Multi-relations in Z: A cross between multi-sets and binary relations

Ian Hayes
Department of Computer Science,
University of Queensland,
St. Lucia, 4072
Australia

July 16, 2004

Abstract

Both the theories of binary relations and multi-sets (or bags) in Z have been usefully applied to software specification and development. In this paper we examine a useful theory — multi-relations — which is a cross between these two theories. One way of viewing relations is as sets of pairs. Here, by analogy, we view multi-relations as multi-sets of pairs, and we define multi-relation equivalents of most of the traditional operators defined on binary relations. Multi-relations can also be viewed as graphs or two-dimensional matrices (with indices over arbitrary sets).

The use of multi-relations is illustrated by specifying a bill-of-materials system. This provides a good example of the paradigm of building a suitable mathematical theory first and then developing a specification in terms of the theory.

Contents

1 Multi-relations

The development of mathematical theories to aid in the specification and development of software systems is an important strategy for simplifying both the description of such systems and reasoning about them. An important bonus is that an abstract mathematical theory can be reused for the specification and development of other systems requiring objects with similar abstract properties. This strategy has been expounded by Dahl [1] and Jones [7]. In this paper we continue the development of this approach by developing a theory of *multi-relations* for Z, which is a cross between the theories of binary relations and multi-sets (or bags).

Section 1.1 gives a brief review of binary relations in Z via an example of a bill-of-materials system. Sections 1.2 to 1.14 present the theory of multi-relations following a pattern of development similar to that used for binary relations in Z. A binary relation is viewed as a set of pairs. By analogy, we view multi-relations as multi-sets of pairs, and develop multi-relation equivalents of most of the operators defined for binary relations. Multi-relations can also be viewed as graphs or two-dimensional integer matrices (with indices over arbitrary sets). During the development we point out the relationship between multi-relation and matrix operators. For the most part we have stuck to notation derived from the Z binary-relation notation, rather than using matix notation. This is to emphasise the close relationship between multi-relations and binary relations.

Section 2 provides a case study of the use of multi-relations to develop a specification of a bill-of-materials system. This specification is not intended to be a specification of a real bill-of-materials system, but rather a realistic illustration of the utility of multi-relations for specifying such systems.

Throughout this paper we make use of the Z notation [3, 8] and make extensive use of the theory and notations of multi-sets developed in [4]. In [4] the more common term *bag* is used where, in this paper, we use the term *multi-set*; this is to emphasise the correspondence between multi-sets and multi-relations. In addition, [4] allows frequencies of occurrence of elements in bags to be negative; we also allow negative frequencies for multi-relations. For the definitions of multi-set operators see Appendix B.

1.1 An example: bill-of-materials

To motivate the development of multi-relations and to highlight their relationship to binary relations, we initially develop a simple bill-of-materials system using binary relations. This example comes from [5, pages 201–203] and [6, page 156].

The system has a two-level hierarchy: products are made from assemblies and assemblies from components. We use a binary relation $Assemblies$ to record the assemblies used in each product, and another relation $Components$ to record the components used in each assembly. These binary relations only record whether or not a component (assembly) is used in an assembly (product), not the number of components (assemblies) used in the assembly (product). Keeping track of the number of components (assemblies) used in an assembly (product) requires multi-relations.

A glossary of the notation used for binary relations in Z is given in Appendix A; for more detail on Z notation see [8].

$$Assemblies : Product \leftrightarrow Assembly$$
$$Components : Assembly \leftrightarrow Component$$

Given this model we can specify such things as the assemblies required for a product P:

$$Assemblies(\!| \{P\} |\!).$$

This is the image through the relation $Assemblies$ of the singleton set containing just P; it gives the set of assemblies related to P. Similarly, the components required for an assembly A are given by,

$$Components(\!| \{A\} |\!).$$

The set of components required for a product P is given by,

$$Components(\!| Assemblies(\!| \{P\} |\!) |\!).$$

The relation between products and components is given by,

$$Assemblies \fatsemi Components,$$

where '$\fatsemi$' is relation composition: a product P is related to a component C if and only if there exists an assembly A related to both P and C.

The above scheme can be generalised to an arbitrary number of levels by considering products, assemblies and components all to be parts and using a single *is a direct sub-part of* relation to represent the bill-of-materials (BOM) state.

$$BOM : Part \leftrightarrow Part$$

Given this relation the direct components of a product P are given by,

$$BOM(\!| \{P\} |\!),$$

and all sub-parts, direct and indirect, are given by,

$$BOM^+(\!| \{P\} |\!),$$

where BOM^+ is the transitive closure of BOM: it relates one part P to another Q if and only if there is a sequence of parts starting with P and ending with Q such that all pairs of adjacent parts in the sequence are related by BOM.

We can define the atomic parts as those that do not contain any sub-parts.

$$atomic : (Part \leftrightarrow Part) \rightarrow \mathbb{P}\, Part$$
$$\forall\, BOM : Part \leftrightarrow Part \bullet$$
$$\quad atomic\ BOM = Part \setminus (\mathrm{dom}\, BOM)$$

A relation that explodes out a part into its constituent atomic parts is given by the following.

$$explode : (Part \leftrightarrow Part) \rightarrow (Part \leftrightarrow Part)$$

$$\forall\, BOM : Part \leftrightarrow Part \bullet$$
$$explode\ BOM = BOM^+ \rhd (atomic\ BOM)$$

The above relations do not keep track of the number of assemblies required for a product, etc., only whether the assembly is required. To keep track of the frequency of occurrence of a sub-part within a part we introduce multi-relations.

1.2 Multi-relations in Z

With a set S one can determine whether or not a value x is in the set: $x \in S$; with a multi-set or bag B one can determine the frequency of occurrence of a value x in the multi-set: $B \sharp x$. With a binary relation R one can determine whether or not two values x and y are associated by the relation: $(x, y) \in R$; with a *multi-relation* MR one can determine the multiplicity of the association between the two values: $MR \sharp (x, y)$.

In Z, we can declare a relation R between the types X and Y by,

$$R : X \leftrightarrow Y.$$

One way of viewing a relation is as a set of pairs (its graph). In fact, this is the definition used for relations in Z:

$$X \leftrightarrow Y == \mathbb{P}(X \times Y),$$

where '$\mathbb{P}$' stands for powerset and '$\times$' is Cartesian product. By analogy, a multi-relation MR between X and Y can be declared by,

$$MR : X \mathrel{\widetilde{\leftrightarrow}} Y,$$

where '$X \mathrel{\widetilde{\leftrightarrow}} Y$' is defined to be all the bags of pairs with the first component from the set X and the second component from the set Y; see Appendix B for more detail on bag notation.

$$X \mathrel{\widetilde{\leftrightarrow}} Y == \mathrm{bag}(X \times Y).$$

We use the notational convention of accenting relation operators to form the equivalent multi-relation operators.

We also introduce the notation $X \mathrel{\widetilde{\leftrightarrow}}_+ Y$ to denote multi-relations with non-negative frequencies:

$$X \mathrel{\widetilde{\leftrightarrow}}_+ Y == \mathrm{bag}(X \times Y).$$

The empty multi-relation between X and Y is just the empty bag of pairs of X and Y:

$$\{\!|\ \ |\!\}[X \times Y].$$

Multi-relations can also be viewed as two-dimensional matrices with integer elements, although the indices of the matrices come from arbitrary (possibly infinite) sets. The empty multi-relation is equivalent to the zero matrix.

We return to our simple two-level bill-of-materials system. This time, however, we take into account frequency of occurrence by using multi-relations.

$$\begin{array}{|l} \mathit{Assemblies} : \mathit{Product} \mathrel{\widetilde{\leftrightarrow}} \mathit{Assembly} \\ \mathit{Components} : \mathit{Assembly} \mathrel{\widetilde{\leftrightarrow}} \mathit{Component} \end{array}$$

The number of A assemblies required for product P is given by

$$\mathit{Assemblies} \sharp (P, A),$$

and the number of C components for assembly A by

$$\mathit{Components} \sharp (A, C).$$

1.3 Conversion to/from multi-relations

Before we begin to define the multi-relation equivalents of operators on binary relations, let us define conversions between relations and multi-relations, as these will help us to see the correspondence between the two concepts. The function 'relof' extracts a relation from a multi-relation by ignoring the frequency, and 'mrelof' forms a multi-relation from a relation by giving related pairs a frequency of one. These are just specialisations of the bag operators 'setof' and 'bagof' (see Appendix B).

$$\begin{array}{l} \mathrm{relof}[X, Y] == \mathrm{setof}[X \times Y] \\ \mathrm{mrelof}[X, Y] == \mathrm{bagof}[X \times Y] \end{array}$$

The functions 'relof' and 'mrelof' are generic in the two sets X and Y, while 'setof' and 'bagof' are generic in a single set, which for these definitions is instantiated to the set $X \times Y$.

Laws Throughout the paper we give laws for each new multi-relation operator that show how it relates to other operators. The majority of these laws are the multi-relation equivalents of laws for binary relations or multi-sets. These laws are given without detailed explanation and may be skipped on first reading.

The conversion operators satisfy the following laws.

$$\begin{array}{l} \forall\, R : X \leftrightarrow Y;\ MR : X \mathrel{\widetilde{\leftrightarrow}} Y;\ x : X;\ y : Y \bullet \\ \quad ((x, y) \in (\mathrm{relof}\ MR) \Leftrightarrow MR \sharp (x, y) \neq 0) \wedge \\ \quad ((x, y) \in R \Rightarrow (\mathrm{mrelof}\ R) \sharp (x, y) = 1) \wedge \\ \quad ((x, y) \notin R \Rightarrow (\mathrm{mrelof}\ R) \sharp (x, y) = 0) \wedge \\ \quad \mathrm{relof}(\mathrm{mrelof}\ R) = R \wedge \\ \quad (MR \in X \mathrel{\widetilde{\leftrightarrow}}_+ Y \Rightarrow \mathrm{mrelof}(\mathrm{relof}\ MR) \sqsubseteq MR) \end{array}$$

The operator '$\sqsubseteq$' is bag containment: the frequency of occurrence of every element of the left bag is less than or equal to its frequency in the right bag. Note that the last law only holds for bags not containing elements with negative frequencies because a pair with a negative frequency is considered to be in the equivalent relation; in the sequel, many of the laws involving 'relof' and 'mrelof' have a similar restriction.

