A Uniform Axiomatic View of Lists, Multisets, and Sets, and the Relevant Unification Algorithms[*]

Agostino Dovier[†]
Università di Verona
dovier@sci.univr.it

Alberto Policriti[‡]
Università di Udine
policrit@dimi.uniud.it

Gianfranco Rossi[§]
Università di Parma
gianfr@prmat.math.unipr.it

Abstract

The first-order theories of lists, multisets, compact lists (i.e., lists where the number of contiguous occurrences of each element is immaterial), and sets are introduced via axioms. Such axiomatizations are shown to be very well-suited for the integration with free functor symbols governed by the classical Clark's axioms in the context of (Constraint) Logic Programming. Adaptations of the extensionality principle to the various theories taken into account is then exploited in the design of unification algorithms for the considered data structures. All the theories presented can be combined providing frameworks to deal with several of the proposed data structures simultaneously. The unification algorithms proposed can be combined (merged) as well, to produce engines for such combination theories.

1 Introduction

Lists are a fundamental data structure in programming languages. In lists, the order of elements and the number of occurrences of each element are meaningful information that can be advantageously exploited by applications. However, there are a number of situations where these properties turn out to be too restrictive: different data abstractions may fit more naturally the problem requirements. In particular, it is often convenient to consider the following data abstractions:

- *sets*, in which the order of elements and the number of occurrences of each element do not matter;
- *multisets* (or bags), in which the number of occurrences of each element is important, whereas the order of elements does not matter.

The first part of this paper is devoted to give a simple and yet sufficiently expressive axiomatic presentation of these data abstractions. Such axiomatizations, starting from lists and considering the usual list constructor operator (here denoted $[\cdot\,|\,\cdot]$, *à la* PROLOG), adds more and more constraints in order to correctly characterize (somehow minimally) the corresponding constructors for sets and multisets.

In particular, in the proposed axiomatization, sets will be presented simply as lists built starting from the emptyset using a set constructor operator having permutativity and absorption properties. Such operator—first defined by Bernays in axiom 2 of group II of its celebrated axiomatization of Set Theory (see [4]), and subsequently named `with` in the SETL programming language (see [23])—takes an element a and a set x as input and produces $x \cup \{a\}$ as output.

Multisets are introduced in a very similar way, using an operator which has only the permutativity property. From this point of view, it will turn out rather natural to introduce also:

- *compact lists*, in which the order of elements is important whereas the number of *contiguous* occurrences of each element is not.

[*]We acknowledge partial support from C.N.R. Grant 97.02426.CT12, C.N.R. project SETA, and from the MURST project *"Tecniche formali per la specifica, l'analisi, la verifica, la sintesi e la trasformazione di sistemi software"*.

[†]Address for correspondence: Dipartimento Scientifico-tecnologico, Università di Verona, Strada Le Grazie, 37134, Verona, Italy

[‡]Address for correspondence: Dipartimento di Matematica e Informatica, Università di Udine, Via delle Scienze 206, 33100, Udine, Italy

[§]Address for correspondence: Dipartimento di Matematica, Università di Parma, Via M. D'Azeglio 85/A, 43100 Parma, Italy

Compact lists will be obtained from lists adding the absorption property only. To give an intuition of their semantics, consider the following problem: assume a module of an operating system must keep track of the processes using a specific resource. The operating system can record the name of the process owning the resource at given time instants. The process can change at any instant, but the fact that in subsequent instants the same process was granted use of the resource is immaterial. Thus, the required information is conveniently modeled by a compact list, in which contiguous occurrences of equal elements collapse (cf. § 2.3 for the formal definition of compact lists).

An important requirement for the axiomatizations we are looking for is that they can be easily (and incrementally) adapted to all the data abstractions that we are considering, from lists to sets. In other words, we would like to have an incrementally developed theory which can include as sub-cases the theories for all the considered data structures.

The framework in which we assume the above mentioned data structures are to be incorporated is a Constraint Logic Programming one [12]. Hence, we assume that the language provides (possibly zero or infinite) free functional symbols in addition to the data structure constructors. The theories we are considering here, therefore, are so-called *hybrid theories*, in which the basic objects are Herbrand terms built out of interpreted as well as uninterpreted symbols. Moreover, in order to clarify the semantics of the proposed extensions of logic programming, all the considered data structures are presented together with suitable (preferential) models having the usual terms of a logic language as support.

The axiomatic approach of the first part of the paper, will lead us naturally, in the second part, to introduce different, though tightly related unification algorithms for the previously defined data structures, starting from the standard unification algorithm (cf., e.g., [11, 15, 18, 1]) for lists, up to the algorithm for the more complex set unification problem (which has been showed to be NP-complete, for instance, in [13]).

The equality criteria introduced in the axiomatic presentation of the data structures can be easily converted into effective procedures for testing equality to be used in the relevant unification algorithms. Moreover, though distinct, the algorithms could be easily merged to solve the unification problem involving all the considered data structures.

The paper is organized as follows. In § 2 we give an axiomatic characterization of the considered data structures, showing the close connections among the relevant theories. Then, in § 3, the unification algorithms somehow suggested by the axioms expressing equality in the different contexts considered in the previous section are presented. Finally, some conclusions and possible directions for future work are hinted at in § 4.

2 An axiomatic view of lists, multisets, compact lists, and sets

In what follows we will use the standard Prolog syntactic conventions and notations for the list constructor symbol $[\,\cdot\,|\,\cdot\,]$; moreover, the constant `nil` will denote the empty list. As usual, the list $[\,s_1\,|\,[\,s_2\,|\,\cdots\,[\,s_n\,|\,t\,]\cdots]]$ will be denoted by $[s_1,\dots,s_n\,|\,t]$ or simply by $[s_1,\dots,s_n]$ when t is `nil`. Thus, for instance, $[\,a\,|\,\mathtt{nil}\,]$ and $[\,a\,|\,[\,b\,|\,X\,]]$, where X is a variable, are two lists, which can also be denoted simply as $[\,a\,]$ and $[\,a,b\,|\,X\,]$, respectively.

In addition, the following functional symbols are introduced to denote multisets, compact lists, and sets (empty multiset, compact list, and set are all denoted by `nil`):

- $\{\!\!\{\,\cdot\,|\,\cdot\,\}\!\!\}$ (of arity 2) for multisets,
- $[\![\,\cdot\,|\,\cdot\,]\!]$ (of arity 2) for compact lists,
- $\{\cdot\,|\,\cdot\}$ (of arity 2) for sets.

Notational conventions similar to those used for lists will be freely exploited also for multisets, compact lists, and sets. For example, $\{\,a,b\,|\,X\}$ is used to denote a partially specified set with two elements a and b and a variable part X.

2.1 Lists

Consider the first-order theory in a language containing equality '$\doteq$' and membership '$\in$' as binary predicate symbols, and consisting of the axioms of equality and of the two following axioms

$$
\begin{array}{ll}
(N) & \exists z \forall x\,(x \notin z) \\
(W) & \forall y\, v \exists w \forall x\,(x \in w \leftrightarrow x \in v \vee x \doteq y).
\end{array}
$$

By skolemizing (N) and (W), we introduce the two functional symbols $\texttt{nil}$ and $[\,\cdot\,|\,\cdot\,]$, and we can rewrite (N) and (W) as

$$
\begin{array}{ll}
(N^l) & \forall x\,(x \notin \texttt{nil}) \text{ and} \\
(W^l) & \forall y\, v\, x\,(x \in [\,y\,|\,v\,] \leftrightarrow x \in v \vee x \doteq y)\ .
\end{array}
$$

The language of the theory consists of the signatures $\Pi = \{\doteq, \in\}$ for the predicate symbols, and $\Sigma = \{\texttt{nil}, [\,\cdot\,|\,\cdot\,], \cdots\}$ for the functional symbols. In Theorem 1 it is shown that any model of NW (the theory consisting of the axioms (N) and (W)) must necessarily be *infinite*. The proof of Theorem 1 follows from the two lemmata below.[1]

Lemma 1. $NW \vdash \forall yv\,(\texttt{nil} \not\doteq [\,y\,|\,v\,])$.

Proof. By (N), $NW \vdash \forall x\,(x \notin \texttt{nil})$. If, by contradiction, $NW \vdash \texttt{nil} \doteq [\,\bar{y}\,|\,\bar{v}\,]$, for some y and v, then $NW \vdash \forall x\,(x \notin [\,\bar{y}\,|\,\bar{v}\,])$ using equality axioms. On the other hand, (W) guarantees that $NW \vdash \bar{y} \in [\,\bar{y}\,|\,\bar{v}\,]$. ☐

From the above result we have that $NW \vdash \texttt{nil} \not\doteq [\,\texttt{nil}\,]$, $NW \vdash \texttt{nil} \not\doteq [\,[\,\texttt{nil}\,]\,]$, $NW \vdash \texttt{nil} \not\doteq [\,[\,[\,\texttt{nil}\,]\,]\,]$, and so on. Let $[\,\texttt{nil}\,]^n$ be defined, by induction, as follows:

$$
\begin{cases}
[\,\texttt{nil}\,]^0 & = \texttt{nil} \\
[\,\texttt{nil}\,]^{n+1} & = [\,[\,\texttt{nil}\,]^n\,]
\end{cases}
$$

Lemma 2. *For any pair of different and non-negative integers i and j, $NW \vdash [\,\texttt{nil}\,]^i \not\doteq [\,\texttt{nil}\,]^j$.*

Proof. Assume, without loss of generality, $i < j$. We prove the lemma by induction on i.
If $i = 0$ the claim follows directly from Lemma 1.
For $i > 0$. First of all notice that for any $k > 0$:

$$
NW \vdash x \in [\,\texttt{nil}\,]^k \to x \doteq [\,\texttt{nil}\,]^{k-1} \quad \text{and} \quad NW \vdash [\,\texttt{nil}\,]^{k-1} \in [\,\texttt{nil}\,]^k\ .
$$

Hence, $NW \vdash [\,\texttt{nil}\,]^{i-1} \in [\,\texttt{nil}\,]^i$ and $NW \vdash [\,\texttt{nil}\,]^{j-1} \in [\,\texttt{nil}\,]^j$. If we assume, by contradiction, that $NW \vdash [\,\texttt{nil}\,]^i \doteq [\,\texttt{nil}\,]^j$, we have that $NW \vdash [\,\texttt{nil}\,]^{i-1} \doteq [\,\texttt{nil}\,]^{j-1}$, which negates the inductive hypothesis. ☐

Lemma 2 guarantees that in any model of NW the $[\,\texttt{nil}\,]^i$s are interpreted as pairwise different objects. Hence,

Theorem 1. *Any model of NW is infinite.*

Proof. Immediate from Lemmata 1 and 2. ☐

It has been shown (cf. [27, 21, 19, 3]) that NW is *undecidable* as well as all its consistent extensions (essential undecidability). Since NW is finitely axiomatizable, from this fact it follows that it is not complete (otherwise it would be decidable), as implicitly proved also in the next sections, where we point out, for instance, that neither $[\texttt{nil}, \texttt{nil}] \doteq [\texttt{nil}]$ nor $[\texttt{nil}, \texttt{nil}] \not\doteq [\texttt{nil}]$ can be derived from NW. Nevertheless, this theory can be strengthened, by adding new axioms, in order to obtain a complete and decidable theory for suitable classes of sentences.

[1]When the context is clear, we identify the axioms (N) and (W) with their skolemized versions (N^l) and (W^l).

The following three axiom schemata (called freeness axioms, or Clark's equality axioms—see [5]) are usually introduced in Logic Programming and will play an important role in our axiomatization:[2]

$$
\begin{array}{lll}
(F_1) & \forall x_1 \cdots x_n y_1 \cdots y_n \left(\begin{array}{c} f(x_1,\ldots,x_n) \doteq f(y_1,\ldots,y_n) \\ \to x_1 \doteq y_1 \wedge \cdots \wedge x_n \doteq y_n \end{array} \right) & f \in \Sigma \\
(F_2) & \forall x_1 \cdots x_m y_1 \cdots y_n \quad f(x_1,\ldots,x_m) \not\doteq g(y_1,\ldots,y_n) & f \not\equiv g \\
(F_3) & \forall x \quad (x \not\doteq t[x]) &
\end{array}
$$

where $t[x]$ denotes a Σ-term having x as a proper subterm.

Axiom (F_1) holds for $[\cdot\,|\,\cdot]$ as a particular case, expressing the adaptation to lists of the classical extensionality principle. Axiom (F_3) states that there exists no term which is also a subterm of itself. Removing this axiom would allow us to accept equalities of the form $x \doteq f(x)$ whose solution requires to extend the interpretation domain so as to take into account also the so-called *rational trees*. Such an extension is considered in [6] and, for sets and multisets only, in [20, 8]. In this paper, for the sake of simplicity, we will not consider this kind of extension.

Remark 1. *Axiom (F_3) recalls the so-called* foundation axiom *(see, e.g., [14]) which has the aim, among others, of guaranteeing the* acyclicity *of membership (i.e., there are no cycles of the form $x_0 \in x_1 \in \cdots \in x_n \in x_0$). However, (F_3) does not imply foundation: consider the structure $\mathcal{D} = \langle D, I \rangle$ where:*

- *the domain D is the Herbrand Universe on the signature $\{\mathtt{nil}, \Omega, [\cdot\,|\,\cdot]\}$;*
- *I is the Herbrand interpretation on terms built from $\{\mathtt{nil}, [\cdot\,|\,\cdot]\}$; moreover,*
- *for all $x, y_1, y_2 \in D$*
  - $I(x \in \mathtt{nil}) = \mathtt{false}$*;*
  - $I(x \in [y_1\,|\,y_2]) = \mathtt{true}$ *if and only if* $I(x) = I(y_1)$ *or* $I(x \in y_2) = \mathtt{true}$*;*
  - $I(\Omega \in \Omega) = \mathtt{true}$*.*

$\mathcal{D}$ is a model of NW. However, $\mathcal{D}$ is not a model of foundation axiom, since Ω belongs to itself (such element is not the interpretation of a term). On the other hand, (F_3) excludes solutions to the equation $X \doteq [\mathtt{nil}\,|\,X]$, which are perfectly consistent with foundation.

