

# A Uniform Approach to Constraint-Solving for Lists, Multisets, Compact Lists, and Sets

AGOSTINO DOVIER and CARLA PIAZZA

Università di Udine

and

GIANFRANCO ROSSI

Università di Parma

Lists, multisets, and sets are well-known data structures whose usefulness is widely recognized in various areas of computer science. They have been analyzed from an axiomatic point of view with a parametric approach in Dovier et al. [1998], where the relevant unification algorithms have been developed. In this article, we extend these results considering more general constraints, namely, equality and membership constraints and their negative counterparts.

Categories and Subject Descriptors: D.3.2 [**Programming Languages**]: Language Classifications—*Constraint and logic languages*; D.3.3 [**Programming Languages**]: Language Constructs and Features; F.4.1 [**Mathematical Logic and Formal Languages**]: Mathematical Logic—*Logic and constraint programming, set theory*; I.2.3 [**Artificial Intelligence**]: Deduction and Theorem Proving—*Logic programming*

General Terms: Theory, Algorithms

Additional Key Words and Phrases: Membership and equality constraints, lists, multisets, compact lists, sets

## ACM Reference Format:

Dovier, A., Piazza, C., and Rossi, G. 2008. A Uniform approach to constraint-solving for lists, multisets, compact lists, and sets. *ACM Trans. Comput. Logic* 9, 3, Article 15 (June 2008), 30 pages. DOI = 10.1145/1352582.1352583 <http://doi.acm.org/10.1145/1352582.1352583>

## 1. INTRODUCTION

Programming and specification languages allow us to represent information by means of data structures, each of them characterized by a specific

This research has been partially supported by MURST project PRIN 2005015491.

Author's addresses: A. Dovier and C. Piazza, Università di Udine, Dip. di Matematica e Informatica, Via delle Scienze 206, 33100 Udine, Italy; email: {dovier,piazza}@dimi.uniud.it; G. Rossi, Università di Parma, Dip. di Matematica, Parco Area delle Scienze 53/A, 43100 Parma, Italy; email: gianfranco.rossi@unipr.it

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or direct commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or [permissions@acm.org](mailto:permissions@acm.org). © 2008 ACM 1529-3785/2008/06-ART15 \$5.00 DOI 10.1145/1352582.1352583 <http://doi.acm.org/10.1145/1352582.1352583>

ACM Transactions on Computational Logic, Vol. 9, No. 3, Article 15, Publication date: June 2008.

organization of the elements involved and by a corresponding access policy. In this article we consider the following structures, which represent distinct though strongly related abstractions: lists, multisets, compact lists, and sets.

Each of these four data structures contains an arbitrary (possibly empty) collection of elements of any type, where each element can be either an elementary data object or a composite object. Let us define an *aggregate* as a data structure with this property. The basic difference among the four considered aggregates lies in the specific handling of order and/or repetitions of elements. *Lists* are ordered collections of elements, where duplicates are allowed. *Multisets*, often called *bags* in the literature, are lists in which the ordering is irrelevant. *Compact lists* are lists in which contiguous occurrences of the same element are collapsed into a single element. Finally, in *sets* both ordering and duplicates are not relevant.

The importance of these data structures is widely recognized in various areas of computer science. Lists are the classical example in use to introduce dynamic data structures in imperative programming languages. They are the fundamental data structure in functional and logic languages. Sets are the main data structure used in specification languages (e.g., in Z [Potter et al. 1996]) and in high-level declarative programming languages [Beeri et al. 1991; Dovier et al. 1996; Gervet 1997; Hill and Lloyd 1994]; moreover, imperative programming languages may take advantage from the set data abstraction (e.g., SETL [Schwartz et al. 1986]). Multisets emerge as the most natural data structure in several areas, ranging from coordination languages [Banatre and Métayer 1993] to Database theory [Grumbach and Milo 1996], from membrane and DNA computing modeling [Păun 2000] to linear logic [Tzouvaras 1998]. The notion of compact list is much less developed and some examples of its application are suggested in Dovier et al. [1998].

Lists, multisets, compact lists, and sets have been analyzed from an axiomatic point of view. In Dovier et al. [1998], they have been studied in the context of Constraint Logic Programming languages, where these aggregates are represented as terms by means on different constructors. Each aggregate is associated with a theory which specifies the properties of the aggregate constructor symbol.

In Dovier et al. [1998], *equalities* between terms in each of the four theories were studied. In particular, the unification problems in the equational theories, which describe the properties of the four aggregates, were solved by providing unification algorithms for all of them. NP-unification algorithms for sets and multisets are also presented in Aliffi et al. [1999] and Dantsin and Voronkov [1999].

In this article we extend the results presented in Dovier et al. [1998] to the case of more general *constraints*. The constraints we consider are conjunctions of literals based on both equality and membership predicate symbols. For each aggregate, we introduce a first-order theory and we investigate the problem of deciding whether a constraint is satisfiable in each model of the theory. We base our decidability results on the introduction of a standard model and a solved form for each aggregate. These results allow us to solve the constraint

satisfiability problems by applying rewriting procedures which map satisfiable constraints into solved-form constraints.

The article is organized as follows. In Section 2 we briefly discuss the existing results for similar problems. After the preliminary definitions of Section 3, in Section 4 we recall from Dovier et al. [1998] the first-order and equational theories of the four aggregates. In Section 5 we define the notion of constraint and we identify the standard models for the theories used to describe the considered aggregates. To ease the presentation, we choose the multiset theory as the working theory and we briefly point out the differences with the other theories. We show that satisfiability of constraints in standard models is equivalent to satisfiability in any model. Then we define the notion of solved form for our constraints, and we prove that solved-form constraints are satisfiable in the proposed standard models. In Section 6 we describe the constraint rewriting procedures used to eliminate all constraints not in solved form. We use these procedures in Section 7 to solve the general satisfiability problem for the considered constraints. Some conclusions are drawn in Section 8.

## 2. RELATED WORK

The problem of *set and multiset unification* has been tackled by several authors, using different representations (see Dovier et al. [2001] for a survey on the set unification problem). These problems are often reduced to *ACI* and *AC* unification problems, respectively (see, e.g., Büttner [1986] and Livesey and Siekmann [1976]). In these cases, a *union-based representation* is usually employed, where the *union* binary function symbols  $\cup$  and  $\uplus$  are used as the set and multiset constructors, respectively. The operators  $\cup$  and  $\uplus$  fulfill associativity (*A*) and commutativity (*C*). Moreover,  $\cup$  is idempotent (*I*). In order to deal with *nested* sets and multisets, the unary function symbols  $\{\cdot\}$  and  $\{\{\cdot\}\}$  are also included. They act as *singleton* constructors for sets and multisets, respectively. Thus, the set  $\{a, b, c\}$  can be represented as a term of the form  $\{a\} \cup \{b\} \cup \{c\}$  and the multiset  $\{[a, b, b, c]\}$  can be represented as  $\{[a]\} \uplus \{[b]\} \uplus \{[b]\} \uplus \{[c]\}$ . Since  $\{\cdot\}$  and  $\{\{\cdot\}\}$  are *free* function symbols, they do not fulfill any particular axiom (see, e.g., Baader and Nipkow [1998]). Equational theories which also allow one to deal with nested sets and multisets are called *general ACI* and *general AC*, respectively.

Unification and disunification algorithms for general *ACI* and *AC* theories can be obtained by exploiting both the results for simpler cases (unification with constants—Baader and Büttner [1988]) and the combining approach developed in Baader and Schulz [1995, 1996]. This approach, however, due to its generality, tends to produce a huge number of failing nondeterministic computation branches, which can be pruned using more ad hoc procedures.

The general problem of solving disequations with respect to a given equational theory has also been addressed in Bückert [1988], where a technique to transform disequations into universally quantified unification problems was presented. The method described in Bückert [1988] cannot be applied in the case of theories over sets, since it can generate undecidable formulas. In fact, existentially quantified formulas containing equations and disequations are

decidable in the case of *AC* theories, as a corollary of the results presented in Comon [1993]. Unfortunately, the same results cannot be applied to *ACI* theories, and hence to sets. These theories were studied in Dovier et al. [2004], where constraint solving procedures were developed.

As far as membership and notmembership are concerned, we are not aware of studies that extend those equational theories to encompass this kind of constraints. Actually, for sets, both membership and notmembership could be easily defined in terms of equality and disequality constraints:  $t \in s$  can be defined as  $\{t\} \cup s = s$  and  $t \notin s$  as  $\{t\} \cup s \neq s$ . Conversely, for multisets, membership  $t \in s$  can be defined as  $\exists X (([t] \uplus X = s))$ , where  $X$  is a new variable, while  $t \notin s$  can be defined as  $\forall X (([t] \uplus X \neq s))$ , that is, using a formula with universal quantification. Note that  $t \notin s$  could be simplified to  $s = \{[X]\} \cup R \wedge X \neq t \wedge t \notin R$ . On the contrary,  $t \notin R$ , where  $R$  is a variable, is not reducible to a system of equalities and disequalities. For lists and compact lists of unknown length, both membership and notmembership cannot be defined in terms of equality and disequality constraints.

The union-based representation can also be used for lists and compact lists, where the union operator is associative for lists, and associative and partially idempotent for compact lists.

An alternative approach consists of considering a *list-like representation* based on an element insertion constructor for each of the four aggregates (see Section 4). In Dovier et al. [2000b] some comparisons between the union-like and list-like representations were presented and they highlight the different expressive powers. In particular, it turns out that the singleton operator is not expressible using existentially quantified formulas with union. Furthermore, the list-based representation was shown to be more natural for dealing with membership constraints. General constraint solving procedures based on this approach, though limited to the case of sets, were presented in [Dovier and Rossi 1993; Dovier et al. 2000b]. In Dovier et al. [1998] we considered the four data structures considered in this article, using the list-like representation for all of them, but limitedly to the case of unification. Note that constraints on sets are particular cases of formulas of multilevel syllogistics, studied in Cantone et al. [2001], where axioms for sets are not simply equational axioms. However, Cantone et al. [2001] were mainly concerned with decidability results rather than with constraint-solving procedures.

In this article we make use of the *list-like representation* constraint-solving procedures (that can be used as decision procedures, as well) for constraints involving equality and membership literals.

### 3. PRELIMINARY NOTIONS

Basic knowledge of first-order logic (e.g., Chang and Keisler [1973]; Enderton [1973]) is assumed. We fix some notations and recall some basic notions that will be used throughout the article.

A first-order language  $\mathcal{L} = \langle \Sigma, \mathcal{V} \rangle$  is defined by a *signature*  $\Sigma = \langle \mathcal{F}, \Pi \rangle$  composed by a set  $\mathcal{F}$  of constant and function symbols, by a set  $\Pi$  of predicate symbols, and by a denumerable set  $\mathcal{V}$  of variables. The capital letters  $X, Y, Z$ ,

etc., are used to represent variables, while  $f$ ,  $g$ , etc., represent constant and function symbols, and  $p$ ,  $q$ , etc., represent predicate symbols.  $\bar{X}$  and  $\bar{t}$  denote a (possibly empty) sequence of variables and terms, respectively.

The set of first-order terms (ground terms) built on  $\mathcal{F}$  and  $\mathcal{V}$  ( $\mathcal{F}$ , respectively) are denoted by  $T(\mathcal{F}, \mathcal{V})$  ( $T(\mathcal{F})$ , respectively). The number of occurrences of constant and function symbols in a term  $t$  is denoted by  $size(t)$ , while  $FV(\bar{t})$  is the set of all the variables which occur in the terms  $\bar{t}$ . If  $\varphi$  is a first-order formula,  $FV(\varphi)$  is the set of free variables in  $\varphi$ . A formula is closed if it has no free variables.  $\exists\varphi$  ( $\forall\varphi$ ) is used to denote the existential (universal, respectively) closure of the formula  $\varphi$ , namely,  $\exists X_1 \dots \exists X_n \varphi$  ( $\forall X_1 \dots \forall X_n \varphi$ , respectively), where  $\{X_1, \dots, X_n\} = FV(\varphi)$ . An axiom is a closed first-order formula. If  $\Theta = \{\varphi_1, \dots, \varphi_n\}$  is a set of axioms and  $A_1, \dots, A_n$  are names for the axioms  $\varphi_1, \dots, \varphi_n$ , we refer to  $\Theta$  simply as  $A_1 \dots A_n$ . In this work we assume that any first-order theory  $\mathcal{T}$  includes standard equality axioms:  $(=1) \forall X (X = X)$  and  $(=2) \forall X \forall Y ((X = Y) \rightarrow (\varphi \rightarrow \varphi'))$ , where  $\varphi$  is any first-order formula,  $X$  and  $Y$  are free in  $\varphi$ , and  $\varphi'$  is obtained from  $\varphi$  by replacing zero or more occurrences of  $X$  with  $Y$  [Chang and Keisler 1973; Enderton 1973].

An *equational axiom* is a formula of the form  $\forall(\ell = r)$ , where  $\ell$  and  $r$  are terms. An *equational theory*  $E$  is an axiomatization whose axioms are equational axioms. Given two terms  $\ell$  and  $r$ , we write  $\ell \approx_E r$  if the axioms in  $E$  can prove that  $\ell$  is equal to  $r$ . A *system of equations*  $S$  is a conjunction of equations  $\ell_1 = r_1 \wedge \dots \wedge \ell_n = r_n$ . An *E-solution* (a solution, when the context is clear) of  $S$  is a substitution  $\sigma$ , which replaces variables with ground terms, such that for all  $i \in \{1, \dots, n\}$  it holds  $\sigma(\ell_i) \approx_E \sigma(r_i)$ .

