Abstract Matrix Arithmetic

Alan P. Sexton and Volker Sorge*
School of Computer Science
University of Birmingham
www.cs.bham.ac.uk/~aps|~vxs

Stephen M. Watt
Computer Science Department
University of Western Ontario
www.csd.uwo.ca/~watt

ABSTRACT

We present an approach to basic arithmetic between abstract matrices, i.e., matrices of symbolic dimension with underspecified components. We define a simple basis function that enables the representation of abstract matrices composed of arbitrary regions in a single term that supports matrix addition and multiplication by regular arithmetic on terms. This can, in particular, be exploited to obtain general arithmetic closure properties for classes of structured matrices. We also describe an approach using alternative basis functions that allow more compact expressions and admit additional arithmetic simplifications.

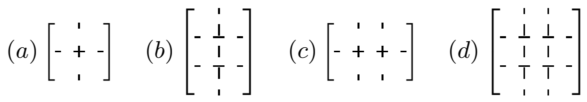
1. INTRODUCTION

It is everyday mathematical practice to represent matrices in an abstract way with symbolic dimensions and containing underspecified parts described by the use of ellipses. While reasoning about matrices in this form is mathematically routine, there is very little automated support for it. In earlier work we have investigated the problem of representing abstract matrices with certain entries given by expressions and others given by interpolating ellipses [4, 5]. Their analysis included determining conditions for boundaries between regions and general expressions for elements within regions of such matrices and has led to a representation that made abstract matrices available as a template for concrete matrices with fully specified dimensions and entries.

In this paper we investigate the problem of performing arithmetic on abstract matrices with arbitrary regions of symbolic size. The main challenge is treating the multiple cases that arise when the relation between the sizes of the regions is not known. Consider for example two 2×2 block matrices

$$A = \begin{bmatrix} A_1 & A_2 \\ A_3 & A_4 \end{bmatrix} \qquad B = \begin{bmatrix} B_1 & B_2 \\ B_3 & B_4 \end{bmatrix}$$

of compatible symbolic dimension, for which the sizes of the blocks A_i, B_i are again given symbolically but are not necessarily compatible. That is, the relationship between the horizontal and vertical extensions of the blocks in matrix A to those in matrix B is undefined. Just computing $A + B$ leads to a number of cases for the different possible overlaps of blocks that can be schematically depicted as:



In a naïve approach we could symbolically represent the sum as a piecewise function consisting of one case for schema (a), two cases each for (b) and (c), and four cases for (d).

As an effective alternative to this naïve approach we introduce a class of basis functions that enables a straightforward representation of abstract matrices as single sums, where each summand represents one region given as the region entry together with a coefficient consisting of products of basis functions representing the region boundaries. This allows us to define addition and multiplication for abstract matrices directly via the corresponding operations on the explicit sums. The different cases of how regions can be combined is given as a product of basis functions in a single explicit formula and no longer needs to be considered discretely in a piecewise function.

This gives rise to a natural way of executing matrix operations and we demonstrate how matrix multiplications can effectively be computed symbolically in a form commonly known from textbooks. We will show how our representation allows us to regain information on the structure of a result matrix in a well defined computational manner. This can in particular be exploited to show general arithmetic closure properties for classes of structured matrices.

There are certain cases in which the naïve, case oriented approach will yield a number of cases factorial in the number of matrix components involved in the computation. In such cases, our approach reduces this to a number of coefficients exponential in the number of components. To address this issue, we present the first steps towards an alternative basis function that, in many cases, reduces this complexity to linear.

Our work is related to previous work by Fateman in Macsyma [1], in which indefinite matrices can be subjected to some basic algebraic manipulations. While his matrices are indefinite in size, their elements are fixed to one particular functional expression and cannot be of arbitrary composition. The work is also similar in spirit to earlier work by Watt [6, 7], which presented algorithms for GCD and

*The author's work was supported in parts by the Transnational Access Program of the Research Institute for Symbolic Computation (RISC), Hagenberg, Austria.

factorization of polynomials with terms of symbolic degree, as well as to work by Knauers and Schneider [3, 2] on indefinite symbolic summation using unspecified sequences of summands.

2. BACKGROUND

In this work we are concerned with doing arithmetic on abstract matrices with underspecified terms and symbolic dimension. We assume we have available appropriately parsed and constructed abstract matrix structures on which to operate. In [5] we have presented a parsing procedure for such abstract matrices. In this section we give a brief introduction to some of the concepts introduced in [5], which are a necessary prerequisite for this paper. Moreover, we simplify some of the concepts to a level sufficient for the problem at hand. We also omit formal definitions here and instead illustrate the major concepts using the following matrix as a running example:

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1n} & & \mathbf{0} & \\ & \ddots & \vdots & & & \\ & & a_{nn} & & & \\ & & & b_{11} & \cdots & b_{1m} \\ & & & & \ddots & \vdots \\ \mathbf{0} & & & & & b_{mm} \end{bmatrix} \tag{1}$$

The general idea of the parsing procedure is to represent an abstract matrix as a set of regions.

Region — A region is a closed polygonal area that can be homogeneously interpolated by a single term. For example, Matrix A contains four regions: one triangular region with terms of the form a_{ij}, one triangular region containing b_{ij}, and two 0 regions, one triangular and one rectangular. Regions are related to each other by their relative positions in the overall matrix and are constrained by the relative lengths of their boundaries. Boundaries can be given explicitly by concrete terms and ellipses or implicitly by the boundaries of neighbouring regions or of the matrix. A region is given in terms of its boundaries, its relative position with respect to other regions in the matrix, and its content in the form of a term schema and functions that interpolate the index variables.

Syntax — The parsing procedure works with respect to a well defined input syntax for abstract matrices. A matrix can contain concrete terms, vertical, horizontal, diagonal and antidiagonal ellipses, as well as fill terms. Ellipses have to be bounded by concrete terms on both sides and diagonal and antidiagonal ellipses are always interpreted to be at a 45° angle. For example in matrix A the upper left triangle consists of one horizontal, one vertical and one diagonal ellipsis and is necessarily equilateral as the diagonal is at a 45° angle. Fill terms are understood to homogeneously fill a region up to the given boundaries. In A we have two regions given by fills terms, both 0.

