Computing with Abstract Matrix Structures

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

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

ABSTRACT

Classes of matrices are often presented with symbolic dimensions using a mixture of terms and ellipsis symbols to describe their internal structure. While working with such classes of matrices is everyday mathematical practice, it has little automated support. We describe an algebraic encoding of such matrices in terms of support functions and define the corresponding addition and multiplication algorithms. It is, however, non-trivial to retrieve the structural description of the matrix resulting from these operations. We therefore define an *abstract matrix* as an encoding of support function combinations that enables simple recovery of the structural properties. This allows us to define arithmetic algorithms for abstract matrices as extensions of those for support function combinations using a normalising term rewrite system.

Categories and Subject Descriptors

I.1.2 [**Symbolic and Algebraic Manipulation**]: Algorithms—*Algebraic algorithms*; I.1.1 [**Symbolic and Algebraic Manipulation**]: Expressions and Their Representation

General Terms

Algorithms

Keywords

Abstract Matrix Arithmetic, Symbolic Computation

1. INTRODUCTION

Matrices are often treated in an abstract way with symbolic dimensions and containing underspecified parts described by the use of ellipsis symbols. Computing and reasoning with these abstract or symbolic matrices is mathematically routine. For example, the following expression represents an infinite class of matrices that exhibit a particular structure:

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

Informally, we view an abstract matrix as a collection of disjoint regions. Each region is a closed polygonal area that can be uniformly evaluated by a single, unconditional term. Non-zero regions must be convex, therefore concave regions must be decomposed into sets of convex regions. For example, Matrix B contains four regions: two zero regions, one triangular region with terms of the form a_{ij}, and one triangular region containing b_{ij}, where i, j are the index variables of the matrix. It is precisely this structure of shaped regions that captures the structural properties of abstract matrices and lets us reason about them, for example showing that the product of upper triangular matrices are upper triangular.

We started to address the problems in this area by developing a conversion procedure for abstract matrices that accepts input in an intuitive format similar to (1), determines their meaning and represents them in terms of the regions they contain, thereby making them available as templates for concrete matrices [6]. Subsequently we have developed a purely algebraic representation of abstract matrices that represents the region structure as a linear combination of simple support functions, leading to a natural way of defining addition and multiplication on abstract matrices [7, 8].

The main drawback of a straightforward abstract matrix arithmetic using support functions is that it obscures the structural properties of the results. That is, while the result can be used to obtain the correct terms for the elements of any concrete matrix in the defined class, the number, shape, size and composition of the regions of the result matrix cannot be easily deduced. Our aim is not merely to be able to compute the symbolic term corresponding to each cell in the result of adding or multiplying abstract matrices, but also to solve the deeper problem of recovering the shape of the regions in the result. In this paper, we present an encoding and algorithms for addition and multiplication that use a term rewrite system for normalisation. The normalised result of a computation directly encodes the shape of the result regions and, for example, shows the upper triangular

*The author's work was supported in parts by RISC Transnational Access Programme of the EC FP6 project Symbolic Computation Infrastructure for Europe (SCIEnce, contract No. 026133).

ISSAC'09, July 28–31, 2009, Seoul, Republic of Korea.

nature of the product of general upper triangular matrices.

Our work is related to unpublished work of Fateman using Macsyma [3], in which indefinite matrices can be subjected to basic algebraic manipulations. While these matrices can be indefinite in size, their elements are fixed to one particular functional expression without systematic treatment of internal structure. In contrast, representing and manipulating the region structure of abstract matrices is one of our primary goals. 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, to work by Knauers and Schneider [4] on indefinite symbolic summation using unspecified sequences of summands, as well as to infinite dimensional exponentiation of matrices using power series approximations in Mathematica [11].

The remainder of the paper is organised as follows: Section 2 provides the formal definitions for the concepts of abstract matrices and a particular support function to represent regions within them conveniently. Section 3 shows how these may be used for naïve abstract matrix arithmetic. Section 4 gives a term grammar for abstract matrices and defines a useful regular form for abstract matrix expressions. Then Sections 5 and 6 give algorithms using this form for matrix addition and multiplication, respectively. Section 7 discusses the rewrite system required for normalisation. Then the paper concludes.