Given  $\mathcal{L} = \langle \Sigma, \mathcal{V} \rangle$ , a  $\Sigma$ -*structure* is a pair  $\mathcal{A} = \langle A, I \rangle$ , where  $A \neq \emptyset$  is the domain and  $I$  is the interpretation function of each constant, function, and predicate symbols of  $\Sigma$  on  $A$ . A *valuation*  $\sigma$  is a function from a subset of the set of variables  $\mathcal{V}$  to  $A$ . When  $\Sigma$  is given,  $\sigma$  can be uniquely extended to terms, and allows to assign truth values to formulas. A valuation  $\sigma$  is said to be *successful* for  $\varphi$  if  $\sigma(\varphi) = \text{true}$  (briefly,  $\mathcal{A} \models \sigma(\varphi)$ ). A formula  $\varphi$  is *satisfiable* in  $\mathcal{A}$ , denoted by  $\mathcal{A} \models \exists\varphi$ , if there exists a valuation  $\sigma$  such that  $\mathcal{A} \models \sigma(\varphi)$ . We say that  $\mathcal{A} \models \varphi$  if for every valuation  $\sigma$  from  $FV(\varphi)$  to  $A$  it holds that  $\mathcal{A} \models \sigma(\varphi)$ . Two formulas  $C_1$  and  $C_2$  are *equisatisfiable* in  $\mathcal{A}$  if:  $C_1$  is satisfiable in  $\mathcal{A}$  if and only if  $C_2$  is satisfiable in  $\mathcal{A}$ . A structure  $\mathcal{A}$  is a *model* of a theory  $\mathcal{T}$  if  $\mathcal{A} \models \varphi$  for all  $\varphi$  in  $\mathcal{T}$ . We say that  $\mathcal{T} \models \varphi$  if  $\mathcal{A} \models \varphi$  for all models  $\mathcal{A}$  of  $\mathcal{T}$ .

#### 4. THE THEORIES

We recall from Dovier et al. [1998] the first-order axiomatic theories for the four aggregates. Each theory has its own signature. Precisely,  $\Pi$  is  $\{=, \in\}$  and  $\mathcal{F}$  contains (at least) the constant symbol `nil` and exactly one among the following binary function symbols:

$$\begin{array}{ll} [\cdot | \cdot] & \text{for lists,} \\ [[\cdot | \cdot]] & \text{for compact lists,} \end{array} \quad \begin{array}{ll} \{\{\cdot | \cdot\}\} & \text{for multisets,} \\ \{\cdot | \cdot\} & \text{for sets.} \end{array}$$

Moreover, each of the four signatures can contain an arbitrary number of fresh constant and function symbols. The four function symbols above are referred as *aggregate constructors*. The empty list, the empty multiset, the empty compact

list, and the empty set are all denoted by the constant symbol `nil`. We simplify syntactic notations for terms built using the aggregate constructors in a standard way. In particular, the (multiset) term  $\{[s_1 | \{[s_2 | \dots \{[s_n | t]\} \dots]\}\}$  will be denoted by  $\{[s_1, \dots, s_n | t]\}$  or by  $\{[s_1, \dots, s_n]\}$  when  $t$  is `nil`. The same conventions will be exploited also for the other aggregates.

In the following sections we introduce the axioms we need to define a theory for each aggregate. Then the four theories are presented in Section 4.5.

#### 4.1 Lists

The language  $\mathcal{L}_{List}$  is defined as  $\langle \Sigma_{List}, \mathcal{V} \rangle$ , where  $\Sigma_{List} = \langle \mathcal{F}_{List}, \Pi \rangle$ ,  $[\cdot | \cdot]$  and `nil` are in  $\mathcal{F}_{List}$ , and  $\Pi = \{=, \in\}$ . We recall that  $\mathcal{F}_{List}$  can contain other constant and function symbols. The first-order theory *List* of lists is shown below.

(K)	$\forall X Y_1 \dots Y_n \quad (X \notin f(Y_1, \dots, Y_n))$ $f \in \mathcal{F}_{List}, f \text{ is not } [\cdot   \cdot]$
(W)	$\forall Y V X \quad (X \in [Y   V] \leftrightarrow X \in V \vee X = Y)$
(F <sub>1</sub> )	$\forall X_1 \dots X_n Y_1 \dots Y_n \left( \begin{array}{l} f(X_1, \dots, X_n) = f(Y_1, \dots, Y_n) \\ \rightarrow X_1 = Y_1 \wedge \dots \wedge X_n = Y_n \end{array} \right)$ $f \in \mathcal{F}_{List}$
(F <sub>2</sub> )	$\forall X_1 \dots X_m Y_1 \dots Y_n \quad (f(X_1, \dots, X_m) \neq g(Y_1, \dots, Y_n))$ $f, g \in \mathcal{F}_{List}, f \text{ is not } g$
(F <sub>3</sub> )	$\forall X \quad (X \neq t[X])$ where $t[X]$ denotes a term $t$ having $X$ as proper subterm

The three axiom schemas (F<sub>1</sub>), (F<sub>2</sub>), and (F<sub>3</sub>) (called *freeness axioms*, or *Clark's equality axioms*—see Clark [1978]) were originally introduced by Mal'cev [1971]. Since  $[\cdot | \cdot]$  belongs to  $\mathcal{F}_{List}$ , axiom schema (F<sub>1</sub>) holds for  $[\cdot | \cdot]$  as a particular case. (F<sub>3</sub>) states that there is no term which is a proper subterm of itself (occurs check). Notice that (K) implies that  $\forall X (X \notin \text{nil})$ .

#### 4.2 Multisets

The language  $\mathcal{L}_{MSet}$  is defined as  $\langle \Sigma_{MSet}, \mathcal{V} \rangle$ , where  $\Sigma_{MSet} = \langle \mathcal{F}_{MSet}, \Pi \rangle$ ,  $\{[\cdot | \cdot]\}$  and `nil` are in  $\mathcal{F}_{MSet}$ , and  $\Pi = \{=, \in\}$ . A theory of multisets—called *MSet*—can be obtained from the theory of lists shown above. The constructor  $[\cdot | \cdot]$  is replaced by the constructor  $\{[\cdot | \cdot]\}$  in axiom schema (K) and axiom (W). The behavior of this new symbol is regulated by the following equational axiom

$$(E_p^m) \quad \forall X Y Z \quad \{[X, Y | Z]\} = \{[Y, X | Z]\} \quad (\text{permutativity})$$

which intuitively states that the order of elements in a multiset is immaterial. Axiom schema (F<sub>1</sub>) does not hold for multisets, when  $f$  is  $\{[\cdot | \cdot]\}$ . It is replaced by axiom schema (F<sub>1</sub><sup>m</sup>):

$$(F_1^m) \quad \forall X_1 \dots X_n Y_1 \dots Y_n \left( \begin{array}{l} f(X_1, \dots, X_n) = f(Y_1, \dots, Y_n) \\ \rightarrow X_1 = Y_1 \wedge \dots \wedge X_n = Y_n \end{array} \right)$$

for any  $f \in \mathcal{F}_{MSet}, f \text{ is not } \{[\cdot | \cdot]\}$



The theory  $KW E_p^m F_1^m F_2 F_3$ , however, is not endowed with a general criterion for establishing equality and disequality between multisets. To obtain it, the following *multiset extensionality* property is introduced: *two multisets are equal if and only if they have the same number of occurrences of each element, regardless of their order*. The axiom proposed in Dovier et al. [1998] to force this property is the following:

$$(E_k^m) \quad \forall Y_1 Y_2 V_1 V_2 \left( \begin{array}{l} \{[Y_1 | V_1]\} = \{[Y_2 | V_2]\} \leftrightarrow \\ (Y_1 = Y_2 \wedge V_1 = V_2) \vee \\ \exists Z (V_1 = \{[Y_2 | Z]\} \wedge V_2 = \{[Y_1 | Z]\}) \end{array} \right)$$

Axiom  $(E_k^m)$  implies  $(E_p^m)$ . Axiom schema  $(F_3^m)$  is also introduced:

$$(F_3^m) \quad \forall X_1 \dots X_m Y_1 \dots Y_n X \left( \begin{array}{l} \{[X_1, \dots, X_m | X]\} = \{[Y_1, \dots, Y_n | X]\} \\ \rightarrow \{[X_1, \dots, X_m]\} = \{[Y_1, \dots, Y_n]\} \end{array} \right)$$

It reinforces the acyclicity condition imposed by standard axiom schema  $(F_3)$ . As a matter of fact,  $X \neq \{[a, b, b | X]\}$  follows from  $(F_3)$ . Axiom schema  $(F_3^m)$  states for instance that, since  $\{[a, a, b]\} \neq \{[a, b, b]\}$ , then  $\{[a, a, b | X]\} \neq \{[a, b, b | X]\}$ . This property is not a consequence of the remaining part of the theory.

### 4.3 Compact Lists

The language  $\mathcal{L}_{CList}$  is defined as  $\mathcal{L}_{CList} = \langle \Sigma_{CList}, \mathcal{V} \rangle$ , where  $\Sigma_{CList} = \langle \mathcal{F}_{CList}, \Pi \rangle$ ,  $[[ \cdot | \cdot ]]$  and `nil` are in  $\mathcal{F}_{CList}$ , and  $\Pi = \{=, \in\}$ . The theory of *compact lists*—called **CList**—is obtained from the theory of lists with only a few changes. The list constructor symbol is replaced by the binary compact list constructor  $[[ \cdot | \cdot ]]$  in  $(K)$  and  $(W)$ . The behavior of this symbol is regulated by the equational axiom

$$(E_a^c) \quad \forall X Y \quad [[X, X | Y]] = [[X | Y]] \quad (absorption)$$

which, intuitively, states that contiguous duplicates in a compact list are immaterial. As for multisets, we introduce a general criterion for establishing both equality and disequality between compact lists. This is obtained by introducing the following axiom:

$$(E_k^c) \quad \forall Y_1 Y_2 V_1 V_2 \left( \begin{array}{l} [[Y_1 | V_1]] = [[Y_2 | V_2]] \leftrightarrow \\ (Y_1 = Y_2 \wedge V_1 = V_2) \vee \\ (Y_1 = Y_2 \wedge V_1 = [[Y_2 | V_2]]) \vee \\ (Y_1 = Y_2 \wedge [[Y_1 | V_1]] = V_2) \end{array} \right)$$

Axiom  $(E_a^c)$  is implied by  $(E_k^c)$ . Axiom schema  $(F_1)$  is replaced by axiom schema  $(F_1^c)$ :

$$(F_1^c) \quad \forall X_1 \dots X_n Y_1 \dots Y_n \left( \begin{array}{l} f(X_1, \dots, X_n) = f(Y_1, \dots, Y_n) \\ \rightarrow X_1 = Y_1 \wedge \dots \wedge X_n = Y_n \end{array} \right) \\ \text{for any } f \in \mathcal{F}_{CList}, f \text{ is not } [[ \cdot | \cdot ]]$$

The freeness axiom ( $F_3$ ) needs to be suitably modified. The introduction of ( $F_3$ ) is motivated by the requirement of finding solutions to equality constraints over  $\Sigma$ -structures whose domain is based on the Herbrand Universe, where each term is modeled by a finite tree. As opposed to lists and multisets, an equation such as  $X = [[\text{nil} \mid X]]$  admits a successful valuation over compact lists. Precisely, a valuation that binds  $X$  to the term  $[[\text{nil} \mid t]]$ , where  $t$  is any term. Therefore, axiom schema ( $F_3$ ) is weakened as follows:

$$(F_3^c) \quad \forall X \ (X \neq t[X]) \\ \text{unless: } t \text{ is of the form } [[t_1, \dots, t_n \mid X]], \text{ with } n > 0, \\ X \notin \text{FV}(t_1, \dots, t_n), \text{ and } t_1 = \dots = t_n$$

#### 4.4 Sets

The language  $\mathcal{L}_{Set}$  is defined as  $\mathcal{L}_{Set} = \langle \Sigma_{Set}, \mathcal{V} \rangle$ , where  $\Sigma_{Set} = \langle \mathcal{F}_{Set}, \Pi \rangle$ ,  $\{\cdot \mid \cdot\}$  and  $\text{nil}$  are in  $\mathcal{F}_{Set}$ , and  $\Pi = \{=, \in\}$ . The last theory we consider is the theory Set of sets. Sets satisfy both the *permutativity* and the *absorption properties* which, in the case of  $\{\cdot \mid \cdot\}$ , can be rewritten as follows:

$$(E_p^s) \quad \forall XYZ \ \{X, Y \mid Z\} = \{Y, X \mid Z\} \\ (E_a^s) \quad \forall XY \ \{X, X \mid Y\} = \{X \mid Y\}$$

A criterion for testing equality (and disequality) between sets is obtained by merging the multiset equality axiom ( $E_k^m$ ) and the compact list equality axiom ( $E_k^c$ ):

$$(E_k^s) \quad \forall Y_1 Y_2 V_1 V_2 \left( \begin{array}{l} \{Y_1 \mid V_1\} = \{Y_2 \mid V_2\} \leftrightarrow \\ (Y_1 = Y_2 \wedge V_1 = V_2) \vee \\ (Y_1 = Y_2 \wedge V_1 = \{Y_2 \mid V_2\}) \vee \\ (Y_1 = Y_2 \wedge \{Y_1 \mid V_1\} = V_2) \vee \\ \exists Z (V_1 = \{Y_2 \mid Z\} \wedge V_2 = \{Y_1 \mid Z\}) \end{array} \right)$$