In the following three subsections we will adapt the above axioms so as to fulfill the intended meaning of multisets, compact lists, and sets. Before proceeding further, however, we need to consider the following problem: what are the elements of an object denoted by a term $f(t_1,\ldots,t_n)$, with f a free functional symbol? This problem needs a solution also in order to establish which are the elements of a term t of the form $[r_1,\ldots,r_m\,|\,a]$, since (W) guarantees that $r_1,\ldots,r_m$ are elements of t, whereas the remaining elements of t are those of a, and a can be of the form $f(t_1,\ldots,t_n)$.

Lists, multisets, compact lists, and sets, are our 'official' constructors to build data-structures containing elements, therefore it is natural enough to forbid the insertion of elements in any other term. Hence, we will strengthen axiom (N) so as to keep *empty* any term which is not a list (a multiset, a compact list, a set) of terms. We introduce the axiom schema

$$
\begin{array}{ll}
(K) & \forall x\, y_1 \cdots y_n\, (x \notin f(y_1,\ldots,y_n)\,) \\
\multicolumn{2}{l}{\textit{for any } f \in \Sigma,\ f \textit{ distinct from } [\cdot\,|\,\cdot],\ \{\!\!\{\cdot\,|\,\cdot\}\!\!\},\ [\![\cdot\,|\,\cdot]\!],\ \{\cdot\,|\,\cdot\}}
\end{array}
$$

which generalizes (N) and will be used in its place hereinafter.[3]

As a consequence of (K) and Clark's equality axioms, all the terms of the form $f(t_1,\ldots,t_n)$, where $f \in \Sigma$, f distinct from $[\cdot\,|\,\cdot]$, $\{\!\!\{\cdot\,|\,\cdot\}\!\!\}$, $[\![\cdot\,|\,\cdot]\!]$, $\{\cdot\,|\,\cdot\}$, $ar(f) = n$, denote pairwise distinct empty sets. Following [26], we will call *urelement* any term of this form; moreover, an *urelement* will be called a *kernel* whenever it is ground. Intuitively, lists (multisets, compact lists and sets) can be seen as built by starting from a kernel (in particular,

[2]Axioms (F_1) and (F_2) were originally introduced by Mal'cev in [17].

[3]Indeed, since nil belongs to Σ, (N^l) is an instance of (K). Note also that when one of the two symbols f and g in (F_2) is the interpreted functional symbol $[\cdot\,|\,\cdot]$ ($\{\!\!\{\cdot\,|\,\cdot\}\!\!\}$, $[\![\cdot\,|\,\cdot]\!]$, $\{\cdot\,|\,\cdot\}$), then (F_2) is a theorem of KW.

from nil) and then adding to the kernel the other elements that compose the data structure. We will use the notation $\mathsf{ker}(t)$ to identify the kernel of a ground term t.

By *privileged model* of a theory T we mean a structure that is a substructure of any model of T. The privileged model for the theory described so far is the classical *Herbrand model*, where the interpretation domain is the Herbrand Universe H_Σ, and $\doteq$ is interpreted as the syntactic equality, whereas $t \in s$ is considered true if and only if s is of the form $[\cdots, t, \cdots]$. Axioms (K), (W^l), (F_1), (F_2), (F_3), along with the standard equality axioms, and the considered privileged model over H_Σ, describe the (abstract) data structure of *hybrid lists* over the alphabets Π and Σ.

Remark 2. *In the context of lists, as well as in the other axiomatic theories we will consider, objects denoted by* ground *terms are forced to have a* finite *number of elements. For the results presented in this paper we can safely ignore objects that are not denoted by terms.*

2.2 Multisets

The signature of a hybrid theory of multisets (or *bags*) must contain the binary functional symbol $\{\!\![\, \cdot \,|\, \cdot \,]\!\!\}$ in place of the previously used $[\,\cdot\,|\,\cdot\,]$, and the constant nil. By skolemization we can rewrite (W) as follows

$$(W^m) \qquad \forall y\, v\, x\, (x \in \{\!\![\, y \,|\, v \,]\!\!\} \leftrightarrow x \in v \vee x \doteq y).$$

The behavior of the interpreted functional symbol $\{\!\![\, \cdot \,|\, \cdot \,]\!\!\}$ is regulated by the following equational axiom (*permutativity property*):

$$(E_p^m) \qquad \forall xyz\, \{\!\![\, x, y \,|\, z \,]\!\!\} \doteq \{\!\![\, y, x \,|\, z \,]\!\!\}$$

which, intuitively, states that the order of elements in a multiset is immaterial. For example:

$$\{\!\![\, a, b, c, d \,]\!\!\} \quad \doteq \quad \{\!\![\, d, c, b, a \,]\!\!\}$$

can be shown to hold in $W^m E_p^m$ by noticing that

$$\begin{array}{llcll}
(1) & \{\!\![\, a, d \,]\!\!\} & \doteq & \{\!\![\, d, a \,]\!\!\} & \text{by } (E_p^m) \\
(2) & \{\!\![\, c, a, d \,]\!\!\} & \doteq & \{\!\![\, c, d, a \,]\!\!\} & \text{by (1) and equality} \\
(3) & \{\!\![\, a, c, d \,]\!\!\} & \doteq & \{\!\![\, d, c, a \,]\!\!\} & \text{by (2) and } (E_p^m) \text{ on both sides}
\end{array}$$

and that

$$(4) \quad \{\!\![\, b, c, a \,]\!\!\} \quad \doteq \quad \{\!\![\, c, b, a \,]\!\!\} \quad \text{by } (E_p^m)$$

and, hence, that

$$\begin{array}{llcll}
(5) & \{\!\![\, a, b, c, d \,]\!\!\} & \doteq & \{\!\![\, b, a, c, d \,]\!\!\} & \text{by } (E_p^m) \\
(6) & \{\!\![\, a, b, c, d \,]\!\!\} & \doteq & \{\!\![\, b, d, c, a \,]\!\!\} & \text{by (5), (3) on the r.h.s., and equality} \\
(7) & \{\!\![\, a, b, c, d \,]\!\!\} & \doteq & \{\!\![\, d, b, c, a \,]\!\!\} & \text{by (6) and } (E_p^m) \\
(8) & \{\!\![\, a, b, c, d \,]\!\!\} & \doteq & \{\!\![\, d, c, b, a \,]\!\!\} & \text{by (7), (4) on the r.h.s., and equality.}
\end{array}$$

This example makes it evident also that, contrary to lists, axiom schema (F_1) does not hold for multisets, i.e. when f is instantiated to $\{\!\![\, \cdot \,|\, \cdot \,]\!\!\}$. Since it will turn out that axiom schema (F_1) does not hold also for compact lists and sets, we modify it once and for all as follows:

$$(F_1') \qquad \forall x_1 \cdots x_n y_1 \cdots y_n \begin{pmatrix} f(x_1, \ldots, x_n) \doteq f(y_1, \ldots, y_n) \\ \to x_1 \doteq y_1 \wedge \cdots \wedge x_n \doteq y_n \end{pmatrix}$$
for any $f \in \Sigma$, f distinct from $\{\!\![\, \cdot \,|\, \cdot \,]\!\!\}$, $[\![\, \cdot \,|\, \cdot \,]\!]$, $\{\, \cdot \,|\, \cdot \,\}$.

A by-product of this modification is that, in $NW^m E_p^m$, we lack in a general principle for establishing equalities and disequalities between two multisets (the same will apply also to compact lists and sets). For instance,

$$KW^m E_p^m \not\vdash \{\!\![\, \mathtt{nil} \,]\!\!\} \neq \{\!\![\, \mathtt{nil}, \mathtt{nil} \,]\!\!\}.$$

In fact, it is easy to see that there are structures such that:

$$\mathcal{M} \models NW^m E_p^m \wedge \mathcal{M} \models \{\!\![\, \mathtt{nil} \,]\!\!\} \doteq \{\!\![\, \mathtt{nil}, \mathtt{nil} \,]\!\!\}$$

where such structures can be the naïve model of sets (formally considered in [9] and hinted at in § 2.4). This argument proves, once more, that the theory $NW^mE^m_p$ is not complete.

A viable approach to express multiset equality is to introduce a *multiset extensionality* axiom which, in a sense, replaces and weakens the classical extensionality axiom for sets stating that two sets are equal if and only if they have the same elements. Multiset extensionality ensures that two (hybrid) multisets are equal if and only if they have the same number of occurrences of each element, regardless of their order. Unfortunately, this axiom does not hold for other considered data structures, namely lists and compact lists. This is quite unsatisfactory for our purposes since we are looking for a criteria that can be easily (and incrementally) adapted to all the theories that are considered, from lists to sets. Moreover, a principle like the one previously mentioned would not be naturally adaptable to kernels, which are all "empty", and hence would force a special treatment of them.

The following simple result can help us in devising a different formulation of multiset equality which turns out to be better suited to our needs than multiset extensionality.

Lemma 3. *For all $n \in \mathbb{N}$ and for all $x, y_1, \dots, y_n$*

$$\begin{array}{lll} KW^mE^m_p & \vdash & x \in \{\!\{ y_1, \dots, y_n \}\!\} \qquad\qquad \text{if and only if} \\ KW^mE^m_p & \vdash & \exists z\,(\{\!\{ y_1, \dots, y_n \}\!\} \doteq \{\!\{ x \,|\, z \}\!\})\ . \end{array}$$

Proof. The "if" direction follows immediately from (W). We prove the "only if" direction by case analysis on $n \geq 0$. If $n = 0$ the result is trivial. Assume $n > 0$ and $x \in \{\!\{ y_1, \dots, y_n \}\!\}$. (W) ensures that $x \doteq y_i$ for some $i \in \{1, \dots, n\}$. Choose z as $\{\!\{ y_1, \dots, y_{i-1}, y_{i+1}, \dots, y_n \}\!\}$. □

Lemma 3 suggests how to express membership in terms of equality. Accordingly, we introduce the following multiset equality axiom:

$$(E^m_k) \qquad \forall y_1 y_2 v_1 v_2 \left(\begin{array}{l} \{\!\{ y_1 \,|\, v_1 \}\!\} \doteq \{\!\{ y_2 \,|\, v_2 \}\!\} \ \leftrightarrow \\ \qquad (y_1 \doteq y_2 \wedge v_1 \doteq v_2) \vee \\ \qquad \exists z\,(v_1 \doteq \{\!\{ y_2 \,|\, z \}\!\} \wedge v_2 \doteq \{\!\{ y_1 \,|\, z \}\!\}) \end{array} \right)$$

(E^m_k) captures the (intuitive) semantics of multisets. Moreover, it holds for multisets with kernels as it is.

(E^m_k) and (E^m_p) are equivalent as far as equality is concerned. On the other hand, (E^m_k) turns out to be more powerful than (E^m_p) for the general case (in particular, for establishing disequalities between bags), as proved by the following theorems.

Theorem 2. *For any formula φ*

$$KW^mE^m_p \vdash \varphi \Rightarrow KW^mE^m_k \vdash \varphi.$$

Proof. We prove this by showing that $(E^m_k) \Rightarrow (E^m_p)$. Consider the two multiset terms $\{\!\{ x, y \,|\, z \}\!\}$ and $\{\!\{ y, x \,|\, z \}\!\}$. If $x \doteq y$ they are equal by equality axioms. Assume $x \not\doteq y$; by (E^m_k) $\{\!\{ x, y \,|\, z \}\!\} \doteq \{\!\{ y, x \,|\, z \}\!\}$ if and only if $\exists k\,(\{\!\{ y \,|\, z \}\!\} \doteq \{\!\{ y \,|\, k \}\!\} \wedge \{\!\{ x \,|\, z \}\!\} \doteq \{\!\{ x \,|\, k \}\!\})$. Choosing k as 'z' the result follows. □

The reverse entailment, however, does not hold in general. One can see this fact by showing, for instance, that

$$(1) \quad KW^mE^m_k \vdash \{\!\{ \texttt{nil} \}\!\} \not\doteq \{\!\{ \texttt{nil}\,, \texttt{nil} \}\!\}$$

while we have already observed that $KW^mE^m_p \not\vdash \{\!\{ \texttt{nil} \}\!\} \not\doteq \{\!\{ \texttt{nil}\,, \texttt{nil} \}\!\}$. To prove (1), apply (E^m_p) to $\{\!\{ \texttt{nil} \}\!\} \not\doteq \{\!\{ \texttt{nil}\,, \texttt{nil} \}\!\}$:

$$\begin{array}{lll} (2) \quad KW^mE^m_k & \vdash & \{\!\{ \texttt{nil} \}\!\} \not\doteq \{\!\{ \texttt{nil}\,, \texttt{nil} \}\!\} \leftrightarrow \\ & & (\texttt{nil} \not\doteq \texttt{nil} \vee \texttt{nil} \not\doteq \{\!\{ \texttt{nil} \}\!\}) \wedge \\ & & \forall z\,(\texttt{nil} \not\doteq \{\!\{ \texttt{nil} \,|\, z \}\!\} \vee \{\!\{ \texttt{nil} \}\!\} \not\doteq \{\!\{ \texttt{nil} \,|\, z \}\!\})\ . \end{array}$$

The right-hand side of the bi-implication is true since one of the disjuncts in both conjuncts is trivially true.

Restricting to equalities, one can prove also the reverse implication between axioms (E^m_k) and (E^m_p). It is useful to begin with the following technical lemma, which will make use of the concept of *size* of a term t (namely, the number of occurrences of functional symbols in t):

$$\mathit{size}(t) \quad = \quad \begin{cases} 0 & \text{if } t \text{ is a variable} \\ 1 + \sum_{i=1}^{n} \mathit{size}(t_i) & \text{if } t \text{ is } f(t_1, \dots, t_n) \end{cases}$$

Lemma 4. *If $KW^mE^m_k \vdash \ell \doteq r$ then $size(\ell) = size(r)$.*

Proof. By induction on the number $n = size(\ell)$.