1.4 Composition

To determine the number of components required for a product taking into account all the assemblies required for the product, we can use the equivalent of relation composition:

$$Assemblies \mathbin{\widetilde{⨾}} Components,$$

which is of type $Product \leftrightsquigarrow Component$. For each product P and component C,

$$(Assemblies \mathbin{\widetilde{⨾}} Components) \sharp (P, C)$$

gives the number of C components required for the product P. This value is given by the sum over all assemblies required for P, of the number of assemblies required times the number of C components required for the assembly. For example, if *Assemblies* and *Components* have values,

$$\begin{aligned} Assemblies = {} & [\![(widget, watsit) \mapsto 2, (widget, thingo) \mapsto 1, \\ & (hanafran, watsit) \mapsto 3]\!] \land \\ Components = {} & [\![(watsit, screw) \mapsto 4, (watsit, rod) \mapsto 2, \\ & (thingo, screw) \mapsto 3, (thingo, rod) \mapsto 3]\!] \end{aligned}$$

then their composition $Assemblies \mathbin{\widetilde{⨾}} Components$ is,

$$\begin{aligned} & [\![(widget, screw) \mapsto 2 * 4 + 1 * 3, \\ & \quad (widget, rod) \mapsto 2 * 2 + 1 * 3, \\ & \quad (hanafran, screw) \mapsto 3 * 4, \\ & \quad (hanafran, rod) \mapsto 3 * 2]\!] \end{aligned}$$

The composition of the two (ordinary) relations $P : X \leftrightarrow Y$ and $Q : Y \leftrightarrow Z$ is denoted $P \fatsemi Q$, and relates all pairs (x, z) such that there is some $y \in Y$ such that P relates x to y, and Q relates y to z. The multi-relation equivalent $MP \mathbin{\widetilde{⨾}} MQ$ of two multi-relations $MP : X \mathrel{\widetilde{\leftrightarrow}} Y$ and $MQ : Y \mathrel{\widetilde{\leftrightarrow}} Z$ takes into account the frequencies of occurrence. If x is multi-related to y by MP n times, and y is multi-related to z by MQ m times, this contributes $n * m$ ways that x is multi-related to z via y. The total number of ways x is multi-related to z can be found by summing over all possible intermediate y values. For this to be well defined there must only be a finite number of intermediate values related to both x and z. The astute reader will recognise that multi-relation composition is matrix multiplication.

Below we give a generic definition in Z of multi-relation composition. It is generic in the sets $[X, Y, Z]$. The definition is given in two parts: the declaration of the type of the composition operator, '$\_ \mathbin{\widetilde{⨾}} \_$', and a predicate defining both the domain of definition of composition and the result of composing two multi-relations in that domain. The definitions of other multi-relation operators that follow take a similar form.

$$
\begin{array}{|l}
\hline
[X, Y, Z] \\
\_ \mathbin{\widetilde{\circ}} \_ : (X \mathbin{\widetilde{\leftrightarrow}} Y) \times (Y \mathbin{\widetilde{\leftrightarrow}} Z) \pfun (X \mathbin{\widetilde{\leftrightarrow}} Z) \\
\hline
\forall\, MP : X \mathbin{\widetilde{\leftrightarrow}} Y;\ MQ : Y \mathbin{\widetilde{\leftrightarrow}} Z;\ x : X;\ z : Z \bullet \\
\quad ((MP, MQ) \in \dom(\_ \mathbin{\widetilde{\circ}} \_) \Leftrightarrow (\forall\, x : X;\ z : Z \bullet \\
\qquad \{y : Y \mid MP \sharp (x, y) \neq 0 \land MQ \sharp (y, z) \neq 0\} \in \mathbb{F}\, Y)) \land \\
\quad ((MP, MQ) \in \dom(\_ \mathbin{\widetilde{\circ}} \_) \Rightarrow \\
\qquad (MP \mathbin{\widetilde{\circ}} MQ) \sharp (x, z) = \\
\qquad\quad \sum [\![y : Y \mid MP \sharp (x, y) \neq 0 \land MQ \sharp (y, z) \neq 0 \bullet \\
\qquad\qquad MP \sharp (x, y) * MQ \sharp (y, z)]\!]) \\
\hline
\end{array}
$$

The notation $\mathbb{F}\, Y$ stands for the set of all finite subsets of Y. The notation in the brackets '$[\![$' and '$]\!]$' is called bag comprehension and stands for the bag of values of the expression after the '$\bullet$' for y ranging over all values in Y such that the predicate after the '$|$' holds (see Appendix B). A bag is required here as there may be multiple values of y for which the expression has the same value.

To see the correspondence to binary relations, replace '$*$' by '$\land$' and summation over the bag by an existential quantifier.

1.5 Finitary multi-relations

The condition for multi-relation composition to be well-defined given above is the most general. However, it is complicated to use in practice as it is a property of pairs of multi-relations rather than individual multi-relations. (For example, the interested reader might like to determine the most general condition required on the three multi-relations MP, MQ and MR for the associative law,

$$(MP \mathbin{\widetilde{\circ}} MQ) \mathbin{\widetilde{\circ}} MR = MP \mathbin{\widetilde{\circ}} (MQ \mathbin{\widetilde{\circ}} MR),$$

to be well-defined.)

A simpler, although more restrictive, approach is to require the multi-relations involved to all satisfy a property individually. A suitable property is that for each x value there are only finitely many y values for which $MR \sharp (x, y)$ is non-zero and for each y there are only finitely many x values for which $MR \sharp (x, y)$ is non-zero. A multi-relation with this property is referred to as being *finitary*. If we think of a multi-relation as a matrix then each row and each column has only finitely many non-zero entries. We introduce the new notation $X \mathbin{\widetilde{\leftrightarrow}_f} Y$ to stand for the set of all finitary multi-relations between X and Y.

$$
\begin{array}{l}
X \mathbin{\widetilde{\leftrightarrow}_f} Y == \{MR : X \mathbin{\widetilde{\leftrightarrow}} Y \mid \\
\qquad (\forall\, x : X \bullet \{y : Y \mid MR \sharp (x, y) \neq 0\} \in \mathbb{F}\, Y) \land \\
\qquad (\forall\, y : Y \bullet \{x : X \mid MR \sharp (x, y) \neq 0\} \in \mathbb{F}\, X)\}
\end{array}
$$

Laws We now give laws for multi-relation composition on these *finitary* multi-relations. Multi-relation composition is well defined for any two finitary multi-

relations, and the result of such a composition is also finitary.

$$\begin{array}{l}\forall\, MP : W \mathrel{\widetilde{\leftrightarrow}_f} X;\ MQ : X \mathrel{\widetilde{\leftrightarrow}_f} Y;\ MR : Y \mathrel{\widetilde{\leftrightarrow}_f} Z \bullet \\ \quad (MP, MQ) \in \mathrm{dom}(\_ \mathbin{\widetilde{⨾}} \_) \land \\ \quad (MP \mathbin{\widetilde{⨾}} MQ) \in (W \mathrel{\widetilde{\leftrightarrow}_f} Y) \land \\ \quad (MP \mathbin{\widetilde{⨾}} MQ) \mathbin{\widetilde{⨾}} MR = MP \mathbin{\widetilde{⨾}} (MQ \mathbin{\widetilde{⨾}} MR) \land \\ \quad (MP \in W \mathrel{\widetilde{\leftrightarrow}_+} X \land MQ \in X \mathrel{\widetilde{\leftrightarrow}_+} Y \Rightarrow \\ \qquad \mathrm{relof}(MP \mathbin{\widetilde{⨾}} MQ) = (\mathrm{relof}\ MP) \mathbin{⨾} (\mathrm{relof}\ MQ))\end{array}$$

1.6 The identity multi-relation

The multi-relation equivalent of the identity relation on a set S is the multi-relation that relates each element $x \in S$ to itself, exactly once.

$$[X]$$

$$\widetilde{\mathrm{id}} : \mathbb{P}\, X \to (X \mathrel{\widetilde{\leftrightarrow}} X)$$

$$\begin{array}{l}\forall\, S : \mathbb{P}\, X \bullet \\ \quad \widetilde{\mathrm{id}}\, S = [\![x : \mathrm{bagof}\ S \bullet (x, x)]\!]\end{array}$$

This corresponds to the identity matrix on S.

Laws

$$\begin{array}{l}\forall\, MR : Y \mathrel{\widetilde{\leftrightarrow}} Z \bullet \\ \quad MR \mathbin{\widetilde{⨾}} (\widetilde{\mathrm{id}}\, Z) = MR = (\widetilde{\mathrm{id}}\, Y) \mathbin{\widetilde{⨾}} MR \\ \forall\, S, T : \mathbb{P}\, X;\ x, y : X \bullet \\ \quad (x \in S \Rightarrow (\widetilde{\mathrm{id}}\, S) \sharp (x, x) = 1) \land \\ \quad ((x \notin S \lor x \neq y) \Rightarrow (\widetilde{\mathrm{id}}\, S) \sharp (x, y) = 0) \land \\ \quad \widetilde{\mathrm{id}}\, S \in (X \mathrel{\widetilde{\leftrightarrow}_f} X) \land \\ \quad (\widetilde{\mathrm{id}}\, S) \mathbin{\widetilde{⨾}} (\widetilde{\mathrm{id}}\, T) = \widetilde{\mathrm{id}}(S \cap T) \land \\ \quad \mathrm{relof}(\widetilde{\mathrm{id}}\, S) = \mathrm{id}\, S \land \\ \quad \mathrm{mrelof}(\mathrm{id}\, S) = \widetilde{\mathrm{id}}\, S\end{array}$$

1.7 Addition and multiplication

Continuing the product/assembly example, if there are two stages of production, then each stage could require a different collection of assemblies to produce a product.

$$stage1, stage2 : Product \mathrel{\widetilde{\leftrightarrow}} Assembly$$

The total assemblies required by the combined stages is the sum of the requirements for the two stages: $stage1 \uplus stage2$, where '$\uplus$' is bag addition (see Appendix B).

In producing an assembly we use a collection of machines a number of times, and each time we use a machine there is a cost associated with the use:

$$\begin{array}{l}machineuse : Assembly \mathrel{\widetilde{\leftrightarrow}} Machine \\ machinecost : Assembly \mathrel{\widetilde{\leftrightarrow}} Machine\end{array}$$

where for *machinecost* we use the frequency as the cost in cents. The cost for each assembly and each machine is the product of the number of times the machine is used and the cost of each use:

$$machineuse \circledast machinecost,$$

where '$\circledast$' is bag multiplication (see Appendix B).