According to ( $E_k^s$ ), duplicates and ordering of elements in sets are immaterial. Thus, ( $E_k^s$ ) implies the equational axioms ( $E_p^s$ ) and ( $E_a^s$ ). In Dovier et al. [1998] it was also proved that they are equivalent in all  $\Sigma$ -structures where the domain is isomorphic to a subset of the set of ground terms (Herbrand Universe). The theory Set also contains axioms ( $K$ ), ( $W$ ) with  $[\cdot \mid \cdot]$  replaced by  $\{\cdot \mid \cdot\}$ , and axiom schemas ( $F_2$ ). Axiom schema ( $F_1$ ) is replaced by

$$(F_1^s) \quad \forall X_1 \dots X_n Y_1 \dots Y_n \left( \begin{array}{l} f(X_1, \dots, X_n) = f(Y_1, \dots, Y_n) \\ \rightarrow X_1 = Y_1 \wedge \dots \wedge X_n = Y_n \end{array} \right) \\ \text{for any } f \in \mathcal{F}_{Set}, f \text{ is not } \{\cdot \mid \cdot\}$$

The modification of axiom schema ( $F_3$ ) for sets simplifies the one used for compact lists:

$$(F_3^s) \quad \forall X \ (X \neq t[X]) \\ \text{unless: } t \text{ is of the form } \{t_1, \dots, t_n \mid X\} \text{ and } X \in \text{FV}(t_1, \dots, t_n)$$



Table I. Axioms for the Four Theories (from Left to Right, the Name of the First-Order Theory, the First-Order Axiom Schemas, and the Name of the Equational Theories)

Theory	Empty	With	Equality		Herbr.	Acycl.		Eq. theory
<i>List</i>	( <i>K</i> )	( <i>W</i> )	$(F_1)$		$(F_2)$	$(F_3)$		$E_{List}$
<i>MSet</i>	( <i>K</i> )	( <i>W</i> )	$(E_k^m)$	$(F_1^m)$	$(F_2)$	$(F_3)$	$(F_3^m)$	$E_{MSet}$
<i>CList</i>	( <i>K</i> )	( <i>W</i> )	$(E_k^c)$	$(F_1^c)$	$(F_2)$	$(F_3^c)$		$E_{CList}$
<i>Set</i>	( <i>K</i> )	( <i>W</i> )	$(E_k^s)$	$(F_1^s)$	$(F_2)$	$(F_3^s)$		$E_{Set}$

Table II. Axioms for the Four Equational Theories (from Left to Right the Name of the Equational Theory and the Equational Axioms)

Eq. theory	Perm.	Abs.
$E_{List}$		
$E_{MSet}$	$(E_p^m)$	
$E_{CList}$		$(E_a^c)$
$E_{Set}$	$(E_p^s)$	$(E_a^s)$

#### 4.5 Equational Theories

We have shown that each aggregate constructor is precisely characterized by zero, one, or two equational axioms. In particular, lists do not require any axiom, multisets need the permutativity axiom ( $E_p^m$ ), compact lists use the absorption axiom ( $E_a^c$ ), and sets are characterized by both the permutativity axiom ( $E_p^s$ ) and the absorption axiom ( $E_a^s$ ).

Tables I and II summarize the axiomatizations of the four theories.

### 5. CONSTRAINTS, STANDARD MODELS, AND SOLVED FORM

In this section we first introduce the set of formulas we are interested in. These formulas are called *constraints* and are basically the existentially quantified formulas of the languages described in the previous section.

**Definition 5.1.** (Constraints). Let  $\mathbb{T}$  be either *List* or *MSet* or *CList* or *Set*. A  $\mathbb{T}$ -constraint  $C_{\mathbb{T}}$  is a conjunction of atomic  $\mathcal{L}_{\mathbb{T}}$ -formulas or negation of atomic  $\mathcal{L}_{\mathbb{T}}$ -formulas of the form  $s \pi t$ , where  $\pi \in \Pi$ , and  $s, t \in T(\mathcal{F}_{\mathbb{T}}, \mathcal{V})$ .

Throughout the article we will use the following terminology to refer to particular kinds of constraints: *equality (disequality) constraints* are conjunctions of atomic formulas of the form  $s = t$  ( $s \neq t$ , respectively), while *membership (not-membership) constraints* are conjunctions of membership atoms (negative membership literals, respectively), that is, formulas of the form  $s \in t$  ( $s \notin t$ , respectively).

We are interested in the problem of deciding whether a formula over one of the aggregates is *satisfiable in each model* of the theory of that aggregate. We start tackling this problem by introducing *standard models* for the four theories and giving a general notion of *solved form* for constraints. We prove

that (1) the satisfiability of a constraint in the standard model is equivalent to its satisfiability in each model (i.e., the theory and the standard model correspond on the class of considered constraints); (2) solved-form constraints are always satisfiable in the corresponding standard model.

### 5.1 Standard Models

Each aggregate constructor is characterized by its equational theory ( $E_{List}$ ,  $E_{MSet}$ ,  $E_{CList}$ , and  $E_{Set}$ ). Using the appropriate equational theory we can define standard models for the first-order theories *List*, *MSet*, *CList*, and *Set*. Each model is obtained as a partition of the Herbrand Universe. To simplify our presentation, we describe in details only the multisets case.

*Definition 5.2.* The  $\Sigma$ -structure  $\mathcal{MSET}$  for *MSet* is defined as follows.

- (1) The *domain* of the  $\Sigma$ -structure is the quotient  $T(\mathcal{F}_{MSet}) / \equiv_{MSet}$  of the Herbrand Universe  $T(\mathcal{F}_{MSet})$  over the smallest congruence relation  $\equiv_{MSet}$  induced by the equational theory  $E_{MSet}$  on  $T(\mathcal{F}_{MSet})$ .
- (2) The interpretation of a term  $t$  is its equivalence class  $(t)$  with respect to  $\equiv_{MSet}$ .
- (3)  $=$  is interpreted as the identity on the domain  $T(\mathcal{F}_{MSet}) / \equiv_{MSet}$ .
- (4) The interpretation of membership is:  $(t) \in (s)$  is true if and only if there is a term of the form  $\{[t_1, \dots, t_n, t \mid r]\}$  in  $(s)$ .

In Lemma A.2 we prove that  $\mathcal{MSET}$  is a model of *MSet*. We call it the *standard model* for *MSet*. For the other aggregates the names of the models are *CLIST*, *CLIST*, and *SET*. The definition of these models is obtained by using the appropriate equational theory, in the very same way as shown for multisets.

*Definition 5.3.* (Jaffar and Maher 1994). Let  $\mathcal{L} = \langle \Sigma, \mathcal{V} \rangle$  be a first-order language,  $\mathcal{T}$  be a theory on  $\mathcal{L}$ ,  $\mathcal{A}$  be a  $\Sigma$ -structure on  $\mathcal{L}$ , and  $\mathcal{C}$  be a class of first-order formulas on  $\mathcal{L}$ . The theory  $\mathcal{T}$  and the structure  $\mathcal{A}$  *correspond* on the class  $\mathcal{C}$  if, for each  $\varphi \in \mathcal{C}$ , we have that  $\mathcal{T} \models \exists \varphi$  if and only if  $\mathcal{A} \models \exists \varphi$ .

This property means that if  $\varphi$  is an element of  $\mathcal{C}$  and  $\varphi$  is satisfiable in  $\mathcal{A}$ , then it is satisfiable in all the models of  $\mathcal{T}$ . We prove that *MSet* and the standard model  $\mathcal{MSET}$  correspond on the class of constraints defined in Definition 5.1. In the proof we use some basic results which can be found in the Appendix (Lemmas A.1–A.3). The proofs for the other theories are similar. Intuitively all these proofs exploit two facts: our standard models are “minimal” models for the theories (i.e., they are contained in each model) and the formulas are only existentially quantified.

**THEOREM 5.4.** *The theory *MSet* and the model  $\mathcal{MSET}$  correspond on *MSet*-constraints.*

**PROOF.** From Lemma A.2 it follows that  $\mathcal{MSET}$  is a model of *MSet*, namely, that if  $C$  is a first-order formula and  $MSet \models C$ , then  $\mathcal{MSET} \models C$ .

On the other hand, if  $\exists C$  is a formula with only existential quantifiers, then  $\mathcal{MSET} \models \exists C$  if and only if there exists a valuation  $\sigma$  such that  $\mathcal{MSET} \models \sigma(C)$ .

Assume that  $\mathcal{M} \models \sigma(C)$ . From Lemmas A.1 and A.3, we have that  $\mathcal{M} \models \exists C$  for all models  $\mathcal{M}$  of  $MSet$ . This implies that  $MSet \models \exists C$ .  $\square$

## 5.2 Solved Form

We have proved that a constraint is satisfiable in each model if and only if it is satisfiable in the standard one. However, we still have to develop a procedure which tests satisfiability in the standard model. Such a procedure will be based on the notion of solved form.

*Definition 5.5.* A constraint  $C = c_1 \wedge \dots \wedge c_n$  is in *solved form* if for  $i \in \{1, \dots, n\}$ ,  $c_i$  has one of the following forms:

- $X = t$  and  $X$  does not occur neither in  $t$  nor elsewhere in  $C$ ,
- $X \neq t$  and  $X$  does not occur in  $t$ ,
- $t \notin X$  and  $X$  does not occur in  $t$ .

*Remark 5.6.* In the case of multisets (sets)  $t \in X$  is equivalent to  $X = \{[t \mid N]\}$  ( $X = \{t \mid N\}$ , respectively) where  $N$  is a new variable. This allows us to always remove membership constraints. The property does not hold for lists and compact lists. In these cases the solved form must include the further case

- $t \in X$  and  $X$  does not occur in  $t$ .

This inclusion, however, requires the introduction of further semantics conditions in the definition of the solved form for lists and compact lists. As a matter of fact, a constraint such as

$$[[a \mid N]] \in Y \wedge [[a, a \mid N]] \notin Y$$

is unsatisfiable in  $\mathcal{CLIST}$ , since  $[[a \mid N]]$  and  $[[a, a \mid N]]$  are equivalent terms in  $E_{CList}$ . Furthermore, the constraint

$$X \in Y \wedge Y \in X$$

is unsatisfiable in both  $\mathcal{LIST}$  and  $\mathcal{CLIST}$ . Intuitively, the additional conditions that must be tested for the solved form constraint  $C$  in the case of lists and compact lists are (i) membership constraints in  $C$  do not form any cycle; (ii) for each pair of literals of the form  $t \notin X, t' \in X$  in  $C$ ,  $t$  and  $t'$  are not equivalent modulo  $\equiv_E$ , where  $E$  is the equational theory for either lists or compact lists. Both conditions can be automatically tested. In particular, as concerns condition (ii), we know from unification theory (see, e.g., Baader and Nipkow [1998]; Siekmann [1989]) that given an equational theory  $E$ , knowing whether two terms are equivalent modulo  $\equiv_E$  is the same as verifying whether the two terms  $t$  and  $t'$  are  $E$ -unifiable with empty mgu ( $\varepsilon$ ). Thus, test (ii) is connected with the availability of a unification algorithm for the theory  $E$ . In Dovier et al. [1998] it is proved that all four equational theories we are dealing with are finitary (i.e., they admit a finite set of mgu's that covers all possible unifiers) and, moreover, the unification algorithms for the four theories are presented. This gives us a decision procedure for the test. A more precise characterization of the additional conditions for lists and compact lists can be found in Dovier et al. [2003].

We prove that solved form constraints are satisfiable in the corresponding standard models. We prove the property for *MSet*-constraints.

**THEOREM 5.7 (SATISFIABILITY OF SOLVED FORM).** *Given a MSet-constraint  $C$  in solved form, it holds that  $\mathcal{MSET} \models \exists C$ .*

**PROOF.** We split  $C$  into the three parts:  $C^=$ ,  $C^\neq$ , and  $C^\neq$ , containing  $=$ ,  $\neq$ , and  $\neq$  literals, respectively. We use the two auxiliary functions *rank* and *find*. The *rank* of a well-founded multiset is basically the maximum nesting of braces needed to write it. Precisely:

$$\text{rank}(s) = \begin{cases} 0, & \text{if } s \text{ is not of the form } \{[u \mid v]\}, \\ \max\{1 + \text{rank}(u), \text{rank}(v)\}, & \text{if } s \text{ is } \{[u \mid v]\}. \end{cases}$$

*find*( $X, t$ ) is a function that produces for each pair ( $X, t$ ) a set of integer numbers indicating the “depth” of the occurrences of the variable  $X$  in  $t$ . It can be defined as

$$\text{find}(X, t) = \begin{cases} \emptyset & \text{if } t \text{ is a constant term,} \\ \{0\} & \text{if } t \text{ is a variable } X, \\ \{1 + n : n \in \text{find}(X, y)\} & \text{if } t \text{ is } \{[y \mid f(t_1, \dots, t_m)]\}, \\ & f \text{ is not } \{[\cdot \mid \cdot]\}; \\ \{1 + n : n \in \text{find}(X, t_1) \cup \dots \cup \text{find}(X, t_m)\} & \text{if } t \text{ is } f(t_1, \dots, t_m), \\ & f \text{ is not } \{[\cdot \mid \cdot]\}; \\ \{1 + n : n \in \text{find}(X, y)\} \cup \text{find}(X, s) & \text{if } t \text{ is } \{[y \mid s]\}, s \neq \text{nil}. \end{cases}$$

We build a successful valuation  $\gamma$  of  $C$ , in various steps; since the valuation is on a domain whose elements are terms, valuations are substitutions.

