# Constraint Programming with Multisets

Zeynep Kiziltan[1] and Toby Walsh[2]

[1] Department of Information Science, Uppsala University, Uppsala, Sweden
Zeynep.Kiziltan@dis.uu.se
[2] Cork Constraint Computation Center, University College Cork, Ireland
tw@4c.ucc.ie

**Abstract.** We propose extending constraint solvers with multiset variables. That is, variables whose values are multisets. Such an extension can help prevent introducing unnecessary symmetry into a model. We identify a number of different representations for multiset variables, and suggest primitive and global constraints on multiset variables. Surprisingly, unlike finite domain variables, decomposition of global constraints on multiset variables often does not hinder constraint propagation. We also study in detail the multiset ordering constraint. This constraint is useful for breaking symmetry between multiset variables. We show how it can be enforced using a simple lexicographical ordering constraint.

## 1 Introduction

Dealing efficiently and effectively with symmetry is one of the major difficulties in constraint programming. Symmetry occurs in many scheduling, assignment, routing and supply chain problems. In addition to the symmetry inherently present in a problem, modelling may introduce additional and, in some cases, unnecessary symmetry. Consider, for example, the template design problem (prob002 in CSPLib) in which we assign designs to printing templates. As there are a fixed number of slots for designs on each template, we can model this problem with a variable for each slot, whose value is the design assigned to this slot. However, slots on a template are essentially indistinguishable. This model therefore introduces an unnecessary symmetry, namely the permutations of the slots. A "better" model would remove this symmetry by having a variable for each template, whose value is the multiset of designs assigned to that template. It is a multiset, not a set, as the designs on a template are often repeated. This second model still introduces a symmetry as the templates are indistinguishable. However, as we show in this paper, orderings on multisets can be used to break such symmetries.

Set variables have been incorporated into all the major constraint solvers (see, for example, [6, 8]). It is therefore surprising that there is little if no work on multiset variables in constraint programming. Our goal is to correct this imbalance. The paper is structured as follows. We first introduce some notation and formal background (Section 2). We then discuss how to represent multiset variables (Section 3) and constraints on them (Section 4). In Section 5 we introduce the multiset ordering, and discuss how to enforce this as a constraint (Section 6). Such a constraint is useful for breaking symmetry in a range of problems. Finally, we end with future work and conclusions.

## 2 Formal background

We will need vectors, sets and multisets. A vector is an ordered list of elements, written $\langle x_0, \ldots x_{n-1}\rangle$. A set is an unordered list of elements without repetition, written $\{x_0, \ldots x_{n-1}\}$. A multiset is an unordered list of elements in which repetition is allowed, written $\{\!\{x_0, \ldots x_{n-1}\}\!\}$. To simplify notation, we assume that the elements of vectors, sets and multisets are integers drawn from some finite domain $[0, d)$. Basic operations on sets generalize to similar operations on multisets. We let $occ(x, M)$ be the number of occurrences of $x$ in the multiset $M$. Multiset union, intersection, difference, and equality are defined by the follow identities: $occ(x, M \cup N) = occ(x, M) + occ(x, N)$, $occ(x, M \cap N) = \min(occ(x, M), occ(x, N))$, $occ(x, M - N) = \max(0, occ(x, M) - occ(x, N))$, and $M = N$ iff $occ(x, M) = occ(x, N)$ for all $x$. For example, if $M = \{\!\{0, 1, 1\}\!\}$ and $N = \{\!\{0, 0, 1, 2\}\!\}$ then $M \cup N = \{\!\{0, 0, 0, 1, 1, 1, 2\}\!\}$, $M \cap N = \{\!\{0, 1\}\!\}$, $M - N = \{\!\{1\}\!\}$, and $M \neq N$. By ignoring the order of elements in a vector, we can view a vector as a multiset. For example, the vector $\langle 0, 1, 0\rangle$ can be viewed as the multiset $\{\!\{0, 0, 1\}\!\}$. We will abuse notation and write $\{\!\{\boldsymbol{m}\}\!\}$ for the multiset view of the vector $\boldsymbol{m}$.

