The Journal of Functional and Logic Programming

The MIT Press

Volume 1997, Article 7

4 December, 1997

ISSN 1080–5230. MIT Press Journals, Five Cambridge Center, Cambridge, MA 02142-1493, USA; (617)253-2889; *journals-orders@mit.edu, journals-info @mit.edu*. Published one article at a time in LaTeX source form on the Internet. Pagination varies from copy to copy. For more information and other articles see:

- *http://www.cs.tu-berlin.de/journal/jflp/*
- *http://mitpress.mit.edu/JFLP/*
- *gopher.mit.edu*
- *ftp://mitpress.mit.edu/pub/JFLP*

©1997 Massachusetts Institute of Technology. Subscribers are licensed to use journal articles in a variety of ways, limited only as required to insure fair attribution to authors and the journal, and to prohibit use in a competing commercial product. See the journal's World Wide Web site for further details. Address inquiries to the Subsidiary Rights Manager, MIT Press Journals; (617)253-2864; *journals-rights@mit.edu*.

The Journal of Functional and Logic Programming is a peer-reviewed and electronically published scholarly journal that covers a broad scope of topics from functional and logic programming. In particular, it focuses on the integration of the functional and the logic paradigms as well as their common foundations.

Editor-in-Chief:	G. Levi	
Editorial Board:	H. Aït-Kaci	L. Augustsson
	Ch. Brzoska	J. Darlington
	Y. Guo	M. Hagiya
	M. Hanus	T. Ida
	J. Jaffar	B. Jayaraman
	M. Köhler*	A. Krall*
	H. Kuchen*	J. Launchbury
	J. Lloyd	A. Middeldorp
	D. Miller	J. J. Moreno-Navarro
	L. Naish	M. J. O'Donnell
	P. Padawitz	C. Palamidessi
	F. Pfenning	D. Plaisted
	R. Plasmeijer	U. Reddy
	M. Rodríguez-Artalejo	F. Silbermann
	P. Van Hentenryck	D. S. Warren
	* Area Editor	
Executive Board:	M. M. T. Chakravarty	A. Hallmann
	H. C. R. Lock	R. Loogen
	A. Mück	
Electronic Mail:	*jflp.request@ls5.informatik.uni-dortmund.de*	

A Minimality Study for Set Unification

Puri Arenas-Sánchez   Agostino Dovier

04 December, 1997

Abstract

A unification algorithm is said to be minimal for a unification problem if it generates exactly a (minimal) complete set of most-general unifiers, without instances, and without repetitions. The aim of this paper is to present a combinatorial minimality study for a significant collection of sample problems that can be used as benchmarks for testing any set-unification algorithm. Based on this combinatorial study, a new Set-Unification Algorithm (named SUA) is also described and proved to be minimal for all the analyzed problems. Furthermore, an existing naïve set-unification algorithm has also been tested to show its bad behavior for most of the sample problems.

1 Introduction

Many papers concerning constraint logic programming with sets (see, e.g., [DR93, Ger94, LL91]) have pointed out that the complexities of the *set-unification problem* and even of the simplest *set-matching problem* (see, e.g., [KN86, BKN87, KN92]) are the real bottlenecks of any attempt to extend logic programming with set entities. In [LL91], the problem is avoided using a delay technique: any set-unification problem is delayed until it is transformed into a simple ground "test." This improves efficiency; however, if the two terms do not become ground, obscure answers such as $\{X_1, f(X_1, X_3)\} \doteq \{Y_1, f(\{Y_3\}, X_1), X_2\}$ are returned. In [Ger94], only sets whose elements are picked from a finite domain are admitted. Also in this case, efficiency is the primary goal; nevertheless, allowed sets are very simple (for instance, nested sets are not allowed).

Set unification is a particular case of $\mathcal{T}$-unification; given an equational theory $\mathcal{T}$ and two terms s and t, four main algorithmic problems arise in $\mathcal{T}$-unification ($\mathcal{T}$-matching when t is a ground term):

1. the decision problem, i.e., to decide whether or not a substitution θ such that $s\theta \doteq_{\mathcal{T}} t\theta$ exists;
2. the counting problem, namely, the problem of computing the minimal number of independent unifiers needed to cover all the solutions of $s \doteq_{\mathcal{T}} t$;
3. to provide an algorithm that returns a complete set (when it is finite) of $\mathcal{T}$-unifiers; and
4. to provide an algorithm that returns the minimal set of unifiers of maximal generality that is also a complete set of unifiers (the cardinality of such a set is the solution to the counting problem).

There are three kinds of $\mathcal{T}$-unifications for structures akin to sets:

Boolean unification: all Boolean operators ($\cup$, $\cap$, $\setminus$, $\bar{\cdot}$, 0, 1,...) are used. If the unification problem is solvable, a unique most general unifier is sufficient (cf., e.g., [BS87]);

***ACI* unification:** an associative, commutative, and idempotent binary operator (such as $\cup$) is considered, as well as a set of constant symbols. As Boolean unification, it can be used for flat sets only, but it requires a great number of most general unifiers (cf., e.g., [Büt86, BB88]);

Set unification: the problem faced in our paper, is considered, whereby:

- Sets can be nested (e.g., $\{\emptyset, \{\emptyset\}, \{\emptyset, \{\emptyset\}\}\}$) and also variously combined with terms built with "free" functor symbols (e.g., $\{\{\emptyset, f(a, \{\emptyset\})\}\{\emptyset\}, a\}$).
- The equational theory is not *ACI* but a different theory, thus allowing us to describe unification problems different from those of the *ACI* case. Accordingly, as studied in the paper, the number of unifiers needed is fewer.

- Although the equational theory is simpler (only two axioms) than *ACI*, by dealing with nested sets, we can solve set-unification problems that cannot be expressed using *ACI* unification; for instance, $\{x, \{y, \{\emptyset, z\}\}\} = \{\{z\}, w\}$.

In [HK95], Hermann and Kolaitis analyze the counting problem for a variety of theories, including a restricted form of *ACI*. They prove that all such problems are $\#P$-complete (i.e., a counting Turing machine—a nondeterministic machine with an auxiliary output device—needs polynomial time to solve such problems).

In [BHK$^+$88] the counting problem is exactly computed for 105 benchmark problems in the theories AC and AC1. Times needed for finding matchers and unifiers for the presented problems with several algorithms are reported.

In [Büt86, BB88], the *ACI* unification problem is solved. In particular, in [BB88], Baader and Büttner elegantly solve the counting problem for *ACI* unification.

In [DOPR96] it is solved. Specifically, the problem of returning a complete set of $(A_b)(C_\ell)$-unifiers (in other words, of $\mathcal{S}et$-unifiers) is resolved for any unification problem involving terms built using a signature comprising the set constructor symbol $\{\cdot \mid \cdot\}$ ((A_b) and (C_ℓ) are the *absorption property* and the *commutativity on the left property*—see Section 2). In particular, the *finitary* nature of set unification is proved. Such an algorithm solves unification problems 1) and 3) above; nevertheless, it has bad behavior with respect to the fourth one: the computed complete set of solutions contains redundancies, namely repetitions of computed solutions as well as instances of other solutions. The algorithm we present in this work can be seen as an optimization of the algorithm of [DOPR96] in the direction of eliminating redundancies.

The contribution of this work is twofold. First, in the stream of [BHK$^+$88], we analyze in detail some significant schemata of sample problems. For these problems, we solve the counting problem by describing the smaller (minimal) set of unifiers of maximal generality. Several reasons justify the choice of such problems: their simplicity (which reflects into a simplification of the analysis); the fact that they maximize the number of solutions for unification problems of given size (the presence of distinct variables as elements of the sets to be unified guarantees the maximum number of solutions); and the fact that in using the unifiers for them we are able to describe a complete set of unifiers of any (nested) set-unification problem.

Second, we design a new general Set-Unification Algorithm (named SUA) which, being able to produce exactly the minimal set of unifiers of maximal generality for the sample presented goals, has a good expected behavior with respect to all set-unification problems. This means that in general, the set of returned solutions has very little redundancy. This fact is important if we consider that the cardinality of the minimal complete set of unifiers for the problems described in this paper (see Appendix A) is huge enough by itself.

The aim of obtaining minimal algorithms for set unification has already been treated in the literature. For instance, in [Sto96], Stolzenburg develops an interesting approach to set unification based on membership constraints. Although efficient, as noted by the same author, the latter is not minimal for some of the problems described in Section 3. Currently, we do not dispose of an implementation of SUA. However, in [Sto96], a table comparing the behaviors of several existing set-unification algorithms (including SUA) is presented. The author's results with Prolog implementations (done by himself) reflect the number and time required by the analyzed algorithms for computing the complete set of unifiers to six set-unification problems. Such a table shows that SUA is the unique algorithm able to compute the minimal complete set of most general unifiers for all problems except for one (for which SUA computes less redundancies than the rest of the algorithms). However, time results are not so good as they are desirable, mainly because SUA is more complex than the rest of the known algorithms. However, efficiency is not our primary goal here; instead, we prefer to base our efforts on the elimination of redundancies.

The paper is organized as follows. In Section 2, we comment briefly on some preliminary concepts needed in the rest of the paper. Section 3 presents the sample unification problems with their corresponding minimality studies, along with the recursive functions computing the number of solutions. Numerical values for such functions are reported in Appendix A. In Section 4, a new set-unification algorithm is presented, and is proved to terminate and to be minimal for all sample suggested problems; another existing set-unification algorithm is also tested, to check its behavior with respect to the analyzed problems. Finally, some conclusions are drawn in Section 5.

2 Preliminaries

We make use of standard *CLP* (see, e.g., [JM94]) and unification theory notations (see, e.g., [Sie90]). Given a signature Σ (a set of functional symbols together with their arities) and a denumerable set of variables $\mathcal{V}$, $T_\Sigma(\mathcal{V})$ and T_Σ will denote the sets of terms and ground terms, respectively. Letters $X, Y, Z, \ldots$ will denote variables; $a, b, c, \ldots$ will denote constant terms; $s, t, \ldots$ will denote generic terms; and $Var(t)$ will denote the set of variables occurring in t.

We require the signature Σ to contain (at least) the constant symbol $\underline{\emptyset}$, representing the empty set, and the binary symbol $\{\cdot \mid \cdot\}$, used as the set-constructor symbol; the intuitive semantics of $\{t \mid s\}$ is $\{t\} \cup s$. We will say that $t \in T_\Sigma(\mathcal{V})$ is a *set term* if t contains $\{\cdot \mid \cdot\}$ as the outermost constructor symbol. Similar to lists in Prolog, the term $\{a \mid \{b \mid \{c \mid \underline{\emptyset}\}\}\}$ will be denoted simply as $\{a, b, c\}$.

The two equational axioms

$$\begin{array}{lrcl} (A_b) & \{X, X \mid Z\} & \doteq & \{X \mid Z\} \\ (C_\ell) & \{X, Y \mid Z\} & \doteq & \{Y, X \mid Z\} \end{array}$$

called *absorption property* and *commutativity on the left property*, respectively (see, e.g., [Sie90]), uniquely identify a finest congruence $=_{\mathcal{S}et}$ on T_Σ.

Definition 1 (Substitions) *A substitution θ is a finite mapping from $\mathcal{V}$ to $T_\Sigma(\mathcal{V})$, and is written as $\theta = [X_1/t_1, \ldots, X_n/t_n]$. The notation implies that the variables $X_1, \ldots, X_n$ are different, and for $i = 1, \ldots, n$, $X_i \not\equiv t_i$. In the following, $range(\theta)$ is defined to be the set $\{t_1, \ldots, t_n\}$, whereas $dom(\theta)$ is $\{X_1, \ldots, X_n\}$. Given substitutions $\theta = [X_1/t_1, \ldots, X_n/t_n]$ and $\eta = [Y_1/s_1, \ldots, Y_m/s_m]$, the composition $\theta \circ \eta$ is defined by removing from the set $\{X_1/t_1\eta, \ldots, X_n/t_n\eta, Y_1/s_1, \ldots, Y_m/s_m\}$ those pairs $X_i/t_i\eta$ for which $X_i \equiv t_i\eta$, as well as those pairs Y_i/s_i for which $Y_i \in \{X_1, \ldots, X_n\}$.*

Definition 2 ($\mathcal{T}$-Unifier) *Given an equational theory $\mathcal{T}$, and two terms s and t, we say that $s =_\mathcal{T} t$ if s can be rewritten into t using a finite number of times the axioms of $\mathcal{T}$ (in other words, $T \vdash \vec{\forall}(s \doteq t)$). Two terms s and t are said to be $\mathcal{T}$-unifiable if and only if there is a substitution σ such that $s\sigma =_\mathcal{T} t\sigma$; such a σ is called a $\mathcal{T}$-unifier. The set of all $\mathcal{T}$-unifiers of two terms s and t is denoted by $U_\mathcal{T}(s, t)$.*

Following [FH86], given a congruence $=_{\mathcal{T}}$ on T_Σ, we write, for any set of variables $W \subseteq \mathcal{V}$, $\sigma =^W_{\mathcal{T}} \rho$ if and only if $\forall x \in W\ x\sigma =_{\mathcal{T}} x\rho$, for any two substitutions σ and ρ. In the same way, σ is *more general than* ρ *in* $\mathcal{T}$ *over* W ($\sigma \leq^W_{\mathcal{T}} \rho$) if and only if $\exists \eta\ \rho =^W_{\mathcal{T}} \sigma \circ \eta$.[1] The corresponding equivalence relation on substitutions is denoted by $\equiv^W_{\mathcal{T}}$; i.e., $\sigma \equiv^W_{\mathcal{T}} \rho$ if and only if $\sigma \leq^W_{\mathcal{T}} \rho$ and $\rho \leq^W_{\mathcal{T}} \sigma$. Two substitutions σ and ρ are said to be $\mathcal{T}$*-independent* if neither $\sigma \leq^W_{\mathcal{T}} \rho$ nor $\rho \leq^W_{\mathcal{T}} \sigma$.

It is important to notice that $\mathcal{S}et$-unification *is not ACI*-unification. For instance, by the above two definitions, we get that $\{a, X\}\theta =_{\mathcal{S}et} \{b, Y\}\theta$ for $\theta = [X/b, Y/a]$. By contrast, in [BB88] the unification problem $\{a, X\} \doteq \{b, Y\}$ admits the *ACI*- (most general) unifier $\theta' = [X/\{b, c\}, Y/\{a, c\}]$. Unifier θ' is not a $\mathcal{S}et$-unifier for the analyzed problem, since $\{a, \{b, c\}\}$ is not the same (nested) set as $\{b, \{a, c\}\}$. As a matter of fact, in [BB88] the problem $\{a, X\} \doteq \{b, Y\}$ is a syntactic sugar for the *ACI*-unification problem $X \cup \{a\} \doteq Y \cup \{b\}$. With the usual language of *ACI* (namely, variables, constants, and the *ACI* symbol $\cup$), it is not possible to write a unification problem with the same intended semantics as $\{a, X\} \doteq \{b, Y\}$.

