

# INTEGRATING LISTS, MULTISSETS, AND SETS IN A LOGIC PROGRAMMING FRAMEWORK

AGOSTINO DOVIER

*Dip. di Informatica, Univ. di Pisa. Corso Italia 40, 56100  
PISA (I). dovier@di.unipi.it*

ALBERTO POLICRITI

*Dip. di Matematica e Informatica, Univ. di Udine. Via delle  
Scienze 206, 33100 UDINE (I). policrit@dimi.uniud.it*

AND

GIANFRANCO ROSSI

*Dip. di Matematica, Univ. di Parma. Via M. D'Azeglio 85/A,  
43100 PARMA (I). gianfr@prmat2.math.unipr.it*

## **Abstract.**

The first order theories of lists, bags, 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 especially suitable 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. Applications can advantageously exploit these properties when dealing with lists. However, there are a number of cases

where these properties turn out to be too restrictive: different data abstractions may fit more naturally the problem requirements. In particular, it seems 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;
- *multi-sets* (or bags), in which the number of occurrences of each element is important, whereas the order of elements does not matter;
- *compact-lists*, in which the order of elements is important whereas the number of contiguous occurrences of each element is not.

Although only relatively few programming languages support sets and multi-sets as primitive features of the language, they are well-known data structures, whose usefulness as a powerful data abstraction mechanism is widely recognized.

In contrast, compact-lists have been scarcely studied. To give an intuitive semantics of compact-lists, consider the following real-life problem: assume a laser printer can print on sheets of different sizes, but it has a unique paper feeder. Any time the size of the paper changes, the printer must switch paper feeder. The information necessary for the scheduler in order to correctly perform such switches is conveniently modeled by a compact-list (cf. § 2.3 for the formal definition of compact-lists).

The framework in which we assume the above mentioned data structures are to be incorporated is a (constraint) logic programming one, where a careful choice of suitable data structures is crucial for the control of the execution. All the considered data structures will be represented by terms of a logic language, moreover, we assume that the language provides, a number (possibly zero or infinite) of free functional symbols in addition to the data structure constructors. Thus, we will consider so-called *hybrid theories*, in which the basic objects are Herbrand terms. Therefore, our data objects will be standard terms, lists, bags, compact-lists, and sets of standard terms, as well as any possible combination of them.

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.

## 2. An axiomatic view of lists, compact-lists, multi-sets, and sets

In what follows we will use standard Prolog syntactic conventions and notations. In particular,  $[\cdot|\cdot]$  will be used as the list constructor and the constant `nil` as the empty list. Thus, for instance,  $[a|\text{nil}]$  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 multi-sets, compact-lists, and sets (empty multi-sets, compact-lists, and sets are all denoted by `nil`):

- $\{\cdot | \cdot\}$  (of arity 2) for multi-sets,
- $\llbracket \cdot | \cdot \rrbracket$  (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 multi-sets, 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 with equality ‘ $\doteq$ ’, and membership ‘ $\in$ ’ predicate symbols, and consisting of the two axioms

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

Skolemizing  $(N)$  and  $(W)$ , we introduce the two functional symbols `nil` and  $[\cdot | \cdot]$ , and we can rewrite  $(N)$  and  $(W)$  as

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

The language of the theory consists of the signatures  $\Pi = \{\doteq, \in\}$  for the predicate symbols and  $\Sigma = \{\text{nil}, [\cdot | \cdot], \dots\}$  for the functional symbols. It can be proved (see (Dovier, 1996)) that any model of  $NW$  (the theory consisting of the axioms  $(N)$  and  $(W)$ ) must necessarily be infinite and that this theory is not complete (nor model-complete). Moreover, it has been shown (e.g. in (Vaught, 1962; Parlamento and Policriti, 1988; Bellé and Parlamento, 1994)) that  $NW$  is not decidable. Nevertheless, this theory can be strengthened, by adding new axioms, in order to obtain a complete and decidable theory for suitable classes of sentences.

The following three axiom shemata (called freeness axioms, or Clark’s equality axioms—see (Clark, 1978)) are usually introduced in logic programming and will play an important role in our axiomatization:

$$\begin{aligned} (F_1) \quad & \forall x_1 \dots x_n y_1 \dots y_n \left( \begin{array}{l} f(x_1, \dots, x_n) \doteq f(y_1, \dots, y_n) \\ \rightarrow x_1 \doteq y_1 \wedge \dots \wedge x_n \doteq y_n \end{array} \right) \quad f \in \Sigma \\ (F_2) \quad & \forall x_1 \dots x_m y_1 \dots y_n \quad f(x_1, \dots, x_m) \not\doteq g(y_1, \dots, y_n) \quad f \not\doteq g \\ (F_3) \quad & \forall x \quad (x \not\doteq t[x]) \\ & \text{where } t[x] \text{ denotes a } \Sigma\text{-term having } x \text{ as a proper subterm.} \end{aligned}$$

Axiom  $(F_1)$  holds also for  $[\cdot | \cdot] \in \Sigma$ , and expresses the adaptation of the classical extensionality principle to lists.

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 terms. Such an extension is considered in (Dovier, 1996) and, with regards to sets and multi-sets only, in (Omodeo, Policriti, and Rossi, 1993). In this paper, on the contrary, we prefer for the sake of simplicity not considering this kind of extension.

In the following three subsections we will adapt the above axioms so as to fulfill the intended meaning of bags, compact-lists, and sets. Before proceeding, however, we need to consider the following problem: what are the elements of an object denoted by a term  $f(t_1, \dots, t_n)$ , with  $f$  free? For example one can write a term  $t$  of the form  $[t_1, \dots, t_n | a]$ , and  $(W)$  states that  $t_1, \dots, t_n$  are elements of  $t$ , whereas the remaining elements of  $t$  are those of  $a$ .

