A uniform approach to constraint-solving for lists, multisets, compact lists, and sets*

Agostino Dovier† Carla Piazza† Gianfranco Rossi‡

October 26, 2006

Abstract

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 [20] where the relevant unification algorithms have been developed. In this paper we extend these results considering more general constraints, namely equality and membership constraints and their negative counterparts.

Keywords: Membership and Equality Constraints, Lists, Multisets, Compact Lists, Sets.

1 Introduction

Programming and specification languages allow to represent information by means of data structures, each of them characterized by a specific organization of the elements involved and by a corresponding access policy. In this paper 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 [33]) and in high-level declarative programming languages [7, 22, 26, 28]; moreover imperative programming languages may take advantage from the set data abstraction (e.g., SETL [34]). Multisets emerge as the most natural data structure in several areas, ranging from coordination languages [6] to Database theory [27], from membrane and DNA computing modeling [32] to linear logic [37]. The notion of compact list is much less developed and some examples of its application are suggested in [20].

Lists, multisets, compact lists, and sets have been analyzed from an axiomatic point of view. In [20], 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 to a theory which specifies the properties of the aggregate constructor symbol.

*This work is partially supported by MIUR Project PRIN 2005015491.

†Dip. di Matematica e Informatica, Università di Udine. Via delle Scienze 206, 33100 Udine (Italy). dovier|piazza@dimi.uniud.it

‡Dip. di Matematica, Università di Parma. Parco Area delle Scienze 53/A, 43100 Parma (Italy). gianfranco.rossi@unipr.it

In [20], *equalities* between terms in each of the four theories are studied. In particular, the unification problems in the equational theories, which describe the properties of the four aggregates, are solved by providing unification algorithms for all of them. NP-unification algorithms for sets and multisets are also presented in [1, 15].

In this paper we extend the results presented in [20] 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 paper 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 [20] 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 in 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 Works

The problem of *set and multiset unification* has been tackled by several authors, using different representations (see [21] for a survey on the set unification problem). These problems are often reduced to ACI and AC unification problems, respectively (see, e.g., [9, 30]). 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., [3]). Equational theories which also allow 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—[2]) and the combining approach developed in [4, 5]. This approach, however, due to its generality, tends to produce a huge number of failing non-deterministic 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 [8], where a technique to transform disequations into universally quantified unification problems is presented. The method described in [8] 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 [13]. Unfortunately, the same results cannot be applied to ACI theories, hence to sets. These theories are studied in [18] where constraint solving procedures are developed.

As far as membership and not-membership 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 not-membership 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 $\forall X\,(\{\!|\, t \,|\!\} \uplus X \neq s)$, i.e., 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 not-membership 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 [17] some comparisons between the union-like and list-like representations are 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 is 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, are presented in [24, 23]. In [20] we consider the four data structures considered in this paper, 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 multi-level syllogistics, studied in [10], where axioms for sets are not simply equational axioms. However, [10] is mainly concerned with decidability results rather than with constraint solving procedures.

In this paper 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., [11, 25]) is assumed. We fix some notations and recall some basic notions that will be used throughout the paper.

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 Π 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 $\mathsf{FV}(\bar{t})$ is the set of all the variables which occur in the terms $\bar{t}$. If φ is a first-order formula, $\mathsf{FV}(\varphi)$ is the set of free variables in φ. 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 φ, namely $\exists X_1 \cdots \exists X_n\, \varphi$ ($\forall X_1 \cdots \forall X_n\, \varphi$, respectively), where $\{X_1, \ldots, X_n\} = \mathsf{FV}(\varphi)$. An axiom is a closed first-order formula. If $\Theta = \{\varphi_1, \ldots, \varphi_n\}$ is a set of axioms and $A_1, \ldots, A_n$ are names for the axioms $\varphi_1, \ldots, \varphi_n$, we refer to Θ simply as $A_1 \cdots A_n$. 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 φ is any first-order formula, X and Y are free in φ, and φ' is obtained from φ by replacing zero or more occurrences of X with Y [11, 25].

An *equational axiom* is a formula of the form $\forall(\ell = r)$ where ℓ and r are terms. An *equational theory* E is an axiomatization whose axioms are equational axioms. Given two terms ℓ and r, we write $\ell \approx_E r$ if the axioms in E can prove that ℓ is equal to r. A *system of equations* $\mathcal{S}$ is a conjunction of equations $\ell_1 = r_1 \wedge \cdots \wedge \ell_n = r_n$. An *$E$-solution* (a solution, when the context is clear) of $\mathcal{S}$ is a substitution σ, which replaces variables with ground terms, such that for all $i \in \{1, \ldots, n\}$ 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 Σ on A. A *valuation* σ is a function from a subset of the set of variables $\mathcal{V}$ to A. When Σ is given, σ can

be uniquely extended to terms, and allows to assign truth values to formulas. A valuation σ is said to be *successful* for φ if $\sigma(\varphi) = \texttt{true}$ (briefly, $\mathcal{A} \models \sigma(\varphi)$). A formula φ is *satisfiable* in $\mathcal{A}$, denoted by $\mathcal{A} \models \exists\varphi$, if there exists a valuation σ such that $\mathcal{A} \models \sigma(\varphi)$. We say that $\mathcal{A} \models \varphi$ if for every valuation σ from $\mathsf{FV}(\varphi)$ to A it holds that $\mathcal{A} \models \sigma(\varphi)$. Two formulas C_1 and C_2 are *equi-satisfiable* 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 φ 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 [20] the first-order axiomatic theories for the four aggregates. Each theory has its own signature. Precisely, Π is $\{=, \in\}$ and $\mathcal{F}$ contains (at least) the constant symbol `nil` and exactly one among the following binary function symbols:

$[\,\cdot \mid \cdot\,]$	for lists,	$\{\!\{\,\cdot \mid \cdot\,\}\!\}$	for multisets,
$[\![\,\cdot \mid \cdot\,]\!]$	for compact lists,	$\{\,\cdot \mid \cdot\,\}$	for sets.

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 $\{\!\{\, s_1 \mid \{\!\{\, s_2 \mid \cdots \{\!\{\, s_n \mid t \,\}\!\} \cdots \}\!\}\,\}\!\}$ will be denoted by $\{\!\{\, s_1, \ldots, s_n \mid t \,\}\!\}$ or by $\{\!\{\, s_1, \ldots, s_n \,\}\!\}$ 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 $\Sigma_{List} = \langle \mathcal{F}_{List}, \Pi \rangle$, $[\,\cdot \mid \cdot\,]$ 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)	$\forall X\, Y_1 \cdots Y_n$	$(X \notin f(Y_1, \ldots, Y_n))$	$f \in \mathcal{F}_{List}, f$ is not $[\,\cdot \mid \cdot\,]$
(W)	$\forall Y\, V\, X$	$(X \in [\,Y \mid V\,] \leftrightarrow X \in V \vee X = Y)$	
(F_1)	$\forall X_1 \cdots X_n Y_1 \cdots Y_n$	$\left(\begin{array}{c} f(X_1, \ldots, X_n) = f(Y_1, \ldots, Y_n) \\ \rightarrow X_1 = Y_1 \wedge \cdots \wedge X_n = Y_n \end{array}\right)$	$f \in \mathcal{F}_{List}$
(F_2)	$\forall X_1 \cdots X_m Y_1 \cdots Y_n$	$(f(X_1, \ldots, X_m) \neq g(Y_1, \ldots, Y_m))$	$f, g \in \mathcal{F}_{List}, f$ is not g
(F_3)	$\forall X$	$(X \neq t[X])$	
	where $t[X]$ *denotes a term* t *having* X *as proper subterm*		

The three axiom schemas $(F_1), (F_2)$, and (F_3) (called *freeness axioms*, or *Clark's equality axioms*—see [12]) have been originally introduced by Mal'cev in [31]. Since $[\,\cdot \mid \cdot\,]$ belongs to $\mathcal{F}_{List}$, axiom schema (F_1) holds for $[\,\cdot \mid \cdot\,]$ as a particular case. (F_3) states that there is no term which is a proper subterm of itself (occurs check). Notice that (K) implies that $\forall X\, (X \notin \texttt{nil})$.