We will need the strict lexicographical ordering relation on $d$-bit vectors. Formally, $\boldsymbol{m} <_{\text{lex}} \boldsymbol{n}$ iff $m_{d-1} \leq n_{d-1}$; $m_{d-2} \leq n_{d-2}$ when $m_{d-1} = n_{d-1}$; $\ldots$; $m_0 < n_0$ when $m_{d-1} = n_{d-1}$, $m_{d-2} = n_{d-2}$, $\ldots$, and $m_1 = n_1$. Note we are treating the $d$th element of the vectors as the high bit.

## 3 Representing multisets

There are a number of ways we can represent variables whose values are multisets of values. The most naive method would be to have a finite domain variable whose values are all the possible multisets. However, this will be computationally intractable as the number of possible multisets can in general be exponential.

### 3.1 Bounds representation

One type of representation for multiset variables generalizes the upper and lower bound representation used for set variables in [6]. For each multiset variable, we maintain two multisets: a least upper and a greatest lower bound. The least upper bound is the smallest multiset containing all those values that can be in the multiset, whilst the greatest lower bound is the largest multiset containing all those values that must be in the multiset. Given a multiset variable $M$, we write $lub(M)$ and $glb(M)$ for the least upper and greatest lower bound respectively. For reasoning with such a representation, we can introduce the notion of multiset bounds consistency (BC). Given a constraint $C$ over multiset variables $X_1, \ldots X_n$, we write $sol(X_i)$ for the set of multiset values for $X_i$ which can be extended to the other variables to satisfy the constraint. That is,

$$sol(X_i) = \{m_i \mid C(m_1, \ldots, m_n) \ \wedge \ \forall j \ . \ glb(X_j) \subseteq m_j \subseteq lub(X_j)\}$$

**Definition 1.** *A constraint over multiset variables $X_1 \ldots X_n$ is BC iff for each $1 \leq i \leq n$, we have*

$$lub(X_i) = \bigcup_{m \in sol(X_i)} m \qquad \text{and} \qquad glb(X_i) = \bigcap_{m \in sol(X_i)} m$$

*A set of constraints is BC iff each constraint in the set is BC.*

The rules given in [6] for constructing bounds consistent upper and lower bounds for set variables should easily adapt to multiset variables. This representation is compact but carries the penalty of not being able to represent all forms of disjunction. Consider, for example, a multiset variable $M$ with two possible multiset values: $\{\!\{0\}\!\}$ or $\{\!\{1\}\!\}$. To represent this, we would need $lub(M) = \{\!\{0, 1\}\!\}$ and $glb(M) = \{\!\{\}\!\}$. However, this representation also permits $M$ to take the multiset values $\{\!\{\}\!\}$ and $\{\!\{0, 1\}\!\}$.

### 3.2 Occurrence representation

Set variables can also be represented by their characteristic function (a vector of $d$ Boolean variables, each of which indicates whether a particular value is in the set or not). A straightforward generalization to multisets is the dual occurrence vector. Each multiset variable $M$ can be represented by a vector $\boldsymbol{m}$ of $d$ finite domain variables with $m_i = occ(i, M)$. For reasoning with such a representation, we can apply consistency techniques like generalized arc consistency (GAC) or bounds consistency (BC) to the variables in these vectors. This representation is also compact but carries the penalty of not being able to represent all forms of disjunction. Consider again the example of a multiset variable $M$ with two possible multiset values: $\{\!\{0\}\!\}$ or $\{\!\{1\}\!\}$. To represent this, we would need an occurrence vector with $m_0 = \{0, 1\}$ (that is, the value 0 can occur zero times or once) and $m_1 = \{0, 1\}$ (that is, the value 1 can occur zero times or once). Like the bounds representation, this also permits $M$ to take the multiset values $\{\!\{\}\!\}$ and $\{\!\{0, 1\}\!\}$.