Ellipsis Lengths — Ellipses in an abstract matrix can be given explicitly by concrete terms and ellipses or implicitly by the boundaries of neighbouring regions. Each ellipsis in a matrix has associated integer constraint variables, representing their relative vertical and horizontal extension, plus explicit constraint variables for height and width. The variables are related via constraints that determine their relative length. For example, in A the ellipses bounding the triangle containing the a_{ij} are all of the same length, similarly those of the b_{ij} triangle. This in turn determines the relative length of implicitly given boundaries of the 0 regions as well as the value for the variables representing height and width.

Generalised Positions — Given the lengths of ellipses in terms of integer constraint variables, we can then determine the relative positions of start and end points of all ellipses with respect to these variables. We call positions containing constraint variables generalised positions. For example, the diagonal ellipses from a_{11} to a_{nn} has its top left corner at position $(1, 1)$, while the position of its bottom right corner is determined by the length of the diagonal ellipsis itself. Although in general, and as explained in [5], we can not associate ellipsis lengths directly with values of the index variables of the terms on the ellipses, for simplicity we will do so here and let the length of the diagonal ellipses be n. Therefore the generalised positon of the bottom right corner is (n, n). This in turn determines the generalised positions of the next diagonal ellipses from b_{11} to b_{mm}, which starts at $(n + 1, n + 1)$ and ends at $(n + m, n + m)$.

Interpolation Functions — With both boundaries and their generalised positions one can now establish the exact content of a region by computing a generalisation of the boundary terms and computing an interpolation function for index functions if necessary. For example, the upper left triangle of A contains terms of the form $a_{\alpha,\beta}$, where α, β are unification variables that can be interpolated with respect to the index variables of the overall matrix. Thus let i, j be the index variables of A, then we can map $\alpha \mapsto i$ and $\alpha \mapsto j$ and all the terms of the region can be expressed as a_{ij}. On the other hand for the lower right triangle we have to take the offset given by generalised positions into account and it can therefore be interpolating by $b_{(i-n)(j-n)}$. Observe also that regions can have a simple term associated with them like the 0 regions in A where no interpolation is necessary.

Abstract Matrix — An abstract matrix is represented as a set of regions, a set of constraints for the variables referred to in these regions, and a (partial) set of bindings for the variables. Two of the variables always present in an abstract matrix are its width and height, although these may or may not have concrete bindings. The width and height variables are used to control the various summations in the algorithms for abstract matrix operations.

3. A SIMPLE BASIS FUNCTION

We wish to represent matrices and vertices in terms of expressions for their component elements. However, these expressions do not record the (symbolic) dimensions of the matrices themselves whereas our internal datastructures do. Therefore we introduce a notation that does precisely that and which limits the extent of the horizontal and vertical indices. These limits will be used explicitly in the summations that occur in vector and matrix addition and multiplication and, hence, guarantee that no term outside the range of the matrices are ever accessed or referred to.

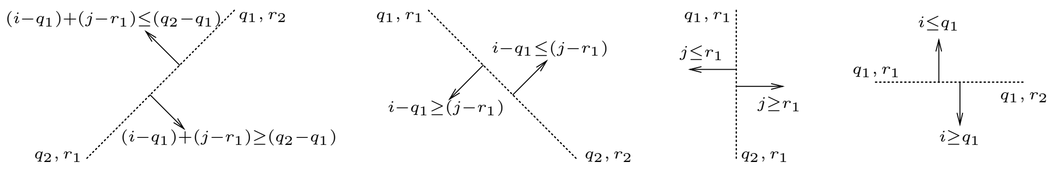


Figure 1: Half plane constraints.

Definition 1

$$[x_{i,j}]_{i,j}^{n,m} \quad ::= \quad \begin{bmatrix} x_{1,1} & \cdots & x_{1,m} \\ \vdots & & \vdots \\ x_{n,1} & \cdots & x_{n,m} \end{bmatrix}$$

where i and j are the names of the index variables used in the expression and range from 1 to n and m respectively. Furthermore, the pair of possible values of the index variables define the dimensions of the matrix, even if those variables do not appear in the element term.

As an example, note that $[1]_{i,j}^{2,3}$ describes the matrix with 2 rows, 3 columns and and in which all the cells contain the value 1.

Definition 2 *Let $x, y \in \mathbb{N}$ then we define*

$$\sigma(x,y) \quad ::= \quad \begin{cases} 1 & \textit{if } x \leqslant y \\ 0 & \textit{otherwise} \end{cases}$$

For convenience, we introduce $\sigma_{x,y}$ as a more compact notation for the function $\sigma(x,y)$

Let $U = [u_1, u_2, \ldots, u_h, u'_1, u'_2, \ldots, u'_{n-h}]$. This vector contains two regions, characterised by their respective generalised terms u_i and u'_i. We represent it as the sum of two terms, where the basis functions control in which part of the vector a particular generalised term is active.

$$U = \left[\sigma_{i,h} u_i + \sigma_{h+1,i} u'_{i-h}\right]_{i,j}^{1,n} \tag{2}$$

Representing general structured matrices pose a slightly more involved problem. Regions are specified by convex polygons, and containment within a region by the conjunction of half planes defined by the equations of the lines of the polygon (c.f. Fig. 1). Containment within a half-plain can be neatly captured by a single σ function, and containment within a region by a product of σs. Hence the elements of the matrix in equation (1) can be written as

$$A = [\sigma_{1,i}\sigma_{j,n}\sigma_{i,j}a_{ij} + \sigma_{n+1,i}\sigma_{j,n+m}\sigma_{i,j}b_{i-n,j-n}]_{i,j}^{m+n,m+n}$$

More explictly, considering the first summand only, $\sigma_{1,i}$ restricts the region to be on or below the top boundary of the a triangular region, $\sigma_{j,n}$ restricts the region to be on or to the left of the right boundary of the triangle, and $\sigma_{i,j}$ restricts it to be on or to the upper right of the diagonal boundary. If any of these half plane constraints are not satisfied, at least one of the σs will force the value of the summand to 0.