4.2 Multisets

The language $\mathcal{L}_{MSet}$ is defined as $\langle \Sigma_{MSet}, \mathcal{V} \rangle$, where $\Sigma_{MSet} = \langle \mathcal{F}_{MSet}, \Pi \rangle$, $\{\!\{\,\cdot \mid \cdot\,\}\!\}$ 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 $[\,\cdot \mid \cdot\,]$ is replaced by the constructor $\{\!\{\,\cdot \mid \cdot\,\}\!\}$ in axiom schema (K) and axiom (W). The behavior of this new symbol is regulated by the following equational axiom

(E_p^m)	$\forall X Y Z\ \ \{\!\{\, X, Y \mid Z \,\}\!\} = \{\!\{\, Y, X \mid Z \,\}\!\}$	*(permutativity)*

which intuitively states that the order of elements in a multiset is immaterial. Axiom schema (F_1) does not hold for multisets, when f is $\{\![\, \cdot \,|\, \cdot \,]\!\}$. It is replaced by axiom schema (F_1^m):

$$\boxed{\begin{array}{ll} (F_1^m) & \forall X_1 \cdots X_n Y_1 \cdots Y_n \left(\begin{array}{c} f(X_1, \ldots, X_n) = f(Y_1, \ldots, Y_n) \\ \rightarrow X_1 = Y_1 \wedge \cdots \wedge X_n = Y_n \end{array} \right) \\ & \textit{for any } f \in \mathcal{F}_{MSet}, \textit{ f is not } \{\![\, \cdot \,|\, \cdot \,]\!\} \end{array}}$$

The theory $KWE_p^m F_1^m F_2 F_3$, 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 [20] to force this property is the following:

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

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

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

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

$$\boxed{(E_a^c) \qquad \forall XY \;\; [\![\, X, X \,|\, Y \,]\!] = [\![\, X \,|\, Y \,]\!] \quad (\textit{absorption})}$$

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:

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

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

$$\boxed{\begin{array}{ll} (F_1^c) & \forall X_1 \cdots X_n Y_1 \cdots Y_n \left(\begin{array}{c} f(X_1, \ldots, X_n) = f(Y_1, \ldots, Y_n) \\ \rightarrow X_1 = Y_1 \wedge \cdots \wedge X_n = Y_n \end{array} \right) \\ & \textit{for any } f \in \mathcal{F}_{CList}, \textit{ f is not } [\![\, \cdot \,|\, \cdot \,]\!] \end{array}}$$

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 Σ-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 $X = [\![\mathtt{nil} \,|\, X]\!]$ admits a successful valuation over compact lists. Precisely, a valuation that binds X to the term $[\![\mathtt{nil} \,|\, t]\!]$, where t is any term. Therefore, axiom schema (F_3) is weakened as follows:

$$\boxed{\begin{array}{ll}(F_3^c) & \forall X \quad (X \neq t[X]) \\ & \textit{unless: } t \textit{ is of the form } [\![t_1, \ldots, t_n \,|\, X]\!], \textit{ with } n > 0, \\ & X \notin \mathsf{FV}(t_1, \ldots, t_n), \textit{ and } t_1 = \cdots = t_n\end{array}}$$

4.4 Sets

The language $\mathcal{L}_{Set}$ is defined as $\mathcal{L}_{Set} = \langle \Sigma_{Set}, \mathcal{V} \rangle$, where $\Sigma_{Set} = \langle \mathcal{F}_{Set}, \Pi \rangle$, $\{ \cdot \,|\, \cdot \}$ and $\mathtt{nil}$ 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 $\{ \cdot \,|\, \cdot \}$, can be rewritten as follows:

$$\boxed{\begin{array}{lrcl}(E_p^s) & \forall XYZ \;\; \{X, Y \,|\, Z\} & = & \{Y, X \,|\, Z\} \\ (E_a^s) & \forall XY \;\; \{X, X \,|\, Y\} & = & \{X \,|\, Y\}\end{array}}$$

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):

$$\boxed{(E_k^s) \qquad \forall Y_1 Y_2 V_1 V_2 \quad \left(\begin{array}{r} \{Y_1 \,|\, V_1\} = \{Y_2 \,|\, V_2\} \;\; \leftrightarrow \\ (Y_1 = Y_2 \wedge V_1 = V_2) \vee \\ (Y_1 = Y_2 \wedge V_1 = \{Y_2 \,|\, V_2\}) \vee \\ (Y_1 = Y_2 \wedge \{Y_1 \,|\, V_1\} = V_2) \vee \\ \exists Z \, (V_1 = \{Y_2 \,|\, Z\} \wedge V_2 = \{Y_1 \,|\, Z\}) \end{array} \right)}$$

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 [20] it is also proved that they are equivalent in all Σ-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 $[\cdot \,|\, \cdot]$ replaced by $\{ \cdot \,|\, \cdot \}$, and axiom schemas (F_2). Axiom schema (F_1) is replaced by:

$$\boxed{\begin{array}{ll}(F_1^s) & \forall X_1 \cdots X_n Y_1 \cdots Y_n \quad \left(\begin{array}{c} f(X_1, \ldots, X_n) = f(Y_1, \ldots, Y_n) \\ \rightarrow X_1 = Y_1 \wedge \cdots \wedge X_n = Y_n \end{array} \right) \\ & \textit{for any } f \in \mathcal{F}_{Set}, \; f \textit{ is not } \{ \cdot \,|\, \cdot \}\end{array}}$$

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

$$\boxed{\begin{array}{ll}(F_3^s) & \forall X \quad (X \neq t[X]) \\ & \textit{unless: } t \textit{ is of the form } \{t_1, \ldots, t_n \,|\, X\} \textit{ and } X \in \mathsf{FV}(t_1, \ldots, t_n)\end{array}}$$

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).

Figure 1 summarizes 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.