If $n = 0$, then ℓ is a variable X. Equality axioms ensure that also r must be the same variable.

Assume $n > 0$; this means that ℓ is either a term of the form $f(t_1, \dots, t_m)$, with f distinct from $\{\!\{\cdot \,|\, \cdot\}\!\}$, or a term of the form $\{\!\{t \,|\, s\}\!\}$.

- $\ell = f(t_1, \dots, t_m)$. Since f is uninterpreted, the equality axioms guarantee that also r must have the form $f(t'_1, \dots, t'_m)$, for some $t'_1, \dots, t'_m$, and, moreover, that $KW^mE^m_k \vdash t_i \doteq t'_i$ for all $i = 1, \dots, m$. By inductive hypothesis, $size(t_i) = size(t'_i)$ for all $i = 1, \dots, m$, hence $size(\ell) = size(r)$.
- if $\ell = \{\!\{t \,|\, s\}\!\}$, then by (K) and (W^m), r must be of the form $\{\!\{t' \,|\, s'\}\!\}$. By (E^m_k), one of the following two cases holds:
  - $t \doteq t'$ and $s \doteq s'$: in this case the result follows as for the uninterpreted case above;
  - there exists z such that $s \doteq \{\!\{t' \,|\, z\}\!\}$ and $s' \doteq \{\!\{t \,|\, z\}\!\}$. Since, by definition of *size*, $size(\ell) = 1 + size(t) + size(s)$, then $size(s) < n$. Hence, by inductive hypothesis, $size(s) = 1 + size(t') + size(z)$. This means that $size(\{\!\{t \,|\, z\}\!\}) < size(\ell)$. Since $KW^mE^m_k \vdash s' \doteq \{\!\{t \,|\, z\}\!\}$, by the symmetrical property of equality and inductive hypothesis, $size(s') = 1 + size(t) + size(s)$. Hence,

$$\begin{array}{rcll} size(\ell) & = & 1 + size(t) + size(s) & = \\ & & 1 + size(t) + size(t') + size(s) & = \\ & & 1 + size(t') + size(s') & = \quad size(r)\,. \end{array}$$

□

Theorem 3. $KW^mE^m_k \vdash \ell \doteq r \Rightarrow KW^mE^m_p \vdash \ell \doteq r$*, ℓ and r terms.*

Proof. By induction on the number $n = size(\ell)$.

If $n = 0$, then ℓ is a variable X. This means that also r must be the same variable, hence $KW^mE^m_p \vdash \ell \doteq r$, by equality axioms.

Assume $n > 0$; this means that ℓ is either a term of the form $f(t_1, \dots, t_m)$, with f distinct from $\{\!\{\cdot \,|\, \cdot\}\!\}$ or a term of the form $\{\!\{t \,|\, s\}\!\}$.

- $\ell = f(t_1, \dots, t_m)$, f distinct from $\{\!\{\cdot \,|\, \cdot\}\!\}$. Since f is uninterpreted, the equality axioms guarantee that also r must have the form $f(t'_1, \dots, t'_m)$, for some $t'_1, \dots, t'_m$, and, moreover, that $KW^mE^m_k \vdash t_i \doteq t'_i$ for all $i = 1, \dots, m$. By inductive hypothesis, also $KW^mE^m_p \vdash t_i \doteq t'_i$ for all $i = 1, \dots, m$, hence, again by the equality axioms, $KW^mE^m_p \vdash \ell \doteq r$.
- if $\ell = \{\!\{t \,|\, s\}\!\}$, then by (K) and (W^m), r must be of the form $\{\!\{t' \,|\, s'\}\!\}$. By (E^m_k), one of the following two cases holds:
  - $t \doteq t'$ and $s \doteq s'$: in this case the result follows as for the uninterpreted case above;
  - there exists z such that $s \doteq \{\!\{t' \,|\, z\}\!\}$ and $s' \doteq \{\!\{t \,|\, z\}\!\}$.
    By Lemma 4, z is a term such that

$$size(z) + size(t') + size(t) + 2 = size(\{\!\{t \,|\, s\}\!\})$$

Hence, both s and $\{\!\{t \,|\, z\}\!\}$ (hence s') have *size* less than n. Therefore, the inductive hypothesis ensures that $KW^mE^m_p \vdash s \doteq \{\!\{t' \,|\, z\}\!\}$ and $KW^mE^m_p \vdash s' \doteq \{\!\{t \,|\, z\}\!\}$. By equality and (E^m_p), the following equalities hold in $KW^mE^m_p$,

$$\{\!\{t \,|\, s\}\!\} \doteq \{\!\{t, t' \,|\, z\}\!\} \doteq \{\!\{t', t \,|\, z\}\!\} \doteq \{\!\{t' \,|\, z\}\!\}\,.$$

Thus, $KW^mE^m_p \vdash \ell \doteq r$.

□

A privileged model for the theory described in this section can be obtained as follows.

- As domain, choose the quotient $H_\Sigma/\equiv$ of the ordinary Herbrand Universe H_Σ on the alphabet Σ over the smallest congruence relation $\equiv$ induced by (E^m_p) on H_Σ.
- The interpretation of a term t is its equivalence class $[t]$.
- $\doteq$ is interpreted as the identity on the domain $H_\Sigma/\equiv$.
- A membership atom of the form $t \in s$ is considered true if and only if there is a term in $[s]$ of the form $\{\!\!\{ t \mid \cdot \}\!\!\}$.

Axioms (K), (W^m), (E^m_k), (F'_1), (F_2), (F_3), along with the standard equality axioms, and the considered privileged model over $H_\Sigma/\equiv$, describe the (abstract) data structure of *hybrid multisets* over the alphabets Π and Σ.

2.3 Compact lists

Let Σ be $\{\mathtt{nil}, [\![\cdot \mid \cdot]\!], \dots\}$, and the new version of (W) for compact lists is

$$(W^c) \qquad \forall y\, v\, x\, (x \in [\![\, y \mid v \,]\!] \leftrightarrow x \in v \vee x \doteq y).$$

The fundamental property of the compact list constructor $[\![\,\cdot \mid \cdot\,]\!]$ is the *absorption property*, described by the following equational axiom

$$(E^c_a) \qquad \forall xy\, [\![\, x, x \mid y \,]\!] \doteq [\![\, x \mid y \,]\!]$$

which, intuitively, states that contiguous duplicates in a compact list are immaterial. An example showing usefulness of compact lists comes from formal language theory; let $s_1, \dots, s_m, t_1, \dots, t_n$ be elements of an alphabet, then

$$\begin{array}{ll} s_1^+ \cdots s_m^+ \text{ and } t_1^+ \cdots t_n^+ & \text{are the same regular expression} \\ \text{if and only if} & [\![\, s_1, \dots, s_m \,]\!] \doteq [\![\, t_1, \dots, t_n \,]\!]\,. \end{array}$$

Similarly to multisets, also for compact lists axiom schema (F_1) does not correctly interpret the behavior of the functional symbol $[\![\cdot \mid \cdot]\!]$, as it ensues from the following example:

$$[\![\mathtt{nil} \mid [\![\mathtt{nil}]\!]\,]\!]\ (\textit{i.e.}, [\![\mathtt{nil}, \mathtt{nil}]\!]) \quad \doteq \quad [\![\mathtt{nil} \mid \mathtt{nil}]\!]\ (\textit{i.e.}, [\![\mathtt{nil}]\!])\,.$$

Therefore, as for multisets, we need a general principle for establishing equality and disequality between compact lists. Indeed, $KW^cE^c_a$ is not powerful enough to derive for instance that $[\![\,\mathtt{nil}\,, [\![\mathtt{nil}]\!]\,]\!] \not\doteq [\![\,[\![\mathtt{nil}]\!]\,, \mathtt{nil}\,]\!]$ (again, sets are a model of $KW^cE^c_a$).

As for multisets, we can introduce a new axiom which allows us to express equality between compact lists (with kernels) in terms of a disjunction of simpler equalities between compact lists:

$$(E^c_k) \qquad \forall y_1 y_2 v_1 v_2 \left(\begin{array}{l} [\![\, y_1 \mid v_1 \,]\!] \doteq [\![\, y_2 \mid v_2 \,]\!] \;\; \leftrightarrow \\ \qquad\qquad (y_1 \doteq y_2 \wedge v_1 \doteq v_2) \vee \\ \qquad\qquad (y_1 \doteq y_2 \wedge v_1 \doteq [\![\, y_2 \mid v_2 \,]\!]) \vee \\ \qquad\qquad (y_1 \doteq y_2 \wedge [\![\, y_1 \mid v_1 \,]\!] \doteq v_2) \end{array} \right)$$

The first disjunct takes care of the simplest case where neither y_1 nor y_2 occur more than once in the given compact lists. The second and third disjuncts, instead, are concerned with cases in which y_1 and y_2 occur more than once in the left-hand side term (e.g., $[\![\, a\,,\, a \,]\!] \doteq [\![\, a \,]\!]$) and in the right-hand side term, respectively.

$KW^cE^c_k$ can be proved to be at least as powerful as $KW^cE^c_a$ as done for multisets, by showing that any formula that is provable in the latter can be proved also in the former (but not vice versa).

In order to agree with the intended meaning of $[\![\cdot \mid \cdot]\!]$, also freeness axiom (F_3) must be suitably modified. Indeed, as opposed to lists, an equation such as $X \doteq [\![\,\mathtt{nil} \mid X\,]\!]$ admits a finite tree solution, namely a solution that binds x to the term $[\![\mathtt{nil} \mid t]\!]$, where t is any term.

The following lemma holds:

Lemma 5. *In any model of $KW^cE^c_k$ such that every element is denoted by a term, the following holds:*

$$\exists x\, (x \doteq [\![\, t_1, \dots, t_n \mid x \,]\!]) \quad \textit{if and only if} \quad (t_1 \doteq t_2 \doteq \cdots \doteq t_n)\,.$$

Proof. The "if" direction follows immediately with x as $[\![\, t_1 \,]\!]$.
To prove the converse, let $n \geq 2$ (if $n \leq 1$ the result is trivial). We introduce the following notation: let *can* be the (normalization) function inductively defined on ground terms as follows:

$$
can(t) \quad = \quad \begin{cases} f(can(t_1), \ldots, can(t_n)) & \text{if } t \text{ is } f(t_1, \ldots, t_n),\ f \not\equiv [\![\, \cdot \,|\, \cdot \,]\!] \\ [\![\, can(s) \,]\!] & \text{if } t \text{ is } [\![\, s \,]\!] \\ can([\![\, t_2 \,|\, s \,]\!]) & \text{if } t \text{ is } [\![\, t_1, t_2 \,|\, s \,]\!],\ can(t_1) = can(t_2) \\ [\![\, can(t_1) \,|\, can([\![\, t_2 \,|\, s \,]\!]) \,]\!] & \text{if } t \text{ is } [\![\, t_1, t_2 \,|\, s \,]\!],\ can(t_1) \neq can(t_2) \end{cases}
$$

A compact list t is said to be *canonical* if $can(t) = t$. Moreover, we define the *initial segment* relation $\sqsubseteq$:

$$
[\![\, s_1, \ldots, s_m \,]\!] \sqsubseteq t \quad \text{if and only if} \quad KW^cE^c_k \vdash \exists r\, (t \doteq [\![\, s_1, \ldots, s_m \,|\, r \,]\!])\,.
$$

As usual for lists, we define the *length* function for compact lists:

$$
length(t) \quad = \quad \begin{cases} 0 & \text{if } t \text{ is } f(t_1, \ldots, t_n),\ f \not\equiv [\![\, \cdot \,|\, \cdot \,]\!] \\ 1 + length(s) & \text{if } t \text{ is } [\![\, r \,|\, s \,]\!] \end{cases}
$$

We first prove that given two terms $[\![\, s_1, \ldots, s_m \,]\!]$ and t, if $[\![\, s_1, \ldots, s_m \,]\!] \sqsubseteq t$ and $[\![\, s_1, \ldots, s_m \,]\!]$ is canonical, then $length(t) \geq m$. We prove this fact by lexicographical induction on pairs of the form $\langle m, length(t) \rangle$.

For any term t, if $m = 0$ the result is trivial.

Assume $m > 0$ and $[\![\, s_1, \ldots, s_m \,]\!] \sqsubseteq t$.

Axioms (K) and (W^c) ensure that t cannot have the form $f(t_1, \ldots, t_n)$, f distint from $[\![\, \cdot \,|\, \cdot \,]\!]$. Hence, t is of the form $[\![\, s'_1 \,|\, t' \,]\!]$ and, by definition of $\sqsubseteq$, for some r, $[\![\, s_1, \ldots, s_m \,|\, r \,]\!] \doteq [\![\, s'_1 \,|\, t' \,]\!]$. By (E^c_k), this means that $s_1 \doteq s'_1$ and therefore

i) $[\![\, s_2, \ldots, s_m \,|\, r \,]\!] \doteq t'$, or

ii) $[\![\, s_2, \ldots, s_m \,|\, r \,]\!] \doteq [\![\, s'_1 \,|\, t' \,]\!]$, or

iii) $[\![\, s_1, \ldots, s_m \,|\, r \,]\!] \doteq t'$.

In case *i)*, the result follows by definition of *length* and by inductive hypothesis since the first element in the pair is $m - 1$. Also in the case *iii)* the result follows by inductive hypothesis. As a matter of fact, the first element in the pair is again m, while the second is $length(t) - 1$.

Case *ii)* does not apply since, being $[\![\, s_1, \ldots, s_m \,]\!]$ canonical, $can(s_1) \neq can(s_2)$, hence $s'_1 \doteq s_1 \not\doteq s_2$.

