(w_1, \ldots, w_{\alpha(f)})]$ is clearly valid for any term s.

Finally, the rule U_4 is mainly due to the so-called **independence of inequations** of [Col84], that can be stated in the following way.

Every purely existentially quantified conjunction of disequations over an infinite universe

has at least one solution, if and only if each of the conjuncts has a solution.

Moreover, the latter condition is always fulfilled unless one of the conjuncts is a trivial disequation of the form $t \neq t$. Then the correctness of the U_4-rule follows from the fact that the subformula $(z_1 = u_1 \vee \cdots \vee z_n = u_n)$ cannot be true for all values of the variables in $\vec{y}$.

2.3 Equational problems with $\exists^*$-prefix

In this section we recall several complexity results on equational problems with purely existential quantifier prefix. For CNF, the complexity in case of an infinite universe is the same as in case of a finite universe, even though the proof of the upper bound on the complexity will differ significantly. In contrast, for DNF, we definitely have different complexity classifications. Surprisingly enough, the case of a finite universe will turn out to have a higher complexity than an infinite universe (provided that P $\neq$ NP holds).

Theorem 2.2 [$\exists^*$-CNF over a nontrivial H] *The satisfiability problem of equational problems in $\exists^*$-CNF over an arbitrary (finite or infinite) Herbrand universe H with $|H| \geq 2$ is* NP*-complete.*

Proof: The NP-hardness can be shown by the obvious reduction from the 3-SAT-problem. Let $E = (l_{11} \vee l_{12} \vee l_{13}) \wedge \cdots \wedge (l_{n1} \vee l_{n2} \vee l_{n3})$ be a Boolean formula, such that the l_{ij}'s are propositional literals over the propositional variables in $P = \{p_1, \ldots, p_k\}$. Moreover, let $a \in H$ be an arbitrary constant in H. Then we define the equational problem

$$P \quad \equiv \quad \exists \vec{x}\, [(l'_{11} \vee l'_{12} \vee l'_{13}) \wedge \cdots \wedge (l'_{n1} \vee l'_{n2} \vee l'_{n3})]$$

in $\exists^*$-CNF with $\vec{x} = (x_1, \ldots, x_k)$, such that the literals l'_{ij} in P are either of the form $x_\gamma = a$ (if l_{ij} is a positive literal p_γ) or of the form $x_\gamma \neq a$ (if l_{ij} is a negative literal $\neg p_\gamma$). It is easy to check that E is satisfiable, if and only if P is satisfiable.

The NP-membership in case of a finite universe is easy. In an NP-algorithm for deciding the satisfiability of an equational problem $P \equiv \exists \vec{x}\, P'$, we first guess a ground substitution σ with domain $\vec{x}$ and then check that $P'\sigma$ evaluates to true.

Note that this NP-algorithm does not work in case of an infinite universe, unless we can find a polynomial bound on the size of the terms in the range of σ. However, the NP-membership in case of an infinite universe can be shown via a different NP-algorithm. Let

$$P \quad \equiv \quad \exists \vec{x}\, [(l_{11} \vee \cdots \vee l_{1k_1}) \wedge \cdots \wedge (l_{n1} \vee \cdots \vee l_{nk_n})]$$

be an equational problem in $\exists^*$-CNF, where the l_{ij}'s are equations or disequations. We can decide the satisfiability of P by first guessing for every $i \in \{1, \ldots, n\}$ a literal (an equation or a disequation) l_{ij_i} from the i-th clause. Then we check *in polynomial time* that the resulting existentially quantified conjunction of equations and disequations is satisfiable. In Lemma 2.3 below, we show that the latter check can indeed be done in polynomial time. □

Lemma 2.3 [parameter-free conjunctions] *Let*

$$P \quad \equiv \quad (\exists \vec{x})\, [e_1 \wedge \cdots \wedge e_k \wedge d_1 \wedge \cdots \wedge d_l]$$

be a conjunction of equations e_i and disequations d_j over some infinite universe H. Then the satisfiability of P can be tested as follows.

Case 1: *If $e_1 \wedge \cdots \wedge e_k$ is unsatisfiable, then P is also unsatisfiable.*

Case 2: *Let $e_1 \wedge \cdots \wedge e_k$ be satisfiable with mgu ϑ. Then P is satisfiable, if and only if $d_1\vartheta \wedge \cdots \wedge d_l\vartheta$ contains no trivial disequation of the form $t \neq t$.*

Proof: Case 1 is trivial. For Case 2, let $\vartheta = \{x_{i_1} \leftarrow s_1, \ldots, x_{i_\alpha} \leftarrow s_\alpha\}$ denote the *mgu* of the equations $e_1 \wedge \cdots \wedge e_k$. Note that the variables x_{i_j} are pairwise distinct and do not occur in the range of ϑ.

By the definition of the *mgu*, the conjunctions $e_1 \wedge \cdots \wedge e_k$ and $x_{i_1} = s_1 \wedge \cdots \wedge x_{i_\alpha} = s_\alpha$ are equivalent. Moreover, by multiple applications of the R_2-rule of [CL89] (see also Section 2.2), ϑ may be applied to the disequations. Thus $P \approx (\exists \vec{x})[x_{i_1} = s_1 \wedge \cdots \wedge x_{i_\alpha} = s_\alpha \wedge d_1\vartheta \wedge \cdots \wedge d_l\vartheta]$ holds. But then, since all variables $x_{i_1}, \ldots, x_{i_\alpha}$ occur only once, the equations may be eliminated by the CR_2-rule of [CL89]. We thus have $P \approx P' \equiv (\exists \vec{x})[d_1\vartheta \wedge \cdots \wedge d_l\vartheta]$. By the independence of inequations recalled in Section 2.2, any conjunction of nontrivial disequations over an infinite universe has at least one solution. Therefore, P' (and, hence, also P) is indeed satisfiable, if and only if P' contains no disequation of the form $t \neq t$. □

We now turn our attention to parameter-free equational problems in DNF. As has already been mentioned above, the cases of a finite universe and an infinite universe lead to different complexity results. The reason for this is the "independence of inequations" recalled in Section 2.2, which only holds in case of an infinite universe. This effect is illustrated by the following example.

Example 2.4 Let the equational problem P be defined as follows.

$$P \quad \equiv \quad (\exists y)\,(x_1 \neq y) \wedge (x_2 \neq y) \wedge (x_3 \neq y) \;\wedge (x_1 \neq x_2) \wedge (x_1 \neq x_3) \wedge (x_2 \neq x_3)$$

If P is interpreted over $H = \{a, b, c\}$, then P is unsatisfiable.

On the other hand, over the infinite universe $H = \{a, f(a), f^2(a), \ldots\}$, the problem P is satisfiable, where $\sigma = \{x_1 \leftarrow a, x_2 \leftarrow f(a), x_2 \leftarrow f^2(a), \ldots\}$ is a solution.

We thus get the following complexity results for equational problems in $\exists^*$-DNF.

Theorem 2.5 [$\exists^*$-DNF over an infinite H] *The satisfiability problem of equational problems in $\exists^*$-DNF over an infinite H is in* P.

Proof: Let $P \equiv (\exists \vec{x})P_1 \vee \cdots \vee P_n$ with $P_i \equiv (e_{i1} \wedge \cdots \wedge e_{ik_i} \wedge d_{i1} \wedge \cdots \wedge d_{il_i})$, such that the e_{ij}'s are equations and the d_{ij}'s are disequations. Then P is satisfiable, if and only if at least one disjunct $(\exists \vec{x})P_i$ is satisfiable. Moreover, by Lemma 2.3, $(\exists \vec{x})P_i$ is satisfiable, if and only if $\mu = mgu(e_{i1} \wedge \cdots \wedge e_{ik_i})$ exists and $(d_{i1} \wedge \cdots \wedge d_{il_i})\mu$ contains no trivial disequation of the form $t \neq t$. □

Theorem 2.6 [$\exists^*$-DNF over a finite H with $|H| \geq 3$] *The satisfiability problem for equational problems in $\exists^*$-DNF over a finite H with $|H| \geq 3$ is* NP*-complete.*

Proof: The NP-hardness is shown by a reduction from the well-known NP-complete problem K-colorability with $K \geq 3$: Let $G = (V, E)$ be a graph with vertices $V = \{v_1, \ldots, v_n\}$ and edges E. Then G is K-colorable (there exists a function $f\colon V \to \{1, \ldots, K\}$, such that $f(v_i) \neq f(v_j)$ holds for every edge $\{v_i, v_j\} \in E$), if and only if the equational problem

$$(\exists \vec{v}) \bigwedge_{\{v_i, v_j\} \in E} v_i \neq v_j$$

	finite universe H		infinite universe
	$\|\mathbf{H}\| = \mathbf{2}$	$\|\mathbf{H}\| \geq \mathbf{3}$	
DNF	in P	NP-complete	in P
CNF	NP-complete	NP-complete	NP-complete

Figure 1: $\exists^*$-formulas

over $H = \{a_1, \ldots, a_K\}$ with $\vec{v} = (v_1, \ldots, v_n)$ is satisfiable. □

Recall from [GJ79] that the K-colorability problem is NP-complete for any $K \geq 3$, whereas it is in P for $K = 2$. Consequently, the reduction in the above proof does not work for $K = 2$. In fact, it is straightforward to show that the satisfiability problem for equational problems in $\exists^*$-DNF over H with $|H| = 2$ is in P. Figure 1 summarizes the results for $\exists^*$-formulas.

