

Reasoning with Generic Cases in the Arithmetic of Abstract Matrices

Alan P. Sexton¹, Volker Sorge¹, and Stephen M. Watt²

¹School of Computer Science, University of Birmingham

www.cs.bham.ac.uk/~aps/~vxs

²Department of Computer Science, University of Western Ontario

www.csd.uwo.ca/~watt

Abstract. In previous work we have developed procedures to analyse, compute with and reason about abstract matrices, that is, matrices represented with symbolic dimensions and with a mixture of terms and ellipsis symbols to describe their structure. A central component in this are the so-called “support functions”, which enable the representation of abstract matrices in closed forms. A key issue in making reasoning about such structures effective is controlling the complexity of the internal term structure of the closed form, which, in turn, hinges critically on the design of the support functions used.

Our earlier support functions were simple, easy to work with and sufficient to capture arithmetic of general partitioned matrices fully. They explicitly represent each potential homogeneous region, usually a triangle or a rectangle, of an abstract matrix with a single term. However, adding or multiplying a sequence of matrices can result in exponentially many different cases of possible regions that have to be represented, and the existence of many of these is mutually exclusive. As this representation can become unwieldy in certain situations, we experiment with a different type of support function that allows us to represent only one of the possible cases explicitly, and have all other cases captured by the representation implicitly.

In this paper we discuss this new support function and develop the full abstract matrix addition algorithm for this representation. We show that we indeed obtain much more concise and intuitive closed forms, retaining the properties necessary for reasoning with abstract matrices and being able to recover the human readable region structure from the combination of abstract matrices under addition. This representation reduces the time and space complexity of performing K abstract matrix additions from $O(N^{dK})$ to $O(K^d N^d)$, for d the number of boundary directions ($1 \leq d \leq 4$) and N the maximum number of boundaries in any direction in the argument matrices.

1 Introduction

Through the contributions of many authors over the past few decades, computer algebra systems have become very effective at computing with values from a variety of mathematical domains. For example, polynomial, linear and differential

algebra can be performed effectively with coefficients being arbitrary precision rational numbers, elements of finite fields, algebraic numbers, and so on. Computer algebra systems have had more difficulty working, not with a particular values from one of these domains, but rather with objects representing sets of values of particular forms. Simple examples would be to factor $x^{2n} - 1$ or compute the determinant of the diagonal matrix $\text{diag}(1, 4, \dots, n^2)$ without specifying n . We call this working with “symbolic” or “abstract” values. In previous work we have given algorithms for a number of problems on such symbolic polynomials [8–10] and abstract matrices [5–7]. The common feature of this work is that it allows computations that simultaneously treat a number of cases.

In this article we concentrate on the problem of expressing abstract matrix arithmetic in a manner that allows more effective reasoning about the cases represented. As before, we consider the problem of arithmetic on structured symbolic matrices. These matrices will be made up of various regions, where a region is a closed polygonal area that can be homogeneously evaluated by a single term. Non-zero regions must be convex. The line segments that compose the region boundaries define lines that we call boundary lines. We allow the size of the matrices and the locations of the boundaries between the regions to be given symbolically, for example, an $(h+k) \times (n+m)$ matrix made up of $h \times n$, $h \times m$, $k \times n$ and $k \times m$ blocks and defines a horizontal and a vertical boundary line (aside from the outer boundaries of the matrix). Our goal is to support automated reasoning and precisely stated algebraic algorithms on abstract matrices of this sort. We will refer throughout to special cases of block matrices, where the regions are rectangular submatrices, and banded matrices, where non-zero entries are confined to a diagonal band, comprising the main diagonal and zero or more diagonals on either side. We treat elsewhere the separate problem of obtaining the closed form expressions for each region from forms that express skipped entries with ellipses.

It would be a natural choice to represent these abstract matrices as having elements that are piecewise defined functions of the indices. However, matrix arithmetic quickly leads to an intractable situation. A chain of N arithmetic operations on matrices with K cases in their piecewise elements requires the analysis of K^N cases, each requiring simplification of boolean expressions to determine feasibility. Useful progress has been made in the scalar case for this problem [1], but we find that with matrices another approach is more fruitful.

Instead of representing the cases as logical conditions, we encode them algebraically with “support functions”, in a manner we shall shortly make more precise. We are then able to handle multiple logical cases simultaneously via algebraic operations. We encounter one problem, however: when doing arithmetic without knowing how the region boundaries in one matrix relate to the region boundaries in the other, the generic case becomes overly complicated, with certain terms becoming mutually exclusive. For example, when adding an $(h_1 + k_1) \times (n_1 + m_1)$ to an $(h_2 + k_2) \times (n_2 + m_2)$ block matrix, the form of the result must satisfy all possible relative orderings $h_1 \lesseqgtr h_2, n_1 \lesseqgtr n_2$. Our earlier choice of support functions can cope with this issue, and, indeed, we have pre-

viously presented full abstract matrix addition and multiplication algorithms in this context, but at a cost of a high complexity in the number of terms in the result as a function of the number of terms in the abstract matrix operands. In this paper we explore using another class of support functions that allow the generic case to be written more more concisely (with asymptotically fewer terms) and with consequent advantages for space and time efficiency, and we present the solution for the addition case.

2 Motivation

Before developing the full *support function* we are aiming for, we motivate its need considering a simpler, interval based support function. While it is slightly different to the support function we have employed to define a full abstract matrix addition and multiplication algorithms in [7], it exhibits similar problems.

Definition 1. *Let $x, y, z \in \mathbb{N}$ then we define the support function $\hat{\xi}(x, y, z)$ as*

$$\hat{\xi}(x, y, z) = \begin{cases} 1 & \text{if } y \leq x < z \\ 0 & \text{otherwise} \end{cases}$$

Thus $\hat{\xi}$ acts as a *characteristic function* for a particular interval. We shall use $\hat{\xi}_{x,y,z}$ as a more compact notation for $\hat{\xi}(x, y, z)$.

Now consider the following two abstract vectors

$$u = [a_1, \dots, a_{h-1}, b_h \dots b_{n-1}] \quad v = [c_1, \dots, c_{k-1}, d_k \dots d_{n-1}] \quad (1)$$

Using our $\hat{\xi}$ functions, the elements of these vectors, u_j, v_j , for $j \in \{1..n-1\}$ can be written:

$$u_j = \hat{\xi}_{j,1,h} a_j + \hat{\xi}_{j,h,n} b_j \quad v_j = \hat{\xi}_{j,1,k} c_j + \hat{\xi}_{j,k,n} d_j \quad (2)$$

Observe that j is here used as the horizontal index. Throughout this paper we will reserve i, j as *vertical and horizontal index variables*.

