Computational Complexity of Simultaneous Elementary Matching Problems

(Extended Abstract)

Miki Hermann[1,*] Phokion G. Kolaitis[2,†]

[1] CRIN (CNRS) and INRIA-Lorraine, BP 239, 54506 Vandœuvre-lès-Nancy, France. (hermann@loria.fr)

[2] Computer and Information Sciences, University of California, Santa Cruz, CA 95064, U.S.A. (kolaitis@cse.ucsc.edu)

Abstract. The simultaneous elementary E-matching problem for an equational theory E is to decide whether there is an E-matcher for a given system of equations in which the only function symbols occurring in the terms to be matched are the ones constrained by the equational axioms of E. We study the computational complexity of simultaneous elementary matching problems for the equational theories A of semigroups, AC of commutative semigroups, and ACU of commutative monoids. In each case, we delineate the boundary between NP-completeness and solvability in polynomial time by considering two parameters, the number of equations in the systems and the number of constant symbols in the signature. Moreover, we analyze further the intractable cases of simultaneous elementary AC-matching and ACU-matching by taking also into account the maximum number of occurrences of each variable. Using graph-theoretic techniques, we show that if each variable is restricted to having at most two occurrences, then several cases of simultaneous elementary AC-matching and ACU-matching can be solved in polynomial time.

1 Introduction and Summary of Results

The design of matching and unification algorithms is one of the principal challenges faced by researchers in automated deduction. This challenge can become particularly intriguing when the terms to be matched or unified contain function symbols satisfying the axioms of an equational theory E, in which case we speak of E-matching and E-unification algorithms. Among the various equational theories that have been investigated with respect to matching and unification during the past twenty years, the equational theory AC of commutative semigroups occupies a prominent place, because of its important rôle in term rewriting and its conspicuous presence in applications (cf. [BHK+88, JK91, BS94]). In addition to AC, there has been also an extensive study of unification in the equational theories A of semigroups and ACU of commutative monoids. Indeed, A-unification algorithms solve the classical Markov's problem for word equations, while ACU-unification algorithms are used widely as building blocks for AC-unification algorithms.

Benanav, Kapur, and Narendran [BKN87] analyzed the computational complexity of decision problems for matching in various equational theories and established that A-matching, AC-matching, and ACU-matching are all NP-complete problems. Benanav et

*Partially supported by a NATO grant. Research was carried out while this author was visiting the University of California, Santa Cruz.

†Partially supported by a Guggenheim Fellowhsip and NSF Grant CCR-9307758.

al. [BKN87] discovered also that AC1-matching, the restriction of AC-matching to terms in which each variable occurs only once, is solvable in polynomial-time. This tractable case of AC-matching turned out to be the rather isolated, since Verma and Ramakrishnan [VR92] showed that AC-matching is NP-complete even if each variable is allowed to have only two occurrences in the terms being matched. Aiming to develop a different perspective on the complexity of matching, Kolaitis and Hermann [HK94] introduced and studied a class of counting problems that arise naturally in this context. More specifically, if E is an equational theory, then #E-matching is the following problem: given a term s and a ground term t, find the cardinality of a minimal complete set of E-matchers of s and t. The motivation for considering these counting problems comes from the fact that matching and unification algorithms should not only decide whether two given terms can be E-matched (E-unified), but should also return a minimal complete set of E-matchers (E-unifiers). In particular, such algorithms can solve at the same time the corresponding #E-matching problem. Thus, by identifying the computational complexity of #E-matching, we gain a deeper insight into the expected behavior of matching and unification algorithms than the insight obtained from the analysis of the corresponding decision problem. In the paper [HK94], it was shown that #A-matching, #AC-matching, and #ACU-matching are all #P-complete problems. The concept of #P-completeness was introduced by Valiant [Val79] as a means of quantifying the computational difficulty of counting problems. In many respects, a #P-completeness result for a counting problem indicates a higher level of intractability than a NP-completeness result for the corresponding desision problem. Valiant [Val79] showed also that there exist polynomial-time decision problems whose counting version is #P-complete. As it turns out, this phenomenon occurs also in matching, because in [HK94] it was shown that that #AC1-matching is a #P-complete problem.