2. REPRESENTATION

An abstract matrix may be viewed as a class of matrices obtained by specialising free parameters with different values. In print, these matrices are typically represented as an array of expressions with entries written as functions of the index variables and with ellipses in place of omitted parts of variable size. This form is convenient neither for mathematical formalisation nor for implementation of algorithms.

We shall represent a matrix with symbolic structure as a general term (in the sense of a term algebra) with free variables for the row and column indices, and possibly other free variables representing parameters. This term will involve certain functions that support subterms or cause them to vanish, depending on the values of the indices. The use of these "support" functions leads to an algebraic treatment of regions in the abstract matrix and avoids having to treat an exponential number of sub-cases.

DEFINITION 1. *A* ring of terms*, $T = (K, V, F)$, over a constant set K, variable set V and function set F is a set of terms including binary addition and multiplication symbols ("+","×"), a unary subtraction symbol ("−") and nullary one and zero symbols ("1","0") modulo the equational theory of rings.*

We deliberately leave the nature of the ring arithmetic flexible to admit a wide variety of desirable models.

DEFINITION 2. *Given a ring of terms $T = (K, V, F)$, an* abstract matrix *over T with* index variables *i and j is a term $\tau \in T' = (K, V', F')$ where $i, j \notin V$, $V' = V \cup \{i, j\}$, $F' \supseteq F$. The term τ is called the* general element *of the abstract matrix.*

An abstract matrix can be *instantiated*, for particular m and n, to a matrix of size $m \times n$ over T by evaluating the general term with integer values of its indices from $\{1, \ldots, m\} \times \{1, \ldots, n\}$. The reason $F \neq F'$ is to admit "support functions", which we now define.

DEFINITION 3. *A* support function *is a function from $\mathbf{Z}^2$ to $\{0, 1\}$. By extension, a support function in the ring of terms of an abstract matrix with indices i, j is a term that evaluates to 0 or 1 for all evaluations of its free variables.*

A support function may be used multiplicatively in subterms of the general element of an abstract matrix to include or exclude that subterm at certain values of the matrix indices.

DEFINITION 4. *If all the other variables are evaluated, the set of values for the index variables that give a support function the value of 1 determines a subset of the elements of a matrix. We call this a* region of support*, or* region *for short. If all the other variables do not have values, then we call the term an* abstract region*.*

In specific instantiations, the most common shapes for regions are blocks, triangles, bands and parallelograms. As we shall show, it is straightforward to form support functions for these cases. General regions can be formed by adding support functions for regions with these special shapes.

In general, we allow for three types of regions in an abstract matrix: single terms, single lines or ellipses (horizontal, vertical, diagonal or anti-diagonal), and closed convex polygons where the edges of the polygons are, again, only horizontal, vertical, diagonal or anti-diagonal. These can all be represented via intersections of half-planes that corresponds to the four different possible ellipsis orientations in the abstract matrix: vertical, horizontal, diagonal and anti-diagonal, the latter two at $\pm 45°$ angles from the horizontal. Each half-plane constrains the indices of the region. The different half-planes and the constraints they impose are given in Fig. 1. In order to capture the idea of half-plane constraints algebraically, we define the following support function:

DEFINITION 5. *Let $x, y \in \mathbb{N}$ then we define*

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

We introduce $\sigma_{x,y}$ as a more compact notation for $\sigma(x, y)$. We will sometimes also only use σ or an indexed version σ_1, etc. if the actual arguments are irrelevant.

One obvious property of the σ function is $\sigma_{x+z,y} = \sigma_{x,y-z}$. Further, the complement of $\sigma_{x,y}$ is $\overline{\sigma_{x,y}} = \sigma_{y,x-1}$.