To keep the notion of list (as well as of bag, compact-list and set) as distinct as possible from the notion of ‘free’ term, we will strenghten axiom  $(N)$  so as to keep *empty* any term which is not a list (a bag, a compact-list, a set) of terms.

Therefore, we introduce the axiom schema

$$(K) \quad \forall x y_1 \cdots y_n (x \notin f(y_1, \dots, y_n))$$

for any  $f \in \Sigma$ ,  $ar(f) = n$ ,  $f$  distinct from  $[\cdot | \cdot]$ ,  $\{\{\cdot | \cdot\}\}$ ,  $\llbracket \cdot | \cdot \rrbracket$ ,  $\{\cdot | \cdot\}$

which generalizes  $(N)$  and will be used in its place hereinafter.<sup>1</sup>

We will call *Ur-Element* (cf. (Tarski and Givant, 1986)) any term of the form  $f(t_1, \dots, t_n)$ , where  $f \in \Sigma$ ,  $f$  distinct from  $[\cdot | \cdot]$ ,  $\{\{\cdot | \cdot\}\}$ ,  $\llbracket \cdot | \cdot \rrbracket$ ,  $\{\cdot | \cdot\}$ ,  $ar(f) = n$ . When an ur-element is ground, it will be called a *kernel*. Intuitively, lists (bags, compact-lists and sets) can be seen as built starting from a kernel (in particular, from `nil`) and then adding to the kernel the other elements composing the data structure.

Axioms  $(K)$ ,  $(W^l)$ ,  $(F_1)$ ,  $(F_2)$ , and  $(F_3)$ , along with standard equality axioms, identify the hybrid theory of lists over the alphabets  $\Pi$  and  $\Sigma$ .

**Remark 1.1** *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. We do not consider the case of objects non-denoted by terms, which can easily be avoided by suitable forms of the domain closure axiom and are outside the scope of this paper.*

<sup>1</sup>Indeed, since `nil` belongs to  $\Sigma$ ,  $(N^l)$  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  $[\cdot | \cdot]$  ( $\{\{\cdot | \cdot\}\}$ ,  $\llbracket \cdot | \cdot \rrbracket$ ,  $\{\cdot | \cdot\}$ ), then  $(F_2)$  is a theorem of  $KW$ .

## 2.2. BAGS

The signature of an hybrid theory of bags must contain the binary functional symbol  $\{\cdot|\cdot\}$  and `nil`. By skolemization of  $(W)$  we can rewrite  $(W)$  as follows

$$(W^m) \quad \forall y v x (x \in \{\{y|v\} \} \leftrightarrow x \in v \vee x \doteq y).$$

For the context of bags the symbol  $[\cdot|\cdot]$  is replaced by  $\{\cdot|\cdot\}$  in  $\Sigma$ . The behavior of the interpreted functional symbol  $\{\cdot|\cdot\}$  is regulated by the following axiom (permutativity axiom):

$$(E_1^m) \quad \forall xyz \{\{x,y|z\} \} \doteq \{\{y,x|z\} \}$$

which, intuitively, states that the order of elements in a bag is immaterial. This means, for example, that

$$\{\{a|\{b\}\} \} \text{ (i.e. } \{\{a,b\}\} \text{)} \doteq \{\{b|\{a\}\} \} \text{ (i.e. } \{\{b,a\}\} \text{)}$$

even if  $a$  is distinct from  $b$ . More generally:

**Lemma 1.2** *Let  $\pi : \{1, \dots, n\} \longrightarrow \{1, \dots, n\}$  be a permutation, then*

$$KW^m E_1^m \vdash \{\{s_1, \dots, s_n | t\} \} \doteq \{\{s_{\pi_1}, \dots, s_{\pi_n} | t\} \}$$

for any tuple of  $\Sigma$ -terms  $s_1, \dots, s_n, t$ .

As opposed to what happens for lists, in the context of bags, axiom schema  $(F_1)$  does not hold when  $f$  is instantiated to  $\{\cdot|\cdot\}$ , as it ensues from the above example. Since it will turn out that axiom schema  $(F_1)$  does not hold also in the cases of compact-lists and sets, we will modify it as follows:

$$(F_1') \quad \forall x_1 \dots x_n y_1 \dots y_n \left( \begin{array}{l} f(x_1, \dots, x_n) \doteq f(y_1, \dots, y_n) \\ \rightarrow x_1 \doteq y_1 \wedge \dots \wedge x_n \doteq y_n \end{array} \right)$$

for any  $f \in \Sigma$ ,  $f$  distinct from  $\{\cdot|\cdot\}$ ,  $[\cdot|\cdot]$ ,  $\{\cdot|\cdot\}$ ,  $\{ \cdot | \cdot \}$ ,  $ar(f) = n$ .

A by-product of the fact that  $(F_1)$  does not hold is that in  $KW^m E_1^m$  we lack in a principle for establishing equality between objects. On the other hand, any such equality principle must be consistent with the following simple result relating ‘ $\in$ ’ and ‘ $\doteq$ ’:

**Lemma 1.3** *For all  $n \in \omega$  and for all  $x, y_1, \dots, y_n$*

$$KW^m E_1^m \vdash x \in \{\{y_1, \dots, y_n\} \} \text{ iff } KW^m E_1^m \vdash \exists z (\{\{y_1, \dots, y_n\} \} \doteq \{\{x|z\} \}).$$

We can introduce the following extensionality axiom for bag comparison: *bag-extensionality* with kernels

$$(E_k^m) \quad \forall y_1 y_2 v_1 v_2 \left( \begin{array}{l} \{\{y_1|v_1\} \} \doteq \{\{y_2|v_2\} \} \leftrightarrow \\ (y_1 \doteq y_2 \wedge v_1 \doteq v_2) \vee \\ \exists z (v_1 \doteq \{\{y_2|z\} \} \wedge v_2 \doteq \{\{y_1|z\} \}) \end{array} \right)$$

Axiom  $(E_k^m)$  states that two non-empty bags are equal if and only if they are based on the same *kernel* (in particular `nil`) and they have the same number of occurrences of each element, regardless their order.

