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

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

Agostino Dovier* Carla Piazza* Gianfranco Rossi†

July 22, 2013

Abstract

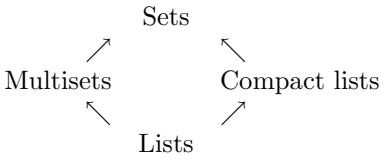
Lists, multisets, and sets are well-known data structures whose usefulness is widely recognized in various areas of Computer Science. These data structures have been analyzed from an axiomatic point of view with a parametric approach in [11] where the relevant unification algorithms have been developed. In this paper we extend these results considering more general constraints including not only equality but also membership constraints as well as their negative counterparts.

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

1 Introduction

Programming and specification languages usually allow the user to represent various forms of aggregates of data objects, characterized by the way elements are organized and accessed. In this paper we consider four different kinds of aggregates: lists, multisets, compact lists, and sets. The basic difference between them lies in the order and/or repetitions of their data objects.

Importance of these forms of aggregates is widely recognized in various areas of Computer Science. Lists are the classical example used 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 [21]) and in high-level declarative programming languages [4, 12, 16, 18]; but also imperative programming languages may take advantage from the set data abstraction (e.g., SETL [22]). Multisets, often called *bags* in the literature, emerge as the most natural data structure in several interesting applications [3, 17, 25]. A compact list is a list in which contiguous occurrences of the same element are immaterial; some possible application examples are suggested in [11].



The lattice of the four aggregates

Lists, multisets, compact lists, and sets have been analyzed from an axiomatic point of view and studied in the context of (Constraint) Logic Programming (CLP) languages [11]—see figure on the left for a lattice induced by their axiomatizations. In this context, these aggregates are conveniently represented as terms, using different constructors.

The theories studied deal with aggregate constructor symbols as well as with an arbitrary number of free constant and function symbols. [11] focuses on *equality* between terms in each of the four theories. This amounts to solve the unification problems in the equational theories describing the properties of the four considered aggregates. Unification algorithms for all of

*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. Via M. D'Azeglio 85/A, 43100 Parma (Italy). gianfranco.rossi@unipr.it

them are provided in [11]; NP-unification algorithms for sets and multisets are also presented in [1, 8]. In Section 3 and 5.1 we recall the main results of [11].

In this paper we extend the results of [11] to the case of more general *constraints*. The constraints we consider are conjunctions of literals based on both equality and membership predicate symbols. For the case of sets, the problem is studied in [13, 14]. In Section 4 we define the notion of constraints and we identify the privileged models for the axiomatic theories used to describe the considered aggregates. We show that satisfiability of constraints in those models is equivalent to satisfiability in any model. We then define the notion of solved form for constraints, and we prove that solved form constraints are satisfiable over the proposed privileged models. In Section 5 we describe, for each kind of aggregate, the constraint rewriting procedures used to eliminate all atomic constraints not in solved form. We use these procedures in Section 6 to solve the general satisfiability problem for the considered constraints. Some conclusions are drawn in Section 7. Throughout the paper the word *aggregate* is used for denoting generically one of the four considered aggregates, namely lists, multisets, compact lists, and sets.

2 Preliminary Notions

Basic knowledge of first-order logic (e.g., [5, 15]) is assumed; in this section we recall some notions and we fix some notations that we will use 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. A *(first-order) theory* $\mathcal{T}$ on a language $\mathcal{L}$ is a set of closed first-order formulas of $\mathcal{L}$ such that each closed formula of $\mathcal{L}$ which can be deduced from $\mathcal{T}$ is in $\mathcal{T}$. A *(first-order) set of axioms* Θ on $\mathcal{L}$ is a set of closed first-order formulas of $\mathcal{L}$. A set of axioms Θ is said to be an *axiomatization* of $\mathcal{T}$ if $\mathcal{T}$ is the smallest theory such that $\Theta \subseteq \mathcal{T}$. Sometimes we use the term *theory* also to refer to an axiomatization of the theory. When $\Theta = \{\varphi_1, \ldots, \varphi_n\}$, and $A_1, \ldots, A_n$ are the names of the formulas $\varphi_1, \ldots, \varphi_n$, we refer to that theory simply as: $A_1 \cdots A_n$.

Capital letters X, Y, Z, etc. are used to represent variables, f, g, etc. to represent constant and function symbols, and p, q, etc. to represent predicate symbols. We also use $\bar{X}$ to denote a (possibly empty) sequence of variables. $T(\mathcal{F}, \mathcal{V})$ ($T(\mathcal{F})$) denotes the set of first-order terms (resp., ground terms) built from $\mathcal{F}$ and $\mathcal{V}$ (resp., $\mathcal{F}$). The function $size : T(\mathcal{F}, \mathcal{V}) \longrightarrow \mathbb{N}$ returns the number of occurrences of constant and function symbols in a term. Given a term t, with $FV(t)$ we denote the set of all variables which occur in the term t. Given a sequence of terms $t_1, \ldots, t_n$, $FV(t_1, \ldots, t_n)$ is the set $\bigcup_{i=1}^{n} FV(t_i)$. When the context is clear, we use $\bar{t}$ to denote a sequence $t_1, \ldots, t_n$ of terms. If φ is a first-order formula, $FV(\varphi)$ denotes the set of free variables in φ. $\exists\varphi$ ($\forall\varphi$) is used to denote the existential (universal) closure of the formula φ, namely $\exists X_1 \cdots \exists X_n \, \varphi$ ($\forall X_1 \cdots \forall X_n \, \varphi$), where $\{X_1, \ldots, X_n\} = FV(\varphi)$. An *equational axiom* is a formula of the form $\forall X_1 \cdots \forall X_n (\ell = r)$ where $FV(\ell = r) = \{X_1, \ldots, X_n\}$. An *equational theory* is an axiomatization whose axioms are equational axioms.

Given a first-order theory $\mathcal{L} = \langle \Sigma, \mathcal{V} \rangle$, a Σ-*structure* is a pair $\mathcal{A} = \langle A, I \rangle$ where A is a non-empty set (the domain) and I is the interpretation function of all 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. σ and I determine uniquely a function σ^I from the set of first-order terms over $\mathcal{L}$ to A and a function from the set of formulas over $\mathcal{L}$ to the set $\{\mathtt{false}, \mathtt{true}\}$. When the Σ-structure is fixed, σ^I depends only by σ. Thus, with abuse of notation, σ^I is simply written as σ. Given a Σ-structure $\mathcal{A}$, a valuation σ is said a *successful valuation* of φ if $\sigma(\varphi) = \mathtt{true}$. This fact is also denoted by: $\mathcal{A} \models \sigma^I(\varphi)$. A formula φ is *satisfiable* in $\mathcal{A}$ if there is a valuation $\sigma : FV(\varphi) \longrightarrow A$ such that $\mathcal{A} \models \sigma(\varphi)$. In this case we say that $\mathcal{A} \models \exists\varphi$. We say that $\mathcal{A} \models \varphi$ if for every valuation σ from $FV(\varphi) \longrightarrow A$ it holds that $\mathcal{A} \models \sigma(\varphi)$. A formula φ is *satisfiable* in $\mathcal{A}$ if there is a valuation $\sigma : FV(\varphi) \longrightarrow A$ such that $\mathcal{A} \models \sigma(\varphi)$. In this case we say that $\mathcal{A} \models \exists\varphi$. We remind that a formula is satisfiable in a Σ-structure $\mathcal{A}$ if and only if its existential closure is satisfiable in $\mathcal{A}$. 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 $\mathcal{T}$.

3 The Theories

For each aggregate considered, we assume that Π is $\{=, \in\}$ and $\mathcal{F}$ contains the constant symbol `nil` and exactly one among the 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 signature can contain an arbitrary number of other constant and function symbols. The four function symbols above are referred as the *aggregate constructors*. The empty list, multiset, compact list, and set are all denoted by the constant symbol `nil`. We use simple syntactic notations for terms built using these symbols. In particular, the list $[\,s_1 \mid [\,s_2 \mid \cdots [\,s_n \mid t\,]\cdots]]$ will be denoted by $[s_1, \ldots, s_n \mid t]$ or simply by $[s_1, \ldots, s_n]$ when t is `nil`. The same conventions will be exploited also for the other aggregates.

3.1 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* for lists is shown in the figure 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 schemata $(F_1), (F_2)$, and (F_3) (called freeness axioms, or Clark's equality axioms—see [6]) have been originally introduced by Mal'cev in [20]. Observe that (F_1) holds for $[\,\cdot \mid \cdot\,]$ as a particular case. (F_3) states that there is no term which is also a subterm of itself. Note that (K) implies that $\forall x\, (x \notin \texttt{nil})$.

3.2 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 simply 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)	$\forall xyz\ \ \{\!\{\,x, y \mid z\,\}\!\} = \{\!\{\,y, x \mid z\,\}\!\}$	*(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 $\{\!\{\,\cdot \mid \cdot\,\}\!\}$. It is replaced by axiom schemata (F_1^m):

(F_1^m)	$\forall x_1 \cdots x_n y_1 \cdots y_n$	$\left(\begin{array}{c} f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n) \\ \rightarrow x_1 = y_1 \wedge \cdots \wedge x_n = y_n \end{array}\right)$
	for any $f \in \mathcal{F}_{MSet}$, f distinct from $\{\!\{\,\cdot \mid \cdot\,\}\!\}$	

In the theory $KWE_p^m F_1^m F_2 F_3$, however, we lack in 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 [11] to force this property is the following:

$$(E_k^m) \qquad \forall y_1 y_2 v_1 v_2 \left(\begin{array}{l} \{\!\{ y_1 \,|\, v_1 \}\!\} = \{\!\{ y_2 \,|\, v_2 \}\!\} \;\leftrightarrow \\ \qquad (y_1 = y_2 \wedge v_1 = v_2) \vee \\ \qquad \exists z \, (v_1 = \{\!\{ y_2 \,|\, z \}\!\} \wedge v_2 = \{\!\{ y_1 \,|\, z \}\!\}) \end{array} \right)$$

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

$$(F_3^m) \quad \forall x_1 \cdots x_m y_1 \cdots y_n x \left(\begin{array}{l} \{\!\{ x_1, \ldots, x_m \,|\, x \}\!\} = \{\!\{ y_1, \ldots, y_n \,|\, x \}\!\} \\ \rightarrow \{\!\{ x_1, \ldots, x_m \}\!\} = \{\!\{ y_1, \ldots, y_n \}\!\} \end{array} \right)$$

Axiom schema (F_3^m) 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 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 the remaining part of the theory.