2.4 Equational problems with $\exists^* \forall^*$-prefix

Now we consider equational problems with $\exists^* \forall^*$-prefix. It will turn out that this alternation of quantifiers pushes the complexity one level higher in the polynomial hierarchy. Moreover, the rôles of CNF and DNF are changed with respect to the case of a purely existential quantifier prefix, since now the innermost quantifier is $\forall$. Thus, for DNF, we get the following complexity classification.

Theorem 2.7 [$\exists^* \forall^*$-DNF over a nontrivial H] *The satisfiability problem of equational problems in $\exists^* \forall^*$-DNF over an arbitrary (finite or infinite) Herbrand universe H with $|H| \geq 2$ is Σ_2P-hard. Moreover, if H is finite, then this problem is Σ_2P-complete.*

Proof: The Σ_2P-hardness is proven via a reduction from the well-known Σ_2P-complete 3-QSAT_2 problem (cf. [Sto76]). This proof follows exactly the same pattern as the NP-hardness proof in Theorem 2.2. Let an instance of the 3-QSAT_2 problem be given through two disjoint sets $P = \{p_1, \ldots, p_k\}$ and $R = \{r_1, \ldots, r_l\}$ of propositional variables and the Boolean formula

$$E \quad = \quad (l_{11} \wedge l_{12} \wedge l_{13}) \vee \cdots \vee (l_{n1} \wedge l_{n2} \wedge l_{n3})$$

such that the l_{ij}'s are literals over the propositional variables in $P \cup R$. Moreover, let a be an arbitrary constant in H. Then we define the equational problem $P \equiv \exists \vec{x} \forall \vec{y} [C_1 \wedge \cdots \wedge C_n]$ over H in such a way that every literal of the form p_γ or $\neg p_\gamma$ in E is encoded by the literal $x_\gamma = a$ or $x_\gamma \neq a$, respectively, in P. Likewise, $y_\beta = a$ and $y_\beta \neq a$ are used to encode literals of the form r_β or $\neg r_\beta$, respectively. Again, this reduction can be clearly done in polynomial time and its correctness is trivial.

To prove the Σ_2P-membership in case of a finite universe H is easy. Guess values for the existentially quantified variables and check the satisfiability of the resulting formula by means of an NP-oracle. □

It is not clear, how the Σ_2P-algorithm from the above proof should be extended to the case of an infinite universe. In particular, we do not know if there exists a polynomial bound on the terms that have to be guessed for the existentially quantified variables. In Theorem 2.9, we shall show that the satisfiability problem of equational problems in $\exists^* \forall^*$-CNF over an *infinite* universe is in

NP. Hence, the obvious upper bound on the complexity of equational problems in $\exists^* \forall^*$-DNF over an infinite universe is NEXPTIME, since we can of course first transform the DNF into CNF via the distributivity of $\wedge$ and $\vee$ (in general, at the expense of an exponential blow-up) and then apply the NP-algorithm sketched in Theorem 2.9. The exact complexity of $\exists^* \forall^*$-DNF over an infinite universe is an open problem.

Theorem 2.8 [$\exists^* \forall^*$-CNF over a finite H with $|H| \geq 3$] *The satisfiability problem for equational problems in $\exists^* \forall^*$-CNF over a finite H with $|H| \geq 3$ is Σ_2P-complete.*

Proof: (Sketch) The Σ_2P-membership can be shown in exactly the same way as in Theorem 2.7. Guess values for the existentially quantified variables and check the satisfiability of the resulting formula by means of an NP-oracle.

The proof of the Σ_2P-hardness is quite involved. It goes by a reduction from some kind of "parameterized K-colorability problem". For details, see [Pic00] and [Pic01]. □

Analogously to equational problems in $\exists^*$-DNF over a finite universe, the complexity of $\exists^* \forall^*$-CNF over a finite H becomes one level lower in the polynomial hierarchy, if H has only two elements. In fact, in [Pic01], it is shown that the satisfiability problem for equational problems in $\exists^* \forall^*$-CNF over a H with $|H| = 2$ is NP-complete.

Now it only remains to consider the case of $\exists^* \forall^*$-CNF over an infinite universe. Analogously to the $\exists^*$-DNF, this satisfiability problem is one level lower in the polynomial hierarchy than for a finite universe.

Theorem 2.9 [$\exists^* \forall^*$-CNF over an infinite universe] *The satisfiability problem of equational problems in $\exists^* \forall^*$-CNF over an infinite H is* NP*-complete.*

Proof: (Sketch) The NP-hardness is clear, since even the case of $\exists^*$-CNF is NP-hard. As for the NP-membership, we give a (very rough) sketch of an NP-algorithm, which works as follows (for details, refer to [Pic99] and [Pic01]).

(1) *Elimination of the parameters from the equations and simplification of the disequations*: Let P be an arbitrary equational problem in $\exists^* \forall^*$-CNF. Then P can be transformed in *polynomial time* into the following form

$$P' \quad \equiv \quad (\exists \vec{x}\,)(\forall \vec{y}\,)[E_1 \vee (\vec{x} \neq \vec{t}_1)] \wedge \cdots \wedge [E_n \vee (\vec{x} \neq \vec{t}_n)],$$

where the E_i's are parameter-free disjunctions of equations and the $\vec{t}_i$'s are term tuples with variables only in $\vec{y}$.

(2) *Elimination of the parameters from the disequations*: Of course, the universal quantifiers can be shifted in front of the disequations, since the subformulas E_i no longer contain any universally quantified variables. Moreover, a universally quantified disequation of the form $(\forall \vec{y}\,)(\vec{x} \neq \vec{t}_i)$ can be transformed by successive applications of the explosion rule recalled in Section 2.2 into a purely existentially quantified disjunction of the form $(\exists \vec{w}\,)[d_{i1} \vee \cdots \vee d_{in_i}]$, where each d_{ij} is either an equation or a conjunction of an equation and a disequation. Note that the number of such disjuncts d_{ij} is basically $k \times p$, where $k = |\Sigma|$ denotes the number of symbols in the signature Σ of H and p denotes the number of positions in the term tuple $\vec{t}_i$.

(3) *Guess and check*: After the above two transformation steps, we (almost) have an equational problem in $\exists^* \forall^*$-CNF. Hence, in principle, we can proceed as in the proof of Theorem 2.2 via

	finite universe H		infinite universe
	$\|\mathbf{H}\| = \mathbf{2}$	$\|\mathbf{H}\| \geq \mathbf{3}$	
CNF	NP-complete	Σ_2P-complete	NP-complete
DNF	Σ_2P-complete	Σ_2P-complete	Σ_2P-hard, in NEXPTIME

Figure 2: $\exists^* \forall^*$-formulas

Lemma 2.3. However, there is a subtle problem with this. As has already been mentioned above, the size of the disjunctions obtained in Step (2) of this algorithm is linear with respect to the number of positions of the term tuples $\vec{t}_i$. Moreover, by the unification steps performed in Step (1), this number of positions may become exponential, even though their representation as directed acyclic graphs is of course polynomially bounded. Hence, the transformation in Step (2) must not be carried out explicitly. Instead, the guess of a certain disjunct d_{ij} has to be done directly by inspecting the term tuple $\vec{t}_i$. □

Figure 2 summarizes the results for $\exists^* \forall^*$-formulas.

2.5 Equational formulas with arbitrary quantifier prefix

Algorithms for deciding the satisfiability of arbitrary equational formulas (in particular, where the quantifier prefix is not restricted to the form $\exists^* \forall^*$), usually work by quantifier elimination (cf. [CL89, Mah88]). To this end, the transformation of an equational problem with $\exists^* \forall^*$-prefix into an equivalent one with $\exists^*$-prefix is applied to equational formulas with arbitrary quantifier prefix in order to reduce the number of quantifier alternations. Let P be an equational formula of the form $P \equiv (\exists \vec{x}_1)(\forall \vec{x}_2) \cdots (\exists \vec{x}_{n-1})(\forall \vec{x}_n)Q$. Moreover, suppose that we can effectively compute a purely existentially quantified formula $(\exists \vec{y})R$ that is equivalent to $(\exists \vec{x}_{n-1})(\forall \vec{x}_n)Q$. Then P is equivalent to $P' \equiv (\exists \vec{x}_1)(\forall \vec{x}_2) \cdots (\forall \vec{x}_{n-2})(\exists \vec{y})R$.

Likewise, if P is of the form $P \equiv (\exists \vec{x}_1)(\forall \vec{x}_2) \cdots (\forall \vec{x}_{n-1})(\exists \vec{x}_n)Q$, then we clearly have the following chain of equivalences.

$$\begin{aligned} P &\equiv (\exists \vec{x}_1)(\forall \vec{x}_2) \cdots (\forall \vec{x}_{n-1})(\exists \vec{x}_n)Q \\ &\approx (\exists \vec{x}_1)(\forall \vec{x}_2) \cdots \neg\neg(\forall \vec{x}_{n-1})(\exists \vec{x}_n)Q \\ &\approx (\exists \vec{x}_1)(\forall \vec{x}_2) \cdots \neg(\exists \vec{x}_{n-1})(\forall \vec{x}_n)(\neg Q). \end{aligned}$$

Hence, also in this case, the innermost quantifiers have been brought into the form $\exists^* \forall^*$ and we can transform $(\exists \vec{x}_{n-1})(\forall \vec{x}_n)(\neg Q)$ into an equivalent formula of the form $(\exists \vec{y})R$. We thus get the equational formula

$$P' \equiv (\exists \vec{x}_1)(\forall \vec{x}_2) \cdots (\exists \vec{x}_{n-2})\neg(\exists \vec{y})R \approx (\exists \vec{x}_1)(\forall \vec{x}_2) \cdots (\exists \vec{x}_{n-2})(\forall \vec{y})\neg R,$$

which is equivalent to P.