Axioms  $(K)$ ,  $(W^m)$ ,  $(E_k^m)$ ,  $(F_1')$ ,  $(F_2)$ , and  $(F_3)$ , along with standard equality axioms, identify the hybrid theory of multi-sets over the alphabets  $\Pi$  and  $\Sigma$ .

### 2.3. COMPACT-LISTS

Let  $\Sigma$  be  $\{\text{nil}, \llbracket \cdot | \cdot \rrbracket, \dots\}$ , relative to the new version of  $(W)$  for compact-lists

$$(W^c) \quad \forall y v x (x \in \llbracket y | v \rrbracket \leftrightarrow x \in v \vee x \doteq y).$$

The fundamental property of the compact-list constructor  $\llbracket \cdot | \cdot \rrbracket$  is the *absorption property*, described by the following axiom

$$(E_2^c) \quad \forall xy \llbracket x, x | y \rrbracket \doteq \llbracket x | y \rrbracket$$

which, intuitively, states that contiguous duplicates in a compact-list are immaterial.

Similarly to bags, also for compact-lists axiom schema  $(F_1)$  does not correctly interpret the behavior of the functional symbol  $\llbracket \cdot | \cdot \rrbracket$ , as it ensues from the following example:

$$\llbracket a | \llbracket a \rrbracket \rrbracket \text{ (i.e. } \llbracket a, a \rrbracket) \doteq \llbracket a | \text{nil} \rrbracket \text{ (i.e. } \llbracket a \rrbracket).$$

Moreover, also the freeness axiom  $(F_3)$  must be modified so as to agree with the intended meaning of  $\llbracket \cdot | \cdot \rrbracket$ . Indeed, one can observe, in particular, that an equation such as  $x \doteq \llbracket a | x \rrbracket$  admits a finite tree solution, namely a solution that binds  $x$  to the term  $\llbracket a | t \rrbracket$ , where  $t$  is any term.

We first modify axiom  $(F_1)$  by introducing the extensionality principle for compact-lists: *compact-lists extensionality* with kernels

$$(E_k^c) \quad \forall y_1 y_2 v_1 v_2 \left( \begin{array}{l} \llbracket y_1 | v_1 \rrbracket \doteq \llbracket y_2 | v_2 \rrbracket \leftrightarrow \\ (y_1 \doteq y_2 \wedge v_1 \doteq v_2) \vee \\ (y_1 \doteq y_2 \wedge v_1 \doteq \llbracket y_2 | v_2 \rrbracket) \vee \\ (y_1 \doteq y_2 \wedge \llbracket y_1 | v_1 \rrbracket \doteq v_2) \end{array} \right)$$

The first disjunct takes care of the simplest case where no duplicates occur in the two compact-lists. The second and third cases, instead, are relative to the case in which there are duplicates in the right-hand side term and in the left-hand side term, respectively.

As far as axiom  $(F_3)$  is concerned, notice that the following lemma holds:

**Lemma 1.4** *In any model of  $KW^cE_2^c$  such that every element has a finite number of members, the following holds:*

$$\exists x (x \doteq \llbracket y_1, \dots, y_n \mid x \rrbracket) \leftrightarrow (y_1 \doteq y_2 \doteq \dots \doteq y_n).$$

Notice that the finiteness requirement is necessary for Lemma 1.4 since, if we would accept infinite solutions, then, for any  $y_1, \dots, y_n$ , the equation  $x \doteq \llbracket y_1, \dots, y_n \mid x \rrbracket$  would admit always the infinite (rational) solution

$$x = \llbracket y_1, \dots, y_n, y_1, \dots, y_n, y_1, \dots, y_n, \dots \rrbracket.$$

Axiom  $F_3$  is therefore replaced by

$$(F_3^c) \quad \forall x (x \neq t[x]) \\ \text{unless } t \text{ has the form } \llbracket t_1, \dots, t_n \mid x \rrbracket, x \text{ not occurring in } t_1, \dots, t_n, \\ \text{and } t_1 \doteq \dots \doteq t_n.$$

The hybrid theory of compact-lists over  $\Pi$  and  $\Sigma$  is identified by axioms  $(K)$ ,  $(W^c)$ ,  $(E_k^c)$ ,  $(F_1')$ ,  $(F_2)$ , and  $(F_3^c)$ , along with standard equality axioms.