Theory	empty	with	Equality		Herbr.	Acycl.		Perm.	Abs.	Eq. Theory
List	(K)	(W)	(F_1)		(F_2)	(F_3)				E_{List}
MSet	(K)	(W)	(E_k^m)	(F_1^m)	(F_2)	(F_3)	(F_3^m)	(E_p^m)		E_{MSet}
CList	(K)	(W)	(E_k^c)	(F_1^c)	(F_2)	(F_3^c)			(E_a^c)	E_{CList}
Set	(K)	(W)	(E_k^s)	(F_1^s)	(F_2)	(F_3^s)		(E_p^s)	(E_a^s)	E_{Set}

Figure 1: Axioms for the four theories. From left to right, the name of the first-order theory, the first-order axiom schemas, the equational axioms and the name of the equational theories.

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 paper 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), i.e. 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* Σ-structure $\mathcal{MSET}$ for *MSet is defined as follows.*

1. *The* domain *of the Σ-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 $\textcircled{t}$ 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: $\textcircled{t} \in \textcircled{s}$ is* `true` *if and only if there is a term of the form $\{\!\{ t_1, \ldots, t_n, t \mid r \}\!\}$ in $\textcircled{s}$.*

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 $\mathcal{LIST}$, $\mathcal{CLIST}$, and $\mathcal{SET}$. 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 ([29]) *Let $\mathcal{L} = \langle \Sigma, \mathcal{V} \rangle$ be a a first-order language, $\mathcal{T}$ be a theory on $\mathcal{L}$, $\mathcal{A}$ be a Σ-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 φ is an element of $\mathcal{C}$ and φ 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 σ 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 $MSet \models \exists C$. □

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 $C = c_1 \wedge \cdots \wedge c_n$ is in* solved form *if for $i \in \{1, \ldots, n\}$, 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 $X = \{\!|\, t \,|\, N \,|\!\}$ ($X = \{t \,|\, N\}$, 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

$$[\![\, a \,|\, N \,]\!] \in Y \wedge [\![\, a, a \,|\, N \,]\!] \notin Y$$

is unsatisfiable in $\mathcal{CLIST}$, since $[\![\, a \,|\, N \,]\!]$ and $[\![\, a, a \,|\, N \,]\!]$ 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 known from unification theory (see, e.g., [3, 35]) 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 (ε). Thus, test (ii) is connected with the availability of a unification algorithm for the theory E. In [20] 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 [19].

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 the Solved Form) *Let C be a MSet-constraint in solved form. Then $\mathcal{MSET} \models \exists C$.*

Proof. We split C into the three parts: $C^{=}$, $C^{\notin}$, and $C^{\neq}$, containing $=, \notin$, 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:

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

$find(X, t)$ 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:

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

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

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

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

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

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

1. $S = \{n_i > \bar{s} : \forall i \in \{1,\ldots,j\}\} \cup \{n_{i_1} \neq n_{i_2} : \forall i_1, i_2 \in \{1,\ldots,j\}, i_1 \neq i_2\}$
2. For each literal $R_i \neq s$ in $\theta_2(C^{\neq})$ and for all k in $\{1,\ldots,j\}$, $i \neq k$
$$S = S \cup \{n_i \neq n_k + c : \forall c \in find(R_k, s)\}$$
3. For each literal $r \notin R_i$ in $\theta_2(C^{\notin})$ and for all k in $\{1,\ldots,j\}$, $i \neq k$
$$S = S \cup \{n_i \neq n_k + c + 1 : \forall c \in find(R_k, r)\}$$

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 $x_1, \ldots, x_h$ 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 $\eta = \{n_1 = \bar{n}_1, \ldots, n_j = \bar{n}_j\}$ be a solution and define

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

where $\{\!\{\texttt{nil}\}\!\}^{\bar{n}}$ denotes the term $\underbrace{\{\!\{\cdots\{\!\{\texttt{nil}\}\!\}\cdots\}\!\}}_{\bar{n}}$.

It is immediate to see that in $KWE_k^m F_1^m F_2 F^3 F_3^m$ it holds that $\{\!\{\texttt{nil}\}\!\}^x = \{\!\{\texttt{nil}\}\!\}^y$ if and only if $x = y$ and $\{\!\{\texttt{nil}\}\!\}^x \in \{\!\{\texttt{nil}\}\!\}^y$ 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 $KWE_k^m F_1^m F_2 F^3 F_3^m \models C\gamma$. We analyze each literal of C.

$X = t$: $\theta_1(X)$ syntactically coincides with $\theta_1(t) = t$. The substitution θ_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 $\{\!|\,\mathtt{nil}\,|\!\}^p$ for some $p < \bar{s}$. $Z\gamma$ is of form $\{\!|\,\mathtt{nil}\,|\!\}^n$ 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 $\{\!|\,\mathtt{nil}\,|\!\}^n$ for some $n > \bar{s}$. Since $W_i\gamma = \mathtt{nil}$, 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 $\mathit{find}(A, t) = \{p\}$. This means that the the constraint $n_Z \neq n_A + p$ is introduced in S and satisfied by the assignment η. Thus $Z\theta = \{\!|\,\mathtt{nil}\,|\!\}^{\underline{n_Z}}$ and $t\theta = \{\!|\,\mathtt{nil}\,|\!\}^{\underline{n_A}+p}$. Since $\underline{n_Z} \neq \underline{n_A} + p$ 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 $\{\!|\,\mathtt{nil}\,|\!\}^p$ for some $p < \bar{s}$. $Y\gamma$ is of form $\{\!|\,\mathtt{nil}\,|\!\}^n$ 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 $\{\!|\,\mathtt{nil}\,|\!\}^n$ for some $n > \bar{s}$. Since $W_i\gamma = \mathtt{nil}$, 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 $\mathit{find}(X, t) = \{p\}$. This means that the the constraint $n_Y \neq n_X + p + 1$ is introduced in S and satisfied by the assignment η. Thus $Y\theta = \{\!|\,\mathtt{nil}\,|\!\}^{\underline{n_Y}}$ and $r\theta = \{\!|\,\mathtt{nil}\,|\!\}^{\underline{n_X}+p}$. Since $\underline{n_Y} \neq \underline{n_X} + p + 1$ 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.

□

Hence a solved-form constraint can be seen as a symbolic representation for a non-empty and possibly infinite set of valuations, i.e., 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 equi-satisfiable 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 2: 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 non-deterministic. Some rewriting rules have two or more possible non-deterministic choices. Each non-deterministic computation returns a constraint in solved form or `false`. Globally, the procedure returns a finite collection $C_1, \ldots, C_k$ of constraints. The input constraint C and the disjunction $C_1 \vee \cdots \vee C_k$ are equi-satisfiable in the standard models. We show the details for the multiset case only. Details for the other procedures can be found in [19].

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 [20]. These algorithms fall in the general schema of Figure 2. Some determinism in the statement select c is added to ensure termination. They are called:

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

```
while C contains an atomic constraint c of the form ℓ π r not in solved form and C ≠ false do
    select c;
    if c = false then return false
    else if c = true then erase c
    else apply to c any rewriting rule for T-constraints of the form ℓ π r;
return C
```

Figure 2: Main loop of constraint rewriting procedures

unify-*List* for lists	unify-*MSet* (called unify-bags in [20]) for multisets
unify-*CList* for compact lists	unify-*Set* for sets

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

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 3 we show the rewriting rules for multiset unification.