3.3 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 $\Pi = \{=, \in\}$. Similarly to multisets, 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

$$(E_a^c) \qquad \forall xy \;\; [\![x, x \,|\, y]\!] = [\![x \,|\, y]\!] \quad (\textit{absorption})$$

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:

$$(E_k^c) \qquad \forall y_1 y_2 v_1 v_2 \left(\begin{array}{l} [\![y_1 \,|\, v_1]\!] = [\![y_2 \,|\, v_2]\!] \;\leftrightarrow \\ \qquad (y_1 = y_2 \wedge v_1 = v_2) \vee \\ \qquad (y_1 = y_2 \wedge v_1 = [\![y_2 \,|\, v_2]\!]) \vee \\ \qquad (y_1 = y_2 \wedge [\![y_1 \,|\, v_1]\!] = v_2) \end{array} \right)$$

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

$$(F_1^c) \qquad \forall x_1 \cdots x_n y_1 \cdots y_n \left(\begin{array}{c} f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n) \\ \rightarrow x_1 = y_1 \wedge \cdots \wedge x_n = y_n \end{array} \right)$$
$$\textit{for any } f \in \mathcal{F}_{CList}, \; f \textit{ distinct from } [\![\cdot \,|\, \cdot]\!]$$

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 Σ-structures with the domain built based on Herbrand Universe, where each term is modeled by a finite tree. As opposed to lists and multisets, an equation such as $X = [\![\texttt{nil} \,|\, X]\!]$ admits a solution in these structures. Precisely, a solution that binds X to the term $[\![\texttt{nil} \,|\, t]\!]$, where t is any term. Therefore, as explained in [11], axiom schema (F_3) should be weakened and, thus, replaced by:

$$(F_3^c) \qquad \forall x \quad (x \neq t[x])$$
$$\textit{unless: } t \textit{ is of the form } [\![t_1, \ldots, t_n \,|\, x]\!], \textit{ with } n > 0,$$
$$x \notin FV(t_1, \ldots, t_n), \textit{ and } t_1 = \cdots = t_n$$

Name	empty	with	Equality		Herbrand	Acycl.	Perm.	Abs.	Equational Name
List	(K)	(W)	(F_1)		(F_2)	(F_3)			E_{List}
MSet	(K)	(W)	(E_k^m)	(F_1^m)	(F_2)	(F_3)	(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

3.4 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 `nil` are in $\mathcal{F}_{Set}$, and $\Pi = \{=, \in\}$. The last theory we consider is the simple theory of sets *Set*. Sets have both the *permutativity* and the *absorption properties* which, in the case of $\{\cdot \mid \cdot\}$, can be rewritten as follows:

$$\begin{array}{lrcl} (E_p^s) & \forall xyz\ \{x, y \mid z\} & = & \{y, x \mid z\} \\ (E_a^s) & \forall xy\ \{x, x \mid y\} & = & \{x \mid y\} \end{array}$$

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):

$$(E_k^s) \qquad \forall y_1 y_2 v_1 v_2 \left(\begin{array}{r} \{y_1 \mid v_1\} = \{y_2 \mid v_2\} \ \leftrightarrow \\ (y_1 = y_2 \wedge v_1 = v_2) \vee \\ (y_1 = y_2 \wedge v_1 = \{y_2 \mid v_2\}) \vee \\ (y_1 = y_2 \wedge \{y_1 \mid v_1\} = v_2) \vee \\ \exists k\, (v_1 = \{y_2 \mid k\} \wedge v_2 = \{y_1 \mid k\}) \end{array} \right)$$

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 [11] it is also proved that they are equivalent when domains are made by terms. The theory *Set* also contains axioms (K), (W) with $[\cdot \mid \cdot]$ replaced by $\{\cdot \mid \cdot\}$, and axiom schemata (F_2) Axiom schema (F_1) is replaced by:

$$(F_1^s) \qquad \forall x_1 \cdots x_n y_1 \cdots y_n \left(\begin{array}{c} f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n) \\ \rightarrow x_1 = y_1 \wedge \cdots \wedge x_n = y_n \end{array} \right)$$
$$\textit{for any } f \in \mathcal{F}_{Set},\ f \textit{ distinct from } \{\cdot \mid \cdot\}$$

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

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

3.5 Equational theories

As we have seen in this section, each aggregate constructor is precisely characterized by zero, one or 2 equational axioms. We define the four corresponding *equational theories* as follows:

E_{List}	the empty theory for *List*,
E_{MSet}	consisting of the *Permutativity* axiom (E_p^m) for *MSet*,
E_{CList}	consisting of the *Absorption* axiom (E_a^c) for *CList*,
E_{Set}	consisting of both the *Permutativity* (E_p^s) and *Absorption* (E_a^s) axioms for *Set*.

Relationships between these equational theories, Σ-structures, and the proposed first-order theories for aggregates are explained in the next section. Figure 1 summarizes the axiomatizations of the four theories.

4 Constraints, Privileged Models, and Solved Form

In this section we introduce the privileged models for the four theories introduced in the previous section. These models are used to testing satisfiability of the particular kind of formulas we are concerned with, namely, *constraints*. We then show that the models and the theories defined in the previous section *correspond* on the class of constraints considered. Moreover, we give a general notion of *solved form* for constraints, and we prove that a solved form constraint is satisfiable in the corresponding privileged model.

Definition 4.1 (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* (resp., *disequality*) *constraints* are conjunctions of atomic formulas of the form $s = t$ (resp., $s \neq t$). *Membership* (resp., *not-membership*) constraints are conjunctions of membership atoms (resp., membership negative literals), i.e. formulas of the kind $s \in t$ (resp., $s \notin t$).

4.1 Privileged Models

As discussed in Section 3.5, each aggregate constructor is precisely characterized by an equational theory, that we have named E_{List}, E_{MSet}, E_{CList}, and E_{Set}. Using the appropriate equational theory we can define a privileged model for the first-order theory *List*, *MSet*, *CList*, and *Set* for each aggregate. Each model is obtained as a partition of the Herbrand Universe.

Definition 4.2 *Let $\mathbb{T}$ be List (resp., MSet, CList, or Set). A* privileged Σ-structure for $\mathbb{T}$ *is defined as follows.*

1. *The* domain *of the Σ-structure is the quotient $T(\mathcal{F}_{\mathbb{T}})/\equiv_{\mathbb{T}}$ of the Herbrand Universe $T(\mathcal{F}_{\mathbb{T}})$ over the smallest congruence relation $\equiv_{\mathbb{T}}$ induced by the equational theory $E_{\mathbb{T}}$ on $T(\mathcal{F}_{\mathbb{T}})$.*
2. *The interpretation of a term t is its equivalence class w.r.t. $\equiv_{\mathbb{T}}$, denoted by $\textcircled{t}$.*
3. *$=$ is interpreted as the identity on the domain $T(\mathcal{F}_{\mathbb{T}})/\equiv_{\mathbb{T}}$.*
4. *The interpretation of membership is: $\textcircled{t} \in \textcircled{s}$ is* `true` *if and only if there is a term in $\textcircled{s}$ of the form $[t_1, \ldots, t_n, t \,|\, r]$ (resp., $\{\!|\, t_1, \ldots, t_n, t \,|\, r \,|\!\}$, $[\![\, t_1, \ldots, t_n, t \,|\, r \,]\!]$, or $\{t_1, \ldots, t_n, t \,|\, r\}$) for some terms $t_1, \ldots, t_n, r$.*

It is easy to prove that the above defined Σ-structures are in fact models of the corresponding theories. In Lemma A.2 we prove this property for multisets. ¿From now on, we will call the privileged Σ-structures above defined *privileged models* for *List*, *MSet*, *CList*, and *Set*. We refer to them as $\mathcal{LIST}$, $\mathcal{MSET}$, $\mathcal{CLIST}$, and $\mathcal{SET}$, respectively.

Remark 4.3 *When $\textcircled{s}$ is the class of a multiset (resp., a set), since the permutativity property holds, the requirement for $\textcircled{t} \in \textcircled{s}$ to be* `true` *can be simplified to: $\{\!|\, t \,|\, r \,|\!\}$ (resp., $\{t \,|\, r\}$) is in $\textcircled{s}$.*

The following notion from [19] is crucial for characterizing the above privileged models.

Definition 4.4 *Given a first-order language $\mathcal{L} = \langle \Sigma, \mathcal{V} \rangle$, a set of first-order formulas $\mathcal{C}$ on $\mathcal{L}$, a theory $\mathcal{T}$ on $\mathcal{L}$, and a Σ-structure $\mathcal{A}$, $\mathcal{T}$ and $\mathcal{A}$* correspond *on the set $\mathcal{C}$ 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 the correspondence property for our theories and the privileged models, when the class $\mathcal{C}$ is the class of constraints defined in Definition 4.1. We show below the proof of this result in the case of the model $\mathcal{MSET}$ and the theory *MSet*. The other cases are similar. In the proof we use some basic results which can be found in the Appendix A (Lemmas A.1–A.3).

Theorem 4.5 *The model $\mathcal{MSET}$ (resp., $\mathcal{LIST}$, $\mathcal{CLIST}$, $\mathcal{SET}$) and the theory MSet (resp., List, CList, and Set) correspond on the class of MSet- (resp., List-, CList-, and Set-)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 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 $MSet \models \exists C$. □

4.2 Solved Form

Solved form constraints play a fundamental rôle in establishing satisfiability of constraints in the corresponding privileged model. The solved form is obtained by defining first a weaker form, called the pre-solved form, and then by adding to this form two further conditions.

Definition 4.6 *A constraint $C = c_1 \wedge \cdots \wedge c_n$ is in* pre-solved form *if for $i \in \{1, \ldots, n\}$, c_i is in* pre-solved form *in C, i.e. in one of the following forms:*

- *$X = t$ and X does not occur elsewhere in C*
- *$t \in X$ and X does not occur in t*
- *$X \neq t$ and X does not occur in t*
- *$t \notin X$ and X does not occur in t.*

A constraint in pre-solved form is not guaranteed to be satisfiable in the corresponding privileged model. For example, the constraint $X \in Y \wedge Y \in X$ is in pre-solved form but it is unsatisfiable in each of the privileged models $\mathcal{LIST}, \mathcal{MSET}, \mathcal{CLIST}$, and $\mathcal{SET}$. The first condition we introduce below takes care of this situation.

Definition 4.7 (Acyclicity Condition) *Let C be a pre-solved form constraint and $C^{\in}$ be the part of C containing only membership constraints. Let $\mathcal{G}_C^{\in}$ be the directed graph obtained as follows:*

Nodes. *Associate a distinct node to each variable X in $C^{\in}$.*

Edges. *If $t \in X$ is in $C^{\in}$, $\nu_1, \ldots, \nu_n$ are the nodes associated with the variables in t, and μ is the node associated with the variable X, then add the edges $\langle \nu_1, \mu \rangle, \ldots, \langle \nu_n, \mu \rangle$.*

We say that a pre-solved form constraint C is acyclic *if $\mathcal{G}_C^{\in}$ is acyclic.*

The acyclicity condition is not sufficient for satisfiability. Consider the constraint $\{A, B\} \in X \wedge \{B, A\} \notin X$. It is in pre-solved form and acyclic but unsatisfiable in all the considered privileged models. Conversely, the constraint $\{A\} \in X \wedge \{a\} \notin X$ is satisfiable in $\mathcal{SET}$ (e.g., $A = b, X = \{\{b\}\}$). We observe that whenever there are two constraints $t \in X$ and $t' \notin X$ in C such that t and t' are equivalent terms in the equational theory $E_{\mathbb{T}}$, the constraint C is unsatisfiable.