#### 2.4. SETS

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

$$(W^s) \quad \forall y v x (x \in \{y \mid v\} \leftrightarrow x \in v \vee x \doteq y).$$

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

$$(E_1^s) \quad \forall xyz \{x, y \mid z\} \doteq \{y, x \mid z\}$$

$$(E_2^s) \quad \forall xy \{x, x \mid y\} \doteq \{x \mid y\}.$$

Similarly to what has been done in the previous two subsections, we define an extensionality criterion for testing equality between two sets

$$(E_k^s) \quad \forall y_1 y_2 v_1 v_2 \left( \begin{array}{l} \{y_1 \mid v_1\} \doteq \{y_2 \mid v_2\} \leftrightarrow \\ (y_1 \doteq y_2 \wedge v_1 \doteq v_2) \vee \\ (y_1 \doteq y_2 \wedge v_1 \doteq \{y_2 \mid v_2\}) \vee \\ (y_1 \doteq y_2 \wedge \{y_1 \mid v_1\} \doteq v_2) \vee \\ \exists k (v_1 \doteq \{y_2 \mid k\} \wedge v_2 \doteq \{y_1 \mid k\}) \end{array} \right)$$

Such principle combines the results proved separately for  $(E_k^m)$  and  $(E_k^c)$ , so that 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:

$(F_3^s) \quad \forall x \ (x \neq t[x])$   
*unless  $t$  has the form  $\{t_1, \dots, t_n \mid x\}$ ,  $x$  not occurring in  $t_1, \dots, t_n$ .*

The hybrid theory of sets (with  $\Pi = \{\doteq, \in\}$ , and  $\Sigma = \{\mathbf{nil}, \{\cdot \mid \cdot\}, \dots\}$ ) is therefore identified by axioms  $(K)$ ,  $(W^s)$ ,  $(E_k^s)$ ,  $(F_1')$ ,  $(F_2)$ , and  $(F_3^s)$ , along with standard equality axioms.

**Remark 1.5** *Notice that it is easily seen that in every model of each of the considered theories in which all elements are denoted by terms (hence finite—cf. remark at the end of § 1.1) the membership cannot form either cycles or infinite descending chains. To this extent the above theories can be considered well-founded.*

**Remark 1.6** *The axiomatizations presented can easily be combined in order to obtain axiomatic theories capable to deal with any subset of the collection of proposed data structures. Moreover, the unification algorithms presented in the next section can easily be merged to solve the unification problem relative to such “combined” context.*

### 3. Unification of bags, compact-lists and sets

While unification for (hybrid) lists can be performed in linear time (cf. (Paterson and Wegman, 1976; Martelli and Montanari, 1982)), the unification problems for all the other data structures analyzed in this paper are NP-complete. NP-hardness can easily be proved for example via reduction of the 3-SAT famous problem to the unification problem for bags, compact-lists and sets (Dovier, 1996; Dovier, Omodeo, Pontelli, and Rossi, 1993). For the NP-completeness, see (Kapur and Narendran, 1986; Omodeo and Policriti, 1994; Dovier, 1996).

We start recalling the unification algorithm for hybrid lists, that is basically the standard unification algorithm à la Robinson. 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.

`Unify_lists` terminates on any input system of equations  $\mathcal{E}$ , returning an equivalent system in solved form<sup>2</sup> from which it is immediate to obtain the *unique most general unifier* for the given unification problem. Moreover, correctness and completeness of the algorithm can be proved w.r.t. the corresponding theory presented in § 2.1 (cf., for instance, (Lloyd, 1987; Lassez, Maher, and Marriot, 1986)).

<sup>2</sup>Remember that, a system of equations  $\mathcal{E}$  is said to be in *solved form* if it has the form  $\{X_1 \doteq t_1, \dots, X_n \doteq t_n\}$  and the  $X_i$ s are distinct variables which do not occur in r.h.s. terms of any equation of  $\mathcal{E}$ .

function `Unify_lists( $\mathcal{E}$ )`:

$$\begin{array}{ll}
(l1) & X \doteq X \wedge \mathcal{E} \quad \mapsto \quad \mathcal{E} \\
(l2) & \left. \begin{array}{l} t \doteq X \wedge \mathcal{E} \\ t \text{ is not a variable} \end{array} \right\} \mapsto X \doteq t \wedge \mathcal{E} \\
(l3) & \left. \begin{array}{l} X \doteq t \wedge \mathcal{E} \\ X \text{ does not occur in } t \\ X \text{ occurs in } \mathcal{E} \end{array} \right\} \mapsto \mathcal{E}[X/t] \wedge X \doteq t \\
(l4) & \left. \begin{array}{l} X \doteq t \wedge \mathcal{E} \\ X \text{ occurs in } t \end{array} \right\} \mapsto \text{fail} \\
(l5) & \left. \begin{array}{l} f(s_1, \dots, s_m) \doteq g(t_1, \dots, t_n) \wedge \mathcal{E} \\ f \text{ different from } g \end{array} \right\} \mapsto \text{fail} \\
(l6) & f(s_1, \dots, s_m) \doteq f(t_1, \dots, t_m) \wedge \mathcal{E} \quad \mapsto \\
& \qquad \qquad \qquad s_1 \doteq t_1 \wedge \dots \wedge s_m \doteq t_m \wedge \mathcal{E}.
\end{array}$$

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 bags, compact-lists, and sets, action (l6), when  $f$  is the corresponding interpreted functional symbol, will be replaced by non-deterministic actions which reflect the axioms ( $E_k^b$ ), ( $E_k^c$ ), and ( $E_k^s$ ), respectively.

### 3.1. UNIFICATION OF HYBRID BAGS

The first five actions are exactly the same as in the algorithm for hybrid lists. In addition, we need to restrict applicability of action (l6) to non-bag terms only (action (m6)), and to introduce a new action to deal with bag-bag equations (action (m7)). This case introduces a source of don't know non-determinism: both alternatives (i) and (ii) must be exploited in order to guarantee completeness.