Assume now that it is not the case that $(t_1 \doteq \cdots \doteq t_n)$ $(n \geq 2)$ and assume that t is a term satisfying

$$
\exists x\, (x \doteq [\![\, t_1, \ldots, t_n \,|\, x \,]\!]).
$$

From the hypothesis we have that $can([\![\, t_1, \ldots, t_n \,]\!]) = [\![\, s_1, \ldots, s_m \,]\!]$, with $m \geq 2$, and that the following hold:

$$
\begin{gathered}
can([\![\, t_1, \ldots, t_n \,]\!]) \sqsubseteq t \\
can([\![\, t_1, \ldots, t_n, t_1, \ldots, t_n \,]\!]) \sqsubseteq t \\
can([\![\, t_1, \ldots, t_n, t_1, \ldots, t_n, t_1, \ldots, t_n \,]\!]) \sqsubseteq t \\
\vdots
\end{gathered}
$$

Since

$$
\begin{aligned}
& length([\![\, t_1, \ldots, t_n \,]\!]) \geq 2, \\
& length([\![\, t_1, \ldots, t_n, t_1, \ldots, t_n \,]\!]) \geq 4, \\
& length([\![\, t_1, \ldots, t_n, t_1, \ldots, t_n, t_1, \ldots, t_n \,]\!]) \geq 6,
\end{aligned}
$$

and so on, the term t cannot have bounded height, which is a contradiction. $\square$

Remark 3. *If we would accept solutions involving infinite compact lists, then, for any $y_1, \ldots, y_n$, the equation*

$$
x \doteq [\![\, y_1, \ldots, y_n \,|\, x \,]\!]
$$

would admit always the infinite (rational) solution

$$
x/[\![\, y_1, \ldots, y_n, y_1, \ldots, y_n, y_1, \ldots, y_n, \ldots \,]\!]\,.
$$

In view of the previous result (Lemma 5), axiom (F_3) is replaced by

(F_3^c) $\quad \forall x \quad (x \not\doteq t[x])$
unless t has the form $[\![\, t_1, \ldots, t_n \,|\, x \,]\!]$, x not occurring in $t_1, \ldots, t_n$,
and $t_1 \doteq \cdots \doteq t_n$.

A privileged model for this theory can be obtained by considering as domain the quotient $H_\Sigma / \equiv$ of the ordinary Herbrand Universe H_Σ over the smallest equivalence relation $\equiv$ on H_Σ induced by (E_a^c). As for multisets, $\doteq$ is still interpreted as the syntactic equality, whereas $t \in s$ is considered true if and only if there is a term in $[s]$ of the form $[\![\, t_1, \ldots, t_n, t \,|\, \cdot \,]\!]$. Axioms (K), (W^c), (E_k^c), (F_1'), (F_2), and (F_3^c), in addition to standard equality axioms, with the considered privileged model $H_\Sigma / \equiv$, describe the (abstract) data structure of *hybrid compact lists* over the alphabets Π and Σ.

2.4 Sets

The last theory we consider is a simple theory of sets. Σ is now required to contain nil and $\{\cdot \,|\, \cdot\}$, and (W) becomes

$$(W^s) \qquad \forall y\, v\, x\, (x \in \{\, y \,|\, v \,\} \leftrightarrow x \in v \vee x \doteq y).$$

Sets have both the *permutativity* and the *absorption properties* which, in the case of the set constructor $\{\cdot \,|\, \cdot\}$, can be rewritten as follows:

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

A criterion for testing equality between sets (with kernels) can be easily obtained by merging the multiset equality axiom (E_k^m) and the compact list equality axiom (E_k^c) of previous sections, suitably adapted to sets:

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

According to (E_k^s) both duplicates and ordering of elements in sets are immaterial.

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

(F_3^s) $\quad \forall x \quad (x \not\doteq t[x])$
unless t has the form $\{t_1, \ldots, t_n \,|\, x\}$, x not occurring in $t_1, \ldots, t_n$.

A privileged model for this theory can be obtained by considering as domain the quotient $H_\Sigma / \equiv$ of the ordinary Herbrand Universe H_Σ over the smallest equivalence relation $\equiv$ on H_Σ induced by (E_a^s) and (E_p^s). As for multisets and compact lists, $\doteq$ is still interpreted as the syntactic equality, whereas $t \in s$ is considered true if and only if there is a term in $[s]$ of the form $\{t \,|\, \cdot\}$. The hybrid theory of sets (with $\Pi = \{\doteq, \in\}$, and $\Sigma = \{\texttt{nil}, \{\cdot \,|\, \cdot\}, \ldots\}$) is therefore identified by axioms (K), (W^s), (E_k^s), (F_1'), (F_2), and (F_3^s), along with standard equality axioms and the privileged model over $H_\Sigma / \equiv$.

Remark 4. *Notice that in every model of each of the considered theories in which all elements are denoted by terms (hence finite—cf. Remark 2) membership can form neither cycles nor infinite descending chains (cycles imply infinite descending chains, but the converse is not true). To this extent, the above theories can be considered well-founded.*

Differently from both multisets and compact lists, one can prove that, for ground terms, (E_k^s) and $(E_a^s E_p^s)$, as well as the classical extensionality axiom (E), extended with kernels when $\Sigma \supset \{\texttt{nil}, \{\cdot \,|\, \cdot\}\,\}$

$$(E_k) \qquad x \doteq y \leftrightarrow \mathsf{ker}(x) \doteq \mathsf{ker}(y) \wedge \forall z\, (z \in x \leftrightarrow z \in y)\,,$$

are all equivalent criteria for testing set equality. We will first prove this fact in the pure case (when $\Sigma = \{\texttt{nil}, \{\cdot \,|\, \cdot\}\,\}$); later on we will enforce the result in a hybrid context.

Lemma 6. *Let $\Sigma = \{\mathtt{nil}, \{\cdot\,|\,\cdot\}\ \}$; then for any term r and any ground term t, there exists t_1 such that:*

$$NW^sE^s_pE^s_a \vdash (r \in t \to t \doteq \{r\,|\,t_1\} \wedge r \notin t_1).$$

Proof. By induction on the number $h = \mathit{size}(t)$.

If $h = 1$, then $t = \mathtt{nil}$ and the result holds trivially.

If $h > 1$, then $t = \{s\,|\,\bar{t}\}$. If $r \notin \bar{t}$, it must be the case that $s \doteq r$. Otherwise, by inductive hypothesis, there exists $\bar{t}_1$ such that $r \notin \bar{t}_1$ and $t \doteq \{s\,|\,\{r\,|\,\bar{t}_1\}\}$, which, by (E^s_p), is equal to $\{r\,|\,\{s\,|\,\bar{t}_1\}\}$.

If $r \not\doteq s$, then put $t_1 \doteq \{s\,|\,\bar{t}_1\}$. Otherwise, by (E^s_a) we have that $t \doteq \{r\,|\,\bar{t}_1\}$, and hence we can take $t_1 = \bar{t}_1$. □

We will make use of the concept of *length* of a set term (for the definition of that concept, see § 2.3, where it has been defined for compact lists).

Theorem 4. *Let $\Sigma = \{\mathtt{nil}, \{\cdot\,|\,\cdot\}\ \}$; assuming (K) and (W^s), for ground terms the following axioms are equivalent:*

(i)	(E) ,
(ii)	(E^s_k) ,
(iii)	$(E^s_p) \wedge (E^s_a)$.

Proof. **(i) ⇒ (ii)** To see that $(E) \Rightarrow (E^s_k)$ assume that $\{y_1\,|\,v_1\} \doteq \{y_2\,|\,v_2\}$.

The following four cases are possible:

1. $y_1 \notin v_1$ and $y_2 \notin v_2$;
2. $y_1 \notin v_1$ and $y_2 \in v_2$;
3. $y_1 \in v_1$ and $y_2 \notin v_2$;
4. $y_1 \in v_1$ and $y_2 \in v_2$.

In case 1, $y_1 \doteq y_2$ implies (by (E)) that $v_1 \doteq v_2$. This case is covered by the first disjunct of (E^s_k). If $y_1 \not\doteq y_2$, then (always by (E)) $y_1 \in v_2$ and $y_2 \in v_1$. By Lemma 6 it holds that $v_2 \doteq \{y_1\,|\,v'_2\}$ and $v_1 \doteq \{y_2\,|\,v'_1\}$. By (E), $v'_1 \doteq v'_2$: they are the existentially quantified variable k of the fourth disjunct of (E^s_k).

In case 2, if $y_1 \not\doteq y_2$ it must be that $y_2 \in v_1$ and, by (E), $v_1 \doteq \{y_2\,|\,v_1\} \doteq v_2$: again the fourth disjunct of (E^s_k) holds. If $y_1 \doteq y_2$, then v_2 and $\{y_1\,|\,v_1\}$ must have the same elements and therefore $v_2 \doteq \{y_1\,|\,v_1\}$ and the second disjunct of (E^s_k) holds.

Case 3 is entirely symmetric to case 2.

In case 4, v_1 and v_2 must have the same elements and therefore, by (E), $v_1 \doteq v_2 \doteq \{y_1\,|\,v_1\}$ and the last disjunct of (E^s_k) holds.

(ii) ⇒ (iii) Immediate.

(iii) ⇒ (i) Let t_1 and t_2 be ground terms, we must prove that

$$NW^sE^s_aE^s_p \vdash \forall x\,(x \in t_1 \leftrightarrow x \in t_2) \to t_1 \doteq t_2\,.$$

We will prove this fact by induction on the maximum h between the lengths of t_1 and t_2; we prove that if $t_1 \not\doteq t_2$, then there exists t such that $t \in t_1 \leftrightarrow t \notin t_2$.

If $h = 0$ the result is trivially true since $t_1 = t_2 = \mathtt{nil}$.

If $h > 0$, then assume, without loss of generality, that t_1 is the term of maximum length. By Lemma 6, let $t_1 \doteq \{t\,|\,t'_1\}$ and $t \notin t'_1$. If $t \notin t_2$ we have concluded, otherwise, again by Lemma 6, let $t_2 \doteq \{t\,|\,t'_2\}$ and $t \notin t'_2$.

If t'_1 and t'_2 were equal, then t_1 and t_2 would be equal and therefore it must be that $t'_1 \not\doteq t'_2$. At this point the thesis follows by the inductive hypothesis.

□

Now, we move to the hybrid case. The kernel $\mathsf{ker}(s)$ of a ground set term s is obtained by decomposing s in the form $\{t_1 \,|\, \{t_2 \,|\, \ldots \{t_n \,|\, k\} \ldots\}\}$, $n \geq 0$, where the main functor of k differs from $\{\cdot \,|\, \cdot\}$.

Theorem 5. *Assuming (K), (W^s), (F_1'), and (F_2), for ground terms, the following axioms are all equivalent*

$$\begin{array}{ll} \textbf{(i)} & (E_k)\,, \\ \textbf{(ii)} & (E_k^s)\,, \\ \textbf{(iii)} & (E_1^s) \wedge (E_2^s)\,. \end{array}$$

Proof. The only difference with respect to the previous Theorem 4 is in the proof **(iii)** $\Rightarrow$ **(i)**. We need to prove that for any t_1 and t_2 ground terms,

$$NW^sE_a^sE_p^s \vdash (\forall x\,(x \in t_1 \leftrightarrow x \in t_2) \wedge \mathsf{ker}(t_1) \doteq \mathsf{ker}(t_2)) \rightarrow t_1 \doteq t_2\,.$$

We will prove this fact by induction on the maximum h between the lengths of t_1 and t_2; we prove that if $t_1 \not\doteq t_2$, then there exists t such that $t \in t_1 \leftrightarrow t \notin t_2$ or $\mathsf{ker}(t_1) \not\doteq \mathsf{ker}(t_2)$.

If $h = 0$ the result is trivially true since $t_1 = \mathsf{ker}(t_1) \not\doteq \mathsf{ker}(t_2) = t_2$.

The case with $h > 0$ is the same as in the previous theorem. □

Hence, (E_k^s), $(E_a^s) \wedge (E_p^s)$ and (E_k) are equivalent criteria for testing set equality. Using (E_k^s), instead of the other two, is motivated by the desire to have an incrementally developed theory which can include as sub-cases the theories for the other considered data structures. Indeed, axioms (E_k^s), (E_k^m) and (E_k^c), as well as the different versions of the (W) axiom and of the freeness axioms, can easily be combined in order to obtain axiomatic theories capable to deal with any subset of the collection of proposed data structures (namely, lists, multisets, compact lists, and sets).

The usage of (E_k^s) (as well as of (E_k^m) and (E_k^c)) is further motivated by the observation that the selected equality criteria can be easily converted into effective procedures for testing equality to be used in the relevant unification algorithms, as shown in the next section.

It will turn out, also, that these algorithms could easily be merged to solve the unification problem relative to the "combined" theories mentioned above.

3 Unification of multisets, compact lists, and sets

We start by proving NP-hardness of the unification problems for multisets, compact lists, and sets. Then, before presenting the proposed unification algorithms for all these data structures, we briefly recall the unification algorithm for hybrid lists, that is basically the standard non deterministic unification algorithm explicitly applying substitutions and studied in [15, 1].[4] This will allow us to introduce the style and the notation we will also employ for the subsequent algorithms. Moreover, all actions of the standard algorithm remain almost unchanged in all the other cases.

3.1 NP-hardness

While unification for (hybrid) lists can be performed in linear time (cf. [22, 18]), the unification problems for multisets, compact lists, and sets are NP-complete (see [13] for an analysis of the complexity of set unification).

NP-hardness of these problems is proved in this section via reduction of the 3-SAT problem to the unification problems at hand.