Laws The following laws are the multi-relation specialisations of equivalent laws for multi-sets.

$$\begin{array}{l}
\forall\, MP, MQ, MR : X \mathrel{\widetilde{\leftrightarrow}} Y;\ x : X;\ y : Y \bullet \\
\quad (MP \uplus MQ) \sharp (x, y) = MP \sharp (x, y) + MQ \sharp (x, y) \land \\
\quad (MP \uplus MQ) \uplus MR = MP \uplus (MQ \uplus MR) \land \\
\quad MP \uplus MQ = MQ \uplus MP \land \\
\quad MP \uplus \{\!|\ \ |\!\} = MP = \{\!|\ \ |\!\} \uplus MP \land \\
\quad (MP \circledast MQ) \sharp (x, y) = MP \sharp (x, y) * MQ \sharp (x, y) \land \\
\quad (MP \circledast MQ) \circledast MR = MP \circledast (MQ \circledast MR) \land \\
\quad MP \circledast MQ = MQ \circledast MP \land \\
\quad MP \circledast \{\!|\ \ |\!\} = \{\!|\ \ |\!\} = \{\!|\ \ |\!\} \circledast MP \land \\
\quad MP \circledast (MQ \uplus MR) = (MP \circledast MQ) \uplus (MP \circledast MR) \\
\forall\, MP, MQ : X \mathrel{\widetilde{\leftrightarrow}}_+ Y \bullet \\
\quad \mathrm{relof}(MP \uplus MQ) = (\mathrm{relof}\ MP) \cup (\mathrm{relof}\ MQ) \land \\
\quad \mathrm{relof}(MP \circledast MQ) = (\mathrm{relof}\ MP) \cap (\mathrm{relof}\ MQ)
\end{array}$$

Viewing multi-relations as matrices, '$\uplus$' is matrix addition, and '$\circledast$' is pairwise multiplication of matrix elements.

1.8 Image

Given the multi-relation *Assemblies* from Section 1.2, which gives assembly counts for each product, and given a bag of products B, it is useful to determine the corresponding bag of assemblies required to produce the products. For this we introduce the multi-relation equivalent of relation image; the assemblies required are given by the expression,

$$Assemblies(\!|\ \widetilde{B}\ |\!).$$

The definition of multi-relation image follows.

$$\begin{array}{|l}
\hline
[X, Y] \\
(\!|\ \widetilde{}\ |\!) : (X \mathrel{\widetilde{\leftrightarrow}} Y) \times \mathrm{bag}\ X \nrightarrow \mathrm{bag}\ Y \\
\hline
\forall\, MR : X \mathrel{\widetilde{\leftrightarrow}} Y;\ B : \mathrm{bag}\ X;\ y : Y \bullet \\
\quad ((MR, B) \in \mathrm{dom}(_(\!|\ \widetilde{_}\ |\!)) \Leftrightarrow \\
\qquad (\forall\, y : Y \bullet \{x : \mathrm{setof}\ B \mid MR \sharp (x, y) \neq 0\} \in \mathbb{F}\, X)) \land \\
\quad ((MR, B) \in \mathrm{dom}(_(\!|\ \widetilde{_}\ |\!)) \Rightarrow \\
\qquad (MR(\!|\ \widetilde{B}\ |\!)) \sharp y = \textstyle\sum [\![x : B \mid MR \sharp (x, y) \neq 0 \bullet MR \sharp (x, y)]\!]) \\
\hline
\end{array}$$

For example, if *Assemblies* has the value,

$$[\![(widget, watsit) \mapsto 2, (widget, thingo) \mapsto 1, (hanafran, watsit) \mapsto 3]\!],$$

then $Assemblies(\!| \, [\![\widetilde{widget} \mapsto 2]\!] \, |\!)$ has the value,

$$[\![watsit \mapsto 4, thingo \mapsto 2]\!].$$

Again, as for multi-relation composition, the well-definedness condition is a property of the pair of multi-relation and bag, rather than a property of the multi-relation and bag individually. We can utilize the finitary property on multi-relations to provide simpler, although more restrictive, laws.

Laws

$$\begin{array}{l} \forall\, MR : X \mathrel{\widetilde{\leftrightarrow}}_f Y;\ B, C : \mathrm{bag}\, X \bullet \\ \quad (MR, B) \in \mathrm{dom}(\_(\!| \widetilde{\_} |\!)) \wedge \\ \quad MR(\!| \, B \mathrel{\widetilde{\uplus}} C \, |\!) = (MR(\!| \, \widetilde{B} \, |\!)) \uplus (MR(\!| \, \widetilde{C} \, |\!)) \wedge \\ \quad (MR \in X \mathrel{\widetilde{\leftrightarrow}}_+ Y \wedge B \in \mathrm{bag}\, X \Rightarrow \\ \qquad \mathrm{setof}(MR(\!| \, \widetilde{B} \, |\!)) = (\mathrm{relof}\, MR)(\!| \, \mathrm{setof}\, B \, |\!)) \end{array}$$

Alternatively, we could have chosen the bags to be finite.

Viewing multi-relations as matrices, multi-relation image is the product of a vector (the bag) and a matrix.

1.9 Iteration

If we take the multi-level view of the bill-of-materials system, we can consider products, assemblies and components all to be parts, and have a single multi-relation BOM giving the sub-part occurrence for all parts.

$$BOM : Part \mathrel{\widetilde{\leftrightarrow}} Part,$$

There is now no fixed limit on depth in the sub-part hierarchy, but we do require that there are no infinite chains (including cycles) in the sub-part hierarchy and hence a particular sub-part hierarchy does have a maximum depth. This issue is addressed in Section 1.10.

BOM gives, for each part, the sub-parts that it is made from. $BOM \mathbin{\widetilde{⨾}} BOM$ gives the sub-sub-parts, $BOM \mathbin{\widetilde{⨾}} BOM \mathbin{\widetilde{⨾}} BOM$ gives the sub-sub-sub-parts, etc. As with relations we can define iteration on homogeneous multi-relations. We only define iteration for finitary multi-relations.

$$\begin{array}{|l} [X] \\ \hline \widetilde{\_^{-}} : (X \mathrel{\widetilde{\leftrightarrow}}_f X) \times \mathbb{N} \to (X \mathrel{\widetilde{\leftrightarrow}}_f X) \\ \hline \forall\, MR : X \mathrel{\widetilde{\leftrightarrow}}_f X;\ n : \mathbb{N} \bullet \\ \quad MR^{\widetilde{0}} = \widetilde{\mathrm{id}}\, X \wedge \\ \quad MR^{\widetilde{(n+1)}} = MR \mathbin{\widetilde{⨾}} MR^{\widetilde{n}} \\ \hline \end{array}$$

For example, the sub-sub-part multi-relation can now be written $BOM^{\widetilde{2}}$.

Laws

$$\begin{aligned}
&\forall\, MR : X \mathrel{\widetilde{\leftrightarrow}_f} X;\ S : \mathbb{P}\, X;\ n, m : \mathbb{N} \bullet \\
&\quad MR^{\widetilde{1}} = MR \land \\
&\quad MR^{\widetilde{2}} = MR \mathbin{\widetilde{;}_9} MR \land \\
&\quad MR^{\widetilde{(n+m)}} = (MR^{\widetilde{n}}) \mathbin{\widetilde{;}_9} (MR^{\widetilde{m}}) \land \\
&\quad MR^{\widetilde{(n*m)}} = (MR^{\widetilde{n}})^{\widetilde{m}} \land \\
&\quad (n \geq 1 \Rightarrow (\widetilde{\mathrm{id}}\, S)^{\widetilde{n}} = \widetilde{\mathrm{id}}\, S) \land \\
&\quad (MR \in X \mathrel{\widetilde{\leftrightarrow}_+} X \Rightarrow \mathrm{relof}(MR^{\widetilde{n}}) = (\mathrm{relof}\ MR)^n)
\end{aligned}$$

Viewing multi-relations as matrices, iteration is raising a matrix to a power.

1.10 Transitive sum

If we want to determine the fully exploded part counts for all parts, we need to sum the sub-part, sub-sub-part, etc, multi-relations:

$$BOM \uplus BOM^{\widetilde{2}} \uplus BOM^{\widetilde{3}} \uplus \cdots$$

For example, if we have the direct sub-part multi-relation BOM,

$$[\![(widget, watsit) \mapsto 2, (widget, screw) \mapsto 3, (watsit, screw) \mapsto 4]\!],$$

then a widget requires two watsits each of which requires four screws, and in addition, a widget also directly requires three screws (no doubt to screw together the two watsits). BOM represents the direct sub-part multi-relation. The sub-sub-part multi-relation is given by $BOM^{\widetilde{2}}$:

$$[\![(widget, screw) \mapsto 8]\!].$$

The sub-sub-sub-part relationship is empty because widget is not a sub-part of any other part and screw has no sub-parts. Hence $BOM^{\widetilde{n}} = \{\!|\ \ |\!\}$ for $n \geq 3$. If we take the fully exploded view, then a widget requires a total of two times four plus three, i.e., eleven, screws. This corresponds to $BOM \uplus BOM^{\widetilde{2}}$:

$$[\![(widget, watsit) \mapsto 2, (widget, screw) \mapsto 11, (watsit, screw) \mapsto 4]\!].$$

We do not need to include any higher powers of BOM as they are all empty.