For simplicity, we will drop the indices on the a, b, c and d terms. If we add these two vectors, we get

$$(u + v)_j = \hat{\xi}_{j,1,h} a + \hat{\xi}_{j,h,n} b + \hat{\xi}_{j,1,k} c + \hat{\xi}_{j,k,n} d \quad (3)$$

However, the four terms in these expressions define overlapping regions in the resultant vector, so we cannot easily visualise the shape of the regions in the result. To fix this, we need to identify the set of disjoint regions in the result, and calculate the term that is valid with each of these regions. We will do this in a slightly long-winded way in order to motivate the algorithms to follow. To find these disjoint regions we construct the characteristic functions of all intersections of regions from the original vectors. This can be done by multiplying the $\hat{\xi}$ components of each term from u with that of each term from v . Since the two

regions from u are disjoint, as are the two regions from v , we do not have to worry about intersections between the a and b regions or between c and d regions and we are left with 4 regions, each of which we can describe with $\hat{\xi}$ functions:

$$\begin{aligned} a \cap c \text{ region: } \hat{\xi}_{j,1,h} \hat{\xi}_{j,1,k} &= \hat{\xi}_{j,1,\min(h,k)} & b \cap c \text{ region: } \hat{\xi}_{j,h,n} \hat{\xi}_{j,1,k} &= \hat{\xi}_{j,h,k} \\ a \cap d \text{ region: } \hat{\xi}_{j,1,h} \hat{\xi}_{j,k,n} &= \hat{\xi}_{j,k,h} & b \cap d \text{ region: } \hat{\xi}_{j,h,n} \hat{\xi}_{j,k,n} &= \hat{\xi}_{j,\max(h,k),n} \end{aligned}$$

Having identified the disjoint regions, we need to find the terms that will be valid within each region. We can do this by multiplying each region characteristic function by the full term for the sum of u and v and simplifying. The characteristic functions will ensure that any component that does not belong with the requisite region will simplify to zero. We show the working in detail for the $a \cap c$ region:

$$\begin{aligned} \hat{\xi}_{j,1,\min(h,k)} (u + v)_j &= \hat{\xi}_{j,1,\min(h,k)} \left(\hat{\xi}_{j,1,h} a + \hat{\xi}_{j,h,n} b + \hat{\xi}_{j,1,k} c + \hat{\xi}_{j,k,n} d \right) \\ &= \hat{\xi}_{j,1,\min(h,k)} \hat{\xi}_{j,1,h} a + \hat{\xi}_{j,1,\min(h,k)} \hat{\xi}_{j,h,n} b + \\ &\quad \hat{\xi}_{j,1,\min(h,k)} \hat{\xi}_{j,1,k} c + \hat{\xi}_{j,1,\min(h,k)} \hat{\xi}_{j,k,n} d \\ &= \hat{\xi}_{j,1,\min(h,k)} a + \hat{\xi}_{j,1,\min(h,k)} c \\ &= \hat{\xi}_{j,1,\min(h,k)} (a + c) \end{aligned}$$

Working out the terms for the other regions and adding, we get:

$$(u + v)_j = \hat{\xi}_{j,1,\min(h,k)} (a + c) + \hat{\xi}_{j,k,h} (a + d) + \hat{\xi}_{j,h,k} (b + c) + \hat{\xi}_{j,\max(h,k),n} (b + d) \quad (4)$$

Thus our expression contains four terms, apparently describing four disjoint regions. However, the two middle terms are mutually exclusive. In any concrete case, either $h < k$, $h = k$ or $h > k$. Depending on which case holds, one or both of the middle terms will reduce to 0. Thus we have a representation that, while correct, explicitly represents every region that can occur in any case obtainable by varying the ordering of the matrix parameters (in this case, h and k).

We can generalise this analysis. If, instead of two vectors with two regions each, we were to add K matrices with N regions each with all the boundaries parallel, we would arrive at a result with N^K terms describing potential regions. This is worse than just leaving the original sum with KN terms symbolic. Note that for any particular choice of region boundaries there will be at most $K(N-1)$ regions in the resulting sum. We next describe a choice of support functions that allow the resulting generic expression to have exactly this many terms. If we were to add K block matrices with $N \times M$ regions, using the first choice of support functions would give a result with $(N \times M)^K$ describing potential regions.

3 Definitions

At this point we can introduce the full ξ functions. The idea behind them is that, for the most part, they act like the $\hat{\xi}$ interval characteristic functions, but that they let us commit to a single case, i.e., a single ordering of the unknowns

in the matrix expression, and, if that ordering is wrong, the negative part of the ξ function will compensate so that the correct result is obtained. In particular, the result will contain one term for each region that occurs in that single case, and hence shows directly the structure of the result for that case. However, if we wish to investigate a different case, we can apply a reordering of the matrix parameters, normalise the expression with respect to this new reordering, and then have our expression directly represent this new case.

Definition 2. Let $x, y, z \in \mathbb{N}$ then we define the support function $\xi(x, y, z)$ as

$$\xi(x, y, z) = \begin{cases} 1 & \text{if } y \leq x < z \\ -1 & \text{if } z \leq x < y \\ 0 & \text{otherwise} \end{cases}$$

We will again use $\xi_{x,y,z}$ as compact notation. The following properties follow immediately from the definition:

$$\xi_{u,x,x} = 0 \tag{5}$$

$$\xi_{u,x,y} = -\xi_{u,y,x} \tag{6}$$

$$\xi_{u,x,y} + \xi_{u,y,z} = \xi_{u,x,z} \tag{7}$$

In this paper, we will restrict our use of ξ support functions to a particular class thereof; namely that of parallel ξ support functions:

Definition 3. A parallel ξ support function is a $\xi_{X,Y,Z}$ function where X is an integer expression restricted to one of the forms i , j , $i + j$ or $i - j$ for respective matrix row and column index variables i and j , and where Y and Z are integer expressions that do not contain any matrix index variables.

We observe that ξ itself is not a characteristic function, as it can lead to negative values for the interval it represents. We therefore define a characteristic function based on ξ using its absolute value.

Definition 4. Let $x, y, z \in \mathbb{N}$. The characteristic function of $\xi_{x,y,z}$ is $|\xi_{x,y,z}|$.

In general, a $\xi_{x,y,z}$ function defines both an interval, $\hat{\xi}_{x,\min(y,z),\max(y,z)}$ or $|\xi_{x,y,z}|$, and a contribution factor, namely $+1$ or -1 , depending on the strict ordering of y and z . We follow a similar procedure to that above, but start by already committing to an ordering, let us say $1 < h < k < n$. In fact, we are not changing the meaning of the expression, merely normalising it with respect to a particular ordering of the parameters. Now we get 3 regions as the fourth simplifies to 0:

$$\begin{array}{ll} a \cap c \text{ region: } |\xi_{j,1,h}| |\xi_{j,1,k}| = |\xi_{j,1,h}| & b \cap c \text{ region: } |\xi_{j,h,n}| |\xi_{j,1,k}| = |\xi_{j,h,k}| \\ a \cap d \text{ region: } |\xi_{j,1,h}| |\xi_{j,k,n}| = 0 & b \cap d \text{ region: } |\xi_{j,h,n}| |\xi_{j,k,n}| = |\xi_{j,k,n}| \end{array}$$