In other words, it suffices to provide a transformation of equational formulas from $\exists^* \forall^*$-form into $\exists^*$-form, in order to solve the satisfiability problem for arbitrary equational formulas. Unfortunately, this quantifier elimination step has exponential cost. However, by the high inherent complexity

of arbitrary equational formulas, this can hardly be helped. Recall from Theorem 1.13, that the $\mathsf{QSAT}(\mathcal{F})$ problem is PSPACE-complete, unless the constraints under consideration are subjected to some severe restrictions. Analogously, it can be shown that the satisfiability problem of equational formulas over a finite universe is PSPACE-complete, if no restrictions are imposed on the quantifier prefix (cf. [Kun87]). In case of an infinite universe, the satisfiability problem of equational formulas with arbitrary quantifier prefix is even non-elementary recursive (cf. [Vor96]).

2.6 Open problems and future research

In Section 2, we have given a survey of complexity results for the satisfiability problem of equational problems. In almost all of the cases thus considered, there is an exact classification of the complexity. Only in case of equational problems in DNF with $\exists^* \forall^*$-prefix over an infinite universe, there is a gap between the Σ_2P lower bound and the NEXPTIME upper bound (cf. Figure 2). Closing this gap is an interesting open problem for future research in this area.

Recall that we have only considered the case where all terms (and, in particular, all variables) in an equational problem are interpreted over the same universe. An extension of these results to the case of many sorts has not been done explicitly yet. Actually, it seems as though this extension is not too difficult. After all, it has turned out that we only have to be careful whether a universe is finite or infinite. Nevertheless, the details of such an extension to many sorts have to be worked out yet. Moreover, little research efforts have been made so far, in order to investigate the complexity, when restrictions different from the ones considered here are imposed, e.g.: what happens, when the number of variables is restricted rather than the quantifier prefix, etc.

As usual, the complexity analysis of a given problem is not the end of the story. In general, one will try to apply the theoretical insight into the inherent complexity of a problem to the construction of new and more efficient algorithms. A major lesson to be learned from the complexity results recalled here is that — in contrast to the algorithm of [CL89] — one should not try to treat the cases of a finite universe and of an infinite universe, respectively, in a uniform way. Actually, the NP-membership proof sketched in Theorem 2.9 can be considered as an improvement of previous algorithms in case of an infinite universe. Searching for further improvements is an important goal for future research.

In this survey, we have concentrated on equational formulae with $\exists^* \forall^*$-prefix. Moreover, in case of equational problems with $\exists^* \forall^*$-prefix, we were unable to present a better approach than the transformation into CNF followed by the NP-algorithm from the proof sketch of Theorem 2.9. As far as the worst case complexity is concerned, this is okay. However, practical experience shows that such a preprocessing step of shifting the quantifiers to the front and transforming the formula into CNF are very costly and sometimes not really necessary. Consequently, in [CD94], an algorithm is presented which neither requires a CNF nor a specific quantifier prefix. Instead, the expensive distributivity rules are only applied, when there is no alternative. Moreover, a whole collection of rules dealing with single quantifiers and combinations of quantifiers are provided. Of course, by the high inherent complexity of equational formulae with no restriction on the quantifier occurrences (cf. Section 2.5) there is a clear limit up to which the worst case complexity can possibly be improved. Nevertheless, a combination of the ideas of [CD94] with the cheap transformations needed for the NP-membership result in Theorem 2.9 may serve as a good starting point for searching for further improvements.

3 Complexity of Equational Matching and Unification Constraint Problems

Matching and unification in equational theories are the keystones of automated deduction. They are used extensively in several areas of computer science, including theorem proving, database systems, natural language processing, logic programming, computer algebra, and program verification. Plotkin [Plo72] was the first to formulate explicitly the idea that theorem provers should have built-in algorithms for matching and unification in equational theories. His pioneering article provided the impetus for the development of the entire field of equational matching and unification.

We briefly introduce the basic notions for equational matching and unification. Additional material can be found in [BS01] or [DJ90].

A *signature* $\mathcal{F}$ is a set of function symbols of designated arities. If $\mathcal{F}$ is a signature and $\mathcal{X}$ is a countable set of variables, then $\mathcal{T}(\mathcal{F}, \mathcal{X})$ denotes the set of all terms over the signature $\mathcal{F}$ and the variables in $\mathcal{X}$. We also write $Var(t)$ for the set of variables occurring in a term t. The *size* of term t is its length $|t|$ as a string. As usual, a *ground term* is a term without variables. A *substitution* is a mapping $\rho\colon \mathcal{X} \longrightarrow \mathcal{T}(\mathcal{F}, \mathcal{X})$ such that $x\rho = x$ for all but finitely many variables x. Consequently, a substitution ρ can be identified with its restriction to the finite set $dom(\rho) = \{x \in \mathcal{X} \mid x\rho \neq x\}$, which is called the *domain* of ρ. A substitution ρ is *ground* if $x\rho$ is a ground term for all $x \in dom(\rho)$.

An *equation* is a pair of terms $l = r$. Each equation is viewed as an *equational axiom*, namely as the first-order sentence $(\forall x_1) \ldots (\forall x_m)(l = r)$ obtained from the equation by universal quantification over all variables occurring in the terms l and r. If E is a set of equational axioms, then the *equational theory* Th(E) presented by E is the smallest congruence relation over $\mathcal{T}(\mathcal{F}, \mathcal{X})$ containing E and closed under substitutions, i.e., Th(E) is the smallest congruence containing all pairs $l\rho = r\rho$, where $l = r$ is in E and ρ is a substitution. By an abuse of terminology, we will often say "the equational theory E" instead of the correct "the equational theory Th(E) presented by E". We write $s =_{\mathrm{E}} t$ to denote that the pair (s, t) of terms is a member of Th(E).

E-equality on terms can be extended to substitutions by setting $\rho =_{\mathrm{E}} \sigma$ if and only if for all variables $x \in \mathcal{X}$ we have that $x\rho =_{\mathrm{E}} x\sigma$. If V is a set of variables and ρ, σ are substitutions, we put $\rho =^V_{\mathrm{E}} \sigma$ if and only if $(\forall x \in V)(x\rho =_{\mathrm{E}} x\sigma)$ holds. We also consider the preorder $\leq^V_{\mathrm{E}}$ on substitutions defined by the condition $\sigma \leq^V_{\mathrm{E}} \rho$ if and only if $(\exists \eta)(\sigma\eta =^V_{\mathrm{E}} \rho)$ holds. In turn, this preorder gives rise to the following equivalence relation $\equiv^V_{\mathrm{E}}$ on substitutions:

$$\rho \equiv^V_{\mathrm{E}} \sigma \quad \Longleftrightarrow \quad \rho \leq^V_{\mathrm{E}} \sigma \quad \text{and} \quad \sigma \leq^V_{\mathrm{E}} \rho.$$

In general, $\rho \leq^V_{\mathrm{E}} \sigma$ does not imply that $\rho \equiv^V_{\mathrm{E}} \sigma$; similarly, $\rho \equiv^V_{\mathrm{E}} \sigma$ does not imply that $\rho =^V_{\mathrm{E}} \sigma$. Nevertheless, these three relations coincide on ground substitutions with the same domain.

In the sequel, we will be concerned with equational theories presented by finite sets E whose axioms are among those depicted in Figure 3. Moreover, AC(linear) is the restriction of the equational theory AC applied to linear terms and *Set* is a special case of the theory ACI in which there is only one ACI-symbol that occurs on the top of the considered terms.

3.1 Complexity of equational matching

Let s be a term, called *pattern*, and let t be a ground term, called *subject*. An E-*matcher* of s and t is a substitution ρ such that $s\rho =_{\mathrm{E}} t$. Whenever such an E-matcher exists, we say that the pattern s E-*matches* the subject t.

A *complete set of* E-*matchers* of s and t is a set S of substitutions such that the following conditions hold:

Associativity	$\mathrm{A}(f)$	$f(f(x,y),z) = f(x,f(y,z))$
Commutativity	$\mathrm{C}(f)$	$f(x,y) = f(y,x)$
Idempotence	$\mathrm{I}(f)$	$f(x,x) = x$
Nilpotence	$\mathrm{N}(f)$	$f(x,x) = 0$
Existence of Unit	$\mathrm{U}(f)$	$f(x,1) = x,\ f(1,x) = x$
Homomorphism	$\mathrm{H}(f,g,h)$	$f(g(x,y)) = h(f(x),f(y))$
Abelian group	$\mathrm{AG}(+,-,e)$	see Figure 4
Boolean ring	$\mathrm{BR}(\wedge,\oplus,0,1)$	see Figure 5

Figure 3: Equational axioms

$$\begin{array}{rcl} x+e & = & x \\ x+(-x) & = & e \end{array} \qquad \begin{array}{rcl} x+y & = & y+x \\ (x+y)+z & = & x+(y+z) \end{array}$$

Figure 4: Abelian group axioms AG

$$\begin{array}{rcl} x \oplus 0 & = & x \\ x \oplus x & = & 0 \\ x \wedge 0 & = & 0 \\ x \wedge 1 & = & x \\ x \wedge x & = & x \end{array} \qquad \begin{array}{rcl} x \oplus y & = & y \oplus x \\ (x \oplus y) \oplus z & = & x \oplus (y \oplus z) \\ x \wedge y & = & y \wedge x \\ (x \wedge y) \wedge z & = & x \wedge (y \wedge z) \\ x \wedge (y \oplus z) & = & (x \wedge y) \oplus (x \wedge z) \end{array}$$

Figure 5: Boolean ring axioms BR

1. Each substitution $\rho \in S$ is an E-matcher of s and t, and, moreover, $dom(\rho) \subseteq V$, where $V = Var(s)$ is the set of variables of s;

2. For every E-matcher σ of s and t, there is a substitution $\rho \in S$ such that $\rho \leq_{\mathrm{E}}^{V} \sigma$.

We say that S is a *minimal complete set of* E*-matchers* of s and t if, in addition, every two distinct members of S are $\leq_{\mathrm{E}}^{V}$-incomparable (this means that for all substitutions $\sigma, \rho \in S$ the condition $\sigma \leq_{\mathrm{E}}^{V} \rho$ implies $\sigma = \rho$).