We need to insist that no part has itself as a sub-part either directly or indirectly, that is, there are no cycles in the dependency relationship, to avoid getting infinite part counts. If the set of parts is infinite, we also need to insist that there is no infinite chain of parts where each part is a sub-part of the previous part. Both these conditions are covered by requiring that for some $n : \mathbb{N}_1$ (and hence all values greater than n), $BOM^{\widetilde{n}} = \{\!|\ \ |\!\}$. A multi-relation satisfying this condition is referred to as being *nilpotent*; i.e., the matrix is nilpotent.

$$\begin{array}{|l}
[X] \\
\hline
\mathrm{nilpotent} : \mathbb{P}(X \mathrel{\widetilde{\leftrightarrow}_f} X) \\
\hline
\mathrm{nilpotent} = \{MR : X \mathrel{\widetilde{\leftrightarrow}_f} X \mid (\exists\, n : \mathbb{N}_1 \bullet MR^{\widetilde{n}} = \{\!|\ \ |\!\})\} \\
\hline
\end{array}$$

The sum $BOM \uplus BOM^{\widetilde{2}} \uplus \cdots$ is the multi-relation equivalent of the transitive closure of a relation, which we call the *transitive sum* (it is no longer a 'closure' so we avoid calling it that). The transitive sum is only well-defined provided the multi-relation is nilpotent, that is, there exists a power of the multi-relation which is the empty multi-relation.

$$
\begin{array}{|l}
[X] \\
\hline
\_^{\widetilde{+}} : (X \mathrel{\widetilde{\leftrightarrow}_f} X) \pfun (X \mathrel{\widetilde{\leftrightarrow}_f} X) \\
\_^{\widetilde{*}} : (X \mathrel{\widetilde{\leftrightarrow}_f} X) \pfun (X \mathrel{\widetilde{\leftrightarrow}_f} X) \\
\hline
\mathrm{dom}(\_^{\widetilde{+}}) = \text{nilpotent} \land \\
\mathrm{dom}(\_^{\widetilde{*}}) = \text{nilpotent} \land \\
(\forall\, MR : \text{nilpotent}[X] \bullet \\
\quad MR^{\widetilde{+}} = \biguplus [\![n : \mathbb{N}_1 \bullet MR^{\widetilde{n}}]\!] \land \\
\quad MR^{\widetilde{*}} = \biguplus [\![n : \mathbb{N} \bullet MR^{\widetilde{n}}]\!]) \\
\hline
\end{array}
$$

The operator '$\biguplus$' is distributed bag addition (see Appendix B). It combines a bag of bags into a single bag. In this case,

$$MR^{\widetilde{+}} = MR^{\widetilde{1}} \uplus MR^{\widetilde{2}} \uplus MR^{\widetilde{3}} \uplus \cdots$$

Laws

$$
\begin{array}{l}
\forall\, MR : \text{nilpotent}[X] \bullet \\
\quad MR^{\widetilde{*}} = \widetilde{\mathrm{id}}\, X \uplus MR^{\widetilde{+}} \land \\
\quad (MR \in X \mathrel{\widetilde{\leftrightarrow}_+} X \Rightarrow \\
\qquad \mathrm{relof}(MR^{\widetilde{+}}) = (\mathrm{relof}\, MR)^+ \land \\
\qquad \mathrm{relof}(MR^{\widetilde{*}}) = (\mathrm{relof}\, MR)^*)
\end{array}
$$

The matrix equivalents of nilpotent and transitive sum should be obvious.

1.11 Transpose

The transpose of a relation is the set of pairs in the relation with the first and second components swapped. The same holds for multi-relations.

$$
\begin{array}{|l}
[X, Y] \\
\hline
\_^{\widetilde{\sim}} : (X \mathrel{\widetilde{\leftrightarrow}} Y) \to (Y \mathrel{\widetilde{\leftrightarrow}} X) \\
\hline
\forall\, x : X;\ y : Y;\ MR : X \mathrel{\widetilde{\leftrightarrow}} Y \bullet \\
\quad MR^{\widetilde{\sim}} \sharp (y, x) = MR \sharp (x, y) \\
\hline
\end{array}
$$

Laws

$$
\begin{array}{l}
\forall\, MR : X \mathrel{\widetilde{\leftrightarrow}} Y;\ x : X;\ y : Y \bullet \\
\quad (MR^{\widetilde{\sim}})^{\widetilde{\sim}} = MR \land \\
\quad \mathrm{relof}(MR^{\widetilde{\sim}}) = (\mathrm{relof}\, MR)^\sim
\end{array}
$$

Viewing multi-relations as matrices, multi-relation transpose is matrix transpose.

1.12 Domain and range

The domain (range) of a relation is the set of first (second) components of the pairs in the relation.

$$\begin{array}{l}\operatorname{dom} R = \{x : X \mid (\exists\, y : Y \bullet (x, y) \in R)\} \land \\ \operatorname{ran} R = \{y : Y \mid (\exists\, x : X \bullet (x, y) \in R)\}\end{array}$$

We can generalise domain (range) to multi-relations by taking the domain (range) of the corresponding relation.

$$\begin{array}{|l}\hline [X, Y] \\ \hline \widetilde{\operatorname{dom}} : (X \mathrel{\widetilde{\leftrightarrow}} Y) \to \mathbb{P}\, X \\ \widetilde{\operatorname{ran}} : (X \mathrel{\widetilde{\leftrightarrow}} Y) \to \mathbb{P}\, Y \\ \hline \forall\, MR : X \mathrel{\widetilde{\leftrightarrow}} Y \bullet \\ \quad \widetilde{\operatorname{dom}}\, MR = \{x : X \mid (\exists\, y : Y \bullet MR \sharp (x, y) \neq 0)\} \land \\ \quad \widetilde{\operatorname{ran}}\, MR = \{y : Y \mid (\exists\, x : X \bullet MR \sharp (x, y) \neq 0)\} \\ \hline\end{array}$$

Laws

$$\begin{array}{l}\widetilde{\operatorname{dom}}(\{\!|\ \ |\!\}[X \times Y]) = \{\ \}[X] \land \\ \widetilde{\operatorname{ran}}(\{\!|\ \ |\!\}[X \times Y]) = \{\ \}[Y] \\ \forall\, MP : X \mathrel{\widetilde{\leftrightarrow}} Y \bullet \\ \quad \widetilde{\operatorname{dom}}\, MP = \operatorname{dom}(\operatorname{relof} MP) \land \\ \quad \widetilde{\operatorname{ran}}\, MP = \operatorname{ran}(\operatorname{relof} MP)\end{array}$$

Aside: An alternative approach to the definition of the domain (range) of a multi-relation is the bag of values where the frequency of a value x in the bag is the sum of the frequencies of all pairs in the multi-relation with first (second) component x. We have chosen not to follow this approach because,

- the set versions are simpler and more useful in practice,
- the bag versions are not always well defined, and
- the bag version of domain is already given by $MR\widetilde{\sim}(\!|\ \widetilde{Y}\ |\!)$ and the bag version of range by $MR(\!|\ \widetilde{X}\ |\!)$.

1.13 Domain/range restriction/subtraction

Binary relation domain and range restriction constrain the domain and range, respectively, of a relation to given sets.

$$\begin{array}{l}\forall\, R : X \leftrightarrow Y;\ S : \mathbb{P}\, X;\ T : \mathbb{P}\, Y \bullet \\ \quad S \lhd R = \{xy : R \mid \mathit{first}\ xy \in S\} \land \\ \quad R \rhd T = \{xy : R \mid \mathit{second}\ xy \in T\}\end{array}$$

We can define multi-relation equivalents of these. We also define equivalents of domain and range subtraction.

$$
\begin{array}{|l}
\hline
[X, Y] \\
\_ \mathrel{\widetilde{\lhd}} \_, \_ \mathrel{\widetilde{\not\lhd}} \_ : \mathbb{P}\, X \times (X \mathrel{\widetilde{\leftrightarrow}} Y) \to (X \mathrel{\widetilde{\leftrightarrow}} Y) \\
\_ \mathrel{\widetilde{\rhd}} \_, \_ \mathrel{\widetilde{\not\rhd}} \_ : (X \mathrel{\widetilde{\leftrightarrow}} Y) \times \mathbb{P}\, Y \to (X \mathrel{\widetilde{\leftrightarrow}} Y) \\
\hline
\forall\, S : \mathbb{P}\, X;\ MR : X \mathrel{\widetilde{\leftrightarrow}} Y;\ T : \mathbb{P}\, Y \bullet \\
\quad S \mathrel{\widetilde{\lhd}} MR = [\![xy : MR \mid \mathit{first}\ xy \in S]\!] \land \\
\quad S \mathrel{\widetilde{\not\lhd}} MR = [\![xy : MR \mid \mathit{first}\ xy \notin S]\!] \land \\
\quad MR \mathrel{\widetilde{\rhd}} T = [\![xy : MR \mid \mathit{second}\ xy \in T]\!] \land \\
\quad MR \mathrel{\widetilde{\not\rhd}} T = [\![xy : MR \mid \mathit{second}\ xy \notin T]\!] \\
\hline
\end{array}
$$

Laws The following laws are the equivalents of similar laws for binary relations.

$$
\begin{array}{l}
\forall\, MR : X \mathrel{\widetilde{\leftrightarrow}} Y;\ S : \mathbb{P}\, X;\ T : \mathbb{P}\, Y \bullet \\
\quad S \mathrel{\widetilde{\lhd}} MR = (\widetilde{\mathrm{id}}\, S) \mathrel{\widetilde{\circ}} MR \land \\
\quad MR \mathrel{\widetilde{\rhd}} T = MR \mathrel{\widetilde{\circ}} (\widetilde{\mathrm{id}}\, T) \land \\
\quad \widetilde{\mathrm{dom}}(S \mathrel{\widetilde{\lhd}} MR) = S \cap (\widetilde{\mathrm{dom}}\, MR) \land \\
\quad \widetilde{\mathrm{ran}}(MR \mathrel{\widetilde{\rhd}} T) = (\widetilde{\mathrm{ran}}\, MR) \cap T \land \\
\quad \mathrm{relof}(S \mathrel{\widetilde{\lhd}} MR) = S \lhd (\mathrm{relof}\, MR) \land \\
\quad \mathrm{relof}(MR \mathrel{\widetilde{\rhd}} T) = (\mathrm{relof}\, MR) \rhd T \land \\
\quad S \mathrel{\widetilde{\lhd}} MR \uplus S \mathrel{\widetilde{\not\lhd}} MR = MR \land \\
\quad MR \mathrel{\widetilde{\rhd}} T \uplus MR \mathrel{\widetilde{\not\rhd}} T = MR
\end{array}
$$

Continuing our bill-of-materials example, the atomic parts are those that have no sub-parts at all.

$$
\begin{array}{|l}
\hline
\mathit{atomic} : (\mathit{Part} \mathrel{\widetilde{\leftrightarrow}} \mathit{Part}) \to \mathbb{P}\, \mathit{Part} \\
\hline
\forall\, \mathit{BOM} : \mathit{Part} \mathrel{\widetilde{\leftrightarrow}} \mathit{Part} \bullet \\
\quad \mathit{atomic}\ \mathit{BOM} = \mathit{Part} \setminus (\widetilde{\mathrm{dom}}\, \mathit{BOM}) \\
\hline
\end{array}
$$

We can define a function that explodes the sub-part multi-relation but only gives the atomic components. Compare these definitions with those in Section 1.1.

$$
\begin{array}{|l}
\hline
\mathit{explode} : (\mathit{Part} \mathrel{\widetilde{\leftrightarrow}_f} \mathit{Part}) \nrightarrow (\mathit{Part} \mathrel{\widetilde{\leftrightarrow}_f} \mathit{Part}) \\
\hline
\mathrm{dom}\, \mathit{explode} = \mathrm{nilpotent} \\
\forall\, \mathit{BOM} : \mathrm{dom}\, \mathit{explode} \bullet \\
\quad \mathit{explode}\ \mathit{BOM} = (\mathit{BOM}^{\widetilde{+}}) \mathrel{\widetilde{\rhd}} (\mathit{atomic}\ \mathit{BOM}) \\
\hline
\end{array}
$$