Let us consider the following instance of 3-SAT:

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

First, we consider set unification. The above instance of 3-SAT can be re-written as the following set unification problem:

- $\{\ \ \{X_1, Y_1\}, \{X_2, Y_2\}, \{X_3, Y_3\},$
  $\{\underline{0}, X_1, X_2, Y_3\}, \{\underline{0}, Y_1, X_2, X_3\}, \{\underline{0}, X_1, Y_2, X_3\}\ \} \doteq$
  $\{\ \ \{\underline{0}, \underline{1}\}\ \}$

[4]The basic idea of the algorithm can be traced back to Herbrand ([11]) and can also be found in [18] as starting point for the almost linear algorithm developed in that paper.

```
Unify_lists($\mathcal{E}$):
while $\mathcal{E}$ is not in solved form apply any of the following actions
    ($l1$)  $X \doteq X \wedge \mathcal{E} \quad \mapsto \quad \mathcal{E}$
    ($l2$)  $\left.\begin{array}{r} t \doteq X \wedge \mathcal{E} \\ t \text{ is not a variable} \end{array}\right\} \quad \mapsto \quad X \doteq t \wedge \mathcal{E}$
    ($l3$)  $\left.\begin{array}{r} X \doteq t \wedge \mathcal{E} \\ X \text{ does not occur in } t \\ X \text{ occurs in } \mathcal{E} \end{array}\right\} \quad \mapsto \quad \mathcal{E}[X/t] \wedge X \doteq t$
    ($l4$)  $\left.\begin{array}{r} X \doteq t \wedge \mathcal{E} \\ X \not\equiv t \text{ and } X \text{ occurs in } t \end{array}\right\} \quad \mapsto \quad \texttt{fail}$
    ($l5$)  $\left.\begin{array}{r} f(s_1,\dots,s_m) \doteq g(t_1,\dots,t_n) \wedge \mathcal{E} \\ f \not\equiv g \end{array}\right\} \quad \mapsto \quad \texttt{fail}$
    ($l6$)  $f(s_1,\dots,s_m) \doteq f(t_1,\dots,t_m) \wedge \mathcal{E} \quad \mapsto$
                $s_1 \doteq t_1 \wedge \cdots \wedge s_m \doteq t_m \wedge \mathcal{E}.$
```

Figure 1: Standard unification algorithm

where $\underline{0}$ and $\underline{1}$ can be viewed either as two constant symbols or as two distinct set terms (e.g., $\underline{0} = \texttt{nil}$, $\underline{1} = \{\texttt{nil}\}$). It is immediate to prove that any substitution fulfilling the set-set equation 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.

Similarly, the same instance of 3-SAT can be re-written using multisets and compact lists:

- $$\begin{array}{ll} \{\!\!\{ & \{\!\!\{ X_1, Y_1 \}\!\!\}, \{\!\!\{ X_2, Y_2 \}\!\!\}, \{\!\!\{ X_3, Y_3 \}\!\!\}, \\ & \{\!\!\{ X_1, X_2, Y_3 \}\!\!\}, \{\!\!\{ Y_1, X_2, X_3 \}\!\!\}, \{\!\!\{ X_1, Y_2, X_3 \}\!\!\} \ \}\!\!\} \doteq \\ \{\!\!\{ & \{\!\!\{ \underline{0}, \underline{1} \}\!\!\}, \{\!\!\{ \underline{0}, \underline{1} \}\!\!\}, \{\!\!\{ \underline{0}, \underline{1} \}\!\!\}, \\ & \{\!\!\{ \underline{1}, A_1, B_1 \}\!\!\}, \{\!\!\{ \underline{1}, A_2, B_2 \}\!\!\}, \{\!\!\{ \underline{1}, A_3, B_3 \}\!\!\} \ \}\!\!\} \end{array}$$

- $$\begin{array}{ll} \llbracket & \llbracket \underline{0}, X_1, \underline{0}, Y_1, \underline{0} \rrbracket, \llbracket \underline{0}, X_2, \underline{0}, Y_2, \underline{0} \rrbracket, \llbracket \underline{0}, X_3, \underline{0}, Y_3, \underline{0} \rrbracket, \\ & \llbracket \underline{1}, \underline{0}, X_1, X_2, Y_3 \rrbracket, \llbracket \underline{0}, Y_1, X_2, X_3 \rrbracket, \llbracket \underline{1}, \underline{0}, X_1, Y_2, X_3 \rrbracket \ \rrbracket \doteq \\ \llbracket & \llbracket \underline{0}, \underline{1}, \underline{0} \rrbracket, \\ & \llbracket \underline{1}, \underline{0}, \underline{1}, A_1, B_1 \rrbracket, \llbracket \underline{0}, \underline{1}, A_2, B_2 \rrbracket, \llbracket \underline{1}, \underline{0}, \underline{1}, A_3, B_3 \rrbracket \ \rrbracket \ . \end{array}$$

In the first equation, multiple occurrences of the same multiset $\{\!\!\{ \underline{0}, \underline{1} \}\!\!\}$ are needed to capture the semantics of the $\{\!\!\{ \cdot \mid \cdot \}\!\!\}$ constructor. For the same reason, the new variables A_is, B_is are introduced.

For the compact list case—second equation—observe that

$$\llbracket \underline{0}, X, \underline{0}, Y, \underline{0} \rrbracket \doteq \llbracket \underline{0}, \underline{1}, \underline{0} \rrbracket$$

admits only two solutions: $[X/\underline{0}, Y/\underline{1}]$ and $[X/\underline{1}, Y/\underline{0}]$. Moreover, the alternation of the first element of the compact lists

$$\llbracket \underline{1}, \underline{0}, \cdots \rrbracket, \llbracket \underline{0}, \cdots \rrbracket, \llbracket \underline{1}, \underline{0}, \cdots \rrbracket, \ldots$$

ensures the correct matching between the remaining compact lists.

The same technique used for rewriting the considered instance of 3-SAT as unification problems can be applied to any instance of the 3-SAT problem. This proves the NP-hardness of the problems at hand.

3.2 Standard unification

Unification of (hybrid) lists is performed by the (standard) unification algorithm Unify_lists shown in Figure 1.

Here we assume all the definitions usually employed for standard unification (cf., e.g., [15, 18]). In particular, an equation $X \doteq t$ of a system of equations $\mathcal{E}$ is said to be in *solved form* with respect to $\mathcal{E}$ if X is a variable which does not occur elsewhere in $\mathcal{E}$ (X is said to be an *eliminable* variable). When all the equations of a system $\mathcal{E}$ are in solved form with respect to $\mathcal{E}$, $\mathcal{E}$ itself is said to be in solved form. (i.e., $\mathcal{E}$ has the form $X_1 \doteq t_1 \wedge \cdots \wedge X_n \doteq t_n$ where the X_is are distinct variables not occurring in right-hand side terms of any equation of $\mathcal{E}$).

Unify_lists terminates on any input system of equations $\mathcal{E}$, returning an equivalent system in solved form from which it is immediate to obtain the (unique) *most general unifier* for the given unification problem.

Theorem 6 (Termination)**.** $\mathsf{Unify\_lists}$ *terminates for any input system* $\mathcal{E}$*.*

Proof. Let $A_\mathcal{E}$ be the number of non-eliminable variables of $\mathcal{E}$. Let $B_\mathcal{E}$ be $\sum_{(\ell \doteq r)\ in\ \mathcal{E}} size(\ell)$.[5] Let $C_\mathcal{E}$ be the number of equations of $\mathcal{E}$.

The triple $\mathcal{C}(\mathcal{E}) \equiv \langle A_\mathcal{E}, B_\mathcal{E}, C_\mathcal{E} \rangle$ will be used as the complexity measure for proving the algorithm termination. We show that for each non-failing action of $\mathsf{Unify\_lists}$ the algorithm decreases the value of $\mathcal{C}(\mathcal{E})$ with respect to the lexicographical ordering. Since the lexicographical ordering on tuples of non-negative integers is a well-ordering, this is sufficient to prove the termination of the algorithm.

Let the number on the left indicate the action analyzed:

($l1$)**:** $A_\mathcal{E}$ and $B_\mathcal{E}$ can not increase. $C_\mathcal{E}$ decreases.

($l2$)**:** $A_\mathcal{E}$ can not increase. $B_\mathcal{E}$ decreases.

($l3$)**:** $A_\mathcal{E}$ decreases by 1.

($l6$)**:** $A_\mathcal{E}$ can not increase. $B_\mathcal{E}$ decreases.

□

Moreover, correctness and completeness of the algorithm can be proved with respect to the corresponding theory presented in § 2.1 (cf., for instance, [1, 16, 15]).

Theorem 7 (Soundness and Completeness)**.** *Let* $e \wedge \mathcal{E}$ *be an equation system,* e *an equation not in solved form, and* $\mathcal{E}_1$ *be the equation system resulting from the application of the action of* $\mathsf{Unify\_lists}$ *fired by* e*. Then* $KW^l F_1 F_2 F_3 \vdash e \wedge \mathcal{E} \leftrightarrow \mathcal{E}_1$*.*

Proof. The proof can be performed by case analysis, proving the claim for each single action. Actions ($l1$), ($l2$), and ($l3$) are justified by equality axioms. Action ($l4$) by axiom schema (F_3) while ($l5$) and ($l6$) by equality axioms and freeness axiom schemata (F_1) and (F_2). □

Action ($l4$) performs the so-called *occur-check*: it checks the well-foundedness of the unique most general solution of the system.

In the algorithms for multisets, compact lists, and sets, action ($l6$), when f is the corresponding interpreted functional symbol, will be integrated by non-deterministic actions which reflect axioms (E^m_k), (E^c_k), and (E^s_k), respectively.

3.3 Unification of hybrid multisets

The first five actions of the unification algorithm for hybrid multisets are exactly the same as those of the algorithm for hybrid lists. In addition, we need to restrict applicability of action ($l6$) to non-multiset terms only (action ($m6$)), and to introduce a new action to deal with bag-bag equations (action ($m7$)), which closely corresponds to axiom (E^m_k). This new action introduces a source of don't know non-determinism: both alternatives (i) and (ii) of ($m7$) must be exploited in order to guarantee completeness (note that N is a newly generated variable not occurring in $\mathcal{E}$, yet).

$$
\begin{array}{lll}
(m1)\text{–}(m5) & \text{as } (l1)\text{–}(l5) \text{ of } \mathsf{Unify\_lists} & \\
(m6) & \left.\begin{array}{l} f(s_1,\dots,s_m) \doteq f(t_1,\dots,t_m) \wedge \mathcal{E} \\ f \in \Sigma \setminus \{\{\!\!\{\cdot\,|\,\cdot\}\!\!\}\} \end{array}\right\} & \mapsto \\
 & \qquad\qquad s_1 \doteq t_1 \wedge \cdots \wedge s_m \doteq t_m \wedge \mathcal{E} & \\
(m7) & \{\!\!\{ t \,|\, s \}\!\!\} \doteq \{\!\!\{ t' \,|\, s' \}\!\!\} \wedge \mathcal{E} & \mapsto \\
 & \quad (i) \quad t \doteq t' \wedge s \doteq s' \wedge \mathcal{E} & \\
 & \quad (ii) \quad s \doteq \{\!\!\{ t' \,|\, N \}\!\!\} \wedge \{\!\!\{ t \,|\, N \}\!\!\} \doteq s' \wedge \mathcal{E} &
\end{array}
$$

Although sound and complete, the above algorithm does not always terminate. As an example, consider the following input system (X and N variables):

$$
\begin{array}{ll}
\{\!\!\{ t \,|\, X \}\!\!\} \doteq \{\!\!\{ t' \,|\, X \}\!\!\} & \overset{m7(ii)}{\mapsto} \\
X \doteq \{\!\!\{ t' \,|\, N \}\!\!\} \wedge \{\!\!\{ t \,|\, N \}\!\!\} \doteq X & \overset{m3}{\mapsto} \\
X \doteq \{\!\!\{ t' \,|\, N \}\!\!\} \wedge \{\!\!\{ t \,|\, N \}\!\!\} \doteq \{\!\!\{ t' \,|\, N \}\!\!\}\,. &
\end{array}
$$

[5]See § 2.2 for the definition of *size*.

```
Unify_bags($\mathcal{E}$):
$\mathcal{E}_2 := \emptyset$;
while $\mathcal{E}_1$ is not in solved form or $\mathcal{E}_2 \neq \emptyset$ do
    while $\mathcal{E}_2 \neq \emptyset$ do
        $e :=$ pop($\mathcal{E}_2$);
        if $e$ is in solved form w.r.t. $\mathcal{E}_1 \wedge \mathcal{E}_2$
        then $\mathcal{E}_1 := \mathcal{E}_1 \wedge e$
        else Unify_bag_actions($\mathcal{E}$, $e$);
    select arbitrarily and remove from $\mathcal{E}_1$ an equation $e$ not in solved form;
    Unify_bag_actions($\mathcal{E}$, $e$).
```

Figure 2: Multiset unification algorithm

The last system is at least as difficult as the starting one.

More generally, for any input system of the form

$$\begin{array}{ccll} \{\!\!\{ \cdots \mid X_1 \}\!\!\} & \doteq & \{\!\!\{ \cdots \mid X_2 \}\!\!\} & \wedge \\ \{\!\!\{ \cdots \mid X_2 \}\!\!\} & \doteq & \{\!\!\{ \cdots \mid X_3 \}\!\!\} & \wedge \\ & \vdots & & \wedge \\ \{\!\!\{ \cdots \mid X_n \}\!\!\} & \doteq & \{\!\!\{ \cdots \mid X_1 \}\!\!\}\,, & \end{array}$$

$X_1, \ldots, X_n$ variables, it is easy to find a non-deterministic sequence of actions leading to non-termination of the multiset unification algorithm. An intuitive reason for non-termination over the equation $\{\!\!\{ t \mid X \}\!\!\} \doteq \{\!\!\{ t' \mid X \}\!\!\}$ is that an infinite number of possibilities, corresponding to subsequent increments of X by t and t', is generated by the unification algorithm. Clearly, if t and t' are not unifiable, there is no solution using finite bags (disregarding the value of X). If, on the contrary, t and t' are unifiable, then there is no need to find any binding for X.