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

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

$$\begin{aligned}
\{\{T | S\} \doteq \{\{T' | S\} \xrightarrow{m7(ii)} \\
S \doteq \{\{T' | N\} \wedge \{\{T | N\} \doteq S \xrightarrow{m3} \\
S \doteq \{\{T' | N\} \wedge \{\{T | N\} \doteq \{\{T' | N\}\}.
\end{aligned}$$

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

More generally, for any situation of the form

$$\begin{aligned}
\{\{\dots | S_1\} \doteq \{\{\dots | S_2\} \wedge \\
\{\{\dots | S_2\} \doteq \{\{\dots | S_3\} \wedge \\
\vdots \wedge \\
\{\{\dots | S_n\} \doteq \{\{\dots | S_1\}\},
\end{aligned}$$

it is easy to find a non-deterministic sequence of actions leading to non-termination of the bag unification algorithm. An intuitive reason for non-termination is that the unification algorithm looks for all possible substitutions for  $S$  such that  $\{\{T | S\} \doteq \{\{T' | S\}$  holds. In this process an infinite number of possibilities, corresponding to subsequently incrementations of  $S$  is generated. Clearly the solution in which  $S$  is unbounded is the most general one.

The situation described in the first example above (the base case) can be easily handled as special. Let `tail` and `de_tail` be the following functions:

$$\begin{aligned}
\text{tail}(\text{nil}) &= \text{nil} & \text{de\_tail}(X) &= \text{nil} \\
\text{tail}(X) &= X & \text{de\_tail}(\{\{t | s\}) &= \{\{t | \text{de\_tail}(s)\}. \\
\text{tail}(\{\{t | s\}) &= \text{tail}(s)
\end{aligned}$$

Action (m7) can be splitted into two sub-actions. If `tail(s)` and `tail(s')` are not the same variable then perform action (m7). Otherwise replace  $\{\{t | s\} \doteq \{\{t' | s'\}$  with `de_tail`( $\{\{t | s\}) \doteq \text{de\_tail}(\{\{t' | s'\})$ .

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. In this way, we can ensure enough determinism in the algorithm to guarantee that when an equation  $\{\{s_1, \dots, s_m | S\} \doteq \{\{t_1, \dots, t_n | S'\}$  is encountered, the sequence of actions executed is such that the algorithm returns a conjunction of equations of the form  $s_{i_1} \doteq t_{j_1}, \dots, s_{i_k} \doteq t_{j_k}$ , along with either:

- two equations of the form  $S \doteq \{\{t_{j_{k+1}}, \dots, t_{j_n} | N\}$ ,  
 $S' \doteq \{\{s_{i_{k+1}}, \dots, s_{i_m} | N\}$ , or
- one equation of the form  $S \doteq \{\{t_{j_{k+1}}, \dots, t_{j_n} | S'\}$ , or
- one of the form  $S' \doteq \{\{s_{i_{k+1}}, \dots, s_{i_m} | S\}$ .

After that, by applying the substitutions for the variables  $S$  and  $S'$ , even if a new variable  $N$  is introduced into the system,  $S$  and/or  $S'$  become eliminable, that is they occur in  $\mathcal{E}$  only as l.h.s. of one equation, so that, in a sense, they disappear from the system.

The bag unification algorithm is shown in Figure 1.

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

$$\begin{aligned} \{ \{ T_1 \mid S_1 \} \doteq \{ \{ T_2 \mid S_2 \} \wedge \{ T_3 \mid S_2 \} \} \doteq \{ \{ T_4 \mid S_1 \} \} & \xrightarrow{m7(ii)} \\ S_1 \doteq \{ \{ T_2 \mid N_1 \} \} \wedge \{ \{ T_1 \mid N_1 \} \} \doteq S_2 \wedge \{ \{ T_3 \mid S_2 \} \} \doteq \{ \{ T_4 \mid S_1 \} \} & \xrightarrow{m2-m3-m3} \\ S_1 \doteq \{ \{ T_2 \mid N_1 \} \} \wedge S_2 \doteq \{ \{ T_1 \mid N_1 \} \} \wedge \{ \{ T_3, T_1 \mid N_1 \} \} \doteq \{ \{ T_4, T_2 \mid N_1 \} \}. \end{aligned}$$

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

More generally, it can be proved that

**Theorem 1.7 (Termination)** *Unify\_bags always terminates, for any input system  $\mathcal{E}$ .*

Moreover, soundness and completeness of this algorithm are assured by the following

**Theorem 1.8** *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\_bags 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  $T \vdash e \wedge \mathcal{E} \leftrightarrow \exists N_1, \dots, N_k \bigvee_{i=1}^h \mathcal{E}_i$ , where  $T$  is the theory of hybrid bags presented in § 2.2.*

The proof of Theorem 1.8 can be performed by case analysis. In particular one can observe that, since two bags are equal if and only if they contain the same number of occurrences of each element, then the problem  $\{ s_1, \dots, s_m \mid X \} \doteq \{ t_1, \dots, t_n \mid X \}$  is perfectly equivalent to the problem  $\{ s_1, \dots, s_m \} \doteq \{ t_1, \dots, t_n \}$  (action (m8)). Moreover, soundness and completeness of action (m7) follows from the strict analogy of this action with axiom ( $E_k^m$ ). (For the complete proofs of Theorems 1.7 and 1.8 see (Dovier, 1996).)

Before concluding this subsection, we want to point out an open problem for the bag unification problem presented above, namely: is there a unification algorithm for bags that incrementally computes a *minimal* complete set of unifiers?