Definition 3 (Complete Set of $\mathcal{T}$-Unifiers) *A complete set of $\mathcal{T}$-unifiers for $s, t \in T_\Sigma(\mathcal{V})$ is any subset $CSU_{\mathcal{T}}(s, t) \subseteq U_{\mathcal{T}}(s, t)$ such that for all $\sigma \in U_{\mathcal{T}}(s, t)$ there exists $\theta \in CSU_{\mathcal{T}}(s, t)$ verifying $\theta \leq^W_{\mathcal{T}} \sigma$, where $W = Var(s) \cup Var(t)$.*

Definition 4 (Complete Set of Independent $\mathcal{T}$-Unifiers) *A complete set of independent $\mathcal{T}$-unifiers for $s, t \in T_\Sigma(\mathcal{V})$ is any complete set of $\mathcal{T}$-unifiers $\mu CSU_{\mathcal{T}}(s, t)$ satisfying the condition $(\forall \sigma\tau \in \mu CSU_{\mathcal{T}}(s, t))(\sigma \neq \tau \Rightarrow \sigma \not\leq^W_{\mathcal{T}} \tau)$, where $W = Var(s) \cup Var(t)$.*

Note that if $CSU_{\mathcal{T}}(s, t)$ exists and is finite for any $s, t \in T_\Sigma(\mathcal{V})$, then $\mu CSU_{\mathcal{T}}(s, t)$ exists and is unique up to $\equiv^{\mathcal{V}}_{\mathcal{T}}$ ([FH86]).

Definition 5 ($\mathcal{T}$-Unifiable Herbrand Systems) *A Herbrand system $\mathcal{E}$ $\{t_1 \doteq s_1, \ldots, t_n \doteq s_n\}$ is $\mathcal{T}$-unifiable if and only if there is a substitution σ such that $t_i\sigma =_{\mathcal{T}} s_i\sigma$, for all $1 \leq i \leq n$. In such a case, σ is called a $\mathcal{T}$-unifier for $\mathcal{E}$. The set of all $\mathcal{T}$-unifiers of $\mathcal{E}$ is denoted by $U_{\mathcal{T}}(\mathcal{E})$.*

Similar definitions to Definitions 3 and 4 can be given starting from a Herbrand system $\mathcal{E}$ instead of the unification problem $s \doteq t$. In the following, when the context is clear, we will omit the prefix $\mathcal{T}$ before the word "unifier."

[1]Note that the empty substitution ε is more general than any other substitution.

Definition 6 (Solved Herbrand Systems) *A set of equations $\mathcal{E}$ is said to be in solved form if it has the form $\{X_1 \doteq t_1, \ldots, X_n \doteq t_n\}$ where $t_i \in T_\Sigma(\mathcal{V})$, $1 \leq i \leq n$, and each X_i is a distinct variable not occurring in t_j, for all $i, j \in \{1, \ldots, n\}$. An equation $X \doteq t$ is said to be solved in $\mathcal{E}$ if X does occur neither in t nor in $\mathcal{E} \setminus \{X \doteq t\}$.*

Note that any solved-form system $\{X_1 \doteq t_1, \ldots, X_n \doteq t_n\}$ can be viewed as the substitution $[X_1/t_1, \ldots, X_n/t_n]$.

As said in the Introduction, in unification context, the *decision problem* is the problem of deciding whether or not a system $\mathcal{E}$ is satisfiable. Given a theory $\mathcal{T}$, solving the *unification problem* means developing an algorithm such that, for any unifiable Herbrand system $\mathcal{E}$ involving terms from $T_\Sigma(\mathcal{V})$, the unification algorithm is able to compute through nondeterminism *each element* of a complete set of unifiers of $\mathcal{E}$. Notice that it is not required that it computes exactly $\mu CSU_{\mathcal{T}}(\mathcal{E})$. However, as we will see in detail in Section 3, the presented theory $\mathcal{S}et$ is such that, even for simple unification problems $s \doteq t$, $\mu CSU_{\mathcal{S}et}(s,t)$ becomes larger and larger. This means that a valid criterion for comparing two $\mathcal{S}et$-unification algorithms is the analysis of the length of the *list* of solutions computed by them (the word "list" here is used to reflect the fact that if a unification algorithm computes exactly $\mu CSU_{\mathcal{T}}(s,t)$, but some solution is returned more than once, it cannot be considered minimal).

If the input system $\mathcal{E}$ is not $\mathcal{T}$-unifiable, any unification algorithm should conclude its computation reporting a *failure* result.

Definition 7 (Minimal Algorithm) *A unification algorithm for a theory $\mathcal{T}$ is said to be minimal for a unification problem $t_1 \doteq t_2$, if the algorithm returns exactly a minimal complete set of independent unifiers for it. The algorithm is said to be minimal for the theory $\mathcal{T}$ if it is minimal for all unification problems written in the language of the theory $\mathcal{T}$.*

For instance, the Robinson unification algorithm is minimal for the empty equational theory (it returns a unique—most general—unifier, or failure).

3 A Case Analysis

Experience in programming with sets teaches us that the number of unifiers returned by a set-unification algorithm on a common input is, in general, very large. Therefore, one of the main goals of a unification algorithm is to reduce, as much as possible, redundancies in the set of returned solutions. Clearly, the optimal situation is that a unification algorithm for a theory $\mathcal{T}$ is *minimal* for it (cf. Definition 7), namely, that it returns exactly a minimal complete set of independent unifiers, without repetitions, without instances, for any unification problem.

Given a unification problem between two sets[2] it is possible, in principle, to determine the number of most-general and independent unifiers for it (i.e., the cardinality of $\mu CSU_{\mathcal{T}}(s,t)$). Nevertheless, it is impossible to experimentally test the *minimality* (namely, the capability of returning exactly $\mid \mu CSU_{\mathcal{T}}(s,t) \mid$ solutions) for all (infinite) problems $s \doteq t$ with s, t in $T_\Sigma(\mathcal{V})$. We will propose a number of (schemata of) reasonable sample goals on which the minimality of an algorithm can be tested. Before presenting our election, note that any set unification problem in our framework responds to one of the following cases:

$$\begin{array}{lrcl}
(a) & \{t_1, \ldots, t_m\} & \doteq & \{s_1, \ldots, s_n\}, \\
(b) & \{t_1, \ldots, t_m \,|\, t\} & \doteq & \{s_1, \ldots, s_n\}, \\
(c) & \{t_1, \ldots, t_m \,|\, t\} & \doteq & \{s_1, \ldots, s_n \,|\, t\}, \\
(d) & \{t_1, \ldots, t_m \,|\, t\} & \doteq & \{s_1, \ldots, s_n \,|\, s\},
\end{array}$$

where $t_i, s_j, s, t \in T_\Sigma(\mathcal{V})$, $1 \leq i \leq m$, $1 \leq j \leq n$, and $s \neq t$. Moreover, since repetitions of elements inside a set are irrelevant (as Axiom (A_b) in Section 2 establishes), we can assume without loss of generality that $t_1, \ldots, t_m$ (respectively, $s_1, \ldots, s_n$) are pairwise-distinct terms. Thus, our first four problems are:

$$\begin{array}{lrcl}
(1) & \{X_1, \ldots, X_m\} & \doteq & \{Y_1, \ldots, Y_n\}, \\
(2) & \{X_1, \ldots, X_m \,|\, Z\} & \doteq & \{Y_1, \ldots, Y_n\}, \\
(3) & \{X_1, \ldots, X_m \,|\, Z\} & \doteq & \{Y_1, \ldots, Y_n \,|\, Z\}, \\
(4) & \{X_1, \ldots, X_m \,|\, W\} & \doteq & \{Y_1, \ldots, Y_n \,|\, Z\},
\end{array}$$

where $X_1, \ldots, X_m, Y_1, \ldots, Y_n, W$, and Z are pairwise-distinct variables. Of course, any problem of type (a)–(d) can be viewed as an instance of Problem (1)–(4), respectively. Furthermore, the presence of distinct variables as elements of the sets guarantees the maximum number of solutions, and in finding the unifiers for them, we are also able to describe a complete set of

[2]e.g., $\{\{X, f(g(Y)) \,|\, R\}, f(X)\} \doteq \{A, \{B, f(C)\} \,|\, R\}$.

unifiers of *any* (nested) set-unification problem. For instance, as we will see in detail, any solution θ to $\{X_1, \ldots, X_m\} \doteq \{Y_1, \ldots, Y_n\}$ consists of a number of mappings of the form X_i/Y_j or Y_j/X_i. Such a solution can be seen as a *template* useful for generating a number of equations between the elements $s_1, \ldots, s_m$ and $t_1, \ldots, t_n$ of the two generic sets forming the equation $\{s_1, \ldots, s_m\} \doteq \{t_1, \ldots, t_n\}$. More precisely, the latter unification problem can be reduced to the conjunction

$$\bigwedge_{X_i/Y_j \; in \; \theta \vee Y_j/X_i \; in \; \theta} s_i \doteq t_j$$

Since the four problem schemata (1)–(4) cover all possible *set-to-set* equation cases, they are sufficient to implement the inductive step of a unification algorithm dealing with sets having a *list* representation. It is very important that no redundancy is introduced by the unification algorithm in solving it: therefore, we will first analyze the problem schemata in detail from a combinatorial point of view, to characterize the number of unifiers needed.

We see, in particular, that an easy upper bound to the number of nonredundant unifiers for all such problems is $(4(m+n))^{m+n}$, while the simple *ACI* set-matching problem

$$X_1 \cup \ldots \cup X_m \doteq \{a_1, \ldots, a_n\}$$

with $X_1, \ldots, X_m$ pairwise-distinct variables and $a_1, \ldots, a_n$ pairwise-distinct constants, admits $(2^m - 1)^n$ independent solutions.[3] Assume $m = n$; then

$$\lim_{n \to \infty} \frac{(4(n+n))^{n+n}}{(2^n - 1)^n} = 0$$

This is one of the reasons for choosing the list-style representation of sets; another reason is the simplicity of the hybrid and nested set axiomatization (cf. [DPR96]).

We also analyze particular cases of Problems (1)–(4) when one of the two sets contains ground elements (that, without loss of generality, we may assume distinct constants). The problems are the following (the first two

[3]The solutions are exactly the functions between the set $\{a_1, \ldots, a_n\}$ and the set of the regions of the Venn diagram individualized by $X_1, \ldots, X_m$, save $\overline{X_1 \cup \ldots \cup X_m}$.

cases are *matching* problems):

$$\begin{array}{lrcl}
(1') & \{X_1,\ldots,X_m\} & \doteq & \{a_1,\ldots,a_n\}\,, \\
(2') & \{X_1,\ldots,X_m\,|\,Z\} & \doteq & \{a_1,\ldots,a_n\}\,, \\
(3') & \{X_1,\ldots,X_m\,|\,Z\} & \doteq & \{a_1,\ldots,a_n\,|\,Z\}\,, \\
(4') & \{X_1,\ldots,X_m\,|\,W\} & \doteq & \{a_1,\ldots,a_n\,|\,Z\}\,,
\end{array}$$

where $X_1,\ldots,X_m,W$, and Z are pairwise-distinct variables, and $a_1,\ldots,\ a_n$ are pairwise-distinct constants. As shown in the tables presented in Appendix A, the cardinality of $\mu CSU_{\mathcal{S}et}$ for Problems (1′)–(4′) is much smaller than that for Problems (1)–(4), respectively.

As a particular case, it is interesting to analyze the cleverness of an algorithm in solving the problem in which the two sets share elements. We will analyze the cases

$$\begin{array}{lrcl}
(5) & \{X_1,\ldots,X_m,Z_1,\ldots,Z_k\} & \doteq & \{Y_1,\ldots,Y_n,Z_1,\ldots,Z_k\} \\
(5') & \{X_1,\ldots,X_m,a_1,\ldots,a_k\} & \doteq & \{Y_1,\ldots,Y_n,a_1,\ldots,a_k\}
\end{array}$$

where $X_1,\ldots,X_m,Y_1,\ldots,Y_n$, and $Z_1,\ldots,Z_k$ are pairwise-distinct variables, and $a_1,\ldots,\ a_n$ are pairwise-distinct constant symbols. As we will show, while 2,945 solutions are needed to $\{X_1,\ldots,X_5\}\doteq\{Y_1,\ldots,Y_5\}$, only 56 are necessary (and sufficient) to describe all the solutions to

$$\{X_1,X_2,Z_1,Z_2,Z_3\}\doteq\{Y_1,Y_2,Z_1,Z_2,Z_3\}$$

Below we describe the functions that compute the cardinality of $\mu CSU_{\mathcal{S}et}$ for Problems (1)–(5) as well as for the simpler Problems (1′)–(5′). Tables reporting some of their values are presented in Appendix A.

3.1 Problem (1)

We first analyze the simple matching Problem (1′). The number of solutions to the problem

$$(1') \qquad \{X_1,\ldots,X_m\} \doteq \{a_1,\ldots,a_n\}$$

is exactly the number of surjective applications from a set of m elements onto a set of n elements. Such a number is denoted as $Surj(m,n)$, and it is clearly bounded by n^m, the number of all the functions from a set of m elements to a set of n elements. We analyze how to compute the function $Surj$:

- if $m < n$, then $Surj(m, n) = 0$;
- if $m \geq n$ and $m = n = 0$, then $Surj(m, n) = 1$; if $m > 0$ and $n = 0$, then $Surj(m, n) = 0$. If $n = 1$, then $Surj(m, n) = 1$; if $m, n > 1$, any surjective function $g : \{X_1, \ldots, X_m\} \longrightarrow \{a_1, \ldots, a_n\}$ can be obtained by:
  - extending a surjective function, $g : \{X_1, \ldots, X_{m-1}\} \longrightarrow \{a_1, \ldots, a_n\}$, so as to fulfill $g(X_m) = a_i$ (there are n possibilities for choosing such an i); or
  - extending uniquely a surjective function, $g : \{X_1, \ldots, X_{m-1}\} \longrightarrow \{a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n\}$, so as to fulfill $g(X_m) = a_i$ (there are n possibilities for choosing such an i).

This suggests the following recursive definition for the function *Surj*:

$$\begin{cases} Surj(m,n) & = & 0 & \text{if } m < n \\ Surj(m,1) & = & 1 & \text{if } 1 \leq m \\ Surj(m,n) & = & n \left(\begin{array}{c} Surj(m-1,n-1) + \\ Surj(m-1,n) \end{array} \right) & \text{if } 1 < n \leq m \end{cases}$$

A more compact description of $Surj(m, n)$ can be given using Stirling numbers of the second type (see, for instance, [Knu68]): $\left\{ {m \atop k} \right\}$ is the number of ways to partition a set of m elements into k nonempty disjoint subsets. Any surjective function $g : \{X_1, \ldots, X_m\} \longrightarrow \{a_1, \ldots, a_n\}$ can be obtained by mapping (with a bijection) an n-partition of $\{X_1, \ldots, X_m\}$ onto the set $\{a_1, \ldots, a_n\}$. Thus,

$$Surj(m, n) = n! \left\{ {m \atop n} \right\}$$

However, the recursive nature of the definition is not removed, but only hidden into the definition of the Stirling number.