Situations like the one descrobed by the equation $\{\!\!\{ t \mid X \}\!\!\} \doteq \{\!\!\{ t' \mid X \}\!\!\}$ can be easily handled as special. Let tail and de_tail be the following functions:[6]

$$\begin{array}{lcll} \mathsf{tail}(f(t_1, \ldots, t_n)) & = & f(t_1, \ldots, t_n) & f \text{ distinct from } \{\!\!\{ \cdot \mid \cdot \}\!\!\} \\ \mathsf{tail}(X) & = & X & \\ \mathsf{tail}(\{\!\!\{ t \mid s \}\!\!\}) & = & \mathsf{tail}(s) & \\ \mathsf{de\_tail}(X) & = & \mathtt{nil} & \\ \mathsf{de\_tail}(\{\!\!\{ t \mid s \}\!\!\}) & = & \{\!\!\{ t \mid \mathsf{de\_tail}(s) \}\!\!\}\,. & \end{array}$$

Action $(m7)$ can be split into two sub-actions. If $\mathsf{tail}(s)$ and $\mathsf{tail}(s')$ are not the same variable then perform action $(m7)$. Otherwise, replace $\{\!\!\{ t \mid s \}\!\!\} \doteq \{\!\!\{ t' \mid s' \}\!\!\}$ with $\mathsf{de\_tail}(\{\!\!\{ t \mid s \}\!\!\}) \doteq \mathsf{de\_tail}(\{\!\!\{ t' \mid s' \}\!\!\})$. The second sub-action is justified by the fact that two multisets are equal if and only if they contain the same number of occurrences of each element (as it turns out from multiset extensionality—cf. § 2.2).

However, to solve the problem also in the more general case we find it convenient to split the system $\mathcal{E}$ into two parts, $\mathcal{E}_1$ and $\mathcal{E}_2$, the second of which is dealt with as a stack (with pop and push operations for element removal and insertion, respectively). In this way, we can ensure enough determinism in the algorithm to guarantee its termination in every cases. The deterministic part of the multiset unification algorithm is shown in Figure 2 (it will turn out that this part is basically the same for all the considered unification algorithms). The (non-deterministic) actions of the agorithm are shown in Figure 3.

As an example, we can see that one of the critical situations pointed out above can be dealt with correctly by this algorithm:

$$\begin{array}{l} \{\!\!\{ T_1 \mid S_1 \}\!\!\} \doteq \{\!\!\{ T_2 \mid S_2 \}\!\!\} \wedge \{\!\!\{ T_3 \mid S_2 \}\!\!\} \doteq \{\!\!\{ T_4 \mid S_1 \}\!\!\} \overset{m7(ii)}{\mapsto} \\ S_1 \doteq \{\!\!\{ T_2 \mid N_1 \}\!\!\} \wedge \{\!\!\{ T_1 \mid N_1 \}\!\!\} \doteq S_2 \wedge \{\!\!\{ T_3 \mid S_2 \}\!\!\} \doteq \{\!\!\{ T_4 \mid S_1 \}\!\!\} \overset{m2-m3-m3}{\mapsto} \\ S_1 \doteq \{\!\!\{ T_2 \mid N_1 \}\!\!\} \wedge S_2 \doteq \{\!\!\{ T_1 \mid N_1 \}\!\!\} \wedge \{\!\!\{ T_3, T_1 \mid N_1 \}\!\!\} \doteq \{\!\!\{ T_4, T_2 \mid N_1 \}\!\!\}\,. \end{array}$$

The last equation will be replaced by $\{\!\!\{ T_3, T_1 \}\!\!\} \doteq \{\!\!\{ T_4, T_2 \}\!\!\}$, avoiding the loop.

[6]In § 3.5 we will use the function tail for sets, assuming $\{\!\!\{ \cdot \mid \cdot \}\!\!\}$ is replaced by $\{ \cdot \mid \cdot \}$ in the definition.

```
Unify_bag_actions($\mathcal{E}$,$e$):
  case $e$ of
  $(m1)$   $X \doteq X$   $\mapsto$   $\mathcal{E}_1 := \mathcal{E}_1$
  $(m2)$   $t \doteq X$
           $t$ is not a variable   $\mapsto$   push$(X \doteq t, \mathcal{E}_2)$
  $(m3)$   $X \doteq t$
           $X$ does not occur in $t$
           $X$ occurs in $\mathcal{E}$   $\mapsto$
               $\mathcal{E}_1 := (X \doteq t) \wedge \mathcal{E}_1[X/t]; \mathcal{E}_2 := \mathcal{E}_2[X/t]$
  $(m4)$   $X \doteq t$
           $X \not\equiv t$ and $X$ occurs in $t$   $\mapsto$   fail
  $(m5)$   $f(s_1, \dots, s_m) \doteq g(t_1, \dots, t_n)$
           $f \not\equiv g$   $\mapsto$   fail
  $(m6)$   $f(s_1, \dots, s_m) \doteq f(t_1, \dots, t_m)$
           $f \not\equiv \{\!\![ \cdot \mid \cdot ]\!\!\}$   $\mapsto$
               $\mathcal{E}_1 := (s_1 \doteq t_1 \wedge \dots \wedge s_m \doteq t_m) \wedge \mathcal{E}_1$
  $(m7)$   $\{\!\![ t \mid s ]\!\!\} \doteq \{\!\![ t' \mid s' ]\!\!\}$
           tail$(s)$ and tail$(s')$
           are not the same variable   $\mapsto$
               $(i)$   $\mathcal{E}_1 := (t \doteq t') \wedge \mathcal{E}_1$; push$(s \doteq s', \mathcal{E}_2)$
               $(ii)$  push$(s \doteq \{\!\![ t' \mid N ]\!\!\}, \mathcal{E}_2)$; push$(\{\!\![ t \mid N ]\!\!\} \doteq s', \mathcal{E}_2)$
  $(m8)$   $\{\!\![ t \mid s ]\!\!\} \doteq \{\!\![ t' \mid s' ]\!\!\}$
           tail$(s)$ and tail$(s')$
           are the same variable   $\mapsto$
               push(de_tail$(\{\!\![ t \mid s ]\!\!\})$ $\doteq$ de_tail$(\{\!\![ t' \mid s' ]\!\!\})$ , $\mathcal{E}_2)$ .
```

Figure 3: Multiset unification rewriting rules

Theorem 8 (Termination). Unify_bags *terminates for any input system $\mathcal{E}$.*

Proof. Let $A_\mathcal{E}$ be the number of non-eliminable variables of $\mathcal{E}$. Let $B_\mathcal{E}$ be $\sum_{(\ell \doteq r) \; in \; \mathcal{E}} size(\ell)$.[7] Let $C_\mathcal{E}$ be the number of equations of $\mathcal{E}$.

The triple $\mathcal{C}(\mathcal{E}) \equiv \langle A_\mathcal{E}, B_\mathcal{E}, C_\mathcal{E} \rangle$ will be used as the complexity measure for proving the algorithm termination.

Let us call a *phase* of Unify_bags the finite sequence of actions starting with one of the actions of Unify_bags_actions and followed by all (possibly none) consecutive actions performed in the inner while loop of Unify_bags applied to all the equations contained in $\mathcal{E}_2$ (until $\mathcal{E}_2$ becomes empty).

Next, we analize the behavior of the triple $\mathcal{C}(\mathcal{E})$ for each action of Unify_bags. It will turn out that, for each different action, the non-failing phase of the algorithm starting with this action decreases the value of $\mathcal{C}(\mathcal{E})$ with respect to the lexicographical ordering. Since the lexicographical ordering on tuples of non-negative integers is a well-ordering, this is sufficient to prove the termination of the algorithm.

Let the number on the left indicate the first action of a phase:

$(m1)$**:** $A_\mathcal{E}$ may decrease; $B_\mathcal{E}$ is left unchanged; $C_\mathcal{E}$ decreases.

$(m2)$**:** $A_\mathcal{E}$ can not increase; $B_\mathcal{E}$ decreases.

$(m3)$**:** $A_\mathcal{E}$ decreases by 1.

$(m6)$**:** $A_\mathcal{E}$ may decrease (note that, for instance, X is non-eliminable in $f(X) \doteq f(t)$, whereas X may turn out to be eliminable in $X \doteq t$); $B_\mathcal{E}$ decreases.

$(m7)(i)$**:** $A_\mathcal{E}$ may decrease; $B_\mathcal{E}$ decreases;

$(m7)(ii)$**:** $A_\mathcal{E}$ may initially increase (by 1) since N is a new non-eliminable variable. However, let us analyze the behaviour of the algorithm at the end of the phase fired by the processing of an equation of the form $\{\!\![s_1, \dots, s_m \mid S]\!\!\} \doteq \{\!\![t_1, \dots, t_n \mid S']\!\!\}$.

[7]See § 2.2 for the definition of *size*.

Due to the assumption on the algorithm excution strategy, a sequence of actions ($m7$) ((i) and (ii)) is repeatedly applied starting from the given equation. At the end of this sequence, the algorithm has returned a conjunction of equations of the form $s_{i_1} \doteq t_{j_1}, \ldots, s_{i_k} \doteq t_{j_k}$, $i_1, \ldots, i_k \in \{0, \ldots, m\}$, $j_1, \ldots, i_k \in \{0, \ldots, n\}$, along with a finite number of further equations, depending on the form of S and S':

(a) if both S and S' are non-variable terms and if it is the case that $S = f(s_1, \ldots, s_n)$ and $S' = f(s'_1, \ldots, s'_n)$, f distinct from $\{\!\{ \cdot \,|\, \cdot \}\!\}$, then the further equations are $s_1 \doteq s'_1, \ldots, s_n \doteq s'_n$. In any other case, namely when $S = f(\cdots)$ and $S' = g(\cdots)$, f distinct from g (one of the two possibly $\{\!\{ \cdot \,|\, \cdot \}\!\}$), then action ($m5$) would detect an immediate failure;

(b) if at least one between S and S' is a variable, then the returned equations may have one of the following forms:

- $S \doteq \{\!\{ t_{j_{k+1}}, \ldots, t_{j_n} \,|\, N \}\!\}$,
  $S' \doteq \{\!\{ s_{i_{k+1}}, \ldots, s_{i_m} \,|\, N \}\!\}$, $m, n \geq k$, S and S' distinct variables, or
- $S \doteq \{\!\{ t_{j_{k+1}}, \ldots, t_{j_n} \,|\, S' \}\!\}$, $n \geq k$, S' variable or $f(\cdots)$, or
- $S' \doteq \{\!\{ s_{i_{k+1}}, \ldots, s_{i_m} \,|\, S \}\!\}$, $m \geq k$, S' variable or $f(\cdots)$.

After completing the processing of the given bag-bag equation, the complexity measure $\mathcal{C}(\mathcal{E})$ has changed as follows:

- case (a): $A_\mathcal{E}$ unchanged; $B_\mathcal{E}$ decreases (like in ($m6$));
- case (b): after applying the substitutions for the variables S and S', even if the new (non-eliminable) variable N is introduced into the system, S and/or S' become eliminable, so that $A_\mathcal{E}$ decreases by 1.

Note that if S and S' were both the same variable, the second and third case of (b) would fail (due to occur-check), whereas the first case of (b) still applies. Applying substitutions for S and S' leads $A_\mathcal{E}$ to decrease just by 1, so that, at the end, $A_\mathcal{E}$ is unchanged. Since also $B_\mathcal{E}$ and $C_\mathcal{E}$ are unchanged in this case, then $\mathcal{C}(\mathcal{E})$ does not decrease. This justifies the introduction of a new special action (action ($m8$)) to deal with the case of identical tail variables.

($m8$)**:** Same as ($m7$) when both S and S' are `nil`.

□

Soundness and completeness of this algorithm are assured by the following:

Theorem 9 (Soundness and Completeness)**.** *Let $e \wedge \mathcal{E}$ be an equation system, e an equation not in solved form, and $\mathcal{E}_1, \ldots, \mathcal{E}_h$ be the equation systems non-deterministically resulting from the application of the action of* Unify_bags *fired by e. Let $N_1, \ldots, N_k$ be the variables occurring in $\mathcal{E}_1, \ldots, \mathcal{E}_h$ but not in $e \wedge \mathcal{E}$; then $KW^m E^m_k F'_1 F_2 F_3 \vdash e \wedge \mathcal{E} \leftrightarrow \exists N_1, \ldots, N_k \bigvee_{i=1}^h \mathcal{E}_i$.*

Proof. (Sketch) The proof can be performed by case analysis, proving the claim for each single action. In particular one can observe that, since two multisets are equal if and only if they contain the same number of occurrences of each element, then the problem

$$\{\!\{ s_1, \ldots, s_m \,|\, X \}\!\} \doteq \{\!\{ t_1, \ldots, t_n \,|\, X \}\!\}$$

(action ($m8$)) is perfectly equivalent to the problem

$$\{\!\{ s_1, \ldots, s_m \}\!\} \doteq \{\!\{ t_1, \ldots, t_n \}\!\}.$$

Moreover, soundness and completeness of action ($m7$) follows from the strict analogy of this action with axiom (E^m_k). □

Before concluding this subsection, we want to point out an open problem for the multiset unification problem presented above, namely: is there a unification algorithm for multisets that incrementally computes a *minimal* complete set (cf., e.g., [24]) of unifiers?

```
Unify_clist_actions($\mathcal{E}$,$e$);
  case $e$ of
    $(c1)$–$(c3)$   as $(m1)$–$(m3)$ of Unify_bag_actions
    $(c4)$   $X \doteq f(t_0,\dots,t_n)$
             $f \not\equiv \llbracket \cdot \mid \cdot \rrbracket$ and $X$ occurs in $t$   $\mapsto$ fail
    $(c5)$   $X \doteq \llbracket t_0,\dots,t_n \mid f(t) \rrbracket$
             $X$ occurs in $t_0$ or ... or in $t_n$ or
             $f \not\equiv \llbracket \cdot \mid \cdot \rrbracket$ and $X$ occurs in $t$   $\mapsto$ fail
    $(c6)$   $X \doteq \llbracket t_0,\dots,t_n \mid X \rrbracket$
             $X$ does not occur in $t_0,\dots,t_n$   $\mapsto$
             $\mathcal{E}_1 := (t_0 \doteq t_1 \wedge \cdots \wedge t_0 \doteq t_n) \wedge \mathcal{E}_1$; push($X \doteq \llbracket t_0 \mid N \rrbracket$ , $\mathcal{E}_2$)
    $(c7)$   $f(s_1,\dots,s_m) \doteq g(t_1,\dots,t_n)$
             $f \not\equiv g$   $\mapsto$ fail
    $(c8)$   $f(s_1,\dots,s_m) \doteq f(t_1,\dots,t_m)$
             $f \not\equiv \llbracket \cdot \mid \cdot \rrbracket$   $\mapsto$
             $\mathcal{E}_1 := (s_1 \doteq t_1 \wedge \cdots \wedge s_m \doteq t_m) \wedge \mathcal{E}_1$
    $(c9)$   $\llbracket t \mid s \rrbracket \doteq \llbracket t' \mid s' \rrbracket \wedge \mathcal{E}$   $\mapsto$
             $(i)$     $\mathcal{E}_1 := (t \doteq t') \wedge \mathcal{E}_1$; push($s \doteq s'$ , $\mathcal{E}_2$)
             $(ii)$    $\mathcal{E}_1 := (t \doteq t') \wedge \mathcal{E}_1$; push($\llbracket t \mid s \rrbracket \doteq s'$ , $\mathcal{E}_2$)
             $(iii)$   $\mathcal{E}_1 := (t \doteq t)' \wedge \mathcal{E}_1$; push($s \doteq \llbracket t' \mid s' \rrbracket$ , $\mathcal{E}_2$).
```

Figure 4: Compact list unification rewriting rules

Indeed, the set of solutions computed by Unify_bags is not minimal, since it may contain also solutions which are obtainable by instantiation of other computed solutions (i.e., less general solutions). Furthermore, Unify_bags may compute through non-determinism repeated equivalent solutions, that is solutions which turn out to be identical in the given theory.