Indeed, Unify\_bags is not minimal in general, since it may compute through non-determinism repeated equivalent solutions, as well as less general solutions. For example, Unify\_bags returns three repeated solutions for

```

function Unify_bags( $\mathcal{E}$ );
 $\mathcal{E}_2 := \emptyset$ ;
repeat
  repeat
    move the first equation  $e$  of  $\mathcal{E}_2$  (if any) to  $\mathcal{E}_1$ ;
    if  $e$  is not in solved form w.r.t.  $\mathcal{E}_1$  and  $\mathcal{E}_2$ 
      then Unify_bags_actions( $\mathcal{E}$ ,  $e$ )
  until  $\mathcal{E}_2 = \emptyset$ ;
  select arbitrarily from  $\mathcal{E}_1$  an equation  $e$  not in solved form;
  Unify_bags_actions( $\mathcal{E}$ ,  $e$ )
until  $\mathcal{E}_1$  is in solved form and  $\mathcal{E}_2 = \emptyset$ .

function Unify_bags_actions( $\mathcal{E}, e$ );
let  $\mathcal{E}'$  be  $\mathcal{E}_1 \setminus \{e\}$ ;
case  $e$  of
(m1)  $X \doteq X$   $\mapsto \mathcal{E}_1 := \mathcal{E}'$ 
(m2)  $\left. \begin{array}{l} t \doteq X \\ t \text{ is not a variable} \end{array} \right\} \mapsto \mathcal{E}_1 := X \doteq t \wedge \mathcal{E}'$ 
(m3)  $\left. \begin{array}{l} X \doteq t \\ X \text{ does not occur in } t \\ X \text{ occurs in } \mathcal{E}' \end{array} \right\} \mapsto$ 
 $\mathcal{E}_1 := \mathcal{E}'[X/t] \wedge X \doteq t; \mathcal{E}_2 := \mathcal{E}_2[X/t]$ 
(m4)  $\left. \begin{array}{l} X \doteq t \\ X \text{ occurs in } t \end{array} \right\} \mapsto \text{fail}$ 
(m5)  $f(s_1, \dots, s_m) \doteq g(t_1, \dots, t_m) \mapsto \text{fail}$ 
(m6)  $\left. \begin{array}{l} f(s_1, \dots, s_m) \doteq f(t_1, \dots, t_m) \\ f \in \Sigma \setminus \{\{\cdot | \cdot\}\} \end{array} \right\} \mapsto$ 
 $\mathcal{E}_1 := s_1 \doteq t_1 \wedge \dots \wedge s_m \doteq t_m \wedge \mathcal{E}'$ 
(m7)  $\left. \begin{array}{l} \{\{t | s\} \doteq \{\{t' | s'\}\} \\ \text{tail}(s) \text{ and } \text{tail}(s') \text{ are} \\ \text{not the same variable} \end{array} \right\} \mapsto$ 
(i)  $\mathcal{E}_1 := t \doteq t' \wedge \mathcal{E}'$ ; add  $s \doteq s'$  to  $\mathcal{E}_2$ 
(ii)  $\mathcal{E}_1 := \mathcal{E}'$ ; add  $s \doteq \{\{t' | N\}\}$  and  $\{\{t | N\}\} \doteq s'$  to  $\mathcal{E}_2$ 
(m8)  $\left. \begin{array}{l} \{\{t | s\} \doteq \{\{t' | s'\}\} \\ \text{tail}(s) \text{ and } \text{tail}(s') \text{ are} \\ \text{the same variable} \end{array} \right\} \mapsto$ 
 $\mathcal{E}_1 := \mathcal{E}'$ ; add  $\text{de\_tail}(\{\{t | s\}\}) \doteq \text{de\_tail}(\{\{t' | s'\}\})$  to  $\mathcal{E}_2$ .

```

Figure 1. Bag unification algorithm

$\{\{A, A | S\}\} \doteq \{\{A | S'\}\}$ , whereas the complete set of unifiers can be described by the unique solution  $S' = \{\{A | S\}\}$ . On the other hand, there are problems for which Unify\_bags can be proved to compute exactly the min-

imal complete set of unifiers (e.g.,  $\{\{A_1, \dots, A_m \mid S\} \doteq \{\{B_1, \dots, B_n \mid S'\}\}$ ,  $A_i$ s,  $B_j$ s,  $S$  and  $S'$  pairwise distinct variables, is such a problem).

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. This type of improvement, very important from a “practical” point of view, is not considered here for space limits. A first contribution for a solution—limited to the set unification problem but easily adaptable to other contexts—is presented in (Arenas and Dovier, 1995). Such optimizations, however, are independent from the non-deterministic complexity analysis of the algorithms, which is another interesting line of research.

### 3.2. UNIFICATION OF HYBRID COMPACT-LISTS

As shown in § 2.3, axiom  $(F_3)$  should be modified accordingly to the semantics of the compact-list constructor  $\llbracket \cdot \mid \cdot \rrbracket$ . This is reflected into the occur-checks performed by the compact-list unification algorithm shown in Figure 2 (actions  $(c4)$ — $(c6)$ ):

The other important difference w.r.t. standard list unification is the addition of action  $(c9)$  which reflects the extensionality principle for compact-lists expressed by axiom  $(E_k^c)$ .

Termination, soundness and completeness of the unification algorithm for compact-lists shown in Figure 2 has been proved in (Dovier, 1996).

### 3.3. UNIFICATION OF HYBRID SETS

The main (deterministic) part of the unification algorithm for hybrid sets is exactly the same as that of the unification algorithm for bags of Figure 1 (clearly calls to `Unify_bag_actions` are replaced by calls to `Unify_sets_actions`) and is not repeated here. The remaining part of the algorithm (function `Unify_sets_actions`—see Figure 3) is in a sense a combination of the unification algorithms for bags and for compact-lists shown in the previous two subsections.

As done with bags, we find 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 a little deterministic control to the algorithm in order to assure its termination.