Given an *order* of products (a bag of parts) we can determine the required atomic components by using multi-relation image.

$$(\mathit{explode}\ \mathit{BOM})(\!|\ \widetilde{\mathit{order}}\ |\!)$$

1.14 Override

Relation override satisfies the following condition.

$$Q \oplus R = (\mathrm{dom}\, R \mathrel{\not\lhd} Q) \cup R$$

We can generalise override to multi-relations by using the corresponding multi-relation operators.

$$[X, Y]$$

$$\_ \widetilde{\oplus} \_ : (X \widetilde{\leftrightarrow} Y) \times (X \widetilde{\leftrightarrow} Y) \to (X \widetilde{\leftrightarrow} Y)$$

$$\forall MQ, MR : X \widetilde{\leftrightarrow} Y \bullet$$
$$MQ \widetilde{\oplus} MR = (\widetilde{\operatorname{dom} MR} \widetilde{\lhd} MQ) \uplus MR$$

Laws The following laws for multi-relation override are equivalents of similar laws for relation override.

$$\forall MP, MQ, MR : X \widetilde{\leftrightarrow} Y \bullet$$
$$MR \widetilde{\oplus} \{\!|\ \ |\!\} = MR = \{\!|\ \ |\!\} \widetilde{\oplus} MR \land$$
$$(MP \widetilde{\oplus} MQ) \widetilde{\oplus} MR = MP \widetilde{\oplus} (MQ \widetilde{\oplus} MR) \land$$
$$(\widetilde{\operatorname{dom} MQ} \cap \widetilde{\operatorname{dom} MR} = \{\ \} \Rightarrow$$
$$MQ \widetilde{\oplus} MR = MQ \uplus MR = MR \widetilde{\oplus} MQ) \land$$
$$(\forall x : X;\ y : Y \bullet$$
$$(x \in \widetilde{\operatorname{dom} MR} \Rightarrow (MQ \widetilde{\oplus} MR) \sharp (x, y) = MR \sharp (x, y)) \land$$
$$(x \notin \widetilde{\operatorname{dom} MR} \Rightarrow (MQ \widetilde{\oplus} MR) \sharp (x, y) = MQ \sharp (x, y)))$$

2 Case study: A bill-of-materials system

The following specification makes use of Z schemata (see [3, 8]) to present the state and operations of a bill-of-materials system. The specification makes extensive use of bags and multi-relations. The description of the bill-of-materials system is split into three major sections:

- product descriptions,
- customer orders and the store, and
- factory production.

2.1 Product descriptions

We introduce the set *Part* which denotes all possible parts, including products, assemblies and atomic components. The specification makes use of a multi-relation for the bill-of-materials (BOM) product descriptions. In addition, we have the set of all known parts and the set of products.

Products

$$allparts,$$
$$products : \mathbb{F}\ Part$$
$$BOM : Part \widetilde{\leftrightarrow}_+ Part$$

$$BOM \in \text{nilpotent} \land$$
$$\widetilde{\operatorname{dom} BOM} \subseteq allparts \land$$
$$\widetilde{\operatorname{ran} BOM} \subseteq allparts \land$$
$$products \subseteq allparts$$

Each product description must not make use of itself as a sub-part either directly or indirectly. Hence the product descriptions are required to form a nilpotent multi-relation, i.e., no cycles.

An alternative approach could have used the following, almost equivalent, structure.

$$BOM : Part \pfun \mathrm{bag}\, Part$$

The main advantage of using a multi-relation is the useful set of operations analogous to those for relations, that have been defined for multi-relations.

We introduce the following change and no change schemata.

$$\Delta Products \mathrel{\widehat=} Products \land Products'$$
$$\Xi Products \mathrel{\widehat=} [\Delta Products \mid \theta Products' = \theta Products]$$

AddPart adds new parts/products to the descriptions.

*AddPart*0

$\Delta Products$
$p? : Part$
$madefrom? : \mathrm{bag}_f\, Part$
$product? : Boolean$

$p? \notin allparts \land$
$\mathrm{setof}\, madefrom? \subseteq allparts \land$
$allparts' = allparts \cup \{p?\} \land$
$(product? = True \Rightarrow products' = products \cup \{p?\}) \land$
$(product? = False \Rightarrow products' = products) \land$
$BOM' = BOM \uplus [\![p : madefrom? \bullet (p?, p)]\!]$

Note that as $p?$ is a new part and all the parts in *madefrom*? must already exist, cyclic dependencies cannot be introduced between parts.

ChangePart changes the description of an existing part.

*ChangePart*0

$\Delta Products$
$p? : Part$
$madefrom? : \mathrm{bag}_f\, Part$
$product? : Boolean$

$p? \in allparts \land$
$\mathrm{setof}\, madefrom? \subseteq allparts \land$
$\neg\, p? \,\mathrm{in}\, \widetilde{BOM^*}(\!|\, \widetilde{madefrom?} \,|\!) \land$
$allparts' = allparts \land$
$(product? = True \Rightarrow products' = products \cup \{p?\}) \land$
$(product? = False \Rightarrow products' = products \setminus \{p?\}) \land$
$BOM' = BOM \mathbin{\widetilde{\oplus}} [\![p : madefrom? \bullet (p?, p)]\!]$

As $p?$ is an existing part we have to avoid introducing cycles by insisting that $p?$ is not used to make anything in the set of parts that $p?$ is made from.

Errors Problems occur if an already existing part is added, or if its sub-parts do not exist, or if a part to be changed does not exist, or if it is used as a component, perhaps indirectly, of its new description.

$$
\begin{aligned}
Report ::= & \ OK \\
| & \ Part\_already\_exists \\
| & \ Parts\_not\_known\langle\langle \mathbb{P}\, Part \rangle\rangle \\
| & \ Part\_not\_known \\
| & \ Part\_used\_to\_make\_itself
\end{aligned}
$$

PartExists

$$
\begin{aligned}
& \Xi Products \\
& p? : Part \\
& rep! : Report \\
\hline
& p? \in allparts \land \\
& rep! = Part\_already\_exists
\end{aligned}
$$

PartMissing

$$
\begin{aligned}
& \Xi Products \\
& madefrom? : \mathrm{bag}_f\, Part \\
& rep! : Report \\
\hline
& \neg\, (\mathrm{setof}\, madefrom? \subseteq allparts) \land \\
& rep! = Parts\_not\_known(\mathrm{setof}\, madefrom? \setminus allparts)
\end{aligned}
$$

PartNonExistent

$$
\begin{aligned}
& \Xi Products \\
& p? : Part \\
& rep! : Report \\
\hline
& p? \notin allparts \land \\
& rep! = Part\_not\_known
\end{aligned}
$$

CyclicDefn

$$
\begin{aligned}
& \Xi Products \\
& p? : Part \\
& madefrom? : \mathrm{bag}_f\, Part \\
& rep! : Report \\
\hline
& p? \,\mathrm{in}\, \widetilde{BOM^*}(\!|\, \widetilde{madefrom?} \,|\!) \land \\
& rep! = Part\_used\_to\_make\_itself
\end{aligned}
$$

The operations with error reports follow.

$$
\begin{aligned}
& Success \mathrel{\widehat=} [rep! : Report \mid rep! = OK] \\
& AddPart \mathrel{\widehat=} (AddPart0 \land Success) \lor PartExists \lor PartMissing \\
& ChangePart \mathrel{\widehat=} (ChangePart0 \land Success) \lor PartNonExistent \lor CyclicDefn
\end{aligned}
$$

2.2 Customer orders and the store

Customers may order a bag of parts. Here we do not keep track of the details of customers, only orders; such details could be easily added. Orders are given an order number from the set *OrderNo*. As well as keeping track of the current unfilled orders, we record the original orders for posterity. The parts currently in stock are recorded in the bag *store*.

Orders

$$
\begin{array}{l}
Products \\
unfilled, \\
orders : OrderNo \pfun \mathrm{bag}\, Part \\
store : \mathrm{bag}\, Part \\
\hline
(\forall B : \mathrm{ran}\, orders \bullet \mathrm{setof}(B) \subseteq allparts) \land \\
(\mathrm{dom}\, unfilled) \ndres orders = unfilled \land \\
\mathrm{setof}\, store \subseteq allparts
\end{array}
$$

The operations to record orders do not change the product descriptions.

$$\Delta Orders \mathrel{\widehat=} Orders \land Orders' \land \Xi Products$$
$$\Xi Orders \mathrel{\widehat=} [\Delta Orders \mid \theta Orders' = \theta Orders]$$

When orders arrive they are recorded and a new order number is generated.

*Order*0

$$
\begin{array}{l}
\Delta Orders \\
order? : \mathrm{bag}\, Part \\
orderid! : OrderNo \\
\hline
order? \neq \{\!|\ \ |\!\} \land \\
\mathrm{setof}\, order? \subseteq products \land \\
orderid! \notin \mathrm{dom}\, orders \land \\
unfilled' = unfilled \cup \{orderid! \mapsto order?\} \land \\
orders' = orders \cup \{orderid! \mapsto order?\} \land \\
store' = store
\end{array}
$$

Given the current crop of orders a useful enquiry is to determine which orders are able to be filled from the current stock.

*Fillable*0

$$
\begin{array}{l}
\Xi Orders \\
fillable! : \mathbb{P}\, OrderNo \\
\hline
fillable! = \{n : OrderNo \mid unfilled(n) \sqsubseteq store\}
\end{array}
$$

Note that, although each order returned is able to be filled, this does not imply any pair of these orders can necessarily be filled together.

Orders are filled from the current stock. Partial filling of orders is not allowed in this simple example.

$$
\begin{array}{|l}
\hline
\textit{Fill}0 \\
\Delta \textit{Orders} \\
\textit{orderid}? : \textit{OrderNo} \\
\hline
\textit{orderid}? \in \operatorname{dom} \textit{unfilled} \land \\
\textit{unfilled}(\textit{orderid}?) \sqsubseteq \textit{store} \land \\
\textit{unfilled}' = \{\textit{orderid}?\} \lhd \textit{unfilled} \land \\
\textit{orders}' = \textit{orders} \land \\
\textit{store}' = \textit{store} \uplus \textit{unfilled}(\textit{orderid}?) \\
\hline
\end{array}
$$

The operator '$\uplus$' is bag subtraction (see Appendix B).