Now multiply these in to the full ξ version of the resultant vector. We again show the working in detail for the $a \cap c$ region, bearing in mind our choice of

ordering of $1 < h < k < n$:

$$\begin{aligned}
|\xi_{j,1,h}| (u+v)_j &= |\xi_{j,1,h}| (\xi_{j,1,h} a + \xi_{j,h,n} b + \xi_{j,1,k} c + \xi_{j,k,n} d) \\
&= |\xi_{j,1,h}| \xi_{j,1,h} a + |\xi_{j,1,h}| \xi_{j,h,n} b + |\xi_{j,1,h}| \xi_{j,1,k} c + |\xi_{j,1,h}| \xi_{j,k,n} d \\
&= \xi_{j,1,h} a + \xi_{j,1,h} c \\
&= \xi_{j,1,h} (a + c)
\end{aligned}$$

Doing the same for the other two regions and adding the three terms we get a final result of:

$$(u+v)_j = \xi_{j,1,h} (a + c) + \xi_{j,h,k} (b + c) + \xi_{j,k,n} (b + d) \quad (8)$$

Now suppose our choice of ordering was incorrect and, in fact, $k < h$. We can show that Equation (8) is still correct by rearranging it with judicious applications of (6) and (7):

$$\begin{aligned}
(u+v)_j &= \xi_{j,1,h} (a + c) + \xi_{j,h,k} (b + c) + \xi_{j,k,n} (b + d) \\
&= (\xi_{j,1,k} + \xi_{j,k,h}) (a + c) - \xi_{j,k,h} (b + c) + (\xi_{j,k,h} + \xi_{j,h,n}) (b + d) \\
&= \xi_{j,1,k} (a + c) + \xi_{j,k,h} ((a + c) - (b + c) + (b + d)) + \xi_{j,h,n} (b + d) \\
&= \xi_{j,1,k} (a + c) + \xi_{j,k,h} (a + d) + \xi_{j,h,n} (b + d)
\end{aligned}$$

This final result is precisely the form that we would get if we had started with the assumption that $k < h$.

With this choice of support functions, our previous example of adding K matrices with N regions each and all boundaries parallel would give a result with $K(N - 1) + 1$ regions, corresponding to *any* arbitrary choice of relative order of the region boundaries among the arguments. Each region would have a sum of K elements, one from each matrix. So the total size of the generic element expression would be $K(K(N - 1) + 1)$. If each of the matrices had L elements, the cost to compute a specialisation of this abstract sum would be $O(LK^2N)$. This compares to a cost of $O(LK)$ if the specialisation is known in advance. If the specialisation were not known in advance, and the sum were not reorganised to represent a generic case, then the cost to compute a specialisation would be LKN . So the cost to present the generic form of the result is a factor of K . If we were to add K block matrices, each with $N \times M$ regions, using the ξ support functions would give $(K(N - 1) + 1)(K(M - 1) + 1)$ regions each with K terms.

4 Describing Regions

We need some tools to describe the spatial extent of regions and their complements. For brevity, we will henceforth refer to the characteristic function of a region or of the complement of a region simply as the region or the complement of the region.

We need the following equations to be satisfied by the complement \overline{A} of a region A :

$$(i) \quad \overline{\overline{A}} = A \qquad (ii) \quad \overline{\overline{\overline{A}}} = \overline{A}$$

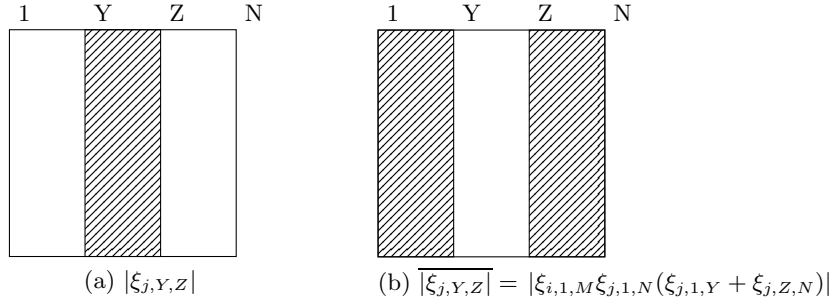


Fig. 1: ξ complement example

The complement of the region described by a single ξ expression will be the absolute value of the sum of the two ξ expressions that describe the spatial extent of the entire matrix minus the region described by the ξ (c.f. Fig. 1):

Definition 5. *The characteristic function of the complement $\overline{|\xi_{X,Y,Z}|}$ of the region described by a parallel ξ support function with respect to an abstract matrix of dimension $(M-1) \times (N-1)$ is defined to be:*

$$\overline{|\xi_{X,Y,Z}|} = \begin{cases} |\xi_{i,1,M} \xi_{j,1,N} (\xi_{i,1,Y} + \xi_{i,Z,M})| & \text{if } X \text{ has the form } i \\ |\xi_{i,1,M} \xi_{j,1,N} (\xi_{j,1,Y} + \xi_{j,Z,N})| & \text{if } X \text{ has the form } j \\ |\xi_{i,1,M} \xi_{j,1,N} (\xi_{i+j,1,Y} + \xi_{i+j,Z,M+N-1})| & \text{if } X \text{ has the form } i+j \\ |\xi_{i,1,M} \xi_{j,1,N} (\xi_{i-j,2-N,Y} + \xi_{i-j,Z,M-2})| & \text{if } X \text{ has the form } i-j \end{cases} \quad (9)$$

We can now define the complement of a region described by the product of ξ support functions as follows:

Definition 6. *Let $\Xi = \xi_1 \xi_2 \dots \xi_n$, where each ξ_i is a parallel ξ support function of the form ξ_{X_i,Y_i,Z_i} . The complement of $|\Xi|$, $|\overline{\Xi}|$, is defined to be:*

$$|\overline{\Xi}| = |\overline{\xi_1}| + |\xi_1 \overline{\xi_2}| + |\xi_1 \xi_2 \overline{\xi_3}| + \dots + |\xi_1 \dots \xi_{n-1} \overline{\xi_n}| \quad (10)$$

Further expressions for complements can be found using elementary set theory: e.g., for sets A, B , $\overline{A \cup B} = \overline{A} \cap \overline{B}$, hence $|\overline{\xi_1 + \xi_2}| = \left(\overline{|\xi_1|}\right) \left(\overline{|\xi_2|}\right)$, etc.

5 Abstract Matrix Addition

We now present a worked example considering the following abstract matrices:

$$A := \begin{bmatrix} a & \dots & a \\ \vdots & & \ddots \\ a & & \mathbf{0} \end{bmatrix} \quad B := \begin{bmatrix} b & \dots & b & c & \dots & c \\ \vdots & & \vdots & & & \vdots \\ \vdots & & \vdots & & & \vdots \\ b & \dots & b & c & \dots & c \end{bmatrix}$$

$$A = \xi_{i,1,p} \xi_{j,1,p} \xi_{i+j,2,p+1} a \quad B = \xi_{i,1,n} \xi_{j,1,q} b + \xi_{i,1,n} \xi_{j,q,n} c$$

Here both matrices are of dimension $n - 1 \times n - 1$, the a triangle ellipses are of length $p - 1$, i.e., columns and rows $1 \dots p - 1$, the b rectangle fills columns $1 \dots q - 1$ and all rows and the c rectangle fills columns $q \dots n - 1$ and all rows. All the -1 terms in these dimensions are just to make the ξ function subscripts simpler.