The most important difference with the algorithms presented so far are actions  $(s9)$  and  $(s10)$ , whose aim is the reduction of set-set equations. In particular, cases  $(ii)$  and  $(iii)$  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  $\{\cdot \mid \cdot\}$ .

function Unify\_clists( $\mathcal{E}$ );  
 (c1)—(c3) as (l1)—(l3) in Unify\_lists

(c4)	$\left. \begin{array}{l} X \doteq t \\ t \text{ is not a compact-list} \\ \text{and } X \text{ occurs in } t \end{array} \right\} \mapsto \text{fail}$
(c5)	$\left. \begin{array}{l} X \doteq \llbracket t_0, \dots, t_n \mid t \rrbracket \wedge \mathcal{E} \\ t \text{ is a variable or } \llbracket \cdot \rrbracket \\ \text{and } X \text{ occurs in } t_i, 0 \leq i \leq n \end{array} \right\} \mapsto \text{fail}$
(c6)	$\left. \begin{array}{l} X \doteq \llbracket t_0, \dots, t_n \mid X \rrbracket \wedge \mathcal{E} \\ X \text{ does not occur in } t_0, \dots, t_n \end{array} \right\} \mapsto$
(c7)	$\left. \begin{array}{l} (Y \doteq t_0 \wedge \dots \wedge Y \doteq t_n \wedge \mathcal{E})[X/\llbracket Y \mid Z \rrbracket] \wedge X \doteq \llbracket Y \mid Z \rrbracket \\ f(s_1, \dots, s_m) \doteq g(t_1, \dots, t_n) \wedge \mathcal{E} \\ f \text{ different from } g \end{array} \right\} \mapsto \text{fail}$
(c8)	$\left. \begin{array}{l} f(s_1, \dots, s_m) \doteq f(t_1, \dots, t_m) \wedge \mathcal{E} \\ f \in \Sigma \setminus \{\llbracket \cdot \mid \cdot \rrbracket\} \end{array} \right\} \mapsto$
(c9)	$\left. \begin{array}{l} s_1 \doteq t_1 \wedge \dots \wedge s_m \doteq t_m \wedge \mathcal{E} \\ \llbracket t \mid s \rrbracket \doteq \llbracket t' \mid s' \rrbracket \wedge \mathcal{E} \end{array} \right\} \mapsto$ <ul style="list-style-type: none"> <li>(i) <math>t \doteq t' \wedge s \doteq s' \wedge \mathcal{E}</math></li> <li>(ii) <math>t \doteq t' \wedge s \doteq \llbracket t' \mid s' \rrbracket \wedge \mathcal{E}</math></li> <li>(iii) <math>t \doteq t' \wedge \llbracket t \mid s \rrbracket \doteq s' \wedge \mathcal{E}</math>.</li> </ul>

Figure 2. Compact-list unification algorithm

Equations of the form  $\{t_0, \dots, t_m \mid X\} \doteq \{t'_0, \dots, t'_n \mid X\}$ , where the two sides are set terms with the same variable tail element, are handled as a special case by action (s10). The problem is the same singled out in § 3.1. 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  $\{a, b \mid X\} \doteq \{b \mid X\}$ , which has no solution as a bag 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 (Arenas and Dovier, 1995) it is shown how to improve the algorithm from this point of view.

The algorithm of Figure 3 is the very same algorithm used in the language  $\{\log\}$ , a logic programming language enriched with finite sets (Dovier, Omodeo, Pontelli, and Rossi, 1991; Dovier, Omodeo, Pontelli, and Rossi,

```

function Unify_sets_actions( $\mathcal{E}$ ,  $e$ );
  let  $\mathcal{E}'$  be  $\mathcal{E}_1 \setminus \{e\}$ ;
  case  $e$  of
    (s1)—(s3) as (l1)—(l3) in Unify_bags
    (s4)    $X \doteq t$ 
            $t$  is not a set term and  $X$  occurs in  $t$  }  $\mapsto$  fail
    (s5)    $X \doteq \{t_0, \dots, t_n \mid t\}$ 
            $t$  is a member-less and  $X$  occurs in  $t$ 
           or  $X \in \text{vars}(t_0) \cup \dots \cup \text{vars}(t_n)$  }  $\mapsto$  fail
    (s6)    $X \doteq \{t_0, \dots, t_n \mid X\}$ 
            $X$  does not occur in  $t_0, \dots, t_n$  }  $\mapsto$ 
            $\mathcal{E}_1 := \mathcal{E}'$ ; add  $X \doteq \{t_0, \dots, t_n \mid N\}$  to  $\mathcal{E}_2$ 
    (s7)    $f(s_1, \dots, s_m) \doteq g(t_1, \dots, t_n)$ 
            $f$  different from  $g$  }  $\mapsto$  fail
    (s8)    $f(s_1, \dots, s_m) \doteq f(t_1, \dots, t_m)$ 
            $f \in \Sigma\{\{\cdot \mid \cdot\}\}$  }  $\mapsto$ 
            $\mathcal{E}_1 := s_1 \doteq t_1 \wedge \dots \wedge s_m \doteq t_m \wedge \mathcal{E}'$ 
    (s9)    $\{t \mid s\} \doteq \{t' \mid s'\}$ 
           tail( $s$ ) and tail( $s'$ ) are
           not the same variable }  $\mapsto$ 
           (i)  $\mathcal{E}_1 := t \doteq t' \wedge \mathcal{E}'$ ; add  $s \doteq s'$  to  $\mathcal{E}_2$ 
           (ii)  $\mathcal{E}_1 := t \doteq t' \wedge \mathcal{E}'$ ; add  $\{t \mid s\} \doteq s'$  to  $\mathcal{E}_2$ 
           (iii)  $\mathcal{E}_1 := t \doteq t' \wedge \mathcal{E}'$ ; add  $s \doteq \{t' \mid s'\}$  to  $\mathcal{E}_2$ 
           (iv)  $\mathcal{E}_1 := \mathcal{E}'$ ; add  $s \doteq \{t' \mid N\}$  and
                 $\{t \mid N\} \doteq s'$  to  $\mathcal{E}_2$ 
    (s10)   $\{t_0, \dots, t_m \mid X\} \doteq \{t'_0, \dots, t'_n \mid X\}$ 
            $X$  variable }  $\mapsto$ 
           select arbitrarily  $i$  in  $\{0, \dots, m\}$ ; choose one of the following actions:
           (i)  $\mathcal{E}_1 := t_0 \doteq t'_i \wedge \mathcal{E}'$ ;
                add  $\{t_1, \dots, t_m \mid X\} \doteq \{t'_0, \dots, t'_{i-1}, t'_{i+1}, \dots, t'_n \mid X\}$  to  $\mathcal{E}_2$ 
           (ii)  $\mathcal{E}_1 := t_0 \doteq t'_i \wedge \mathcal{E}'$ ;
                add  $\{t_0, \dots, t_m \mid X\} \doteq \{t'_0, \dots, t'_{i-1}, t'_{i+1}, \dots, t'_n \mid X\}$  to  $\mathcal{E}_2$ 
           (iii)  $\mathcal{E}_1 := t_0 \doteq t'_i \wedge \mathcal{E}'$ ;
                add  $\{t_1, \dots, t_m \mid X\} \doteq \{t'_0, \dots, t'_n \mid X\}$  to  $\mathcal{E}_2$ 
           (iv)  $\mathcal{E}_1 := \mathcal{E}'$ ; add  $X \doteq \{t_0 \mid N\}$  and
                 $\{t_1, \dots, t_m \mid N\} \doteq \{t'_0, \dots, t'_n \mid N\}$  to  $\mathcal{E}_2$ .