Containment within a half-plane can be neatly captured by a single σ function, and containment within a region by a product of σs. Clearly a product of σs always describes a convex region. For example we can represent the a triangular region in (1) as $\sigma_{1,i}\sigma_{j,n}\sigma_{i,j}a_{ij}$, where $\sigma_{1,i}$ restricts the region to be on or below the top boundary of the triangle, $\sigma_{j,n}$ 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 to 0.

To make index variables and the overall dimension of an abstract matrix explicit we introduce the following notation:

NOTATION 1. *Let $n, m \in \mathbb{N}$ and i, j be names of index variables ranging from 1 to n and m, respectively. Then we*

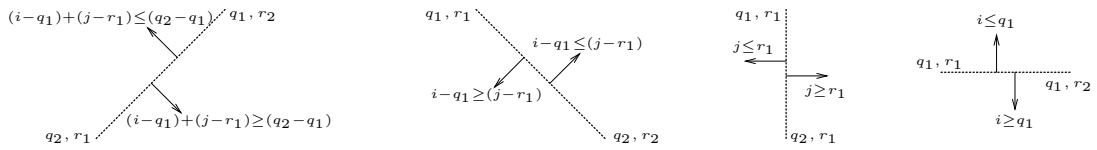


Figure 1: Half-plane constraints. r_1, r_2, q_1, q_2 are the general coordinates representing the start and end points of ellipses in an abstract matrix.

denote an abstract matrix as

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

The pair of possible values of the index variables i, j define the dimensions of the matrix. x is a term that can, but does not have to, be a functional expression involving the variables i, j.

We write the non-zero regions of B as a linear combination of support functions:

$$B = [\sigma_{1,i}\sigma_{j,n}\sigma_{i,j}a_{ij} + \sigma_{n+1,i}\sigma_{j,n+m}\sigma_{i,j}b_{i-n,j-n}]_{i,j}^{m+n,m+n} \quad (2)$$

This notation describes B as a piecewise function in the variable i, j. We therefore call this notation the *algebraic* or *closed form* of the matrix B. Each summand represents one single convex region. Moreover the σ coefficients of the single summands are mutually exclusive, that is, for each different pair of values i, j at most one summand will be different from 0. This corresponds to the disjointness condition on the regions.

Note that the term for B 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 in (2) by removing the unnecessary σs:

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

This expression now fully captures the abstract matrix B. Moreover, one can still easily see the form of the matrix as the occurring regions are clearly given as the different summands and their boundaries can be easily regained from analysing the σ coefficients.

For the remainder of the paper we will omit indices for the region entries, as this will make the formulae significantly more legible.

We find it useful to consider an alternative view of products of sigma terms; namely as the regions that the underlying intersection of half planes include. This then leads to a set theoretic language of combining sigmas and the usual results, based on the inclusion/exclusion principle, on indicator or characteristic functions.

DEFINITION 6. *Let Γ_1, Γ_2 be products of σ terms. We define the set notation:*

(i) $\Gamma_1 \cap \Gamma_2 = \Gamma_1\Gamma_2$
(ii) $\Gamma_1 \cup \Gamma_2 = \Gamma_1 + \Gamma_2 - \Gamma_1\Gamma_2$
(iii) $\Gamma_1 \setminus \Gamma_2 = \Gamma_1\overline{\Gamma_2}$

Here $\overline{\Gamma_2}$ denotes the complement of Γ_2, i.e., the outside of the region described by Γ_2. To calculate the complement of a convex region, we consider the half-plane components, the σs of the shape, and sequentially add the complement of each half-plane in the product of σs, restricted to the intersection of the half-planes that have already been captured:

DEFINITION 7. *Let $\Gamma = \sigma_1\sigma_2\ldots\sigma_n$, then its complement Γ is defined as $\overline{\Gamma} = \overline{\sigma_1\sigma_2\ldots\sigma_n} = \overline{\sigma_1} + \overline{\sigma_2}\sigma_1 + \overline{\sigma_3}\sigma_1\sigma_2 + \cdots + \overline{\sigma_n}\sigma_1\sigma_2\ldots\sigma_n$*

Observe that since the complement of a region is a sum, it is generally concave. When intersecting it with a convex region, i.e., multiplying the sum with a product of σs we get a sum of convex regions as a result.