Now we describe a function $\Phi : \omega^2 \longrightarrow \omega$, which computes the number of most general and independent solutions to the problem

$$(1) \qquad \{X_1, \ldots, X_m\} \doteq \{Y_1, \ldots, Y_n\}$$

(notice that it admits a solution even if $m < n$):

- if $m = 0$ and $n = 0$, then $\Phi(m, n) = 1$ (the most general solution is the substitution ε);
- if $m > 0$ and $n = 0$, or $m = 0$ and $n > 0$, then $\Phi(m, n) = 0$;
- if we assume $m, n > 0$, and if $m = 1$ or $n = 1$, then $\Phi(m, n) = 1$. Letting $m, n > 1$, we analyze the form of a solution θ. Fix an element in the first set, say X_m. Two cases can appear:

  (a) θ binds X_m to all the elements of a proper subset S of $\{X_1, \ldots, X_{m-1}\}$ (possibly empty) and to exactly one element Y_i, for some $i = 1, \ldots, n$ (n ways).[4] The remaining part of θ is a solution to the subproblem

  $$\{X_1, \ldots, X_{m-1}\} \setminus S \doteq \{Y_1, \ldots, Y_{i-1}, Y_{i+1}, \ldots, Y_n\}$$

  (b) θ binds X_m to all the elements of a proper subset T of $\{Y_1, \ldots, Y_n\}$ such that $|T| \geq 2$ (if $|T| = 1$, then it is one of the cases analyzed in the case above). The remaining part of θ is a solution to the subproblem

  $$\{X_1, \ldots, X_{m-1}\} \doteq \{Y_1, \ldots, Y_n\} \setminus T$$

Thus, for $m, n > 0$ the function Φ is recursively described as follows:

$$\begin{cases} \Phi(m,1) & = & 1 & m \geq 1 \\ \Phi(1,n) & = & 1 & n > 1 \\ \Phi(m,n) & = & n \sum_{i=0}^{m-2} \binom{m-1}{i} \Phi(m-1-i, n-1) + & m, n > 1 \\ & & \sum_{j=2}^{n-1} \binom{n}{j} \Phi(m-1, n-j) & \end{cases}$$

Theorem 1 *The set of substitutions counted by Φ constitutes a minimal complete set of independent $\mathcal{S}et$-unifiers to Problem* (1)*.*

Proof of Theorem 1 By Axioms (A_b) and (C_ℓ), and Definition 2, μ is a unifier to Problem (1) if (and only if)

$$\begin{array}{lllll} (*) & \forall i \in \{1, \ldots, m\} & \exists j \in \{1, \ldots, n\} & (X_i\mu = Y_j\mu) & \text{and} \\ & \forall j \in \{1, \ldots, n\} & \exists i \in \{1, \ldots, m\} & (X_i\mu = Y_j\mu) & \end{array}$$

[4]Throughout the paper, the verb "to bind" is used with a bidirectional meaning: if X is bound to Y, also Y is bound to X.

It is easy (but tedious) to see that all θs counted by Φ fulfill the formula $(*)$ (thus, they are all unifiers), and that they are pairwise independent. It remains to prove that they are sufficient to cover all possible solutions. We prove this fact by induction on the pair $\langle m, n\rangle$, with $m, n \geq 1$. If $m = 1$ or $n = 1$, then the result is trivial. Let μ be a unifier to Problem (1) and both $m > 1$ and $n > 1$.

Consider the element X_m. Since μ is a unifier, then there is $j \in \{1, \ldots, n\}$ such that $X_m\mu = Y_j\mu$. Two cases must be considered:

1. Assume that for all $k \in \{1, \ldots, j-1, j+1, \ldots, n\}$, $Y_k\mu \neq Y_j\mu$. Let S be the maximal subset of $\{X_1, \ldots, X_{m-1}\}$ such that for all $X \in S$, $X\mu = X_m\mu = Y_j\mu$. By $(*)$ it is clear that μ is a unifier to the subproblem:

   $$(p)\ \{X_1, \ldots, X_{m-1}\} \setminus S \doteq \{Y_1, \ldots, Y_{j-1}, Y_{j+1}, \ldots, Y_n\}$$

   By induction hypothesis, there is a unifier θ (counted by Φ) to Problem (p) such that $\theta \leq^W_{\mathcal{S}et} \mu$, where W is the set of variables occurring in (p). Considering Rule (a) in the construction of Φ, we get $\theta' = \theta \cup [X/Y_j \mid X \in S \cup \{X_m\}]$, a unifier to Problem (1) verifying $\theta' \leq^W_{\mathcal{S}et} \mu$, where $W' = W \cup \{X_m\} \cup S$.

2. Otherwise, let T be the maximal subset of $\{Y_1, \ldots, Y_n\}$ such that $X_m\mu = Y\mu$ for all $Y \in T$. If $T = \{Y_j\}$, we are in the previous case. Assume $|\,T\,| > 1$. Let S be the maximal subset of $\{X_1, \ldots, X_{m-1}\}$ such that for all $X \in S$, $X_m\mu = X\mu$.

   - If S is assumed empty, the proof follows trivially from $(*)$ and induction hypothesis, using Rule (b) above.
   - If we assume S is not empty, by $(*)$, μ is also a unifier to the subproblem:

     $$\{X_1, \ldots, X_{m-1}\} \doteq \{Y_1, \ldots, Y_{j-1}, Y_{j+1}, \ldots, Y_n\}$$

     By induction hypothesis, a unifier θ more general than the restriction of μ to the variables $X_1, \ldots, X_{m-1}, Y_1, \ldots, Y_{j-1}$, and $Y_{j+1}, \ldots, Y_n$ is counted by Φ. Thus, the substitution $\theta \cup [X_m/Y_j]$, computed by Rule (a), is more general than μ.

Proof of Theorem 1 $\square$

The following alternative analysis of Φ will be useful for the study of Problems (2) and (4), and for easily proving that $\Phi(m,n) \leq m^n \cdot n^m$; any most general unifier θ of $\{X_1,\ldots,X_m\} \doteq \{Y_1,\ldots,Y_n\}$ binds X-variables with Y-variables in one of the following ways:

(a') $X_i\theta = Y_j\theta$, and $X_h\theta$ and $Y_k\theta$ are all distinct from $X_i\theta$ for all $h = 1,\ldots,m$, $h \neq i$ and for all $k = 1,\ldots,n$, $k \neq j$. In such a case, we say that X_i/Y_j is a *simple binding* in the solution θ.

(b') There are $j_1,\ldots,j_q$, $q \geq 2$, such that $X_i\theta = Y_{j_1}\theta = \ldots = Y_{j_q}\theta$, and for all $h = 1,\ldots,m$, $h \neq i$, and for all $k = 1,\ldots,n$, $k \neq j_1,\ldots,k \neq j_q$, and $X_h\theta$ and $Y_k\theta$ are all distinct from $X_i\theta$. In this case, we say there is a *q-fork* in the solution θ.

(c') There are $i_1,\ldots,i_p$, $p \geq 2$, such that $X_{i_1}\theta = \ldots = X_{i_p}\theta = Y_j\theta$, and for all $h = 1,\ldots,m$, $h \neq i_1,\ldots,h \neq i_p$, and for all $k = 1,\ldots,n$, $k \neq j$, and $X_h\theta$ and $Y_k\theta$ are all distinct from $X_{i_1}\theta$. In this case, we say there is a *p-cone* in the solution θ.

The next example will clarify the notions of q-fork and p-cone defined above.

Example 1 *Consider the following two most general unifiers:*

$$\begin{aligned} \theta &= [X_1/Y_1, X_2/Y_2, X_3/Y_3] \\ \theta' &= [X_1/Y_1, Y_2/Y_1, X_2/Y_3, X_3/Y_3] \end{aligned}$$

to the problem $\{X_1, X_2, X_3\} \doteq \{Y_1, Y_2, Y_3\}$.

While θ is composed of three simple bindings, θ' contains a 2-fork composed of $X_1\theta' = Y_1\theta' = Y_2\theta'$ and a 2-cone consisting in $X_2\theta' = X_3\theta' = Y_3\theta'$. Graphically, θ and θ' can be represented as follows:

$$\theta \equiv \begin{pmatrix} X_1 & \xrightarrow{\textit{simple binding}} & Y_1 \\ X_2 & \xrightarrow{\textit{simple binding}} & Y_2 \\ X_3 & \xrightarrow{\textit{simple binding}} & Y_3 \end{pmatrix} \quad \theta' \equiv \begin{pmatrix} X_1 & \xrightarrow{\textit{2-fork}} & \left\{ \begin{array}{l} Y_1 \\ Y_2 \end{array} \right. \\ \left. \begin{array}{l} X_2 \\ X_3 \end{array} \right\} & \xrightarrow{\textit{2-cone}} & Y_3 \end{pmatrix}$$

Below we will describe a procedure for counting all such solutions.

Given $n > 0$, an n-tuple $c \equiv [i_1,\ldots,i_n]$ is said to be a *configuration* for the set $\{Y_1,\ldots,Y_n\}$ if the non-negative integers $i_1,\ldots,i_n$ are such that $\sum_{j=1}^{n} i_j \cdot j = n$.

Let k $(1 \leq k \leq n)$ be $\sum_{j=1}^{n} i_j$; the configuration c is a witness of any partition of $\{Y_1, \ldots, Y_n\}$ into k nonempty disjoint subsets such that there are exactly i_1 singleton subsets, i_2 doubleton subsets, and so on. Let $\mathcal{C}_n$ be the set of all the configurations for a fixed n; for instance, when $n = 6$, the following 11 configurations are possible:

i_1	i_2	i_3	i_4	i_5	i_6
6	0	0	0	0	0
4	1	0	0	0	0
3	0	1	0	0	0
2	0	0	1	0	0
1	0	0	0	1	0
0	0	0	0	0	1

i_1	i_2	i_3	i_4	i_5	i_6
2	2	0	0	0	0
0	3	0	0	0	0
0	0	2	0	0	0
1	1	1	0	0	0
0	1	0	1	0	0

It is easy to write a Prolog program that computes the set $\mathcal{C}_n$ for a given n (see Figure 1).

Let $\|[i_1, \ldots, i_n]\|$ be the number of partitions of $\{Y_1, \ldots, Y_n\}$ of the form uniquely determined by $[i_1, \ldots, i_n]$. Clearly,

$$\sum_{\substack{i_1+\ldots+i_n=k \\ 1\cdot i_1+\ldots+n\cdot i_n=n}} \|[i_1, \ldots, i_n]\| = \left\{ {n \atop k} \right\}$$

The following simple example clarifies how to compute $\|[i_1, \ldots, i_n]\|$; let $n = 16$, and $c \equiv [5, 4, 1, 0, \ldots, 0]$. There are $\binom{16}{2}$ ways for selecting the first doubleton, and $\binom{14}{2}$, $\binom{12}{2}$, and $\binom{10}{2}$ ways for selecting the second, the third, and the fourth doubletons, respectively. In this way, however, any partition will be counted $4!$ $(= i_2!)$ times, since there is no difference in selecting two elements Y_i and Y_j in the first, second, third, or fourth attempts. There are still $\binom{8}{3}$ ways for selecting the unique subset of three elements. The remaining elements will constitute the singletons of the partition. Thus,

$$\|c\| = \frac{\binom{16}{2}\binom{14}{2}\binom{12}{2}\binom{10}{2}}{4!} \cdot \frac{\binom{8}{3}}{1!} = \frac{16!}{((2!)^4(3!)^1)(5!4!1!)}$$

Such a situation is easy to generalize so as to obtain:

$$\|[i_1, \ldots, i_n]\| = \frac{n!}{\Pi_{j=2}^{n}(j!)^{i_j} \cdot \Pi_{j=1}^{n}(i_j!)}$$

```
configurations(N, Cn) :-
    setof(C, conf(N, 1, N, C), Cn).
conf(0, _, M, []) :-
    !, M = 0.
conf(N, _, 0, C) :-
    !, zerolist(N, C).
conf(N, W, M, [A | C]) :-
    T is (M div W), in(A, T),
    M1 is M - W * A,
    N1 is N - 1, W1 is W + 1,
    conf(N1, W1, M1, C).

in(T, T).
in(A, T) :-
    T > 0, T1 is T - 1,
    in(A, T1).
zerolist(0, []) :- !.
zerolist(N, [0 | R]) :-
    M is N - 1,
    zerolist(M, R).
```

Figure 1: The Prolog program-generating configurations.

Assume a partition S of $\{Y_1, \ldots, Y_n\}$ $(S \subseteq \mathcal{P}(\{Y_1, \ldots, Y_n\}))$ has the configuration $c \equiv [i_1, \ldots, i_n]$. We want to compute the number of (most general and independent) solutions θ to Problem (1) such that S is the smallest set fulfilling $S = \{Z : (\forall Y_i Y_j \in Z)(Y_i\theta = Y_j\theta)\}$.

Figure 2 illustrates the form of any such a θ, including:

- subsets of $\{X_1, \ldots, X_m\}$ consisting of $i_2, i_3, \ldots, i_n$ elements are selected;
- for any $j = 2, \ldots, n$, θ binds with a bijection the subset identified by i_j to the subset of S constituted by sets of exactly j elements; and
- moreover, the remaining $m - \sum_{j=2}^{n} i_j$ elements of $\{X_1, \ldots, X_m\}$ are connected by a surjection to the remaining i_1 elements of $\{Y_1, \ldots, Y_n\}$.

Thus, there are:

$$\binom{m}{i_2}\binom{m - i_2}{i_3} \cdots \binom{m - \sum_{j=2}^{n-1} i_j}{i_1}$$

ways for choosing the elements reflecting the situation described; once they are fixed, there are:

$$\mathit{Surj}(i_2, i_2) \ldots \mathit{Surj}(i_n, i_n)\mathit{Surj}\Big(m - \sum_{j=2}^{n} i_j, i_1\Big)$$

$$\begin{array}{rcl}
\underbrace{X_1,\ldots,X_m} & & \underbrace{Y_1,\ldots,Y_n} \\
i_2\left\{\begin{array}{c}\circ\\ \vdots\\ \circ\end{array}\right. & \begin{array}{c}\longrightarrow\\ Surj(i_2,i_2)\\ \longrightarrow\end{array} & \left.\begin{array}{l}\circ\circ\\ \vdots\\ \circ\circ\end{array}\right\} i_2 \\
i_3\left\{\begin{array}{c}\circ\\ \vdots\\ \circ\end{array}\right. & \begin{array}{c}\longrightarrow\\ Surj(i_3,i_3)\\ \longrightarrow\end{array} & \left.\begin{array}{l}\circ\circ\circ\\ \vdots\\ \circ\circ\circ\end{array}\right\} i_3 \\
\vdots & \vdots & \vdots \\
i_n\left\{\begin{array}{c}\circ\\ \vdots\\ \circ\end{array}\right. & \begin{array}{c}\longrightarrow\\ Surj(i_n,i_n)\\ \longrightarrow\end{array} & \left.\begin{array}{l}\circ\ldots\circ\\ \vdots\\ \underbrace{\circ\ldots\circ}_{n}\end{array}\right\} i_n \\
m-\sum_{j=2}^{n} i_j\left\{\begin{array}{c}\circ\\ \vdots\\ \circ\end{array}\right. & \begin{array}{c}\longrightarrow\\ Surj\left(m-\sum_{j=2}^{n} i_j, i_1\right)\\ \longrightarrow\end{array} & \left.\begin{array}{l}\circ\\ \vdots\\ \circ\end{array}\right\} i_1
\end{array}$$

Figure 2: The form of a generic solution.

possible situations. Hence, the number of most general unifiers we are looking for is:

$$\frac{m!}{\left(m-\sum_{j=2}^{n} i_j\right)!} \cdot \mathit{Surj}\left(m-\sum_{j=2}^{n} i_j, i_1\right) = \frac{m! \cdot i_1!}{(m-\sum_{j=2}^{n} i_j)!} \cdot \left\{ {m-\sum_{j=2}^{n} i_j \atop i_1} \right\}$$

Finally, given a configuration $c \equiv [i_1, \ldots, i_n]$, the number of possible most general unifiers from the set $\{X_1, \ldots, X_m\}$ to any partition of $\{Y_1, \ldots, Y_n\}$ having configuration c (we call this number $\|c\|^m$) will be:

$$\begin{aligned} \|[i_1, \ldots, i_n]\|^m &= \|[i_1, \ldots, i_n]\| \cdot \frac{m! \cdot i_1!}{\left(m-\sum_{j=2}^{n} i_j\right)!} \cdot \left\{ {m-\sum_{j=2}^{n} i_j \atop i_1} \right\} \\ &= \frac{m! \cdot n! \cdot i_1!}{\left(\Pi_{j=2}^{n} (j!)^{i_j}\right) \cdot \left(\Pi_{j=1}^{n} (i_j!)\right) \cdot \left(m-\sum_{j=2}^{n} i_j\right)!} \cdot \left\{ {m-\sum_{j=2}^{n} i_j \atop i_1} \right\} \end{aligned}$$

With this alternative point of view, the function Φ can be defined as:

$$\Phi(m, n) = \sum_{c \in \mathcal{C}_n} \|c\|^m$$

We end this subsection by showing an upper bound for the function Φ:

Lemma 1 *For any $m, n > 0$, $\Phi(m, n) \leq m^n \cdot n^m$.*

Proof of Lemma 1 Any solution θ to a unification problem of Type (1) is composed of two parts:

- a part that describes a surjective function from a subset S of $\{X_1, \ldots, X_m\}$ to a subset T of $\{Y_1, \ldots, Y_n\}$; and
- the remaining part, which describes a surjective function from $\{Y_1, \ldots, Y_n\} \setminus T$ to $\{X_1, \ldots, X_m\} \setminus S$.