If one takes a closer look at the above NP-hardness and #P-hardness results for matching, then one realizes that their proofs make use of terms containing *free* function symbols, i.e., function symbols that are not constrained by the axioms of the underlying equational theory. To put it differently, in these hardness results the signature over which the terms are built is allowed to vary and is given as part of the input of the decision and counting problem under consideration. In turn, this raises the question: does the complexity of the matching problems change, if the signature contains no free function symbols? There are several other situations where it has been established that the presence of free function symbols affects the properties of matching and unification. As Baader and Siekman [BS94] write, "It is important to note that the signature over which the terms of the unification problems may be built has considerable influence on the unification type and on the existence of unification algorithms". For this reason, in studying an equational theory E one distinguishes between the case of *elementary* E-matching (E-unification), where the signature contains no free function symbols, and the case of *general* E-matching (E-unification), where the signature contains free function symbols of arbitrary arity. In both cases the signature may contain one or more free constant symbols. Elementary E-matching (E-unification) extends naturally to *simultaneous elementary* E-matching (E-unification), where, instead of just a single equation $s \stackrel{?}{=}_{\mathrm{E}} t$, one is given a *system* of equations $s_1 \stackrel{?}{=}_{\mathrm{E}} t_1, \ldots, s_k \stackrel{?}{=}_{\mathrm{E}} t_k$ for which an E-matcher (E-unifier) is sought. Note that in the case of general E-matching (E-unification) such systems reduce to a single equation $f(s_1, \ldots, s_k) \stackrel{?}{=}_{\mathrm{E}} f(t_1, \ldots, t_k)$, where f is a free function symbol.

In this paper, we carry out a systematic investigation of the computational complexity of simultaneous elementary E-matching decision and counting problems, where E is one of the equational theories A, AC or ACU. Our goal is to identify the rôle of the signature on the complexity of matching and to delineate the boundary between intractability and tractability for elementary matching in these theories. We classify simultaneous elementary matching problems according to the number of equations in the system and the number of free constant symbols in the signature. Eker [Eke93] proved that elementary AC-matching is NP-complete for single equations over signatures with an unbounded number of free constant symbols. At the other end of the scale, Baader and Siekmann [BS94] pointed out that simultaneous elementary AC-matching is NP-complete for systems of unbounded length over signatures with two free constant symbols. In fact, a slight modification of that reduction shows that a single free constant symbol suffices for establishing NP-hardness. We complement the above results in two ways. First, we point out that in these cases the corresponding counting problem is #P-complete. After this, we establish that if the systems are of fixed length and the signature contains a fixed number of free constant symbols, then both the decision problem and the counting problem for simultaneous elementary AC-matching can be solved in polynomial time using dynamic programming. Thus, intractability occurs in AC-matching only when the length of the systems or the number of free constant symbols grow beyond any bounds.

We use the same classification to study and identify the complexity of decision and counting problems in simultaneous elementary A-matching and ACU-matching. It turns out that A-matching is tractable in fewer cases than AC-matching, whereas ACU-matching is tractable in more cases. We show that simultaneous elementary A-matching is solvable in polynomial time only when the systems are of fixed length and the signature has a single free constant symbol. In all other cases, the decision problem is NP-complete and the counting problem is #P-complete. In contrast, simultaneous elementary ACU-matching is solvable in polynomial time as long as the systems are of fixed length, even if the signature has an unbounded number of free constant symbols.

Finally, we take a closer look at the intractable cases of simultaneous elementary AC-matching and ACU-matching by considering a third parameter as a resource, namely the maximum number of variable occurrences in the systems. While for general AC-matching one occurrence suffices for establishing #P-hardenss and two occurrences suffice for establishing NP-hardness, we show that here the dividing line between tractability and intractability is one level higher at two occurrences and three occurrences, respectively. In particular, we show that the decision problem for simultaneous elementary AC-matching with each variable occurring twice is solvable in polynomial time over signatures with one free constant symbol, even if the systems are of unbounded length. Moreover, the decision problem for simultaneous elementary ACU-matching with each variable occurring twice is solvable in polynomial time, even if the signature contains an unbounded number of free constant symbols and the systems are of unbounded length. We derive polynomial-time algorithms for these problems by reducing them to a class of graph-theoretic problems, known as *b-matching*, that ask whether a given graph contains a subgraph whose nodes satisfy certain degree constraints. Although b-matching problems have been studied extensively in the context of graph theory to our knowledge this is the first time that a connection has been made between these graph-theoretic problems and matching problems in equational theories. This connection opens the road for incorporating efficient b-matching algorithms into AC-matching and ACU-matching algorithms.

2 Preliminaries

This section contains the definitions of the main concepts and a minimum amount of the necessary background material from equational matching and computational complexity. Additional material on these topics can be found in [JK91, BS94] and [Pap94].