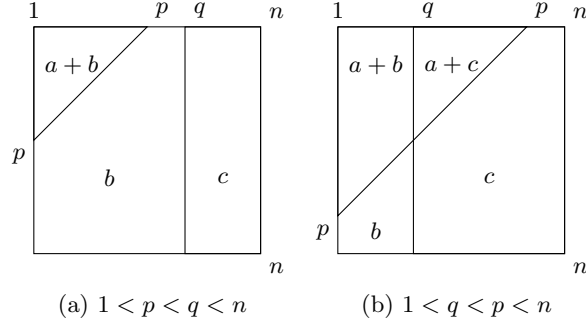


Fig. 2: Example: two of the possible orderings of $1, p, q, n$

When we evaluate $A + B$, we can get a number of different cases. For any non-trivial matrix we always have that $1 < n$. However, we can consider p to be anywhere in the range $1..2n - 1$ and q in the range $1..n$. In the extreme cases, $p = 1$, $p = 2n - 1$, $q = 1$ or $q = n$, one or more of the regions disappear. For the purposes of our example, we will restrict ourselves to two cases: $1 < p < q < n$ and $1 < q < p < n$, shown in Fig 2.

Writing out the terms for the sum of the matrices, we get

$$(A + B)_{i,j} = \xi_{i,1,p} \xi_{j,1,p} \xi_{i+j,2,p+1} a + \xi_{i,1,n} \xi_{j,1,q} b + \xi_{i,1,n} \xi_{j,q,n} c \quad (11)$$

This has three terms, but the terms describe potentially overlapping regions. To be able to extract the shape of the resulting regions, as shown in Fig. 2, we need to choose an ordering and normalise the expression with respect to that ordering. To normalise to this order, we start by identifying every potential region as the intersection of every region from A with every region from B .

Note that A has a 0 region, which is not explicitly represented in the expression for A . To get the characteristic function for this region, we need to find the characteristic function of the complement of the a region. Since the a region is

$|\xi_{i,1,p} \xi_{j,1,p} \xi_{i+j,2,p+1}|$, we can calculate its complement as

$$\begin{aligned}
|\overline{\xi_{i,1,p} \xi_{j,1,p} \xi_{i+j,2,p+1}}| &= |\overline{\xi_{i,1,p}}| + |\xi_{i,1,p} \overline{|\xi_{j,1,p}|}| + |\xi_{i,1,p} \xi_{j,1,p} \overline{|\xi_{i+j,2,p+1}|}| \\
&= |\xi_{i,1,n} \xi_{j,1,n} (\xi_{i,1,1} + \xi_{i,p,n})| + |\xi_{i,1,p} \overline{|\xi_{j,1,p}|}| + |\xi_{i,1,p} \xi_{j,1,p} \overline{|\xi_{i+j,2,p+1}|}| \\
&= |\xi_{i,p,n} \xi_{j,1,n}| + |\xi_{i,1,p} \overline{|\xi_{j,1,p}|}| + |\xi_{i,1,p} \xi_{j,1,p} \overline{|\xi_{i+j,2,p+1}|}| \\
&= |\xi_{i,p,n} \xi_{j,1,n}| + |\xi_{i,1,p} \xi_{i,1,n} \xi_{j,1,n} (\xi_{j,1,1} + \xi_{j,p,n})| + |\xi_{i,1,p} \xi_{j,1,p} \overline{|\xi_{i+j,2,p+1}|}| \\
&= |\xi_{i,p,n} \xi_{j,1,n}| + |\xi_{i,1,p} \xi_{j,p,n}| + |\xi_{i,1,p} \xi_{j,1,p} \overline{|\xi_{i+j,2,p+1}|}| \\
&= |\xi_{i,p,n} \xi_{j,1,n}| + |\xi_{i,1,p} \xi_{j,p,n}| + |\xi_{i,1,p} \xi_{j,1,p} \xi_{i,1,n} \xi_{j,1,n} (\xi_{i+j,2,2} + \xi_{i+j,p+1,2n-1})| \\
&= |\xi_{i,p,n} \xi_{j,1,n}| + |\xi_{i,1,p} \xi_{j,p,n}| + |\xi_{i,1,p} \xi_{j,1,p} \xi_{i+j,p+1,2n-1}| \tag{12} \\
&= |\xi_{i,1,n} \xi_{j,1,n} \xi_{i+j,p+1,2n-1}| \tag{13}
\end{aligned}$$

The step from (12) to (13) is valid, but obscure. Consider the middle summand: $\xi_{i,1,p} \xi_{j,p,n}$. We can multiply this by $\xi_{i+j,p+1,2n-1}$ without changing its value for any i, j in the range of the matrix, as it already implies that $1 + p \leq i + j < \min(p + n, 2n - 1) \leq 2n - 1$. If we do that multiplication, we can then combine it with the final summand to replace both with $\xi_{i,1,p} \xi_{j,1,n} \xi_{i+j,p+1,2n-1}$. We can similarly fold in the first summand to get the required result.

Note that (12) shows that the complement of the a is constructed it as a sum (union) of three disjoint regions (c.f. Fig. 3). The step to (13) is simply reconstructing the single region from the three partial ones.

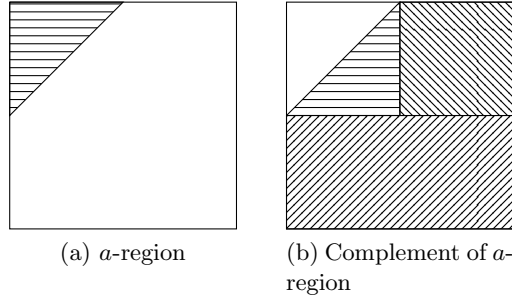


Fig. 3: a region and its complement

Now that we can produce an expression for every region, including 0 regions, in the component matrices, we can choose an order and normalise to it. We choose $1 < q < p < n$. We can construct the disjoint region descriptions of the result by intersecting each of the two regions of A with each of the two regions of B and simplifying under the assumption that $1 < q < p < n$:

$$\begin{aligned}
a \cap b \text{ region: } &|\xi_{i,1,p} \xi_{j,1,p} \xi_{i+j,2,p+1} \xi_{i,1,n} \xi_{j,1,q}| = |\xi_{i,1,p} \xi_{j,1,q} \xi_{i+j,2,p+1}| \\
a \cap c \text{ region: } &|\xi_{i,1,p} \xi_{j,1,p} \xi_{i+j,2,p+1} \xi_{i,1,n} \xi_{j,q,n}| = |\xi_{i,1,p} \xi_{j,q,p} \xi_{i+j,q+1,p+1}|
\end{aligned}$$

$$\begin{aligned} \bar{a} \cap b \text{ region: } & |\xi_{i,1,n} \xi_{j,1,n} \xi_{i+j,p+1,2n-1} \xi_{i,1,n} \xi_{j,1,q}| = |\xi_{i,1,n} \xi_{j,1,q} \xi_{i+j,p+1,n+q}| \\ \bar{a} \cap c \text{ region: } & |\xi_{i,1,n} \xi_{j,1,n} \xi_{i+j,p+1,2n-1} \xi_{i,1,n} \xi_{j,q,n}| = |\xi_{i,1,n} \xi_{j,q,n} \xi_{i+j,p+1,2n-1}| \end{aligned}$$