Note that the term for A above contains some redundancies. Since i and j are limited to the bounds of the matrix, the $\sigma_{1,i}$ of the first summand and the $\sigma_{j,n+m}$ will always evaluate to 1. Hence we can optimise the expression by removing the unnecessary σs, as we have already done in Equation (2):

$$A = [\sigma_{j,n}\sigma_{i,j}a_{ij} + \sigma_{n+1,i}\sigma_{i,j}b_{i-n,j-n}]_{i,j}^{m+n,m+n} \tag{3}$$

In manipulating terms with basis functions, we will make use of a number of basic properties which we present here without proof.

$$\sigma_{x,x} = 1 \tag{4}$$
$$\sigma_{x,y} = \sigma_{x+z,y+z} \tag{5}$$
$$\sigma_{x,y} = \sigma_{-y,-x} \tag{6}$$
$$\sigma_{x,y}\sigma_{y,x} = 1 \Leftrightarrow x = y \tag{7}$$
$$\sigma_{y+1,x}\sigma_{x,y} = 0 \tag{8}$$

Note that we can get full logical combinations of our basis functions: given we are using $\mathsf{true} = 1$, $\mathsf{false} = 0$, we have $A \wedge B = A \times B$, $\neg A = 1 - A$ and $A \vee B = A + B - A \times B$. Thus we can get unions, complements and intersections of our regions. This, in turn, lets us handle min and max nicely.

$$\sigma_{x,\min(y,z)} = \sigma_{x,y}\sigma_{x,z} \tag{9}$$
$$\sigma_{x,\max(y,z)} = \sigma_{x,y} + \sigma_{x,z} - \sigma_{x,y}\sigma_{x,z} \tag{10}$$
$$\sigma_{\min(x,y),z} = \sigma_{x,z} + \sigma_{y,z} - \sigma_{x,z}\sigma_{y,z} \tag{11}$$
$$\sigma_{\max(x,y),z} = \sigma_{x,z}\sigma_{y,z} \tag{12}$$

4. MATRIX ADDITION

Consider addition on two vectors, $U^T + V^T$, where

$$\begin{aligned} U^T &= \left[u_1, u_2, \ldots, u_h, u'_1, u'_2, \ldots, u'_{n-h}\right] \\ &= \left[\sigma_{j,h}\ u_j + \sigma_{h+1,j}\ u'_{j-h}\right]_{i,j}^{1,n} \\ V^T &= \left[v_1, v_2, \ldots, v_k, v'_1, v'_2, \ldots, v'_{n-k}\right] \\ &= \left[\sigma_{j,k}\ v_j + \sigma_{k+1,j}\ v'_{j-k}\right]_{i,j}^{1,n} \end{aligned} \tag{13}$$

A naïve approach leads to dealing with a case analysis, with one form of the result for each ordering of h and k. Namely, when $h \leqslant k$, we get a sum of three terms, for $1 \leqslant i \leqslant h$, we get $u_i + v_i$, for $h+1 \leqslant i \leqslant k$, we get $u'_i + v_i$, while for $k \leqslant i \leqslant n$, we get $u'_i + v'_i$. Similar reasoning leads to the form for the case $k < h$:

$$U^T + V^T$$
$$= \left[\begin{cases} h \leqslant k & \begin{array}{l}\sigma_{j,h}\ (u_j + v_j) + \sigma_{h+1,j}\sigma_{j,k}\ \left(u'_j + v_j\right) \\ \quad + \sigma_{k+1,j}\sigma_{j,n}\ \left(u'_j + v'_j\right)\end{array} \\ k < h & \begin{array}{l}\sigma_{j,k}\ (u_j + v_j) + \sigma_{k+1,j}\sigma_{j,h}\ \left(u_j + v'_j\right) \\ \quad + \sigma_{h+1,j}\sigma_{j,n}\ \left(u'_j + v'_j\right)\end{array} \end{cases}\right]_{i,j}^{1,n}$$

However, there is a better, and simpler approach. We simply plug the representations into the standard matrix addition definition:

$$[a_{i,j}]_{i,j}^{m,n} + [b_{i,j}]_{i,j}^{m,n} = [a_{i,j} + b_{i,j}]_{i,j}^{m,n} \tag{14}$$

Our use of the σ basis functions, allows us to encode the cases in a simpler expression:

$$\begin{aligned}
&U^T + V^T \\
&\quad = \left[\sigma_{j,h}\ u_j + \sigma_{h+1,j}\ u'_{j-h} + \sigma_{j,k}\ v_j + \sigma_{k+1,j}\ v'_{j-k}\right]_{i,j}^{1,n} \\
&\quad = \left[\begin{array}{c} \sigma_{j,min(h,k)}\ \left(u_j + v_j\right) + \sigma_{h+1,j}\sigma_{j,k}\ \left(u'_j + v_j\right) + \\ \sigma_{k+1,j}\sigma_{j,h}\ \left(u_j + v'_j\right) + \sigma_{max(h,k)+1,j}\sigma_{j,n}\ \left(u'_j + v'_j\right) \end{array}\right]_{i,j}^{1,n} \\
&\quad = \left[\begin{array}{c} \sigma_{j,h}\sigma_{j,k}\ \left(u_j + v_j\right) + \sigma_{h+1,j}\sigma_{j,k}\ \left(u'_j + v_j\right) + \\ \sigma_{k+1,j}\sigma_{j,h}\ \left(u_j + v'_j\right) + \sigma_{h+1,j}\sigma_{k+1,j}\sigma_{j,n}\ \left(u'_j + v'_j\right) \end{array}\right]_{i,j}^{1,n}
\end{aligned}$$

Observe that at least one of the middle two summands must reduce to zero as, for both to be non-zero, we must have that $h+1 \leqslant k \wedge k+1 \leqslant h$. Therefore, if the variables h and k are eventually bound to concrete numbers, at least one of the summands will disappear. Both will disappear if and only if $h = k$. In this way, the multiple cases of the naïve approach is reduced to a single expression without cases.