### 3.3 Fixed cardinality

Multiset variables of a fixed cardinality are common in a number of problems. For example, the template design problem can be modelled as finding a multiset of designs for each template which is of a fixed cardinality. In such situations, we can represent each of the finite elements of the multiset with a variable whose values are possible values for this multiset element. It may appear that this representation introduces symmetry into the problem (via permutations of these variables). This is not the case as we will be posting (non-binary) constraints on these variables which ignore their permutation. Of course, this may make it harder work to define and post constraints. To channel efficiently between the occurrence vector representation and this representation, we can use Regin's global cardinality constraint [9]. In addition, we can post the constraint that the sum of the values taken by the variables in the occurrence vector is the multiset cardinality. This representation is also compact but again carries the penalty of not being able to represent all forms of disjunction. Consider, for example, a multiset variable $M$ of cardinality 3 with two possible multiset values: $\{\!\{0,0,0\}\!\}$ or $\{\!\{1,1,1\}\!\}$. To represent this, we would need three variables: $M_1 = \{0,1\}$, $M_2 = \{0,1\}$ and $M_3 = \{0,1\}$. Each finite domain variable represents one of the possible elements of the multiset. However, this representation also permits $M$ to take the multiset values: $\{\!\{0,0,1\}\!\}$, and $\{\!\{0,1,1\}\!\}$.

If the multiset variables are not of a fixed cardinality but there are upper bounds on their maximum cardinality, we can use a similar representation to the fixed cardinality representation. We need, however, to introduce an additional value which represents no value being assigned to a particular finite domain variable. For example, suppose we have a multiset variable of maximum cardinality $n$ drawing elements from $(0, d)$. We can represent this with $n$ variables, each with a finite domain $[0, d)$. The additional value 0 represents no assignment. With the multiset ordering constraint (see Section 6), this additional value will be reasoned with transparently.

### 3.4 Comparing the representations

We can compare the expressivity of the different representations. We say that one representation is **as expressive** than another if it can represent the same sets of multiset values, **more expressive** if it is as expressive and there is one set of multiset values that it can represent that the other cannot, and **incomparable** if neither representation is as expressive as the other.

**Theorem 1.**

1. *The occurrence representation is more expressive than the bounds representation.*
2. *The fixed cardinality representation is incomparable to both the bounds and the occurrence representation.*

*Proof.* 1. Clearly the occurrence representation is as expressive as the bounds. Consider a multiset variable $M$ with two possible multiset values: $\{\!\{\}\!\}$ or $\{\!\{0,0\}\!\}$. This can be represented exactly with the occurrence variable $m_0 = \{0,2\}$. By comparison, a bounds representation would need $lub(M) = \{\!\{\}\!\}$ and $glb(M) = \{\!\{0,0\}\!\}$, and this permits $M$ to take the additional multiset value $\{\!\{0\}\!\}$.

2. Consider a multiset variable $M$ of cardinality 2 with six possible multiset values: $\{\!\{0,1\}\!\}$, $\{\!\{0,2\}\!\}$, $\{\!\{0,3\}\!\}$, $\{\!\{1,1\}\!\}$, $\{\!\{1,2\}\!\}$, or $\{\!\{1,3\}\!\}$. The fixed cardinality representation can represent this exactly with two finite domain variables $M_1 =$

$\{0,1\}$ and $M_2 = \{1,2,3\}$. Both the bounds and the occurrence representations of this set of multiset values would also permit $M$ to take the additional multiset value $\{\!\{2,3\}\!\}$.

Consider a multiset variable $M$ of cardinality 2 with three possible multiset values: $\{\!\{0,1\}\!\}$, $\{\!\{0,2\}\!\}$, or $\{\!\{1,2\}\!\}$. In the bounds representation, we need $lub(M) = \{\!\{0,1,2\}\!\}$ and $glb(M) = \{\!\{\}\!\}$. The only two element multisets between these bounds are exactly $\{\!\{0,1\}\!\}$, $\{\!\{0,2\}\!\}$, or $\{\!\{1,2\}\!\}$ as required. Similarly with an occurrence representation, we need $m_0 = m_1 = m_2 = \{0,1\}$. The only two element multisets between these bounds are again the required ones. A fixed cardinality representation cannot, on the other hand, represent this set of multiset values exactly. We would need two finite domain variables with, say, $M_1 = \{0,1\}$ and $M_2 = \{1,2\}$. These would permit $M$ to take the additional two element mutliset value $\{\!\{1,1\}\!\}$. □

Note that if we restrict the occurrence representation to maintain just bounds on the number of occurrences of a value in the multiset then we obtain a representation that is equally expressive as the original multiset bounds representation. We can also compare the amount of pruning constraint propagation performs on the different representations.