This analysis, however, does not cover all the possible cases in which an acyclic constraint in pre-solved form is unsatisfiable, as it ensues from the following example:

$$a \in X \wedge X \in Y \wedge \{a \,|\, X\} \notin Y.$$

Observe that there are no pairs of terms t, t' of the form singled out above. Nevertheless, since the satisfiability of $a \in X$ is equivalent in *Set* to that of $X = \{a \,|\, N\}$ (N is a new variable), we have that the constraint is equi-satisfiable to:

$$X = \{a \,|\, N\} \wedge \{a \,|\, N\} \in Y \wedge \{a, a \,|\, N\} \notin Y.$$

Now, $\{a \,|\, N\}$ and $\{a, a \,|\, N\}$ are equivalent terms in E_{Set}, and thus the constraint is unsatisfiable.

To formally define the second condition for solved form constraints, taking into account all the possible cases informally described above, we introduce the following definitions.

Definition 4.8 *Let $\theta \equiv [X_1/t_1, \ldots, X_n/t_n]$ be a substitution and $m \in \mathbb{N}$. We recursively define the substitution θ^m as:*

$$\begin{cases} \theta^1 &= \theta \\ \theta^{m+1} &= [X_1/\theta^m(t_1), \ldots, X_n/\theta^m(t_n)] \quad m > 0 \end{cases}$$

If there exists $m > 0$ such that $\theta^{m+1} \equiv \theta^m$ we say that θ is stabilizing. *Given a stabilizing substitution θ, the* closure θ^* *of θ is the substitution θ^m such that $\forall k > m$ we have that $\theta^k \equiv \theta^m$.*

Definition 4.9 *Let C be a constraint in pre-solved form over the language $\mathcal{L}_{List}$ ($\mathcal{L}_{MSet}, \mathcal{L}_{CList}, \mathcal{L}_{Set}$) and let $t_1^1 \in X_1, \ldots, t_1^{k_1} \in X_1, \ldots, t_q^1 \in X_q, \ldots, t_q^{k_q} \in X_q$ be all membership atoms of C. We define the* member substitution σ_C *as follows:*

$$\sigma_C \equiv [X_1/[F_1, t_1^1, \ldots, t_1^{k_1} \mid M_1], \ldots, X_q/[F_q, t_q^1, \ldots, t_q^{k_q} \mid M_q]]$$

(respectively, $\sigma_C \equiv [X_1/\{\!\!\{ F_1, t_1^1, \ldots, t_1^{k_1} \mid M_1 \}\!\!\}, \ldots]$, $\sigma_C \equiv [X_1/[\![F_1, t_1^1, \ldots, t_1^{k_1} \mid M_1]\!], \ldots]$, $\sigma_C \equiv [X_1/\{F_1, t_1^1, \ldots, t_1^{k_1} \mid M_1\}, \ldots]$) where F_i and M_i are new variables not occurring in C.

The member substitution σ_C forces all the terms t_i^j's to be member of the aggregate represented by X_i. The variable F_i in X_i is necessary in the case of compact lists. As a matter of fact, in every valuation σ satisfying the constraint:

$$Y \in X_1 \wedge [\![Y \mid X_1]\!] \in X_2 \wedge X_1 \notin X_2$$

it must be $\sigma(X_1) \neq \sigma([\![Y \mid X_1]\!])$. Thus, in σ_C we give the possibility to the first element of $\sigma(X_1)$ to be different from $\sigma(Y)$. We show in the Appendix A that if C is a constraint in pre-solved form and acyclic, then σ_C is stabilizing (Lemma A.4).

We are now ready to state the second condition for the solved form.

Definition 4.10 (Membership Consistency Condition) *Let $E_{\mathbb{T}}$ be one of the four equational theories for aggregates. A constraint C in pre-solved form and acyclic is* membership consistent *if for each pair of literals of the form $t \notin X, t' \in X$ in C we have that:*

$$E_{\mathbb{T}} \not\models \forall(\sigma_C^*(t) = \sigma_C^*(t')).$$

The definition of solved form, therefore, can be given simply as follows:

Definition 4.11 (Solved Form) *A constraint C in pre-solved form is said to be in* solved form *if it satisfies the membership consistent condition.*

Observe that the membership consistency condition implies the acyclicity condition. It is a semantic requirement of equivalence of two terms under a given equational theory. However, this test can be automatized in the following way. As well-known from unification theory (see, e.g., [2, 23]), 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 m.g.u. (ε). Thus, the test is connected with the availability of a unification algorithm for the theory $E_{\mathbb{T}}$. In [11] it is proved that the 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 give us a decision procedure for the above test.

As an example, let C be the pre-solved form and acyclic *Set*-constraint: $a \in Y \wedge Y \in X \wedge X \in Z \wedge \{\{a \mid Y\} \mid X\} \notin Z$. It holds that:

$$\begin{aligned} \sigma_C &= [Y/\{F_Y, a \mid M_Y\}, X/\{F_X, Y \mid M_X\}, Z/\{F_Z, X \mid M_Z\}], \\ \sigma_C^* &= [Y/\{F_Y, a \mid M_Y\}, X/\{F_X, \{F_Y, a \mid M_Y\} \mid M_X\}, \\ &\quad\; Z/\{F_Z, \{F_X, \{F_Y, a \mid M_Y\} \mid M_X\} \mid M_Z\}] \\ \sigma_C^*(X) &= \{F_X, \{F_Y, a \mid M_Y\} \mid M_X\} \\ \sigma_C^*(\{\{a \mid Y\} \mid X\}) &= \{\{a, F_Y, a \mid M_Y\}, F_X, \{F_Y, a \mid M_Y\} \mid M_X\} \end{aligned}$$

The constraint is not in solved form since $E_{Set} \models \forall(\sigma_C^*(X) = \sigma_C^*(\{\{a \mid Y\} \mid X\}))$.

We prove now that solved form constraints are satisfiable in the corresponding privileged models. We prove the property for *Set*-constraints. The proof is similar for the other cases.

Theorem 4.12 (Satisfiability of the Solved Form) *Let C be a constraint in solved form over the language $\mathcal{L}_{Set}$ (resp., $\mathcal{L}_{List}$, $\mathcal{L}_{MSet}$, $\mathcal{L}_{CList}$). Then $\mathcal{SET} \models \exists C$ (resp., $\mathcal{LIST}$, $\mathcal{MSET}$, and $\mathcal{CLIST}$).*

Proof. We split C into the four parts: $C^{=}$, $C^{\in}$, $C^{\notin}$, and $C^{\neq}$, containing $=, \in, \notin$, and $\neq$ literals, respectively. For all pairs of literals $p \in V, r \notin V$ in C let NEQ_{pr} be an auxiliary variable, that will be used as a 'constraint store', initialized to the empty set $\emptyset$. We will use the two auxiliary functions *rank* and *find*. The *rank* of a well-founded set 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:

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

We build a successful valuation γ of C, in various steps.

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

$C^{\in}$ is of the form $p_1^1 \in V_1 \wedge \cdots \wedge p_1^{v1} \in V_1 \wedge \cdots \wedge p_q^{vq} \in V_q$. Consider the member substitution

$$\sigma_C = [V_1/\{F_1, p_1^1, \ldots, p_1^{v1} \mid M_1\}, \ldots, V_q/\{F_q, p_q^1, \ldots, p_q^{vq} \mid M_q\}].$$

Since, by hypothesis, C is acyclic, then σ_C^* can be computed (see Lemma A.4).

For each pair of literals $p \in V$, $r \notin V$ of C consider the equality constraints in solved form $D_1, \ldots, D_k$ that are the solutions to the unification problem $\sigma_C^*(p) = \sigma_C^*(r)$ (since C is in solved form they are all different from the empty substitution). By the results concerning unification (cf. [11]) we have that

$$\sigma_C^*(p) = \sigma_C^*(r) \leftrightarrow \bigvee_{j=1}^{k} (\exists \bar{N}(D_j)),$$

where $\bar{N}$ are new variables, and each D_j is a conjunction of equations which contains at least one atom of the form $A = \{a_1, \ldots, a_h \mid B\}$ with $A \in FV(\sigma_C^*(p)) \cup FV(\sigma_C^*(r))$ and $FV(a_i) \subseteq FV(\sigma_C^*(p)) \cup FV(\sigma_C^*(r))$, or one atom of the form $A = B$ with $A, B \in FV(\sigma_C^*(p)) \cup FV(\sigma_C^*(r))$. Since we want to satisfy $\sigma_C^*(r) \notin \sigma_C^*(V)$ we are interested in satisfying $\sigma_C^*(r) \neq \sigma_C^*(p)$, which is in turn equivalent to:

$$\bigwedge_{j=1}^{k} (\forall N \neg D_j).$$

For doing that, for each D_j we choose an atom of the form $A = \{a_1, \ldots, a_h \mid B\}$ or $A = B$ and we store it in the variable NEQ_{pr}. Points (5) and (6) below will take care of this constraint store.

$C^{\notin}$ 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) and $C^{\neq}$ 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). Let $W_1, \ldots, W_h$ be the variables occurring in C other than $X_1, \ldots, X_m, V_1, \ldots, V_q, Y_1, \ldots, Y_n, Z_1, \ldots, Z_o$.

Let $\bar{s} = \max\{rank(t) :\ t \text{ occurs in } \sigma_C^*(\theta_1(C))\} + 1 + h$.

Let $R_1, \ldots, R_j$ be the variables occurring in $\sigma_C^*(\theta_1(C^{\notin} \wedge C^{\neq}))$ (actually, the variables $\bar{F}, \bar{M}$, and some of the $\bar{Y}$ and $\bar{Z}$) and $n_1, \ldots, n_j$ be auxiliary variables ranging over $\mathbb{N}$.