Rules for unify-*MSet*		
(1)	$X = X$	$\mapsto$ `true`
(2)	$t = X$ t is not a variable	$\mapsto$ $X = t$
(3)	$X = t$ $X \notin \mathsf{FV}(t)$, X occurs elsewhere in C	$\mapsto$ $X = t$ and apply the substitution $[X/t]$ to C
(4)	$X = t$ t is not X, $X \in \mathsf{FV}(t)$	$\mapsto$ `false`
(5)	$f(s_1, \ldots, s_m) = g(t_1, \ldots, t_n)$ f is not g	$\mapsto$ `false`
(6)	$f(s_1, \ldots, s_m) = f(t_1, \ldots, t_m)$ $m \geq 0, f$ is not $\{\!\{ \cdot \mid \cdot \}\!\}$	$\mapsto$ $s_1 = t_1 \wedge \ldots \wedge s_m = t_m$
(7)	$\{\!\{ t \mid s \}\!\} = \{\!\{ t' \mid s' \}\!\}$ $\mathsf{tail}(s)$ and $\mathsf{tail}(s')$ are not the same variable	$\mapsto$ $(i) \quad (t = t' \wedge s = s') \vee$ $(ii) \quad (s = \{\!\{ t' \mid N \}\!\} \wedge \{\!\{ t \mid N \}\!\} = s')$
(8)	$\{\!\{ t \mid s \}\!\} = \{\!\{ t' \mid s' \}\!\}$ $\mathsf{tail}(s)$ and $\mathsf{tail}(s')$ are the same variable	$\mapsto$ $\mathsf{untail}(\{\!\{ t \mid s \}\!\}) = \mathsf{untail}(\{\!\{ t' \mid s' \}\!\})$

Figure 3: Rewriting rules of the unification algorithm for multisets

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

$$
\begin{array}{lcll}
\mathsf{tail}(f(t_1, \ldots, t_n)) & = & f(t_1, \ldots, t_n) & f \text{ is not } \{\!\{ \cdot \mid \cdot \}\!\},\ n \geq 0 \\
\mathsf{tail}(X) & = & X & X \text{ is a variable} \\
\mathsf{tail}(\{\!\{ t \mid s \}\!\}) & = & \mathsf{tail}(s) & \\
\mathsf{untail}(X) & = & \texttt{nil} & X \text{ is a variable} \\
\mathsf{untail}(\{\!\{ t \mid s \}\!\}) & = & \{\!\{ t \mid \mathsf{untail}(s) \}\!\} &
\end{array}
$$

The following lemma, which states the soundness and completeness of the unification rules, has been proved in [20]. We report here the proof for the sake of completeness.

Lemma 6.1 ([20]) *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, $C_1, \ldots, C_k$ be the constraints non-deterministically returned by* unify-$\mathbb{T}(C)$*, and $\bar{N}_i = \mathsf{FV}(C_i) \setminus \mathsf{FV}(C)$. Then $\mathcal{A}_\mathbb{T} \models \forall \left(C \leftrightarrow \bigvee_{i=1}^{k} \exists \bar{N}_i C_i \right)$.*

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 funcion untail replace the variable that occurs as tail of the two multisets with `nil`).

□

Remark 6.2 *Consider the constraint*

$$\{\!\{ a \,|\, X \}\!\} = \{\!\{ b \,|\, Y \}\!\} \wedge \{\!\{ d \,|\, X \}\!\} = \{\!\{ e \,|\, Y \}\!\}$$

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

$$X = \{\!\{ b \,|\, N_1 \}\!\} \wedge \{\!\{ a \,|\, N_1 \}\!\} = Y \wedge X = \{\!\{ e \,|\, N_2 \}\!\} \wedge \{\!\{ d \,|\, N_2 \}\!\} = Y$$

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

$$X = \{\!\{ b \,|\, N_1 \}\!\} \wedge Y = \{\!\{ a \,|\, N_1 \}\!\} \wedge \{\!\{ b \,|\, N_1 \}\!\} = \{\!\{ e \,|\, N_2 \}\!\} \wedge \{\!\{ d \,|\, N_2 \}\!\} = \{\!\{ a \,|\, N_1 \}\!\}$$

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 non-termination. However, a simple selection strategy is sufficient to avoid this problem. From the initial system, apply rule (7*ii*) *to the first equation and then the two substotutions induced:*

$$X = \{\!\{ b \,|\, N_1 \}\!\} \wedge Y = \{\!\{ a \,|\, N_1 \}\!\} \wedge \{\!\{ d, b \,|\, N_1 \}\!\} = \{\!\{ e, a \,|\, N_1 \}\!\}$$

Then rule (8) *can be used removing "tail" variables:*

$$X = \{\!\{ b \,|\, N_1 \}\!\} \wedge Y = \{\!\{ a \,|\, N_1 \}\!\} \wedge \{\!\{ d, b \}\!\} = \{\!\{ e, a \}\!\}$$

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 easy implemented using a stack. For more details, see [20].

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 2, 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 4 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.

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, $C_1, \ldots, C_k$ be the constraints non-deterministically returned by* nin-$\mathbb{T}$(in-$\mathbb{T}(C)$)*, and $\bar{N}_i = \mathsf{FV}(C_i) \setminus \mathsf{FV}(C)$. Then $\mathcal{A}_\mathbb{T} \models \forall \left(C \leftrightarrow \bigvee_{i=1}^{k} \exists \bar{N}_i C_i \right)$.*

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.

Rules for in-*MSet*				
(1)	$r \in f(t_1, \ldots, t_n)$ f is not $\{\!\{ \cdot \mid \cdot \}\!\}$	$\mapsto$	`false`	
(2)	$r \in \{\!\{ t \mid s \}\!\}$	$\mapsto$	$r = t \vee$ $r \in s$	(a) (b)
(3)	$r \in X$ $X \in \mathsf{FV}(r)$	$\mapsto$	`false`	
(4)	$r \in X$ $X \notin \mathsf{FV}(r)$	$\mapsto$	$X = \{\!\{ r \mid N \}\!\}$	

Rules for nin-*MSet*			
(1)	$r \notin f(t_1, \ldots, t_n)$ f is not $\{\!\{ \cdot \mid \cdot \}\!\}$	$\mapsto$	`true`
(2)	$r \notin \{\!\{ t \mid s \}\!\})$	$\mapsto$	$r \neq t \wedge r \notin s$
(3)	$r \notin X$ $X \in \mathsf{FV}(r)$	$\mapsto$	`true`

Figure 4: Rewriting rules for membership and not-membership constraints

in-*MSet*(1) $r \in f(t_1, \ldots, t_n)$, with f different from $\{\!\{ \cdot \mid \cdot \}\!\}$ is equivalent to `false` by axiom (K).

in-*MSet*(2) This is exactly axiom (W).

in-*MSet*(3) Assume that there is a valuation σ 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: $\{\!\{ s_1, \ldots, s_n, r' \mid t \}\!\}$ for some terms $s_1, \ldots, s_n, t$ 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 σ 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: $\{\!\{ s_1, \ldots, s_n, r' \mid t \}\!\}$ for some terms $s_1, \ldots, s_n, t$ 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 $\{\!\{ r', s_1, \ldots, s_n \mid t \}\!\}$ for some terms $s_1, \ldots, s_n, t$. Thus, it is a model of $X = \{\!\{ r \mid N \}\!\}$. The other direction is similar.

nin-*MSet*(1), (2), (3) Same proofs as for the corresponding in-*MSet* rules, using the same axioms.