**Theorem 2.**

1. *BC on the occurrence representation is equivalent to BC on the bounds representation;*
2. *GAC on the occurrence representation is strictly stronger than BC on the occurrence representation;*
3. *BC on the bounds or occurrence representation is incomparable to BC on the fixed cardinality representation*
4. *GAC on the occurrence representation is incomparable to GAC on the fixed cardinality representation;*

*Proof.* 1. Suppose that a bounds representation of a constraint is BC. Consider any multiset variable in this constraint, $X$ with bounds $lub(X)$ and $glb(X)$. We can construct an equivalent occurrence representation. Suppose $a_{\max} = occ(a, lub(X))$ and $a_{\min} = occ(a, glb(X))$. Then we let the finite domain variable $X_a$ in the occurrence vector have a domain $[a_{\min}, a_{\max}]$. Consider $X_a = a_{\max}$. Then, from the definition of BC of the bounds representation and the generalized multiset union operator, there must be a satisfying solution to the constraint in which $occ(a, X) = a_{\max}$. If there are several, we choose one non-deterministically. This solution is support for the bounds consistency of the occurrence representation. A similar argument holds for $X_a = a_{\min}$. Hence, BC of the bounds representation implies BC of the occurrence representation. The proof reverses directly.

2. It is clearly at least as strong. To show strictness, consider 3 multiset variables, $M_1$ to $M_3$ with domains containing the multiset values $\{\!\{\}\!\}$, or $\{\!\{0,0\}\!\}$. Suppose we have a constraint that $distinct(M_1, M_2, M_2)$. This constraint is not GAC (indeed, it has no solution). In the bounds representation, we have $lub(M_i) = \{\!\{0,0\}\!\}$ and $glb(M_i) = \{\!\{\}\!\}$, and the constraint is BC.

3 & 4. Consider seven multiset variables $M$ of cardinality 2, each with six possible multiset values: $\{\!\{0,1\}\!\}$, $\{\!\{0,2\}\!\}$, $\{\!\{0,3\}\!\}$, $\{\!\{1,1\}\!\}$, $\{\!\{1,2\}\!\}$, or $\{\!\{1,3\}\!\}$. The fixed cardinality representation can represent these values exactly. However, the occurrence representation of these multiset values would also permit the additional multiset value $\{\!\{2,3\}\!\}$. Suppose that we have a constraint that all seven multiset variables are distinct. The occurrence representation is GAC (and hence BC) but the fixed cardinality representation is not BC (and hence not GAC).

Consider four multiset variables $M$ of cardinality 2 with three possible multiset values: $\{\!\{0,1\}\!\}$, $\{\!\{0,2\}\!\}$, or $\{\!\{1,2\}\!\}$. The bounds and occurrence representations can represent these values exactly. However, the fixed cardinality representation would permit each multiset variable to take the additional mutliset value $\{\!\{1,1\}\!\}$. Suppose that we have a constraint that all four multiset variables are distinct. The fixed cardinality representation is GAC (and hence BC) but the bounds and occurrence representation are not BC (and hence the occurrence representation is not GAC). □

Since BC on the bounds representation is equivalent to BC on the occurrence representation, in the rest of this paper we will write BC without specifying which representation is used. In addition, when we write GAC on multiset variables, we will assume the occurrence representation unless otherwise indicated.

## 4 Multiset constraints

We need to support a number of different constraints on multiset variables.

### 4.1 Primitive constraints

Multiset expressions are made from multiset variables, ground multisets including the empty multiset $\{\!\{\}\!\}$, and the function symbols: $\cup$, $\cap$ and $-$. Multiset constraints are constructed from the inclusion relation, $\subseteq$ and its negation, $\not\subseteq$. Multiset equality, strict inclusion, and membership constraints can all be implemented as defined relations:

$$\begin{aligned} M = N &\implies M \subseteq N \ \&\ N \subseteq M \\ M \subset N &\implies M \subseteq N \ \&\ N \not\subseteq M \\ x \in M &\implies \{\!\{x\}\!\} \subseteq M \end{aligned}$$

Another interesting extension is to graded constraints [6]. An example of a graded constraints is the multiset cardinality function $|M|$. A graded function, $f$ is a mapping onto the positive integers that satisfies the property $M \subset N$ implies $f(M) < f(N)$. A graded constraint restricts $f$ to be within some integer interval.