We build an integer disequation system S 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 $R_{i_1} \neq t$ in $\sigma_C^*(C^{\neq})$

$$S = S \cup \{n_{i_1} \neq n_{i_2} + c : \forall i_2 \neq i_1, \forall c \in find(R_{i_2}, t)\}$$

3. For each literal $\{R_{i_1}, p_j^1, \ldots, p_j^{vj} \mid R_h\} \neq t$ in $\sigma_C^*(C^{\neq})$

$$S = S \cup \{n_{i_1} \neq n_{i_2} + c - 1 : \forall i_2 \neq i_1, \forall c \in \mathit{find}(R_{i_2}, t)\}$$

4. For each literal $t \notin R_{i_1}$ in $\sigma_C^*(C^{\notin})$

$$S = S \cup \{n_{i_1} \neq n_{i_2} + c + 1 : \forall i_2 \neq i_1, \forall c \in \mathit{find}(R_{i_2}, t)\}$$

5. For each literal $t \notin \{R_h, p_j^1, \ldots, p_j^{vj} \mid R_{i_1}\}$, for each $k \leq vj$, for all $R_{i_2} = \{a_1, \ldots, a_h \mid B\}$ in $NEQ_{p_j^k t}$

$$S = S \cup \{n_{i_2} \neq n_{i_3} + c + 1 : \forall i_3 \neq i_2, \forall c \in \mathit{find}(R_{i_3}, a_1)\}$$

6. For each literal $t \notin \{R_h, p_j^1, \ldots, p_j^{vj} \mid R_{i_1}\}$, for each $k \leq vj$, for all $R_{i_2} = R_{i_3}$ in $NEQ_{p_j^k t}$

$$S = S \cup \{n_{i_2} \neq n_{i_3}\}$$

7. For each literal $t \notin \{R_{i_1}, p_j^1, \ldots, p_j^{vj} \mid R_h\}$,

$$S = S \cup \{n_{i_1} \neq n_{i_2} + c : \forall i_2 \neq i_1, \forall c \in \mathit{find}(R_{i_2}, t)\}$$

8. For each literal $t \notin \{R_h, p_j^1, \ldots, p_j^{vj} \mid R_{i_1}\}$,

$$S = S \cup \{n_{i_1} \neq n_{i_2} + c + 1 : \forall i_2 \neq i_1, \forall c \in \mathit{find}(R_{i_2}, t)\}$$

An integer disequation is *safe* if, after expression evaluation, it is not of the form $u \neq u$. A safe disequation has always an infinite number of solutions. A finite set of safe disequations has always an infinite number of solutions. We show that all disequations of S are safe. The disequations generated at point (1) are safe by definition; those introduced in points $(2), (4), (5), (6), (7)$, and (8) are safe since c is always a positive number. We prove that the disequations generated at point (3) are safe. If in C there was a situation of the form $p_1 \in Y \wedge \ldots \wedge p_m \in Y \wedge Y \neq t$ from which we have obtained $\{F_Y, \sigma_C^*(p_1), \ldots, \sigma_C^*(p_m) \mid M_Y\} \neq \sigma_C^*(t)$, then we had, from the definition of solved form, that Y does not occur in t, hence F_Y does not occur at depth 1 in $\sigma_C^*(t)$, hence we do not obtain a disequation of the form $n_{F_Y} \neq n_{F_Y} + 1 - 1$.

From the safeness property, it is possible to find an integer solution to the system S by choosing arbitrarily large values satisfying the constraints. Let $\{n_1 = \bar{n}_1, \ldots, n_j = \bar{n}_j\}$ be a solution and define

$$\theta_2 = [R_i / \{\mathtt{nil}\}^{\bar{n}_i} : \forall i \in \{1, \ldots, j\}] \,.$$

where $\{\mathtt{nil}\}^{\bar{n}}$ denotes the term $\underbrace{\{\cdots\{}_{\bar{n}}\mathtt{nil}\}\cdots\}$ (similarly for the other theories employed).

Let $\gamma = \theta_1 \sigma_C^* \theta_2$ (where $s\mu\nu$ stands for $(s\mu)\nu$) and observe that $C\gamma$ is a conjunction of ground literals. We show that $KE_k^s F_1^s F_2 F_s^3 \models C\gamma$. We analyze each literal of C.

$X = t$: $\theta_1(X)$ coincides syntactically with $\theta_1(t) = t$. Hence, a literal of this form is true in any model of equality.

$t \in X$: $\theta_2(\sigma_C^*(X)) = \{\ldots, \theta_2(\sigma_C^*(t)), \ldots\}$, so the atom is satisfied.

$Z \neq u$: two cases are possible:

1. if there are no atoms of the form $t \in Z$ in C, then the conditions in S and over $\bar{s}$ ensure that $\mathit{rank}(\gamma(Z)) \neq \mathit{rank}(\gamma(u))$;
2. if there is at least one atom of the form $t \in Z$ in C, then $\sigma_C^*(Z) = \{F, t_1, \ldots, t_k \mid M\}$, the conditions in S and over $\bar{s}$ ensure that $\mathit{rank}(\gamma(F)) \neq \mathit{rank}(\gamma(u)) - 1$, hence $\gamma(F)$ is not an element of $\gamma(u)$.

$r \notin Y$: two cases are possible:

1. no atoms of the form $t \in Y$ occur in C: if r is ground, then it can not be an element of Y since $\gamma(Y) = \{\mathtt{nil}\}^i$, with $i \geq \bar{s}$; if r is not ground, then the conditions in S ensure that $\mathit{rank}(\gamma(Y)) \neq \mathit{rank}(\gamma(r)) + 1$;
2. at least one atom of the form $t \in Y$ occurs in C, hence $\sigma_C^*(Y) = \{F, t_1, \ldots, t_k \mid M\}$: if r is ground the result is trivial; if r is not ground then the conditions in S ensure that $\mathit{rank}(\gamma(t_j)) \neq \mathit{rank}(\gamma(r))$ for all $j \leq k$, $\mathit{rank}(\gamma(F)) \neq \mathit{rank}(\gamma(r))$, and $\mathit{rank}(\gamma(M)) \neq \mathit{rank}(\gamma(r)) + 1$.

□

Remark 4.13 *The task of testing whether a pre-solved form constraint C is in solved form could be avoided in the cases of multisets and sets, where all membership atoms can be removed. As a matter of fact, in the privileged models considered for sets and multisets it holds that:*

$$s \in t \leftrightarrow \exists N(t = \{s \mid N\}).$$

We can therefore replace each membership atom $s \in t$ with an equi-satisfiable equality atom $t = \{s \mid N\}$ with N a new variable. This implies that the additional conditions on the pre-solved form are not required at all, since membership atoms can be removed.

5 Constraint Rewriting Procedures

In this section we describe the procedures that can be used to rewrite a given constraint C into a equi-satisfiable disjunction of constraints in pre-solved form. All the procedures have the same overall structure shown in Figure 2: they take a constraint C as their input and repeatedly select an conjunct c in C not in pre-solved form (if any) and apply one of the rewriting rules to it. The procedure stops when the constraint C is in pre-solved or `false` is a conjunct of the constraint. The procedure is non-deterministic. Some rewriting rules have two or more possible non-deterministic choices. Each non deterministic computation returns a constraint of the form above. However there is globally a finite set $C_1, \ldots, C_k$ of constraints non-deterministically returned. The input constraint C and the disjunction $C_1 \vee \cdots \vee C_k$ are equi-satisfiable.