□

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 5 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 5, we would have that:

$$\begin{array}{rcl} \{\!\{ t_1 \mid s_1 \}\!\} \neq \{\!\{ t_2 \mid s_2 \}\!\} & \leftrightarrow & (t_1 \neq t_2 \vee s_1 \neq s_2) \wedge \\ & & \forall N \, (s_2 \neq \{\!\{ t_2 \mid N \}\!\} \vee s_1 \neq \{\!\{ t_1 \mid N \}\!\}) \end{array}$$

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:

$$\begin{array}{rcl} \{\!\{ t_1 \mid s_1 \}\!\} \neq \{\!\{ t_2 \mid s_2 \}\!\} & \leftrightarrow & \exists X \exists n \, (n \in \mathbb{N} \wedge \\ & & (X \in^n \{\!\{ t_1 \mid s_1 \}\!\} \wedge X \notin^n \{\!\{ t_2 \mid s_2 \}\!\}) \vee \\ & & (X \in^n \{\!\{ t_2 \mid s_2 \}\!\} \wedge X \notin^n \{\!\{ t_1 \mid s_1 \}\!\})) \end{array}$$

Rules for neq-*MSet*			
(1)	$d \neq d$ d is a constant	$\mapsto$	`false`
(2)	$f(s_1,\ldots,s_m) \neq g(t_1,\ldots,t_n)$ f is not g	$\mapsto$	`true`
(3)	$t \neq X$ t is not a variable	$\mapsto$	$X \neq t$
(4)	$X \neq X$ X is a variable	$\mapsto$	`false`
(5)	$f(s_1,\ldots,s_n) \neq f(t_1,\ldots,t_n)$ $n > 0, f$ is not $\mathtt{cons}_{\mathbb{T}}(\cdot,\cdot)$	$\mapsto$	$s_1 \neq t_1 \vee$ (1) $\vdots \quad \vdots$ $s_n \neq t_n$ (n)
(6.1)	$\{\!\{t_1 \mid s_1\}\!\} \neq \{\!\{t_2 \mid s_2\}\!\}$ $\mathsf{tail}(s_1)$ and $\mathsf{tail}(s_2)$ are the same variable	$\mapsto$	$\mathsf{untail}(\{\!\{t_1 \mid s_1\}\!\}) \neq \mathsf{untail}(\{\!\{t_2 \mid s_2\}\!\})$
(6.2)	$\{\!\{t_1 \mid s_1\}\!\} \neq \{\!\{t_2 \mid s_2\}\!\}$ $\mathsf{tail}(s_1)$ and $\mathsf{tail}(s_2)$ are not the same variable	$\mapsto$	$(t_1 \neq t_2 \wedge t_1 \notin s_2) \vee$ (a) $(\{\!\{t_2 \mid s_2\}\!\} = \{\!\{t_1 \mid N\}\!\} \wedge s_1 \neq N)$ (b)
(7)	$X \neq f(t_1,\ldots,t_n)$ $X \in \mathsf{FV}(t_1,\ldots,t_n)$	$\mapsto$	`true`

Figure 5: Rewriting rules for disequality constraints on multisets

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 5 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 5) cannot be used in this context. In fact, the property that* $s_1 \neq N$ *implies* $\{\!\{t_1 \mid s_1\}\!\} \neq \{\!\{t_1 \mid N\}\!\}$*, 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*

$$\boxed{(E_k) \quad x = y \quad \leftrightarrow \quad \forall z\,(z \in x \leftrightarrow z \in y)}$$

This leads to the following rewriting rule that replaces rules (6.1) *and* (6.1) *of Figure 5 in the case of disequality constraints on sets.*

$$(6) \quad \{t_1 \mid s_1\} \neq \{t_2 \mid s_2\} \ \} \ \mapsto \ Z \in \{t_1 \mid s_1\} \wedge Z \notin \{t_2 \mid s_2\} \vee \quad (a)$$
$$Z \in \{t_2 \mid s_2\} \wedge Z \notin \{t_1 \mid s_1\} \quad (b)$$

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

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

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:

$$t_1 \notin \{\!\{ t_2 \,|\, s_2 \}\!\} \wedge \{\!\{ t_1 \,|\, s_1 \}\!\} \neq \{\!\{ t_2 \,|\, s_2 \}\!\} \vee \tag{1}$$

$$t_1 \in \{\!\{ t_2 \,|\, s_2 \}\!\} \wedge \{\!\{ t_1 \,|\, s_1 \}\!\} \neq \{\!\{ t_2 \,|\, s_2 \}\!\} \tag{2}$$

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

$$\mathcal{MSET} \models \forall (t_1 \in \{\!\{ t_2 \,|\, s_2 \}\!\} \leftrightarrow \exists M \, (\{\!\{ t_1 \,|\, M \}\!\} = \{\!\{ t_2 \,|\, s_2 \}\!\})) \tag{3}$$

Thus, (2) is equivalent to

$$\exists M \, (\{\!\{ t_1 \,|\, M \}\!\} = \{\!\{ t_2 \,|\, s_2 \}\!\} \wedge \{\!\{ t_1 \,|\, s_1 \}\!\} \neq \{\!\{ t_2 \,|\, s_2 \}\!\}) \tag{4}$$

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

$$\exists N \, (s_1 \neq N \wedge \{\!\{ t_2 \,|\, s_2 \}\!\} = \{\!\{ t_1 \,|\, N \}\!\}) \tag{5}$$

(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 $M = N$.

□

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 [24]. Basically, a new constraint (*ker*) is introduced and the rewriting rule (6) is endowed with a third non-deterministic case:* $\mathrm{ker}(s_1) \neq \mathrm{ker}(s_2)$*. For further details, see [19].*

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 non-deterministic rewriting procedure $\mathsf{SAT}_{\mathbb{T}}$ shown in Figure 6. $\mathsf{SAT}_{\mathbb{T}}$ 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, i.e., 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 $\mathsf{SAT}_{\mathbb{T}}$ is equi-satisfiable in the standard model with the original constraint C. Therefore, $\mathsf{SAT}_{\mathbb{T}}$ 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. The generated constraint in solved form can be exploited to compute all possible successful valuations for C.

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

Figure 6: The satisfiability procedure, parametric with respect to $\mathbb{T}$

The rest of this section is devoted to prove the crucial result of termination of the procedure $\mathsf{SAT}_{\mathbb{T}}(C)$, to prove its soundness and completeness, and, finally, to give some hints on its complexity.

Theorem 7.1 (Termination) *Let $\mathbb{T}$ be one of the theories List, MSet, CList, Set, and C be a $\mathbb{T}$-constraint. Each non-deterministic execution of $\mathsf{SAT}_{\mathbb{T}}(C)$ 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 [19].