Matching Problems in Equational Theories and their Complexity

A *signature* $\mathcal{F}$ is a countable set of function and constant symbols. If $\mathcal{X}$ 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 $\mathcal{X}$. Capital letters $X, Y, Z, \ldots$, as well as capital letters with subscripts $X_i, Y_i, Z_i, \ldots$ will be used to denote variables in $\mathcal{X}$.

An *identity* over $\mathcal{F}$ is a first-order sentence of the form $(\forall X_1)\ldots(\forall X_n)(l = r)$, where l and r are terms in $\mathcal{T}(\mathcal{F},\mathcal{X})$ with variables among $X_1,\ldots,X_n$. Every set E of identities can be viewed as the set of *equational axioms* of an *equational theory* Th(E) consisting of all identities over $\mathcal{F}$ that are logically implied by E. By an abuse of terminology, we will often say the "equational theory E", instead of "the equational theory Th(E)". The notation $s =_{\mathrm{E}} t$ denotes that the identity $(\forall X_1)\ldots(\forall X_n)(s = t)$ is a member of Th(E).

Our main focus will be on the equational theory AC of commutative semigroups and on the equational theory ACU of commutative monoids. For both these theories, the signature $\mathcal{F}$ contains a binary function symbol $+$ that is assumed to be associative and commutative. Thus, the equational axioms of AC are

$$\text{A: } (\forall X)(\forall Y)(\forall Z)(X + (Y + Z) = (X + Y) + Z) \qquad \text{C: } (\forall X)(\forall Y)(X + Y = Y + X)$$

For ACU, the signature $\mathcal{F}$ contains also a constant symbol 0, which is the *unit* element for $+$. Thus, ACU satisfies also the identity U: $(\forall X)(X + 0 = X)$. We also consider the equational theory A of semigroups, whose only equational axiom is associativity.

A *substitution* is a mapping $\rho\colon\mathcal{X} \to \mathcal{T}(\mathcal{F},\mathcal{X})$ such that $X\rho = X$ for all but finitely many variables X. We say that a term s E-*matches* a ground term t if there is a substitution ρ such that $s\rho =_{\mathrm{E}} t$. In this case, we call ρ an E-*matcher* of s and t. It can be shown that a term s E-matches a ground term t if and only if the equation $s \stackrel{?}{=}_{\mathrm{E}} t$ can be solved in the quotient algebra $\mathcal{T}(\mathcal{F},\mathcal{X})/=_{\mathrm{E}}$.

E-*matching* is the following decision problem: given a term s and a ground term t over a signature $\mathcal{F}$, decide whether s E-matches t. Benanav, Kapur, and Narendran [BKN87] investigated the computational complexity of E-matching and showed that A-matching, AC-matching, and ACU-matching are all NP-complete problems.

Beside the decision problem, there is an another important (and often more challenging) algorithmic problem arising in matching, namely the problem of designing algorithms which not only decide whether a given term s E-matches a ground term t, but also return as value a *minimal complete* set $\mu\mathrm{CSM}_{\mathrm{E}}(s,t)$ of E-matchers of s and t, provided that s E-matches t. An insight into this problem can be gained by studying a related *counting* problem in equational matching, that asks for the number of E-matchers in a minimal complete set. Before describing this approach in more detail, we state the basic relevant facts from computational complexity.

#P is the class of all counting problems that are computable in polynomial-time using *counting* Turing machines, i.e., non-deterministic Turing machines equipped with an

additional output tape on which it prints in binary the number of its accepting computations. The class #P was introduced by Valiant [Val79], who established the existence of #P-*complete* problems, i.e., counting problems in #P such that every problem in #P can be reduced to them via restricted polynomial-time reductions that preserve the number of solutions (*parsimonious* reductions) or at least make it possible to compute the number of solutions of one problem from the other (*counting* reductions). The prototypical #P-complete problem is #SAT, which asks for the number of satisfying assignments of a given Boolean formula. Valiant [Val79] discovered also that there are polynomial-time decision problems, such as perfect matching in bipartite graphs, whose corresponding counting problem is #P-complete. The prevalent view in complexity theory is that a #P-completeness result indicates a higher level of intractability than a NP-completeness result does. No #P-complete problem is known to be (nor is believed to be) a member of the class FPH, the functional analog of the polynomial hierarchy PH (cf. [Pap94]).