3. NAÏVE ARITHMETIC

3.1 Naïve Addition

Given the representation of two abstract matrices using support functions, the obvious algorithm will compute a support function representation of their addition. It suffices to show an example. Consider addition of two $N \times M$ abstract matrices A and B where $N = q + r$ and $M = n + m$.

$$A + B = \underbrace{\begin{bmatrix} a & \cdots & a & 0 & \cdots & 0 \\ \vdots & & \vdots & \vdots & & \vdots \\ a & \cdots & a & 0 & \cdots & 0 \end{bmatrix}}_{n \qquad m} + \begin{bmatrix} r\left\{\begin{matrix} b & \cdots & b \\ \vdots & & \vdots \\ b & \cdots & b \end{matrix}\right. \\ q\left\{\begin{matrix} c & \cdots & c \\ \vdots & & \vdots \\ c & \cdots & c \end{matrix}\right. \end{bmatrix} \quad (3)$$

This sum is computed by simply adding the algebraic representations of A and B:

$$\begin{aligned} A + B &= [\sigma_{j,n}\ a]_{i,j}^{N,M} + [\sigma_{i,r}\ b + \sigma_{r+1,i}\ c]_{i,j}^{N,M} \\ &= [\sigma_{j,n}\ a + \sigma_{i,r}\ b + \sigma_{r+1,i}\ c]_{i,j}^{N,M} \end{aligned} \quad (4)$$

This indeed yields the correct result that fully describes the $A + B$ matrix in 7; depending on the i, j values the expression either evaluates to $a + b$, $a + c$, b, or c. Note, however, that, in this form, the 3 summands of the result do not correspond simply to the 4 rectangular regions that the resulting abstract matrix has.

3.2 Naïve Multiplication

Consider the following matrix multiplication between two matrices of size $N \times N$, where $N = n + m$. Note that the block sizes have been reversed between the two matrices to ensure an "interesting" interaction between their respective regions on multiplication:

$$A \cdot B = \begin{matrix} \\ m \\ n \end{matrix}\overset{\begin{matrix} m & n \end{matrix}}{\begin{bmatrix} c & \cdots & c & d & \cdots & d \\ & \ddots & \vdots & \vdots & & \vdots \\ & & c & \vdots & & \vdots \\ \mathbf{0} & & & \vdots & & \vdots \\ & & & d & \cdots & d \end{bmatrix}} \cdot \overset{\begin{matrix} n & m \end{matrix}}{\begin{bmatrix} a & \cdots & a & & \\ & \ddots & \vdots & & \mathbf{0} \\ & & a & & \\ & & & b & \cdots & b \\ \mathbf{0} & & & & \ddots & \vdots \\ & & & & & b \end{bmatrix}} \begin{matrix} \\ \left.\begin{matrix} \\ \\ \end{matrix}\right\} n \\ \left.\begin{matrix} \\ \\ \end{matrix}\right\} m \end{matrix}$$

The general calculation we have to perform is:

$$A \cdot B = [A_{ij}]_{i,j}^{N,N} \cdot [B_{ij}]_{i,j}^{N,N} = \left[\sum_{k=1}^{N} A_{ik} B_{kj}\right]_{i,j}^{N,N}$$

Translating A and B into their closed forms we get:

$$\begin{aligned} A \cdot B \quad = \quad & [\sigma_{j,m}\sigma_{i,j}\ c + \sigma_{m+1,j}\ d]_{i,j}^{N,N} \\ & \cdot [\sigma_{j,n}\sigma_{i,j}\ a + \sigma_{n+1,i}\sigma_{i,j}\ b]_{i,j}^{N,N} \end{aligned} \tag{5}$$