```

Figure 3. Set unification algorithm

1996). In (Dovier, Omodeo, Pontelli, and Rossi, 1996) it is proved to terminate, and to be sound and complete.

$\{\log\}$  has been also reconsidered as an instance of the general CLP scheme (Dovier and Rossi, 1993). Also in this case, the algorithm of Figure 3 is used as a fundamental component (the one dealing with equality constraints) of the constraint satisfiability procedure.

#### 4. Conclusions and further researches

In the first part of the paper we have presented axiomatically specified first order theories of lists, bags, 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 simultaneously. 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 lists, bags, 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 (Omodeo and Policriti, 1994; Dovier and Rossi, 1993).

#### Acknowledgements

We are grateful to Eugenio G. Omodeo for his precious suggestions and comments in many parts of the paper.

#### References

- Arenas-Sánchez, P., and Dovier, A. Minimal Set Unification. In *Proc. Seventh Int'l Symp. on Programming Language Implementation and Logic Programming (1995)*, M. Hermenegildo and S. D. Swierstra, Eds., vol. 982 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, pp. 397–414.

- 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). Siena.
- Clark, K. L. Negation as Failure. In *Logic and Databases*, H. Gallaire and J. Minker, Eds. Plenum Press, 1978, pp. 293–321.
- Dovier, A. *Computable Set Theory and Logic Programming*. PhD thesis, Università degli Studi di Pisa, 1996. In preparation.
- Dovier, A., Omodeo, E. G., and Policriti, A. Hyperset constraint handling. Rr 21/94, Dipartimento di Matematica ed Informatica, Univ. di Udine, December 1994.
- Dovier, A., Omodeo, E. G., Pontelli, E., and Rossi, G. {log}: A Logic Programming Language with Finite Sets. In *Proc. Eighth Int'l Conf. on Logic Programming* (1991), K. Furukawa, Ed., The MIT Press, Cambridge, Mass., pp. 111–124.
- Dovier, A., Omodeo, E. G., Pontelli, E., and Rossi, G. Embedding Finite Sets in a Logic Programming Language. In *Selected papers from 3<sup>rd</sup> Int'l Workshop on Extension of Logic Programming* (1993), E. Lamma and P. Mello, Eds., vol. 660 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, Berlin, pp. 150–167.
- Dovier, A., Omodeo, E. G., Pontelli, E., and Rossi, G. {log}: A Language for Programming in Logic with Finite Sets. To appear in the *Journal of Logic Programming*, 1996.
- 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.
- 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.
- Lassez, J. L., Maher, M. J., and Marriot, K. Unification revisited. In *Lecture Notes in Computer Science* (1986), vol. 306.
- Lloyd, J. W. *Foundations of Logic Programming*. Springer-Verlag, Berlin, 1987. Second edition.
- Martelli, A., and Montanari, U. An efficient unification algorithm. *ACM Transactions on Programming Languages and Systems* 4 (1982), pp. 258–282.
- Omodeo, E. G., Policriti, A., and Rossi, G. Che genere di insiemi/multi-insiemi/iperinsiemi incorporare nella programmazione logica? In D. Saccà, Ed., *Proc. Eighth Italian Conference on Logic Programming* (1993), pp. 55–70.
- Parlamento, F., and Policriti, A. Decision Procedures for Elementary Sublanguages of Set Theory IX. Unsolvability of the Decision Problem for a Restricted Subclass of  $\delta_0$ -Formulas in Set Theory. *Communications of Pure and Applied Mathematics* 41 (1988), pp. 221–251.
- Paterson, M. S., and Wegman, M. N. Linear unification. Tech. rep., IBM Thomas J. Watson Research Center, Yorktown Heights, 1976.
- Tarski, A., and Givant, S. *A Formalization of Set Theory without Variables*, vol. 41 of *Colloquium Publications*. American Mathematical Society, 1986.
- Vaught, R. L. On a Theorem of Cobham Concerning Undecidable Theories. In *Proceedings of the 1960 International Congress* (1962), E. Nagel, P. Suppes, and A. Tarski, Eds., Stanford University Press, Stanford, pp. 14–25.