$C^=$  is of the form  $X_1 = t_1 \wedge \dots \wedge X_m = t_m$ . We define the substitution:  $\theta_1 = [X_1/t_1, \dots, X_m/t_m]$ .

$C^\neq$  is of the form  $Z_1 \neq s_1 \wedge \dots \wedge Z_o \neq s_o$  ( $Z_i$  does not occur in  $s_i$ ), and  $C^\neq$  is of the form  $r_1 \notin Y_1 \wedge \dots \wedge r_n \notin Y_n$  ( $Y_i$  does not occur in  $r_i$ ). Let  $W_1, \dots, W_h$  be the variables in  $C$  different from the variables  $\bar{X}, \bar{Y}, \bar{Z}$  and let  $\theta_2 = [W_1/\text{nil}, \dots, W_h/\text{nil}]$ .

Let  $\bar{s} = 1 + \max\{\text{rank}(t) : t \notin \bar{X} \text{ occurs in } \theta_2(C) \text{ or } \bar{X} \neq t \text{ occurs in } \theta_2(C)\}$ .

Let  $R_1, \dots, R_j$  be the all the variables occurring in  $\theta_2(C^\neq \wedge C^\neq)$  (actually, all the variables  $\bar{Y}$  and  $\bar{Z}$ ). Let  $n_1, \dots, n_j$  be auxiliary variables ranging over  $\mathbb{N}$ . We build a system  $S$  of linear disequations over the integers in the following way:

- (1)  $S = \{n_i > \bar{s} : \forall i \in \{1, \dots, j\}\} \cup \{n_{i_1} \neq n_{i_2} : \forall i_1, i_2 \in \{1, \dots, j\}, i_1 \neq i_2\}$ .
- (2) For each literal  $R_i \neq s$  in  $\theta_2(C^\neq)$  and for all  $k$  in  $\{1, \dots, j\}, i \neq k$

$$S = S \cup \{n_i \neq n_k + c : \forall c \in \text{find}(R_k, s)\}.$$

- (3) For each literal  $r \notin R_i$  in  $\theta_2(C^\neq)$  and for all  $k$  in  $\{1, \dots, j\}, i \neq k$

$$S = S \cup \{n_i \neq n_k + c + 1 : \forall c \in \text{find}(R_k, r)\}.$$

We say that a linear disequality  $a \neq b$  over the integers is *safe* if, after expressions evaluation, it is not of the form  $u \neq u$ . We say that a system  $A$  of linear disequations over the integers with variables  $x_1, \dots, x_h$  is *safe* if each disequation in  $A$  is either a safe disequality or it is of the form  $x_i > m$ , where  $m$  is an integer number. A finite set of safe linear disequalities has always an

infinite number of solutions (see Lemma A.4 in the Appendix). We show that all disequalities of  $S$  are safe. The disequalities generated at point (1) are safe by definition; those introduced in points (2) and (3) are safe since  $c$  is always a positive number. Thus, it is possible to find an integer solution for the system  $S$ . Let  $\eta = \{n_1 = \bar{n}_1, \dots, n_j = \bar{n}_j\}$  be a solution and define

$$\theta_3 = [R_i / \{[\text{nil}]^{\bar{n}_i} : \forall i \in \{1, \dots, j\}\},$$

where  $\{[\text{nil}]^{\bar{n}}\}$  denotes the term  $\underbrace{\{[\dots\{[\text{nil}]\}\dots]\}}_{\bar{n}}$ .

It is immediately easy to see that in  $KWE_k^m F_1^m F_2^3 F_3^m$  it holds that  $\{[\text{nil}]\}^x = \{[\text{nil}]\}^y$  if and only if  $x = y$  and  $\{[\text{nil}]\}^x \in \{[\text{nil}]\}^y$  if and only if  $x = y - 1$  (see Lemma A.5 in the Appendix).

Let  $\gamma = \theta_1\theta_2\theta_3$  (where  $s\theta_1\theta_2\theta_3$  stands for  $\theta_3(\theta_2(\theta_1(s)))$ ) and observe that  $C\gamma$  is a conjunction of ground literals. We show that  $KWE_k^m F_1^m F_2^3 F_3^m \models C\gamma$ . We analyze each literal of  $C$ .

- $X = t$ .  $\theta_1(X)$  syntactically coincides with  $\theta_1(t) = t$ . The substitution  $\theta_2$  makes the two identical terms ground. A literal of this form is true in any model of equality.
- $Z \neq s$ . the following cases are possible:
  - $s$  is of the form  $\{[\text{nil}]\}^p$  for some  $p < \bar{s}$ .  $Z\gamma$  is of form  $\{[\text{nil}]\}^n$  for some  $n > \bar{s}$ . Since  $n > p$  the result follows.
  - $s$  is of the form  $\{[W_i]\}^p$  for some variable  $W_i$  and some  $p < \bar{s}$ .  $Z\gamma$  is of form  $\{[\text{nil}]\}^n$  for some  $n > \bar{s}$ . Since  $W_i\gamma = \text{nil}$ , the situation is identical to the previous case.
  - $s$  is of the form  $\{[A]\}^p$  for some variable  $A$  among the  $\bar{Y}, \bar{Z}$ , and some  $p < \bar{s}$ . Then  $\text{find}(A, t) = \{p\}$ . This means that the constraint  $n_Z \neq n_A + p$  is introduced in  $S$  and satisfied by the assignment  $\eta$ . Thus  $Z\theta = \{[\text{nil}]\}^{n_Z}$  and  $t\theta = \{[\text{nil}]\}^{n_A + p}$ . Since  $n_Z \neq n_A + p$  the result follows as in the previous cases.
  - If  $s$  is not in any of the previous forms, then  $s\gamma$  can be proved different from  $Z\gamma$  using a sequence of applications of  $E_k^m$  and  $F_2$ .
- $r \notin Y$ . four cases are possible:
  - $r$  is of the form  $\{[\text{nil}]\}^p$  for some  $p < \bar{s}$ .  $Y\gamma$  is of form  $\{[\text{nil}]\}^n$  for some  $n > \bar{s}$ . Since  $n \neq p + 1$  the result follows.
  - $r$  is of the form  $\{[W_i]\}^p$  for some variable  $W_i$  and some  $p < \bar{s}$ .  $Z\gamma$  is of form  $\{[\text{nil}]\}^n$  for some  $n > \bar{s}$ . Since  $W_i\gamma = \text{nil}$ , the situation is identical to the previous case.
  - $r$  is of the form  $\{[X]\}^p$  for some variable  $X$  among the  $\bar{Y}, \bar{Z}$ , and some  $p \leq \bar{s}$ . Then  $\text{find}(X, t) = \{p\}$ . This means that the constraint  $n_Y \neq n_X + p + 1$  is introduced in  $S$  and satisfied by the assignment  $\eta$ . Thus  $Y\theta = \{[\text{nil}]\}^{n_Y}$  and  $r\theta = \{[\text{nil}]\}^{n_X + p}$ . Since  $n_Y \neq n_X + p + 1$  the result follows as in the previous cases.
  - If  $r$  is not in any of the previous forms, then  $r\gamma$  can be proved different from  $Y\gamma$  using axiom  $W$  and a sequence of applications of  $E_k^m$  and  $F_2$ .  $\square$

Let  $\mathbb{T}$  be one of the theories *List*, *MSet*, *CList*, *Set*,  $\pi$  be a symbol in  $\{=, \neq, \in, \notin\}$ , and  $C$  be a  $\mathbb{T}$ -constraint

```

while  $C$  contains an atomic constraint  $c$  not in solved form and  $C \neq \text{false}$  do
  select  $c$ ;
  if  $c = \text{false}$  then return false
  else if  $c = \text{true}$  then erase  $c$ 
  else if  $c = \ell \pi r$  apply any rewriting rule for  $\mathbb{T}$ -constraints of the form  $\ell \pi r$ ;
return  $C$ 

```

Fig. 1. Main loop of constraint rewriting procedures.

Hence a solved-form constraint can be seen as a symbolic representation for a nonempty and possibly infinite set of valuations, that is, the valuations satisfying it.

## 6. CONSTRAINT REWRITING PROCEDURES

In this section we describe the procedures that allow us to obtain solved-form constraints from any given constraint  $C$ . Precisely, these procedures rewrite the constraint  $C$  either into an equisatisfiable disjunction of constraints in solved form or false. The constraint is rewritten to false if and only if it is not satisfiable in the standard model. As a consequence of the results of the previous sections these procedures decide the satisfiability of a constraint in each model of the theory. Moreover, the disjunction of constraints in solved form given as output is a finite representation for the valuations satisfying the input constraint.

All procedures have the same overall structure shown in Figure 1: they take a constraint  $C$  as input and repeatedly select a conjunct  $c$  in  $C$  not in solved form (if any) and apply one of the rewriting rules to it. The procedure stops when the constraint  $C$  is in solved form or it contains false as one of its conjuncts.

The procedure is nondeterministic. Some rewriting rules have two or more possible nondeterministic choices. Each nondeterministic computation returns a constraint in solved form or false. Globally, the procedure returns a finite collection  $C_1, \dots, C_k$  of constraints. The input constraint  $C$  and the disjunction  $C_1 \vee \dots \vee C_k$  are equisatisfiable in the standard models. We show the details for the multiset case only. Details for the other procedures can be found in Dovier et al. [2003].

### 6.1 Equality Constraints

Unification algorithms for verifying the satisfiability and producing the solutions of equality constraints in the four aggregate theories have been proposed in Dovier et al. [1998]. These algorithms fall in the general schema of Figure 1. Some determinism in the statement select  $c$  is added to ensure termination. They are called

- unify-*List* for lists,
- unify-*MSet* (called unify-bags in Dovier et al. [1998]) for multisets,
- unify-*CList* for compact lists,
- unify-*Set* for sets,



Rules for unify- <i>MSet</i>	
(1)	$X = X \mapsto \mathbf{true}$
(2)	$\left. \begin{array}{l} t = X \\ t \text{ is not a variable} \end{array} \right\} \mapsto X = t$
(3)	$\left. \begin{array}{l} X = t \\ X \notin \text{FV}(t), X \text{ occurs elsewhere in } C \\ X = t \text{ and apply the substitution } [X/t] \text{ to } C \end{array} \right\} \mapsto$
(4)	$\left. \begin{array}{l} X = t \\ t \text{ is not } X, X \in \text{FV}(t) \end{array} \right\} \mapsto \mathbf{false}$
(5)	$\left. \begin{array}{l} f(s_1, \dots, s_m) = g(t_1, \dots, t_n) \\ f \text{ is not } g \end{array} \right\} \mapsto \mathbf{false}$
(6)	$\left. \begin{array}{l} f(s_1, \dots, s_m) = f(t_1, \dots, t_m) \\ m \geq 0, f \text{ is not } \{\cdot   \cdot\} \end{array} \right\} \mapsto$ $s_1 = t_1 \wedge \dots \wedge s_m = t_m$
(7)	$\left. \begin{array}{l} \{\{t   s\} = \{\{t'   s'\}\} \\ \text{tail}(s) \text{ and tail}(s') \text{ are not the same variable} \end{array} \right\} \mapsto$ $(i) (t = t' \wedge s = s') \vee$ $(ii) (s = \{\{t'   N\} \wedge \{\{t   N\} = s'\})$
(8)	$\left. \begin{array}{l} \{\{t   s\} = \{\{t'   s'\}\} \\ \text{tail}(s) \text{ and tail}(s') \text{ are the same variable} \end{array} \right\} \mapsto$ $\text{untail}(\{\{t   s\}\}) = \text{untail}(\{\{t'   s'\}\})$

Fig. 2. Rewriting rules of the unification algorithm for multisets.

and they are used unaltered in the four global constraint solvers that we propose in this article.

The output of the algorithms is either false, when the input constraint is unsatisfiable, or a collection of solved form constraints composed only by equality atoms. In Figure 2 we show the rewriting rules for multiset unification.

The algorithm uses the auxiliary functions *tail* and *untail* defined as follows:

$$\begin{aligned}
 \text{tail}(f(t_1, \dots, t_n)) &= f(t_1, \dots, t_n) & f \text{ is not } \{\cdot | \cdot\}, n \geq 0, \\
 \text{tail}(X) &= X & X \text{ is a variable,} \\
 \text{tail}(\{\{t | s\}\}) &= \text{tail}(s), \\
 \text{untail}(X) &= \text{nil}, & X \text{ is a variable,} \\
 \text{untail}(\{\{t | s\}\}) &= \{\{t | \text{untail}(s)\}\}.
 \end{aligned}$$

The following lemma, which states the soundness and completeness of the unification rules, has been proved in Dövier et al. [1998]. We report here the proof for the sake of completeness.