Thus we have now identified that in the case that $1 < q < p < n$ there are four disjoint regions. We still have to obtain the full term for each region, but that is easily done by multiplying each region expression by the full original abstract expression to get the full projection of the abstract matrix expression onto each disjoint region. We show the detailed derivation for the $a \cap b$ region:

$$\begin{aligned} & |\xi_{i,1,p} \xi_{j,1,q} \xi_{i+j,2,p+1}| (\xi_{i,1,p} \xi_{j,1,p} \xi_{i+j,2,p+1} a + \xi_{i,1,n} \xi_{j,1,q} b + \xi_{i,1,n} \xi_{j,q,n} c) \\ & = \xi_{i,1,p} \xi_{j,1,q} \xi_{i+j,2,p+1} a + \xi_{i,1,p} \xi_{j,1,q} \xi_{i+j,2,p+1} b + 0 \end{aligned} \quad (14)$$

$$= \xi_{i,1,p} \xi_{j,1,q} \xi_{i+j,2,p+1} (a + b) \quad (15)$$

The final result is then obtained by adding the terms obtained.

$$\begin{aligned} & \xi_{i,1,p} \xi_{j,1,q} \xi_{i+j,2,p+1} (a + b) + \\ & \xi_{i,1,p-q} \xi_{j,q,p} \xi_{i+j,q+1,p+1} (a + c) + \\ & \xi_{i,1,n} \xi_{j,1,q} \xi_{i+j,p+1,n+q-1} b + \\ & \xi_{i,1,n} \xi_{j,q,n} \xi_{i+j,p+1,2n-1} c \end{aligned} \quad (16)$$

Equation (16) clearly corresponds to Fig. (2b). As a final check, we reconsider (16) in the alternative case of $1 < p < q < n$. We try reversing the order of p and q and depend upon the reversing properties of the ξ support functions to retrieve the situation.

$$\begin{aligned} \text{Eq. (16)} & = \xi_{i,1,p} \xi_{j,1,q} \xi_{i+j,2,p+1} (a + b) + \\ & \xi_{i,1,n} \xi_{j,1,q} \xi_{i+j,p+1,n+q-1} b + \xi_{i,1,n} \xi_{j,q,n} \xi_{i+j,p+1,2n-1} c \end{aligned} \quad (17)$$

$$\begin{aligned} & = \xi_{i,1,p} (\xi_{j,1,p} + \xi_{j,p,q}) \xi_{i+j,2,p+1} (a + b) + \\ & \xi_{i,1,n} \xi_{j,1,q} \xi_{i+j,p+1,n+q-1} b + \xi_{i,1,n} \xi_{j,q,n} \xi_{i+j,p+1,2n-1} c \end{aligned} \quad (18)$$

$$\begin{aligned} & = \xi_{i,1,p} \xi_{j,1,p} \xi_{i+j,2,p+1} (a + b) + \xi_{i,1,p} \xi_{j,p,q} \xi_{i+j,2,p+1} (a + b) + \\ & \xi_{i,1,n} \xi_{j,1,q} \xi_{i+j,p+1,n+q-1} b + \xi_{i,1,n} \xi_{j,q,n} \xi_{i+j,p+1,2n-1} c \end{aligned} \quad (19)$$

$$\begin{aligned} & = \xi_{i,1,p} \xi_{j,1,p} \xi_{i+j,2,p+1} (a + b) + \\ & \xi_{i,1,n} \xi_{j,1,q} \xi_{i+j,p+1,n+q-1} b + \xi_{i,1,n} \xi_{j,q,n} \xi_{i+j,p+1,2n-1} c \end{aligned} \quad (20)$$

$$\begin{aligned} & = \xi_{i,1,p} \xi_{j,1,p} \xi_{i+j,2,p+1} (a + b) + \\ & \xi_{i,1,n} \xi_{j,1,q} \xi_{i+j,p+1,n+q-1} b + \xi_{i,1,n} \xi_{j,q,n} c \end{aligned} \quad (21)$$

The steps involved in this derivation can be explained as follows:

(16) \longrightarrow (17): The coefficient of $(a + c)$ is 0 within the matrix if $p < q$ because $p - q$ is negative and $\xi_{i,1,p-q}$ is only non-zero outside the matrix bounds.

(17) \longrightarrow (18): use of (7)

(18) \longrightarrow (19): distributing sum

- (19) \longrightarrow (20): The coefficient of the second $(a + b)$ term is 0 because $\xi_{i,1,p} \xi_{j,p,q}$ and $p < q$ implies that $i + j$ is in the range $p + 1 \dots p + q$, but $\xi_{i+j,2,p+1}$ is always zero in this range.
- (20) \longrightarrow (21): $\xi_{i,1,n} \xi_{j,q,n}$ already limits $i + j$ to the range $q + 1 \dots 2n - 1$, and $p < q$, so the $\xi_{i+j,p+1,2n-1}$ is redundant and can be dropped.

Equation (21) clearly corresponds to Fig. (2a) and so we have been able to represent the matrix using one term for each region of only one case, but still capture, with this expression, the information required for the alternative case.

However, it is important to note that this transformation can not be achieved in all possible cases, only certain representations contain implicitly all possible shapes of the abstract matrix. For example, if we had chosen the ordering $1 < p < q < n$ initially we could have not reached the matrix shape for the ordering $1 < q < p < n$. The desired implicit representation needs to have a certain maximality property with respect to the reachability of all cases for the abstract matrix. We will define this notion more precisely in the next section.

6 Matrix Addition Algorithm

The primary advantage of our choice of ξ support functions is that they allow the entire set of possible relative orderings to be represented immediately by one generic order choice for certain important classes of matrices. If all boundaries are vertical and horizontal or all boundaries are diagonal or all boundaries are anti-diagonal, then the form of the generic sum can be written down by picking an arbitrary relative order of the boundaries of each kind and directly writing the resulting generic term. In particular, this situation includes the classes of block matrices and banded matrices as special cases. The class of block matrices is closed under addition (adding block matrices gives a block matrix, possibly with smaller blocks) as is the class of banded matrices (with all boundaries along diagonals with $i - j = \text{constant}$). It is not necessary in these cases to attempt to merge regions or use region complements. We need the following definitions before giving the algorithm for this matrix addition:

Definition 7. *A generic placement of a boundary in an abstract matrix is one that intersects all boundary lines to which it is not parallel in the abstract matrix and does not cause an intersection of three boundaries at one point.*

Definition 8. *An abstract matrix is in a generic configuration if all its boundary lines have generic placement.*

Algorithm 1 (Special Matrix Addition)

INPUT:

Two abstract matrices A and B for which one of the following holds

- 1. both are abstract block matrices*
- 2. both are abstract banded matrices with diagonal region boundaries*
- 3. both are abstract banded matrices with anti-diagonal region boundaries.*

OUTPUT:

An abstract matrix of the same form as the two inputs, with an expression for a generic entry.