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:

$$\begin{array}{rcll} compl(C) & = & \langle & \alpha(C) = \#\text{ vars in } C, \\ & & & \beta(C) = \{\!\!\{\, size(s) + size(t) : s\ op\ t \in C \,\}\!\!\}, \\ & & & \gamma(C) = \sum_{s\ op\ t \in C} size(s) \qquad\qquad \rangle \end{array}$$

The first and third element of the triple are non-negative integers. The second is a multiset of non-negative integers. Multisets of non-negative integers are well-ordered [16] by the ordering obtained as the transitive closure of the rule:

$$\{\!\!\{\, s_1, \ldots, s_{i-1}, t_1, \ldots, t_n, s_{i+1}, \ldots, s_m \,\}\!\!\} \prec \{\!\!\{\, s_1, \ldots, s_m \,\}\!\!\},$$

for $i \in \{1, \ldots, m\}$, $n \geq 0$, $t_1 < s_i, \ldots, t_n < s_i$. 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 non-failing 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) α does not increase, β decreases.

unify-*MSet*(2) α and β do not increase. γ decreases, since $size(X) = 0$ and $size(t) > 0$.

unify-*MSet*(3) α decreases by 1.

unify-*MSet*(6) α does not increase. β decreases, since an equation of size $1 + \sum_{i=1}^{m} size(s_i) + size(t_i)$ is replaced by m smaller equations of size $size(s_i) + size(t_i)$.

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 α decreases or α does not change and β decreases (see Remark 6.2).

unify-*MSet*(8) After one rule application, we are in case (7) with both the tails of the multisets non-variables. After a finite number of steps, we enter the situation where α is unchanged and β decreases.

in-*MSet*(2) α does not increase. β decreases, since a constraint of size $1 + size(r) + size(s) + size(t)$ is non-deterministically replaced by one of smaller size, i.e. either $size(r)+size(s)$ or $size(r)+size(t)$.

nin-*MSet*(1), (3) α does not increase and β decreases.

nin-*MSet*(2) α does not increase. β decreases, since a constraint of size $1 + size(r) + size(s) + size(t)$ is non-deterministically replaced by two constraints of smaller size $size(r) + size(s)$ and $size(r) + size(t)$.

neq-*MSet*(2), (7) Trivially, α does not increase and β decreases.

neq-*MSet*(3) α and β do not increase. γ decreases, since $size(X) = 0$ and $size(t) > 0$.

neq-*MSet*(5) α does not increase. β decreases, since a constraint of size $1 + \sum_{i=1}^{m} size(s_i) + size(t_i)$ is non-deterministically replaced by one of size $size(s_i) + size(t_i)$.

neq-*MSet*(6.2) A unique application of this rule may not decrease the constraint complexity. However, the rule removes $\{\!\{ t_1 \mid s_1 \}\!\} \neq \{\!\{ t_2 \mid s_2 \}\!\}$ and introduces

$$\{\!\{ t_2 \mid s_2 \}\!\} = \{\!\{ t_1 \mid N \}\!\} \wedge \tag{6}$$

$$s_1 \neq N \tag{7}$$

Consider now the two cases:

1. $\{\!\{ t_2 \mid s_2 \}\!\}$ is $\{\!\{ r_1, \ldots, r_n \}\!\}$
2. $\{\!\{ t_2 \mid s_2 \}\!\}$ is $\{\!\{ r_1, \ldots, r_n \mid A \}\!\}$, 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:

$$t_1 = r_i \wedge N = \{\!\{ r_1, \ldots, r_{i-1}, r_{i+1}, \ldots, r_n \}\!\}$$

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

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

The equation $N = \{\!\{ r_1, \ldots, r_{i-1}, r_{i+1}, \ldots, r_n \}\!\}$ is eliminated by applying the substitution for N. N occurs only in the constraint $s_1 \neq N$, that becomes $s_1 \neq \{\!\{ r_1, \ldots, r_{i-1}, r_{i+1}, \ldots, r_n \}\!\}$. Again, its *size* is strictly smaller than that of the original disequality constraint. Thus, after some further steps, α remains unchanged while β 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 α decreases.

neq-*MSet*(6.1) After one step, we are in the above situation (6.2).

□

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 $C_1, \ldots, C_k$ be the solved form constraints non-deterministically returned by $\mathsf{SAT}_{\mathbb{T}}(C)$, and $\bar{N}_i$ be $\mathsf{FV}(C_i) \setminus \mathsf{FV}(C)$. Then $\mathcal{A}_{\mathbb{T}} \models \forall \left(C \leftrightarrow \bigvee_{i=1}^{k} \exists \bar{N}_i C_i \right)$, 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 non-deterministic branch. At each branch point, the number of non-deterministic choices is finite. Thus, $C_1, \ldots, C_k$ 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*. □

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 $\mathcal{LIST}$, $\mathcal{MSET}$, $\mathcal{CLIST}$, $\mathcal{SET}$.*

Proof. From Theorem 7.2 we know that C is equi-satisfiable with $C_1 \vee \cdots \vee C_k$. If all the C_i are `false`, then C is unsatisfiable in $\mathcal{LIST}$ ($\mathcal{MSET}$, $\mathcal{CLIST}$, $\mathcal{SET}$). Otherwise, it is satisfiable, since solved form constraints are satisfiable (Theorem 5.7). □

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 are studied in [20]: 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 polinomial, 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 [3, 14]. 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 [19]. 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 [18].

Let us now comment on the complexity of our constraint rewriting procedures. The unification algorithm presented in [20] and briefly recalled here can generate terms which grow exponentially. Consider for instance the constraint

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

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 [1, 15], it is possible to avoid explicit substitutions, thus obtaining an implementation of the unification algorithm which works in non-deterministic polynomial time. In our context, at the implementation level, terms can be represented by linked structures. Precisely, a term $f(t_1, \ldots, t_n)$ can be represented by a node labeled by f pointing to the nodes representing $t_1, \ldots, t_n$. 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 $\mathsf{SAT}_{\mathbb{T}}$, 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, i.e., 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 non-deterministic polynomial time, we can obtain a non-deterministic polynomial time implementation for $\mathsf{SAT}_{\mathbb{T}}$.

8 Conclusions

We have extended the results of [20] 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 paper can be found in the Constraint Logic Programming language {log} (`http://prmat.math.unipr.it/~gianfr/SETLOG/setlog_fd.pl`). 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 paper [17].

References

[1] D. Aliffi, A. Dovier, and G. Rossi. From Set to Hyperset Unification. *Journal of Functional and Logic Programming*, 1999(10):1–48. The MIT Press, September 1999.

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

[3] F. Baader and T. Nipkow. *Term Rewriting and All That.* Cambridge University Press, Cambridge, 1998

[4] F. Baader and K. U. Schulz. Combination Techniques and Decision Problems for Disunification. *Theoretical Computer Science*, 142:229–255, 1995.

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

[6] J. Banatre and D. Le Metayer. Programming by Multiset Transformation. *Communications of the ACM*, 36(1):98–111. January 1993.

[7] C. Beeri, S. Naqvi, O. Shmueli, and S. Tsur. Set Constructors in a Logic Database Language. *Journal of Logic Programming 10*, 3 (1991), 181–232.

[8] H. Bückert. Solving Disequations in Equational Theories. In E. L. Lusk and R. A. Overbeek eds., *CADE 1988*, Lecture Notes in Computer Science, Vol. 310, pages 517–526, 1988.