The product, AB, is then calculated as:

$$\left[\sum_{k=1}^{N} \begin{array}{l} \sigma_{k,m}\sigma_{i,k}\sigma_{j,n}\sigma_{k,j}\ ca + \sigma_{m+1,k}\sigma_{j,n}\sigma_{k,j}\ da + \\ \sigma_{k,m}\sigma_{i,k}\sigma_{n+1,k}\sigma_{k,j}\ cb + \sigma_{m+1,k}\sigma_{n+1,k}\sigma_{k,j}\ db \end{array}\right]_{i,j}^{N,N} \tag{6}$$

This sum captures all possible results of the multiplication. The single summation can clearly be split into 4 separate sums, one for each of the summands in the summation. Each of the resulting terms represents one of four possible regions. Each region is convex, because their σ coefficients corresponds to a product of half-planes, which always results in a convex region. However, these regions are not disjoint and therefore a direct reading of the region structure of the result is difficult.

In summary, the results for both naïve addition and multiplication are correct but the form of these results have serious problems with respect to recovering their region structure: (a) The single summands may no longer constitute single regions. (b) The regions they describe may overlap and we no longer have mutually exclusive σ coefficients for each summand. (c) How to determining the shape of region described by a single summand is not obvious.

4. TERM GRAMMAR

We first define a term grammar on which our procedure works and give a normal form for the closed representations of abstract matrices that ensures that the region structure is easily recoverable from abstract matrix terms in normal form.

Definition 8. *The abstract matrix term grammar is:*

$$\begin{array}{lll} M ::= [F]_{\mathsf{var},\mathsf{var}}^{\mathsf{iexp},\mathsf{iexp}} & \quad \Gamma ::= \sigma_{\mathsf{iexp},\mathsf{iexp}}\ \Gamma \mid \epsilon \\ F ::= R \mid R + F & \quad T ::= \mathsf{exp} \mid \sum_{\mathsf{var}=1}^{\mathsf{iexp}} (F) \mid F \\ R ::= \Gamma\ T \mid T & \\ \mathsf{iexp} ::= \mathsf{var} \mid \mathsf{int} \mid -\mathsf{iexp} \mid \mathsf{iexp} + \mathsf{iexp} \mid \mathsf{iexp} - \mathsf{iexp} & \end{array}$$

Where we assume the definition of the following terms: var *for a single variable name,* int *for a single, non-negative integer, and* exp *for an arbitrary functional expression, that can contain* var *and* int *terms.*

For notational convenience we shall use the non-terminals to represent the particular terms they generate, indexing them if necessary, throughout the paper. We denote abstract matrices in general as

$$[M]_{i,j}^{m,n} = [R_1 + \cdots + R_k]_{i,j}^{m,n} = [\Gamma_1 T_1 + \cdots + \Gamma_k T_k]_{i,j}^{n,m}$$

We now specify the conditions under which the region structure of an abstract matrix is easily analyseable. The intuition is that the abstract matrix term should be of the form $[\Gamma_1 T_1 + \cdots + \Gamma_k T_k]_{i,j}^{n,m}$, where each $\Gamma_s T_s$ summand identifies one region where the size, shape and position of a region is captured by the Γ_S term. Here we use the set notation $w \in x$ and $y \setminus z$ between terms w, x, y, z to mean the occurrence of w in x and the result of removing all occurrences of y from z.

Definition 9. *Let* $[M]_{i,j}^{m,n} = [\Gamma_1 T_1 + \cdots + \Gamma_k T_k]_{i,j}^{n,m}$ *be an abstract matrix, then we say* M *is in* regular form, *if the following holds:*

(i) **Disjoint:** $\Gamma_s \Gamma_{s'} = 0$ *for* s, s' *in* $1 \ldots k$ *with* $s \neq s'$.
(ii) **Convex:** $\Gamma_s = \sigma_{x_1^s, y_1^s} \cdots \sigma_{x_p^s, y_p^s}$, *for* s *in* $1 \ldots k$, $p \geq 1$.
(iii) **Γ-Partitioned:** *for* s *in* $1 \ldots k$, *there does not exist a* $\sigma_{x,y} \in T_s$ *such that* $\sigma_{x,y} T_s = T_s$.
(iv) **Non-Empty:** $\Gamma_s \neq 0$ *for* s *in* $1 \ldots k$
(v) **Γ-Minimal:** *for every* Γ *in* $\Gamma_1 \ldots \Gamma_k$ *and every* $\sigma_{x,y} \in \Gamma$ *we have* $\Gamma \setminus \sigma_{x,y} \neq \Gamma$.