The first part is bounded by n^m; the second part is bounded by m^n.

Proof of Lemma 1 □

In [Sto96], an interesting approach to finding the solution to Problem (1) using Taylor's series is presented. In particular, it is implicitly proved that:

$$\Phi(m, n) = (\Delta_x^m \Delta_y^n e^{(x(e^y-1)+y(e^x-1)-xy)})_{\langle 0,0 \rangle}$$

where $\Delta_v^k f(\ldots, v, \ldots)$ means to derive k times with respect to the variable v.

3.2 Problem (2)

Before facing the analysis of the solutions to the problem

$$(2) \qquad \{X_1, \ldots, X_m \,|\, Z\} \;\doteq\; \{Y_1, \ldots, Y_n\}$$

we briefly discuss how to compute a solution to the corresponding matching Problem (2′):

$$(2') \qquad \{X_1, \ldots, X_m \,|\, Z\} \;\doteq\; \{a_1, \ldots, a_n\}$$

Let S be a nonempty subset of $\{a_1, \ldots, a_n\}$, and θ_1 be a solution to the problem of Type (1′) $\{X_1, \ldots, X_m\} \doteq S$; then any substitution θ extending θ_1 with the mapping $Z/(\{a_1, \ldots, a_n\} \setminus S) \cup T$, for any $T \subseteq S$, is a solution to Problem (2′) above.

It is easy to see that all the solutions obtained in this way are independent and, furthermore, the collection of them is a complete set of unifiers. The total number of solutions to Problem (2′) is therefore:

$$\sum_{i=1}^{n} \binom{n}{i} \Big(\mathit{Surj}(m, i) \cdot 2^i \Big)$$

For the more general case of Problem (2), if S is a nonempty subset of $\{Y_1, \ldots, Y_n\}$, and θ_1 is a solution to the problem of Type (1) $\{X_1, \ldots, X_m\} \doteq S$, then any substitution θ extending θ_1 with the mapping $Z/(\{Y_1, \ldots, Y_n\} \setminus S) \cup T$ is a solution to Problem (2), for any $T \subseteq S$. However, in this case, solutions are not all pairwise independent, as the following example shows.

Example 2 *Consider the unification problem of Type (2): $\{X_1 \,|\, Z\} \doteq \{Y_1, Y_2\}$:*

- *If $S = \{Y_1, Y_2\}$, then $\{X_1\} \doteq \{Y_1, Y_2\}$ has the solution $[Y_1/X_1, Y_2/X_1]$. Such a (unique) solution can be extended with $[Z/\{Y_2\}]$ (by choosing $T = \{Y_2\}$).*
- *If $S = \{Y_1\}$, then the problem $\{X_1\} \doteq \{Y_1\}$ has the unique solution $[X_1/Y_1]$. Such a solution can be extended with $[Z/\{Y_2\}]$ (by choosing $T = \emptyset$), a more general solution than the first presented.*

A closer analysis of the solutions to the problem $\{X_1, \ldots, X_m\} \doteq S$ must be performed to identify whether a solution θ—obtained by extending it with a substitution of the form $[Z/(\{Y_1, \ldots, Y_n\} \setminus S) \cup T]$, with $T \subseteq S$—is general or not. Assume an element Y_i of S belongs to T. Two cases are possible:

- θ does not bind Y_i to any Y_j, for $j = 1, \ldots, n$, $i \neq j$. In other words:

$$\begin{array}{ll} X_{i_1}\theta = \ldots = X_{i_k}\theta = Y_i\theta & \text{for some } i_1, \ldots, i_k \in \{1, \ldots, m\},\ k \geq 1 \\ Y_i\theta \neq Y_j\theta & \text{for } j = 1, \ldots, n,\ i \neq j \end{array}$$

- θ binds Y_i to some Y_js, for $j = 1, \ldots, n$, $i \neq j$. This means (see Problem (1)) that there is a $(k+1)$-fork in the solution θ of the form:

$X_h\theta = Y_i\theta = Y_{i_1}\theta = \ldots = Y_{i_k}\theta$
for exactly one $h \in \{1, \ldots, m\}$ and for some $i_1, \ldots, i_k \in \{1, \ldots, n\}$.

In the former case, we can insert Y_i into T. As a matter of fact, if we consider the subproblem obtained by removing it from S, there are no possibilities to map $X_{i_1}, \ldots, X_{i_k}$ in a way that subsumes the solution.

In the latter case, the situation is radically different. Note that θ has the following structure: $\theta = [X_h/Y_i, Y_{i_1}/Y_i, \ldots, Y_{i_k}/Y_i] \cup \theta^* \cup [Z/T']$, where $Y_i \in T'$. If we consider the subproblem obtained by removing Y_i from S, we could obtain a solution $\theta' = [X_h/Y_{i_1}, Y_{i_2}/Y_{i_1}, \ldots, Y_{i_k}/Y_{i_1}] \cup \theta^* \cup [Z/T']$, which is more general than the previous one. This means that if we insert Y_i into T, we generate an instance of a solution already computed. An analogous reasoning can be performed for any of $Y_{i_1}, \ldots, Y_{i_k}$ and for all k-forks in θ. Thus, the number of such solutions can be computed by means of:

$$\sum_{j=1}^{n} \binom{n}{j} \Phi'(m, j)$$

where Φ' has the same structure as Φ (see Problem (1)), but now, considering the possible extensions of Z with Y-variables occurring either in a simple binding or in a cone. Therefore:

$$\begin{cases} \Phi'(m,n) & = & 0 & (m > 0, n = 0) \text{ or } (m = 0, n > 0) \\ \Phi'(m,1) & = & 2 & (m \geq 1) \\ \Phi'(1,n) & = & 1 & (n > 1) \\ \Phi'(m,n) & = & 2n \sum_{i=0}^{m-2} \binom{m-1}{i} \Phi'(m-1-i, n-1) + & m, n > 1 \\ & & \sum_{j=2}^{n-1} \binom{n}{j} \Phi'(m-1, n-j) & \end{cases}$$

The factor 2 represents the possibility of extending Z with the selected Y-variable of the second set, in the generation of a cone or a simple binding.

Another interesting but equivalent way for counting the number of solutions to Problem (2) uses the concept of *configuration* defined to describe solutions to Problem (1). Given a solution θ to the unification problem $\{X_1, \ldots, X_m\} \doteq S$, for some $S \subseteq \{Y_1, \ldots, Y_n\}$ such that $|S| = j$, consider its configuration $c_\theta = [i_1, \ldots, i_j]$. There are $2^{j - \mathit{forks}(\theta)}$ possible values for T, where $\mathit{forks}(\theta) = 2 \cdot i_2 + \ldots + j \cdot i_j$. Hence,

$$\sum_{j=1}^{n} \binom{n}{j} \sum_{\substack{\theta \text{ is a solution to} \\ \{X_1,\ldots,X_m\} \doteq \{Y_1,\ldots,Y_j\}}} 2^{j-\mathit{forks}(\theta)}$$

is the minimal number of solutions for Problem (2).

3.3 Problem (3)

A generic solution to the problem:

$$(3) \qquad \{X_1, \ldots, X_m \,|\, Z\} \;\doteq\; \{Y_1, \ldots, Y_n \,|\, Z\}$$

must satisfy the constraints:

$$\begin{array}{rcl} S_0 & \doteq & S_1 \text{ and} \\ Z & \supseteq & (\{X_1, \ldots, X_m\} \setminus S_0) \cup (\{Y_1, \ldots, Y_n\} \setminus S_1) \end{array}$$

where S_0 is a subset of $\{X_1, \ldots, X_m\}$ and S_1 is a subset of $\{Y_1, \ldots, Y_n\}$. In other words, if θ is a most general solution to $S_0 \doteq S_1$, then:

$$\theta \cup [Z/(\{X_1, \ldots, X_m\} \setminus S_0) \cup (\{Y_1, \ldots, Y_n\} \setminus S_1) \cup N]$$

where N is a new variable, whose intended meaning is "any set," and is a solution to Problem (3). Furthermore, it is easy to observe that they are all pairwise independent. The number of most general and independent solutions to Problem (3) is therefore

$$\sum_{i=0}^{m} \binom{m}{i} \sum_{j=0}^{n} \binom{n}{j} \Phi(i, j)$$

The analysis of the simpler Problem (3′):

$$(3') \qquad \{X_1, \ldots, X_m \,|\, Z\} \;\doteq\; \{a_1, \ldots, a_n \,|\, Z\}$$

follows the same guidelines as the analysis of the more general Problem (3). The only difference is that any unification between subsets S_0 and S_1 is of Type (1′) instead of Type (1). This means that the number of most general and independent solutions to Problem (3′) is computed by:

$$\sum_{i=0}^{m} \binom{m}{i} \sum_{j=0}^{n} \binom{n}{j} Surj(i,j)$$

3.4 Problem (4)

Analogously to Problem (3), the problem

$$(4) \qquad \{X_1, \ldots, X_m \,|\, W\} \;\doteq\; \{Y_1, \ldots, Y_n \,|\, Z\}$$

can be reduced to the family of problems

$$\begin{array}{rcl} S_0 & \doteq & S_1 \\ Z & \doteq & (\{X_1, \ldots, X_m\} \setminus S_0) \cup N \cup T_0 \text{ and} \\ W & \doteq & (\{Y_1, \ldots, Y_n\} \setminus S_1) \cup N \cup T_1 \end{array}$$

where:

- $S_0 \subseteq \{X_1, \ldots, X_m\}$ and $S_1 \subseteq \{Y_1, \ldots, Y_n\}$,
- T_i is a subset of S_i, for $i = 0, 1$, and
- N is a new variable (whose intended meaning is "any set").

Nevertheless, similar to Problem (2), we need to bound the range of each T_i to avoid the generation of instances of other generated solutions.

As the analysis of Problem (1) shows, any solution θ to the problem $S_0 \doteq S_1$ can generate three different situations, (a'), (b'), and (c') (see Section 3.1). Analyzing each of these situations, we can observe that when X_i and Y_j are of Case (a'), they can be inserted into T_0 and T_1, respectively, but never simultaneously, because the solution:

$$[X_i/Y_j, \ldots, Z/\{\ldots \,|\, N\}, W/\{\ldots \,|\, N\}]$$

is more general than:

$$[X_i/Y_j, \ldots, Z/\{\ldots, X_i \,|\, N\}, W/\{\ldots, Y_j \,|\, N\}]$$

Following the same reasoning as in the analysis of Problem (2), X_i of Case (b') can be inserted into T_0, while the introduction of Y_{j_ℓ}, for $\ell = 1, \ldots, q$ in T_1 generates an instance of another solution. Similarly, Y_j of Case (c') can be inserted into T_1, while X_{i_ℓ}, for $\ell = 1, \ldots, p$ must not be introduced in T_0, to guarantee the minimality property of the solutions.

Given a solution θ to $S_0 \doteq S_1$, we define the following:

- $\mathit{vert}_0(\theta)$—the sum of the number of k-forks for $k = 2, \ldots, |S_1|$;
- $\mathit{vert}_1(\theta)$—the sum of the number of h-cones for $h = 2, \ldots, |S_0|$;
- $\mathit{cones}(\theta)$—the sum $\sum_{h=2}^{|S_0|} h \cdot (\#\text{ of } h\text{-cones})$; and
- $\mathit{forks}(\theta)$—the same function defined in the solution to Problem (2).

Clearly, $|S_0| - \mathit{cones}(\theta) - \mathit{vert}_0(\theta) = |S_1| - \mathit{forks}(\theta) - \mathit{vert}_1(\theta)$; such a number (say p) is the number of elements connected with a bijection (see Figure 2). As it has already been explained, such elements can all be inserted in T_0 and T_1, but not simultaneously: there are

$$|\,\{\langle A, B\rangle : A, B \subseteq \{1, \ldots, p\}, A \cap B = \emptyset\}\,| \;= 3^p$$

possibilities to extend θ.

Thus, we are ready to count all the solutions to Problem (4):

$$\sum_{a=0}^{m} \binom{m}{a} \sum_{b=0}^{n} \binom{n}{b} \sum_{\substack{\theta \text{ is a solution to} \\ \{X_1, \ldots, X_a\} \doteq \{Y_1, \ldots, Y_b\}}} 2^{\mathit{vert}_0(\theta)} 2^{\mathit{vert}_1(\theta)} 3^{a - \mathit{vert}_0(\theta) - \mathit{cones}(\theta)}$$

We now analyze the simpler problem:

$$(4') \qquad \{X_1, \ldots, X_m \,|\, W\} \;\doteq\; \{a_1, \ldots, a_n \,|\, Z\}$$

It is simpler because no fork can occur in the solution of $S_0 \doteq S_1$ when $S_1 \subseteq \{a_1, \ldots, a_n\}$. In particular, we can prove that the number of solutions to Problem (4$'$) is computed by the following function f:

$$\begin{cases} f(m,0) = & 1 \\ f(m,n+1) = & f(m,n) + 3 \cdot \binom{m}{1} f(m-1,n) + 2 \cdot \sum_{k=2}^{m} \binom{m}{k} f(m-k,n) \end{cases}$$

If $n = 0$ the result is trivial. Assume $n > 0$. Any solution for

$$\{X_1, \ldots, X_m \,|\, W\} \doteq \{a_1, \ldots, a_n \,|\, Z\}$$

can be obtained:

- Extending uniquely a solution to

  $$\{X_1, \ldots, X_m \,|\, W\} \doteq \{a_2, \ldots, a_n \,|\, Z\}$$

  adding the element a_1 to the set assigned to W.

- Extending any solution to

  $$\{X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_m \,|\, W\} \doteq \{a_2, \ldots, a_n \,|\, Z\}$$

  with the binding $[X_i/a_1]$ plus (possibly)

  - adding a_1 to the set assigned to W;
  - adding X_i (or, equivalently, a_1) to the set assigned to Z.

  Either the former or the latter extension can be done, but not simultaneously. However, adding both of them to the solution is an instance of the solution without extensions.

- Let $k \geq 2$ and $j : \{1, \ldots, m\} \longrightarrow \{1, \ldots, n\}$ be a permutation. A solution for the $\langle m, n \rangle$ problem can be obtained extending any solution to

  $$\{X_{j_{k+1}}, \ldots, X_{j_m} \,|\, W\} \doteq \{a_2, \ldots, a_n \,|\, Z\}$$

  with the binding $[X_{j_1}/a_1, \ldots, X_{j_k}/a_1]$ plus (possibly) adding a_1 to the set assigned to W. Observe that if X_{j_b}, $b = 1, \ldots, k$ (or, equivalently, a_1) is added to the set assigned to Z, then it would be an instance of one of the solutions obtained extending

  $$\{X_{j_b}, X_{j_{k+1}}, \ldots, X_{j_m} \,|\, W\} \doteq \{a_2, \ldots, a_n \,|\, Z\}$$

  in which X_{j_b} is not bound to any of $a_2, \ldots, a_n$ in one of the admitted ways.