In general, it may be the case that s E-matches t, but there is no minimal complete set of E-matchers of s and t. On the other hand, it is well known that if a minimal complete set of E-unifiers of s and t exists, then it is unique up to $\equiv_{\mathrm{E}}^{V}$.

From now on, we assume that E is a set of equational axioms such that if s E-matches t, then there exists a minimal complete set of E-matchers of s and t. We let $\mu\mathrm{CSM}_{\mathrm{E}}(s,t)$ denote the (unique up to $\equiv_{\mathrm{E}}^{V}$) minimal complete set of E-matchers of s and t, if s E-matches t, or the empty set, otherwise.

E-matching is said to be *unitary* if for every pattern s and every subject t we have that $|\mu\mathrm{CSM}_{\mathrm{E}}(s,t)| \leq 1$. Similarly, E-matching is said to be *finitary* if for every pattern s and every subject t the set $\mu\mathrm{CSM}_{\mathrm{E}}(s,t)$ is finite.

If E is a set of equational axioms then we associate with E the following E-matching decision problem.

E-MATCHING
Input: A pattern s, a subject t, and an equational theory E.
Question: Can s be E-matched with t, i.e., is there a substitution ρ, such that $s\rho =_{\mathrm{E}} t$?

By examining the signature $\mathcal{F}$ over which the terms of matching problems in the theory Th(E) have been built, we distinguish between two different kinds of E-matching. Let $sig(\mathrm{E})$ be the set of all function and constant symbols occurring in the equational axioms of E. If the signature $\mathcal{F}$ contains $sig(\mathrm{E})$ and free constant symbols, but no free function symbols, then we speak about *elementary* E*-matching*. If the signature $\mathcal{F}$ contains free function symbols of arbitrary arities, then we speak about *general* E*-matching*.

Most of the equational matching decision problems are NP-complete. A remarkable exception is the case of AC(linear)-matching that is proved to be polynomial by means of graph matching techniques. These complexity results were proved by Benanav *et al.* [BKN87], Kapur and Narendran [KN92] Baader [Baa98], Klíma and Srba [KS00], Eker [Eke93], and others. Clearly, an NP-hardness result for the elementary case naturally extends to NP-hardness of the general problem. In the same spirit, a polynomial result for the general case extends to a polynomial-time decidable elementary problem.

The results on equational matching decision problems are summarized in Figure 6.

Assume that E is some finitary equational theory and $\mathcal{A}$ is an algorithm such that, given two terms s and t as input, the algorithm $\mathcal{A}$ returns a minimal complete set of E-matchers of s and t, if s and t can be E-matched, or the empty set, otherwise. As a byproduct of this algorithm and within the same complexity bounds, we can solve a related *counting problem*, namely we can compute the number of most general E-matchers of two given terms. In many respects, this counting problem is closer to the problem of computing a complete set of E-matchers than the decision problem for E-matching. If E is a set of equational axioms such that E-matching is finitary, then we associate with E the following counting problem.

theory	complexity	
	elementary	general
$\emptyset$	$\longleftarrow$	linear [PW78]
A	NP-complete [Ang80]	NP-complete [BKN87]
C		NP-complete [BKN87]
AC	NP-complete [Eke93, BS94]	NP-complete [BKN87]
AC(linear)	$\longleftarrow$	in P [BKN87]
AI	NP-complete [KS00]	NP-complete
AU	NP-complete (unpublished)	NP-complete
ACI		NP-complete [KN92]
ACU	NP-complete [HK99]	NP-complete [KN92]
I		NP-complete (unpublished)
U		NP-complete [TA87]
N	NP-complete [GNW00]	NP-complete
ACN	NP-complete [GNW00]	NP-complete
ACUN	in P [GNW00]	NP-complete [GNW00]
ACUNH	in P [GNW00]	NP-complete [GNW00]
Set		NP-complete [KN86]
AG	in P (Gaussian elimination in $\mathbb{Z}$)	NP-complete [Sch97]
BR	Π_2P-complete [Baa98]	PSPACE-complete [Baa98]

Figure 6: Complexity results for equational matching decision problems

#E-MATCHING
Input: A pattern s, a subject t, and an equational theory E.
Output: Cardinality of the minimal complete set of E-matchers $\mu\mathrm{CSM}_{\mathrm{E}}(s,t)$.

Counting problems for equational matching were considered by Hermann and Kolaitis in [HK95] and indirectly also in [HK00]. They showed that most of the NP-completeness results carry over to #P-completeness results for the counting versions for corresponding equational matching problems. However, the case of AC(linear)-matching becomes #P-complete for counting, whereas the corresponding decision problem is in P.

Theorem 3.1 ([HK95]) *The general* #E-*matching problems are* #P-*complete for the equational theories* A, C, AC, AC(linear), ACH, I, U, IU, ACI, Set, ACU *and* ACIU.

The *elementary* E-*matching problem* is the restriction of E-matching to signatures with no free function symbols. Thus, given a pair (s,t), where s is a term and t is a ground term with function symbols among those in the equational axioms of E, the question is to decide whether there is a substitution ρ such that $s\rho =_{\mathrm{E}} t$. The *elementary* #E-*matching problem* is the analogous restriction of #*E*-matching. The *simultaneous elementary* E-*matching problem* is the following decision problem.

SIMULTANEOUS ELEMENTARY E-MATCHING
Input: A finite set $S = \{(s_1,t_1),\ldots,(s_k,t_k)\}$, where each s_i is a term and each t_i is a ground term with function symbols among those in the equational axioms of E.
Question: Is there a substitution ρ such that $s_i\rho =_{\mathrm{E}} t_i$ for every $i \leq k$?

Such a substitution is called an E-*matcher of the set* $\{(s_1,t_1),\ldots,(s_k,t_k)\}$. Similarly, the *simultaneous elementary* #E-*matching problem* is the following counting problem.

SIMULTANEOUS ELEMENTARY #E-MATCHING
Input: A finite set $S = \{(s_1,t_1),\ldots,(s_k,t_k)\}$, where each s_i is a term and each t_i is a ground term with function symbols among those in the equational axioms of E.
Output: Cardinality of the minimal complete set of E-matchers $\mu\mathrm{CSM}_{\mathrm{E}}(S)$.

The notation $s_1 \doteq_{\mathrm{E}} t_1,\ldots,s_k \doteq_{\mathrm{E}} t_k$ will be used to represent an instance of the simultaneous elementary E-matching (or #E-matching) problem.

In effect, the simultaneous elementary E-matching problem asks for the solution of a *system* of equations $s_1 \doteq_{\mathrm{E}} t_1,\ldots,s_k \doteq_{\mathrm{E}} t_k$, where the function symbols of $\mathcal{F}$ are exactly the function symbols occurring in the equational axioms of E. Of course, one can consider simultaneous E-matching problems over arbitrary signatures. However, the simultaneous E-matching problem over arbitrary signatures is reducible to the E-matching problem, because one can use free function symbols to encode a system of equations into a single equation. Indeed, a system of equations $s_1 \doteq_{\mathrm{E}} t_1,\ldots,s_k \doteq_{\mathrm{E}} t_k$ can be written as a single equation $f(s_1,\ldots,s_k) \doteq_{\mathrm{E}} f(t_1,\ldots,t_k)$ with the help of a free function symbol f. We will classify simultaneous elementary matching problems using two parameters, namely the *number of equations* in a given system, called the *length* of the system, and the *number of free constants* in the signature. Note that the number of free constant symbols is unimportant for matching problems over signatures with free function symbols, since a set $\{C_1,C_2,\ldots,C_m\}$ of free constant symbols can be represented by the set $\{g(C),g(g(C)),\ldots,g^m(C)\}$, where C is a free constant symbol and g is a free unary function symbol.

If k and m are two positive integers, then the $\epsilon\mathrm{E}(k,m)$-*matching* problem consists of all instances of simultaneous elementary E-matching with at most k equations and at most m free constants. We also put

$$\epsilon\mathrm{E}(k,\omega) = \bigcup_{m=1}^{\infty} \epsilon\mathrm{E}(k,m) \quad \text{and} \quad \epsilon\mathrm{E}(\omega,m) = \bigcup_{k=1}^{\infty} \epsilon\mathrm{E}(k,m)$$

Thus, in $\epsilon\mathrm{E}(k,\omega)$-matching the signature has an unbounded number of free constant symbols, while in $\epsilon\mathrm{E}(\omega,m)$-matching the systems of equations have unbounded length. We define similarly $\#\epsilon\mathrm{E}(k,m)$-matching, $\#\epsilon\mathrm{E}(k,\omega)$-matching, and $\#\epsilon\mathrm{E}(\omega,m)$-matching.