Note that Γ_s can evaluate to 1, meaning that T_s spans the entire matrix.

The disjointness property means that no two summands can define overlapping regions, hence each summand is the unique generator of values within its region.

The convexity property ensures that each region is properly an intersection of half-planes and hence convex. Note that this condition is actually required by the term grammar itself and is therefore a requirement of all output from the full addition and multiplication algorithms.

The Γ-Partitioned requirement makes sure that all possible structural information on a region ΓT is indeed given in Γ and no additional structural information might be hidden in T. If this condition is not satisfied, then T will be forced to 0 in some half-plane, independent of the value of Γ. Hence Γ does not properly describe the true region but may be merely a non-minimal upper bound of it.

The non-empty property insures that we do not have redundant ΓT terms: i.e. terms that appear to define regions but which covers no space.

The Γ-Minimal property ensures that the region description defined by Γ is minimal, in the sense that it contains the smallest possible number of σ coefficients necessary to define the full region.

5. ADDITION

We now develop addition for abstract matrices revisiting the example from §3.1. Recall that the result of simply adding the support function representation given in (4) could be interpreted as three, non-disjoint regions. If we observe the matrix addition using the structural depiction we instead get the following result, which clearly contains four disjoint regions, some containing sums of elements:

$$A + B \quad = \quad \begin{bmatrix} (a+b) & \cdots & (a+b) & b & \cdots & b \\ \vdots & & \vdots & \vdots & & \vdots \\ (a+b) & \cdots & (a+b) & b & \cdots & b \\ (a+c) & \cdots & (a+c) & c & \cdots & c \\ \vdots & & \vdots & \vdots & & \vdots \\ (a+c) & \cdots & (a+c) & c & \cdots & c \end{bmatrix} \tag{7}$$

In order to motivate how we can separate the resulting regions, we shift to a set view of regions. Let $\mathcal{A}, \mathcal{B}, \mathcal{C}$ be sets representing the regions containing a, b, c in the matrices A, B, respectively. Then we can describe the addition of the regions as the union of the sets. Since we know that $\mathcal{B}$

and $\mathcal{C}$ are disjoint, this set is composed of the five subsets $\mathcal{A}\cap\mathcal{B}$, $\mathcal{A}\cap\mathcal{C}$, $\mathcal{A}\setminus\mathcal{B}\setminus\mathcal{C}$, $\mathcal{B}\setminus\mathcal{A}$, and $\mathcal{C}\setminus\mathcal{A}$. We can compute an intersection of two sets by simply taking the intersection of the half-planes that define the sets, which corresponds to the product of the respective σ terms. A set of the form $\mathcal{B}\setminus\mathcal{A}$ corresponds to the intersection of the half-planes defining $\mathcal{B}$ and those defining the exterior of $\mathcal{A}$, which are the complements of the σs defining $\mathcal{A}$. Thus we can rewrite (4) into

$$\begin{bmatrix} \sigma_{j,n}\sigma_{i,m}\ (a{+}b) + \sigma_{j,n}\sigma_{m+1,i}\ (a{+}c){+} \\ \sigma_{j,n}\overline{\sigma_{i,m}\sigma_{m+1,i}}\ a + \overline{\sigma_{j,n}}\sigma_{i,m}\ b + \overline{\sigma_{j,n}}\sigma_{m+1,i}\ c \end{bmatrix}^{N,M}_{i,j}$$
$$= \begin{bmatrix} \sigma_{j,n}\sigma_{i,m}\ (a{+}b) + \sigma_{j,n}\sigma_{m+1,i}\ (a{+}c){+} \\ \sigma_{j,n}\sigma_{m+1,i}\sigma_{i,m}a + \sigma_{n+1,j}\sigma_{i,m}b + \sigma_{n+1,j}\sigma_{m+1,i}c \end{bmatrix}^{N,M}_{i,j} \quad (8)$$