Finally, consider the problem of adding vectors $V^{(i)}$, $i = 1, \ldots, m$, where each vector $A^{(i)}$ has components given by one function of the index up to the point k_i and by another function of the index from that point on. Depending on the relative order of the values $k_1, ..., k_m$, the sum $\sum_{i=1}^m A^{(i)}$ may take on any of $m!$ different forms. In a symbolic setting, the order of the k_i is not known so, naïvely, the sum is given by a piece-wise function with $m!$ cases. Using our approach, however, reduces the $m!$ number of cases to a single expression without cases; namely a sum of 2^m terms.

5. MATRIX MULTIPLICATION

For multiplication we express the formal product

$$[a_{i,j}]_{i,j}^{m,n} [b_{i,j}]_{i,j}^{n,p} = \left[\sum_{k=1}^{n} a_{i,k} b_{k,j}\right]_{i,j}^{m,p} \tag{15}$$

We take, as an example, $U^T \times V$ where U and V are taken, appropriately transposed, from Equation (13).

$$U^T V = \left[\sum_{l=1}^{n} \left(\sigma_{l,h}\ u_l + \sigma_{h+1,l}\ u'_{l-h}\right) \left(\sigma_{l,k}\ v_l + \sigma_{k+1,l}\ v'_{l-k}\right)\right]_{i,j}^{1,1}$$

as the result is a 1×1 matrix, we drop the matrix structure and present it as a value

$$= \sum_{l=1}^{n} \left(\begin{array}{c} \sigma_{l,h}\sigma_{l,k}\ u_l v_l + \sigma_{l,h}\sigma_{k+1,l}\ u_l v'_{l-k} + \\ \sigma_{h+1,l}\sigma_{l,k}\ u'_{l-h} v_l + \sigma_{h+1,l}\sigma_{k+1,l}\ u'_{l-h} v'_{l-k} \end{array}\right)$$

Again we observe that the two cases coresponding to the two possible orderings of h and k are encoded more efficiently in the resulting term without need for case expressions.

To demonstrate how our representation allows us to compute even complex results symbolically we present the example of polynomial multiplication: Let $p = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_0$ and $q = b_m x^m + b_{m-1} x^{m-1} + \cdots + b_0$. Then,

$$pq = \sum_{l=0}^{m+n} x^l \sum_{k=0}^{l} a_{l-k} b_k = \sum_{l=0}^{m+n} \Big(\sum_{\mu+\nu=l} a_{\mu-\nu} b_\mu \Big) x^l \tag{16}$$

where the a_i and b_i are considered to be infinite sequences, defined as zero for $i > m, n$, respectively. This equation can be written in matrix form as:

$$x^T A b = \left[x^{m+n} \ldots x^0\right] \begin{bmatrix} a_n & & 0 \\ \vdots & \ddots & \\ a_0 & & a_n \\ & \ddots & \vdots \\ 0 & & a_0 \end{bmatrix} \begin{bmatrix} b_m \\ \vdots \\ b_0 \end{bmatrix} \tag{17}$$

Next we write x, A and b in our notation:

$$\begin{aligned}
x^T &= \left[x^{m+n+1-j}\right]_{i,j}^{1,m+n+1} \\
A &= \left[\sigma_{j,i}\sigma_{i,j+n}\ a_{n-i+j}\right]_{i,j}^{m+n+1,m+1} \\
b &= \left[b_{m+1-i}\right]_{i,j}^{m+1,1}
\end{aligned}$$

Observe that we do not need any σ functions in the terms for x^T and b since they completely fill their respective matrix cells and the matrix bounds provide the only constraints necessary. Thus we get

$$Ab = \left[\sum_{k=1}^{m+1} \sigma_{k,i}\sigma_{i,k+n}\ a_{n-i+k} b_{m+1-k}\right]_{i,j}^{m+n+1,1} \tag{18}$$

which, not surprisingly, only varies in i since it is a column vector. We then multiply this expression with x:

$$x^T A b = \left[\sum_{l=1}^{m+n+1} x^{m+n+1-l} \sum_{k=1}^{m+1} \sigma_{k,l}\sigma_{l,k+n}\ a_{n-l+k} b_{m+1-k} a\right]_{i,j}^{1,1}$$

Since the result is a 1×1 matrix, we can treat it as a single value rather than a matrix. Furthermore we can reindex both sums by $l \to l+1$ and $k \to k+1$, respectively:

$$= \sum_{l=0}^{m+n} x^{m+n-l} \sum_{k=0}^{m} \sigma_{k+1,l+1}\sigma_{l+1,k+1+n}\ a_{n-l-1+k+1} b_{m+1-k-1}$$

We then simplify indices and σ functions using property (5) and reverse the order of the summands of the first sum:

$$= \sum_{l=0}^{m+n} x^l \sum_{k=0}^{m} \sigma_{k,m+n-l}\sigma_{m+n-l,k+n}\ a_{n-(m+n-l)+k} b_{m-k}$$

After further simplification and reordering we get:

$$= \sum_{l=0}^{m+n} x^l \sum_{k=0}^{m} \sigma_{-k,n-l}\sigma_{-l,-k}\ a_{l-k} b_k$$

Here we simplify the σ functions using property (6):

$$= \sum_{l=0}^{m+n} x^l \sum_{k=0}^{m} \sigma_{l-n,k}\sigma_{k,l}\ a_{l-k} b_k$$