**LEMMA 6.1.** [Dövier et al. 1998] *Let  $\mathbb{T}$  be one of the theories *List*, *MSet*, *CList*, *Set*, and  $\mathcal{A}_{\mathbb{T}}$  be the standard model for  $\mathbb{T}$ . Let  $C$  be a  $\mathbb{T}$ -constraint,  $C_1, \dots, C_k$  be the constraints nondeterministically returned by  $\text{unify-}\mathbb{T}(C)$ , and  $\tilde{N}_i = \text{FV}(C_i) \setminus \text{FV}(C)$ . Then  $\mathcal{A}_{\mathbb{T}} \models \forall (C \leftrightarrow \bigvee_{i=1}^k \exists \tilde{N}_i C_i)$ .*

**PROOF.** Let us prove the property for each rule separately.

—unify-*MSet*(1), (2), (3).] They immediately follow from equality axioms.

—unify-*MSet*(4).] It is justified by axiom ( $F_3^m$ ).

- unify-*MSet*(5).] It is immediately justified by axiom schema ( $F^2$ ).
- unify-*MSet*(6).] One direction follows from the equality axioms, the other one from axiom ( $F_1$ )
- unify-*MSet*(7).] It is immediately justified by axiom ( $E_k^m$ ).
- unify-*MSet*(8).] It is immediately justified by axiom ( $F_3^m$ ) (the auxiliary function untail replace the variable that occurs as tail of the two multisets with nil).  $\square$

*Remark 6.2.* Consider the constraint

$$\{[a \mid X]\} = \{[b \mid Y]\} \wedge \{[d \mid X]\} = \{[e \mid Y]\}.$$

If we apply rule (7ii) to the first equation and then to the second equation, we obtain

$$X = \{[b \mid N_1]\} \wedge \{[a \mid N_1]\} = Y \wedge X = \{[e \mid N_2]\} \wedge \{[d \mid N_2]\} = Y.$$

Then, applying rule (2) to the second equation and substitution (rule (4)) to the first and second equation, we get

$$X = \{[b \mid N_1]\} \wedge Y = \{[a \mid N_1]\} \wedge \{[b \mid N_1]\} = \{[e \mid N_2]\} \wedge \{[d \mid N_2]\} = \{[a \mid N_1]\}.$$

The first two equations are in solved form. The third and fourth equations constitute a constraint absolutely equivalent to the starting one. This is a possible source of nontermination. However, a simple selection strategy is sufficient to avoid this problem. From the initial system, apply rule (7ii) to the first equation and then the two substitutions induced:

$$X = \{[b \mid N_1]\} \wedge Y = \{[a \mid N_1]\} \wedge \{[d, b \mid N_1]\} = \{[e, a \mid N_1]\}.$$

Then rule (8) can be used removing “tail” variables:

$$X = \{[b \mid N_1]\} \wedge Y = \{[a \mid N_1]\} \wedge \{[d, b]\} = \{[e, a]\}.$$

And in few steps termination (with failure) occurs. Basically the rule is “when a multiset-multiset equation is selected, recursively processing first all the equations introduced by it.” This rule is easily implemented using a stack. For more details, see Dovier et al. [1998].

## 6.2 Membership and Not-Membership Constraints

The rewriting procedures for membership and not-membership constraints on a specific aggregate are obtained from the general schema of Figure 1, using the rewriting rules for membership and not-membership constraints suitably instantiated with the corresponding theory. These rules are justified by axioms ( $K$ ) and ( $W$ ) that hold in all the four theories. In Figure 3 we show the rules in the case of multisets. Note that the rewriting rule (4) for *in-MSet* can be used for sets and multisets, but not for the other theories (see also Remark 5.6). Thus, the rules for membership constraints in the case of lists and compact lists only deal with cases (1)–(3), while constraints of the form  $r \in X$  remain unchanged in the solved form.

Rules for in- $MSet$		
(1)	$\left. \begin{array}{l} r \in f(t_1, \dots, t_n) \\ f \text{ is not } \llbracket \cdot \mid \cdot \rrbracket \end{array} \right\} \mapsto \text{false}$	
(2)	$r \in \llbracket t \mid s \rrbracket \mapsto r = t \vee r \in s$	(a) (b)
(3)	$\left. \begin{array}{l} r \in X \\ X \in \text{FV}(r) \end{array} \right\} \mapsto \text{false}$	
(4)	$\left. \begin{array}{l} r \in X \\ X \notin \text{FV}(r) \end{array} \right\} \mapsto X = \llbracket r \mid N \rrbracket$	

  

Rules for nin- $MSet$		
(1)	$\left. \begin{array}{l} r \notin f(t_1, \dots, t_n) \\ f \text{ is not } \llbracket \cdot \mid \cdot \rrbracket \end{array} \right\} \mapsto \text{true}$	
(2)	$r \notin \llbracket t \mid s \rrbracket \mapsto r \neq t \wedge r \notin s$	
(3)	$\left. \begin{array}{l} r \notin X \\ X \in \text{FV}(r) \end{array} \right\} \mapsto \text{true}$	

Fig. 3. Rewriting rules for membership and not-membership constraints.

LEMMA 6.3. Let  $\mathbb{T}$  be one of the theories *List*, *MSet*, *CList*, *Set*, and  $\mathcal{A}_{\mathbb{T}}$  be the standard model for the theory  $\mathbb{T}$ . Let  $C$  be a  $\mathbb{T}$ -constraint,  $C_1, \dots, C_k$  be the constraints nondeterministically returned by  $\text{nin-}\mathbb{T}(\text{in-}\mathbb{T}(C))$ , and  $\tilde{N}_i = \text{FV}(C_i) \setminus \text{FV}(C)$ . Then  $\mathcal{A}_{\mathbb{T}} \models \forall (C \leftrightarrow \bigvee_{i=1}^k \exists \tilde{N}_i C_i)$ .

PROOF. We prove soundness and completeness for multisets, thus with respect to the model  $\mathcal{MSET}$ . Soundness and completeness for the other aggregates can be proved in the very same way (with the exception of rule (4)). Soundness and completeness is proved for each rewriting rule separately since the rules are mutually exclusive.

- in- $MSet$ (1).]  $r \in f(t_1, \dots, t_n)$ , with  $f$  different from  $\llbracket \cdot \mid \cdot \rrbracket$  is equivalent to **false** by axiom ( $K$ ).
- in- $MSet$ (2).] This is exactly axiom ( $W$ ).
- in- $MSet$ (3).] Assume that there is a valuation  $\sigma$  such that  $\mathcal{MSET} \models \sigma(r \in X)$ . This means that  $\sigma(X)$  is an equivalence class which contains a term of the form:  $\llbracket s_1, \dots, s_n, r' \mid t \rrbracket$  for some terms  $s_1, \dots, s_n, t$  and for some term  $r'$  in the equivalence class  $\sigma(r)$ . Axiom ( $F_3$ ) ensures that  $X$  cannot be a subterm of  $r$ .
- in- $MSet$ (4).] Assume that there is a valuation  $\sigma$  such that  $\mathcal{MSET} \models \sigma(r \in X)$ . This means that  $\sigma(X)$  is an equivalence class which contains a term of the form:  $\llbracket s_1, \dots, s_n, r' \mid t \rrbracket$  for some terms  $s_1, \dots, s_n, t$  and for some term  $r'$  in  $\sigma(r)$ . Since  $\mathcal{MSET}$  is a model of  $(E_k^m)$ , this means that the class  $\sigma(X)$  contains also  $\llbracket r', s_1, \dots, s_n \mid t \rrbracket$  for some terms  $s_1, \dots, s_n, t$ . Thus, it is a model of  $X = \llbracket r \mid N \rrbracket$ . The other direction is similar.
- nin- $MSet$ (1), (2), (3).] Same proofs as for the corresponding in- $MSet$  rules, using the same axioms.  $\square$

Rules for neq- <i>MSet</i>		
(1)	$\left. \begin{array}{l} d \neq d \\ d \text{ is a constant} \end{array} \right\}$	$\mapsto \text{false}$
(2)	$\left. \begin{array}{l} f(s_1, \dots, s_m) \neq g(t_1, \dots, t_n) \\ f \text{ is not } g \end{array} \right\}$	$\mapsto \text{true}$
(3)	$\left. \begin{array}{l} t \neq X \\ t \text{ is not a variable} \end{array} \right\}$	$\mapsto X \neq t$
(4)	$\left. \begin{array}{l} X \neq X \\ X \text{ is a variable} \end{array} \right\}$	$\mapsto \text{false}$
(5)	$\left. \begin{array}{l} f(s_1, \dots, s_n) \neq f(t_1, \dots, t_n) \\ n > 0, f \text{ is not } \text{cons}_{\mathbb{T}}(\cdot, \cdot) \end{array} \right\}$	$\mapsto \begin{array}{l} s_1 \neq t_1 \vee (1) \\ \vdots \\ s_n \neq t_n \quad (n) \end{array}$
(6.1)	$\left. \begin{array}{l} \{t_1 \mid s_1\} \neq \{t_2 \mid s_2\} \\ \text{tail}(s_1) \text{ and } \text{tail}(s_2) \\ \text{are the same variable} \end{array} \right\}$	$\mapsto \text{untail}(\{t_1 \mid s_1\}) \neq \text{untail}(\{t_2 \mid s_2\})$
(6.2)	$\left. \begin{array}{l} \{t_1 \mid s_1\} \neq \{t_2 \mid s_2\} \\ \text{tail}(s_1) \text{ and } \text{tail}(s_2) \\ \text{are not the same variable} \end{array} \right\}$	$\mapsto \begin{array}{l} (t_1 \neq t_2 \wedge t_1 \notin s_2) \vee \quad (a) \\ (\{t_2 \mid s_2\} = \{t_1 \mid N\} \wedge s_1 \neq N) \quad (b) \end{array}$
(7)	$\left. \begin{array}{l} X \neq f(t_1, \dots, t_n) \\ X \in \text{FV}(t_1, \dots, t_n) \end{array} \right\}$	$\mapsto \text{true}$

Fig. 4. Rewriting rules for disequality constraints on multisets.

### 6.3 Disequality Constraints

Rewriting rules for disequality constraints consist of a part common to the four theories (rules (1)–(5)), and a part which is specific to each theory. In Figure 4 we show the rules for the multiset case.

Some words are necessary to explain the rules which manage disequalities between multisets. In particular, if we used directly axiom  $(E_k^m)$  in rule(6.2) of Figure 4, we would have that

$$\{t_1 \mid s_1\} \neq \{t_2 \mid s_2\} \leftrightarrow (t_1 \neq t_2 \vee s_1 \neq s_2) \wedge \forall N (s_2 \neq \{t_2 \mid N\} \vee s_1 \neq \{t_1 \mid N\}).$$

Since an universal quantification is introduced, this is no longer a constraint according to Definition 5.1.

Alternatively, we could use the intuitive notion of multi-membership:  $x \in^i y$  if  $x$  belongs at least  $i$  times to the multiset  $y$ . This way, one can provide an alternative version of equality and disequality between multisets. In particular, we would have that

$$\{t_1 \mid s_1\} \neq \{t_2 \mid s_2\} \leftrightarrow \exists X \exists n (n \in \mathbb{N} \wedge (X \in^n \{t_1 \mid s_1\} \wedge X \notin^n \{t_2 \mid s_2\}) \vee (X \in^n \{t_2 \mid s_2\} \wedge X \notin^n \{t_1 \mid s_1\})).$$

In this case, however, the quantification over natural numbers is outside the language we are studying. Conversely, the rewriting rule (6.2) adopted in Figure 4 avoids these difficulties introducing only one existential quantification ( $\exists N$  in the set of terms  $T(\mathcal{F}_{MSet})$ ).

*Remark 6.4.* Observe that, differently from multisets, the rewriting rule for disequality between compact lists follows immediately from axiom  $(E_k^c)$ . As a matter of fact, this axiom does not introduce any new variable.

As concerns sets, axiom  $(E_k^s)$  introduces an existentially quantified variable, as for multisets. Thus, its direct application for stating disequality would require universally quantified constraints that go outside the language. On the other hand, the rewriting rule (6.2) used for multisets (Figure 4) cannot be used in this context. In fact, the property that  $s_1 \neq N$  implies  $\{[t_1 | s_1]\} \neq \{[t_1 | N]\}$ , holding for finite multisets, does not hold for sets. For instance,  $\{a\} \neq \{a, b\}$  but  $\{b, a\} = \{b, a, b\}$ . Thus, this rewriting rule would be not correct for sets.

A rewriting rule for disequality constraints on sets, however, can be easily obtained by taking the negation of the *standard extensionality axiom* for sets

$$(E_k) \ x = y \leftrightarrow \forall z (z \in x \leftrightarrow z \in y)$$

This leads to the following rewriting rule that replaces rules (6.1) and (6.1) of Figure 4 in the case of disequality constraints on sets:

$$(6) \ \{t_1 | s_1\} \neq \{t_2 | s_2\} \} \mapsto \begin{array}{l} Z \in \{t_1 | s_1\} \wedge Z \notin \{t_2 | s_2\} \vee (a) \\ Z \in \{t_2 | s_2\} \wedge Z \notin \{t_1 | s_1\} \quad (b). \end{array}$$

Soundness and completeness of *neq-MSet* are proved by the following lemma.