METHOD:

1. Form the sets $\mathbf{v}(A)$, $\mathbf{h}(A)$, $\mathbf{d}(A)$, $\mathbf{a}(A)$ containing the expressions for $j = \text{expr}$, $i = \text{expr}$, $i - j = \text{expr}$ $i + j = \text{expr}$ defining the region boundaries in A . Likewise for B . If $\mathbf{d}(A)$ or $\mathbf{d}(B)$ are non empty, then the other sets must be empty. If $\mathbf{a}(A)$ or $\mathbf{a}(B)$ are non empty, then the other sets must be empty. If any of $\mathbf{v}(A)$, $\mathbf{v}(B)$, $\mathbf{h}(A)$, $\mathbf{h}(B)$ are non empty then the diagonal and anti-diagonal sets will be empty.
2. Impose an arbitrary order on each of these sets (if one has not been imposed already).
3. Form the sets $\mathbf{v}(C) = \mathbf{v}(A) \cup \mathbf{v}(B)$, $\mathbf{h}(C) = \mathbf{h}(A) \cup \mathbf{h}(B)$, $\mathbf{d}(C) = \mathbf{d}(A) \cup \mathbf{d}(B)$, and $\mathbf{a}(C) = \mathbf{a}(A) \cup \mathbf{a}(B)$
 Impose an arbitrary order on the sets $\mathbf{v}(C)$, $\mathbf{h}(C)$, $\mathbf{d}(C)$ and $\mathbf{a}(C)$ that
 (a) is consistent with the orders on $\mathbf{v}(X)$, $\mathbf{h}(X)$, $\mathbf{d}(X)$ and $\mathbf{a}(X)$ for $X = A$ and $X = B$
 (b) places the resulting abstract matrix in generic configuration.
 Let $\mathbf{v}(C)_k$, $\mathbf{h}(C)_k$, $\mathbf{d}(C)_k$ and $\mathbf{a}(C)_k$ denote the k -th element of the corresponding set in the imposed order.
 Let $\mathbf{v}(C)_0 = 1$, $\mathbf{v}(C)_{\#\mathbf{v}(C)+1} = \#\text{cols} + 1$
 Let $\mathbf{h}(C)_0 = 1$, $\mathbf{h}(C)_{\#\mathbf{h}(C)+1} = \#\text{rows} + 1$
 Let $\mathbf{d}(C)_0 = 1 - \#\text{cols}$, $\mathbf{d}(C)_{\#\mathbf{a}(C)+1} = \#\text{rows}$
 Let $\mathbf{a}(C)_0 = 2$, $\mathbf{a}(C)_{\#\mathbf{a}(C)+1} = \#\text{rows} + \#\text{cols} + 1$
4. Form the regions, depending on input cases 1 to 3 respectively,
 - 1: $\xi_{i, \mathbf{h}(C)_h, \mathbf{h}(C)_{h+1}} \xi_{j, \mathbf{v}(C)_k, \mathbf{v}(C)_{k+1}}$ for $h = 0.. \#\mathbf{h}(C)$, $k = 0.. \#\mathbf{v}(C)$,
 - 2: $\xi_{i-j, \mathbf{d}(C)_h, \mathbf{d}(C)_{h+1}}$ for $h = 0.. \#\mathbf{d}(C)$,
 - 3: $\xi_{i-j, \mathbf{a}(C)_h, \mathbf{a}(C)_{h+1}}$ for $h = 0.. \#\mathbf{a}(C)$
5. Construct the expression. Multiply the ξ term for each region by the corresponding sum of a term from A and a term from B .

Theorem 1. *Algorithm 1 gives an expression that correctly evaluates all elements of $A+B$ under any permitted reordering of the boundaries.*

We omit the proof for theorem 1 as it is a lengthy and tedious case analysis. Instead we now consider the general case.

In order to generalise algorithm 1, we observe that to represent the maximal number of possible regions in a matrix, we have to choose boundary placements such that at most two boundary lines intersect at a single point. However, the generic configuration of an abstract matrix then possibly has intersection points of two boundary lines outside the boundaries of the matrix. But it is easy to see that we can always embed a given abstract matrix as a rectangular submatrix into a larger abstract matrix that contains the intersection points, such that at most two boundary lines intersect, as summarised in the following two results:

Lemma 1. *Every abstract matrix may be written as a submatrix of an abstract matrix in generic configuration, possibly after a shift of indices.*

Theorem 2. *Given two abstract matrices A, B in generic configuration. Then let $(\mathbf{v}(A), \mathbf{h}(A), \mathbf{d}(A), \mathbf{a}(A))$ and $(\mathbf{v}(B), \mathbf{h}(B), \mathbf{d}(B), \mathbf{a}(B))$ be the corresponding vertical, horizontal, diagonal, and anti-diagonal boundary sets as defined in algorithm 1. It is then possible to impose orders on the unions of the boundary sets $(\mathbf{v}(A) \cup \mathbf{v}(B), \mathbf{h}(A) \cup \mathbf{h}(B), \mathbf{d}(A) \cup \mathbf{d}(B), \mathbf{a}(A) \cup \mathbf{a}(B))$ such that the resulting boundaries are in generic placement, possibly in a submatrix of a larger matrix.*

Again we omit the proofs and now present a method for the addition of general abstract matrices based on the results. Although we have not discovered any counterexamples, we have not as yet proven its correctness in all cases.

Algorithm 2 (General Matrix Addition)

INPUT:

Two abstract matrices A and B in generic configuration.

OUTPUT:

An abstract matrix in generic configuration and an expression for the generic entry.

METHOD:

1. *Form the sets $\mathbf{v}(A), \mathbf{h}(A), \mathbf{d}(A)$, and $\mathbf{a}(A)$ containing the expressions for $j = \text{expr}, i = \text{expr}, i - j = \text{expr}$ $i + j = \text{expr}$ defining the region boundaries in A . Likewise for B .*

2. *Impose an arbitrary order on each of these sets (if one has not yet been imposed).*

3. *Form the sets $\mathbf{v}(C) = \mathbf{v}(A) \cup \mathbf{v}(B), \mathbf{h}(C) = \mathbf{h}(A) \cup \mathbf{h}(B), \mathbf{d}(C) = \mathbf{d}(A) \cup \mathbf{d}(B)$, and $\mathbf{a}(C) = \mathbf{a}(A) \cup \mathbf{a}(B)$*

Impose an arbitrary order on the sets $\mathbf{v}(C), \mathbf{h}(C), \mathbf{d}(C)$ and $\mathbf{a}(C)$ that

- (a) *is consistent with the orders on $\mathbf{v}(X), \mathbf{h}(X), \mathbf{d}(X)$ and $\mathbf{a}(X)$ for $X = A$ and $X = B$*

(b) places the resulting abstract matrix in generic configuration.