Note that as in Problem (2), it is possible to count the number of solutions to Problem (4) by means of

$$\sum_{a=0}^{m} \binom{m}{a} \sum_{b=0}^{n} \binom{n}{b} \Phi''(a, b)$$

where again, Φ'' is similar to Φ, but now taking care of the possible extensions of Z and W. That is the reason why factors 2 and 3 occur when cones, forks, and simple bindings are generated:

$$\begin{cases} \Phi''(m,n) = 0 & (m>0, n=0) \text{ or } (m=0, n>0) \\ \Phi''(m,n) = 3 & (m=n=1) \\ \Phi''(1,n) = 2 & (n>1) \\ \Phi''(m,1) = 2 & (m>1) \\ \Phi''(m,n) = 3n\Phi''(m-1,n-1) & \\ \qquad 2n\sum_{i=1}^{m-2}\binom{m-1}{i}\Phi''(m-1-i,n-1) + & m,n>1 \\ \qquad 2\sum_{j=2}^{n-1}\binom{n}{j}\Phi''(m-1,n-j) & \end{cases}$$

3.5 Problem (5)

As explained in the introduction to this section, it is interesting to analyze whether or not the two sets involved in the equations share elements. This allows for a great reduction in the number of the most general unifiers. We will first analyze a ground sharing (Case (5′)), and then the more general Case (5); the two problems will admit the same number of most general solutions.

Any solution θ to the problem

$$(5') \qquad \{X_1,\ldots,X_m,a_1,\ldots,a_k\} \doteq \{Y_1,\ldots,Y_n,a_1,\ldots,a_k\}$$

maps every element of the lefthand side into an element of the righthand side of the equation. Clearly, for $i=1,\ldots,k$, a_i is implicitly mapped into itself.

Assume θ has the form $[X_{i_1}/a_\ell,\ldots,X_{i_\alpha}/a_\ell,Y_{j_1}/a_\ell,\ldots,Y_{j_\beta}/a_\ell]\cup\theta'$, where $\alpha,\beta\geq 1$. Such a θ is an instance of the substitution $\theta''\cup\theta'$, for any θ'' solution to the problem of Type (1): $\{X_{i_1},\ldots,X_{i_\alpha}\}\doteq\{Y_{j_1},\ldots,Y_{j_\beta}\}$.

This means, in particular, that we do not have to count solutions in which both X_i/a_ℓ and Y_j/a_ℓ occur in the solution, for any i,j, and ℓ.

Any most general solution θ to Problem (5′) can be obtained as follows:

1. choose two disjoint subsets S_0 and S_1 of $\{a_1,\ldots,a_k\}$;
2. choose a subset T_0 of $\{X_1,\ldots,X_m\}$ and a subset T_1 of $\{Y_1,\ldots,Y_n\}$;
3. θ_i is a solution to the problem of Type (1′), $T_i\doteq S_i$, for $i=0,1$;

4. θ_2 is a solution to the problem of Type (1), $\{X_1, \ldots, X_m\} \backslash T_0 \doteq \{Y_1, \ldots, Y_n\} \setminus T_1$; and

5. θ is $\theta_0 \cup \theta_1 \cup \theta_2$.

Hence, the number of most general and independent solutions to Problem (5′) is:

$$\begin{aligned}\Psi(m,n,k) \;=\; & \sum_{i=0}^{k} \binom{k}{i} \sum_{j=0}^{k-i} \binom{k-i}{j} \sum_{a=0}^{m} \binom{m}{a} \sum_{b=0}^{n} \binom{n}{b} \\ & \Big(\mathit{Surj}(a,i) \cdot \mathit{Surj}(b,j) \cdot \Phi(m-a, n-b) \Big)\end{aligned}$$

From any solution θ to Problem (5′), a corresponding solution to Problem

$$(5) \qquad \{X_1, \ldots, X_m, Z_1, \ldots, Z_k\} \;\doteq\; \{Y_1, \ldots, Y_n, Z_1, \ldots, Z_k\}$$

can be obtained by replacing any occurrence of a_i with Z_i, for $i = 1, \ldots, k$. Although it seems that Problem (5) has a number of solutions strictly greater than Problem (5′), since it is consistent to consider solutions to $T_0 \doteq S_0$ and $T_1 \doteq S_1$ containing p-forks ($p \leq k$), note that any solution $\theta = [Z_{i_1}/X_i, \ldots, Z_{i_p}/X_i] \cup \theta'$, $p \geq 2$, of the form described above, is an instance of each of the solutions $\theta_{i_1} = [Z_{i_1}/X_i] \cup \theta'$, …,$\theta_{i_p} = [Z_{i_p}/X_i] \cup \theta'$.

Thus, Problem (5) has *exactly* the same number of most general and independent solutions as Problem (5′).

Before ending this section, we will show that the number of solutions to Problems (1)–(5) is bounded by $(4(m+n))^{(m+n)}$. All such functions are bounded by the function $\delta(m,n)$ defined as:

$$\sum_{a=0}^{m} \binom{m}{a} \sum_{b=0}^{n} \binom{n}{b} \Phi(a,b) 2^{m-a} 2^{n-b}$$

The function $\delta(m,n)$ represents the number of solutions to Problem (4) without the careful removal of redundant solutions performed in Section 3.4.

Lemma 2 $\delta(m,n) \leq (4(m+n))^{(m+n)}$.

Proof of Lemma 2 Since Φ is monotonic, then

$$\begin{array}{rcl} \delta(m,n) & \leq & \sum_{a=0}^{m} \binom{m}{a} \sum_{b=0}^{n} \binom{n}{b} \Phi(m,n) 2^{m-a} 2^{n-b} \\ & = & \Phi(m,n) \sum_{a=0}^{m} \binom{m}{a} \sum_{b=0}^{n} \binom{n}{b} 2^{m-a} 2^{n-b} \\ & \leq & \Phi(m,n) \sum_{a=0}^{m} \binom{m}{a} \sum_{b=0}^{n} \binom{n}{b} 2^{m} 2^{n} \\ & = & \Phi(m,n) 2^{m} 2^{n} \sum_{a=0}^{m} \binom{m}{a} \sum_{b=0}^{n} \binom{n}{b} \\ & = & \Phi(m,n) 2^{m} 2^{n} 2^{m} 2^{n} \\ & = & 4^{m} 4^{n} \Phi(m,n) \end{array}$$

Lemma 1 ensures that

$$\delta(m,n) \leq 4^{m} 4^{n} m^{n} n^{m} \leq (4(m+n))^{(m+n)}$$

Proof of Lemma 2 □

4 The Algorithm SUA

This section is devoted to describing a new set-unification algorithm, named SUA, which is proved to be minimal for all the sample problems presented in the previous section. Moreover, a brief analysis of a naïve algorithm is presented, to compare it to SUA.

The code associated with SUA is presented in Figure 3. To simplify the description of the algorithm, some local notation is defined. We make use of a meta-function Can, which computes to any term t a canonical representation $Can(t)$ of its equivalence class with respect to $=_{\mathcal{Set}}$. In more detail, when t is a set term, $Can(t)$ returns t', obtained by first removing repeated elements in t and ordering the elements using the lexicographic order. For instance, $Can(\{a,a,b\})$ returns $\{a,b\}$. From an implementation point of view, the "canonization" of sets allows us to easily detect if two sets are syntactically equal (see Action 1 of functions unify_set, unify_set2, limit_1, and limit_2 in Figures 4, 6, 7, and 8, respectively).

The set-operations $|\cdot|$ (cardinality), $\subseteq$ (inclusion), $\subset$ (strict inclusion), $\cup$ (union), $\cap$ (intersection), and $-$ (set difference) are used on terms denoting sets. The meaning of the set operators is purely syntactical; for instance, $|\{X_1, X_2\}| = 2$: we do not need to distinguish the two cases $X_1 = X_2 \wedge |\{X_1, X_2\}| = 1$ and $X_1 \neq X_2 \wedge |\{X_1, X_2\}| = 2$.

In the algorithm, X, W, W', Z, and Z' denote generic variables, t, t_1, s_1, t_2, and $s_2, \ldots$, denote generic terms, and N denotes a new variable introduced

```
function SUA($\mathcal{E}$);
  If $\mathcal{E}$ is in solved form, then return $\mathcal{E}$
  elseif $\mathcal{E}^*$ is not empty, then choose n.d. one active equation $s \overset{*}{=} t$ in $\mathcal{E}^*$,
        consider $\mathcal{E}' := \mathcal{E} - \{s \overset{*}{=} t\}$ and:
          0. i. $t \equiv X$: if $X \in Var(s)$ then fail, else return SUA($\mathcal{E}'[X/s] \cup \{X \doteq s\}$);
             ii. $s \equiv X$: if $X \in Var(t)$ then fail, else return SUA($\mathcal{E}'[X/t] \cup \{X \doteq t\}$);
            iii. $s \equiv f(s_1, \ldots, s_m)$ and $t \equiv f'(t_1, \ldots, t_n)$:
                   if $f \not\equiv f'$, then fail
                   else (i.e., $f \equiv f'$ and $m = n$): return SUA($\{s_1 \doteq t_1, \ldots, s_n \doteq t_n\} \cup \mathcal{E}'$);
  else choose n.d. one equation $e$ (not in solved form) in $\mathcal{E}$; $\mathcal{E}' := \mathcal{E} - \{e\}$;
        case $e$ of:
          1. $X \doteq X$: return SUA($\mathcal{E}'$);
          2. $t \doteq X$ and $t \notin \mathcal{V}$: return SUA($\{X \doteq t\} \cup \mathcal{E}'$);
          3. $X \doteq t$, $X \in Var(\mathcal{E}')$, $X \notin Var(t)$: return SUA($\mathcal{E}'[X/t] \cup \{X \doteq t\}$);
          4. $X \doteq \{t_1, \ldots, t_m \,|\, X\}$: return SUA($\mathcal{E}' \cup \{X \overset{*}{=} \{t_1, \ldots, t_m \,|\, N\}\}$);
          5. $X \doteq \{t_1, \ldots, t_m \,|\, t_0\}$:
               if $t_0 \in \mathcal{V}$ and $t_0 \not\equiv X$ and $X \in \bigcup_{i=1}^m Var(t_i)$: fail
               elseif $t_0 \equiv f(s_1, \ldots, s_n)$, $f \not\equiv \{\cdot \,|\, \cdot\}$ and $X \in \bigcup_{i=0}^m Var(t_i)$: fail;
          6. $X \doteq t$ and $t \equiv f(t_1, \ldots, t_n)$, $f \not\equiv \{\cdot \,|\, \cdot\}$ and $X \in Var(t)$: fail;
          7. $f(s_1, \ldots, s_n) \doteq g(t_1, \ldots t_m)$, $f \not\equiv g$: fail;
          8. $f(s_1, \ldots, s_n) \doteq f(t_1, \ldots t_n)$, $f \not\equiv \{\cdot \,|\, \cdot\}$:
               return SUA($\mathcal{E}' \cup \{s_1 \doteq t_1, \ldots, s_n \doteq t_n\}$);
          9. $\{t_1, \ldots, t_m \,|\, k\} \doteq \{s_1, \ldots, s_n \,|\, k'\}$; let $T := Can(\{t_1, \ldots, t_m\})$ and
             $S := Can(\{s_1, \ldots, s_n\})$: return SUA(unify_set($T, S$) $\cup \{k \overset{*}{=} k'\} \cup \mathcal{E}'$);
         10. $\{s_1, \ldots, s_n \,|\, k\} \doteq \{t_1, \ldots, t_m \,|\, Z\}$:
               return SUA($\{\{t_1, \ldots, t_m \,|\, Z\} \doteq \{s_1, \ldots, s_n \,|\, k\}\} \cup \mathcal{E}'$);
         11. $\{t_1, \ldots, t_m \,|\, Z\} \doteq \{s_1, \ldots, s_n \,|\, k\}$: let $T := Can(\{t_1, \ldots, t_m\})$ and
               $S := Can(\{s_1, \ldots, s_n\})$. choose n.d. $\emptyset \neq S' \subseteq S$:
               return SUA($\{Z \overset{*}{=} (S - S') \cup Z' \cup \{k\}\} \cup$ limit_1$(T, S', Z')|_{S \cup T \cup \{Z\}} \cup \mathcal{E}'$);
         12. $\{t_1, \ldots, t_m \,|\, Z\} \doteq \{s_1, \ldots, s_n \,|\, Z\}$: let $T := Can(\{t_1, \ldots, t_m\})$ and
               $S := Can(\{s_1, \ldots, s_n\})$. Select n.d. $T' \subseteq T$ and $S' \subseteq S$:
                 if $T' \cup S' \neq T \cup S$ then
                    return SUA(unify_set($T', S'$) $\cup \{Z \overset{*}{=} (T - T') \cup (S - S') \cup N\} \cup \mathcal{E}'$)
                 else return SUA(unify_set($T, S$) $\cup \mathcal{E}'$);
         13. $\{t_1, \ldots, t_m \,|\, W\} \doteq \{s_1, \ldots, s_n \,|\, Z\}$ where $Z$ and $W$ are different variables:
               let $T := Can(\{t_1, \ldots, t_m\})$ and $S := Can(\{s_1, \ldots, s_n\})$.
               Choose n.d. $T' \subseteq T$, $S' \subseteq S$:
               return SUA($\{(S - S') \cup W' \overset{*}{=} W, (T - T') \cup Z' \overset{*}{=} Z\} \cup$
                           limit_2($T'$, $S'$, $Z'$, $W'$) $|_{S \cup T \cup \{Z, W\}} \cup \mathcal{E}'$)).
```

Figure 3: The set-unification algorithm SUA.

```
function unify_set($\{t_1,\dots,t_m\},\{s_1,\dots,s_n\}$);
  1. If $\{t_1,\dots,t_m\}$ and $\{s_1,\dots,s_n\}$ are syntactically equal then return $\{\ \}$;
  2. elseif $m = 1$ and $n > 1$ then return $\{s_i \doteq t_1 : 1 \leq i \leq n\}$;
  3. elseif $m \geq 1$ and $n = 1$ then return $\{t_i \doteq s_1 : 1 \leq i \leq m\}$;
  4. else consider $Common\_part := \{t_1,\dots,t_m\} \cap \{s_1,\dots,s_n\}$;
                   $Disagr_1 := \{t_1,\dots,t_m\} - Common\_part$;
                   $Disagr_2 := \{s_1,\dots,s_n\} - Common\_part$;
    (a) if $Common\_part = \emptyset$ then fix an $i \in \{1,\dots,m\}$ and
          select n.d. one of the following actions:
            i. return $\overset{1}{=}(t_i,\{t_1,\dots,t_m\},\{s_1,\dots,s_n\})$;
            ii. return $\overset{2}{=}(t_i,\{t_1,\dots,t_m\},\{s_1,\dots,s_n\})$;
            iii. return $\overset{3}{=}(t_i,\{t_1,\dots,t_m\},\{s_1,\dots,s_n\})$;
    (b) if $Common\_part \neq \emptyset$ then choose n.d. $S_0, S_1 \subseteq Common\_part$, $S_0 \cap S_1 = \emptyset$;
        choose n.d. $T_0 \subseteq Disagr_1$, $T_1 \subseteq Disagr_2$ such that $|T_0| \geq |S_0|$ and $|T_1| \geq |S_1|$:
          return unify_set($Disagr_1 - T_0, Disagr_2 - T_1$) $\cup$
          unify_set2($T_1, S_1$) $\cup$ unify_set2($T_0, S_0$).
```

Figure 4: Function unify_set.

by SUA. The terms k and k' denote nonvariables whose main functional symbol is distinct from $\{\cdot\,|\,\cdot\}$.[5] Given a set of terms S and a substitution θ, $\theta|_S$ constrains the domain of θ to the variables contained in S. The abbreviation "n.d." stands for "nondeterministically."