Such primitive constraints can be implemented via equality and inequality constraints on the associated occurrence vectors as follows:

$$\begin{aligned} M \subseteq N &\implies m_i \leq n_i \text{ for all } i \\ M \not\subseteq N &\implies m_i > n_i \text{ for some } i \end{aligned}$$

$$
\begin{aligned}
M \cup N &\implies m_i + n_i \\
M \cap N &\implies \min(m_i, n_i) \\
M - N &\implies \max(0, m_i - n_i) \\
l \leq |M| \leq u &\implies l \leq \sum_i m_i \leq u
\end{aligned}
$$

### 4.2 Global constraints

We can also define global (or non-binary) constraints on multiset variables. An important question about such global constraints is whether decomposition hurts. Consider a global constraint on finite domain variables like the all-different constraint. This can be decomposed into binary not-equals constraints. However, such decomposition tends to hinder constraint propagation. For instance, GAC on an all-different constraint is strictly stronger than arc-consistency (AC) on the decomposed binary not-equals constraints [5]. Surprisingly, similar results often do *not* hold for a number of global constraints involving multiset variables. This is good news. We can provide global constraints on multiset variables to help users compactly specify models. However, we do not need to develop complex algorithms for reasoning about them as is often the case with finite domain variables. We can simply decompose such global constraints into primitive constraints. The following results also map over to global constraints on set variables, where sets are implemented either via characteristic functions, or upper and lower bounds. Decomposition of global set constraints thus also often does not hinder constraint propagation. In the rest of this section, we give results to show that decomposition on the occurrence representation often does not hinder GAC. Very similar results can be given to show that decomposition on the occurrence or bounds representation does not hinder BC.

**Disjoint constraint** The constraint $disjoint([M_1, \ldots, M_n])$ ensures that the multiset variables are pairwise disjoint. This global constraint can be decomposed into the binary constraints: $M_i \cap M_j = \{\!\{\}\!\}$ for all $i \neq j$. Such decomposition does not appear to hinder constraint propagation.

**Theorem 3.** *GAC (resp. BC) on a disjoint constraint is equivalent to AC (resp. BC) on the binary decomposition.*

*Proof.* Clearly GAC (resp. BC) on a disjoint constraint is as strong as AC (resp. BC) on the decomposition. To show the reverse, suppose that the binary decomposition is AC (resp. BC). If the disjoint constraint is not GAC or BC then there must be at least two multiset variables, $M_i$ and $M_j$ with a value $k$ in common. That is, $m_{ik} \geq 1$ and $m_{jk} \geq 1$. However, in such a situation, the decomposed constraint $\min(m_{ik}, m_{jk}) = 0$ would neither be AC nor BC. □

**Partition constraint** The constraint $partition([M_1, \ldots, M_n], M)$ ensures that the multiset variables, $M_i$ are pairwise disjoint and union together to give $M$. By introducing new auxiliary variables, it can be decomposed into binary and union constraints of the form: $M_i \cap M_j = \{\!\{\}\!\}$ for all $i \neq j$, and $M_1 \cup \ldots \cup M_n = M$. Decomposition again causes no loss in pruning.

**Theorem 4.** *GAC (resp. BC) on a partition constraint is equivalent to GAC (resp. BC) on the decomposition.*

*Proof.* Clearly GAC (resp. BC) on a partition constraint is as strong as GAC (resp. BC) on the decomposition. To show the reverse, by Theorem 3, we need focus just on the union constraints. Suppose that the decomposition is GAC (resp. BC). If the partition constraint is not GAC or BC then there must be one value $k$ that does not occur frequently enough in the upper bounds of the multiset variables. But, in this case, the decomposed constraint (which is equivalent to $\sum_{i=1}^{n} m_{ik} = m_k$) would neither be GAC nor BC. □

This result continues to hold even if the union constraint is decomposed into the set of ternary union constraints by introducing new auxiliary variables: $M_1 \cup M_2 = M_{12}, M_{12} \cup M_3 = M_{13}, \ldots, M_{1n-1} \cup M_n = M$. We can also consider the non-empty partition constraint which ensures we have a partition and that each multiset variable is not the empty multiset. Decomposition now appears to hinder constraint propagation.