Let $\mathbb{T}$ be one of the theories List, MSet, CList, Set, π 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 pre-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 ·π·;
return C
```

Figure 2: Main loop of constraint rewriting procedures

5.1 Equality Constraints

Unification algorithms for verifying the satisfiability and producing the solutions of equality constraints in the four aggregate's theories have been proposed in [11]. The unification algorithms proposed in [11] fall in the general schema of Figure 2. Some determinism in the statement select c is added to ensure termination. They are called:

Unify_lists for lists	Unify_msets (Unify_bags in [11]) for multisets
Unify_clists for compact lists	Unify_sets 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 constraint is unsatisfiable, or a collection of solved form constraints (Def. 4.11) composed only by equality atoms. In Figure 3 we have reported the rewriting rules for the multisets unification used in algorithm Unify_msets.

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

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

Rules for Unify_msets			
(1)	$X = X$	$\mapsto$	`true`
(2)	$t = X$ t is not a variable	$\mapsto$	$X = t$
(3)	$X = t$ X does not occur in t, X occurs in C	$\mapsto$	$X = t$ and apply the substitution X/t to C
(4)	$X = t$ X is not t and X occurs in 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 for the Unification algorithm for multisets

5.2 Membership and not-Membership Constraints

The rewriting rules for membership and not-membership constraints are justified by axioms (K) and (W) that hold in all the four theories. Therefore, in Figure 4 we give a single definition of these rules. They are used within the main loop in Figure 2 to define the rewriting procedures for membership and not-membership constraints over the considered aggregate. When useful, we will refer to these procedures with the generic names in-$\mathbb{T}$ and nin-$\mathbb{T}$, where $\mathbb{T}$ is any of the aggregate theories.

Let $\mathtt{cons}_{\mathbb{T}}(\cdot, \cdot)$ *be the aggregate constructor for the theory* $\mathbb{T}$

Rules for **in-**$\mathbb{T}$			
(1)	$r \in f(t_1, \ldots, t_n)$ f is not $\mathtt{cons}_{\mathbb{T}}(\cdot, \cdot)$	$\mapsto$	`false`
(2)	$r \in \mathtt{cons}_{\mathbb{T}}(t, s)$	$\mapsto$	$r = t \vee \quad (a)$ $r \in s \quad (b)$
(3)	$r \in X$ $X \in FV(r)$	$\mapsto$	`false`

Rules for **nin-**$\mathbb{T}$			
(1)	$r \notin f(t_1, \ldots, t_n)$ f is not $\mathtt{cons}_{\mathbb{T}}(\cdot, \cdot)$	$\mapsto$	`true`
(2)	$r \notin \mathtt{cons}_{\mathbb{T}}(t, s)$	$\mapsto$	$r \neq t \wedge r \notin s$
(3)	$r \notin X$ $X \in FV(r)$	$\mapsto$	`true`

Figure 4: Parametric rewriting rules for membership and not-membership constraints

Lemma 5.1 *Let* $\mathbb{T}$ *be one of the theories List, MSet, CList, Set, and* $\mathcal{A}_{\mathbb{T}}$ *the privileged 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-$\mathbb{T}(\mathsf{in}$-$\mathbb{T}(C))$*, and* $\bar{N}_i = FV(C_i) \setminus FV(C)$*. Then* $\mathcal{A}_{\mathbb{T}} \models \forall \left(C \leftrightarrow \bigvee_{i=1}^{k} \exists \bar{N}_i C_i \right)$.

Proof. We prove correctness and completeness for lists, thus with respect to the model $\mathcal{LIST}$. Soundness and completeness for the other aggregates are proved in the very same way. Soundness and completeness is proved for each rewriting rule separately since the rules are mutually exclusive. When possible, we simply point out the axioms of the corresponding theory *List* involved in the proof (note that $\mathcal{LIST}$ is a model of those axioms):

in-List, rule (1). $r \in f(t_1, \ldots, t_n)$, with f different from $[\cdot \mid \cdot]$ is equivalent to `true` by axiom (K).

in-List, rule (2). This is exactly axiom (W).

in-List, rule (3). Assume that there is a valuation σ such that $\mathcal{LIST} \models \sigma(r \in X)$. This means that $\sigma(X)$ contains a term of the form: $[s_1, \ldots, s_n, r' \mid t]$ for some terms $s_1, \ldots, s_n, t$, and some term r' in $\sigma(r)$. Axiom (F_3) ensures that X can not be a subterm of r.

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

□

In the above lemma it holds that the lists of variables $\bar{N}_i$ are all empty. However, for the sake of uniformity with respect to the other similar correctness results, we have made them explicit. Let us observe that the rewriting rules for procedure in-MSet and in-Set could safely be extended by the rule:

(4)	$r \in X$ $X \notin FV(r)$	$\mapsto$	$X = \{\!\!\{ r \mid N \}\!\!\}$ $\quad (X = \{r \mid N\})$

where N is a new variable (see also Remark 4.13). In this way, we are sure to completely remove membership atoms from the constraints and that the pre-solved form constraints obtained are in solved form.

5.3 Disequality constraints

Rewriting rules for disequality constraints consist of a part which is the same for the four theories (although parametric with respect to the considered theory), and a part which is specific for each one of the four theories. Rules of the common part are shown in Figure 5, while specific rules are described in the next subsections.

Let $\mathtt{cons}_{\mathbb{T}}(\cdot, \cdot)$ *be the aggregate constructor for the theory* $\mathbb{T}$

Rules for **neq-$\mathbb{T}$**			
(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) $\vdots$ $\vdots$ $s_n \neq t_n$ (n)

Figure 5: General rewriting rules for disequality constraints

5.3.1 Lists

Specific rules for the theory *List* are presented in Figure 6. These rules are inserted in the general schema of Figure 2 to generate the procedure neq-List.

Rules for **neq-List**				
(1)−(5)	see Figure 5			
(6)	$[s_1 \mid s_2] \neq [t_1 \mid t_2]$	$\mapsto$	$s_1 \neq t_1 \vee$	(i)
			$s_2 \neq t_2$	(ii)
(7)	$X \neq f(t_1, \ldots, t_n)$ $X \in FV(t_1, \ldots, t_n)$	$\mapsto$	`true`	

Figure 6: Rewriting rules for disequality constraints over lists

Lemma 5.2 *Let C be a List-constraint, $C_1, \ldots, C_k$ be the constraints non-deterministically returned by* neq-List(C)*, and $\bar{N}_i = FV(C_i) \setminus FV(C)$. Then List $\models \forall \left(C \leftrightarrow \bigvee_{i=1}^{k} \exists \bar{N}_i C_i \right)$.*

Proof. Soundness and completeness of the rewriting rules (and, hence, of the whole rewriting procedure neq-List) are immediate consequence of standard equality axioms and axiom schemata $(F_1), (F_2)$, and (F_3). □

5.3.2 Multisets

Disequality constraints over multisets are simplified using the rewriting rules presented in Figure 7. They make use of functions tail and untail defined in Section 5.1. Using these rules within the generic rewriting scheme of Figure 2 we get the rewriting procedure for disequality constraints over multisets, called neq-MSet.

Rules for **neq-MSet**				
(1)−(5)	see Figure 5			
(6.1)	$\{\!\{ t_1 \mid s_1 \}\!\} \neq \{\!\{ t_2 \mid s_2 \}\!\}$ tail(s_1) and tail(s_2) are the same variable	$\mapsto$	untail$(\{\!\{ t_1 \mid s_1 \}\!\}) \neq$ untail$(\{\!\{ t_2 \mid s_2 \}\!\})$	
(6.2)	$\{\!\{ t_1 \mid s_1 \}\!\} \neq \{\!\{ t_2 \mid s_2 \}\!\}$ tail(s_1) and 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)	$X \neq f(t_1, \ldots, t_n)$ $X \in FV(t_1, \ldots, t_n)$	$\mapsto$	`true`	

Figure 7: Rewriting rules for disequality constraints over multisets

Some words are needed for explaining the rules related to the management of disequalities between multisets; in particular rule (6.2) of Figure 7. If we use directly axiom (E_k^m), we have that:

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

This way, an universal quantification is introduced: this is no longer a constraint according to Definition 4.1.

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 have:

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

In this case, however, we have a quantification on natural numbers: we are outside the language we are studying. The rewriting rule (6.2) adopted in Figure 7 avoids these difficulties introducing

only existential quantification. Its correctness and completeness are proved in the following lemma.

Lemma 5.3 *Let C be a MSet-constraint, $C_1, \ldots, C_k$ be the constraints non-deterministically returned by* neq-MSet(C)*, and $\bar{N}_i = FV(C_i) \setminus FV(C)$. Then $\mathcal{MSET} \models \forall \left(C \leftrightarrow \bigvee_{i=1}^{k} \exists \bar{N}_i C_i \right)$.*

Proof. From Lemma 5.2 we know that the result holds for rules (1)–(5) and (7) for the model $\mathcal{LIST}$. Since permutativity has not been used for that result, and axiom (F_3) holds for both the theories, the same holds for the model $\mathcal{MSET}$. We need to prove correctness and completeness of rewriting rules (6.1) and (6.2).

(6.1) It is immediately justified by axiom schema (F_3^m).

(6.2) The constraint $\{\!\{ t_1 \mid s_1 \}\!\} \neq \{\!\{ t_2 \mid s_2 \}\!\}$ is equivalent to:

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

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

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

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

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

Thus, (2) is equivalent to

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

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

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

(4) $\rightarrow$ (5) Assume there is M so as to satisfy (4). $M = s_1$ will immediately lead to a contradiction. Thus, (5) is satisfied by $N = M$.

(5) $\rightarrow$ (4) Assume there is N so as to satisfy (5). It follows immediately from the fact, true for finite multisets, that $s_1 \neq N$ implies $\{\!\{ t_1 \mid s_1 \}\!\} \neq \{\!\{ t_1 \mid N \}\!\}$. Thus, choose $M = N$.

□

5.3.3 Compact Lists

The rewriting rules for disequality constraints over compact lists are shown in Figure 8. These rules can be immediately exploited in conjunction with the generic scheme of Figure 2 to obtain a rewriting procedure for disequality constraints over multisets–called neq-CList. Soundness and completeness of neq-CList are stated by the following lemma.

Lemma 5.4 *Let C be a CList-constraint, $C_1, \ldots, C_k$ be the constraints non-deterministically returned by* neq-CList(C)*, and $\bar{N}_i = FV(C_i) \setminus FV(C)$. Then $\mathcal{CLIST} \models \forall \left(C \leftrightarrow \bigvee_{i=1}^{k} \exists \bar{N}_i C_i \right)$.*

Proof. For rules (1)–(5) the result follows immediately from those for lists. Rules (7.1)–(7.3) follows from axiom (F_3^c). Rule (6) is exactly axiom (E_k^c). □

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.

Rules for **neq-CList**		
(1)−(5)	see Figure 5	
(6)	$\left.\begin{array}{r}[\![t_1 \mid s_1]\!] \neq [\![t_2 \mid s_2]\!]\end{array}\right\} \mapsto$	
	$t_1 \neq t_2 \vee$	(a)
	$s_1 \neq s_2 \wedge [\![t_1 \mid s_1]\!] \neq s_2 \wedge s_1 \neq [\![t_2 \mid s_2]\!]$	(b)
(7.1)	$\left.\begin{array}{r}X \neq f(t_1, \ldots, t_n) \\ X \in FV(t_1, \ldots, t_n), f \text{ is not } [\![\cdot \mid \cdot]\!]\end{array}\right\} \mapsto$ `true`	
(7.2)	$\left.\begin{array}{r}X \neq [\![t_1, \ldots, t_n \mid X]\!] \\ X \in FV(t_1, \ldots, t_n)\end{array}\right\} \mapsto$ `true`	
(7.3)	$\left.\begin{array}{r}X \neq [\![t_1, \ldots, t_n \mid X]\!] \\ X \notin FV(t_1, \ldots, t_n)\end{array}\right\} \mapsto$	
	$t_1 \neq t_2 \vee$	$(a.1)$
	$\vdots$	$\vdots$
	$t_1 \neq t_n \vee$	$(a.n)$
	$X = \texttt{nil} \vee$	(b)
	$X = [\![N_1 \mid N_2]\!] \wedge N_1 \neq t_1$	(c)

Figure 8: Rewriting rules for disequality constraints over compact lists

Rules for **neq-Set**		
(1)−(5)	see Figure 5	
(6)	$\left.\begin{array}{r}\{t_1 \mid s_1\} \neq \{t_2 \mid s_2\}\end{array}\right\} \mapsto$	
	$Z \in \{t_1 \mid s_1\} \wedge Z \notin \{t_2 \mid s_2\} \vee$	(a)
	$Z \in \{t_2 \mid s_2\} \wedge Z \notin \{t_1 \mid s_1\}$	(b)
(7.1)	$\left.\begin{array}{r}X \neq f(t_1, \ldots, t_n) \\ X \in FV(t_1, \ldots, t_n), f \text{ is not } \{\cdot \mid \cdot\}\end{array}\right\} \mapsto$ `true`	
(7.2)	$\left.\begin{array}{r}X \neq \{t_1, \ldots, t_n \mid X\} \\ X \in FV(t_1, \ldots, t_n)\end{array}\right\} \mapsto$ `true`	
(7.3)	$\left.\begin{array}{r}X \neq \{t_1, \ldots, t_n \mid X\} \\ X \notin FV(t_1, \ldots, t_n)\end{array}\right\} \mapsto$	
	$t_1 \notin X \vee$	(i)
	$\vdots$	$\vdots$
	$t_n \notin X$	(n)

Figure 9: Rewriting rules for disequality constraints over sets

5.3.4 Sets

Disequality constraints over sets are dealt with by the rewriting rules shown in Figure 9, and they constitute the procedure neq-Set.

Some remarks are needed regarding rule (6). As for multisets, axiom (E_k^s) introduces an existentially quantified variable to state equality. Thus, its direct application for stating disequality requires universally quantified constraints that go outside the language. On the other hand, the rewriting rule (6.2) used for multisets can not be used in this context. In fact, the property that $s_1 \neq N$ implies $\{\!\{ t_1 \mid s_1 \}\!\} \neq \{\!\{ t_1 \mid N \}\!\}$, that holds 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 over sets can be obtained by taking the negation of the standard extensionality axiom

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

Lemma 5.5 *Let C be a Set-constraint, $C_1, \ldots, C_k$ be the constraints non-deterministically returned by* neq-Set(C)*, and $\bar{N}_i = FV(C_i) \setminus FV(C)$. Then $\mathcal{SET} \models \forall \left(C \leftarrow \bigvee_{i=1}^{k} \exists \bar{N}_i C_i \right)$.*

Proof. For rules (1)–(5) the result follows from those for lists. Rules (7.1) and (7.2) are exactly axiom (F_3^s). Rule (6) is axiom (E_k), implied by (E_k^s) on $\mathcal{SET}$. □

Remark 5.6 *In our theories an aggregate can be built by starting from any ground uninterpreted Herbrand term—called the* kernel*—and then adding to this term the elements that compose the aggregate. Thus, two aggregates can contain the same elements but nevertheless they can be different because of their different kernels. For example, the two terms* $\{a \,|\, b\}$ *and* $\{a \,|\, c\}$ *denote two different sets containing the same elements (*a*) but based on different kernels (*b *and* c*, respectively).*

Rewriting rules for disequality constraints over aggregates other than sets are formulated in such a way to take care of the (possibly different) kernels in the two aggregates without having to explicitly resort to kernels. Conversely, the rewriting rule for disequality constraints over sets (rule (6)) is not able to "force" disequality between two sets when they have the same elements but different kernels. This the reason why the $(\rightarrow)$ *direction of Lemma 5.5 does not hold.*

*A possible completion of the above procedures to take care of this case is presented in [14]; for doing that some technical complications are introduced. Basically, a new constraint (*ker*) is introduced and the rewriting rule (6) is endowed with a third non-deterministic case:* $\ker(s_1) \neq \ker(s_2)$*. The advantage of this solution is completeness (the* $(\rightarrow)$ *direction of Lemma 5.5). However, for the sake of simplicity, we do not add here the details on the modifications of the rewriting rules for dealing with* ker *that are instead presented in [14].*

6 Constraint solving

In this section we address the problem of establishing if a constraint C is satisfiable or not in the corresponding privileged model. The correspondence result (Theorem 4.5) ensures that the property is inherited by any model.

Constraint satisfiability for the theory $\mathbb{T}$ is checked by the non-deterministic rewriting procedure $\mathsf{SAT}_{\mathbb{T}}$ shown in Figure 10. Its definition is completely parametric with respect to the theory involved. $\mathsf{SAT}_{\mathbb{T}}$ uses iteratively the various rewriting procedures presented in the previous section, until a fixed-point is reached—i.e., any new rewritings do not further simplify the constraint. This happens exactly when the constraint is in pre-solved form or it is `false`. The two conditions that guarantee that a constraint in pre-solved form is in solved form are tested by function $\mathsf{is\_solved}_{\mathbb{T}}$ shown in Figure 11.

By Theorem 4.12 a constraint in solved form is guaranteed to be satisfiable in the corresponding model. Moreover, it will be proved (see Theorem 6.2) that the disjunction of solved form constraints returned by $\mathsf{SAT}_{\mathbb{T}}$ is equi-satisfiable in that model to 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. Moreover, the generated constraint in solved form can be immediately exploited to compute all possible solutions for C.

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

Figure 10: The satisfiability procedure, parametric with respect to $\mathbb{T}$

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