The algorithm SUA takes as input a system of equations $\mathcal{E}$ between terms, and returns either *fail*—$\mathcal{E}$ is not unifiable—or, nondeterministically, a substitution θ. The set of all such θs constitutes a complete set of $\mathcal{S}et$-unifiers. The algorithm temporarily generates equations (introduced by actions 4, 9, 11, 12, and 13) marked by $*$; they are called *active* equations, and are immediately removed by action 0. This is fundamental to guaranteeing termination (see details in the proof of Theorem 2). The set of active equations is denoted by $\mathcal{E}^*$.

Actions 1–8 of SUA are identical to those used in the unification algorithm presented in [DOPR96], and most of them are based on Clark's equality theory [Cla78]. Note that Action 4 does not return a failure. The reason is that the unification problem $X \doteq \{t_1,\dots,t_m|X\}$, where X does not occur in $t_1,\dots,t_m$, admits as (most general) solution $\theta = [X/\{t_1,\dots,t_m|N\}]$, where N is a fresh variable. Of course it holds that $X\theta =_{\mathcal{S}et} \{t_1,\dots,t_m|X\}\theta$,

[5]Such entities are named *kernels* in [DOPR96].

```
function $\overset{1}{=}(t_i, \{t_1, \ldots, t_m\}, \{s_1, \ldots, s_n\})$;
  choose n.d. one $j \in \{1, \ldots, n\}$:
  return $\{t_i \doteq s_j\} \cup$ unify_set$(\{t_1, \ldots, t_{i-1}, t_{i+1}, \ldots, t_m\}, \{s_1, \ldots, s_{j-1}, s_{j+1}, \ldots, s_n\})$.

function $\overset{2}{=}(t_i, \{t_1, \ldots, t_m\}, \{s_1, \ldots, s_n\})$;
  choose n.d. $S \subset \{s_1, \ldots, s_n\}$ such that $|S| \geq 2$:
  return $\{s \doteq t_i$ : for all $s \in S\} \cup$ unify_set$(\{t_1, \ldots, t_{i-1}, t_{i+1}, \ldots, t_m\}, \{s_1, \ldots, s_n\} - S)$.

function $\overset{3}{=}(t_i, \{t_1, \ldots, t_m\}, \{s_1, \ldots, s_n\})$;
  choose n.d. $\emptyset \neq T \subset \{t_1, \ldots, t_{i-1}, t_{i+1}, \ldots, t_m\}$:
  return $\{t \doteq t_i$ : for all $t \in T\} \cup \overset{1}{=}(t_i, \{t_1, \ldots, t_m\} - T, \{s_1, \ldots, s_n\})$.
```

Figure 5: Functions $\overset{1}{=}$, $\overset{2}{=}$, and $\overset{3}{=}$.

because of the *absorption property*.

Action 9 corresponds to the resolution of Problems (1), (1′), (5), and (5′), whereas actions 11, 12, and 13 solve, respectively, Problems (2), (2′), and (3) and (3′), (4), and (4′). As we show later, all these actions faithfully follow the ideas presented in the minimality studies described in Section 3. This justifies the minimality of SUA for all set-unification problems studied in this work.

The function unify_set defined in Figure 4 takes as input two terms $\{t_1, \ldots, t_m\}$ and $\{s_1, \ldots, s_n\}$, representing nonempty sets, and selects nondeterministically which equalities between elements of $\{t_1, \ldots, t_m\}$ and $\{s_1, \ldots, s_n\}$ should accompany the system $\mathcal{E}$ in a recursive call to SUA. Some comments on unify_set are needed to relate it to the unification Problems (1), (1′), (5), and (5′). Action 4(a) is motivated by problems of Type (1) and (1′). As the analysis of Problem (1) shows, we need to distinguish three different possibilities, (a'), (b'), and (c') (see Section 3.1), to compute all solutions for such a problem. These three cases are subsumed by functions $\overset{1}{=}$, $\overset{2}{=}$, and $\overset{3}{=}$ in Figure 5, respectively. Function $\overset{1}{=}$ generates *simple bindings* by matching one element of the first set, t_i, with one element of the second set, s_j. Afterwards, it combines the two sets deprived of the selected elements. Function $\overset{2}{=}$ captures the concept of k-fork, and Function $\overset{3}{=}$ captures the concept of k-cone. In particular, when an answer is computed by SUA without using Function $\overset{2}{=}$, then the answer may be considered as a surjective function from the leftmost set onto the rightmost set. Since Function $\overset{2}{=}$ cannot be applied

```
function unify_set2($\{t_1,\ldots,t_m\},\{s_1,\ldots,s_n\}$);
  1. If $\{t_1,\ldots,t_m\}$ and $\{s_1,\ldots,s_n\}$ are syntactically equal then return { };
  2. elseif $m \geq 1$ and $n = 1$ then return $\{t_i \doteq s_1 : 1 \leq i \leq m\}$;
  3. else fix an $i \in \{1,\ldots,m\}$: select n.d. one of the following actions:
       i. return $\overset{1'}{\doteq}(t_i,\{t_1,\ldots,t_m\},\{s_1,\ldots,s_n\})$;
       ii. return $\overset{3'}{\doteq}(t_i,\{t_1,\ldots,t_m\},\{s_1,\ldots,s_n\})$;
```

Figure 6: Function unify_set2.

to Problem (1′) (otherwise, SUA would produce *fail*), then the algorithm computes surjective functions as solutions to Problem (1′).

As we noted in the analysis of Problem (5), $\{X_1,\ldots,X_m,Z_1,\ldots,Z_k\} \doteq \{Y_1,\ldots,Y_n,Z_1,\ldots,Z_k\}$, solutions associated with $T_i \doteq S_i$, $i \in \{0,1\}$, $S_0 \cap S_1 = \emptyset$, $S_i \subseteq \{Z_1,\ldots,Z_k\}$, $T_0 \subseteq \{X_1,\ldots,X_m\}$, and $T_1 \subseteq \{Y_1,\ldots,Y_n\}$ containing forks are redundant. For this reason, Action 4(b) of unify_set requires us to use a new function, unify_set2 (see Figure 6). The intended meaning of this new function is to consider nondeterministically either $\overset{1}{\doteq}$ or $\overset{3}{\doteq}$, but never $\overset{2}{\doteq}$.[6] The formal definition of unify_set2 is equal to the definition of unify_set, but with the removal of those actions corresponding to the generation of forks, namely, Actions 2 and 4(a)ii. Notice that in particular, the case $m = 1$, $n > 1$ never arises, since the initial call to unify_set2 requires $m \geq n$. The auxiliary Functions $\overset{1'}{\doteq}$ and $\overset{3'}{\doteq}$ are defined as $\overset{1}{\doteq}$ and $\overset{3}{\doteq}$, respectively, changing the recursive calls to unify_set by calls to unify_set2.

A problem of Type (2) or (2′) can only be solved by using Action 11 of SUA. The Function limit_1, presented in Figure 7, must simultaneously solve the unification problem $\{X_1,\ldots,X_m\} \doteq S$ for some $S \subseteq \{Y_1,\ldots,Y_n\}$ (for such reasons, the definitions of $\overset{1}{\doteq}$, $\overset{2}{\doteq}$, and $\overset{3}{\doteq}$ are embedded in its definition and bind Z to any subset of S not containing variables occurring in any h-fork corresponding to the solution being computed. The definition of limit_2 in Figure 8 is similar to that of limit_1. In particular, for problems of Type (4), the values of W and Z are constrained to avoid the introduction of variables occurring in some h-fork or h-cone, respectively, of the solution θ being computed. On the other hand, limit_2 must also control that those variables bounded in θ by a simple binding are not introduced in Z and W simultaneously.

[6]Note that for Problem (5′), the use of Function $\overset{2}{\doteq}$ is not possible.

```
function limit_1($\{t_1,\dots,t_m\}$,$\{s_1,\dots,s_n\}$, $Z$);
  1. if $\{t_1,\dots,t_m\}$ and $\{s_1,\dots,s_n\}$ are syntactically equal, then choose n.d.
       $S \subseteq \{s_1,\dots,s_n\}$: return $\{Z \doteq S\}$;
  2. elseif $m = 1$ and $n > 1$, then return $\{s_i \doteq t_1 :\ 1 \le i \le n\} \cup \{Z \doteq \underline{\emptyset}\}$;
  3. elseif $m = 1$ and $n = 1$ or $m > 1$ and $n = 1$, then choose n.d. $T \subseteq \{s_1\}$:
       return $\{t_i \doteq s_1 :\ 1 \le i \le m\} \cup \{Z \doteq T\}$;
  4. else fix $i \in \{1,\dots,m\}$ and choose n.d. one of the following actions:
       i. choose n.d. $j \in \{1,\dots,n\}$ and $T \subseteq \{s_j\}$:
            return $\{t_i \doteq s_j\} \cup \{Z \doteq T \cup Z'\} \cup$
              limit_1($\{t_1,\dots,t_{i-1},t_{i+1},\dots,t_m\},\{s_1,\dots,s_{j-1},s_{j+1},\dots,s_n\},Z'$);
      ii. choose n.d. $S \subset \{s_1,\dots,s_n\}$ such that $|S| \ge 2$:
            return $\{s \doteq t_i :$ for all $s \in S\} \cup$
              limit_1($\{t_1,\dots,t_{i-1},t_{i+1},\dots,t_m\},\{s_1,\dots,s_n\} - S, Z$);
     iii. choose n.d. $T' \subset \{t_1,\dots,t_{i-1},t_{i+1},\dots,t_m\}$, $j \in \{1,\dots,n\}$ and $T \subseteq \{s_j\}$:
            return $\{t \doteq s_j :$ for all $t \in T'\} \cup \{t_i \doteq s_j\} \cup \{Z \doteq T \cup Z'\} \cup$
              limit_1($\{t_1,\dots,t_{i-1},t_{i+1},\dots,t_m\} - T',\{s_1,\dots,s_{j-1},s_{j+1},\dots,s_n\},Z'$).
```

Figure 7: Function limit_1.

4.1 Termination, Correctness, and Minimality of SUA

The following definition is helpful for the termination proof. Let $\mathcal{E}$ be a system of equations, and let p be the total number of occurrences of function symbols in the system. Then a function $lev : Var(\mathcal{E}) \longrightarrow \mathcal{N}$ extended to nonvariable terms as follows:

$$\begin{array}{rcll} lev(f(t_0,\dots,t_n)) & = & \max\{1+lev(t_0),\dots,1+lev(t_n)\} & f \in \Sigma,\ f \not\equiv \{\cdot\,|\,\cdot\} \\ lev(\{t\,|\,s\}) & = & \max\{1+lev(t), lev(s)\} & \end{array}$$

and fulfilling condition *(i)* $lev(\ell), lev(r) \le p$, for any equation $\ell \doteq r$ in $\mathcal{E}$, always exists. If we require the further condition *(ii)* $lev(\ell) = lev(r)$ for any equation $\ell \doteq r$ in $\mathcal{E}$, then such a *lev* may not exist (e.g., when $\mathcal{E} = \{X \doteq f(X)\}$).

Lemma 3 *Let $\mathcal{E}$ be a unifiable Herbrand system; then a function lev whose extension to terms fulfills both conditions (i) and (ii) always exists.*

Let $\{X \doteq t\} \cup \mathcal{E}$ be an equation system, and let p be the number of occurrences of functional symbols in it. Assume X does not occur in t and assume the function lev fulfills condition (i). Moreover, assume that lev is such that $lev(X) = lev(t)$. Then lev fulfills condition (i) also for the system $\mathcal{E}[X/t] \cup \{X \doteq t\}$.

Algorithm SUA calls to the functions unify_set, unify_set2, limit_1, and limit_2. To prove the termination of them is straightforward. In the proof of the following theorem, we also make use of their semantics.

Theorem 2 (Termination) *For any input system $\mathcal{E}$,* SUA *always terminates, no matter what nondeterministic sequence of choices is made.*

Proof of Theorem 2 Assume that there is an infinite sequence of nondeterministic choices such that SUA($\mathcal{E}$) does not terminate. Let $E^{(0)}, E^{(1)}, E^{(2)}, \ldots$ be the values of $\mathcal{E}$ at the $0^{th}, 1^{st}, 2^{nd}, \ldots$ iteration, respectively, and let p be the number of occurrences of functional symbols in $E^{(0)}$. A function $lev : Var(\bigcup_{j \geq 0} E^{(j)}) \longrightarrow \mathcal{N}$ must necessarily exist such that:

- the function fulfills condition (i) for all equation sets $E^{(j)}$, and
- any time a substitution $[X/t]$ has been applied (Actions 0 and 3), then $lev(X) = lev(t)$.

If the function did not exist, then a failure situation caused by occur check would rise, causing the computation to be finite.

Picking such a *lev*, we define a measure of complexity $\mathcal{L}_{\mathcal{E}}$ for the equation set $\mathcal{E}$: $\mathcal{L}_{\mathcal{E}} \ =_{Def} \ [\#(2p), \#(2p-1), \#(2p-2), \ldots, \#(1), \#(0)]$ where $\#(j)$ returns the number of equations *not in solved form* $\ell \doteq r$ in $\mathcal{E}$ such that $lev(\ell) + lev(r) = j$. The ordering between two lists of this form is the well-founded lexicographical ordering.

It is easy to see that Actions 0, 1, 3, and 8 cause $\mathcal{L}_{\mathcal{E}}$ to strictly decrease. Actions 4, 9, 11, 12, and 13 do not increase $\mathcal{L}_{\mathcal{E}}$; however, they introduce active (i.e., marked by $*$) equations that will be eliminated immediately by Action 0. Actions 2 and 10 leave $\mathcal{L}_{\mathcal{E}}$ unchanged; nevertheless, they at most can double the number of actions, hence we may forget them. Because the lexicographical ordering on constant-length lists of non-negative integers is well-ordered, this is sufficient to prove termination.