**Theorem 5.** *GAC (resp. BC) on a non-empty partition constraint is strictly stronger than GAC (resp. BC) on the decomposition.*

*Proof.* Clearly it is at least as strong. To show strictness, consider 3 multiset variables with $glb(M_1) = glb(M_2) = glb(M_3) = \{\!\{\}\!\}$, $lub(M_1) = lub(M_2) = \{\!\{1, 2\}\!\}$ and $lub(M_3) = \{\!\{1, 2, 3\}\!\}$. The decomposition is both GAC and BC. However, enforcing GAC or BC on the non-empty partition constraint gives $glb(M_3) = lub(M_3) = \{\!\{3\}\!\}$. □

**Distinct constraint** Consider the constraint $distinct([M_1, \ldots, M_n])$ which ensures that all the multisets are distinct from each other. This decomposes into pairwise not equals constraints: $M_i \neq M_j$ for all $i \neq j$. Decomposition in this case appears to hinder constraint propagation.

**Theorem 6.** *GAC (resp. BC) on a distinct constraint is strictly stronger than AC (resp. BC) on the decomposition.*

*Proof.* Clearly it is at least as strong. To show strictness, consider a distinct constraint on 3 multiset variables with $glb(M_1) = glb(M_2) = \{\!\{\}\!\}$, $lub(M_1) = lub(M_2) = \{\!\{0\}\!\}$, $glb(M_3) = \{\!\{0\}\!\}$, and $lub(M_3) = \{\!\{0, 0\}\!\}$. The decomposition is both AC and BC. But enforcing GAC or BC on the distinct constraint gives $glb(M_3) = lub(M_3) = \{\!\{0, 0\}\!\}$. □

## 5 Ordering multisets

We will often want to order multiset variables. For example, in the template design problem we can break the symmetry between templates by insisting that their multiset values are ordered. Another application (suggested by Alan Frisch and since also considered in [7]) is for breaking row and column symmetry in matrix models [1]. Many problems can be modelled by matrices of decision variables in which the rows and/or columns are indistinguishable and can be permuted. Whilst in theory we can eliminate such symmetry using techniques like SBDS [3], in practice there are often too many symmetries to deal with them exhaustively. One strategy is to use SBDS to eliminate all column symmetry, and then add symmetry breaking constraints to eliminate some (but perhaps not all) the row symmetry. Since we can no longer freely permute the columns, whatever symmetry breaking constraints we add must be invariant to column permutation. We can, for instance, insist that the row sums are non-decreasing [4]. A stronger symmetry breaking constraint is to insist that the rows, when viewed as multisets, are ordered. This is stronger because two vectors may have the same sum, whereas two vectors treated as multisets are never equal unless they are identical. For instance, the vectors $\langle 3, 3, 2, 1\rangle$ and $\langle 3, 3, 3, 0\rangle$ have the same sum, but when viewed as multisets they are different. Ordering constraints on multisets are therefore very useful.

We write $\max(M)$ for the maximum element of a multiset $M$. This presuppose a total ordering on elements of the multiset. Such an ordering induces a total ordering on multisets. Formally, $M \prec_m N$ iff $x = \max(M)$, $y = \max(N)$ and:

$$x < y \ \vee \ (x = y \ \wedge \ M - \{\!\{x\}\!\} \prec_m N - \{\!\{y\}\!\})$$

That is, either the largest element in $M$ is smaller than the largest element in $N$, or they are the same and we eliminate this occurrence and recurse. This is called the **multiset ordering**. We can derive almost identical results if we weaken the ordering to include multiset equality (that is, for the ordering relation $M \preceq_m N$ defined by $M = N$ or $M \prec_m N$). As the following theorem shows, the multiset ordering is equivalent to the lexicographical ordering on the associated occurrence vectors. As we have efficient algorithms for reasoning about occurrence constraints [9] and about lexicographical ordering constraints [2], this may be the most effective route to reasoning about multiset ordering constraints.