It may be necessary to embed in a larger matrix to do this.

Let $\mathbf{v}(C)_k, \mathbf{h}(C)_k, \mathbf{d}(C)_k$ and $\mathbf{a}(C)_k$ denote the k -th element of the corresponding set in the imposed order.

Let $\mathbf{v}(C)_0 = 1, \mathbf{v}(C)_{\#\mathbf{v}(C)+1} = \#\text{cols} + 1$

Let $\mathbf{h}(C)_0 = 1, \mathbf{h}(C)_{\#\mathbf{h}(C)+1} = \#\text{rows} + 1$

Let $\mathbf{d}(C)_0 = 1 - \#\text{cols}, \mathbf{d}(C)_{\#\mathbf{d}(C)+1} = \#\text{rows}$

Let $\mathbf{a}(C)_0 = 2, \mathbf{a}(C)_{\#\mathbf{a}(C)+1} = \#\text{rows} + \#\text{cols} + 1$

4. *Form the regions*

$\xi_{i, \mathbf{h}(C)_h, \mathbf{h}(C)_{h+1}} \xi_{j, \mathbf{v}(C)_k, \mathbf{v}(C)_{k+1}} \xi_{i-j, \mathbf{d}(C)_l, \mathbf{d}(C)_{l+1}} \xi_{i-j, \mathbf{a}(C)_m, \mathbf{a}(C)_{m+1}}$

for $h = 0..\#\mathbf{h}(C), k = 0..\#\mathbf{v}(C), l = 0..\#\mathbf{d}(C), m = 0..\#\mathbf{a}(C)$

Note: if any of the sets are empty, the corresponding ξ factors are 1.

5. *Construct the expression.*

Multiply the ξ term for each region by the corresponding sum of a term from A and a term from B .

7 Block Matrix Example

As a further example of our matrix addition algorithm we consider the problem of adding two 2×2 block matrices $A_{n-1 \times m-1} + B_{n-1 \times m-1}$:

$$\begin{bmatrix} a & \dots & a & b & \dots & b \\ \vdots & & \vdots & \vdots & & \vdots \\ a & \dots & a & b & \dots & b \\ c & \dots & c & d & \dots & d \\ \vdots & & \vdots & \vdots & & \vdots \\ c & \dots & c & d & \dots & d \end{bmatrix} + \begin{bmatrix} a' & \dots & a' & b' & \dots & b' \\ \vdots & & \vdots & \vdots & & \vdots \\ a' & \dots & a' & b' & \dots & b' \\ c' & \dots & c' & d' & \dots & d' \\ \vdots & & \vdots & \vdots & & \vdots \\ c' & \dots & c' & d' & \dots & d' \end{bmatrix} = \begin{bmatrix} a_{k \times l} & b_{k \times m-l} \\ c_{n-k \times l} & d_{n-k \times m-l} \end{bmatrix} + \begin{bmatrix} a'_{k' \times l'} & b'_{k' \times m-l'} \\ c'_{n-k' \times l'} & d'_{n-k' \times m-l'} \end{bmatrix}$$

We get 4 different shapes for the combination of the blocks in the sum matrix:

$$(a) \begin{bmatrix} \vdots & \vdots & \vdots \\ - - & \vdots & - - \\ \vdots & \vdots & \vdots \end{bmatrix} \quad (b) \begin{bmatrix} \vdots & \vdots & \vdots \\ - - & \vdots & - - \\ - - & \vdots & - - \\ \vdots & \vdots & \vdots \end{bmatrix} \quad (c) \begin{bmatrix} \vdots & \vdots & \vdots \\ - - & \vdots & - - \\ \vdots & \vdots & \vdots \end{bmatrix} \quad (d) \begin{bmatrix} \vdots & \vdots & \vdots \\ - - & \vdots & - - \\ - - & \vdots & - - \\ \vdots & \vdots & \vdots \end{bmatrix}$$

which leads to nine different cases, depending on the order of the index variables k, l, k', l' :

- (a): $k = k', l = l'$
- (b): $k < k', l = l'$ or $k > k', l = l'$
- (c): $k = k', l < l'$ or $k = k', l > l'$
- (d): $k < k', l < l'$ or $k > k', l < l'$ or $k < k', l > l'$ or $k > k', l > l'$

Observe that we assume that the index variables are strictly within the boundaries, i.e., none of the blocks in the original matrices vanish.

After constructing all possible disjoint regions we get the following sum of region terms:

$$\begin{aligned} & \xi_{i,1,k} \xi_{j,1,l} \xi_{i,1,k'} \xi_{j,1,l'} (a + a') + \xi_{i,1,k} \xi_{j,1,l} \xi_{i,1,k'} \xi_{j,l',m} (a + b') \\ & + \xi_{i,1,k} \xi_{j,1,l} \xi_{i,k',n} \xi_{j,1,l'} (a + c') + \xi_{i,1,k} \xi_{j,1,l} \xi_{i,k',n} \xi_{j,l',m} (a + d') \\ & + \xi_{i,1,k} \xi_{j,l,m} \xi_{i,1,k'} \xi_{j,1,l'} (b + a') + \xi_{i,1,k} \xi_{j,l,m} \xi_{i,1,k'} \xi_{j,l',m} (b + b') \\ & + \xi_{i,1,k} \xi_{j,l,m} \xi_{i,k',n} \xi_{j,1,l'} (b + c') + \xi_{i,1,k} \xi_{j,l,m} \xi_{i,k',n} \xi_{j,l',m} (b + d') \\ & + \xi_{i,k,n} \xi_{j,1,l} \xi_{i,1,k'} \xi_{j,1,l'} (c + a') + \xi_{i,k,n} \xi_{j,1,l} \xi_{i,1,k'} \xi_{j,l',m} (c + b') \\ & + \xi_{i,k,n} \xi_{j,1,l} \xi_{i,k',n} \xi_{j,1,l'} (c + c') + \xi_{i,k,n} \xi_{j,1,l} \xi_{i,k',n} \xi_{j,l',m} (c + d') \\ & + \xi_{i,k,n} \xi_{j,l,m} \xi_{i,1,k'} \xi_{j,1,l'} (d + a') + \xi_{i,k,n} \xi_{j,l,m} \xi_{i,1,k'} \xi_{j,l',m} (d + b') \\ & + \xi_{i,k,n} \xi_{j,l,m} \xi_{i,k',n} \xi_{j,1,l'} (d + c') + \xi_{i,k,n} \xi_{j,l,m} \xi_{i,k',n} \xi_{j,l',m} (d + d') \end{aligned}$$

Our algorithm now selects one maximal expression, which is one of the 3×3 matrices under case (d), e.g., the first case with order ($k < k', l < l'$). This corresponds to the left matrix below. Observe, that the choice of variable orderings is independent for the vertical and horizontal parameters in this particular case.