In [HK94], we introduced a class of *counting* problems in equational matching and studied their computational complexity using tools from the theory of #P-completeness. More precisely, #E-*matching* is the following counting problem: given a term s and a ground term t, find the cardinality of $\mu\mathrm{CSM}_{\mathrm{E}}(s,t)$. This problem is well-defined for every *finitary* equational theory E. Observe that, on the face of it, #E-matching is a problem of intermediate difficulty between the decision problem for E-matching and the problem of designing matching algorithms that return minimal complete sets of E-matchers. Thus, a #P-completeness result about the #E-matching problem of some equational theory E suggests that computing minimal complete sets of E-matchers is a truly intractable problem. In [HK94] we showed that #A-matching, #AC-matching, and #ACU-matching are all #P-complete problems. Moreover, we established there that even #AC1-matching is #P-complete, where AC1 is the restriction of AC to *linear terms* (terms in which each variable occurs only once). In contrast, [BKN87] showed that the decision problem for AC1-matching is solvable in polynomial time.

Simultaneous Elementary Matching

Let $\mathcal{F}$ be a signature, $\mathcal{X}$ a set of variables, and E an equational theory whose axioms are identities over $\mathcal{F}$. The signature $\mathcal{F}$ may contain function or constant symbols that do not occur in the equational axioms of E. Such symbols are called *free*, since they are not constrained in any way by E. As mentioned earlier, the existence of free function or constant symbols in the signature may affect the structural properties of an equational theory E and have an impact on unification or matching algorithms for E (cf. [BS94]). Closer to our interests here, it should be noted that the proof in [BKN87] that AC-matching is NP-hard makes an essential use of free function symbols in the signature. More recently, Eker [Eke93] showed that AC-matching remains NP-hard over signatures with no free function symbols; in the proof of this result, however, the signature at hand contains an unbounded number of free constant symbols. The #P-hardness proofs for #AC-matching and #AC1-matching in [HK94] depend on free function symbols and free constant symbols in the signature. Thus, it is natural to ask whether and how the computational complexity of E-matching and #E-matching changes, if the instances of these problems are restricted to terms over signatures having no free function symbols and possibly only a bounded number of free constant symbols.

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. In the sequel, we will also be interested in extensions of elementary matching problems where each instance can be a finite set of pairs of terms, instead of just a single pair of terms.

The *simultaneous elementary* E-*matching problem* is the following decision problem: given a finite set $\{(s_1, t_1), \ldots, (s_k, t_k)\}$, 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, decide whether there is 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: given a finite set of pairs of terms as above, find the cardinality of a minimal complete set of E-matchers of that set. From now on, the notation $s_1 \stackrel{?}{=}_{\mathrm{E}} t_1, \ldots, s_k \stackrel{?}{=}_{\mathrm{E}} t_k$ 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 $s_1 \stackrel{?}{=}_{\mathrm{E}} t_1, \ldots, s_k \stackrel{?}{=}_{\mathrm{E}} t_k$ in the quotient algebra $\mathcal{T}(\mathcal{F}, \mathcal{X})/=_{\mathrm{E}}$, 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. 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.

3 Simultaneous Elementary AC-Matching

If $+$ is an associative and commutative binary function symbol, then every term built using $+$, variables from $\mathcal{X}$, and free constants can be brought into an equivalent *flattened* form. This means that all parentheses have been removed and all occurrences of identical variables and free constants have been grouped together using multiplicity coefficients, so that if t is a variable or a free constant, then every term of the form $t + \cdots + t$ with α summands equal to t is replaced by the expression αt. Thus, an instance of the elementary

AC-matching problem is an equation of the form

$$\alpha_1 X_1 + \cdots + \alpha_n X_n \stackrel{?}{=}_{\mathrm{AC}} \gamma_1 C_1 + \cdots + \gamma_m C_m,$$

where each X_i is a variable from $\mathcal{X}$, each C_j is a free constant, and each α_i and each γ_j is a positive integer. The *size* of such an instance of elementary AC-matching is equal to $n \max\{\alpha_1, \ldots, \alpha_n\} + m \max\{\gamma_1, \ldots, \gamma_m\}$, i.e., all integers occurring in this instance are viewed as written in *unary notation*. In other words, the size of an instance $s \stackrel{?}{=}_{\mathrm{AC}} t$ of elementary AC-matching is essentially the sum of all occurrences of variables and free constants occurring in s and t, before s and t are flattened. By the same token, the *size* of an instance $s_1 \stackrel{?}{=}_{\mathrm{AC}} t_1, \ldots, s_k \stackrel{?}{=}_{\mathrm{AC}} t_k$ of the simultaneous AC-matching problem is equal to the sum of the sizes of each instance $s_i \stackrel{?}{=}_{\mathrm{AC}} t_i$, $1 \leq i \leq k$.