Orders may be modified if they have not been filled.

$$
\begin{array}{|l}
\hline
\textit{Modify}0 \\
\Delta \textit{Orders} \\
\textit{order}? : \operatorname{bag} \textit{Part} \\
\textit{orderid}? : \textit{OrderNo} \\
\hline
\textit{order}? \neq \{\!|\ \ |\!\} \land \\
\operatorname{setof} \textit{order}? \subseteq \textit{products} \land \\
\textit{orderid}? \in \operatorname{dom} \textit{unfilled} \land \\
\textit{unfilled}' = \textit{unfilled} \oplus \{\textit{orderid}? \mapsto \textit{order}?\} \land \\
\textit{orders}' = \textit{orders} \oplus \{\textit{orderid}? \mapsto \textit{order}?\} \land \\
\textit{store}' = \textit{store} \\
\hline
\end{array}
$$

Orders may be cancelled if they have not been filled. The cancelled order number cannot be reused; this is ensured by retaining it in the domain of *orders*, but mapping it to the empty order.

$$
\begin{array}{|l}
\hline
\textit{Cancel}0 \\
\Delta \textit{Orders} \\
\textit{orderid}? : \textit{OrderNo} \\
\hline
\textit{orderid}? \in \operatorname{dom} \textit{unfilled} \land \\
\textit{unfilled}' = \{\textit{orderid}?\} \lhd \textit{unfilled} \land \\
\textit{orders}' = \textit{orders} \oplus \{\textit{orderid}? \mapsto \{\!|\ \ |\!\}\} \land \\
\textit{store}' = \textit{store} \\
\hline
\end{array}
$$

When parts are delivered to the store they are recorded.

$$
\begin{array}{|l}
\hline
\textit{Stow}0 \\
\Delta \textit{Orders} \\
\textit{parts}? : \operatorname{bag} \textit{Part} \\
\hline
\operatorname{setof} \textit{parts}? \subseteq \textit{allparts} \land \\
\textit{store}' = \textit{store} \uplus \textit{parts}? \land \\
\textit{unfilled}' = \textit{unfilled} \land \\
\textit{orders}' = \textit{orders} \\
\hline
\end{array}
$$

Errors Orders must be non-empty and may only be made for products; orders may only be filled if the request is completely in stock; and only known orders

may be filled, modified or cancelled. We extend the previous definition of error reports with these reports plus two we will need later.

$$
\begin{array}{rcl}
\mathit{Report} & ::= & \mathit{OK} \\
& | & \mathit{Part\_already\_exists} \\
& | & \mathit{Parts\_not\_known}\langle\langle \mathbb{P}\, \mathit{Part} \rangle\rangle \\
& | & \mathit{Part\_not\_known} \\
& | & \mathit{Part\_used\_to\_make\_itself} \\
& | & \mathit{Empty\_order} \\
& | & \mathit{Parts\_not\_products}\langle\langle \mathbb{P}\, \mathit{Part} \rangle\rangle \\
& | & \mathit{Not\_in\_stock}\langle\langle \mathrm{bag}\, \mathit{Part} \rangle\rangle \\
& | & \mathit{Order\_filled\_or\_not\_known} \\
& | & \mathit{Not\_made\_by\_factory}\langle\langle \mathbb{P}\, \mathit{Part} \rangle\rangle \\
& | & \mathit{Not\_in\_production}\langle\langle \mathrm{bag}\, \mathit{Part} \rangle\rangle
\end{array}
$$

Empty

$$
\begin{array}{l}
\Xi \mathit{Orders} \\
\mathit{order}? : \mathrm{bag}\, \mathit{Part} \\
\mathit{rep}! : \mathit{Report} \\
\hline
\mathit{order}? = \{\!|\ \ |\!\} \land \\
\mathit{rep}! = \mathit{Empty\_order}
\end{array}
$$

If requested parts are not products, the set of non-product parts requested is reported.

NotProduct

$$
\begin{array}{l}
\Xi \mathit{Orders} \\
\mathit{order}? : \mathrm{bag}\, \mathit{Part} \\
\mathit{rep}! : \mathit{Report} \\
\hline
\neg\, (\mathrm{setof}\, \mathit{order}? \subseteq \mathit{products}) \land \\
\mathit{rep}! = \mathit{Parts\_not\_products}(\mathrm{setof}\, \mathit{order}? \setminus \mathit{products})
\end{array}
$$

If an order is not able to be filled, the bag of parts that are not in stock is reported. This is determined by subtracting the parts in stock from the parts required for the order. Parts with a positive frequency in the result do not have sufficient stock to fulfill the order, while parts with a zero or negative frequency do have sufficient stock. The function 'pbagof' selects just those parts with positive frequencies.

NotFillable

$$
\begin{array}{l}
\Xi \mathit{Orders} \\
\mathit{orderid}? : \mathit{OrderNo} \\
\mathit{rep}! : \mathit{Report} \\
\hline
\mathit{orderid}? \in \mathrm{dom}\, \mathit{unfilled} \land \\
\neg\, (\mathit{unfilled}(\mathit{orderid}?) \sqsubseteq \mathit{store}) \land \\
\mathit{rep}! = \mathit{Not\_in\_stock}(\mathrm{pbagof}(\mathit{unfilled}(\mathit{orderid}?) \uplus \mathit{store}))
\end{array}
$$

InvalidOrderNo

$\Xi Orders$
$orderid? : OrderNo$
$rep! : Report$

$orderid? \notin \operatorname{dom} unfilled \land$
$rep! = Order\_filled\_or\_not\_known$

The operations complete with error alternatives follow.

$Order \mathrel{\widehat=} (Order0 \land Success) \lor NotProduct \lor Empty$
$Fillable \mathrel{\widehat=} Fillable0 \land Success$
$Fill \mathrel{\widehat=} (Fill0 \land Success) \lor InvalidOrderNo \lor NotFillable$
$Modify \mathrel{\widehat=} (Modify0 \land Success) \lor InvalidOrderNo \lor NotProduct \lor Empty$
$Cancel \mathrel{\widehat=} (Cancel0 \land Success) \lor InvalidOrderNo$
$Stow \mathrel{\widehat=} (Stow0 \land Success) \lor PartMissing[parts?/madefrom?]$

2.3 The factory

For the operations that we define for the factory, we need access to the information about orders and the store. At any time the factory is building a number of parts.

Factory

$Orders$
$inproduction : \operatorname{bag} Part$

$\operatorname{setof} inproduction \subseteq allparts$

Operations on the factory do not update the product descriptions or the orders.

$\Delta Factory$

$Factory$
$Factory'$
$\Xi Products$

$unfilled' = unfilled \land orders' = orders$

$\Xi Factory \mathrel{\widehat=} [\Delta Factory \mid \theta Factory' = \theta Factory]$

The major enquiry operation is to determine the total parts required to fulfill all orders excluding those already in stock or in production. This is split into two components: the atomic parts needed to be ordered into the store and the parts that need to be produced by the factory.

To define this operation we introduce the function *maximal* which, given a set of bags, SB, returns a set containing those bags within SB which are maximal. A bag B is maximal within SB if there does not exist another bag within SB which strictly contains B.

$$[X]$$

$$maximal : \mathbb{P}(\text{bag } X) \rightarrow \mathbb{P}(\text{bag } X)$$

$$\forall\, SB : \mathbb{P}(\text{bag } X) \bullet$$
$$maximal(SB) = \{B : SB \mid \neg\, (\exists\, B' : SB \bullet B \sqsubset B')\}$$

The total parts on order by customers consists of the items in unfilled orders:

$$onorder = \biguplus(items\ unfilled).$$

The function *items* when applied to a function f gives the bag of items that are contained in the range of f. The frequency of each item in the bag is the number of members of the domain of f that map to that item (see Appendix B). In the above use *items unfilled* gives a bag of bags of parts, which is then collapsed into a single bag by '$\biguplus$'.

To calculate the requirements we obviously need to subtract those parts which are already in hand.

$$inhand = store \uplus inproduction$$

But the true picture is more complex than this. If a part is on order but not in hand, then not only does this part need to be produced, but the sub-parts from which it is made are also needed. The complete requirements can be ascertained by fully exploding the parts that are on order:

$$\widetilde{BOM^*}(\!|\ \widetilde{onorder}\ |\!).$$

This gives the parts on order as well as the sub-parts required to produce them, as well as the sub-sub-parts required to produce the sub-parts, etc. In the calculation of the requirements we need to take into account the fact that some of the parts or sub-parts or sub-sub-parts, etc., may be in hand. Such parts fulfill the exploded requirement for not only the parts themselves but also their direct and indirect sub-parts. The bag of parts that are in hand, that can be used in some way to fulfill the exploded requirements of the orders, is the subbag of *inhand* whose explosion is maximal in the exploded order requirements.

$$\widetilde{BOM^*}(\!|\ \widetilde{frominhand}\ |\!) \in maximal\{B : \text{bag } Part \mid B \sqsubseteq inhand \wedge$$
$$\widetilde{BOM^*}(\!|\ \widetilde{B}\ |\!) \sqsubseteq \widetilde{BOM^*}(\!|\ \widetilde{onorder}\ |\!) \bullet \widetilde{BOM^*}(\!|\ \widetilde{B}\ |\!)\}$$

Note that *frominhand* is unique.

*Requirements*0

$\Xi Factory$
$reqdall!,$
$reqdatomic!,$
$reqdproduce! : \text{bag}\, Part$

$\exists\, onorder, inhand, frominhand : \text{bag}\, Part \bullet$
$\quad onorder = \biguplus(items\ unfilled) \land$
$\quad inhand = store \uplus inproduction \land$
$\quad \widetilde{BOM^*}(\!|\ \widetilde{frominhand}\ |\!) \in maximal\{B : \text{bag}\, Part \mid B \sqsubseteq inhand \land$
$\qquad \widetilde{BOM^*}(\!|\ \widetilde{B}\ |\!) \sqsubseteq \widetilde{BOM^*}(\!|\ \widetilde{onorder}\ |\!) \bullet \widetilde{BOM^*}(\!|\ \widetilde{B}\ |\!)\} \land$
$\quad reqdall! = (\widetilde{BOM^*}(\!|\ \widetilde{onorder}\ |\!)) \uplus (\widetilde{BOM^*}(\!|\ \widetilde{frominhand}\ |\!)) \land$
$\quad reqdatomic! = [\![p : reqdall! \mid p \notin \widetilde{\text{dom}}\, BOM]\!] \land$
$\quad reqdproduce! = [\![p : reqdall! \mid p \in \widetilde{\text{dom}}\, BOM]\!]$

The factory manager can decide to start making some parts. If not all the parts can be made from the parts in stock then a maximal sub-bag of parts (*beingmade*!) is chosen and the residual that cannot currently be made is reported in *noparts*!. The parts being made are recorded as being in production and the parts required to make them are removed from stock.