**LEMMA 6.5.** *Let  $C$  be a MSet-constraint,  $C_1, \dots, C_k$  be the constraints nondeterministically returned by  $\text{neq-MSet}(C)$ , and  $\bar{N}_i = \text{FV}(C_i) \setminus \text{FV}(C)$ . Then  $\mathcal{MSET} \models \forall (C \leftrightarrow \bigvee_{i=1}^k \exists \bar{N}_i C_i)$ .*

**PROOF.** Let us prove the property for each rule separately.

- neq-MSet*(1), (3), (4).] They immediately follow from equality axioms.
- neq-MSet*(2).] It is justified by axiom  $(F_2)$ .
- neq-MSet*(5).] One direction follows from the equality axioms, the other one from axiom  $(F_1^m)$ .
- neq-MSet*(6.1).] It is immediately justified by axiom schema  $(F_3^m)$ .
- neq-MSet*(6.2).] The constraint  $\{[t_1 | s_1]\} \neq \{[t_2 | s_2]\}$  is equivalent to

$$t_1 \notin \{[t_2 | s_2]\} \wedge \{[t_1 | s_1]\} \neq \{[t_2 | s_2]\} \vee \quad (1)$$

$$t_1 \in \{[t_2 | s_2]\} \wedge \{[t_1 | s_1]\} \neq \{[t_2 | s_2]\}. \quad (2)$$

Since we are looking for successful valuations over  $\mathcal{MSET}$  that deal with multisets of finite elements, axiom  $(E_k^m)$  ensures that  $t_1 \notin \{[t_2 | s_2]\}$  implies  $\{[t_1 | s_1]\} \neq \{[t_2 | s_2]\}$ . Thus, formula (1) is equivalent to  $t_1 \in \{[t_2 | s_2]\}$  which, in turn, is equivalent by (W) to the disjunct (a) generated by the rewriting rule.

Consider now formula (2). It is easy to see that

$$\mathcal{MSET} \models \forall (t_1 \in \{[t_2 | s_2]\} \leftrightarrow \exists M (\{[t_1 | M]\} = \{[t_2 | s_2]\})). \quad (3)$$

Thus, (2) is equivalent to

$$\exists M (\{[t_1 | M]\} = \{[t_2 | s_2]\} \wedge \{[t_1 | s_1]\} \neq \{[t_2 | s_2]\}). \quad (4)$$

It remains to prove that (4) is equivalent to the disjunct (b), namely:

$$\exists N (s_1 \neq N \wedge \{[t_2 | s_2]\} = \{[t_1 | N]\}). \quad (5)$$

- (4)  $\rightarrow$  (5).] Assume that there exists  $M$  which satisfies (4).  $M = s_1$  will immediately lead to a contradiction. Thus, (5) is satisfied by  $N = M$ .
- (5)  $\rightarrow$  (4).] Assume that there exists  $N$  which satisfies (5). It immediately follows from the fact, true for finite multisets, that  $s_1 \neq N$  implies  $\{[t_1 | s_1]\} \neq \{[t_1 | N]\}$ . Thus, choose  $M = N$ .  $\square$

*Remark 6.6.* In our theories an aggregate can be built starting from any ground uninterpreted Herbrand term, called the *kernel*, and by adding to it the elements that compose the aggregate. Thus, two aggregates can contain the same elements but nevertheless they can be different because they have different kernels. For instance, the two terms  $\{a | b\}$  and  $\{a | c\}$  denote two different sets containing the same elements (i.e., only  $a$ ) but based on different kernels (i.e.,  $b$  and  $c$ , respectively).

Rewriting rules for disequality constraints on aggregates other than sets are formulated in such a way to take care of the possibly different kernels without having to explicitly resort to kernels. Conversely, the rewriting rule for disequality constraints on sets (similar to rule (6) of *neq\_MSet* and its subrules) is not able to “force” disequality between two sets when they have the same elements but different kernels. A possible completion of the above procedures to take care of this case is presented in Dovier and Rossi [1993]. Basically, a new constraint (*ker*) is introduced and the rewriting rule (6) is endowed with a third nondeterministic case:  $\text{ker}(s_1) \neq \text{ker}(s_2)$ . For further details, see Dovier et al. [2003].

## 7. CONSTRAINT SOLVING

Now we have all ingredients to address the problem of establishing whether a constraint  $C$  is satisfiable in the corresponding standard model. Theorem 5.4 ensures that the property is inherited by any model.

Constraint satisfiability for a theory  $\mathbb{T}$  is checked by the nondeterministic rewriting procedure  $\text{SAT}_{\mathbb{T}}$  shown in Figure 5.  $\text{SAT}_{\mathbb{T}}$  is completely parametric with respect to the theory involved and it iteratively uses the rewriting procedures presented in the previous sections, until a fixed-point is reached, that is, until any new rewritings do not further simplify the constraint. This happens when the constraint is either in solved form or it is false.

By Theorem 5.7 a constraint in solved form is guaranteed to be satisfiable in the corresponding model. Moreover, it will be proved in Theorem 7.2 that the disjunction of solved form constraints returned by  $\text{SAT}_{\mathbb{T}}$  is equisatisfiable in the standard model with the original constraint  $C$ . Therefore,  $\text{SAT}_{\mathbb{T}}$  can be used as a test procedure to check satisfiability of  $C$ : if it is able to reduce  $C$  to at least one solved-form constraint  $C'$ , then  $C$  is satisfiable; otherwise,  $C$  is unsatisfiable.



```

function SATT(C)
  repeat
    C' := C;
    C := unify-T(neq-T(nin-T(in-T(C))))
  until C = C';
  return(C)
    
```

Fig. 5. The satisfiability procedure, parametric with respect to  $T$ .

The generated constraint in solved form can be exploited to compute all possible successful valuations for  $C$ .

The rest of this section is devoted to prove the crucial result of termination of the procedure  $SAT_T(C)$ , to prove its soundness and completeness, and, finally, to give some hints on its complexity.

**THEOREM 7.1 (TERMINATION).** *Let  $T$  be one of the theories *List*, *MSet*, *CList*, *Set*, and  $C$  be a  $T$ -constraint. Each nondeterministic execution of  $SAT_T(C)$  terminates in a finite number of steps. Moreover, the constraint returned is either false or a solved-form constraint.*

**PROOF.** We give the proof for the case of *MSet*. The other proofs can be found in Dovier et al. [2003].

It is immediate to see, by the definitions of the procedures, that if  $C$  is different from false and not in solved form, then some rewriting rule can be applied. If we apply a rewriting rule that leads to false, then the process terminates. Thus, we do not analyze such rules in the rest of this proof.

We prove that the repeat cycle cannot loop forever. For doing that, we define a complexity measure for constraints. Let us assume that constraints of the form  $X = t$ , with  $X$  neither in  $t$  nor elsewhere in  $C$ , are removed from  $C$ . Similarly, we assume that true constraints are not counted in the complexity measure. These two assumptions are safe since those constraints do not fire any new rule application. The complexity measure that we associate with a constraint is the following triple:

$$\begin{aligned}
 compl(C) = \langle & \alpha(C) = \# \text{ vars in } C, \\
 & \beta(C) = \{[size(s) + size(t) : s \text{ op } t \in C]\}, \\
 & \gamma(C) = \sum_{s \text{ op } t \in C} size(s) \rangle.
 \end{aligned}$$

The first and third element of the triple are nonnegative integers. The second is a multiset of nonnegative integers. Multisets of nonnegative integers are well ordered [Dershowitz and Manna 1979] by the ordering obtained as the transitive closure of the rule:

$$\{[s_1, \dots, s_{i-1}, t_1, \dots, t_n, s_{i+1}, \dots, s_m]\} < \{[s_1, \dots, s_m]\},$$

for  $i \in \{1, \dots, m\}$ ,  $n \geq 0$ ,  $t_1 < s_i, \dots, t_n < s_i$ . The ordering on triples is the (well-founded) lexicographical ordering.

We will prove that given a constraint  $C$  a constraint  $C'$  with lower complexity is reached in a finite number of nonfailing successive rule applications. We show

this property by case analysis. Most rule applications decrease the complexity in one step. When this does not happen, we enter in more detail.

- unify-*MSet*(1).]  $\alpha$  does not increase,  $\beta$  decreases.
- unify-*MSet*(2).]  $\alpha$  and  $\beta$  do not increase.  $\gamma$  decreases, since  $\text{size}(X) = 0$  and  $\text{size}(t) > 0$ .
- unify-*MSet*(3).]  $\alpha$  decreases by 1.
- unify-*MSet*(6).]  $\alpha$  does not increase.  $\beta$  decreases, since an equation of size  $1 + \sum_{i=1}^m \text{size}(s_i) + \text{size}(t_i)$  is replaced by  $m$  smaller equations of size  $\text{size}(s_i) + \text{size}(t_i)$ .
- unify-*MSet*(7).] In this case the complexity may remain unchanged at the first step. However, the unification algorithm adopts a selection strategy that ensures that, after a finite number of steps, either  $\alpha$  decreases or  $\alpha$  does not change and  $\beta$  decreases (see Remark 6.2).
- unify-*MSet*(8).] After one rule application, we are in case (7) with both the tails of the multisets nonvariables. After a finite number of steps, we enter the situation where  $\alpha$  is unchanged and  $\beta$  decreases.
- in-*MSet*(2).]  $\alpha$  does not increase.  $\beta$  decreases, since a constraint of size  $1 + \text{size}(r) + \text{size}(s) + \text{size}(t)$  is nondeterministically replaced by one of smaller size, that is, either  $\text{size}(r) + \text{size}(s)$  or  $\text{size}(r) + \text{size}(t)$ .
- nin-*MSet*(1), (3).]  $\alpha$  does not increase and  $\beta$  decreases.
- nin-*MSet*(2).]  $\alpha$  does not increase.  $\beta$  decreases, since a constraint of size  $1 + \text{size}(r) + \text{size}(s) + \text{size}(t)$  is non-deterministically replaced by two constraints of smaller size  $\text{size}(r) + \text{size}(s)$  and  $\text{size}(r) + \text{size}(t)$ .
- neq-*MSet*(2), (7).] Trivially,  $\alpha$  does not increase and  $\beta$  decreases.
- neq-*MSet*(3).]  $\alpha$  and  $\beta$  do not increase.  $\gamma$  decreases, since  $\text{size}(X) = 0$  and  $\text{size}(t) > 0$ .
- neq-*MSet*(5).]  $\alpha$  does not increase.  $\beta$  decreases, since a constraint of size  $1 + \sum_{i=1}^m \text{size}(s_i) + \text{size}(t_i)$  is nondeterministically replaced by one of size  $\text{size}(s_i) + \text{size}(t_i)$ .
- neq-*MSet*(6.2).] A unique application of this rule may not decrease the constraint complexity. However, the rule removes  $\{[t_1 | s_1]\} \neq \{[t_2 | s_2]\}$  and introduces

$$\{[t_2 | s_2]\} = \{[t_1 | N]\} \wedge, \quad (6)$$

$$s_1 \neq N. \quad (7)$$

Consider now the following two cases:

- (a)  $\{[t_2 | s_2]\}$  is  $\{[r_1, \dots, r_n]\}$ ;
- (b)  $\{[t_2 | s_2]\}$  is  $\{[r_1, \dots, r_n | A]\}$ , for some variable  $A$  distinct from  $N$  that has just been introduced.

In the first case the successive execution of unify-*MSet* replaces equation (6) by

$$t_1 = r_i \wedge N = \{[r_1, \dots, r_{i-1}, r_{i+1}, \dots, r_n]\}$$

for some  $i = 1, \dots, n$ . We have that

$$\text{size}(t_1) + \text{size}(r_i) < \text{size}([t_1 \mid s_1]) + \text{size}([t_2 \mid s_2]).$$

The equation  $N = \{[r_1, \dots, r_{i-1}, r_{i+1}, \dots, r_n]\}$  is eliminated by applying the substitution for  $N$ .  $N$  occurs only in the constraint  $s_1 \neq N$ , which becomes  $s_1 \neq \{[r_1, \dots, r_{i-1}, r_{i+1}, \dots, r_n]\}$ . Again, its *size* is strictly smaller than that of the original disequality constraint. Thus, after some further steps,  $\alpha$  remains unchanged while  $\beta$  decreases. Strictly speaking, some other actions may occur during that sequence of actions. However, if no other rule (6.2) is executed, then all rules decrease the complexity tuples. Conversely, if other rules of this form are executed, then we need to wait for all the substitutions of this form to be applied. But they are all independent processes.

The second case is similar, but in such a case a substitution also for  $A$  is computed, ensuring that  $\alpha$  decreases.

—*neq-MSet*(6.1).] After one step, we are in the above situation (6.2).  $\square$

The soundness and completeness result of the global constraint solving procedure for *List*, *MSet*, *CList*, and *Set* follows from the lemmas in the previous sections. As observed in Remark 6.6, the completeness of the *Set* case needs some care to deal with kernels.