$$1 < k < k' < n; 1 < l < l' < m$$

$$\begin{bmatrix} a + a' & b + b' \\ c + c' & d + d' \\ c + c' & d + d' \end{bmatrix}$$

$$1 < k' < k < n; 1 < l' < l < m$$

$$\begin{bmatrix} a + a' & b + b' \\ a + a' & d + d' \\ c + c' & c + c' \end{bmatrix}$$

We now observe what happens if we rewrite the expression for the left matrix above into the right matrix above by changing the variable ordering. Here the right hand side to the last case in (d), respectively. Firstly, after fixing the order for the chosen maximal representation we get the following reduced sum of region expressions.

$$\begin{aligned}
& \xi_{i,1,k}\xi_{j,1,l}(a+a') + \xi_{i,1,k}\xi_{j,l,l'}(b+a') + \xi_{i,1,k}\xi_{j,l',m}(b+b') \\
& + \xi_{i,k,k'}\xi_{j,1,l}(c+a') + \xi_{i,k',n}\xi_{j,1,l}(c+c') + \xi_{i,k,k'}\xi_{j,l,l'}(d+a') \\
& + \xi_{i,k,k'}\xi_{j,l',m}(d+b') + \xi_{i,k',n}\xi_{j,l,l'}(d+c') + \xi_{i,k',n}\xi_{j,l',m}(d+d')
\end{aligned}$$

After changing the ordering of variables the ξ terms are rewritten using the algorithm of the previous section, which will introduce sums of ξ terms as well as negative regions:

$$\begin{aligned}
& (\xi_{i,1,k'}\xi_{j,1,l'} + \xi_{i,1,k'}\xi_{j,l',l} + \xi_{i,k',k}\xi_{j,1,l'} + \xi_{i,k',k}\xi_{j,l',l})(a+a') \\
& + (-\xi_{i,1,k'}\xi_{j,l',l} - \xi_{i,k',k}\xi_{j,l',l})(b+a') \\
& + (\xi_{i,1,k'}\xi_{j,l',l} + \xi_{i,1,k'}\xi_{j,l,m} + \xi_{i,k',k}\xi_{j,l',l} + \xi_{i,k',k}\xi_{j,l,m})(b+b') \\
& + (-\xi_{i,k',k}\xi_{j,1,l'} - \xi_{i,k',k}\xi_{j,l',l})(c+a') \\
& + (\xi_{i,k',k}\xi_{j,1,l'} + \xi_{i,k,n}\xi_{j,1,l'} + \xi_{i,k',k}\xi_{j,l',l} + \xi_{i,k,n}\xi_{j,l',l})(c+c') \\
& + (\xi_{i,k',k}\xi_{j,l',l})(d+a') \\
& + (-\xi_{i,k',k}\xi_{j,l',l} - \xi_{i,k',k}\xi_{j,l,m})(d+b') \\
& + (-\xi_{i,k',k}\xi_{j,l',l} - \xi_{i,k,n}\xi_{j,l',l})(d+c') \\
& + (\xi_{i,k,n}\xi_{j,l',l} + \xi_{i,k',k}\xi_{j,l,m} + \xi_{i,k',k}\xi_{j,l',l} + \xi_{i,k,n}\xi_{j,l,m})(d+d')
\end{aligned}$$

Finally subsequent simplification of the above sum yields a new expression that corresponds indeed to the desired result matrix. Moreover, the result is again a maximal representation and thus would allow for further transformation into any one of the 8 other cases.

$$\begin{aligned}
& \xi_{i,1,k'}\xi_{j,1,l'}(a+a') + \xi_{i,1,k'}\xi_{j,l',l}(a+b') + \xi_{i,k',k}\xi_{j,1,l'}(a+c') \\
& + \xi_{i,k',k}\xi_{j,l',l}(a+d') + \xi_{i,1,k'}\xi_{j,l,m}(b+b') + \xi_{i,k',k}\xi_{j,l,m}(b+d') \\
& + \xi_{i,k,n}\xi_{j,1,l'}(c+c') + \xi_{i,k,n}\xi_{j,l',l}(c+d') + \xi_{i,k,n}\xi_{j,l,m}(d+d')
\end{aligned}$$

8 Conclusion

We have presented an approach to abstract matrix addition, that enables reasoning on arithmetic closure properties for classes of structured matrices. The approach addresses the shortcomings of our previous approach as presented in [7] that uses half-plane constraints as support functions and that can suffer combinatorial problems in that all regions of all resulting matrix cases have to be represented explicitly. The new approach instead works with a more compact representation that implicitly contains all possible cases and therefore scales up better during a sequence of computations.

The work can use our previously developed parsing procedure for abstract matrices that determines their meaning and represents them in terms of the homogeneous regions they contain, thereby making them available as templates for concrete matrices [4]. Our work is related to previous work by Fateman in Macsyma [2], 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 [9, 10], which presented algorithms for GCD, factorisation and functional decomposition of polynomials with terms of symbolic degree and to work by Knauers and Schneider [3] on indefinite symbolic summation using unspecified sequences of summands.

While in [7] we have presented a full system for abstract matrix arithmetic, i.e., addition, multiplication, and their combination, our approach presented in this paper so far only extends to addition. Since the primary property of ξ relies on cancellation of region elements when orders of parameters are reversed, a comparable approach for multiplication would need to involve division by the region elements. A first approach of lifting ξ expressions to abstract matrix multiplication has been presented in [6], which had the drawback that abstract matrix representations for addition and multiplication were incompatible. This is a problem to be addressed further in the future.

References

1. J. Carette. A canonical form for some piecewise defined functions. In *Proc. ISSAC 2007*, p. 77–84. ACM Press, 2007.
2. R. Fateman. Manipulation of matrices symbolically. Available from <http://http.cs.berkeley.edu/~fateman/papers/symmat2.pdf>, 2003.
3. M. Kauers and C. Schneider. Application of unspecified sequences in symbolic summation. In *Proceedings of ISSAC 2006*, p. 177–183, 2006.
4. A. P. Sexton and V. Sorge. Abstract matrices in symbolic computation. In *Proceedings of ISSAC 2006*, p. 318–325. ACM Press, 2006.
5. A. P. Sexton, V. Sorge, and S. M. Watt. Abstract matrix arithmetic. Submitted to International Symposium on Symbolic and Algebraic Computation (ISSAC), 2008.
6. A. P. Sexton, V. Sorge, and S. M. Watt. Abstract matrix arithmetic. In *Proceedings of SYNASC-2008*. IEEE Computer Society Press, 2009.
7. A. P. Sexton, V. Sorge, and S. M. Watt. Computing with abstract matrix structures. Submitted to ISSAC 2009., 2009.
8. S. M. Watt. Making computer algebra more symbolic. In *Transgressive Computing*, p. 43–49, 2006.
9. S. M. Watt. Two families of algorithms for symbolic polynomials. In *Computer Algebra 2006: Advances in Symb. Algorithms*, p. 193–210. World Scientific, 2006.
10. S. M. Watt. Functional decomposition of symbolic polynomials. In *Proc. of ICCSA*, p. 353–362. IEEE Computer Society, 2008.