A final application of property (5) gives us:

$$= \sum_{l=0}^{m+n} x^l \sum_{k=0}^{m} \sigma_{l-k,n}\sigma_{k,l}\ a_{l-k}b_k$$

At this point we have the closest expression possible to the standard presentation, Equation (16), without relying on the coefficients outside their specified range to be zero. The next step is to eliminate the first σ. To do so, we merely have to observe that if $l-k \nleqslant n$, then $a_{l-k}=0$:

$$= \sum_{l=0}^{m+n} x^l \sum_{k=0}^{m} \sigma_{k,l}\ a_{l-k}b_k$$

We still need to eliminate the final σ and correct the inner summation limit. We start by splitting the inner sum into two:

$$= \sum_{l=0}^{m+n} x^l \left(\sum_{k=0}^{min(l,m)} \sigma_{k,l}\ a_{l-k}b_k + \sum_{k=min(l,m)+1}^{m} \sigma_{k,l}\ a_{l-k}b_k \right)$$

In the first inner sum, k is always less than or equal to l so the σ can be eliminated. In the second inner sum, the lower bound of the summation implies that either $m < k \Rightarrow b_k = 0$, or $l < k \Rightarrow l-k < 0 \Rightarrow a_{l-k} = 0$. Hence the second inner sum is always zero irrespective of the σ term and irrespective of the upper bound. Thus we can safely eliminate the σ and change its upperbound to l:

$$= \sum_{l=0}^{m+n} x^l \left(\sum_{k=0}^{min(l,m)} a_{l-k}b_k + \sum_{k=min(l,m)+1}^{l} a_{l-k}b_k \right)$$

Finally we can recombine the inner sums to get the desired result:

$$= \sum_{l=0}^{m+n} x^l \sum_{k=0}^{l} a_{l-k}b_k$$

6. SHOWING STRUCTURAL PROPERTIES

Up to this point, we have shown how, using our basis functions, we can provide a simple unified approach to computing with abstract matrices that does not involve having to carry out complex analyses with combinatorially prohibitive numbers of cases. However, the question remains of whether we can recover the region oriented structural description from the result of arithmetic combinations of abstract matrices, thereby obtaining closure for abstract matrix operations. For example, can we tell, after multiplying two upper triangular abstract matrices, that the result is upper triangular, and indeed, construct the corresponding abstract matrix object for it. We can, and the basis for doing so is to organise the term computed into the form of a sum of terms, where each term is an arithmetic combination of generalised terms of the operand abstract matrices, which become the generalised term of the new region, multiplied by a product of basis functions, where these basis function products describe shapes which are disjoint from the corresponding shapes of the other terms. This shape becomes the shape of the new region. The ability to reconstruct abstract matrices from the underlying expressions allows exploring and demonstrating structural properties of classes of matrice. We demonstrate this approach with an example of matrix multiplication.

Consider matrices of a structure similar to matrix A given in section 2, Equation (1). Let us suppose we want to examine the arithmetic properties of structured matrices of this classes and concentrate on the multiplication. Assume that we multiply matrix A with the following matrix B.

$$B = \begin{bmatrix} c_{11} & \cdots & c_{1m} & & & \mathbf{0} \\ & \ddots & \vdots & & & \\ & & c_{mm} & & & \\ & & & d_{11} & \cdots & d_{1n} \\ & & & & \ddots & \vdots \\ \mathbf{0} & & & & & d_{nn} \end{bmatrix} \tag{19}$$

Observe that in B we have reversed the indices n and m and thus the size of the two triangles. This choice aims to minimise confusion caused by employing too many variable names. Although it is slightly less general than using entirely different index variables, the example nevertheless serves the same purpose of showing how properties of structure matrices can be computed.

We first translate both matrices A and B into our syntax:

$$A = [\sigma_{j,n}\sigma_{i,j}\ a_{ij} + \sigma_{n+1,i}\sigma_{i,j}\ b_{i-n,j-n}]_{i,j}^{n+m,n+m} \tag{20}$$

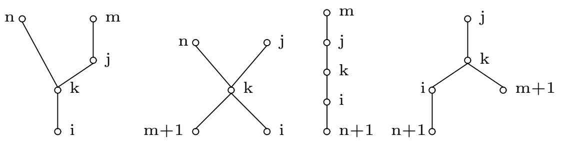
$$B = [\sigma_{j,m}\sigma_{i,j}\ c_{ij} + \sigma_{m+1,i}\sigma_{i,j}\ d_{i-m,j-m}]_{i,j}^{m+n,m+n} \tag{21}$$

Their product, AB, is:

$$\left[\sum_{k=1}^{n+m} \begin{pmatrix} \sigma_{k,n}\sigma_{i,k}\sigma_{j,m}\sigma_{k,j}\ a_{ik}c_{kj} + \\ \sigma_{k,n}\sigma_{i,k}\sigma_{m+1,k}\sigma_{k,j}\ a_{ik}d_{k-m,j-m} + \\ \sigma_{n+1,i}\sigma_{i,k}\sigma_{j,m}\sigma_{k,j}\ b_{i-n,k-n}c_{kj} + \\ \sigma_{n+1,i}\sigma_{i,k}\sigma_{m+1,k}\sigma_{k,j}\ b_{i-n,k-n}d_{k-m,j-m} \end{pmatrix} \right]_{i,j}^{n+m,n+m}$$

The sum can clearly be split into 4 separate sums, one for each of the summands in the summation. Each of the resulting terms is nearly, but not quite in the form we are looking for: a generalised term expression multiplied by a product of basis functions, where each product describes a disjoint shape. To get them into the correct form, we need to find the product of all the possible σs that are implied by the term but are independent of the summation variable. This product will be the most general σ product that can be moved outside the summation and will then provide us with the required form.

First we show that that the regions associated with each term is dijoint: Consider the full set of inequalities implied by the σs of each term. We can do this most easily by completing the partial order on the σ limits, and displaying them as a set of Hasse diagrams:



We can now see that all four partial orders are incompatible (e.g. $j \leqslant m$ in (1), but $m+1 \leqslant j$ in (2), etc.) and hence the space defined by the possible i,j pairs describe disjoint regions.

To extract the required most general product of σs from each term, we generate a *sigma* expression from every pair of elements in the partial order for the term, excluding only the summation variable.