**THEOREM 7.2 (SOUNDNESS—COMPLETENESS).** *Let  $\mathbb{T}$  be one of the theories *List*, *MSet*, *CList*, and *Set*,  $C$  be a  $\mathbb{T}$ -constraint, and  $C_1, \dots, C_k$  be the solved-form constraints nondeterministically returned by  $\text{SAT}_{\mathbb{T}}(C)$ , and  $\tilde{N}_i$  be  $\text{FV}(C_i) \setminus \text{FV}(C)$ . Then  $\mathcal{A}_{\mathbb{T}} \models \forall (C \leftrightarrow \bigvee_{i=1}^k \exists \tilde{N}_i C_i)$ , where  $\mathcal{A}_{\mathbb{T}}$  is the model which corresponds with  $\mathbb{T}$ .*

**PROOF.** We specialize the proof for the multiset case. Theorem 7.1 ensures the termination of each nondeterministic branch. At each branch point, the number of nondeterministic choices is finite. Thus,  $C_1, \dots, C_k$  can be effectively computed. Both soundness and completeness results about the global constraint solving procedure follow from the results proved individually for the procedures involved: Lemma 6.1 for unification, Lemma 6.3 for *in-MSet* and *nin-MSet*, and Lemma 6.5 for *neq-MSet*.  $\square$

**COROLLARY 7.3 (DECIDABILITY).** *Given a  $\mathbb{T}$ -constraint  $C$ , it is decidable whether  $\mathcal{A}_{\mathbb{T}} \models \exists C$ , where  $\mathcal{A}_{\mathbb{T}}$  is one of the standard models *LIST*, *MSET*, *CLIST*, *SET*.*

**PROOF.** From Theorem 7.2 we know that  $C$  is equisatisfiable with  $C_1 \vee \dots \vee C_k$ . If all the  $C_i$  are false, then  $C$  is unsatisfiable in *LIST* (*MSET*, *CLIST*, *SET*). Otherwise, it is satisfiable, since solved-form constraints are satisfiable (Theorem 5.7).  $\square$

As far as complexity is concerned, we first need to distinguish between the complexity of the constraint satisfiability problem and the complexity of the satisfiability procedure we present.

Complexities of the four unification problems were studied in Dovier et al. [1998]: the decision problem for unification is proved to be solvable in linear time for lists, while it is NP-complete for the other cases. In the case of lists, not only the unification problem is polynomial, but also the problem involving equalities and disequalities. In particular, if a constraint on lists is a conjunction of equalities and disequalities, then its satisfiability is solvable in deterministic quadratic time [Baader and Nipkow 1998; Corbin and Bidoit 1983]. On the other hand, the satisfiability problem for conjunctions of membership and disequality constraints on lists is NP-hard. A reduction from 3-SAT is briefly discussed in Dovier et al. [2003]. The same reduction can be applied to the other aggregates. Since  $X \neq Y$  is equivalent to  $X \notin \{[Y]\}$ , the above-mentioned reduction can be adapted to prove the NP-hardness of the satisfiability problem for constraints involving only membership and not-membership. In the four aggregate theories, the satisfiability of a conjunction of disequalities and not-membership can be tested in polynomial time by simply applying some reorderings on the terms and syntactic checks. The case of disequalities on sets with a union-based approach has been considered in Dovier et al. [2004].

Let us now comment on the complexity of our constraint rewriting procedures. The unification algorithm presented in Dovier et al. [1998] and briefly recalled here can generate terms which grow exponentially. Consider for instance the constraint

$$X_1 = f(X_2, X_2) \wedge X_2 = f(X_3, X_3) \wedge \dots \wedge X_{n-1} = f(X_n, X_n).$$

It is easy to see that, if we apply all the substitutions, then  $X_1$  will be bound to a term whose size is exponential with respect to  $n$ . However, as explained in Aliffi et al. [1999] and Dantsin and Voronkov [1999], it is possible to avoid explicit substitutions, thus obtaining an implementation of the unification algorithm which works in nondeterministic polynomial time. In our context, at the implementation level, terms can be represented by linked structures. Precisely, a term  $f(t_1, \dots, t_n)$  can be represented by a node labeled by  $f$  pointing to the nodes representing  $t_1, \dots, t_n$ . Each occurrence of a variable  $X$  is associated to a unique node. In this way, explicit substitutions are implemented by node collapsing. If we exploit such implementation in our constraint satisfiability procedure  $\text{SAT}_{\mathbb{T}}$ , we only need to perform some further checks at the end of the computation to guarantee the satisfiability of the returned constraint. For instance, if we get a conjunct of the form  $X \neq t$ , we need to check that this is coherent with the equalities, that is, we have to check that the pointers of  $X$  and  $t$  do not syntactically generate the same terms. Hence, since the procedures for membership, not-membership, and disequalities, work in nondeterministic polynomial time, we can obtain a nondeterministic polynomial time implementation for  $\text{SAT}_{\mathbb{T}}$ .

## 8. CONCLUSIONS

We have extended the results of Dovier et al. [1998] studying the constraint solving problem for four different theories, namely, the theories of lists, multisets, compact lists, and sets. The analyzed constraints are conjunctions of

literals based on equality and membership predicate symbols. We have identified the standard models for these theories by showing that they correspond with the theories on the class of considered constraints. We have developed a notion of solved form (proved to be satisfiable) and presented the rewriting algorithms which allow this notion to be used to decide the satisfiability problems for the four aggregates. In particular, we presented a constraint solving technique parametric with respect to these theories and we have pointed out the differences and similarities among the four kinds of aggregates.

An implementation of the results described in this article can be found in the Constraint Logic Programming language `{log}` (<http://prmat.math.unipr.it/~gianfr/setlog.Home.html>). In this language the aggregate theories discussed in this paper (except that for compact lists) are combined all together to provide a general framework where to deal with several of the proposed forms of aggregates simultaneously. As a matter of fact, the choices made in the axiomatic definition of the theories, as well as the parametric definition of the relevant constraint rewriting procedures, make their combination into a single general framework immediately feasible, with only a very limited effort.

As further work it could be interesting to study the properties of the four aggregates in presence of append-like operators (*append* for lists,  $\cup$  for sets,  $\uplus$  for multisets). These operators cannot be defined without using universal quantifiers (or recursion) with the languages analyzed in this article [Dovier et al. 2000b].

## APPENDIX

We recall some technical definitions. Given two  $\Sigma$ -structures  $\mathcal{A}$  and  $\mathcal{B}$ ,  $\mathcal{B} = \langle B, (\cdot)^{\mathcal{B}} \rangle$  is a *substructure* of  $\mathcal{A} = \langle A, (\cdot)^{\mathcal{A}} \rangle$  if  $B \subseteq A$  and for all  $x \in B$  it holds that  $(x)^{\mathcal{A}} = (x)^{\mathcal{B}}$ . Given two  $\Sigma$ -structures  $\mathcal{A}$  and  $\mathcal{B}$ , a function  $h : A \rightarrow B$  is said to be an *homomorphism* from  $\mathcal{A}$  to  $\mathcal{B}$  if

- (i)  $\forall f \in \mathcal{F}, a_1, \dots, a_n \in A (h(f^{\mathcal{A}}(a_1, \dots, a_n)) = f^{\mathcal{B}}(h(a_1), \dots, h(a_n)))$ ;
- (ii)  $\forall p \in \Pi, a_1, \dots, a_m \in A (p^{\mathcal{A}}(a_1, \dots, a_m) \rightarrow p^{\mathcal{B}}(h(a_1), \dots, h(a_m)))$ .

The function  $h$  is said to be an *isomorphism* if  $f$  is bijective and in the property (ii) also the  $\leftarrow$  implication holds. Given two  $\Sigma$ -structures  $\mathcal{A}$  and  $\mathcal{B}$ , an *embedding* of  $\mathcal{A}$  in  $\mathcal{B}$  is an isomorphism from  $\mathcal{A}$  to a substructure of  $\mathcal{B}$ .

**LEMMA A.1.** [Chang and Keisler 1973] *Let  $\mathcal{A}$  and  $\mathcal{B}$  be two  $\Sigma$ -structures and let  $h$  be an embedding of  $\mathcal{A}$  in  $\mathcal{B}$ . If  $\varphi$  is an open formula of  $\mathcal{L} = \langle \Sigma, \mathcal{V} \rangle$ , then for each valuation  $\sigma$  on  $A$ :*

$$\mathcal{A} \models \sigma(\varphi) \leftrightarrow \mathcal{B} \models h(\sigma(\varphi)).$$

**LEMMA A.2.**  *$\mathcal{MSET}$  is a model of the theory  $\mathcal{MSet}$ .*

**PROOF.** For each axioms/axiom schemas (A) of the theory  $\mathcal{MSet}$  we need to prove that  $\mathcal{MSET}$  models (A) (briefly,  $\mathcal{MSET} \models (A)$ ). We give only the sketch of the proof.

- $(K), (W).$ ] The fact that  $\mathcal{MSET}$  is a model of  $(K)$  and  $(W)$  is a consequence of the membership predicate interpretation in  $\mathcal{MSET}$  (cf. point (4) of Definition 5.2).
- $(F_1^m).$ ] This axiom holds in  $\mathcal{MSET}$ , since  $f(t_1, \dots, t_n)$  and  $f(s_1, \dots, s_n)$  belong to the same class in  $\mathcal{MSET}$ , only if for all  $i = 1, \dots, n$  it holds that  $t_i$  and  $s_i$  belong to the same class.
- $(F_2).$ ] It holds, by definition of  $\mathcal{MSET}$ , since terms beginning with different free symbols belong to different classes.
- $(F_3), (F_3^m).$ ] Since each ground term has a finite size, both  $\mathcal{MSET} \models (F_3)$  and  $\mathcal{MSET} \models (F_3^m)$  hold; it can be formally proved by induction on the complexity of the terms.
- $(E_p^m).$ ]  $\mathcal{MSET}$  is a model of  $(E_p^m)$ , since for any equational theory  $E$ ,  $T(\mathcal{F})/\equiv_E$  is a model of  $E$  [Siekmann 1989].
- $(E_k^m).$ ]  $\mathcal{MSET}$  is a model of  $(E_p^m)$ , as seen in the previous point, but it is also the *initial* model, namely, two terms  $s$  and  $t$  are in the same class if and only if  $(E_p^m)$  can prove that  $s = t$ . This is exactly the meaning of the axiom  $(E_k^m)$ .  $\square$

LEMMA A.3. *Let  $\mathcal{M}$  be a model of  $MSet$ . Let  $h : T(\mathcal{F}_{MSet})/\equiv_{E_{MSet}} \longrightarrow \mathcal{M}$  be defined as  $h(\overline{t}) = t^{\mathcal{M}}$ . The function  $h$  is an embedding of  $\mathcal{MSET}$  in  $\mathcal{M}$ .*

PROOF. We will prove the following facts:

- (1) The definition of  $h(\overline{t})$  does not depend on the choice of the representative of the class;
- (2)  $h$  is an homomorphism;
- (3)  $h$  is injective;
- (4) If  $h(\overline{t}) \in^{\mathcal{M}} h(\overline{s})$ , then  $\overline{t} \in^{\mathcal{MSET}} \overline{s}$ .

These facts imply the thesis.

- (1) If  $t_1$  and  $t_2$  are two terms such that  $\overline{t_1} = \overline{t_2}$ , then by definition  $(E_p^m) \models t_1 = t_2$ . Since  $\mathcal{A} \models t_1 = t_2$  holds in every model  $\mathcal{A}$  of  $(E_p^m)$ , then in particular it holds in  $\mathcal{M}$ , that is,  $t_1^{\mathcal{M}} = t_2^{\mathcal{M}}$ .
- (2) We need to prove that

- (a) for all  $f \in \mathcal{F}_{MSet}$  and for all terms  $t_1, \dots, t_n \in T(\mathcal{F}_{MSet})$  it holds that
$$h(f^{\mathcal{MSET}}(\overline{t_1}, \dots, \overline{t_n})) = f^{\mathcal{M}}(h(\overline{t_1}), \dots, h(\overline{t_n})).$$

Now,

$$\begin{aligned}
 h(f^{\mathcal{MSET}}(\overline{t_1}, \dots, \overline{t_n})) &= h(f(t_1, \dots, t_n)) && \text{by fact (1) above} \\
 &= (f(t_1, \dots, t_n))^{\mathcal{M}} && \text{by def. of } h \\
 &= f^{\mathcal{M}}(t_1^{\mathcal{M}}, \dots, t_n^{\mathcal{M}}) && \text{by def. of structure} \\
 &= f^{\mathcal{M}}(h(\overline{t_1}), \dots, h(\overline{t_n})) && \text{by def. of } h.
 \end{aligned}$$

- (b) For all terms  $t$  and  $s$ , if  $\overline{t} \in^{\mathcal{MSET}} \overline{s}$ , then  $h(\overline{t}) \in^{\mathcal{M}} h(\overline{s})$ . From  $\overline{t} \in^{\mathcal{MSET}} \overline{s}$ , using fact 1. above, we have that there is a term  $s'$  in  $\overline{s}$  of the form



$\{[t \mid r]\}$  and that  $h(\overline{s}) = s^M$ . Hence, we have that  $h(\overline{s}) = \{[t^M \mid r^M]\}^M$ ;  $(W)$  ensures that  $h(\overline{t}) = t^M$  belongs to it.

- (3) We prove, by structural induction on  $t_1$ , that if  $h(\overline{t_1}) = h(\overline{t_2})$ , then  $\overline{t_1} = \overline{t_2}$ .

**BASIS.** Let  $t_1$  be a constant  $c$ . Since  $\mathcal{M}$  is a model of axiom schema  $(F_2)$ , it can not be that  $t_2 = f(s_1, \dots, s_n)$ , with  $f$  different from  $c$ . Hence, it must be that  $t_2 = c$ .