```
function is_solved_T(C)
    build the directed graph G^∈_C;
    if G^∈_C has a cycle
        then return false
        else
            compute σ*_C
            if there is a pair t ∈ X, t' ∉ X in C s.t. T ⊨ ∀(σ*_C(t) = σ*_C(t'))
                then return false
                else return C.
```

Figure 11: Final check for solved form constraints

Theorem 6.1 (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 are in Appendix B.

It is immediate to see, by the definition of the procedures, that if C is different from `false` and not in pre-solved form, then some rewriting rule can be applied. The function $\mathsf{is\_solved}_{MSet}$, whose termination follows from termination of Unify_msets [11], needed for the solved form test $\mathbb{T} \models \forall(\sigma^*_C(t) = \sigma^*_C(t'))$, produces by definition solved form constraints or `false`.

We prove that the repeat cycle can not 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:

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

The first and third element of the tuple are non-negative integers. The second is a multiset of non-negative integers. They are well-ordered [9] 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, in a finite number of non-failing successive rule applications, a constraint C' with lower complexity is reached. We show, by case analysis, this property. Most rule applications decreases the complexity in one step. When this does not happen, we enter in more detail.

Unify_msets(1) α does not increase, β decreases.

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

Unify_msets(3) α decreases by 1.

Unify_msets(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_msets(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, we either reach a situation such that α decreases or a situation where α is unchanged and β decreases (see [11] for details).

Unify_msets(8) After one rule application, we are in the 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 $size(r) + size(s)$ or $size(r) + size(t)$.

nin-MSet(1), (3) Trivially, 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 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. Thus, we enter in some detail. 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_bags replaces equation (6) by:

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

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 this 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*, and *CList* follows from the lemmas in the previous section and two lemmas in the Appendix A.

Theorem 6.2 (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 $FV(C_i) \setminus FV(C)$. 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. Theorem 6.1 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. Soundness and completeness follow from the results proved individually for the procedures involved: Lemma 5.1 for in-$\mathbb{T}$ and nin-$\mathbb{T}$; Lemma 5.2, Lemma 5.3, Lemma 5.4, and Lemma 5.5 for neq-MSet, neq-List, neq-CList, and neq-Set, respectively; Lemma A.6 for is_solved$_{\mathbb{T}}$(C); [11] for unification. □

Corollary 6.3 (Decidability) *Given a $\mathbb{T}$-constraint C, it is decidable whether $\mathcal{A} \models \exists\, C$, where $\mathcal{A}$ is one of the privileged models $\mathcal{LIST}$, $\mathcal{MSET}$, $\mathcal{CLIST}$, $\mathcal{SET}$.*

Proof. From Theorem 6.2 we know that C is equi-satisfiable to $C_1 \vee \cdots \vee C_k$. 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 4.12). □

6.1 Complexity Issues

Complexity of the four unification problems is studied in [11]: the decision problem for unification is proved to be solvable in linear time for lists, and it is NP-complete for the other cases.

In the case of lists, if the constraint is a conjunction of equality and disequality constraints, then the satisfiability problem for a constraint C is solvable in $O(n^2)$ where $n = |C|$ [2, 7]. Instead, the satisfiability problem for conjunctions of membership and disequality constraints over lists is NP-hard. As a matter of fact, let us consider the following instance of 3-SAT:

$$(X_1 \vee X_2 \vee \neg X_3) \wedge (\neg X_1 \vee X_2 \vee X_3) \wedge (X_1 \vee \neg X_2 \vee X_3)\ .$$

The above instance of 3-SAT can be re-written as the following constraint problem:

$$\begin{array}{ccccccc}
X_1 \in [\underline{0}, \underline{1}] & \wedge & Y_1 \in [\underline{0}, \underline{1}] & \wedge & & & \\
[X_1, Y_1] \neq [\underline{0}, \underline{0}] & \wedge & [X_1, Y_1] \neq [\underline{1}, \underline{1}] & \wedge & & & \\
X_2 \in [\underline{0}, \underline{1}] & \wedge & Y_2 \in [\underline{0}, \underline{1}] & \wedge & & & \\
[X_2, Y_2] \neq [\underline{0}, \underline{0}] & \wedge & [X_2, Y_2] \neq [\underline{1}, \underline{1}] & \wedge & & & \\
X_3 \in [\underline{0}, \underline{1}] & \wedge & Y_3 \in [\underline{0}, \underline{1}] & \wedge & & & \\
[X_3, Y_3] \neq [\underline{0}, \underline{0}] & \wedge & [X_3, Y_3] \neq [\underline{1}, \underline{1}] & \wedge & & & \\
[X_1, X_2, Y_3] \neq [\underline{0}, \underline{0}, \underline{0}] & \wedge & [Y_1, X_2, X_3] \neq [\underline{0}, \underline{0}, \underline{0}] & \wedge & [X_1, Y_2, X_3] \neq [\underline{0}, \underline{0}, \underline{0}] & &
\end{array}$$

where $\underline{0}$ and $\underline{1}$ can be represented by `nil` and [`nil`], respectively, and Y_i takes the place of $\neg X_i$ and vice versa. It is immediate to prove that any substitution satisfying the constraint problem is also a solution for the above formula, provided $\underline{0}$ is interpreted as `false` and $\underline{1}$ is interpreted as `true`, and vice versa.

7 Conclusions

In this paper we have extended the results of [11] studying the constraint solving problem for four different theories: 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 privileged 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 in the four contexts.

In particular, we have shown how constraint solving can be developed parametrically for these theories and we have pointed out the differences and similarities between the four kinds of aggregates.

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 can not be defined without using universal quantifiers (or recursion) with the languages analyzed in this paper [10].

Acknowledgments

The authors wish to thank Alberto Policriti, Ashish Tiwari, and Silvia Monica for useful discussions on the topics of this paper. The anonymous referee greatly helped us in improving the presentation of the paper. This work is partially supported by MIUR project *Ragionamento su aggregati e numeri a supporto della programmazione e relative verifiche.*

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 T. Nipkow. *Term Rewriting and All That.* Cambridge University Press, Cambridge, 1998

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

[4] 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.

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

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

[7] 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.

[8] 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.

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

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

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

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

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

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

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

[20] 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.

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

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

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

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

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

A Proofs of Model Properties

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) $\forall p \in \Pi, a_1, \ldots, a_m \in A\, (p^{\mathcal{A}}(a_1, \ldots, a_m) \rightarrow p^{\mathcal{B}}(h(a_1), \ldots, h(a_m)))$. 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 ([5]) *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 it holds that:*

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

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

Proof. For each axioms/axiom schemata (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 interpretation of the membership predicate in $\mathcal{MSET}$ (cf. point (4) of Def. 4.2).

(F_1^m)**:** This axiom holds in $\mathcal{MSET}$, since $f(t_1, \ldots, t_n)$ and $f(s_1, \ldots, s_n)$ can be in 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 trivially, by definition of $\mathcal{MSET}$, since terms beginning with different free symbols belong to different classes.

$(F_3), (F_3^m)$**:** The fact that $\mathcal{MSET} \models (F_3)$ and $\mathcal{MSET} \models (F_3^m)$ holds in virtue of the finite size of each ground term; 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 [23].

(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 $h(\overline{t}) = t^{\mathcal{M}}$ is an embedding of $\mathcal{MSET}$ in $\mathcal{M}$.*

Proof. We will prove the following facts:

1. The definition of $h(\overline{t})$ 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 $\mathcal{M}$, 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

$$h(f^{\mathcal{MSET}}(\overline{t_1}, \ldots, \overline{t_n})) = f^{\mathcal{M}}(h(t_1), \ldots, h(t_n))$$

   Now,

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

(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 $h(\overline{\textcircled{t_1}}) = h(\overline{\textcircled{t_2}})$, then $\overline{\textcircled{t_1}} = \overline{\textcircled{t_2}}$.

   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 $\overline{\textcircled{t_1}} = \overline{\textcircled{t_2}}$.