*Make*0

$\Delta Factory$
$tomake?,$
$noparts!,$
$beingmade! : \text{bag}\, Part$

$\text{setof}\, tomake? \subseteq \widetilde{\text{dom}}\, BOM \land$
$(\exists\, makeable : \mathbb{P}(\text{bag}\, Part) \bullet$
$\quad makeable = \{B : \text{bag}\, Part \mid B \sqsubseteq tomake? \land$
$\qquad BOM(\!|\ \widetilde{B}\ |\!) \sqsubseteq store\} \land$
$\quad beingmade! \in maximal(makeable)) \land$
$noparts! = tomake? \uplus beingmade! \land$
$inproduction' = inproduction \uplus beingmade! \land$
$store' = store \uplus (BOM(\!|\ \widetilde{beingmade!}\ |\!))$

The set *makeable* gives all possible bags of parts that are directly makeable from the parts in stock as well as being sub-bags of the parts requested to be made. The parts *beingmade*! are a maximal bag contained in this set. Note that this does not uniquely determine *beingmade*! as there can be more than one makeable sub-bag of *tomake*? that cannot be extended to a larger makeable bag.

When parts have been produced by the factory they are placed back in the store for later distribution.

*Made*0

$\Delta Factory$
$made? : \text{bag}\ Part$

$made? \sqsubseteq inproduction \land$
$inproduction' = inproduction \uplus made? \land$
$store' = store \uplus made?$

Errors Errors can occur if a part to be made is not one produced by the factory, or the factory reports that it has made a part that was not one in production.

NotProduced

$\Xi Factory$
$tomake? : \text{bag}\ Part$
$rep! : Report$

$\neg\ (\text{setof}\ tomake? \subseteq \widetilde{\text{dom}}\ BOM) \land$
$rep! = Not\_made\_by\_factory(\text{setof}\ tomake? \setminus \widetilde{\text{dom}}\ BOM)$

NotInProduction

$\Xi Factory$
$made? : \text{bag}\ Part$
$rep! : Report$

$\neg\ (made? \sqsubseteq inproduction) \land$
$rep! = Not\_in\_production(\text{pbagof}(made? \uplus inproduction))$

The total operations follow.

$Requirements \mathrel{\widehat=} Requirements0 \land Success$
$Make \mathrel{\widehat=} (Make0 \land Success) \lor NotProduced$
$Made \mathrel{\widehat=} (Made0 \land Success) \lor NotInProduction$

3 Conclusion

In Z, the theory of binary relations is built on a model consisting of sets of pairs. The theory of multi-relations introduced above is built on a model consisting of multi-sets (or bags) of pairs. As multi-relations are (special forms of) bags all the bag operators can be used directly on multi-relations, in the same way that the set operators can be used on ordinary relations. In addition, we have introduced a comprehensive set of multi-relation operators that are multi-relation equivalents of the ordinary relation operators. All of the ordinary relation operators can be extended to provide useful operations on multi-relations.

Multi-relations are essentially two-dimensional integer matrices, and there are equivalent matrix operators for many of the multi-relation operators defined above. However, one of the main aims of this paper is to show the relationship between ordinary relations and multi-relations. To emphasise this we have used

the term multi-relation and chosen to introduce operators on multi-relations as equivalents of operators on ordinary relations.

The bill-of-materials specification shows the usefulness of both the theories of multi-sets (bags) and multi-relations. The details of the product descriptions and the calculations of requirements, etc., based on these are complex operations that can be modelled succinctly using multi-relations. The example bill-of-materials system given above is somewhat simplified. It is not intended to be the be-all and end-all of bill-of-materials systems. It is intended, however, to show that multi-relations provide a useful abstraction for specifying such systems.

The development of the bill-of-materials specification relied on the existence of the theory of multi-relations. Even if such a theory was not available it is worthwhile for the specifier to develop such a theory as the first stage in producing a specification of a bill-of-materials system. The advantage of such an approach is that the theory is an *abstraction* that concentrates on developing the underlying *concepts* of such systems independent of the details of the particular system. These concepts are used many times over in the specification of the system. Furthermore, the laws developed for the abstract concepts developed in the theory of multi-relations are considerably simpler than the properties of the particular bill-of-materials system. In reasoning about such a system these laws will be used over and over again.

Another significant advantage to developing a separate mathematical theory is that it can be reused in the specification of other systems. The bill-of-materials system given in Section 2 is one possible approach to such a system, but another bill-of-materials system may take a quite different approach. In this case it is unlikely that the specification given above can be reused, but it is likely that the theory of multi-relations can be reused.

Not only is the theory of multi-relations useful for describing a bill-of-materials system, it can be used to describe and reason about other quite different systems that require objects with similar abstract properties. For example, a reference count memory garbage collector can be modelled in terms of multi-relations. One object in the memory may reference another object a number of times. A multi-relation can record the frequency of such references. Multi-relations may also be used to describe aspects of computer networks, such as the multiplicity of links between nodes or the bandwidth between nodes.

The theory of multi-relations for Z presented above is based on the mathematics of matrices and graph theory. Multi-relations correspond in graph theory to graphs with integer weights. Graph theory is a well established field [2] that has been successfully applied to many computing applications especially in the area of optimisation problems. The theory of multi-relations can be used for specifying and reasoning about such applications in Z.

Acknowledgements

This work was begun while I was visiting the Programming Research Group of Oxford University and I would like to acknowledge the hospitality of the Programming Research Group and Wolfson College, and the financial assistance of the Special Studies Program of the University of Queensland.

The inspiration for this work came from a discussion with Cliff Jones about

the earlier work on bags [4], in which Cliff pointed me at the bill-of-materials example. The bill-of-materials example was also used as a project case study in the *Software Specification and Development* graduate subject at Queensland University in 1990, and I would like to thank the students of that subject for their ideas and participation. I would also like to thank Cliff Jones, Brendan Mahony, Luke Wildman, the referees of this paper and members of IFIP Working Group 2.3 on Programming Methodology for useful comments on earlier drafts of this paper.

References

[1] O-J. Dahl. Can program proving be made practical? In M. Amirchahy and D. Neel, editors, *EEC-Crest Course on Programming Foundations*, pages 57–114, B.P.105 78150 Le Chesnay, France, 1977. IRIA.

[2] Narsingh Deo. *Graph Theory with Applications to Engineering and Computer Science*. Prentice-Hall, 1974.

[3] I. J. Hayes, editor. *Specification Case Studies*. Prentice Hall, 1987. Second edition 1993.

[4] I. J. Hayes. A generalisation of bags in Z. In J. E. Nicholls, editor, *Z User Workshop: Proceedings of the Fourth Annual Z User Meeting, Oxford, December 1989*, Workshops in Computing, pages 113–127. Springer, 1990.

[5] C. B. Jones. *Software Development: A Rigorous Approach*. Prentice Hall International, 1980.

[6] C. B. Jones. *Systematic Software Development Using VDM*. Prentice Hall International, second edition, 1990.

[7] C.B. Jones. Constructing a theory of a data structure as an aid to program development. *Acta Informatica*, 11:119–137, 1979.

[8] J. M. Spivey. *The Z Notation: A Reference Manual*. Prentice Hall International, second edition, 1992.

A Binary relations

A binary relation is modelled by a set of ordered pairs. Hence operators defined for sets can be used on relations. Let X, Y and Z be sets; $x, x_1, \ldots, x_n : X$; $y, y_1, y_2, \ldots, y_n : Y$; S be a subset of X; T be a subset of Y; and R a relation between X and Y.

$X \leftrightarrow Y$ $\quad == \mathbb{P}(X \times Y)$
The set of relations between X and Y. The set X is referred to as the *source* of the relation R and the set Y as its *destination*.

$x \mathrel{\underline{R}} y$ $\quad == (x, y) \in R$
x is related by R to y. The name of a relation may either be an identifier or an infix operator symbol. A relation with an identifier

name may be used as an infix operator by underlining it. For an infix relation operator, the whole relation may be referred to by placing underscores either side of the symbol and enclosing that in parentheses. For example, the whole relation corresponding to the infix operator '$<$' is referred to by '$(\_ < \_)$', so $(x < y) \Leftrightarrow (x, y) \in (\_ < \_)$.

$x \mapsto y$ $== (x, y)$

$\{x_1 \mapsto y_1, x_2 \mapsto y_2, \ldots, x_n \mapsto y_n\}$
$== \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$
The relation relating x_1 to y_1, x_2 to y_2, ..., and x_n to y_n.

$\operatorname{dom} R$ $== \{x : X \mid (\exists\, y : Y \bullet x \mathrel{\underline{R}} y)\}$
The domain of a relation: the set of x components that are related to some y.

$\operatorname{ran} R$ $== \{y : Y \mid (\exists\, x : X \bullet x \mathrel{\underline{R}} y)\}$
The range of a relation: the set of y components that some x is related to.

$R_1 \fatsemi R_2$ $== \{x : X;\ z : Z \mid (\exists\, y : Y \bullet x \mathrel{\underline{R_1}} y \land y \mathrel{\underline{R_2}} z)\}$
Forward relational composition; $R_1 : X \leftrightarrow Y$; $R_2 : Y \leftrightarrow Z$. The composition relates x to z if there is some y such that x is related to y by R_1 and y is related to z by R_2.

$R_1 \circ R_2$ $== R_2 \fatsemi R_1$
Relational composition. This form is primarily used when R_1 and R_2 are functions.

$R^\sim$ $== \{y : Y;\ x : X \mid x \mathrel{\underline{R}} y\}$
Transpose of a relation R. $R^\sim$ relates y to x if and only if R relates x to y.

$\operatorname{id} S$ $== \{x : S \bullet x \mapsto x\}$
Identity function on the set S.

R^k The relation R composed with itself k times. This operator (sometimes called *iteration*) is only defined for homogeneous relations: relations that have the same source and destination sets. Given a homogeneous relation $R : X \leftrightarrow X$ and $k : \mathbb{N}$
$R^0 = \operatorname{id} X$ and $R^{k+1} = R^k \fatsemi R$.