It is well known that there is a close relationship between the elementary AC-matching problem and the problem of solving systems of linear Diophantine equations subject to certain additional constraints. Let $\alpha_1 X_1 + \cdots + \alpha_n X_n \stackrel{?}{=}_{\mathrm{AC}} \gamma_1 C_1 + \cdots + \gamma_m C_m$ be an instance of elementary AC-matching and assume that ρ is an AC-matcher for it. The substitution ρ assigns to each variable X_i, $1 \leq i \leq n$, a certain number (possibly zero) of copies of each constant symbol C_j, $1 \leq j \leq m$. This can be expressed formally by $X_i \mapsto x_{i1} C_1 + \cdots + x_{im} C_m$, $1 \leq i \leq n$, where each x_{ij} is an *arithmetic* variable ranging over non-negative integers and expressing how many copies of the constant symbol C_j will be assigned to the variable X_i. Let us consider now how the copies of each constant symbol C_j are distributed among the variables X_i: out of a total of γ_j copies of C_j, we have that x_{1j} copies are assigned to X_1, x_{2j} copies are assigned to X_2, and so on until x_{nj} copies are assigned to X_n. We arrive at the following system of linear Diophantine equations

$$\begin{array}{rcl} \alpha_1 x_{11} + \cdots + \alpha_n x_{n1} & = & \gamma_1 \\ \vdots \qquad\qquad\qquad \vdots & & \vdots \\ \alpha_1 x_{1m} + \cdots + \alpha_n x_{nm} & = & \gamma_m \end{array} \tag{1}$$

Note that an integer solution of the above system does not necessarily give rise to an AC-matcher of the instance $\alpha_1 X_1 + \cdots + \alpha_n X_n \stackrel{?}{=}_{\mathrm{AC}} \gamma_1 C_1 + \cdots + \gamma_m C_m$, unless it satisfies certain constraints arising from the assignments $X_i \mapsto x_{i1} C_1 + \cdots + x_{im} C_m$, $1 \leq i \leq n$. More specifically, each x_{ij} must be a non-negative integer and each variable must be assigned *at least one* copy of *at least one* of the constants symbols. Thus, there is a one-to-one and onto correspondence between AC-matchers of $\alpha_1 X_1 + \cdots + \alpha_n X_n \stackrel{?}{=}_{\mathrm{AC}} \gamma_1 C_1 + \cdots + \gamma_m C_m$ and integer solutions of the system (1) that satisfy the constraints

$$x_{ij} \quad \geq \quad 0, \quad 1 \leq i \leq n,\ 1 \leq j \leq m \tag{2}$$

$$\sum_{j=1}^{m} x_{ij} \quad \geq \quad 1, \quad 1 \leq i \leq n. \tag{3}$$

In elementary ACU-matching, constraints (3) are not necessary, because we can always assign the constant 0 of the unit axiom U to a variable. As a result, ACU-matchers correspond to solutions of the system (1) that satisfy just the non-negativity constraints (2).

Eker [Eke93] established that $\epsilon\mathrm{AC}(1, \omega)$-matching is a NP-complete problem by reducing the following 3-PARTITION problem to the elementary AC-matching problem over signatures with an unbounded number of free constants symbols.

3-PARTITION: Given a finite set $S = \{a_1, \ldots, a_{3m}\}$ with $3m$ elements, a positive integer γ, and a positive integer weight $s(a_i)$ for every $a_i \in S$ such that $\gamma/4 < s(a_i) < \gamma/2$ and $\sum_{i=1}^{3m} s(a_i) = m\gamma$, decide whether S can be partitioned into m disjoint sets $S_1, \ldots, S_m$ such that $\sum_{a \in S_j} s(a) = \gamma$ for every $j \leq m$.

Using systems of linear Diophantine equations with constraints of the form (2) and (3) as an intermediary, an instance of 3-PARTITION can be reduced to an instance $s(a_1)X_1 + \cdots + s(a_{3m})X_{3m} \stackrel{?}{=}_{\mathrm{AC}} \gamma C_1 + \cdots + \gamma C_m$ of the elementary AC-matching problem, where $C_1, \ldots, C_{3m}$ are free constant symbols. It is known that 3-PARTITION is a *strongly* NP-complete problem, which means that it remains NP-complete even when all integers occurring in it are given in unary (cf. [GJ79]). This property of 3-PARTITION is indispensable here, since the preceding reduction is in polynomial-time only when the weights $s(a_i)$, $1 \leq i \leq 3m$, and the integer γ are in unary[3]. Moreover, this reduction of 3-PARTITION to $\epsilon\mathrm{AC}(1,\omega)$-matching is parsimonious, which in turn implies that the counting problem $\#\epsilon\mathrm{AC}(1,\omega)$-matching is #P-hard. The preceding findings establish the following result.