We explain the process in detail for the first region:

$$\left[\sum_{k=1}^{n+m} \sigma_{k,n}\sigma_{i,k}\sigma_{j,m}\sigma_{k,j}\ a_{ik}c_{kj}\dots\right]_{i,j}^{n+m,n+m}$$

Our task is to factor out σ functions from the sum. We generate all inequalities not featuring k from the partial order for this term: $i \leqslant j$, $j \leqslant m$, $i \leqslant n$ and $i \leqslant m$. In practice, we only need a minimal set of inequalities that imply all the inequalities in the list so we can omit $i \leqslant m$. We turn each one of these into the corresponding σ expression and multiply the term by it. This does not modify the meaning of the term as these constraints were implied by it anyway. Observe that although we add these as new coefficients, we can nevertheless not simply remove any of the σ functions from inside the sum, since although the inner σs imply the outer ones, the reverse is not, in general, true. Continuing in this way we obtain the following set of 4 outer σ products, where, in all cases, both i and j ranges from 1 to $m+n$:

$$\sigma_{j,m}\sigma_{i,n}\sigma_{i,j} \tag{S1}$$

$$\sigma_{i,j}\sigma_{i,n}\sigma_{m+1,j}\sigma_{m+1,n} \tag{S2}$$

$$\sigma_{n+1,i}\sigma_{i,j}\sigma_{j,m} \tag{S3}$$

$$\sigma_{n+1,i}\sigma_{i,j}\sigma_{m+1,j} \tag{S4}$$

Reconstructing the shape of a region is done by translating each σ back into one of the half-plane constraints given in Figure 1. We consider each shape in turn

S1 $\sigma_{i,j}$ means that this region is constrained to be on, or to the upper left, of the main diagonal of the matrix. $\sigma_{j,m}$ means this region extends to the right only as far as column m, $\sigma_{i,n}$ means it extends downwards only as far as row n. Hence, if $m \leqslant n$, the region will be a triangle with its base on the main diagonal, one side on the top boundary of the matrix (limited only be the lower bound of 1 on i) and the remaining side being the vertical column at $j = m$. If, however, $n < m$, the bottom corner of the triangle will be clipped off horizontally by the row $i = n$.

S2 Firstly, this region only appears at all when $m < n$. It is bounded horizontally at $m + 1$ from the left and vertically at n from below. In addition it is bounded by the main diagonal. Otherwise it is bounded only by the implied limits of the matrix. This means that the region is a rectangle in the upper right corner of the matrix that potentially has its lower left corner clipped off diagonally by the main diagonal of the matrix.

S3 Note here that $n < m$ is implied by the set of σs here, and was only eliminated to reduce unnecessary expansion of the term. Hence this region only appears if $n < m$. This region is bounded above at row $n + 1$, to the right at column $m + 1$ and also by the main diagonal. Thus it is a proper triangle sitting in the middle of the matrix, above the main diagonal.

S4 This region is also bounded above at $n + 1$, to the left by column $m+1$ and by the main diagonal. Otherwise it goes all the way to the right edge and the bottom of the matrix. It is therefore a triangular region that might have its top left corner clipped horizontally.

Now that we know the form of the single regions, we are interested in what the structure of the overall matrix looks like and if it is always of a particular form regardless of the concrete values of m and n. Firstly, we observe that every region contains the term $\sigma_{i,j}$. Thus every non-zero region is situated above the main diagonal and the matrix has an upper triangular form.

Secondly, we note that regions S2 and S3 contain mutually exclusive variable orders: We therefore perform a case analysis on the relation between n and m.

$\mathbf{m} < \mathbf{n}$: Region S2 is non-zero and region S3 vanishes. The result matrix is of the form:

$$\begin{bmatrix}
ac & \cdots & ac & ad & \ldots & \ldots & \ldots & ad \\
 & \ddots & \vdots & \vdots & & & & \vdots \\
 & & ac & \vdots & & & & \vdots \\
 & & & ad & & & & \vdots \\
 & & & & \ddots & & & \vdots \\
 & & & & & ad & \ldots & ad \\
 & & & & & bd & \ldots & bd \\
 & & & & & & \ddots & \vdots \\
\mathbf{0} & & & & & & & bd
\end{bmatrix}$$

Note, that we have given the matrix entries without indices.

$\mathbf{n} < \mathbf{m}$: Region 3 is non-zero and region 2 vanishes. Since region 1 is bounded from the right, and both region 3 and 4 are bounded from the top we also get a rectangular zero region in the top right-hand corner of size $n \times n$. The result matrix is therefore of the form:

$$\begin{bmatrix}
ac & \cdots & \ldots & \ldots & ac & & & & \mathbf{0} \\
 & \ddots & & & \vdots & & & & \\
 & & ac & \ldots & ac & & & & \\
 & & & bc & \ldots & bc & bd & \ldots & bd \\
 & & & & \ddots & \vdots & \vdots & & \vdots \\
 & & & & & bc & \vdots & & \vdots \\
 & & & & & & bd & & \vdots \\
 & & & & & & & \ddots & \vdots \\
\mathbf{0} & & & & & & & & bd
\end{bmatrix}$$

$\mathbf{n} = \mathbf{m}$: Both region 2 and 3 vanish. The result matrix is of a form similar to the original matrices A and B.

7. ALTERNATIVE APPROACH

The σ basis functions are well behaved, simple and easy to manipulate, however, in certain cases they might lead to overly complex terms. An obvious alternative is to try interval, instead of half plane, basis functions. It turns out, however, that a naïve interval function does not solve the problem. We therefore define a variant of an interval basis function that exhibits some interesting properties. Its basic idea is to eliminate explicit case analysis of the results of arithmetic operations altogether by committing to one ordering of index variables, and, automatically removing any superfluous terms in case the commitment was erroneous.