Eker [Eke93] established that $\epsilon\mathrm{AC}(1,\omega)$-matching is a NP-complete problem. Similarly, Hermann and Kolaitis [HK99] proved that the corresponding counting problem is #P-complete.

Theorem 3.2 ([Eke93, HK99]) *$\epsilon\mathrm{AC}(1,\omega)$-matching is* NP*-complete and $\#\epsilon\mathrm{AC}(1,\omega)$-matching is* #P*-complete.*

Baader and Siekmann [BS94] showed that 1-IN-3 SAT can be reduced to elementary AC-matching with an unbounded number of equations and two free constants. A slight refinement of their reduction shows that actually one free constant suffices to yield NP-completeness, provided the number of equations is unbounded.

Theorem 3.3 ([HK99]) *The decision problem $\epsilon\mathrm{AC}(\omega,1)$-matching is* NP*-complete and the counting problem $\#\epsilon\mathrm{AC}(\omega,1)$-matching is* #P*-complete.*

If both the length of the system and the number of free constants are kept bounded, then the elementary AC-matching decision and counting problems are tractable.

Theorem 3.4 ([HK99]) *$\epsilon\mathrm{AC}(k,m)$-matching is in* P *and $\#\epsilon\mathrm{AC}(k,m)$-matching is in* FP*, for all $k \geq 1$ and all $m \geq 1$.*

The preceding theorems give a complete picture of the computational complexity of simultaneous elementary AC-matching problems. Next, we study the complexity of elementary matching for the equational theory ACU of commutative monoids and unveil a different picture. Indeed, simultaneous elementary ACU-matching turns out to be tractable for systems of bounded length, even if the signature contains an unbounded number of free constants.

Theorem 3.5 ([HK99]) *$\epsilon\mathrm{ACU}(\omega,1)$-matching is* NP*-complete and $\#\epsilon\mathrm{ACU}(\omega,1)$-matching is* #P*-complete. In contrast, $\epsilon\mathrm{ACU}(k,\omega)$-matching is in* P *and $\#\epsilon\mathrm{ACU}(k,\omega)$-matching is in* FP*, for every $k \geq 1$.*

Finally, we examine the complexity of elementary matching for the equational theory A of semigroups. Angluin [Ang80] showed that the problem $\epsilon\mathrm{A}(1,2)$-matching is NP-complete; moreover, Benanav *et al.* [BKN87] proved that $\epsilon\mathrm{A}(\omega,1)$-matching is NP-complete. The following result shows that all other cases of elementary A-matching are tractable.

Theorem 3.6 ([Ang80, BKN87, HK99]) *The $\epsilon\mathrm{A}(1,m)$-matching is* NP*-complete and $\#\epsilon\mathrm{A}(1,m)$-matching is* #P*-complete, for every $m \geq 2$. Moreover, $\epsilon\mathrm{A}(\omega,1)$-matching is* NP*-complete and $\#\epsilon\mathrm{A}(\omega,1)$-matching is* #P*-complete. In contrast, $\epsilon\mathrm{A}(k,1)$-matching is in* P *and $\#\epsilon\mathrm{A}(k,1)$-matching is in* FP*, for all $k \geq 1$.*

Simultaneous elementary A-matching

number of equations	number of constants		
	1	$m \geq 2$	ω
$k \geq 1$	P / FP		
ω	NP-complete / #P-complete		

Simultaneous elementary AC-matching

number of equations	number of constants		
	1	$m \geq 2$	ω
$k \geq 1$	P / FP		
ω	NP-complete / #P-complete		

Simultaneous elementary ACU-matching

number of equations	number of constants		
	1	$m \geq 2$	ω
$k \geq 1$	P / FP		
ω	NP-complete / #P-complete		

Figure 7: Complexity results for simultaneous elementary E-matching

Hermann and Kolaitis [HK99] investigate further the complexity of simultaneous elementary E-matching problems for the equational theories AC and ACU with bounded variable occurrence.

Figure 7 summarizes the previously mentioned complexity results for elementary A-, AC-, and ACU-matching

3.2 Complexity of equational unification

An E-*unifier* of s and t is a substitution ρ such that $s\rho =_{\mathrm{E}} t\rho$ holds. Whenever such an E-unifier exists, we say that the terms s and t can be E-unified.

A *complete set of* E-*unifiers* of s and t is a set S of substitutions such that the following hold:

1. Each substitution $\rho \in S$ is an E-unifier of s and t, and, moreover, $dom(\rho) \subseteq V$, where $V = Var(s) \cup Var(t)$ is the set of variables occurring in s or t;
2. For every E-unifier σ of s and t, there is a substitution $\rho \in S$ such that $\rho \leq^V_{\mathrm{E}} \sigma$.

S is a *minimal complete set of* E-*unifiers* of s and t if, in addition, every two distinct members of S are $\leq^V_{\mathrm{E}}$-incomparable, that is, for all substitutions $\sigma, \rho \in S$ the condition $\sigma \leq^V_{\mathrm{E}} \rho$ implies $\sigma = \rho$.

It is possible that two terms s and t are E-unifiable, but no minimal complete set of E-unifiers of s and t exists. On the other hand, if a minimal complete set of E-unifiers of s and t exists, then it is unique up to $\equiv^V_{\mathrm{E}}$. In this case, we let $\mu\mathrm{CSU}_{\mathrm{E}}(s,t)$ denote the (unique up to $\equiv^V_{\mathrm{E}}$) minimal complete set of E-unifiers of s and t, if s and t are unifiable, or the empty set, otherwise. Equational theories can be classified according to their *unification type*, which takes into account the existence and the cardinalities of the sets $\mu\mathrm{CSU}_{\mathrm{E}}(s,t)$. In particular, a theory E is said to be *unitary* if for every pair of terms (s,t) the set $\mu\mathrm{CSU}_{\mathrm{E}}(s,t)$ exists and $|\mu\mathrm{CSU}_{\mathrm{E}}(s,t)| \leq 1$. Similarly, E is said to be *finitary* if for every pair of terms (s,t) the set $\mu\mathrm{CSU}_{\mathrm{E}}(s,t)$ exists and is finite.

With every equational theory E we associate the following decision and counting problems, analogously to equational matching. Note that for the counting problem the equational theory must be finitary.

E-UNIFICATION
Input: Two terms s and a t, and an equational theory E.
Question: Can s be E-unified with t, i.e., is there a substitution ρ, such that $s\rho =_{\mathrm{E}} t\rho$?

#E-UNIFICATION
Input: Two terms s and a t, and an equational theory E.
Output: Cardinality of the set $\mu\mathrm{CSU}_{\mathrm{E}}(s,t)$.

By examining the signature $\mathcal{F}$ over which the terms of unification problems in the theory Th(E) have been built, we distinguish between three different kinds of E-unification. Let $sig(\mathrm{E})$ be the set of all function and constant symbols occurring in the equational axioms of E. If $\mathcal{F} = sig(\mathrm{E})$ holds, then we speak about *elementary* E-*unification*. If the signature $\mathcal{F}$ contains in addition free constant symbols, but no free function symbols, then we speak about E-*unification with constants*. Finally, if the signature $\mathcal{F}$ contains free function symbols of arbitrary arities, then we speak about *general* E-*unification*.

The aforementioned equational matching problems give immediately the lower bounds for the corresponding unification problems. This is, for instance, the case of A-unification where the lower bound is exactly that of A-matching, whereas the upper bound is PSPACE. There are equational theories, like Boolean rings or Abelian groups, where the unification problems can be

theory	complexity		
	elementary	with constants	general
$\emptyset$	$\longleftarrow$	$\longleftarrow$	linear [PW78]
A	$\longleftarrow$	NP-hard and in PSPACE [Pla99]	NP-hard
C		NP-complete	NP-complete
AC	NP-complete	NP-complete	NP-complete [KN92]
ACI	$\longleftarrow$	in P	NP-complete [KN92]
ACIU	$\longleftarrow$	in P	NP-complete [Nar96]
AG	$\longleftarrow$	in P	NP-complete
BR	NP-complete	Π_2P-complete	PSPACE-complete

Figure 8: Complexity results for equational unification decision problems

always transformed into an equivalent matching problem, because of the presence of the nilpotence axiom or the inverse axiom. Figure 8 summarizes the complexity results for some equational unification decision problems.

The work presented here concerning #E-matching problems suggests that a similar investigation should be carried out for #E-unification problems. Although the mentioned counting complexity results imply that several #E-unification problems are #P-hard, we already know that there are equational theories E, such as AC, for which #E-unification is not a member of #P. Indeed, Domenjoud [Dom92] found AC-unification problems with n variables whose minimal complete set of AC-unifiers has $O(2^{2^n})$ elements. Since a counting problem in #P takes values that are bounded by a single exponential in the size of the input, it follows that #AC-unification is not in #P. It is an interesting open problem to analyze the computational complexity of #AC-unification and to determine whether it is complete for some higher counting complexity class. Results along these lines will delineate the computational difference between matching and unification in a precise manner and will confirm the intuition that unification is harder than matching.

Two interesting results were proved by Hermann and Kolaitis [HK00] concerning the difference between computing the minimal complete sets of unifiers for Boolean rings and Abelian groups in the case of unification with constants on one hand and general unification on the other. The counting problem #AG-unification with constants was known to be in FP, since it can be solved by polynomial-time methods over the integers $\mathbb{Z}$ known from linear algebra. However, the following theorem indicates that the general counting problems for Abelian groups and Boolean rings are intractable.