Theorem 3.1 *$\epsilon\mathrm{AC}(1,\omega)$-matching is* NP*-complete and $\#\epsilon\mathrm{AC}(1,\omega)$-matching is* #P*-complete.*

The above result identifies the computational complexity of simultaneous elementary AC-matching for the case in which the system consists of a single equation and the signature contains an unbounded number of free constant symbols. At the other end of the classification of elementary matching problems according to the length of the system and the number of free constants, we have the case in which the systems of equations are of unbounded length and the signature contains a single free constant symbol.

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

Proof: (*Hint*) Refinement of the reduction of 1-IN-3 SAT to $\epsilon\mathrm{AC}(\omega,2)$-matching, which was given in [BS94]. □

The main result of this section shows that 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. In what follows, P stands for the class of decision problems solvable in deterministic polynomial time, while FP denotes the class of functions computable in deterministic polynomial time.

Theorem 3.3 *$\epsilon\mathrm{AC}(k,m)$-matching is in* P *and $\#\epsilon\mathrm{AC}(k,m)$-matching is in* FP*, for all $k \geq 1$ and all $m \geq 1$.*

Proof: (*Hint*) Dynamic programming algorithm for counting the solutions of systems of linear Diophantine equations (given in unary) subject to certain constraints. □

The preceding Theorems 3.1, 3.2, and 3.3 give a complete picture of the computational complexity of simultaneous elementary AC-matching problems. In the full paper, we study the complexity of ACU-matching and unveil a different picture, since 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.

[3] Benanav et al. [BKN87] state that Chandra and Kanellakis (unpublished) showed that elementary AC-matching is NP-hard by reducing the BIN PACKING problem to it. BIN PACKING is a strongly NP-complete problem that contains 3-PARTITION as a special case (cf. [GJ79]).

Table 1: Complexity Results for Simultaneous Elementary Matching

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		

Theorem 3.4 *$\epsilon\mathrm{ACU}(\omega, 1)$-matching is* NP*-complete and $\#\epsilon\mathrm{ACU}(\omega, 1)$-matching is* #P*-complete. On the other hand, $\epsilon\mathrm{ACU}(k, \omega)$-matching is in* P *and $\#\epsilon\mathrm{ACU}(k, \omega)$-matching is in* FP*, for all $k \geq 1$.*

We analyze also the complexity of elementary matching for the equational theory A of semigroups. Our findings are summarized in the following result, which shows that simultaneous elementary A-matching becomes intractable as soon as the signature $\mathcal{F}$ contains two free constant symbols, even if the system consists of a single equation.

Theorem 3.5 *$\epsilon\mathrm{A}(1, m)$-matching is* NP*-complete and $\#\epsilon\mathrm{A}(1, m)$-matching is* #P*-complete, for all $m \geq 2$. $\epsilon\mathrm{A}(\omega, 1)$-matching is* NP*-complete and $\#\epsilon\mathrm{A}(\omega, 1)$-matching is* #P*-complete. On the other hand, $\epsilon\mathrm{A}(k, 1)$-matching is in* P *and $\#\epsilon\mathrm{A}(k, 1)$-matching is in* FP*, for all $k \geq 1$.*

The results of this section are illustrated in Table 1.

4 Elementary Matching with Bounded Occurrences of Variables

Up to this point, we classified and studied simultaneous elementary matching problems by utilizing two parameters, the number of equations in a given system and the number of free constants in the signature. There is, however, a third natural parameter that often comes into play in equational matching, namely the maximum number of occurrences of variables in the instances of the matching problem under consideration. The rôle of this parameter is completely understood for the case of AC-matching over signatures containing free function symbols. Indeed, let ACi-matching be the restriction of AC-matching to instances in which each variable has at most i occurrences, where i is a positive integer. As mentioned earlier, Benanav et al. [BKN87] showed that the decision problem for AC1-matching is solvable in polynomial time, whereas Hermann and Kolaitis [HK94] proved that the counting problem #AC1-matching is #P-complete (and, hence, for every $i \geq 2$, #ACi-matching is #P-complete as well). Benanav et al. [BKN87] also pointed out that

the decision problem for AC3-matching is NP-complete, but left the complexity of AC2-matching as an open question. This was settled by Verma and Ramakrishnan [VR92], who established that the decision problem for AC2-matching is NP-complete. By exploiting the existence of free function symbols in a clever way, Verma and Ramakrishnan [VR92] showed that SAT restricted to instances in which each Boolean variable has at most three occurrences can be reduced to AC2-matching[4]. Free function symbols are also used in a crucial way when proving that #AC1-matching is #P-hard [HK94].