Definition 3

$$\xi(i,y,z) = \begin{cases} 1 & \textit{if } y \leqslant i < z \\ -1 & \textit{if } z \leqslant i < y \\ 0 & \textit{otherwise} \end{cases}$$

The following properties follow immediately from the definition:

$$\xi(i,y,y) = 0 \tag{22}$$
$$\xi(i,y,z) = -\xi(i,z,y) \tag{23}$$
$$\xi(i,y,x) + \xi(i,x,z) = \xi(i,y,z) \tag{24}$$

Using this basis function, we will attempt a vector addition. Consider two vectors:

$$U^T = \left[u_1, u_2, \ldots, u_{h-1}, u'_1, u'_2, \ldots, u'_{n-h}\right]$$
$$V^T = \left[v_1, v_2, \ldots, v_{k-1}, v'_1, v'_2, \ldots, v'_{n-k}\right]$$

We can write these vectors as

$$U^T = \left[\xi(i,1,h) \times u_i + \xi(i,h,n) \times u'_{i-h+1}\right]^{1,n}_{i,j}$$
$$V^T = \left[\xi(i,1,k) \times v_i + \xi(i,k,n) \times v'_{i-k+1}\right]^{1,n}_{i,j}$$

Adding the vectors, we get:

$$U^T + V^T = \begin{bmatrix} \xi(i,1,h) \times u_i + \xi(i,h,n) \times u'_{i-h+1} + \\ \xi(i,1,k) \times v_i + \xi(i,k,n) \times v'_{i-k+1} \end{bmatrix}^{1,n}_{i,j} \tag{25}$$

To get the benefit of using the negative values and compressing into 3 cases, we have to do a different type of case analysis, rather than the obvious one, where we choose a configuration of $h \leqslant k$ and use the negative values of $\xi(x,y,z)$ to fix the problem if it turns out that $k < h$.

h < k Here (25) yields a sum of three terms:

$$\begin{aligned} &\xi(i,1,h) \times (u_i + v_i) \\ &+ \xi(i,h,k) \times (u'_i + v_i) \\ &+ \xi(i,k,n) \times (u'_i + v'_i) \end{aligned} \tag{26}$$

k < h Here we claim that the solution for the previous case is also the correct solution for this case. Using the properties shown above for $\xi(x,y,z)$ we have:

$$\begin{aligned} & (\xi(i,1,k) + \xi(i,k,h)) \times (u_i + v_i) \\ & - \xi(i,k,h) \times (u'_i + v_i) \\ & + (\xi(i,k,h) + \xi(i,h,n)) \times (u'_i + v'_i) \\ = \quad & \xi(i,1,k) \times (u_i + v_i) \\ & + \xi(i,k,h) \times (u_i + v_i - u'_i - v_i + u'_i + v'_i) \\ & + \xi(i,h,n) \times (u'_i + v'_i) \\ = \quad & \xi(i,1,k) \times (u_i + v_i) \\ & + \xi(i,k,h) \times (u_i + v'_i) \\ & + \xi(i,h,n) \times (u'_i + v'_i) \end{aligned}$$

This is exactly what one gets directly from $h < k$.
h = k In this case, Equation (26) reduces to the correct equation for $h = k$ because $\xi(i,h,k) = \xi(i,h,h) = 0$.

A naïve, case based approach would have produced one case for each orderings of the lengths of the first components of the vectors: in this case, two cases, with 3 terms each. If we were adding m such vectors, we would have one case for each possible ordering, leading to $m!$ different cases, each with $m+1$ terms.

Recall that the σ basis function approach produced a sum of 4 terms, one for each of the four possible combinations of each of the two terms from the first vector with each of the two terms from the second, where two of the terms are mutually exclusive in that, when the relative lengths of the first component in each vector is resolved, one of those terms will be reduced to zero. If we were adding m such vectors, there are 2^m possible combinations that would need terms, hence 2^m terms in the result with only one case.

When using the $\xi(x,y,z)$ representation, we have produced a sum of 3 terms only, which corresponds to a single ordering of the possible first component lengths, and have shown that this expression is exactly equal to the corresponding expression for the alternative ordering. This was possible because, if we commit to a single ordering and choose incorrectly, the basis functions allow reversing the direction of traversal of elements in the vector, undoing the commitment to the first order by subtracting the values erroneously committed to, and applying the values of the alternative ordering. Since we are thus encoding all possible orderings with the number of terms necessary for only one ordering, the number of terms necessary for adding m such vectors is $m+1$ and there is only one case.

The $\xi(i,x,y)$ basis vectors work so well for addition because, when the interval limits are reversed, they apply the additive inverse on to the carried term. Unfortunately, when multiplying vectors, if we want the same trick to work we must use the multiplicative inverse instead and thus have to change $\xi(i,x,y)$ to a multiplicative version:

Definition 4

$$\xi_\times(i,y,z)\,(e) = \begin{cases} e & \textit{if } y \leqslant i < z \\ \frac{1}{e} & \textit{if } z \leqslant i < y \\ 1 & \textit{otherwise} \end{cases}$$

It is worth noting that this definition leads to the following relationship:

$$\xi_\times(i,y,z)\,(e) = e^{\xi(i,y,z)}$$

We have a new set of properties that correspond to the $\xi_\times(i,y,z)\,(e)$

$$\xi_\times(i,y,y)\,(e) = 1 \tag{27}$$
$$\xi_\times(i,y,z)\,(e) = \frac{1}{\xi_\times(i,z,y)\,(e)} \tag{28}$$
$$\xi_\times(i,y,x)\,(e) \times \xi_\times(i,x,z)\,(e) = \xi_\times(i,y,z)\,(e) \tag{29}$$

Note that to construct vectors and matrices out of terms involving $\xi_\times(i,y,z)\,(e)$, we have to multiply them together, instead of add them as in the case of $\xi(i,y,z)$.