**Theorem 7.** *$M \prec_m N$ iff $\boldsymbol{m} <_{\text{lex}} \boldsymbol{n}$.*

*Proof.* Suppose $M \prec_m N$. There are two cases. In the first case, $\max(M) < \max(N)$. Then $n_{\max(N)} > 0$ and for all $i \geq \max(N)$, we have $m_i = 0$. Hence, $\boldsymbol{m} <_{\text{lex}} \boldsymbol{n}$. In the second case, $\max(M) = \max(N) = a$. Then either $occ(a, M) = occ(a, N)$ or $occ(a, M) < occ(a, N)$. In the first subcase, we can delete all occurrences of $a$ from $M$ and $N$ and recurse. In the second subcase, $m_a < n_a$ and for all $i > a$ we have $m_i = n_i = 0$. Hence, $\boldsymbol{m} <_{\text{lex}} \boldsymbol{n}$. The proof reverses easily. □

One special case that will concern us is ordering 0/1 vectors viewed as multisets. This occurs when we are dealing with row and column symmetry in a matrix model of Boolean decision variables. As the following theorem demonstrates, the multiset ordering then reduces to ordering the vector sum. As bounds consistency techniques will reason about such sums quickly and effectively, we need not consider multiset orderings when dealing with 0/1 vectors.

**Theorem 8.** *On 0/1 vectors $\boldsymbol{m}$ and $\boldsymbol{n}$, $\{\!\{\boldsymbol{m}\}\!\} \prec_m \{\!\{\boldsymbol{n}\}\!\}$ iff $\sum_i m_i < \sum_i n_i$.*

*Proof.* Suppose $\{\!\{\boldsymbol{m}\}\!\} \prec_m \{\!\{\boldsymbol{n}\}\!\}$. There are two cases. In the first case, $1 \not\in \{\!\{\boldsymbol{m}\}\!\}$ and $1 \in \{\!\{\boldsymbol{n}\}\!\}$. Then $(\sum_i m_i = 0) < (\sum_i n_i > 0)$. In the second case, $1 \in \{\!\{\boldsymbol{m}\}\!\}$ and $1 \in \{\!\{\boldsymbol{n}\}\!\}$. Then $occ(1, \{\!\{\boldsymbol{m}\}\!\}) < occ(1, \{\!\{\boldsymbol{n}\}\!\})$ and $\sum_i m_i < \sum_i n_i$. The proof again reverses easily. □

## 6 Multiset ordering constraints

How do we enforce multiset ordering constraints? If multisets are being represented by occurrence vectors then by Theorem 7 we can simply post a lexicographic ordering constraint on these occurrence vectors and use the linear GAC algorithm given in [2]. If, however, multisets are being represented using the fixed cardinality representation, we would first have to channel into occurrence vectors with cardinality constraints. This would need to occur when, for example, we are symmetry breaking in a matrix model by viewing rows or columns as fixed cardinality multisets. Such channeling can be done efficiently using the polynomial GAC algorithm for cardinality constraints given in [9]. However, as the following theorem demonstrates, such channelling hinders constraint propagation.

**Theorem 9.** *GAC on a multiset ordering constraint represented using the fixed cardinality representation is strictly stronger than GAC applied simultaneously to a lexicographical ordering constraint on the occurrence vectors and to cardinality constraints between the fixed cardinality and occurrence representations.*

*Proof.* Clearly it is as strong. To show strictness, consider two multiset variables in the fixed cardinality representation, each of which contains four elements:

$$M_1 = \{\!\{\{1,2\},\{1,2\},\{2\},\{2\}\}\!\}$$
$$M_2 = \{\!\{\{1,2\},\{1,2\},\{0,1,2\},\{0,1\}\}\!\}$$

A multiset ordering constraint, $M_1 \prec M_2$ on this representation is not GAC since the third finite domain variable in $M_2$ needs to be reduced to give:

$$M_2 = \{\!\{\{1,2\},\{1,2\},\{1,2\},\{0,1\}\}\!\}$$

The corresponding occurrence vectors are:

$$\boldsymbol{m_1} = \langle\{0\},\{0,1,2\},\{2,3,4\}\rangle$$
$$\boldsymbol{m_2} = \langle\{0,1,2\},\{0,1,2,3,4\},\{0,1,2,3\}\rangle$$

Enforcing GAC on the lexicographical ordering constraint gives:

$$\boldsymbol{m_1} = \langle\{0\},\{0,1,2\},\{2,3\}\rangle$$
$$\boldsymbol{m_2} = \langle\{0,1,2\},\{0,1,2,3,4\},\{2,3\}\rangle$$

This does not allow us to prune any values from the multisets. GAC on the fixed cardinality representation therefore does more pruning.