Next, we investigate the computational complexity of simultaneous elementary AC-matching problems in which variables have a bounded number of occurrences. This is done by carrying out a finer analysis of $\epsilon\mathrm{AC}(k,\omega)$-matching and $\epsilon\mathrm{AC}(\omega,m)$-matching, which in the previous section were shown to be the intractable cases of simultaneous elementary AC-matching. If i, k, and m are positive integers, then $\epsilon\mathrm{AC}i(k,\omega)$-matching is the restriction of $\epsilon\mathrm{AC}(k,\omega)$-matching to instances in which each variable has at most i occurrences in a given system of k equations. Similarly, we define the classes $\epsilon\mathrm{AC}i(\omega,m)$-matching, $\epsilon\mathrm{AC}i(\omega,\omega)$-matching, and the corresponding classes of counting problems.

All cases of simultaneous elementary AC1-matching turn out to be tractable. Thus, free function symbols are indispensable for showing that #AC1-matching is #P-hard.

Theorem 4.1 *$\epsilon\mathrm{AC}1(\omega,\omega)$-matching is in* P *and $\#\epsilon\mathrm{AC}1(\omega,\omega)$-matching is in* FP.

Proof: (*Hint*) For each equation, we find the number of elementary ACU-matchers using Theorem 3.4 and subtract the number of ACU-matchers violating the constraints (3). If each variable occurs once, the latter computation can be done in polynomial time. □

Recall that $\epsilon\mathrm{AC}(1,\omega)$-matching is proved NP-hard and that $\#\epsilon\mathrm{AC}(1,\omega)$-matching is proved #P-hard using a parsimonious reduction from 3-PARTITION (cf. Theorem 3.1). This reduction generates instances of AC-matching in which variables have an unbounded number of occurrences. The following result reveals that if we bound the number of occurrences of variables, then we cross the dividing line between intractability and tractability.

Theorem 4.2 *$\epsilon\mathrm{AC}i(k,\omega)$ is in* P *and $\#\epsilon\mathrm{AC}i(k,\omega)$ is in* FP*, for all $i \geq 1$ and all $k \geq 1$.*

Proof: (*Hint*) Dynamic programming algorithm for systems of linear Diophantine equations (in unary) of bounded length and with bounded coefficients. □

To complete the analysis, we consider $\epsilon\mathrm{AC}(\omega,m)$-matching with bounds on the number of occurrences of variables. It turns out that here three occurrences suffice to establish NP-hardness and #P-hardness, even over signatures with only one free constant symbol.

Theorem 4.3 *$\epsilon\mathrm{AC}3(\omega,1)$-matching is* NP*-complete and $\#\epsilon\mathrm{AC}3(\omega,1)$-matching is* #P*-complete.*

Proof: (*Hint*) Parsimonius reduction from POSITIVE 1-IN-3 SAT. □

It remains to examine simultaneous elementary AC-matching problems in which the systems are of unbounded length, but each variable has at most two occurrences. In order to analyze this case, we bring into the picture concepts and techniques from graph theory. Recall that if $G = (V, E)$ is a graph, then a *matching* is a subset M of the set E of edges such that no two edges in M have a common node, while a *complete matching*

[4]This restriction of SAT is NP-complete (cf. [GJ79]). In contrast, SAT restricted to instances in which each variable has at most two occurrences can be decided in polynomial time using resolution.

is a matching M such that every node of G is incident upon an edge in M. The following generalizations of these concepts turn out to be extremely useful here.

If $G = (V, E)$ is a graph (not necessarily a bipartite one) and $b = (b_i : i \in V)$ is a sequence of positive integers, then a *b-matching* is a subset M of E such that every node i of G is incident upon at most b_i edges in M. A *complete b-matching* is a b-matching such that every node i of G is incident upon exactly b_i edges in M. The *b-matching problem* asks: given a graph $G = (V, E)$ and a sequence $b = (b_i : i \in V)$ of positive integers, is there a complete b-matching of G? This problem has been studied extensively in the literature and efficient algorithmic solutions have been found. In particular, Edmonds and Johnson [EJ69] established that b-matching is solvable in polynomial time. Moreover, Berge [Ber73] considered the b-matching problem for *multigraphs* and showed that it has a polynomial-time reduction to the b-matching problem for graphs. We now have all the necessary tools to obtain the following result.