The resulting matrix still contains five regions. However, we can observe that the term representing the a region contains an inconsistent σ coefficient in $\sigma_{m+1,i}\sigma_{i,m}$. Thus this region vanishes and we can rewrite (8) into

$$\begin{bmatrix} \sigma_{j,n}\sigma_{i,m}\ (a{+}b) + \sigma_{j,n}\sigma_{m+1,i}\ (a{+}c){+} \\ \sigma_{n+1,j}\sigma_{i,m}\ b + \sigma_{n+1,j}\sigma_{m+1,i}\ c \end{bmatrix}^{N,M}_{i,j}$$

This expression is indeed an abstract matrix. (a) Each summand represents a non-empty single region, (b) the regions are disjoint, (c) convex as they are solely given by intersections of half-planes, (d) and Γ-Partitioned as the shape of each region is fully captured by the preceding σ term.

We now generalise the procedure to obtain the addition algorithm ADD. Let A, B be abstract matrices of the form

$$A = \left[\Gamma^A_1 T^A_1 + \cdots + \Gamma^A_1 T^A_R\right]^{m,n}_{i,j} \quad B = \left[\Gamma^B_1 T^B_1 + \cdots + \Gamma^B_S T^B_S\right]^{m,n}_{i,j}$$

where Γ^A_l,Γ^B_l indicates that the respective Γ term originates from matrix A or B, respectively. To compute the set of all possible component regions of $A+B$ we first define the set notation for all regions as $P_r = \Gamma^A_r T^A_r$ and $Q_s = \Gamma^B_s T^B_s$, where $r \in \mathcal{R} = \{1,\dots R\}$, $s \in \mathcal{S} = \{1,\dots S\}$. We then can combine the sets of regions into disjoint component regions of the sum by

$$\begin{array}{lll} \{P_r \cap Q_s\}_{(r,s)\in(\mathcal{R},\mathcal{S})} & \cup & \{P_r \setminus Q_1 \setminus \cdots \setminus Q_S\}_{r\in\mathcal{R}} \\ & \cup & \{Q_s \setminus P_1 \setminus \cdots \setminus P_R\}_{s\in\mathcal{S}} \end{array}$$

Exploiting Def 6 we can finally define $\mathsf{ADD}(A,B) :=$

$$\begin{bmatrix} \Gamma^A_1\Gamma^B_1(T^A_1{+}T^B_1) + \Gamma^A_1\Gamma^B_2(T^A_1{+}T^B_2) + \cdots + \Gamma^A_R\Gamma^B_S(T^A_R{+}T^B_S){+} \\ \Gamma^A_1\overline{\Gamma^B_1\cdots\Gamma^B_S}T^A_1 + \Gamma^A_2\overline{\Gamma^B_1\cdots\Gamma^B_S}T^A_2 + \cdots + \Gamma^A_R\overline{\Gamma^B_1\cdots\Gamma^B_S}T^A_R{+} \\ \Gamma^B_1\overline{\Gamma^A_1\cdots\Gamma^A_R}T^B_1 + \Gamma^B_2\overline{\Gamma^A_1\cdots\Gamma^A_R}T^B_2 + \cdots + \Gamma^B_S\overline{\Gamma^A_1\cdots\Gamma^A_R}T^B_S \end{bmatrix}^{m,n}_{i,j}$$