Note that we also know that the multisets are of cardinality 4. We could therefore have a combined lexicographical and vector sum constraint to ensure this. Enforcing GAC on such a combined constraint gives:

$$\boldsymbol{m_1} = \langle \{0\}, \{1, 2\}, \{2, 3\} \rangle$$
$$\boldsymbol{m_2} = \langle \{0, 1\}, \{0, 1, 2\}, \{2, 3\} \rangle$$

Even though we now know that the value 0 can only occur at most once in $M_2$, we cannot say where. We still therefore cannot do as much pruning as GAC on the fixed cardinality representation. $\square$

For multisets of fixed cardinality $n$, multiset ordering constraints $M \prec_m N$ can be enforced via the arithmetic constraint $n^{m_0} + \ldots n^{m_{n-1}} < n^{n_0} + \ldots n^{n_{n-1}}$ [7]. It is not hard to show that BC on such a constraint is equivalent to GAC on the original multiset ordering constraint. However, such an arithmetic constraint is only feasible for small $n$.

## 7 Future work and conclusions

We have proposed that constraint solvers be extended with multiset variables. That is, variables whose values are multisets. Such an extension will help prevent introducing unnecessary symmetry into models. We have identified a number of different representations for multiset variables, and suggested a set of primitive and global constraints on multiset variables. Surprisingly, unlike finite domain variables, decomposition of global constraints on multiset variables often does not hinder constraint propagation. We also studied in detail the multiset ordering constraint. This constraint is useful for breaking symmetry between multiset variables, and also breaking row and column symmetry in matrix models. We have shown how a multiset ordering can be enforced by lexicographically ordering the associated vectors of value occurrences. Although channelling into the occurrence representation hinders constraint propagation, we conjecture that the impact will not be great, and that this will prove an effective way of enforcing multiset ordering constraints. We are currently running experiments to test this thesis.

## References


1. P. Flener, A. Frisch, B. Hnich, Z. Kiziltan, I. Miguel, J. Pearson, and T. Walsh. Breaking row and column symmetry in matrix models. In *Proc. of CP'2002*. Springer, 2002.
2. A. Frisch, B. Hnich, Z. Kiziltan, I. Miguel, and T. Walsh. Global constraints for lexicographic orderings. In *Proc. of CP'2002*. Springer, 2002.
3. I. Gent and B. Smith. Symmetry breaking in constraint programming. In *Proc. of ECAI-2000*, pp 599–603. IOS Press, 2000.
4. I. Gent and B. Smith. Reducing symmetry in matrix models: Sbds vs. constraints. Technical Report APES-31-2001, APES Research Group, 2001. Available from http://www.dcs.st-and.ac.uk/ apes/apesreports.html.

5. I. Gent, K. Stergiou, and T. Walsh. Decomposable constraints. *Artificial Intelligence* 123:133–156, 2000.
6. C. Gervet. Conjunto: constraint logic programming with finite set domains. In *Proc. of the 1994 International Symposium on Logic Programming*, pp 339–358. MIT Press, 1994.
7. Z. Kiziltan and B.M. Smith. Symmetry-breaking constraints for matrix models. In *Proc. of SymCon'02, the CP'02 Workshop on Symmetry in Constraints*, 2002.
8. T. Müller and M. Müller. Finite set constraints in Oz. In *Proc. of 13th Logic Programming Workshop*, pp 104–115. Technische Universität München, 1997.
9. J. Régin. Generalized arc consistency for global cardinality constraints. In *Proc. of AAAI'96*, pp 209–215. AAAI Press/The MIT Press, 1996.