Theorem 4.4 *ϵAC2$(\omega, 1)$-matching is in* P*, but $\#\epsilon$AC2$(\omega, 1)$-matching is* #P*-complete.*

Proof: (*Hint*) Every instance of ϵAC2$(\omega, 1)$-matching can be transformed into a system of linear Diophantine equations such that the elements of the matrix are either 0 or 1, and each column has exactly two non-zero entries. Such a matrix can be viewed as the node-edge incidence matrix of a multigraph. As a result, every instance of the original ϵAC2$(\omega, 1)$-matching problem can be reduced in polynomial time to an instance of a b-matching problem on a multigraph. For the counting problem, use a parsimonious reduction of #PERFECT MATCHINGS [Val79] to $\#\epsilon$AC2$(\omega, 1)$-matching. □

It is an open problem to identify the exact complexity of ϵAC2(ω, m)-matching for $m \geq 2$. In this case the decision problem appears to become more difficult, because it reduces to a b-matching problem with additional *coloring* constraints. The state of affairs is clear, however, for the counting problem, since the preceding Theorem 4.4 implies that $\#\epsilon$AC2(ω, m)-matching is #P-complete for every $m \geq 2$.

In the full paper, we study also the complexity of simultaneous elementary ACU-matching problems with bounds on the number of occurrences of variables. The following result summarizes our findings.

Theorem 4.5 *ϵACU2(ω, ω)-matching is in* P*, whereas ϵACU3$(\omega, 1)$-matching is* NP*-complete. Moreover, $\#\epsilon$ACU2$(\omega, 1)$-matching is* #P*-complete.*

The results of this section are illustrated in Table 2.

We conclude by pointing out that some of our results in this section provide a partial explanation as to why all known combination algorithms for equational unification have superpolynomial worst-case behavior. Indeed, by Theorem 4.5, simultaneous elementary ACU2-matching is solvable in polynomial time, while general ACU2-matching is NP-complete [VR92]. Thus, unless P = NP, the decision problem for general ACU2-matching can not be solved via a polynomial-time algorithm that combines a decision procedure for simultaneous elementary ACU2-matching with a syntactic matching algorithm for terms in which each variable occurs at most twice. Moreover, by Theorem 4.1, simultaneous elementary #AC1-matching is in FP, while general #AC1-matching is #P-complete [HK94]. Thus, unless FP = #P, no general AC1-matching algorithm can be designed using a polynomial-time combination algorithm that combines an elementary AC1-matching algorithm with a syntactic matching algorithm for linear terms.

Table 2: Simultaneous Elementary Matching with Bounded Variable Occurrence

Simultaneous Elementary AC1-Matching & ACU1-Matching

number of equations	number of constants		
	1	$m \geq 2$	ω
$k \geq 1$	P / FP		
ω			

Simultaneous Elementary AC2-Matching

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

Simultaneous Elementary ACU2-Matching

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

Simultaneous Elementary ACi-Matching & ACUi-Matching, $i \geq 3$

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

References

[Ber73] C. Berge. *Graphs and hypergraphs*. North-Holland, Amsterdam, 2nd edition, 1973.

[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.

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

[EJ69] J. Edmonds and E.L. Johnson. Matching: A well-solved class of integer linear programs. In *Combinatorial Structures and Their Applications, Calgary (Canada)*, pages 89–92. Gordon and Breach, 1969.

[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.

[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.

[HK94] M. Hermann and P.G. Kolaitis. The complexity of counting problems in equational matching. In A. Bundy, editor, *Proceedings 12th International Conference on Automated Deduction (CADE'94), Nancy (France)*, volume 814 of *Lecture Notes in Computer Science (in Artificial Intelligence)*, pages 560–574. Springer-Verlag, June 1994.

[JK91] J.-P. Jouannaud and C. Kirchner. Solving equations in abstract algebras: A rule-based survey of unification. In J.-L. Lassez and G. Plotkin, editors, *Computational Logic. Essays in honor of Alan Robinson*, chapter 8, pages 257–321. MIT Press, Cambridge (MA, USA), 1991.

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

[Val79] L.G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8(2):189–201, 1979.

[VR92] R.M. Verma and I.V. Ramakrishnan. Tight complexity bounds for term matching problems. *Information and Computation*, 101:33–69, 1992.