For example, given the equation $\{\!\!\{ A, A \mid S \}\!\!\} \doteq \{\!\!\{ A \mid S' \}\!\!\}$, whose complete set of unifiers contains the unique solution $[S'/\{\!\!\{ A \mid S \}\!\!\}]$, Unify_bags computes the three solutions, $[S'/\{\!\!\{ A \mid S \}\!\!\}]$, $[S'/\{\!\!\{ A \mid S \}\!\!\}]$, $[S/\{\!\!\{ A \mid N \}\!\!\}, S'/\{\!\!\{ A, A \mid N \}\!\!\}]$, where the first two are identical solutions, and the third one is less general than the previous two.

The corresponding minimality problems for compact lists and sets is also open. To obtain minimality in general, the unification algorithms should be modified so as to deal with various significant special cases. First contributions for a solution—limited to the set unification problem but easily adaptable to other contexts—are presented in [2, 25], and are not considered here for space limits. Such optimizations, however, are independent from the non-deterministic complexity analysis of the algorithms, which is another interesting line of research.

3.4 Unification of hybrid compact lists

The main part of the unification algorithm for hybrid compact lists is exactly the same as that of the unification algorithm for multisets shown in Figure 2 (where, of course, calls to Unify_bag_actions are replaced by calls to Unify_clist_actions) and is not repeated here.

As discussed in § 2.3, axiom (F_3) of the standard case must be modified so as to adhere to the semantics of the compact list constructor $\llbracket \cdot \mid \cdot \rrbracket$. This is reflected into the occur-checks performed by actions $(c4)$–$(c6)$ of the compact list unification algorithm of Figure 4.

The other important difference with respect to standard list unification is the addition of action $(c9)$ which reflects the extensionality principle for compact lists expressed by axiom (E^c_k).

Soundness and completeness of the algorithm are easy to prove, as done for the corresponding Theorem 9.

Theorem 10 (Soundness and Completeness)**.** *Let $e \wedge \mathcal{E}$ be an equation system, e an equation not in solved form, and $\mathcal{E}_1,\dots,\mathcal{E}_h$ be the equation systems non-deterministically resulting from the application of the action of* Unify_clists *fired by e. Let $N_1,\dots,N_k$ be the variables occurring in $\mathcal{E}_1,\dots,\mathcal{E}_h$ but not in $e \wedge \mathcal{E}$; then $KW^cE^c_kF'_1F_2F^c_3 \vdash e \wedge \mathcal{E} \leftrightarrow \exists N_1,\dots,N_k \bigvee_{i=1}^{h} \mathcal{E}_i$.*

The termination issue, on the contrary, deserves some special attention and will be discussed in detail in the next subsection.

3.4.1 Termination of the compact list unification algorithm

The complexity measure adopted for proving termination of the multiset unification algorithm does not work well with action ($c6$) of Unify_clists. Indeed, action ($c6$) leaves $A_{\mathcal{E}}$ (i.e., the number of non-eliminable variables) unchanged, whereas $B_{\mathcal{E}}$ may increase (in particular, when $n = 0$, $X \doteq \llbracket t_0 \,|\, X \rrbracket$ gets replaced by $X \doteq \llbracket t_0 \,|\, N \rrbracket$ and the subsequent substitution $[X/\llbracket t_0 \,|\, N \rrbracket]$ causes $B_{\mathcal{E}}$ to increase if X occurs elsewhere in the system).

To prove that, in spite of this, the algorithm still terminates for any input system $\mathcal{E}$, we need to introduce a different complexity measure. Intuitively, the problem with the measure adopted for multiset unification is that the size of a variable is 0, whereas we need to be able to associate with a variable a "measure" which is left unchanged (or possibly lowered) by subsequent substitutions over that variable.

The following definitions and results will serve to prove termination of the unification algorithm for compact lists. Let $p \geq 0$ and $\Phi_{\mathcal{E}}$ be the set of all possible functions mapping variables of the system $\mathcal{E}$ to the integer numbers $0, \ldots, p$. Let us define the following (higher-order) function PRK (*pseudo-rank*):

$$PRK_{\mathcal{E}}(\eta)(t) = \begin{cases} \eta(t), \eta \in \Phi_{\mathcal{E}} & \text{if } t \text{ is a variable} \\ 1 + \max_{j=1}^{m} PRK_{\mathcal{E}}(\eta)(t_j) & \text{if } t \text{ is } f(t_1, \ldots, t_m) \\ \max\{1 + PRK_{\mathcal{E}}(\eta)(s), PRK_{\mathcal{E}}(\eta)(r)\} & \text{if } t \text{ is } \llbracket s \,|\, r \rrbracket \end{cases}$$

The pseudo-rank function combines the notion of height of a term with the (equally widespread) notion of rank of a set. Note that compact list terms which are equivalent in the theory, though not syntactically identical (e.g., $\llbracket a \,|\, N \rrbracket$ and $\llbracket a, a \,|\, N \rrbracket$) have the same pseudo-rank value.

Lemma 7. *For any satisfiable system E, there exists a $k \geq 0$ and a function $\eta \in \Phi_E$ such that*

$$PRK_E(\eta)(\ell) = PRK_E(\eta)(r) \leq k$$

for each equation $\ell \doteq r$ in E (η is called a valid variable assignment *for E).*

Proof. If E is satisfiable then there is at least a substitution $\sigma = [X_1/t_1, \ldots, X_n/t_n]$ for the variables in E that makes all pairs of terms $\ell \doteq r$ in E simultaneously identical in the considered theory. Thus, one can take $\bar{\eta}(X_j) = PRK_E(\bar{\eta})(t_j)$, for all $j \in \{1, \ldots, n\}$, where $\bar{\eta}$ is the (constant) variable assignment function mapping all (possibly none) free variables of t_j to 0. It turns out also that the value of k can be chosen to be not greater than the number p of occurrences of non-variable symbols in E. □

Let $E_0, E_1, E_2, \ldots$ be the successive values of $\mathcal{E}$ along a branch of the Unify_clists($\mathcal{E}$) execution not ending with a failure.

Some of the functions in Φ_{E_i} which are valid variable assigments for E_i may turn out to be no longer valid for E_{i+1}, as it follows from the following example: let $E_0 \equiv f(X, Y) \doteq f(Y, X)$ and consider the valid variable assignment $\eta_0 = [X/0, Y/1]$. The latter is not a valid variable assignment for the system $E_1 \equiv X \doteq Y \wedge Y \doteq X$ obtained by the application of action (c_8) to E_0.

Conversely, it is easy to prove that a valid variable assignment for E_{i+1} is necessarily valid also for E_i. Moreover,

Lemma 8. *Let $E_0, \ldots, E_k$ be the successive values of $\mathcal{E}$ along a branch of a* Unify_clists($\mathcal{E}$) *execution. If E_k is satisfiable, then there is a valid variable assignment for E_k bounded by the number p of occurrences of functional symbols in the initial system E_0.*

Proof. By induction on $k \geq 0$. The base case ($k = 0$) follows by Lemma 7.

Assume the result holds for E_i. All actions, but ($c6$), do not introduce new terms in the system: they simply use the existing terms, whose PRK is, by hypothesis, not greater than p. In particular, variable substitution (action ($c3$)) replaces a variable X with a (already existing) term t whose PRK is necessarily the same as the PRK of X due to the presence in the system of the equation $X \doteq t$. Action ($c8$), as well as action ($c9$), replaces two terms with two or more (sub) terms of lower or equal PRK. Action ($c6$), on the contrary, builds a new term, $\llbracket t_0 \,|\, N \rrbracket$. However, the presence in E_{i+1} of the equation $X \doteq \llbracket t_0 \,|\, N \rrbracket$, forces the pseudo-rank of N to be not greater than the pseudo-rank of X (that is, by hypothesis, not greater than p). □

The non-determinism of the algorithm might generate sequences of systems such that E_k is not satisfiable even if E_0 is satisfiable. We need to be sure that when there is no valid variable assignment for a system E_k (e.g., when $E_k \equiv X \doteq f(Y) \wedge Y \doteq f(X)$) the computation terminates by failure. To this aim, we will define a weaker notion of validity for a variable assignment: a function η is said to be a *partially valid variable assignment* for E_k in $E_0, \ldots, E_k$ if

1. $PRK(\eta, \ell) \leq p$ and $PRK(\eta, r) \leq p$ for all $\ell \doteq r$ in E_k, where p is the number of occurrences of functional symbols in the initial system E_0; furthermore,

2. $PRK(\eta, X) = PRK(\eta, t)$ for all solved form equations $X \doteq t$ of E_k.

Clearly, a valid variable assignment is also a partially valid variable assignment. Moreover, any sequence $E_0, \dots, E_k$ of systems produced by the computation admits a partially valid variable assignment. As a matter of fact, only at the time of an *occur-check failure* there are the preconditions that avoids the existence of a partially valid variable assignment.

Theorem 11. Unify_clists *terminates for any input system $\mathcal{E}$.*

Proof. Assume η be a partially valid variable assignment function for $E_0, E_1, E_2, \dots$ (hence, the algorithm is not halted by occur-check failure). Let the *measure* of an equation $e \equiv \ell \doteq r$ of a system $\mathcal{E}$ with respect to η be defined as

$$\mathit{measure}(\eta, e) = PRK_{\mathcal{E}}(\eta)(\ell) + PRK_{\mathcal{E}}(\eta)(r)\,,$$

and let $\mathcal{E}'$ denote the system obtained from $\mathcal{E}$ by considering only its non-solved equations.

The following tuple of integers will be used as a complexity measure to prove termination of the algorithm:

$$\begin{array}{rcl} \mathcal{C}(\eta, \mathcal{E}) & = & \langle \;\; |\{e \text{ in } \mathcal{E}' \,:\, \mathit{measure}(\eta, e) = 2 \cdot p\}\,|, \\ & & \quad \dots, \\ & & \quad |\{e \text{ in } \mathcal{E}' \,:\, \mathit{measure}(\eta, e) = 0\}\,| \,\rangle\,. \end{array}$$

For each different action of the algorithm one can see that any successful phase (as in the proof of Theorem 8 a phase is a single action when the stack is empty, a finite sequence of actions devoted to empty the stack, otherwise) starting with this action, decreases $\mathcal{C}(\eta, \mathcal{E})$, according to the lexicographic ordering among tuples of integers. In particular (the number on the left indicates the first action of a phase):

6 : The presence in the new system of the equation $X \doteq [\![\, t_0, \dots, t_n \,|\, N \,]\!]$ forces the pseudo-rank of N to be not greater than the pseudo-rank of X, which implies that $\mathcal{C}(\eta, \mathcal{E})$ does not increase in consequence of the replacement of $X \doteq [\![\, t_0, \dots, t_n \,|\, X \,]\!]$ by $X \doteq [\![\, t_0 \,|\, N \,]\!]$ in $\mathcal{E}_1$. $\mathcal{C}(\eta, \mathcal{E})$ will decrease thanks to action ($c3$), which is required to be performed immediately after, as part of the same phase.

9 : One can view the global effect of the phase starting with action ($c9$) as that of replacing the selected equation

$$e \equiv [\![\, t_0, \dots, t_n \,|\, h \,]\!] \doteq [\![\, t'_0, \dots, t'_m \,|\, k \,]\!]$$

by a collection $t_{i_1} \doteq t'_{j_1}, \dots, t_{i_p} \doteq t'_{j_p}$ of equations relating elements of the two compact lists, plus one equation e' regarding the urelements h and k, as explained in detail below.

The *measure* of the selected equation e exceeds that of any $t_i \doteq t'_j$ equation generated by action ($c9$). Hence we only need to focus on the equations concerning h and k. Two cases need to be considered.

- Neither h nor k is a variable: since the *measure* of e' cannot exceed the *measure* of e, action ($c8$), immediately performed on e', will cause $\mathcal{C}(\eta, \mathcal{E})$ to decrease.
- At least one between h and k is a variable: e' has the form $h \doteq [\![\, t'_{i_1}, \dots, t'_{i_q} \,|\, k \,]\!]$, or $k \doteq [\![\, t_{i_1}, \dots, t_{i_q} \,|\, h \,]\!]$. Action ($c3$), immediately performed on e', will cause $\mathcal{C}(\eta, \mathcal{E})$ to decrease.