**STEP.** Let  $t_1$  be  $f(s_1, \dots, s_n)$ , with  $f \neq [\cdot \mid \cdot]$ . It cannot be  $t_2 \equiv g(r_1, \dots, r_m)$ , with  $g \neq f$ , since  $\mathcal{M}$  is a model of  $(F_2)$ . So, it must be  $t_2 \equiv f(r_1, \dots, r_n)$ , and, by  $(F_1)$ ,  $s_i^M = r_i^M$ , for all  $i \leq n$ . Using the inductive hypothesis we have  $\overline{t_1} = \overline{t_2}$ .

Let  $t_1$  be  $[s_1, \dots, s_n \mid r]$ , with  $r$  not of the form  $[r_1 \mid r_2]$ . Since it cannot be that  $t_2$  is  $f(v_1, \dots, v_n)$  (from the previous case applied to  $t_2$ ), then it must be  $t_2$  is  $[u_1, \dots, u_m \mid v]$ , for some  $v$  not of the form  $[v_1 \mid v_2]$ . Let us assume, by contradiction, that  $\overline{t_1} \neq \overline{t_2}$ , and  $t_1^M = t_2^M$ , while the thesis holds for all terms of lower complexity. From  $t_1^M = t_2^M$  we obtain that the two terms have in  $\mathcal{M}$  the same elements. Since  $\mathcal{M}$  is a model of  $(W)$ , the elements of  $t_1^M$  are exactly  $s_1^M, \dots, s_n^M$  and the elements of  $t_2^M$  are exactly  $u_1^M, \dots, u_m^M$ . So, by inductive hypothesis, there is a bijection  $b : \{1, \dots, n\} \rightarrow \{1, \dots, m\}$  such that  $\overline{s_i} = \overline{u_{b(i)}}$ . This means that  $m = n$  and that there is a term  $t'_2$  in  $\overline{t_2}$  of the form  $[s_1, \dots, s_n \mid v]$ . Applying  $n$  times  $(E_k^m)$ , in all possible ways, we obtain that  $r^M = v^M$ , hence by inductive hypothesis  $\overline{r} = \overline{v}$ . From this fact, we conclude that  $\overline{t_2} = \overline{t'_2} = \overline{[s_1, \dots, s_n \mid r]} = \overline{t_1}$ , which is in contradiction with our assumption.

- (4) If  $h(\overline{t}) \in^M h(\overline{s})$ , then  $t^M \in^M s^M$  and hence  $(K)$  implies that  $s$  must be a term of the form  $[t_1 \mid t_2]$ . By induction on  $s$  using  $(W)$ , we can prove that in particular  $s$  must be a term of the form  $[t_1, \dots, t_i, \dots, t_n \mid r]$ , with  $t_1^M = t^M = h(\overline{t})$ . We have already proved that  $h$  is injective; hence it must be  $t_1 \in \overline{t}$ , and from this we obtain  $\overline{t} \in^{MSE\mathcal{T}} \overline{s}$ .  $\square$

We say that a linear disequality  $a \neq b$  over the integers is *safe* if, after expressions evaluation, it is not of the form  $u \neq u$ . We say that a system  $A$  of linear disequations over the integers with variables  $x_1, \dots, x_h$  is *safe* if each disequation in  $A$  is either a safe disequality or it is of the form  $x_i > m$ , where  $m$  is an integer number.

**LEMMA A.4.** *Let  $A$  be a safe system of linear disequations over the integers.  $A$  has always an infinite number of solutions.*

**PROOF.** Let us partition the system  $A$  into two systems  $A_{\neq}$ , which contains all the disequalities of  $A$ , and  $A_{>}$  which contains all the disequations of the form  $x_i > m$  of  $A$ . We proceed by induction on the number of variables in  $A$ .

If in  $A$  there is only one variable  $x_1$ , then we can rewrite all the disequalities of  $A_{\neq}$  in the form  $x_1 \neq a_i$ , where  $a_i$  is a rational number. Let  $max$  be the maximum of all the  $a_i$  and of all the integers occurring in  $A_{>}$ . All the integers greater of  $max$  are solutions of  $A$ .

If in  $A$  there are  $n$  variables  $x_1, \dots, x_n$ , then we concentrate on the variable  $x_1$ . Each disequality of  $A_{\neq}$  can be rewritten in the form  $a_{i,1}x_1 \neq p_i(x_2, \dots, x_n)$ , where  $p_i(x_2, \dots, x_n) = a_{i,2}x_2 + \dots + a_{i,n}x_n + a_{i,n+1}$  is a linear expression with integer coefficients over the variables  $x_2, \dots, x_n$ . Let  $max$  be the maximum of all the  $|a_{i,j}|$  and of all the integers occurring in  $A_{>}$ . We assign to  $x_1$  value  $max + 1$ . We prove that the system  $A'$  obtained from  $A$  by replacing  $x_1$  with  $max + 1$  is a safe system in  $n - 1$  variables. To prove this we have to prove that all the disequalities in  $A'$  are safe. If in  $A$  there is a disequality of the form  $a_{i,1}x_1 \neq a_{i,n+1}$ , then in  $A'$  we have a disequality of the form  $a_{i,1}(max + 1) \neq a_{i,n+1}$  which is not reducible to  $u \neq u$  since the absolute value on the left side is greater than that on the right side. If in  $A$  there is a disequality of the form  $a_{i,1}x_1 \neq a_{i,2}x_2 + \dots + a_{i,n}x_n + a_{i,n+1}$  with at least one of the  $a_{i,2}, \dots, a_{i,n}$  different from 0, then this trivially become a safe disequality in  $A'$ . Now we have that each solution of  $A'$  completed with  $x_1 = max + 1$  is a solution of  $A$ . Since by inductive hypothesis  $A'$  has an infinite number of solutions,  $A$  has an infinite number of solutions.  $\square$

LEMMA A.5. In  $KW E_k^m F_1^m F_2 F_3^3 F_3^m$  it holds that

- (1)  $\{[nil]\}^x = \{[nil]\}^y$  if and only if  $x = y$ ;
- (2)  $\{[nil]\}^x \in \{[nil]\}^y$  if and only if  $x = y - 1$ .

PROOF. Let us prove (1). If  $x = y$ , then we immediately get the thesis, since the two terms are syntactically the same. On the other hand, let us assume that in  $KW E_k^m F_1^m F_2 F_3^3 F_3^m$  we can prove  $\{[nil]\}^x = \{[nil]\}^y$ . We can safely assume that  $x \geq y \geq 0$ . We proceed by induction on  $x$ . If  $x = 0$ , then we immediately have  $y = 0$ . If  $x > 0$ , then when we apply axiom  $E_k^m$  to  $\{[nil]\}^x = \{[nil]\}^y$  we get  $\{[nil]\}^{x-1} = \{[nil]\}^{y-1}$ . By inductive hypothesis this implies  $x - 1 = y - 1$ , and hence  $x = y$ .

The proof of (2) can be similarly done exploiting axiom  $W$  instead of axiom  $E_k^m$ .  $\square$

## REFERENCES

- ALIFFI, D., DOVIER, A., AND ROSSI, G. 1999. From set to hyperset unification. *J. Funct. Logic Program.* 10, 1–48.
- BAADER, F. AND BÜTTNER, W. 1988. Unification in commutative and idempotent monoids. *Theoret. Comput. Sci.* 56, 345–352.
- BAADER, F. AND NIPKOW, T. 1998. *Term Rewriting and All That*. Cambridge University Press, Cambridge, U.K.
- BAADER, F. AND SCHULZ, K. U. 1995. Combination techniques and decision problems for disunification. *Theoret. Comput. Sci.* 142, 229–255.
- BAADER, F. AND SCHULZ, K. U. 1996. Unification in the union of disjoint equational theories: Combining decision procedures. *J. Symbol. Computat.* 21, 211–243.
- BANATRE, J. AND MÉTAYER, D. L. 1993. Programming by multiset transformation. *Commun. ACM* 36, 1, 98–111.
- BEERI, C., NAQVI, S., SHMUELI, O., AND TSUR, S. 1991. Set constructors in a logic database language. *J. Logic Program.* 10, 3, 181–232.
- BÜCKERT, H. 1988. Solving disequations in equational theories. In *Proceedings of the International Conference on Automated Deduction*, E. L. Lusk and R. A. Overbeek, Eds. Lecture Notes in Computer Science, vol. 310. Springer, Argonne, IL, 517–526.

- BÜTTNER, W. 1986. Unification in the data structure Sets. In *Proceedings of the International Conference on Automated Deduction*, J. K. Siekmann, Ed. Lecture Notes in Computer Science, vol. 230. Springer, Oxford, U.K., 470–488.
- CANTONE, D., OMODEO, E. G., AND POLICRITI, A. 2001. *Set Theory for Computing. From Decision Procedures to Declarative Programming with Sets*. Monographs in Computer Science. Springer, New York, NY.
- CHANG, C. C. AND KEISLER, H. J. 1973. *Model Theory*. Studies in Logic. North Holland, Amsterdam, The Netherlands.
- CLARK, K. L. 1978. Negation as Failure. In *Logic and Databases*, H. Gallaire and J. Minker, Eds. Plenum Press, New York, NY, 293–321.
- COMON, H. 1993. Complete Axiomatizations of some quotient term algebras. *Theoret. Comput. Sci.* 118, 2, 167–191.
- CORBIN, J. AND BIDOIT, M. 1983. A rehabilitation of Robinson’s unification algorithm. In *Information Processing, IFIP 9th World Computer Congress*, R. Mason, Ed. North Holland, Paris, France, 909–914.
- DANTSIN, E. AND VORONKOV, A. 1999. A nondeterministic polynomial-time unification algorithm for bags, sets and trees. In *Foundations of Software Science and Computation Structure*, W. Thomas, Ed. Lecture Notes in Computer Science, vol. 1578. Springer, Amsterdam, The Netherlands, 180–196.
- DERSHOWITZ, N. AND MANNA, Z. 1979. Proving termination with multiset ordering. *Commun. ACM* 22, 8, 465–476.
- DOVIER, A., OMODEO, E. G., PONTELLI, E., AND ROSSI, G. 1996. {log}: A Language for Programming in Logic with Finite Sets. *J. Logic Program.* 28, 1, 1–44.
- DOVIER, A., PIAZZA, C., AND POLICRITI, A. 2000a. Comparing expressiveness of set constructor symbols. In *Frontiers of Combining Systems*, H. Kirchner and C. Ringeissen, Eds. Lecture Notes in Computer Science, vol. 1794. Springer, Nancy, France, 275–289.
- DOVIER, A., PIAZZA, C., AND PONTELLI, E. 2004. Disunification in ACI1 theories. *Constraints (Internat. J.)* 9, 1, 35–91.
- DOVIER, A., PIAZZA, C., PONTELLI, E., AND ROSSI, G. 2000b. Sets and constraint logic programming. *ACM Trans. Program. Lang. Syst.* 22, 5, 861–931.
- DOVIER, A., PIAZZA, C., AND ROSSI, G. 2003. A uniform approach to constraint-solving for lists, multisets, compact lists, and sets. CoRR cs.PL/030945.
- DOVIER, A., POLICRITI, A., AND ROSSI, G. 1998. A uniform axiomatic view of lists, multisets, and sets, and the relevant unification algorithms. *Fundamenta Informaticae* 36, 2/3, 201–234.
- DOVIER, A., PONTELLI, E., AND ROSSI, G. 2001. Set unification. CoRR cs.LO/0110023. To appear in *Theory and Practice of Logic Programming*.
- DOVIER, A. AND ROSSI, G. 1993. Embedding Extensional Finite Sets in CLP. In *International Logic Programming Symposium*, D. Miller, Ed. MIT Press, Cambridge, MA, 540–556.
- ENDERTON, H. B. 1973. *A Mathematical Introduction to Logic*, 2nd ed. Academic Press, New York, NY.
- GERVET, C. 1997. Interval propagation to reason about sets: Definition and implementation of a practical language. *Constraints (Int. J.)* 1, 191–246.
- GRUMBACH, S. AND MILO, T. 1996. Towards tractable algebras for bags. *J. Comput. Syst. Sci.* 52, 3, 570–588.
- HILL, P. M. AND LLOYD, J. W. 1994. *The Gödel Programming Language*. MIT Press, Cambridge, MA.
- JAFFAR, J. AND MAHER, M. J. 1994. Constraint logic programming: A survey. *J. Logic Program.* 19–20, 503–581.
- LIVESEY, M. AND SIEKMANN, J. 1976. Unification of sets and multisets. Tech. rep. Institut für Informatik I, Universität Karlsruhe, Karlsruhe, Germany.
- MAŁCZEV, A. 1971. Axiomatizable classes of locally free algebras of various types. In *The Mathematics of Algebraic Systems*. North Holland, Amsterdam, The Netherlands, Chapter 23, 262–281.
- PÄUN, G. 2000. Computing with membranes. *J. Comput. Syst. Sci.* 61, 1, 108–143.
- POTTER, B., SINCLAIR, J., AND TILL, D. 1996. *An Introduction to Formal Specification and Z*, 2nd ed. Prentice Hall, Upper Saddle River, NJ.

- SCHWARTZ, J. T., DEWAR, R. B. K., AND SCHONBERG, E. 1986. *Programming with Sets: An Introduction to SETL*. Springer, Berlin, Germany.
- SIEKMANN, J. H. 1989. Unification theory: A survey. *J. Symbol. Computat.* (Special issue on unification) 7, 3,4, 207–274.
- TZOUVARAS, A. 1998. The linear logic of multisets. *Logic J. IGPL* 6, 6, 901–916.

Received May 2005; Revised May 2006; accepted October 2006