Proof of Theorem 2 □

The above theorem proves that the SUA search tree does not contain infinite branches: this guarantees that SUA computes a finite number of unifiers.

```
function limit_2($\{t_1,\dots,t_m\}$,$\{s_1,\dots,s_n\}$ $Z$,$W$);
  1. If $\{t_1,\dots,t_m\}$ and $\{s_1,\dots,s_n\}$ are syntactically equal then choose n.d.
       $T \subseteq \{t_1,\dots,t_m\}$ and $S \subseteq \{s_1,\dots,s_n\}$: return $\{Z \doteq T, W \doteq S\}$;
  2. elseif $m = 1$ and $n > 1$ then choose n.d. $T \subseteq \{t_1\}$:
         return $\{s_i \doteq t_1 :\ 1 \le i \le n\} \cup \{Z \doteq T, W \doteq \underline{\emptyset}\}$;
  3. elseif $n = 1$ and $m > 1$ then choose n.d. $S \subseteq \{s_1\}$:
         return $\{t_i \doteq s_1 :\ 1 \le i \le m\} \cup \{Z \doteq \underline{\emptyset}, W \doteq S\}$;
  4. elseif $m = 1$ and $n = 1$ then choose n.d. $T \subseteq \{t_1\}$, $S \subseteq \{s_1\}$
         such that $T \cup S \neq \{t_1, s_1\}$:
         return $\{t_1 \doteq s_1, Z \doteq T, W \doteq S\}$;
  5. else fix $i \in \{1,\dots,m\}$ and choose n.d. one of the following actions:
         i. choose n.d. $j \in \{1,\dots,n\}$ and $S \subseteq \{s_j\}$, $T \subseteq \{t_j\}$
            such that $T \cup S \neq \{s_j, t_i\}$:
            return $\{t_i \doteq s_j, W \doteq S \cup W', Z \doteq T \cup Z'\} \cup$
            limit_2($\{t_1,\dots,t_{i-1},t_{i+1},\dots,t_m\}$,
                    $\{s_1,\dots,s_{j-1},s_{j+1},\dots,s_n\}$, $Z'$, $W'$);
        ii. choose n.d. $S \subset \{s_1,\dots,s_n\}$ such that $|S| \ge 2$, $T \subseteq \{t_i\}$:
            return $\{s \doteq t_i :\ \text{for all } s \in S\} \cup \{Z \doteq T \cup Z'\} \cup$
            limit_2($\{t_1,\dots,t_{i-1},t_{i+1},\dots,t_m\}$, $\{s_1,\dots,s_n\} - S$, $Z'$, $W$);
       iii. choose n.d. $T \subset \{t_1,\dots,t_{i-1},t_{i+1},\dots,t_m\}$,
            $j \in \{1,\dots,n\}$ and $S \subseteq \{s_j\}$:
            return $\{t \doteq s_j :\ \text{for all } t \in T\} \cup \{t_i \doteq s_j, W \doteq S \cup W'\} \cup$
            limit_2($\{t_1,\dots,t_{i-1},t_{i+1},\dots,t_m\} - T$, $\{s_1,\dots,s_{j-1},s_{j+1},\dots,s_n\}$, $Z$, $W'$).
```

Figure 8: Function limit_2.

Theorem 3 (Correctness and Completeness) *The algorithm* SUA *is correct and complete with respect to the well-founded theory of hybrid sets $\mathcal{S}et$ presented in Section 2.*

Proof of Theorem 3 Actions 1 and 2 are justified by reflexivity and symmetry of equality, respectively. Action 3 is the substitution application. Its correctness (and completeness) follows from equality axioms. Actions 5 and 6 are the occur check, justified by well-foundedness on (set) terms. Actions 0i and 0ii are combinations of Actions 3, 5, and 6.

Actions 7 and 8 are justified by Clark's equality theory or CET (see [Cla78]). Action 0iii is a combination of Actions 7 and 8.

Note that in Action 4, a unifier of $X = \{t_1,\dots,t_n \,|\, X\}$ is $\theta = [X/\{t_1,\dots,t_n \,|\, N\}]$, where N is a new variable. As a matter of fact, $\{t_1,\dots,t_n \,|\, N\}$ can be proved equal to $\{t_1,\dots,t_n,t_1,\dots,t_n \,|\, N\}$ by (A_b) and (C_ℓ). Moreover, any unifier μ of the above equation must guarantee that $t_1\mu \in X\mu,\dots,t_n\mu \in X\mu$.

Hence, θ is the most general unifier.

Soundness and completeness of Actions 9–13 follow because they constitute a general schemata for every set-to-set equation. As shown in Section 3, finding all solutions from Problems (1)–(4) is sufficient to be able to compute all solutions to any set-to-set equation. This allows us to say that completeness follows from the minimality result (Theorem 4 below).

Proof of Theorem 3 $\square$

It remains to show that the unification algorithm SUA is minimal with respect to all chosen sample problems. We will deal only with set terms ended by the constant symbol $\underline{\emptyset}$. This simplifies the analysis of Actions 9 and 11 of the algorithm (k and k' are both $\underline{\emptyset}$). The very long and technical proof is not given completely here, due to space limitations. We only show the minimal behavior of SUA for Problem (1), in which most of the analyzed problems are based. Interested readers can consult [ASD95] for a complete proof.

Lemma 4 *A call to* unify_set$(\{X_1, \ldots, X_m\}, \{Y_1, \ldots, Y_n\})$ *computes exactly* $\Phi(m, n)$ *solutions.*

Proof of Lemma 4 Straightforward (by induction on m).

Proof of Lemma 4 $\square$

Lemma 5 *If* θ *is an answer computed by* unify_set$(\{X_1, \ldots, X_m\}, \{Y_1, \ldots, Y_n\})$*,* $m, n \geq 1$*, then*

$$\theta \in CSU_{\mathcal{S}et}(\{X_1, \ldots, X_m\}, \{Y_1, \ldots, Y_n\})$$

Proof of Lemma 5 Let θ be a computed answer. We will prove, by induction on m, that $\theta \in CSU_{\mathcal{S}et}(\{X_1, \ldots, X_m\}, \{Y_1, \ldots, Y_n\})$.
Base) If $m = 1$, the result holds trivially (only Action 2 or Action 3 can be fired).
Step) Let θ be a solution computed by unify_set$(\{X_1, \ldots, X_{m+1}\}, \{Y_1, \ldots, Y_n\})$, $m \geq 1$.

If $n = 1$, the result holds trivially, since only Action 3 can be fired.

Otherwise, Action 4(a) was the first action executed. Suppose i, $(1 \leq i \leq (m+1))$ was the fixed number related to such an action. Three different alternatives depending on the first subaction used in the computation of θ are possible:

1. If $\overset{1}{=}$ was used, then θ is of the form $[X_i/Y_j] \cup \theta'$, where, θ' is a solution computed by

   $$\mathsf{unify\_set}(\{X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_{m+1}\}, \{Y_1, \ldots, Y_{j-1}, Y_{j+1}, \ldots, Y_n\})$$

   for some j, $1 \leq j \leq n$. By induction hypothesis,

   $$\theta' \in CSU_{\mathcal{S}et}(\{X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_{m+1}\}, \{Y_1, \ldots, Y_{j-1}, Y_{j+1}, \ldots, Y_n\})$$

   thus, θ is a unifier for the problem at hand. It remains to prove that

   $$\theta \in CSU_{\mathcal{S}et}(\{X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_{m+1}\}, \{Y_1, \ldots, Y_{j-1}, Y_{j+1}, \ldots, Y_n\})$$

By contradiction, assume that there exists a unifier η for the sets $\{Y_1, \ldots, Y_n\}$ and $\{X_1, \ldots, X_{m+1}\}$ such that $\theta = \eta \circ \gamma$, for some substitution γ. Consider the two substitutions:

$$\begin{array}{rcl} \eta_1 & = & \{X_i/s \in \eta \ : \ s\gamma \equiv Y_j\} \\ \eta_2 & = & \{t/s \ \in \eta : \ t \not\equiv X_i \text{ or } s\gamma \not\equiv Y_j\} \end{array}$$

It is easy to see that $\eta = \eta_1 \cup \eta_2$. On the other hand, η_1 is not empty; as a matter of fact, if this were the case, then X_i/t or t/X_i would belong to η_2:[7]

- if $X_i/t \in \eta_2$, since $X_i/Y_j \in \theta$, then $t\gamma \equiv Y_j$ (a contradiction with respect to the definition of η_2);
- assuming $t/X_i \in \eta_2$, since $X_i/Y_j \in \theta$ and $\theta = \eta \circ \gamma$, then $X_i/Y_j \in \gamma$; thus $t/Y_j, X_i/Y_j \in \theta$ (a contradiction with respect to the structure of θ).

Hence, $\eta_1 = \{X_i/t\}$ and $t\gamma = Y_j$. Notice that $t \equiv Y_j$. As $t\gamma = Y_j$, then $t/Y_j \in \gamma$, and $t \not\equiv Y_j$. Now, since $X_i/t \in \eta$, and η is a unifier, then $t \notin dom(\eta)$, and $t \not\equiv X_i$. Thus $t/Y_j \in \eta \circ \gamma \equiv \theta$. But $t \not\equiv X_i$, then $t/Y_j \in \theta'$ (a contradiction with respect to the structure of θ').

[7]Note that if θ is a unifier to Problem (1), then $\{X_1, \ldots, X_m, Y_1, \ldots, Y_n\} \subseteq dom(\theta) \cup range(\theta)$.

On the other hand, since $X_i/Y_j \in \eta_1 \subseteq \eta$, then $Y_j \notin dom(\eta)$. Thus $Y_j/\cdot \notin \gamma$. Note that Y_j does not occur in η_2. Otherwise, $Z/Y_j \in \eta_2$, $Z \not\equiv Y_j$, and $Z \not\equiv X_i$. As $Y_j/\cdot \notin \gamma$, then $Z/Y_j \in \theta'$, and this is again a contradiction with respect to the structure of θ'.

Thus $\eta_1 = \{X_i/Y_j\}$. Since η is a unifier for the problem at hand, and $X_i, Y_j \notin dom(\eta_2) \cup range(\eta_2)$, then:

$$\eta_2 \in CSU_{\mathcal{S}et}(\{X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_{m+1}\}, \{Y_1, \ldots, Y_{j-1}, Y_{j+1}, \ldots, Y_n\})$$

Now $[X_i/Y_j] \cup \theta' = [X_i/Y_j] \circ \gamma \cup \eta_2 \circ \gamma$; necessarily $\theta' = \eta_2 \circ \gamma$, but this is a contradiction since θ' is a most general unifier for $\{X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_{m+1}\}$ and $\{Y_1, \ldots, Y_{j-1}, Y_{j+1}, \ldots, Y_n\}$, by induction hypothesis.

2. If $\overset{2}{\doteq}$ or $\overset{3}{\doteq}$ were used as the first action in the computation of θ, then the result would follow similarly to the previous case.

Proof of Lemma 5 □

Lemma 6 *If θ and θ' are solutions computed by two distinct nondeterministic computations of* unify_set$(\{X_1, \ldots, X_m\}, \{Y_1, \ldots, Y_n\})$*, $m, n \geq 1$, then $\theta \not\equiv \theta'$.*

Proof of Lemma 6 By induction on m.
Base) If $m = 1$, then the result follows trivially.
Step) Assume that the result holds for $m \geq 1$. Let θ and θ' be computed answers for $\{X_1, \ldots, X_{m+1}\} \doteq \{Y_1, \ldots, Y_n\}$. If the first actions executed to compute θ and θ' are $\overset{i}{\doteq}$ and $\overset{j}{\doteq}$, $i \neq j$, respectively, then the result holds trivially. Otherwise, suppose we have applied $\overset{1}{\doteq}$ as the first action in the computation of both solutions. Then:

$$\begin{aligned} \theta = {} & [X_i/Y_j] \cup \\ & \underbrace{\mathsf{unify\_set}(\{X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_{m+1}\}, \{Y_1, \ldots, Y_{j-1}, Y_{j+1}, \ldots, Y_n\})}_{\theta_1} \\ \theta' = {} & [X_i/Y_p] \cup \\ & \underbrace{\mathsf{unify\_set}(\{X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_{m+1}, \{Y_1, \ldots, Y_{p-1}, Y_{p+1}, \ldots, Y_n\})}_{\theta_2} \end{aligned}$$

where $j, p \in \{1, \ldots, n\}$. If $j \neq p$, then the result holds trivially. If $j = p$, then by induction hypothesis, $\theta_1 \not\equiv \theta_2$; since $X_i \notin dom(\theta_1) \cup dom(\theta_2) \cup range(\theta_1) \cup range(\theta_2)$, then $\theta \not\equiv \theta'$.

Applying similar reasonings, it can be proved that the result holds when $\overset{2}{=}$ or $\overset{3}{=}$ are used as first computation steps for θ and θ'.

Proof of Lemma 6 □

Proposition 1 *The algorithm* SUA *enumerates without repetitions a minimal complete set of independent unifiers to Problem (1).*

Proof of Proposition 1 It is a consequence of Lemmas 4, 5, and 6.

Proof of Proposition 1 □

Theorem 4 (Minimality) *The algorithm* SUA *is minimal with respect to all 10 sample problems described in Section 3, i.e.,* SUA *enumerates without repetitions a minimal complete set of independent unifiers, to any given unification problem belonging to one of the 10 given kinds.*

Proof of Theorem 4 The minimality of SUA for Problem (1) is proved in Proposition 1. For the other analyzed problems, see [ASD95] (for each of them, the proof follows the reasoning performed in the corresponding part of Section 3).

Proof of Theorem 4 □

4.2 Solutions Computed by a Naïve Algorithm

The aim of this section is to test an existing set-unification algorithm to show its behavior for problems presented in Section 3. We see that the algorithm is only minimal for Problem (1′). For the rest, the number of computed unifiers is strictly greater than the minimal number of independent unifiers, as is reported in the tables in Appendices B and C.

The Prolog code given in Figure 9 defines the predicate naive, and it is the core of the general set-unification algorithm presented in [Jay92]. Both algorithms do not terminate for Problem (3) (same rest variables). An extension of these algorithms covering this case can be found in [DOPR96]. For the sake of simplicity, we have not written here a Prolog code for the latter algorithm; however, we will refer to it as naive*.

```
naive(A, B) :−                        naive([T | Trest], [S | Srest]) :−
    (var(A); var(B)), !, A = B.           naive(T, S), naive(Trest, Srest).
naive(A, B) :−                        naive([T | Trest], [S | Srest]) :−
    A =.. [F | Alist],                    naive(T, S), naive([T | Trest], Srest).
    B =.. [F | Blist],                naive([T | Trest], [S | Srest]) :−
    F ≠ '.', !,                           naive(T, S), naive(Trest, [S | Srest]).
    naive_all(Alist, Blist).          naive([T | Trest], [S | Srest]) :−
                                          naive([T | New], Srest),
                                          naive(Trest, [S | New]).
```

Figure 9: A naive set-unification algorithm.

Functional symbols $\underline{\emptyset}$ and $\{\cdot \mid \cdot\}$ are represented by $[\,]$ and $[\cdot \mid \cdot]$, respectively.

Predicate naive_all is recursively defined on lists in the natural way. The naive algorithm has a minimal behavior for Problem $(1')$ only. This is stated in the following lemmata. As in Section 3, X_i, Y_j, W, and Z denote pairwise-distinct variables, while $a_1, \ldots, a_n$ denote distinct constants.

Lemma 7 *The Prolog execution of the goal*

$$:-\ \mathsf{naive}([X_1, \ldots, X_m], [a_1, \ldots, a_n])$$

returns exactly Surj(m, n) solutions, namely, it is minimal with respect to Problem $(1')$.

Lemma 8 *The number of solutions computed by the Prolog execution of the goal*

$$:-\ \mathsf{naive}([X_1, \ldots, X_m], [Y_1, \ldots, Y_n])$$

is computed by the following function

$$\begin{cases} f_2(m, 1) & = & 1 & m \geq 1 \\ f_2(1, n) & = & 1 & n > 1 \\ f_2(m, n) & = & f_2(m-1, n-1) + f_2(m, n-1) + & m, n > 1 \\ & & n f_2(m-1, n) + \sum_{i=0}^{n-2} \binom{n}{i} f_2(m-1, i+1) & \end{cases}$$

The naive program treats Problems (1) and (5) exactly the same way. For instance, when $m = 2$, $n = 2$, and $k = 3$, 95,401 solutions are computed instead of the 56 that are needed, showing its bad behavior in some cases.