   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 $\overline{\textcircled{t_1}} \neq \overline{\textcircled{t_2}}$, 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 $\overline{\textcircled{s_i}} = \overline{\textcircled{u_{b(i)}}}$. This means that $m = n$ and that there is a term t_2' in $\overline{\textcircled{t_2}}$ 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 $\overline{\textcircled{t_2}} = \overline{\textcircled{t_2'}} = \overline{\textcircled{\{\!|\, s_1, \ldots, s_n \,|\, r \,|\!\}}} = \overline{\textcircled{t_1}}$, 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 $\textcircled{t} \in^{\mathcal{MSET}} \textcircled{s}$.

□

Lemma A.4 *If C is a constraint in pre-solved form and acyclic, then σ_C is stabilizing.*

Proof. We prove that $\sigma_C^* \equiv \sigma_C^{q-1}$, where q is the number of variables which occur in the right-hand side of membership atoms.

The acyclicity condition ensures that there are no loops in the graph $\mathcal{G}_C^{\in}$. Consider now the substitution σ_C and let B be the set of the nodes of the graph that belong to its domain (we identify variables and corresponding nodes). Each application of σ_C on the terms of its codomain can be intuitively mimicked by a game that updates the value of B with the nodes corresponding to the variables occurring in the terms $\sigma_C(B)$. These nodes can be computed by collecting the nodes that can be reached by crossing an edge from a node of B (new variables F_i, M_i are all different, and they are not in the domain of σ_C, so we can forget them). The process will terminate when either B is empty or it contains only variables that are not in the domain of σ_C. Since $\mathcal{G}_C^{\in}$ is acyclic, this process must terminate, and since the longest path in the graph is shorter than q, it is plain o see that $q - 1$ is an upper bound to the number of iterations. □

Lemma A.5 *Let $\mathbb{T}$ be one of the theories List, MSet, CList, $\mathcal{A}_{\mathbb{T}}$ the model (structure) which corresponds with $\mathbb{T}$, and $E_{\mathbb{T}}$ the associated equational theory. Let t, t' be two terms and C a solved form constraint over the language $\mathcal{L}_{\mathbb{T}}$, such that $FV(t) \cup FV(t') \subseteq FV(C)$. If $\mathcal{A}_{\mathbb{T}} \not\models \forall(t = t')$, then $E_{\mathbb{T}} \not\models \forall(\sigma_C^*(t) = \sigma_C^*(t'))$.*

Proof. Let $R = \{X_1, \ldots, X_n\}$ be the set of variables over which σ_C is defined. By induction on the sum of the complexities of t and t' we prove the following property that implies the thesis of the lemma.

> *If there exists θ such that $\mathcal{A}_{\mathbb{T}} \models \theta(t) \neq \theta(t')$, then there exists θ' such that $\mathcal{A}_{\mathbb{T}} \models \theta'(\sigma_C^*(t)) \neq \theta'(\sigma_C^*(t'))$.*

Let us consider the valuation θ'' defined as:

$$\theta''(Y) = \begin{cases} \theta(Y) & \text{if } Y \notin R \\ \theta(X_i) & \text{if } Y \equiv M_{X_i} \end{cases}$$

Observe that θ'' is not defined over the variables $F_{X_1}, \ldots, F_{X_n}$.

Let $m = \max\{size(\theta''(\sigma_C^*(t))), size(\theta''(\sigma_C^*(t')))\} + 1$. We can now define the valuation θ' in the following way:

$$\theta'(Y) = \begin{cases} [\mathtt{nil}]^{m*i}(\{\!|\,\mathtt{nil}\,|\!\}^{m*i}, [\![\,\mathtt{nil}\,]\!]^{m*i}) & \text{if } Y \equiv F_{X_i} \\ \theta''(Y) & \text{otherwise} \end{cases}$$

If $t = Y_1$ and $t' = Y_2$ are variables then:

- if σ_C is not defined neither on Y_1 nor on Y_2, then $\theta'(\sigma_C^*(Y_1)) = \theta'(Y_1) = \theta(Y_1) \neq \theta(Y_2) = \theta'(Y_2) = \theta'(\sigma_C^*(Y_2))$;
- if σ_C is defined on Y_1 and not on Y_2 (or viceversa), then $size(\theta'(\sigma_C^*(Y_1))) \geq size(\theta'(F_{Y_1})) > size(\theta'(Y_2))$;
- if σ_C is defined both on Y_1 and on Y_2, then:

  List and *CList*: $\theta'(\sigma_C^*(Y_1))$ and $\theta'(\sigma_C^*(Y_2))$ differ on their first element.

  MSet: $\theta'(\sigma_C^*(Y_1))$ and $\theta'(\sigma_C^*(Y_2))$ differ on their elements $\theta'(F_{Y_1})$ and $\theta'(F_{Y_2})$.

If $t = Y$ is a variable and t' is $f(t'_1, \ldots, t'_h)$, also when f is of the form $[\,\cdot \mid \cdot\,], [\![\,\cdot \mid \cdot\,]\!], \{\!|\,\cdot \mid \cdot\,|\!\}$, then:

- if σ_C is not defined on $FV(t') \cup Y$, then we have immediately the thesis since $\theta'(\sigma_C^*(Y)) = \theta(Y)$ and $\theta'(\sigma_C^*(t')) = \theta(t')$;
- if σ_C is defined on Y, but not on $FV(t')$, then we have the thesis since $size(\theta'(\sigma_C^*(Y))) > size(\theta(t'))$;
- if σ_C is defined on at least one variable of t' and not on Y, then as in the previous case we have the thesis;
- if σ_C is defined on Y and on at least one of the variables of t', then:

  List and *CList*: it can never be the case that the first element of $\theta'(\sigma_C^*(Y))$—i.e. $\theta'(F_Y)$—is equal to the first element of $\theta'(\sigma_C^*(t'))$; this follows from the conditions we have imposed on all the $\theta'(F_{X_i})$.

  MSet: two cases are possible: $\theta'(F_Y)$ is not an element of $\theta'(\sigma_C^*(t'))$, from which we have the thesis.
  $\theta'(F_Y)$ is an element of $\theta'(\sigma_C^*(t'))$: this means that $\mathsf{tail}(t') = Y$, hence the thesis follows.

If t is $f(t_1, \ldots, t_h)$ and $t' = g(t'_1, \ldots, t'_k)$, with f different from g, then it is trivial.
If t is $f(t_1, \ldots, t_h)$ and t' is $f(t'_1, \ldots, t'_h)$, with f different from $[\,\cdot \mid \cdot\,], [\![\,\cdot \mid \cdot\,]\!], \{\!|\,\cdot \mid \cdot\,|\!\}$, then by inductive hypothesis we have the thesis.
If t is $[t_1 \mid t_2]$ and t' is $[t'_1 \mid t'_2]$, then from $\mathcal{LIST} \models \theta(t) \neq \theta(t')$ we have that it must be $\mathcal{LIST} \models \theta(t_1) \neq \theta(t'_1)$ or $\mathcal{LIST} \models \theta(t_2) \neq \theta(t'_2)$, hence, in both cases, we obtain the thesis by inductive hypothesis.
If t is $[\![\, t_1 \mid t_2 \,]\!]$ and t' is $[\![\, t'_1 \mid t'_2 \,]\!]$, then from $\mathcal{CLIST} \models \theta(t) \neq \theta(t')$ we have that it must be $\mathcal{CLIST} \models \theta(t_1) \neq \theta(t'_1)$ or $\mathcal{CLIST} \models \theta(t_2) \neq \theta(t'_2) \wedge \theta(t_2) \neq [\![\, \theta(t'_1) \mid \theta(t'_2) \,]\!] \wedge \theta(t'_2) \neq [\![\, \theta(t_1) \mid \theta(t_2) \,]\!]$, hence:

- in the first case we obtain the thesis by inductive hypothesis on t_1 and t'_1.
- in the second case by inductive hypothesis on t_2 and t'_2, on t_2 and $[\![\, t'_1 \mid t'_2 \,]\!]$, on t'_2 and $[\![\, t_1 \mid t_2 \,]\!]$, we obtain that $\mathcal{CLIST} \models \theta'(\sigma_C^*(t_2)) \neq \theta'(\sigma_C^*(t'_2))$ and $\mathcal{CLIST} \models \theta'(\sigma_C^*(t_2)) \neq \theta'(\sigma_C^*([\![\, t'_1 \mid t'_2 \,]\!]))$ and $\mathcal{CLIST} \models \theta'(\sigma_C^*(t'_2)) \neq \theta'(\sigma_C^*([\![\, t_1 \mid t_2 \,]\!]))$, which implies our thesis.

If t is $\{\!|\, t_1 \mid t_2 \,|\!\}$ and t' is $\{\!|\, t'_1 \mid t'_2 \,|\!\}$, then:

- if $\mathsf{tail}(t_2)$ and $\mathsf{tail}(t'_2)$ are the same variable, the we obtain the thesis by inductive hypothesis on $\mathsf{untail}(\{\!|\, t_1 \mid t_2 \,|\!\})$ and $\mathsf{untail}(\{\!|\, t'_1 \mid t'_2 \,|\!\})$;
- if $\mathsf{tail}(t_2) = Y$ and $\mathsf{tail}(t'_2) = Y'$ are not the same variable and σ_C is not defined on Y or on Y', then $\theta'(F_Y)$ or $\theta'(F_{Y'}$ is not an element of both $\theta'(\sigma_C^*(t))$ and $\theta'(\sigma_C^*(t'))$;
- if $\mathsf{tail}(t_2) = Y$ and $\mathsf{tail}(t'_2) = Y'$ are not the same variable and σ_C is not defined on Y and on Y', then we can restrict ourselves to the case in which there is an element s of $\theta(t)$ which is not an element of $\theta(t')$ (in the general case we would have to consider that there exists s such that there are m occurrences of s in $\theta(t)$ and n occurrences in $\theta(t')$ with $m \neq n$):
  - if s is an element of $\theta(Y)$, then, from the fact that σ_C is not defined on Y, we have the thesis, since it cannot be the case that one of the elements of $\mathsf{untail}(t')$ becomes equal to $\theta(s)$ (the new elements have a size which is greater);
  - if s is an element of $\mathsf{untail}(t)$, then we have $t = \{\!|\, u_1, \ldots, u_h, \ldots, u_m \mid Y \,|\!\}$ and $s = \theta(u_h)$, hence, from the inductive hypothesis, we have that $\theta'(\sigma_C^*(u_h))$ is still different from all elements of $\theta'(\sigma_C^*(\mathsf{untail}(t')))$, and it is immediate that it is different from all the elements of $\theta'(Y')$, hence $\theta'(\sigma_C^*(u_h))$ is an element of $\theta'(\sigma_C^*(t))$ which is not in $\theta'(\sigma_C^*(t'))$.

□

Lemma A.6 *Let $\mathbb{T}$ be one of the theories List, CList, MSet and Set, and C a constraint in pre-solved form over the language of $\mathbb{T}$. If* $\mathsf{is\_solved}_{\mathbb{T}}(\mathsf{C})$ *returns* `false`*, then C is not satisfiable in the model $\mathcal{A}_{\mathbb{T}}$ which corresponds with $\mathbb{T}$.*

Proof. If $\mathsf{is\_solved}_{\mathbb{T}}(\mathsf{C})$ returns `false` because $\mathcal{G}_C^{\in}$ has a cycle then the result is trivial, since all aggregates in $\mathcal{A}$ are well-founded. Otherwise:

For *List*, *MSet*, *CList*: From Lemma A.5 we know that $\mathbb{T} \models \forall(\sigma_C^*(t) = \sigma_C^*(t'))$ implies $\mathbb{T} \models \forall(t = t')$, hence, since $t \in X$ and $t' \notin X$ are in C, C is not satisfiable in the model $\mathcal{A}$ which corresponds with $\mathbb{T}$.

For *Set*: Let $\sigma_C^* \equiv [X_1/\{F_1, p_1^1, \ldots, p_1^{k_1} \mid M_1\}, \ldots, X_q/\{F_q, p_q^1, \ldots, p_q^{k_q} \mid M_q\}]$, we have that if $\mathcal{SET} \models C\gamma$, then $\mathcal{SET} \models (C\sigma_C^*)\gamma'$, where γ' is defined as follows

$$\gamma'(Y) = \begin{cases} \gamma(X_i) & \text{if } Y \equiv M_i \\ p_i^1 & \text{if } Y \equiv F_i \\ \gamma(Y) & \text{otherwise} \end{cases}$$

Hence, if $\mathsf{is\_solved}_{Set}$ returns `false` this means that $C\sigma_C^*$ is not satisfiable in $\mathcal{SET}$, which implies that C is not satisfiable in $\mathcal{SET}$.

□

B Termination Proofs (Theorem 6.1)

Termination of SAT_{List}

Using the same measure as for SAT_{MSet} termination follows. □

Termination of SAT_{CList}