Theorem 3.7 ([HK00]) *The problems general* #AG*-unification and general #BR-unification are both* #P*-hard.*

This result proves that the cardinality of the corresponding minimal complete sets of unifiers for

Boolean rings and for Abelian groups cannot be computed in polynomial time unless P = NP.

3.3 Special interest for AC and Hilbert bases

The Hilbert basis of a homogeneous system of linear Diophantine equations over the non-negative integers is the set of all non-zero vectors that are *minimal* solutions with respect to the pointwise order. This set forms indeed a basis of the space of solutions of the system, that is, every solution can be written as a positive linear combination of vectors from the Hilbert basis, and no member of the Hilbert basis can be expressed as a positive linear combination of other members. Moreover, this basis is essentially unique.

Computing the Hilbert basis of a homogeneous system of linear Diophantine equations over non-negative integers has turned out to be one of the key problems in automated deduction. Its importance in this area emerged through the work of Stickel [Sti81], who designed the first algorithm for AC-unification. Stickel showed that the minimal complete set of unifiers of a simultaneous elementary AC-unification problem can be obtained from the Hilbert basis of an associated homogeneous system of linear Diophantine equations over non-negative integers. Indeed, the minimal complete set of AC-unifiers is the set of all *compatible* subsets of the Hilbert basis of that system, where compatible in this context means that every variable can be instantiated by a non-zero linear combination of the members of the compatible subset. Following the publication of Stickel's algorithm [Sti81], researchers became interested in algorithms for computing the Hilbert basis. Every algorithm for computing the Hilbert basis of a system can also be used to count at the same time the number of elements of the Hilbert basis, therefore we consider the counting complexity of the Hilbert basis problem.

A homogeneous linear Diophantine system over non-negative integers is a system of equations $S\colon Ax = 0$, where $A = (a_i^j)_k^n$ is a $k \times n$ integer matrix and $x = (x_1, \ldots, x_n)$ is a vector of variables ranging over non-negative integers. We say that a solution s of S is *nontrivial* if it is different from the all-zero solution $(0, \ldots, 0)$. We say that a solution $s = (s_1, \ldots, s_n)$ of S is *smaller* than a solution $s' = (s'_1, \ldots, s'_n)$, and write $s < s'$, if $s \neq s'$ and, for all $i = 1, \ldots, n$, it is the case that $s_i \leq s'_i$. The relation $<$ is called the *pointwise order* on solutions. A solution s is *minimal* if it is nontrivial and there is no smaller nontrivial solution s'', that is, $s'' < s$ is false for every nontrivial solution s'' of S.

The *Hilbert basis* $H(S)$ of the system S is the set of all minimal solutions of S. This set is indeed a *basis* for the space of nontrivial solutions of S, which means that no minimal solution can be expressed as a positive linear combination of the other minimal solutions, whereas every nontrivial solution can be expressed as a positive linear combination of minimal solutions. The Hilbert basis $H(S)$ is finite and it is the unique basis of the space of nontrivial solutions of S.

It is well known that Hilbert bases can be used to compute minimal complete sets of AC-unifiers. Indeed, let $AX \doteq_{\mathrm{AC}} A'X'$ be a simultaneous elementary AC-unification problem, where A and A' are matrices over non-negative integers, $X = (X_1, \ldots, X_j)$ and $X' = (X_{j+1}, \ldots, X_n)$ are not necessarily disjunctive vectors of formal variables, and $+$ is the unique AC-symbol. With this AC-unification problem, associate the homogeneous linear Diophantine system $S\colon (A - A')x = 0$, where the arithmetic variable x_i corresponds to the formal variable X_i for $i = 1, \ldots, n$. Consider the Hilbert basis $H(S)$ of the system S over the variables $x_1, \ldots, x_n$. Let $\{\alpha_1, \ldots, \alpha_m\}$ be a subset of $H(S)$ and $v = (v_1, \ldots, v_m)$ be a vector of new variables. For each $i = 1, \ldots, n$, assign the linear expression $\alpha_1^i v_1 + \cdots + \alpha_m^i v_m$ to the variable x_i, where α_j^i is the i-th coordinate of the vector α_j. We say that $\{\alpha_1, \ldots, \alpha_m\}$ is a *compatible subset* of $H(S)$ if, for each variable x_i, there exists a vector α_j such that $\alpha_j^i \neq 0$, that is, the variable x_i is not assigned the value 0. The minimal complete set

of unifiers of the AC-unification problem $AX \doteq_{\mathrm{AC}} A'X'$ turns out to be the set of all compatible subsets of $H(S)$ of the system S above, where $x_i \mapsto \alpha_1^i v_1 + \cdots + \alpha_m^i v_m$ is the substitution of the variable x_i, when $\{\alpha_1, \ldots, \alpha_m\}$ is the chosen compatible subset.

Hermann, Juban, and Kolaitis [HJK99] considered the following counting problems that characterize well the computational complexity of generating the Hilbert basis and, subsequently, the AC-unifiers.

#HILBERT
Input: A system of homogeneous linear Diophantine equations $S\colon Ax = 0$ over non-negative integers.
Output: The cardinality of the Hilbert basis $H(S)$ of S.

#COMPATIBLE SUBSETS
Input: A set T of vectors of non-negative integers that are pairwise incomparable in the pointwise order and linearly independent with respect to linear combinations with non-negative coefficients.
Output: The cardinality of the set of all compatible subsets of T.

Hermann, Juban, and Kolaitis [HJK99] proved the following reasonably tight upper and lower bound for the Hilbert basis counting problem, even though they do not decisively pin down its exact complexity. The #P-hardness proof is done using Hall's theorem.

Theorem 3.8 *The counting problem* #HILBERT *is* #P*-hard and belongs to the class* #NP*.*

The preceding theorem yields upper and lower bounds for the complexity of counting the Hilbert basis. An inspection of the #P-hardness proof reveals that #HILBERT would be in #P, if testing a solution for minimality were solvable in polynomial time. Durand, Hermann, and Juban [DHJ99], however, have shown that it is a coNP-complete problem to tell whether a given solution of a homogeneous linear Diophantine system is minimal for a homogeneous linear Diophantine system. Thus, assuming P $\neq$ NP, to prove that #HILBERT is in #P would require one to come up with a very different set of witnesses for #HILBERT and show that membership in that witness set is in polynomial time. It is believed that this is not possible and conjectured that #HILBERT is *not* in #P.

Let #HILBERT(m) be the restriction of #HILBERT to systems of equations with at most m occurrences of each variable. It turns out that #HILBERT(3) has the same complexity as the original problem, what carries over to each $m > 3$, whereas #HILBERT(1) is clearly in FP by a simple combinatorial argument mentioned in [LC89]. The counting problem #HILBERT(2) has been proved in [HJK99] to be included in #P by means of a long case analysis. However the general lower bound proof does not work for this special case any more.

Stickel's algorithm [Sti81] for simultaneous elementary AC-unification proceeds by first finding the Hilbert basis of the associated homogeneous linear Diophantine system, and then producing the set of all compatible subsets of that basis. To gain insight into the inherent complexity of the latter algorithm, Hermann, Juban, and Kolaitis [HJK99] examined the computational complexity of counting the number of compatible subsets of a given set T of linearly independent and pairwise incomparable vectors of non-negative integers.

Theorem 3.9 *The counting problem* #COMPATIBLE SUBSETS *is* #P*-complete.*

3.4 Combination of unification algorithms

The development of combination algorithms originated with Stickel's algorithm for general AC-unification [Sti81]. Stickel first constructed an algorithm for elementary AC-unification and then introduced a special-purpose combination algorithm for general AC-unification that used the algorithm for elementary AC-unification and the algorithm for syntactic unification as subroutines. Similar work was carried out by others. This triggered the research on the combination of unification algorithms for disjoint equational theories, a problem that was finally solved by Schmidt-Schauß [SS89]. Using a new approach, Baader and Schulz [BS96] presented a combination method for decision problems in disjoint equational theories; a slight modification gives rise to a method for combining algorithms for unification in two disjoint equational theories.

Every known combination algorithm for equational unification has an exponential running time. In particular, even if there exist polynomial-time unification algorithms A_1 and A_2 for the disjoint theories $\mathrm{Th}(\mathrm{E}_1)$ and $\mathrm{Th}(\mathrm{E}_2)$, every known general combination method will give rise to an exponential algorithm A for unification in the theory $\mathrm{Th}(\mathrm{E}_1 \cup \mathrm{E}_2)$. There was even a quest for a polynomial-time combination method launched within the AC-unification case [BHK+88]. Hermann and Kolaitis [HK00] showed that this exponential-time behavior is not a deficiency of the known combination algorithms, but rather is caused by the inherent intractability of the combination problem. More precisely, they show that there is no polynomial-time general combination algorithm for unification in finitary equational theories, unless the complexity class #P of counting problems is contained in the class FP of function problems solvable in polynomial time.

Theorem 3.10 ([HK00]) *Unless* #P *is contained in* FP*, there does not exist a combination algorithm A for $\mathrm{E}_1 \cup \mathrm{E}_2$-unification, where E_1 and E_2 are disjoint equational theories, such that A runs in polynomial time using oracles for the E_1-unification problem and the E_2-unification problem.*

The previous result holds already in the presence of a single unary function symbol. Based on similar ideas, Schulz [Sch00] investigated a large class of tractable and intractable instances of combination problems for unification and disunification decision problems. Following from his analysis, it seems that already very simple and natural conditions on the equational theory E imply NP-hardness of the combination problem for the decision case of unification, where E is one of the involved disjoint theories.