R^+ $== \bigcup \{n : \mathbb{N}_1 \bullet R^n\}$
$= \bigcap \{Q : X \leftrightarrow X \mid R \subseteq Q \land Q \fatsemi Q \subseteq Q\}$
Transitive closure of relation R. A pair (x_1, x_n) is in the relation R^+ if and only if there exists a finite sequence of values $x_1, x_2, ..., x_n$, where $n \geq 2$, such that $(x_1, x_2) \in R$, $(x_2, x_3) \in R$, ... and $(x_{n-1}, x_n) \in R$.

R^* $== \bigcup \{n : \mathbb{N} \bullet R^n\}$
$= R^+ \cup \operatorname{id} X$
$= \bigcap \{Q : X \leftrightarrow X \mid \operatorname{id} X \subseteq Q \land R \subseteq Q \land Q \fatsemi Q \subseteq Q\}$
Reflexive transitive closure.

$R(\!|\ S\ |\!)$ $== \{y : Y \mid (\exists\, x : S \bullet x\ \underline{R}\ y)\}$
Image of the set S through the relation R.

$S \lhd R$ $== \{x : X;\ y : Y \mid x \in S \land x\ \underline{R}\ y\}$
Domain restriction: the relation R with its domain restricted to the set S.

$S \lhd\!\!\!\!- R$ $== (X \setminus S) \lhd R$
Domain exclusion: the relation R with the members of S excluded from its domain.

$R \rhd T$ $== \{x : X;\ y : Y \mid x\ \underline{R}\ y \land y \in T\}$
Range restriction to T.

$R \rhd\!\!\!\!- T$ $== R \rhd (Y \setminus T)$
Range exclusion: the relation R with the members of T excluded from its range.

$R_1 \oplus R_2$ $== ((\mathrm{dom}\ R_2) \lhd\!\!\!\!- R_1) \cup R_2$
Overriding; $R_1, R_2 : X \leftrightarrow Y$.

B Bags

Let $B, B1, B2, \ldots$ be bags with elements from the set X; t, t_1, t_2, $\ldots$, t_n be expressions of type X; x, x_1, x_2, $\ldots$, x_n be variables; and $n, k_1, k_2, \ldots, k_n$ be integers.

$\mathrm{bag}\ X$ $== X \nrightarrow \mathbb{N}_1$
The set of bags whose elements are drawn from the set X. Only positive frequencies are recorded.

$\{\!|\ \ |\!\}$ $== \{\ \}$
The empty bag.

$[\![t \mapsto n]\!]$ The bag which contains (only) t, n times.
$= \{t \mapsto n\}$, if $n \neq 0$
$= \{\!|\ \ |\!\}$, if $n = 0$.

$[\![t_1 \mapsto k_1, t_2 \mapsto k_2, \ldots, t_n \mapsto k_n]\!]$
$== [\![t_1 \mapsto k_1]\!] \uplus [\![t_2 \mapsto k_2]\!] \uplus \cdots \uplus [\![t_n \mapsto k_n]\!]$
Note that with this definition we do not require the t_j to be distinct; for example,
$[\![t \mapsto k_1, t \mapsto k_2]\!] = [\![t \mapsto k_1]\!] \uplus [\![t \mapsto k_2]\!]$
$= [\![t \mapsto k_1 + k_2]\!]$.

$[\![t_1, t_2, \ldots, t_n]\!]$
$== [\![t_1 \mapsto 1, t_2 \mapsto 1, \ldots, t_n \mapsto 1]\!]$
The bag containing the elements $t_1, t_2, \ldots, t_n$ with the frequency in which they occur in that list.

$B \sharp t$ The frequency of occurrence of the value of t in the bag B:
$(t \in \mathrm{dom}\ B \Rightarrow B \sharp t = B(t))$, and
$(t \notin \mathrm{dom}\ B \Rightarrow B \sharp t = 0)$.

$count(B)(t)$ The frequency of occurrence of the value of t in the bag B:
$(t \in \operatorname{dom} B \Rightarrow count(B)(t) = B(t))$, and
$(t \notin \operatorname{dom} B \Rightarrow count(B)(t) = 0)$.

$t \operatorname{in} B$ $== B \sharp t \neq 0$
Test whether the element t occurs in the bag B with non-zero frequency.

$B1 \uplus B2$ The sum of two bags. Each element of the sum of the bags has a frequency which is the sum of its frequencies in the two bags:
$(B1 \uplus B2) \sharp x = (B1 \sharp x) + (B2 \sharp x)$.

$B1 \circledast B2$ The (pairwise) product of two bags. Each element of the product has a frequency which is the product of its frequencies in the two bags:
$(B1 \circledast B2) \sharp x = (B1 \sharp x) * (B2 \sharp x)$.

$n \otimes B$ An integer constant times a bag. Each element of the product has a frequency which is the product of its frequency in B and the constant:
$(n \otimes B) \sharp x = n * (B \sharp x)$.

$B1 \sqcap B2$ The pairwise minimum of two bags.
$(B1 \sqcap B2) \sharp x = min\{B1 \sharp x, B2 \sharp x\}$.

$B1 \sqcup B2$ The pairwise maximum of two bags.
$(B1 \sqcup B2) \sharp x = max\{B1 \sharp x, B2 \sharp x\}$.

$B1 \sqsubseteq B2$ $== (\forall x : X \bullet (B1 \sharp x) \leq (B2 \sharp x))$.
$B1$ is a sub-bag of $B2$. One bag is contained in another if the frequency of every element in the first bag does not exceed its corresponding frequency in the second bag.

$B1 \sqsubset B2$ $== B1 \sqsubseteq B2 \land B1 \neq B2$
$B1$ is a proper sub-bag of $B2$.

$\operatorname{setof} B$ $== \{x : X \mid B \sharp x \neq 0\}$
The set of items in the bag B that occur with non-zero frequency.

$\operatorname{bagof} S$ The bag formed from the set S by including all the elements of S (and no others) with a frequency of one.
$\operatorname{dom}(\operatorname{bagof} S) = S \land \operatorname{ran}(\operatorname{bagof} S) = \{1\}$.

$\operatorname{bag}_f X$ $== \{B : \operatorname{bag} X \mid (\operatorname{setof} B) \in \mathbb{F}\, X\}$
The set of all finite bags: those bags with only a finite number of elements with non-zero frequency.

$\operatorname{size} B$ The size of a finite bag is the total number of items in the bag taking into account the frequency of occurrence of each item.

$$\operatorname{size}\{\!|\ \ |\!\} = 0$$
$$\operatorname{size}[\![t]\!] = 1$$
$$\operatorname{size}(B1 \uplus B2) = (\operatorname{size} B1) + (\operatorname{size} B2)$$
$$\operatorname{size}(n \otimes B) = n * \operatorname{size} B$$

$\sum B$ The sum of all the items in the finite bag of numbers B taking into account their frequency in B.

$$\begin{array}{l}\sum \{\!|\ \ |\!\} = 0 \\ \sum [\![t]\!] = t \\ \sum (B1 \uplus B2) = (\sum B1) + (\sum B2) \\ \sum (n \otimes B) = n * \sum B\end{array}$$

$items(R)$ The bag of items which occur in the range of the relation R. The frequency of each item is the number of domain elements that are paired with the item in R. The relation must be 'finitary', that is, for each element in the range of R there are only finitely many domain elements related to it by R. Given $R : W \leftrightarrow X$

$$\begin{array}{l} R \in \mathrm{dom}\, items \Leftrightarrow \\ \quad (\forall x : \mathrm{ran}\, R \bullet \{w : \mathrm{dom}\, R \mid (w, x) \in R\} \in \mathbb{F}\ W) \\ R \in \mathrm{dom}\, items \Rightarrow \\ \quad (items\ R) \sharp x = \#\{w : \mathrm{dom}\, R \mid (w, x) \in R\} \end{array}$$

As both functions and sequences can be considered as special cases of relations, *items* can be used on functions and sequences.

$[\![x : B \mid P \bullet t]\!]$
The bag of all the values of the expression t, for x ranging over all the items in the bag B such that the predicate P holds. If a value of x occurs multiple times in the bag B, then we add the corresponding value of t that many times to the resultant bag;

$$\begin{array}{l} [\![x : B \mid P \bullet t]\!] \sharp y = \\ \quad \sum items(\lambda x : \mathrm{setof}\, B \mid P \wedge y = t \bullet B \sharp x) \end{array}$$

A bag comprehension is only well-defined if each value of t occurs only finitely often. If the expression t is omitted, the default expression is x.

$[\![x_1 : B_1;\ x_2 : B_2;\ \ldots;\ x_n : B_n \mid P \bullet t]\!]$
Multiple variables may be declared in a bag comprehension; each declared variable ranges over the values in the associated bag with the frequency of occurrence of the value in that bag. If the expression t is omitted, the default expression is the tuple of the variables: $(x_1, x_2, \ldots, x_n)$.

$[\![D \bullet t]\!]$ $== [\![D \mid true \bullet t]\!]$
For example, if $B : \mathrm{bag}\, X$ and $C : \mathrm{bag}\, Y$ then

$$[\![x : B;\ y : C \bullet (x, y)]\!],$$

is of type $\mathrm{bag}(X \times Y)$, and the pair (x, y) occurs in this bag $(B \sharp x) * (C \sharp y)$ times; this is the bag generalisation of Cartesian product.

$\biguplus BB$ The distributed bag sum of the bag of bags BB taking into account the frequency of each bag in BB as well as the frequencies of the items in the individual bags. Given $BB : \text{bag}(\text{bag}\, X)$
$(\biguplus BB) \sharp x = \sum [\![B : BB \bullet B \sharp x]\!]$.

Bags can be generalised to allow both positive and negative frequencies. All the operators from the previous section can be generalised to work with bags allowing negative frequencies. The operator definitions given in the previous section have been written so that they are appropriately defined if occurrences of bag are replaced by bag. See [4] for further details and examples.

bag X $== X \pfun (\mathbb{Z} \setminus \{0\})$
The set of generalised bags whose elements are drawn from the set X. Both positive and negative frequencies are allowed in generalised bags.

$-B$ The negation of bag B. Each element of the negation has a frequency which is the negation of its frequencies in B:
$(-B) \sharp x = -(B \sharp x)$.

$B1 \uminus B2$ The difference between two bags. Each element of the difference between the bags has a frequency which is the difference of its frequencies in the two bags:
$(B1 \uminus B2) \sharp x = (B1 \sharp x) - (B2 \sharp x)$.

pbagof B $== [\![p : B \mid B \sharp p > 0]\!]$
The bag with only positive frequency items included.