[9] W. Büttner. Unification in the Data Structure Sets. In *Proc. of the Eight International Conference on Automated Deduction* (1986), J. K. Siekmann, Ed., vol. 230, Springer-Verlag, Berlin, pp. 470–488.

[10] D. Cantone, E. G. Omodeo, and A. Policriti. *Set Theory for Computing. From Decision Procedures to Declarative Programming with Sets.* Monographs in Computer Science. Springer-Verlag, Berlin, 2001.

[11] C. C. Chang and H. J. Keisler. *Model Theory.* Studies in Logic. North Holland, 1973.

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

[13] H. Comon. Complete Axiomatixations of Some Quotient Term Algebras. *Theoretical Computer Science*, 118(2):167–191, 1993.

[14] J. Corbin and M. Bidoit. A rehabilitation of Robinson's unification algorithm. In R.Mason ed., *Information Processing 1983*, Elevisier Science Publishers (North Holland), pp. 909–914.

[15] E. Dantsin and A. Voronkov. A Nondeterministic Polynomial-Time Unification Algorithm for Bags, Sets and Trees. In W. Thomas ed., *Foundations of Software Science and Computation Structure*, Lecture Notes in Computer Science, Vol. 1578, pages 180–196, 1999.

[16] N. Dershowitz and Z. Manna. Proving Termination with Multiset Ordering. *Communication of the ACM 22*, 8 (1979), 465–476.

[17] A. Dovier, C. Piazza, and A. Policriti. Comparing expressiveness of set constructor symbols. In H. Kirchner and C. Ringeissen, eds., *FROCOS'00*, LNCS No. 1794, pp. 275–289, 2000.

[18] A. Dovier, C. Piazza, and E. Pontelli. Disunification in ACI1 Theories. *Constraints (International Journal)*, 9(1):35–91, 2004.

[19] A. Dovier, C. Piazza, and G. Rossi. A uniform approach to constraint-solving for lists, multisets, compact lists, and sets. CoRR cs.PL/0309045.

[20] A. Dovier, A. Policriti, and G. Rossi. A uniform axiomatic view of lists, multisets, and sets, and the relevant unification algorithms. *Fundamenta Informaticae*, 36(2/3):201–234, 1998.

[21] A. Dovier, E. Pontelli, and G. Rossi. Set unification. Tech. Rep. cs.LO/0110023, The Computing Research Repository (CoRR), October 2001. http://arxiv.org/abs/cs.LO/0110023. To appear in *Theory and Practice of Logic Programming*.

[22] A. Dovier, E. G. Omodeo, E. Pontelli, and G. Rossi. {log}: A Language for Programming in Logic with Finite Sets. *Journal of Logic Programming*, 28(1):1–44, 1996.

[23] A. Dovier, C. Piazza, E. Pontelli, and G. Rossi. Sets and constraint logic programming. *ACM Transaction on Programming Language and Systems*, 22(5):861–931, 2000.

[24] A. Dovier and G. Rossi. Embedding Extensional Finite Sets in CLP. In D. Miller, editor, *Proc. of International Logic Programming Symposium, ILPS'93*. The MIT Press, Cambridge, Mass., October 1993, pages 540–556.

[25] H. B. Enderton. *A mathematical introduction to logic*. Academic Press, 1973. 2^{nd} printing.

[26] C. Gervet. Interval Propagation to Reason about Sets: Definition and Implementation of a Practical Language. *Constraints*, 1:191–246, 1997.

[27] S. Grumbach and T. Milo. Towards tractable algebras for bags. *Journal of Computer and System Sciences*, 52(3):570–588, 1996.

[28] P. M. Hill and J. W. Lloyd. *The Gödel Programming Language*. The MIT Press, Cambridge, Mass., 1994.

[29] J. Jaffar and M. J. Maher. Constraint Logic Programming: A Survey. *Journal of Logic Programming*, 19–20:503–581, 1994.

[30] M. Livesey and J. Siekmann. Unification of Sets and Multisets. Technical report, Institut fur Informatik I, Universitat Karlsruhe, 1976.

[31] A. Mal'cev. Axiomatizable Classes of Locally Free Algebras of Various Types. In *The Metamathematics of Algebraic Systems*, Collected Papers, chapter 23. North Holland, 1971.

[32] G. Pāun. Computing with Membranes. Journal of Computer and System Science, 61, 2000.

[33] B. Potter, J. Sinclair, and D. Till. *An Introduction to Formal Specification and Z*, Second Edition. Prentice Hall, 1996.

[34] J. T. Schwartz, R. B. K. Dewar, E. Dubinsky, and E. Schonberg. *Programming with sets, an introduction to SETL*. Springer-Verlag, Berlin, 1986.

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

[36] P. J. Stuckey. Negation and Constraint Logic Programming. *Information and Computation 1*, 12–33.

[37] A. Tzouvaras. The Linear Logic of Multisets. *Logic Journal of the IGPL*, Vol. 6, No. 6,. pp. 901–916, 1998.

A Appendix: auxiliary proofs

We recall some technical definitions. Given two Σ-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 Σ-structures $\mathcal{A}$ and $\mathcal{B}$, a function $h : A \longrightarrow B$ is said to be an *homomorphism* from $\mathcal{A}$ to $\mathcal{B}$ if: (i) $\forall f \in \mathcal{F}, a_1, \ldots, a_n \in A \, (h(f^{\mathcal{A}}(a_1, \ldots, a_n)) = f^{\mathcal{B}}(h(a_1), \ldots, h(a_n)))$ and (ii) $\forall p \in \Pi, a_1, \ldots, a_m \in A \, (p^{\mathcal{A}}(a_1, \ldots, a_m) \rightarrow p^{\mathcal{B}}(h(a_1), \ldots, h(a_m)))$. 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 Σ-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 ([11]) *Let $\mathcal{A}$ and $\mathcal{B}$ be two Σ-structures and let h be an embedding of $\mathcal{A}$ in $\mathcal{B}$. If φ is an open formula of $\mathcal{L} = \langle \Sigma, \mathcal{V} \rangle$, then for each valuation σ 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 MSet.*

Proof. For each axioms/axiom schemas (A) of the theory *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 $f(t_1, \ldots, t_n)$ and $f(s_1, \ldots, s_n)$ belong to the same class in $\mathcal{MSET}$, only if for all $i = 1, \ldots, n$ 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 [35].

(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).

□

Lemma A.3 *If $\mathcal{M}$ is a model of MSet, then the function $h : T(\mathcal{F}_{MSet})/\equiv_{E_{MSet}} \longrightarrow M$, defined as $h(\textcircled{t}) = t^{\mathcal{M}}$, is an embedding of $\mathcal{MSET}$ in $\mathcal{M}$.*

Proof. We will prove the following facts:

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

These facts imply the thesis.