3.5 Open problems

Probably the most interesting problem in the complexity of equational matching and unification is the problem of determining the exact complexity of A-unification. The upper bound was recently pushed down from multiple exponential to PSPACE by Plandowski [Pla99]. The lower bound is still NP-hard, coming from the lower bound for A-matching. Is it possible to push the lower bound higher in the polynomial hierarchy or even to PSPACE? Another possibility would be to push down the upper bound even further. However, in the scope of the very simple lower bound proof as opposed to Plandowski's sophisticated method for proving PSPACE membership, it is more probable to find a higher lower bound, even if there exists a conjecture that A-unification could be NP-complete.

Another interesting open question consists of determining the exact complexity of the problem #HILBERT to count the cardinality of a homogeneous linear Diophantine system of equations over non-negative integers. The lower bound is #P-hard, whereas the upper bound is #NP, what makes a difference of two counting complexity classes. Hermann, Juban, and Kolaitis conjecture that the problem is #NP-complete. There are two reasons to believe that the conjecture is right. The first

one is that testing whether a given vector belongs to the Hilbert basis of a given system is coNP-complete [DHJ99]. The second is that a similar problem of counting the minimal solutions of a propositional formula has been proved #NP-complete by Durand, Hermann, and Kolaitis [DHK00].

References

[Ang80] D. Angluin. Finding patterns common to a set of strings. *Journal of Computer and System Science*, 21:46–62, 1980.

[APT79] B. Aspvall, M. R. Plass, and R. E. Tarjan. A linear-time algorithm for testing the truth of certain quantified Boolean formulas. *Information Processing Letters*, 8(3):121–123, 1979.

[Baa98] F. Baader. On the complexity of Boolean unification. *Information Processing Letters*, 67(4):215–220, 1998.

[BHK+88] H.-J. Bürckert, A. Herold, D. Kapur, J. H. Siekmann, M. E. Stickel, M. Tepp, and H. Zhang. Opening the AC-unification race. *Journal of Automated Reasoning*, 4(4):465–474, 1988.

[BHRV01] E. Böhler, E. Hemaspaandra, S. Reith, and H. Vollmer. Equivalence problems for boolean constraint satisfaction. Technical Report 282, Institut für Informatik, Universität Würzburg, 2001.

[BKN87] D. Benanav, D. Kapur, and P. Narendran. Complexity of matching problems. *Journal of Symbolic Computation*, 3(1-2):203–216, 1987.

[BS94] F. Baader and J. H. Siekmann. Unification theory. In D.M. Gabbay, C.J. Hogger, and J.A. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 2: Deduction Methodologies, pages 41–125. Oxford University Press, Oxford (UK), 1994.

[BS96] F. Baader and K. U. Schulz. Unification in the union of disjoint equational theories: Combining decision procedures. *Journal of Symbolic Computation*, 21(2):211–243, 1996.

[BS01] F. Baader and W. Snyder. Unification theory. In J. A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume I, pages 447–533. Elsevier, 2001.

[Cad92] M. Cadoli. The complexity of model checking for circumscriptive formulae. *Information Processing Letters*, 44(3):113–118, 1992.

[CCJ94] M. C. Cooper, D. A. Cohen, and P. Jeavons. Characterising tractable constraints. *Artificial Intelligence*, 65(2):347–361, 1994.

[CD94] H. Comon and C. Delor. Equational formulae with membership constraints. *Information and Computation*, 112(2):167–216, 1994.

[CH96] N. Creignou and M. Hermann. Complexity of generalized satisfiability counting problems. *Information and Computation*, 125(1):1–12, 1996.

[CH97] N. Creignou and J-J. Hébrard. On generating all solutions of generalized satsifiability problems. *Informatique Théorique et Applications*, 31(6):499–511, 1997.

[CKS01] N. Creignou, S. Khanna, and M. Sudan. *Complexity Classifications of Boolean Constraint Satisfaction Problems.* SIAM Monographs on Discrete Mathematics and Applications. SIAM, Philadelphia (PA), 2001.

[CL89] H. Comon and P. Lescanne. Equational problems and disunification. *Journal of Symbolic Computation*, 7(2 & 3):371–425, 1989.

[Col84] A. Colmerauer. Equations and inequations on finite and infinite trees. In *Proceedings International Conference on Fifth Generation Computer Systems (FGCS'84), Tokyo (Japan)*, pages 85–99. OHMSHA Ltd. Tokyo and North-Holland, 1984.

[CP95a] R. Caferra and N. Peltier. Decision procedures using model building techniques. In H. Kleine Büning, editor, *Proceedings 9th International Workshop on Computer Science Logic (CSL'96), Paderborn (Germany)*, volume 1092 of *Lecture Notes in Computer Science*, pages 130–144. Springer-Verlag, September 1995.

[CP95b] R. Caferra and N. Peltier. Extending semantic resolution via automated model building: Applications. In *Proceedings 14th International Joint Conference on Artificial Intelligence (IJCAI'95), Montreal (Canada)*, pages 328–334. Morgan Kaufmann, 1995.

[Cre95] N. Creignou. A dichotomy theorem for maximum generalized satisfiability problems. *Journal of Computer and System Science*, 51(3):511–522, 1995.

[CZ90] R. Caferra and N. Zabel. Extending resolution for model construction. In J. van Eijck, editor, *Proceedings Logics in AI, European Workshop (JELIA'90), Amsterdam (The Netherlands)*, volume 478 of *Lecture Notes in Computer Science (in Artificial Intelligence)*, pages 153–169. Springer-Verlag, September 1990.

[Dal97] V. Dalmau. Some dichotomy theorems on constant free boolean formulas. Technical Report LSI-97-43-R, Departament LSI, Universitat Politècnica de Catalunya, 1997.

[DG00] M. E. Dyer and C. S. Greenhill. The complexity of counting graph homomorphisms. *Random Structures and Algorithms*, 17(3-4):260–289, 2000.

[DHJ99] A. Durand, M. Hermann, and L. Juban. On the complexity of recognizing the Hilbert basis of a linear Diophantine system. In M. Kutyłowski, L. Pacholski, and T. Wierzbicki, editors, *Proceedings 24th International Symposium on Mathematical Foundations of Computer Science (MFCS'99), Szklarska Porçba (Poland)*, volume 1672 of *Lecture Notes in Computer Science*, pages 92–102. Springer-Verlag, September 1999.

[DHK00] A. Durand, M. Hermann, and P. G. Kolaitis. Subtractive reductions and complete problems for counting complexity classes. In M. Nielsen and B. Rovan, editors, *Proceedings 25th International Symposium on Mathematical Foundations of Computer Science (MFCS 2000), Bratislava (Slovakia)*, volume 1893 of *Lecture Notes in Computer Science*, pages 323–332. Springer-Verlag, August 2000.

[DJ90] N. Dershowitz and J.-P. Jouannaud. Rewrite systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science B: Formal Methods and Semantics*, chapter 6, pages 243–309. Elsevier, Amsterdam, 1990.

[Dom92] E. Domenjoud. Number of minimal unifiers of the equation $\alpha x_1 + \cdots + \alpha x_b =_{AC} \beta y_1 + \cdots + \beta y_q$. *Journal of Automated Reasoning*, 8(1):39–44, 1992.

[Eke93] S. M. Eker. Improving the efficiency of AC matching and unification. Research report 2104, Institut de Recherche en Informatique et en Automatique, November 1993.

[FL96] C. Fermüller and A. Leitsch. Hyperresolution and automated model building. *Journal of Logic and Computation*, 6(2):173–203, 1996.

[FV98] T. Feder and M. Y. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: a study through Datalog and group theory. *SIAM Journal on Computing*, 28(1):57–104, 1998.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and intractability: A guide to the theory of NP-completeness*. W.H. Freeman and Co, 1979.

[GNW00] Q. Guo, P. Narendran, and D. A. Wolfram. Unification and matching modulo nilpotence. *Information and Computation*, 162(1-2):3–23, 2000.

[HJK99] M. Hermann, L. Juban, and P. G. Kolaitis. On the complexity of counting the Hilbert basis of a linear Diophantine system. In H. Ganzinger, D. McAllester, and A. Voronkov, editors, *Proceedings 6th International Conference on Logic for Programming and Automated Reasoning (LPAR'99), Tbilisi (Republic of Georgia)*, volume 1705 of *Lecture Notes in Computer Science (in Artificial Intelligence)*, pages 13–32, September, 1999. Springer-Verlag.

[HK95] M. Hermann and P. G. Kolaitis. The complexity of counting problems in equational matching. *Journal of Symbolic Computation*, 20(3):343–362, 1995.

[HK99] M. Hermann and P. G. Kolaitis. Computational complexity of simultaneous elementary matching problems. *Journal of Automated Reasoning*, 23(2):107–136, 1999.

[HK00] M. Hermann and P. G. Kolaitis. Unification algorithms cannot be combined in polynomial time. *Information and Computation*, 162(1-2):24–42, 2000.

[HN90] P. Hell and J. Nešetřil. On the complexity of H-coloring. *Journal of Combinatorial Theory, Series B*, 48:92–110, 1990.

[Ist97] G. Istrate. Counting, structure identification, and maximum consistency for binary constraint satisfaction problems. In G. Smolka, editor, *Proceedings 3rd International Conference on Principles and Practice of Constraint Programming (CP'97), Linz (Austria)*, volume 1330 of *Lecture Notes in Computer Science*, pages 136–149. Springer-Verlag, October 1997.