Lemma 9 *The function f_8, computing the number of solutions generated by the Prolog execution of the goal*

$$:\!-\ \mathsf{naive}([X_1,\ldots,X_m \,|\, W],[Y_1,\ldots,Y_n \,|\, Z])$$

can be defined as follows:

$$\begin{cases} f_8(1,1) &=& 4 & \\ f_8(m,1) &=& 3\cdot 2^m - 2 & m>1 \\ f_8(1,n) &=& 3\cdot 2^n - 2 & n>1 \\ f_8(m,n) &=& f_8(m-1,n-1)+f_8(m,n-1)+ & m,n>1 \\ & & (n+1)f_8(m-1,n)+\sum_{i=0}^{n-2}(\binom{n}{i}+\binom{n-1}{i})f_8(m-1,i+1) & \end{cases}$$

As already sketched, the presented naive program is not sufficient to deal with Problem (3) (same rest variables). However, referring to the Prolog implementation of the complete algorithm presented in [DOPR96] (assume it is named naive*), it is easy to prove the following.

Lemma 10 *The function f_7 returning the number of solutions computed by the Prolog execution of the goal*

$$:\!-\ \mathsf{naive}^*([X_1,\ldots,X_m \,|\, Z],[Y_1,\ldots,Y_n \,|\, Z])$$

is recursively defined as follows:

$$\begin{cases} f_7(0,n) &=& 1 & n\geq 0 \\ f_7(m,0) &=& 1 & m>0 \\ f_7(m,n) &=& n(f_7(m-1,n-1)+f_7(m,n-1))+ & m>0,n>0 \\ & & (n+1)f_7(m-1,n) & \end{cases}$$

5 Conclusions

In this paper, we have presented a deep combinatorial study for a significant collection of set-unification problems, to determine the cardinality of their minimal complete set of independent unifiers. Functions computing the minimal number of unifiers for such selected problems have also been reported. Furthermore, due to the recursive nature of such functions, it has been possible to develop tables showing some values for them. Comparing these tables with the computed ones for an existing naïve algorithm, it is possible to conclude that this naïve algorithm presents a bad behavior for all analyzed problems except Problem ($1'$), for which it is minimal. These tables also show that searching minimal set-unification algorithms is an interesting line of research: since the minimal number of solutions is big by itself, it is very important to avoid repeated solutions and solutions that are instances of other solutions.

The paper also describes a new set-unification algorithm, SUA, based on the ideas behind the previous combinatorial analysis. Thus, SUA has a minimal behavior for all the presented problems, that is, it computes exactly the complete set of minimal unifiers for the problems. Although the minimality of SUA has only been proved for the problems presented in Section 3, due to their relevance—they can be used for testing any set-unification algorithm—it is reasonable to expect good behavior from SUA for any instance of the set-unification problem. Finally, we deem that SUA can advantageously replace any other known set-unification algorithms in implementations of set-based logic programming languages (for instance, in the implementation of {log} presented in [DP93]).

A Minimal Unifiers

The following tables report some numerical values for the functions computing the minimal number of most general unifiers for the sample problems presented in Section 3. The number on the x-axis denotes the value for m (the first argument).

$$(1') \qquad \{X_1, \ldots, X_m\} \doteq \{a_1, \ldots, a_n\}$$

$(1')$	1	2	3	4	5	6	7
1	1	1	1	1	1	1	1
2		2	6	14	30	62	126
3			6	36	150	540	1,806
4				24	240	1,560	8,400
5					120	1,800	16,800
6						720	15,120
7							5,040

$$(1) \qquad \{X_1, \ldots, X_m\} \doteq \{Y_1, \ldots, Y_n\}$$

(1)	1	2	3	4	5	6	7
1	1	1	1	1	1	1	1
2		2	6	14	30	62	126
3			15	48	165	558	1,827
4				184	680	2,664	11,032
5					2,945	13,080	59,605
6						63,756	320,292
7							1,748,803

$$(2') \qquad \{X_1, \ldots, X_m \,|\, Z\} \;\doteq\; \{a_1, \ldots, a_n\}$$

(2′)	1	2	3	4	5	6	7
1	2	2	2	2	2	2	2
2	4	12	28	60	124	252	508
3	6	30	126	462	1,566	5,070	15,966
4	8	56	344	1,880	9,368	43,736	195,224
5	10	90	730	5,370	36,250	228,090	1,359,130
6	12	132	1,332	12,372	106,452	856,212	6,505,812
7	14	182	2,198	24,710	259,574	2,562,182	23,928,758

$$(2) \qquad \{X_1, \ldots, X_m \,|\, Z\} \;\doteq\; \{Y_1, \ldots, Y_n\}$$

(2)	1	2	3	4	5	6	7
1	2	2	2	2	2	2	2
2	5	12	28	60	124	252	508
3	10	42	144	486	1,596	5,106	16,008
4	19	126	584	2,584	11,208	48,248	205,864
5	36	360	2,200	11,930	63,000	330,450	1,733,000
6	69	1,016	8,118	52,740	325,812	1,983,084	12,073,836
7	134	2,870	29,876	231,518	1,641,444	11,310,530	77,511,140

$$(3') \qquad \{X_1, \ldots, X_m \,|\, Z\} \;\doteq\; \{a_1, \ldots, a_n \,|\, Z\}$$

(3′)	1	2	3	4	5	6	7
1	2	4	8	16	32	64	128
2		11	30	85	248	735	2,194
3			94	308	1,104	4,210	16,538
4				1,041	3,920	16,981	80,260
5					14,006	59,412	303,428
6						221,971	1,048,054
7							4,063,382

$$(3) \qquad \{X_1, \ldots, X_m \,|\, Z\} \;\doteq\; \{Y_1, \ldots, Y_n \,|\, Z\}$$

(3)	1	2	3	4	5	6	7
1	2	4	8	16	32	64	128
2		11	30	85	248	735	2,194
3			103	356	1,269	4,678	17,735
4				1,441	5,940	25,237	110,668
5					27,631	131,142	640,513
6						685,507	3,660,958
7							21,169,037

$$(4') \qquad \{X_1, \ldots, X_m \,|\, W\} \ \doteq\ \{a_1, \ldots, a_n \,|\, Z\}$$

(4′)	1	2	3	4	5	6	7
1	4	9	18	35	68	133	262
2	7	35	125	393	1,175	3,447	10,057
3	10	79	484	2,371	10,342	42,523	169,624
4	13	141	1,257	9,209	57,269	321,661	1,700,281
5	16	221	2,606	26,091	223,496	1,683,441	11,599,186
6	19	319	4,693	60,145	671,563	6,600,163	58,356,577
7	22	435	7,680	120,443	1,674,170	20,690,167	229,717,972

$$(4) \qquad \{X_1, \ldots, X_m \,|\, W\} \ \doteq\ \{Y_1, \ldots, Y_n \,|\, Z\}$$

(4)	1	2	3	4	5	6	7
1	4	9	18	35	68	133	262
2		39	131	413	1,185	3,459	10,071
3			652	2,811	11,402	44,983	175,224
4				15,937	82,499	409,897	1,997,795
5					524,056	3,133,773	18,217,350
6						21,998,671	148,144,723
7							1,136,372,140

Problem (5) (numerically equal to Problem (5′)) would require a three-dimensional matrix to represent its values.

$$\begin{array}{lrcl} (5) & \{X_1, \ldots, X_m, Z_1, \ldots, Z_k\} & \doteq & \{Y_1, \ldots, Y_n, Z_1, \ldots, Z_k\} \\ (5') & \{X_1, \ldots, X_m, a_1, \ldots, a_k\} & \doteq & \{Y_1, \ldots, Y_n, a_1, \ldots, a_k\} \end{array}$$

Assume $k = 1$:

(5)	1	2	3	4	5	6	7
1	1	3	7	15	31	63	127
2		6	17	56	187	610	1,941
3			57	193	713	2,853	11,917
4				744	3,069	13,288	60,449
5					13,655	64,231	317,917
6						324,828	1,716,889
7							9,641,737

$k = 2$:

(5)	1	2	3	4	5	6	7
1	3	7	21	67	213	667	2,061
2		20	60	204	776	3,148	13,096
3			203	766	3,109	13,484	62,223
4				3,082	13,454	62,978	311,814
5					62,657	310,834	1,632,265
6						1,627,418	9,000,178
7							52,179,619

$k = 3$:

(5)	1	2	3	4	5	6	7
1	7	19	61	223	877	3,559	14,581
2		56	195	746	3,093	13,808	65,391
3			705	2,859	12,681	60,231	302,829
4				12,226	56,891	284,286	1,510,483
5					277,091	1,448,325	8,044,117
6						7,888,698	45,590,823
7							273,498,973

B Unifiers Computed by Naive

The algorithm naive, presented in Section 4.2, is minimal for Problem ($1'$). The following tables report some values concerning with the number of unifiers returned by this algorithm on the sample Problems (1) and (4). For Problem (3) we have used[8] the algorithm presented in [DOPR96] (named naive* in Section 4.2).

(1)	1	2	3	4	5	6	7
1	1	1	1	1	1	1	1
2		5	13	29	61	125	253
3			73	301	1,081	3,613	11,953
4				2,069	11,581	57,749	268,381
5					95,401	673,261	4,306,681
6						6,487,445	55,213,453
7							610,093,513

[8]Remember that naive does not terminate for Problem (3).

(3)	1	2	3	4	5	6	7
1	4	10	22	46	94	190	382
2	13	67	265	931	3,073	9,787	30,505
3	46	424	2,692	14,356	69,436	316,324	1,386,172
4	193	2,845	26,689	201,637	1,343,353	8,259,805	48,109,009
5	976	21,046	273,946	2,785,306	24,436,786	194,636,506	1,449,663,106
6	5,869	173,215	2,982,457	39,232,711	437,961,529	4,380,170,455	40,526,990,857
7	41,098	1,582,372	34,748,680	573,495,616	3,913,855,304	65,037,766,320	834,652,259,744

(4)	1	2	3	4	5	6	7
1	4	10	22	46	94	190	382
2		52	208	736	2,440	7,792	24,328
3			1,372	7,516	37,012	170,668	754,132
4				60,316	418,996	2,653,036	15,780,916
5					3,964,684	33,340,420	258,420,172
6						363,503,932	3,587,040,388
7							44,280,657,292

C Comparing Results

We conclude the appendix by juxtaposing some results to make the improvements of SUA vs. naive (Problems (1) and (4)) and naive* (Problem (3)) more graphic. We consider the diagonal of some of the presented tables (i.e., when $m = n$).

(1)	1	2	3	4	5	6	7
Naive	1	5	73	2,069	95,401	6,487,445	610,093,513
SUA	1	2	15	184	2,945	63,756	1,748,803
Ratio	1	2.5	4.9	11.2	32.4	101.8	348.9

(3)	1	2	3	4	5	6	7
Naive*	4	67	2,692	201,637	24,436,786	4,380,170,455	834,652,259,744
SUA	2	11	103	1,441	27,631	685,507	21,169,037
Ratio	2	6.1	26.1	139.9	884.4	6,389.7	39,428.0

(4)	1	2	3	4	5	6	7
Naive	4	52	1,372	60,316	3,964,684	363,503,932	44,280,657,292
SUA	4	39	652	15,937	524,056	21,998,671	1,136,372,140
Ratio	1	1.3	2.1	3.8	7.6	16.5	39.0

Notice that in each of these cases, the ratio grows (at least) exponentially!

Acknowledgements

We are very grateful to Giorgio Levi for giving us the possibility of working together in Pisa. We would like to thank Eugenio Omodeo, Alberto Policriti, Enrico Pontelli, Mario Rodríguez-Artalejo, Gianfranco Rossi, Alvaro Ruiz-Andino, Frieder Stolzenburg, and the anonymous referees for their wise advice. A special "thank you" to Ana Gil-Luezas for her unconditional help.

Acknowledgement of support: P. Arenas-Sánchez has been partially supported by the the Spanish National Project TIC95-0433-C03-01 "CPD" and the Esprit BRA Working Group EP-22457 "CCLII." The core of this work has been done when both the authors were affiliated at the Dipartimento di Informatica, Universitá di Pisa (IT).

References

[ASD95] P. Arenas-Sánchez and A. Dovier. Minimal set unification. Technical Report TR-6/95, Universitá di Pisa, Dipartimento di Informatica, April 1995.

[BB88] F. Baader and W. Büttner. Unification in commutative and idempotent monoids. *Theoretical Computer Science*, 56:345–352, 1988.

[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.

[BS87] W. Büttner and H. Simonis. Embedding Boolean expressions into logic programming. *Journal of Symbolic Computation*, 4:191–205, 1987.

[Büt86] W. Büttner. Unification in the data structure sets. In J. K. Siekmann, editor, *Proceedings of the Eighth International Conference on Automated Deduction*, volume 230 of *Lecture Notes in Computer Science*, pages 470–488, Berlin, 1986. Springer-Verlag.

[Cla78] K. L. Clark. Negation as failure. In H. Gallaire and J. Minker, editors, *In Logic and Databases*, pages 293–321, New York, 1978. Plenum Press.

[DOPR96] A. Dovier, E. G. Omodeo, E. Pontelli, and G. Rossi. {log}: A language for programming in logic with finite sets. *Journal of Logic Programming*, 28(1):1–44, 1996.

[DP93] A. Dovier and E. Pontelli. A WAM-based implementation of a logic language with sets. In M. Bruynooghe and J. Penjam, editors, *Proceedings of the Fifth International Symposium on Programming Language Implementation and Logic Programming*, volume 714 of *Lecture Notes in Computer Science*, pages 275–290, Berlin, 1993. Springer-Verlag.

[DPR96] A. Dovier, A. Policriti, and G. Rossi. Integrating lists, multisets, and sets in a logic programming framework. In F. Baader and K. U. Schulz, editors, *Proceedings of the First International Workshop on Frontier of Combining Systems*, pages 213–229. Kluwer Academic Publishers, 1996.

[DR93] A. Dovier and G. Rossi. Embedding extensional finite sets in CLP. In D. Miller, editor, *Proceedings of the International Logic Programming Symposium, ILPS'93*, pages 540–556, Cambridge, MA, 1993. The MIT Press.

[FH86] F. Fages and G. Huet. Complete sets of unifiers and matchers in equational theories. *Theoretical Computer Science*, 43:189–200, 1986.

[Ger94] C. Gervet. Conjunto: constraint logic programming with finite set domains. In M. Bruynooghe, editor, *Proceedings of the International Logic Programming Symposium, ILPS'94*, pages 339–358, Cambridge, MA, 1994. The MIT Press.

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

[Jay92] B. Jayaraman. Implementation of subset-equational programs. *Journal of Logic Programming*, 12(4):299–324, 1992.

[JM94] J. Jaffar and M. J. Maher. Constraint logic programming: a survey. *Journal of Logic Programming*, 19/20:503–581, 1994.

[KN86] D. Kapur and P. Narendran. NP-completeness of the set unification and matching problems. In J. H. Siekmann, editor, *Proceedings of the Eighth International Conference on Automated Deduction*, volume 230 of *Lecture Notes in Computer Science*, pages 489–495, Berlin, 1986. Springer-Verlag.

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

[Knu68] D. E. Knuth. In *The Art of Computer Programming*, volume 1 of *Fundamental Algorithms*, Reading, MA, 1968. Addison-Wesley.

[LL91] B. Legeard and E. Legros. Short overview of the CLPS system. In J. Maluszynsky and M. Wirsing, editors, *Proceedings of the Third International Symposium on Programming Language Implementation and Logic Programming*, volume 528 of *Lecture Notes in Computer Science*, pages 431–433, Berlin, 1991. Springer-Verlag.

[Sie90] J. H. Siekmann. Unification theory. In C. Kirchner, editor, *Unification*, London, 1990. Academic Press.

[Sto96] F. Stolzenburg. Membership-constraint and complexity in logic programming with sets. In F. Baader and K. U. Schulz, editors, *Frontiers of Combining Systems*, pages 285–302, Norwell, MA, 1996. Kluwer Academic Publishers.