1. If t_1 and t_2 are two terms such that $\textcircled{t_1} = \textcircled{t_2}$, 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 $\mathcal{M}$, i.e., $t_1^{\mathcal{M}} = t_2^{\mathcal{M}}$.
2. We need to prove that:
   (a) for all $f \in \mathcal{F}_{MSet}$ and for all terms $t_1, \ldots, t_n \in T(\mathcal{F}_{MSet})$ it holds that
   $$h(f^{\mathcal{MSET}}(\textcircled{t_1}, \ldots, \textcircled{t_n})) = f^{\mathcal{M}}(h(t_1), \ldots, h(t_n))$$
   Now,
   $$\begin{array}{rcll} h(f^{\mathcal{MSET}}(\textcircled{t_1}, \ldots, \textcircled{t_n})) & = & h(f(t_1, \ldots, t_n)) & \text{By fact (1) above} \\ & = & (f(t_1, \ldots, t_n))^{\mathcal{M}} & \text{By def. of } h \\ & = & f^{\mathcal{M}}(t_1^{\mathcal{M}}, \ldots, t_n^{\mathcal{M}}) & \text{By def. of structure} \\ & = & f^{\mathcal{M}}(h(t_1), \ldots, h(t_n)) & \text{By def. of } h \end{array}$$
   (b) For all terms t and s, if $\textcircled{t} \in^{\mathcal{MSET}} \textcircled{s}$, then $h(\textcircled{t}) \in^{\mathcal{M}} h(\textcircled{s})$. From $\textcircled{t} \in^{\mathcal{MSET}} \textcircled{s}$, using fact 1. above, we have that there is a term s' in $\textcircled{s}$ of the form $\{\!\{ t \,|\, r \}\!\}$ and that $h(\textcircled{s}) = s'^{\mathcal{M}}$. Hence, we have that $h(\textcircled{s}) = \{\!\{ t^{\mathcal{M}} \,|\, r^{\mathcal{M}} \}\!\}^{\mathcal{M}}$; (W) ensures that $h(\textcircled{t}) = t^{\mathcal{M}}$ belongs to it.
3. We prove, by structural induction on t_1, that if $h(\textcircled{t_1}) = h(\textcircled{t_2})$, then $\textcircled{t_1} = \textcircled{t_2}$.
   Basis. Let t_1 be a constant c. Since $\mathcal{M}$ is a model of axiom schema (F_2), it can not be that $t_2 = f(s_1, \ldots, s_n)$, with f different from c. Hence, it must be that $t_2 = c$.
   Step. Let t_1 be $f(s_1, \ldots, s_n)$, with $f \not\equiv \{\!\{ \cdot \,|\, \cdot \}\!\}$. It cannot be $t_2 \equiv g(r_1, \ldots, r_m)$, with $g \not\equiv f$, since $\mathcal{M}$ is a model of (F_2). So, it must be $t_2 \equiv f(r_1, \ldots, r_n)$, and, by (F_1), $s_i^{\mathcal{M}} = r_i^{\mathcal{M}}$, for all $i \leq n$. Using the inductive hypothesis we have $\textcircled{t_1} = \textcircled{t_2}$.
   Let t_1 be $\{\!\{ s_1, \ldots, s_n \,|\, r \}\!\}$, with r not of the form $\{\!\{ r_1 \,|\, r_2 \}\!\}$. Since it cannot be that t_2 is $f(v_1, \ldots, v_n)$ (from the previous case applied to t_2), then it must be t_2 is $\{\!\{ u_1, \ldots, u_m \,|\, v \}\!\}$, for some v not of the form $\{\!\{ v_1 \,|\, v_2 \}\!\}$. Let us assume, by contradiction, that $\textcircled{t_1} \neq \textcircled{t_2}$, and $t_1^{\mathcal{M}} = t_2^{\mathcal{M}}$, while the thesis holds for all terms of lower complexity. From $t_1^{\mathcal{M}} = t_2^{\mathcal{M}}$ we obtain that the two terms have in $\mathcal{M}$ the same elements. Since $\mathcal{M}$ is a model of (W), the elements of $t_1^{\mathcal{M}}$ are exactly $s_1^{\mathcal{M}}, \ldots, s_n^{\mathcal{M}}$ and the elements of $t_2^{\mathcal{M}}$ are exactly $u_1^{\mathcal{M}}, \ldots, u_m^{\mathcal{M}}$. So, by inductive hypothesis, there is a bijection $b : \{1, \ldots, n\} \longrightarrow \{1, \ldots, m\}$ such that $\textcircled{s_i} = \textcircled{u_{b(i)}}$. This means that $m = n$ and that there is a term t_2' in $\textcircled{t_2}$ of the form $\{\!\{ s_1, \ldots, s_m \,|\, v \}\!\}$. Applying n times (E_k^m), in all possible ways, we obtain that $r^{\mathcal{M}} = v^{\mathcal{M}}$, hence by inductive hypothesis $\textcircled{r} = \textcircled{v}$. From this fact, we conclude that $\textcircled{t_2} = \textcircled{t_2'} = \textcircled{\{\!\{ s_1, \ldots, s_n \,|\, r \}\!\}} = \textcircled{t_1}$, which is in contradiction with our assumption.
4. If $h(\textcircled{t}) \in^{\mathcal{M}} h(\textcircled{s})$, then $t^{\mathcal{M}} \in^{\mathcal{M}} s^{\mathcal{M}}$ and hence (K) implies that s must be a term of the form $\{\!\{ t_1 \,|\, t_2 \}\!\}$. By induction on s using (W), we can prove that in particular s must be a term of the form $\{\!\{ t_1, \ldots, t_i, \ldots, t_n \,|\, r \}\!\}$, with $t_1^{\mathcal{M}} = t^{\mathcal{M}} = h(\textcircled{t})$. We have already proved that h is injective, hence it must be $t_1 \in \textcircled{t}$, and from this we obtain $\textcircled{t} \in^{\mathcal{MSET}} \textcircled{s}$.

□

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 $x_1, \ldots, x_h$ 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 $x_1, \ldots, x_n$, then we concentrate on the variable x_1. Each disequality of $A_{\neq}$ can be rewritten in the form $a_{i,1}x_1 \neq p_i(x_2, \ldots, x_n)$, where $p_i(x_2, \ldots, x_n) = a_{i,2}x_2 + \ldots + a_{i,n}x_n + a_{i,n+1}$ is a linear expression with integer coefficients over the variables $x_2, \ldots, x_n$. 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 $a_{i,1}x_1 \neq a_{i,2}x_2 + \ldots + a_{i,n}x_n + a_{i,n+1}$ with at least one of the $a_{i,2}, \ldots, a_{i,n}$ 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. □

Lemma A.5 *In* $KWE_k^m F_1^m F_2 F^3 F_3^m$ *it holds that:*

(1) $\{\!|\,\texttt{nil}\,|\!\}^x = \{\!|\,\texttt{nil}\,|\!\}^y$ *if and only if* $x = y$*;*

(2) $\{\!|\,\texttt{nil}\,|\!\}^x \in \{\!|\,\texttt{nil}\,|\!\}^y$ *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 $KWE_k^m F_1^m F_2 F^3 F_3^m$ we can prove $\{\!|\,\texttt{nil}\,|\!\}^x = \{\!|\,\texttt{nil}\,|\!\}^y$. 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 $\{\!|\,\texttt{nil}\,|\!\}^x = \{\!|\,\texttt{nil}\,|\!\}^y$ we get $\{\!|\,\texttt{nil}\,|\!\}^{x-1} = \{\!|\,\texttt{nil}\,|\!\}^{y-1}$. 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. □