Thanks to the well-foundedness of the lexicographic ordering on tuples of non-negative integers, it will follow that the E_i sequence eventually terminates, which proves our thesis. □

Remark 5. *Comparing $\mathcal{C}(\eta, \mathcal{E})$ with the complexity measure used for proving termination of the multiset unification algorithm, one can observe that: $\mathcal{C}(\eta, \mathcal{E})$ subsumes both $A_{\mathcal{E}}$—the transformation of a variable from non-eliminable to eliminable reduces $\mathcal{C}(\eta, \mathcal{E})$—and $C_{\mathcal{E}}$—the removal of an equation from $\mathcal{E}$ reduces $\mathcal{C}(\eta, \mathcal{E})$. As concerns $B_{\mathcal{E}}$, instead, the notion of size of a term is replaced by the notion of pseudo-rank which allows us to give an appropriate weight to variables in advance.*

```
Unify_set_actions($\mathcal{E}$, $e$);
  case $e$ of
    $(s1)$–$(s5)$    as $(c1)$–$(c5)$ of Unify_clist_actions
                      with $[\![\cdot\,|\,\cdot]\!]$ replaced by $\{\cdot\,|\,\cdot\}$
    $(s6)$           $X \doteq \{t_0,\dots,t_n\,|\,X\}$
                     $X$ does not occur in $t_0,\dots,t_n$   $\mapsto$
                         push($X \doteq \{t_0,\dots,t_n\,|\,N\}, \mathcal{E}_2$)
    $(s7)$–$(s8)$    as $(c7)$–$(c8)$ of Unify_clist_actions
                      with $[\![\cdot\,|\,\cdot]\!]$ replaced by $\{\cdot\,|\,\cdot\}$
    $(s9)$           $\{t\,|\,s\} \doteq \{t'\,|\,s'\}$
                     tail($s$) and tail($s'$) are
                     not the same variable   $\mapsto$
                       $(i)$    $\mathcal{E}_1 := (t \doteq t') \wedge \mathcal{E}_1$; push($s \doteq s'$ , $\mathcal{E}_2$)
                       $(ii)$   $\mathcal{E}_1 := (t \doteq t') \wedge \mathcal{E}_1$; push($\{t\,|\,s\} \doteq s'$ , $\mathcal{E}_2$)
                       $(iii)$  $\mathcal{E}_1 := (t \doteq t') \wedge \mathcal{E}_1$; push($s \doteq \{t'\,|\,s'\}$ , $\mathcal{E}_2$)
                       $(iv)$   push($s \doteq \{t'\,|\,N\}$ , $\mathcal{E}_2$); push($\{t\,|\,N\} \doteq s'$ , $\mathcal{E}_2$)
    $(s10)$          $\{t_0,\dots,t_m\,|\,X\} \doteq \{t'_0,\dots,t'_n\,|\,X\}$
                     $X$ variable   $\mapsto$
                     select arbitrarily $i$ in $\{0,\dots,n\}$; choose one among:
                       $(i)$    $\mathcal{E}_1 := (t_0 \doteq t'_i) \wedge \mathcal{E}_1$;
                                push($\{t_1,\dots,t_m\,|\,X\} \doteq \{t'_0,\dots,t'_{i-1},t'_{i+1},\dots,t'_n\,|\,X\}, \mathcal{E}_2$)
                       $(ii)$   $\mathcal{E}_1 := (t_0 \doteq t'_i) \wedge \mathcal{E}_1$;
                                push($\{t_0,\dots,t_m\,|\,X\} \doteq \{t'_0,\dots,t'_{i-1},t'_{i+1},\dots,t'_n\,|\,X\}, \mathcal{E}_2$)
                       $(iii)$  $\mathcal{E}_1 := (t_0 \doteq t'_i) \wedge \mathcal{E}_1$;
                                push($\{t_1,\dots,t_m\,|\,X\} \doteq \{t'_0,\dots,t'_n\,|\,X\}, \mathcal{E}_2$)
                       $(iv)$   push($X \doteq \{t_0\,|\,N\}, \mathcal{E}_2$);
                                push($\{t_1,\dots,t_m\,|\,N\} \doteq \{t'_0,\dots,t'_n\,|\,N\}, \mathcal{E}_2$).
```

Figure 5: Set unification rewriting rules

3.5 Unification of hybrid sets

The main (deterministic) part of the unification algorithm for hybrid sets is still the same as that of the multiset unification algorithm of Figure 2 (still replacing calls to Unify_bag_actions by calls to Unify_set_actions). The remaining part of the algorithm (function Unify_set_actions—see Figure 5) is in a sense a combination of the unification algorithms for multisets and for compact lists shown in the previous two subsections.

As done with multisets and compact lists, we find it convenient to split $\mathcal{E}$ into two parts, $\mathcal{E}_1$ and $\mathcal{E}_2$, where $\mathcal{E}_2$ is dealt with as a stack, and to add some deterministic control to the algorithm in order to assure its termination.

The most important differences with respect to the algorithms presented so far are actions $(s6)$, $(s9)$, and $(s10)$.

Action $(s6)$ differs from the corresponding action $(c6)$ of the compact list unification algorithm in that a solution to the equation $X \doteq \{t_0,\dots,t_n\,|\,X\}$ does not necessarily require that $t_0,\dots,t_n$ are unifiable terms, since duplicates are immaterial in a set.

Action $(s9)$—whose aim is the reduction of set-set equations—is simply obtained by merging actions $(m7)$ and $(c9)$ of the previous two algorithms, since, as already noticed in § 2.4, the set equality axiom E^s_k comprises E^m_k and E^c_k. In particular, cases *(ii)* and *(iii)* of action $(s9)$ take care of duplicates in the left-hand side term and in the right-hand side term, respectively. Case *(iv)*, instead, takes care of permutativity of the set constructor $\{\cdot\,|\,\cdot\}$.

Finally, action $(s10)$ deals with the case of equations of the form

$$\{t_0,\dots,t_m\,|\,X\} \doteq \{t'_0,\dots,t'_n\,|\,X\}\,,$$

where the two sides are set terms with the same variable tail element. The problem with this kind of equations is the same singled out in § 3.3. However, in this case, the given equation can not be simply replaced by the equation $\{t_0,\dots,t_m\} \doteq \{t'_0,\dots,t'_n\}$, since there are other possible solutions not covered by this new equation. For instance, the equation $\{a,b\,|\,X\} \doteq \{b\,|\,X\}$, which has no solution as a multiset unification problem, has the two distinct solutions, $X = \{a \mid N\}$ and $X = \{a, b \mid N\}$, as a set unification problem. To preserve completeness, therefore, the algorithm is forced to consider non-deterministically each element of one of the two sets involved in the set-set equation, so as to explore all possible combinations. Clearly, this solution opens a big (though finite) number of alternatives, possibly leading to redundant solutions. In [2] it is shown how to improve the algorithm from this point of view.

Termination of this algorithm can be easily proved along the lines of the termination proof of the compact list unification algorithm.

Theorem 12 (Termination)**.** Unify_bags *terminates for any input system $\mathcal{E}$.*

Sketch. Use the same complexity measure $\mathcal{C}(\eta, \mathcal{E})$ adopted in proving Theorem 11. In order to extend this proof to the case of sets, one can further see that both actions $(s6)$ and $(s9)iv$ (not present in Unify_clist_actions) lower the value of $\mathcal{C}(\eta, \mathcal{E})$, and that the combination of these with other actions does not negatively influence the behaviour of $\mathcal{C}(\eta, \mathcal{E})$. Furthermore, in action $(s10)$ an equation of the form $X \doteq \{t_{i_1}, \dots, t_{i_q}, t'_{j_1}, \dots, t'_{j_{q'}} \mid X\}$ is added to the working system. After actions $(s6)$ and $(s3)$ are performed on this equation, $\mathcal{C}$ decreases. □

Theorem 13 (Soundness and Completeness)**.** *Let $e \wedge \mathcal{E}$ be an equation system, e an equation not in solved form, and $\mathcal{E}_1, \dots, \mathcal{E}_h$ be the equation systems non-deterministically resulting from the application of the action of* Unify_sets *fired by e. Let $N_1, \dots, N_k$ be the variables occurring in $\mathcal{E}_1, \dots, \mathcal{E}_h$ but not in $e \wedge \mathcal{E}$; then $KW^sE^s_kF'_1F_2F^s_3 \vdash e \wedge \mathcal{E} \leftrightarrow \exists N_1, \dots, N_k \bigvee_{i=1}^{h} \mathcal{E}_i$.*

Complete proofs that the algorithm is sound and complete with respect to the given set theory, along with the complete termination proof, can be found in [9, 6].

Finally, let us remark that the algorithm of Figure 5 is the algorithm used in the language {log}, a logic programming language enriched with finite sets [9]. This algorithm has been also used almost unchanged within the set constraint satisfiability procedure of the {log} language reconsidered as an instance of the general CLP scheme [10, 7].

4 Conclusions and further research

In the first part of the paper we have presented axiomatically specified first-order theories of lists, multisets, compact lists, and sets. Such axiomatizations have been shown to be especially suitable for the integration with free functor symbols governed by the classical Clark's axioms in the context of Constraint Logic Programming. Moreover, the adaptations of the extensionality principle to the various theories taken into account have been exploited in the design of unification algorithms presented in the second part of the paper.

All the theories presented can be combined providing frameworks to deal with several of the proposed data structures simoultaneously. The unification algorithms proposed can be combined (merged) as well to produce engines for such combination theories.

The open problem of devising minimal (irredundant) unification algorithms for multisets, compact lists, and sets, has been briefly discussed and represents one of the most important (at least from a practical point of view) next step in this research.

Other two important lines for further research are the following: the adaptation of the results presented to the case of non-well-founded sets and/or rational terms; the extension of the unification algorithms into general purpose constraint solvers capable of dealing with negative information. Both these problems have been tackled, in the case of sets, in [8, 10].

Acknowledgements

We are grateful to Eugenio G. Omodeo for his precious suggestions and comments in many parts of the paper. Enrico Pontelli actively helped us in proving termination of the set unification algorithm.

References

[1] Apt, K. R. Introduction of Logic Programming. In *Handbook of Theoretical Computer Science, vol. B: Formal Models and Semantics*, J. van Leeuwen, Ed. Elsevier and The MIT Press, 1990.

[2] Arenas-Sánchez, P., and Dovier, A. A minimality study for set unification. *Journal of Functional and Logic Programming*, 1997(7):1–49.

[3] Bellé, D., and Parlamento, F. Undecidability of Weak Membership Theories. In *Proceedings of the International Conference on Logic and Algebra (in memory of R. Magari)* (1994).

[4] Bernays, P. A system of axiomatic set theory. Part I. *The Journal of symbolic logic 2* (1937), 65–77.

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

[6] Dovier, A. *Computable Set Theory and Logic Programming.* PhD thesis, Università degli Studi di Pisa, March 1996. TD–1/96.

[7] Dovier, A., Piazza, C., Pontelli, E., and Rossi, G. On the Representation and Management of Finite Sets in CLP-languages. In J. Jaffar, editor, *Proceedings of 1998 Joint International Conference and Symposium on Logic Programming*, pages 40–54. The MIT Press, Cambridge, Mass., June 1998.

[8] Dovier, A., Omodeo, E. G., and Policriti, A. Solvable set/hyperset contexts: II. a goal-driven unification algorithm for the blended case. *Applicable Algebra in Engineering, Communication and Computing*, 1998. (To appear)

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

[10] Dovier, A., and Rossi, G. Embedding Extensional Finite Sets in CLP. In *Proc. of Int'l Logic Programming Symposium, ILPS'93* (1993), D. Miller, Ed., The MIT Press, Cambridge, Mass., pp. 540–556.

[11] Herbrand, J. Recherches sur la theorie de la demonstration. Master's thesis, Université de Paris, 1930. Also in *Ecrits logiques de Jacques Herbrand*, PUF, Paris, 1968.

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

[13] Kapur, D., and Narendran, P. NP-completeness of the set unification and matching problems. In *8th International Conference on Automated Deduction* (1986), J. H. Siekmann, Ed., vol. 230 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, pp. 489–495.

[14] Kunen, K. *Set Theory. An Introduction to Independence Proofs.* Studies in Logic. North Holland, Amsterdam, 1980.

[15] Lassez, J. L., Maher, M. J., and Marriot, K. Unification Revisited. In *Foundations of Deductive Databases and Logic Programming*, J. Minker ed., Morgan Kaufmann Publishers, pp. 587–625, 1987.

[16] Lloyd, J. W. *Foundations of Logic Programming.* Springer-Verlag, Berlin, 1987. Second edition.

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

[18] Martelli, A., and Montanari, U. An efficient unification algorithm. *ACM Transactions on Programming Languages and Systems 4* (1982), 258–282.

[19] Montagna, F., and Mancini, A. A minimal predicative set theory. *Notre Dame Journal of Formal Logic 35* (1994), 186–203.

[20] Omodeo, E. G., Policriti, A., and Rossi., G. F. Che genere di insiemi/ multi-insiemi/ iperinsiemi incorporare nella programmazione logica? In *Proc. Eighth Italian Conference on Logic Programming* (1993), D. Saccà, Ed., pp. 55–70.

[21] Parlamento, F., and Policriti, A. Decision Procedures for Elementary Sublanguages of Set Theory IX. Unsolvability of the Decision Problem for a Restricted Subclass of δ_0-Formulas in Set Theory. *Communications of Pure and Applied Mathematics 41* (1988), 221–251.

[22] Paterson, M. S., and Wegman, M. N. Linear unification. *Journal of Computer System Science 16*, 2 (1978), 158–167.

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

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

[25] Stolzenburg, F. Membership-Constraint and Complexity in Logic Programming with Sets. In *Proc. First Int'l Workshop on Frontier of Combining Systems* (1996), F. Baader and K. U. Schulz, Eds., Kluwer Academic Publishers.

[26] Tarski, A., and Givant, S. *A Formalization of Set Theory without Variables*, vol. 41 of *Colloquium Publications.* American Mathematical Society, 1986.

[27] Vaught, R. L. On a Theorem of Cobham Concerning Undecidable Theories. In *Proceedings of the 1960 International Concress* (1962), E. Nagel, P. Suppes, and A. Tarski, Eds., Stanford University Press, Stanford, pp. 14–25.