[JCG97] P. Jeavons, D. Cohen, and M. Gyssens. Closure properties of constraints. *Journal of the Association for Computing Machinery*, 44(4):527–548, 1997.

[Jea98] P. Jeavons. On the algebraic structure of combinatorial problems. *Theoretical Computer Science*, 200(1-2):185–204, 1998.

[Joh90] D. S. Johnson. A catalog of complexity classes. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity*, chapter 2, pages 67–161. North-Holland, Amsterdam, 1990.

[Jub99] L. Juban. Dichotomy theorem for the generalized unique satisfiability problem. In G. Ciobanu and G. Păun, editors, *Proceedings 12th Fundamentals of Computation Theory (FCT'99) Iaşi (Romania)*, volume 1684 of *Lecture Notes in Computer Science*, pages 327–337. Springer-Verlag, August 1999.

[JYP88] D. S. Johnson, M. Yannakakis, and C. H. Papadimitriou. On generating all maximal independent sets. *Information Processing Letters*, 27(3):119–123, 1988.

[KK00] L. M. Kirousis and P. G. Kolaitis. The complexity of minimal satisfiability problems. *Electronic Colloquium on Computational Complexity*, 7(TR00-082), 2000. URL = ftp://ftp.eccc.uni-trier.de/pub/eccc/reports/2000/TR00-082/index.html.

[KK01] L. M. Kirousis and P. G. Kolaitis. The complexity of minimal satisfiability problems. In A. Ferreira and H. Reichel, editors, *Proceedings 18th Symposium on Theoretical Aspects of Computer Science (STACS 2001), Dresden (Germany)*, volume 2010 of *Lecture Notes in Computer Science*, pages 407–418. Springer-Verlag, February 2001.

[KN86] D. Kapur and P. Narendran. NP-completeness of the set unification and matching problems. In J.H. Siekmann, editor, *Proceedings 8th International Conference on Automated Deduction (CADE'86), Oxford (England)*, volume 230 of *Lecture Notes in Computer Science*, pages 489–495. Springer-Verlag, July 1986.

[KN92] D. Kapur and P. Narendran. Complexity of unification problems with associative-commutative operators. *Journal of Automated Reasoning*, 9(2):261–288, 1992.

[KS96] S. Khanna and M. Sudan. The optimization complexity of constraint satisfaction problems. Tech. Note STAN-CS-TN-96-29, Stanford University, 1996.

[KS98] D. Kavvadias and M. Sideri. The inverse satisfiability problem. *SIAM Journal on Computing*, 28(1):152–163, 1998.

[KS00] O. Klíma and J. Srba. Matching modulo associativity and idempotency is NP-complete. In M. Nielsen and B. Rovan, editors, *Proceedings 25th International Symposium on Mathematical Foundations of Computer Science (MFCS 2000), Bratislava (Slovakia)*, volume 1893 of *Lecture Notes in Computer Science*, pages 456–466. Springer-Verlag, August 2000.

[KSL97] S. Khanna, M. Sudan, and L.Trevisan. Constraint satisfaction: The approximability of minimization problems. In *Proceedings 12th IEEE Conference on Computational Complexity (CCC'97), Ulm (Germany)*, pages 282–296, June 1997.

[KSW97] S. Khanna, M. Sudan, and D. P. Williamson. A complete classification of the approximability of maximization problems derived from boolean constraint satisfaction. In *Proceedings 29th Symposium on Theory of Computing (STOC'97), El Paso (Texas, USA)*, pages 11–20, 1997.

[Kun87] K. Kunen. Answer sets and negation-as-failure. In J.-L. Lassez, editor, *Proceedings 4th International Conference on Logic Programming (ICLP'87), Melbourne (Australia)*, pages 219–228. MIT Press, May 1987.

[LC89] P. Lincoln and J. Christian. Adventures in associative-commutative unification. *Journal of Symbolic Computation*, 8(1-2):217–240, 1989.

[LM87] J.-L. Lassez and K. Marriott. Explicit representation of terms defined by counter examples. *Journal of Automated Reasoning*, 3(3):301–317, 1987.

[LMM91] J.-L. Lassez, M. Maher, and K. Marriott. Elimination of negation in term algebras. In A. Tarlecki, editor, *Proceedings 16th International Symposium on Mathematical Foundations of Computer Science (MFCS'91), Kazimierz Dolny (Poland)*, volume 520 of *Lecture Notes in Computer Science*, pages 1–16. Springer-Verlag, September 1991.

[Lug89] D. Lugiez. A deduction procedure for first order programs. In G. Levi and M. Martelli, editors, *Proceedings 6th International Conference on Logic Programming (ICLP'87), Lisabon (Portugal)*, pages 585–599. MIT Press, June 1989.

[Mah88] M. J. Maher. Complete axiomatizations of the algebras of finite, rational and infinite trees. In *Proceedings 3rd IEEE Symposium on Logic in Computer Science (LICS'88), Edinburgh (Scotland)*, pages 348–357, July 1988.

[MM82] A. Martelli and U. Montanari. An efficient unification algorithm. *ACM Transactions on Programming Languages and Systems*, 4(2):258–282, 1982.

[Nar96] P. Narendran. Unification modulo ACI+1+0. *Fundamenta Informaticae*, 25(1):49–57, 1996.

[Pap94] C. H. Papadimitriou. *Computational complexity.* Addison-Wesley, 1994.

[PB83] J. S. Provan and M. O. Ball. The complexity of counting cuts and of computing the probability that a graph is connected. *SIAM Journal on Computing*, 12(4):777–788, 1983.

[Pic99] R. Pichler. Solving equational problems efficiently. In H. Ganzinger, editor, *Proceedings 16th International Conference on Automated Deduction (CADE'99), Trento (Italy)*, volume 1632 of *Lecture Notes in Computer Science (in Artificial Intelligence)*, pages 97–111. Springer-Verlag, July 1999.

[Pic00] R. Pichler. On the complexity of equational problems in CNF over a finite domain. In P. Baumgartner and H. Zhang, editors, *Proceedings 3rd International Workshop on First-Order Theorem Proving (FTP 2000), St. Andrews (Scotland, UK)*, Fachberichte Informatik der Universität Koblenz-Landau, pages 183–193, 2000. Available at http://www.uni-koblenz.de/fb4/publikationen/gelbereihe/RR-5-2000/.

[Pic01] R. Pichler. On the complexity of equational problems in CNF. *Journal of Symbolic Computation*, x(x):xx–xx, 2001. To appear.

[Pla99] W. Plandowski. Satisfiability of word equations with constants is in PSPACE. In *Proceedings 40th Symposium on Foundations of Computer Science (FOCS'99), New York (NY, USA)*, pages 495–500. IEEE Computer Society, October 1999.

[Plo72] G. D. Plotkin. Building-in equational theories. In B. Meltzer and D. Mitchie, editors, *Machine Intelligence*, volume 7, pages 73–90. Edinburgh University Press, Edinburgh, UK, 1972.

[PW78] M. S. Paterson and M. N. Wegman. Linear unification. *Journal of Computer and System Science*, 16(2):158–167, 1978.

[RV00] S. Reith and H. Vollmer. Optimal satisfiability for propositional calculi and constraint satisfaction problems. In M. Nielsen and B. Rovan, editors, *Proceedings 25th International Symposium on Mathematical Foundations of Computer Science (MFCS 2000), Bratislava (Slovakia)*, volume 1893 of *Lecture Notes in Computer Science*, pages 640–649. Springer-Verlag, August 2000.

[Sch78] T. J. Schaefer. The complexity of satisfiability problems. In *Proceedings 10th Symposium on Theory of Computing (STOC'78), San Diego (California, USA)*, pages 216–226, 1978.

[Sch97] K. U. Schulz. A criterion for intractability of E-unification with free fucntion symbols and its relevance for combination of unification algorithms. In H. Comon, editor, *Proceedings 8th Conference on Rewriting Techniques and Applications (RTA'97), Sitges (Spain)*, volume 1232 of *Lecture Notes in Computer Science*, pages 284–298. Springer-Verlag, June 1997.

[Sch00] K. U. Schulz. Tractable and intractable instances of combination problems for unification and disunification. *Journal of Logic and Computation*, 10(1):105–135, 2000.

[SM91] T. Sato and F. Motoyoshi. A complete top-down interpreter for first order programs. In V. A. Saraswat and K. Ueda, editors, *Proceedings 8th International Symposium on Logic Programming (ISLP'91), San Diego (California, USA)*, pages 35–53. MIT Press, October 1991.

[SS89] M. Schmidt-Schauß. Unification in a combination of arbitrary disjoint equational theories. *Journal of Symbolic Computation*, 8:51–99, 1989.

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

[Sto76] L. J. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3(1):1–22, 1976.

[TA87] E. Tidén and S. Arnborg. Unification problems with one-sided distributivity. *Journal of Symbolic Computation*, 3(1 & 2):183–202, 1987.

[Val79] L. G. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3):410–421, 1979.

[Vor96] S. Vorobyov. An improved lower bound for the elementary theories of trees. In M.A. McRobbie and J.K. Slaney, editors, *Proceedings 13th International Conference on Automated Deduction (CADE'96), New Brunswick, NJ (USA)*, volume 1104 of *Lecture Notes in Computer Science (in Artificial Intelligence)*, pages 275–287. Springer-Verlag, July/August 1996.