Finding a global decreasing measure implies that this measure is decreased by each rule of each algorithm involved. The measure developed in [11] for proving termination of Unify_clists is rather complex. This is due to the fact that new variables are (apparently) freely introduced in the constraint by this procedure. Instead of extending such complex measure to the general case, we use here a different approach for proving termination. The proof is based:

- on the fact that each single rewriting procedure terminates (for Unify_clists it follows from [11]; for the other three procedures the result is trivial) and
- on the fact that it is possible to find a bound on the number of possible repeat cycles.

The remaining part of the proof is devoted to find this bound. First of all observe that:

- After the execution of in-CList there are only membership atoms of the form $t \in X$ with $X \notin FV(t)$. New equations can be introduced.
- After the execution of in-CList there are only not-membership literals of the form $t \notin X$ with $X \notin FV(t)$. New disequality constraints can be introduced. Membership atoms are not introduced.
- After the execution of neq-CList there are only disequality constraints of the form $X \neq t$ with $X \notin FV(t)$. New equations can be introduced. $\in$ and $\notin$-constraints are not introduced.
- Unify_clists eliminates all equality constraints producing a substitution. This substitution, when applied to membership, not-membership, and disequality literals in pre-solved form can force a new execution of the procedures in-CList, nin-CList, and neq-CList. However, new executions of Unify_clists are possible only if in-CList and neq-CList introduce new equations. In the following we will find a bound on the number of possible new equations inserted.

Let us analyze membership constraints. Each membership atom of the form $t \in [\![s' \mid s'']\!]$ is rewritten to `false` or to $t = s' \vee t \in s''$. This means that in each non-deterministic branch of the rewriting process *at most* one equation is introduced for each initial membership atom. Thus, if k is the number of membership atoms in C at the beginning of the computation, at most k equality atoms (that can fire

Unify_clists) can be introduced. If we prove termination with $k = 0$ then full termination easily follows, since it is the same as considering k successive (terminating) executions.

Let us consider the procedure neq-CList. Action (7.2) can replace a disequality constraint of the form: $X \neq [\![t_1, \ldots, t_n \,|\, X]\!]$ with the following equations, identifying a substitution:

$$X = \texttt{nil} \tag{8}$$

$$X = [\![N_1 \,|\, N_2]\!] \text{ with } N_1, N_2 \text{ new variables.} \tag{9}$$

Let us analyze the various cases in which substitutions of this form have some effects on the constraint.

- there is $t \in X$ in C. This is not possible by hypothesis, since $k = 0$.
- $t \notin X$ or $X \neq t$ and we know that X does not occur in t. This implies a finite number of executions of rules of nin-CList or neq-CList. Since X is not in t and the variables N_1 and N_2 are newly introduced, it is impossible to generate a situation firing rule (7.2).
- Assume there are more than one equation introduced for the same variable X.
  - If they are all of the form (8), then Unify_clists will apply the substitution and remove the redundant equations.
  - If they are all of the form (9), then Unify_clists will perform a unification process between these new equations. The particular form of the equations allows us to see that the effect is to introduce new equations of the form $N_1 = N_1'$ between all the new variables used as elements and equations of the form $N_2 = N_2'$ or $N_2 = [\![N_1' \,|\, N_2']\!]$ between the new variables used as rests. The situation is similar to that in which a unique substitution is computed.
  - If there are both equations of the form (8) and of the form (9), then a failing (thus, terminating) situation will be detected by Unify_clists. □

Termination of SAT_{Set}

Finding a global decreasing measure implies that this measure is decreased by each rule of each algorithm involved. This is rather complex since it must subsume the measure developed in [11] for proving termination of Unify_sets. Thus, instead of extending such complex measure, we use here a different approach for proving termination. The proof is based:

- on the fact that each single rewriting procedure terminates (for Unify_sets it follows from [11]; for the other three procedures the result is trivial) and
- on the fact that it is possible to control the number of new calls to unification.

In order to simplify the proof we assume a strategy for handling the non-determinism. The strategy will be pointed out during the discussion.

As observed in the proof of SAT_{CList}, if k is the number of membership atoms in C at the beginning of the computation, at most k equality atoms (that can fire Unify_sets) can be introduced. For this reason, we can safely forget this kind of constraints from the whole reasoning.

The only problem for termination is given by rules (6a) and (6b) of neq-CList. As a strategy, we can unfold the application of this rules (actually, adding a bit of determinism to the whole procedure). This means that rule (6a) (for (6b) the situation is symmetrical) is as follows: assume that $\{t_1 \,|\, s_1\}$ is $\{v_1, \ldots, v_m \,|\, h\}$ and $\{t_2 \,|\, s_2\}$ is $\{w_1, \ldots, w_n \,|\, k\}$, with h, k variables or terms of the form $f(\ldots), g(\ldots)$, f and g different from $\{\cdot \,|\, \cdot\}$. The global effect of the subcomputation is that of returning a constraint of the form ($1 \leq i \leq m$):

$$N = v_i, v_i \neq w_1, \ldots, v_i \neq w_n, v_i \notin k \tag{10}$$

or one constraint of the form

$$h = \{N \,|\, N'\}, N \neq w_1, \ldots, N \neq w_n, N \notin k \tag{11}$$

if h is a variable. Notice that the application of this substitution is a sort of application of rule (4) of the procedure in-Set.

In the following discussion let us assume that termination by failure do not occur (but, in this case, termination follows trivially). Suppose to have already executed the first cycle of the repeat loop. Local termination ensures that this can be done in finite time. In the constraint there are no equations, while there can be negated membership and disequality literals not necessarily in pre-solved form.

Let us execute procedure nin-Set. No equations are introduced. In the constraint there are not-membership literals in pre-solved form and disequality constraints not necessarily in pre-solved form.

Let us execute the procedure neq-Set. We adopt a weak strategy to face the non-determinism: delay the constraints that fire action (6) as much as possible. This means that after a finite time the constraint is composed by a number of constraints of the form $X \neq t$ or $t \notin X$ with $X \notin FV(t)$ plus a (possibly empty) constraint $\tilde{C}$ of constraints all firing action (6) of neq-Set. Pick one constraint c from $\tilde{C}$ and consider the possible non-deterministic executions.

- Assume that the situation of case (11) above does not occur in a non-deterministic branch. Then (see case (10)) the constraint c is replaced in C by a number of constraints $v_i \neq w_j$ of fewer size. If they do not fire action (6) they can be directly processed to reach a pre-solved form. Otherwise, they are inserted in $\tilde{C}$, but since they are of fewer size, if the situation of case (11) never occur, this again implies termination.
- Assume now that the situation of case (11) occurs when processing the constraint c. Constraints

$$N \neq w_1, \ldots, N \neq w_n, N \notin k$$

are introduced. Constraints in pre-solved form of the form above, with N a variable introduced as element of a set by action (6), are said *passive constraints*. Variables N of this form are inserted in the constraint only by this step. We will see that passive disequality constraints remain in pre-solved form forever while negated membership passive literals have a controlled growth.

Assume to apply immediately the substitution $h/\{N \mid N'\}$. Its effect can be the following, according to the position of h in a constraint:

  - $X \neq t[h]$ or $t[h] \notin X$: the terms gets changed but the constraints remain in pre-solved form.
  - $s[h] \neq t$ or $s \neq t[h]$ or $s[h] \neq t[h]$: the terms change but the constraints remain in $\tilde{C}$ to be processed later.
  - $t \notin h$ is transformed to $t \notin \{N \mid N'\}$. One step of nin-Set is applied to obtain: $t \neq N \wedge t \notin N'$. The first constraint is immediately transformed into $N \neq t$ (a passive constraint) while the second is in pre-solved form. Observe that if $t \notin h$ is passive (i.e., t is a variable of type N), then only passive constraints are introduced.
  - $h \neq t$ is transformed to $\{N \mid N'\} \neq t$. Observe that $h \neq t$ can not be a passive constraint since h is a 'rest' variable while the variables of passive constraints are 'element' variables, like N here. A constraint in pre-solved form is no longer in pre-solved form. Let us apply the rewriting rules to it. It is immediately rewritten to `true` (e.g., when t is $f(\cdots)$, $f \neq \{\cdot \mid \cdot\}$) or it becomes in pre-solved form (when t is a variable) or action (6) can be applied.

    Both in cases (10) and in the case (11) we introduce a number of passive constraints and, in the last case, a substitution $N'/\{N_1 \mid N_1'\}$ is applied. Notice that the global effect on the system it the fact that in the other constraints the original variable h is replaced by $\{N, N_1 \mid N_1'\}$. This means that this situation can be performed at most once per each occurrence of h. And, the reasoning starting from substitutions of the form $\{N, N_1, \ldots, N_\ell \mid N_\ell'\}$ is the same as that done here for $N'/\{N_1 \mid N_1'\}$. At the end of the process, the number of constraints in $\tilde{C}$ is decreased and we have only introduced pre-solved form and passive constraints. □