Using the same vectors as before, we get that

$$U = \left[\xi_\times(i,1,h)(u_i) \times \xi_\times(i,h,n)\left(u'_{i-h+1}\right)\right]_{i,j}^{n,1}$$
$$V = \left[\xi_\times(i,1,k)(v_i) \times \xi_\times(i,k,n)\left(v'_{i-k+1}\right)\right]_{i,j}^{n,1}$$

and multiplying U and V yields

$$U \cdot V = \sum_{i=1}^{n} \left(\begin{array}{l} \xi_\times(i,1,h)(u_i) \times \xi_\times(i,h,n)(u'_{i-h+1}) \times \\ \xi_\times(i,1,k)(v_i) \times \xi_\times(i,k,n)(v'_{i-k+1}) \end{array} \right) \tag{30}$$

Just as with the additive version, we could do essentially the same case analysis and rewrite to show that the result is a product (rather than a sum) of 3 terms that correspond to precisely one possible ordering of the initial vector component lengths and that the result is equal to the expression that one would obtain with the other order. Instead we show a more direct derivation of the 3 term version from the initial expression.

The procedure starts with (30) and rewrites all $\xi_\times(i,1,h)(e)$ into $\xi_\times(i,1,k)(e) \times \xi_\times(i,k,h)(e)$, and similarly $\xi_\times(i,h,n)(e)$ into $\xi_\times(i,h,k)(e) \times \xi_\times(i,k,n)(e)$ and simplify. This works analogously for addition of course.

$$\begin{aligned} U \cdot V &= \sum_{i=1}^{n} U_i \times V_i \\ &= \sum_{i=1}^{n} \left(\begin{array}{l} \xi_\times(i,1,h)(u_i) \times \xi_\times(i,h,n)(u'_{i-h+1}) \times \\ \xi_\times(i,1,k)(v_i) \times \xi_\times(i,k,n)(v'_{i-k+1}) \end{array} \right) \\ &= \sum_{i=1}^{n} \left(\begin{array}{l} \xi_\times(i,1,k)(u_i) \times \xi_\times(i,k,h)(u_i) \times \\ \xi_\times(i,h,n)(u'_{i-h+1}) \times \\ \xi_\times(i,1,k)(v_i) \times \\ \xi_\times(i,k,h)(v'_{i-k+1}) \times \xi_\times(i,h,n)(v'_{i-k+1}) \end{array} \right) \\ &= \sum_{i=1}^{n} \left(\begin{array}{l} \xi_\times(i,1,k)(u_i \times v_i) \times \\ \xi_\times(i,k,h)(u_i \times v'_{i-k+1}) \times \\ \xi_\times(i,h,n)(u'_{i-h+1} \times v'_{i-k+1}) \end{array} \right) \end{aligned}$$

While we believe these basis functions are promising and worthy of further study, they are not without problems. Using the additive version works well for addition, but the undo/redo trick will not work when multiplying. The same applies when using the multiplicative version during addition. Zero regions require careful handling for $\xi_\times$: in order to use multiplication to combine disjoint regions into a single matrix, we need to define $\xi_\times$ to be 1 outside its area instead of 0. Hence we need to explicitly represent regions containing zero, unlike with the additive version or with σ. Inversing zero regions cause difficulty for the $\xi_\times$ as, when reverting a section of zeros, there is nothing that we can multiply 0 by to turn it into the neutral 1 so that we can apply the alternative values. In practice we define $\frac{1}{0} \times 0 = 1$. Finally, while the $\xi_\times$ basis function works well on vector manipulations, we have not yet successfully extended them to full matrices.

8. CONCLUSION

We have presented a computational approach to arithmetic on abstract matrices. We have defined a simple basis function that allows us to represent every abstract matrix regardless of its structural composition as a sum of region terms. Given this representation we could define matrix addition and multiplication straightforwardly as addition and multiplication of the sums. In fact we could show that our representation enables symbolic computations on abstract matrices that are considered mathematically routine but for which only limited automated support exists. In a next step we therefore intend to implement bespoke algorithms for abstract matrix arithmetic and combine them with our parsing procedure presented in [5]. Moreover, we intend to use our representation as a basis for developing other operations on abstract matrices such as computing Jordan normal forms or determinants.

Another advantage of our representation is that the result of an arithmetic operation on two abstract matrices can be examined by systematic arithmetic manipulations and exploitation of the partial order structure of the basis function to yield structural properties of the resulting matrix. This could be further exploited to perform and automate general proofs of closure properties for certain classes of structural matrices in a computational manner. Furthermore, this fact indicates that the representation could be used as a starting point for translating the algebraic expression back into a graphical representation of the abstract matrix, i.e. with ellipses etc.

While our approach is encompassing enough to deal with abstract matrices of arbitrary structure we have identified cases in which the growth of terms is potentially exponential. We have started addressing this issue by investigating an alternative basis function that can avoid this exponential growth. Although the alternative basis function requires distinct representations of abstract matrices, depending on the arithmetic operation we want to perform, we could already show that it is effective for addition and multiplication on abstract vectors. However, its correct extension to the two-dimensional case of abstract matrices remains the subject of future work.

9. REFERENCES

[1] R. Fateman. Manipulation of matrices symbolically. Available from `http://http.cs.berkeley.edu/~fateman/papers/symmat2.pdf`, 2003.

[2] M. Kauers and C. Schneider. Application of unspecified sequences in symbolic summation. In *Proceedings of ISSAC 2006*, p. 177–183. ACM Press, 2006.

[3] M. Kauers and C. Schneider. Indefinite summation with unspecified summands. *Discrete Mathematics*, 306(17):2021–2140, 2006.

[4] A. Sexton and V. Sorge. Processing textbook-style matrices. In *Proceedings of MKM-2006*, volume 3863 of *LNCS*, p. 111–125. Springer Verlag, 2005.

[5] A. Sexton and V. Sorge. Abstract matrices in symbolic computation. In *Proceedings of ISSAC 2006*, p. 318–325. ACM Press, 2006.

[6] S. M. Watt. Making computer algebra more symbolic. In *Transgressive Computing*, p. 43–49, 2006.

[7] S. M. Watt. Two families of algorithms for symbolic polynomials. In *Computer Algebra 2006: Latest Advances in Symbolic Algorithms*, p. 193–210. World Scientific, 2006.