With Def 7, complemented σ products are replaced by the appropriate sums of σ products (the negative intersection terms vanish as the union terms are mutually disjoint), and the entire expression is expanded into a simple sum of ΓT expressions of the form

$$[\Gamma_1 T_1 + \Gamma_2 T_2 + \cdots + \Gamma_N T_N]^{m,n}_{i,j}$$

Observe that the above representation might still contain empty regions, similar to the a-region in (8). But we can show the following properties:

THEOREM 10. *Let A, B be abstract matrices, and let $C = \mathsf{ADD}(A,B) = [\Gamma_1 T_1 + \Gamma_2 T_2 + \cdots + \Gamma_N T_N]^{m,n}_{i,j}$, then the regions in C are (i) disjoint, (ii) convex, (iii) Γ-Partitioned,*

PROOF. (i) The regions are disjoint by construction. (ii) Each Γ_i is a simple product of σ terms and thus convex. (iii) As A, B are Γ-Partitioned, no term T_j in A, B contains any relevant σ. Since ADD does not introduce any new σ expressions in terms T_i of C, it is trivially Γ-Partitioned. □

We shall rely on the normalisation process (§7) to rewrite the result so that the non-empty and Γ-Minimal properties are also satisfied.

6. MULTIPLICATION

To develop the multiplication algorithm for abstract matrices we revisit again the example from §3.2. Recall that the result of multiplying the support representation of our matrices A and B was

$$\begin{bmatrix} \sum_{k=1}^{n+m} \sigma_{k,m}\sigma_{i,k}\sigma_{j,n}\sigma_{k,j}ca + \sum_{k=1}^{n+m} \sigma_{m+1,k}\sigma_{j,n}\sigma_{k,j}da{+} \\ \sum_{k=1}^{n+m} \sigma_{k,m}\sigma_{i,k}\sigma_{n+1,k}\sigma_{k,j}cb + \sum_{k=1}^{n+m} \sigma_{m+1,k}\sigma_{n+1,k}\sigma_{k,j}db \end{bmatrix}^{N,N}_{i,j}$$

If we compare this expression with the structural depiction of the product matrix we not only observe that there are three different possible results, depending on the relationship between the symbolic variables n and m, but also that there are more than four different possible regions in those cases. Observe that in the matrices below we abstract away from the actual multiplicity of the single elements occurring in the different regions, thus omitting complex sum expressions, as this can be subsumed in the presentation of the evaluating terms. Hence the elements only indicate the interaction of regions from the original matrices A and B. Thus the entries ca means that the region contains entries that are products of entries from the a-region in A and the c-region in B.

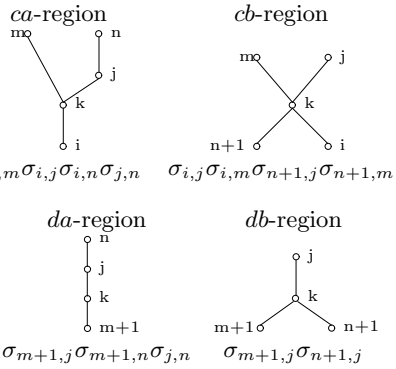
$\mathbf{n = m}$:
$$\overbrace{\begin{bmatrix} ca & \cdots & ca \\ & \ddots & \vdots \\ & & ca \\ \multicolumn{3}{c}{\mathbf{0}} \end{bmatrix}}^{n=m} \overbrace{\begin{bmatrix} db & \cdots & db \\ \vdots & & \vdots \\ \vdots & & \vdots \\ db & \cdots & db \end{bmatrix}}^{n=m}$$

$\mathbf{n > m}$:
$$\overbrace{\begin{bmatrix} ca & \cdots & ca \\ & \ddots & \vdots \\ & & ca \\ \multicolumn{3}{c}{\mathbf{0}} \end{bmatrix}}^{m} \overbrace{\begin{bmatrix} ca+da & \cdots & ca+da \\ \vdots & & \vdots \\ ca+da & \cdots & ca+da \\ da & \cdots & da \\ \vdots & & \vdots \\ da & \cdots & da \end{bmatrix}}^{n-m} \overbrace{\begin{bmatrix} db & \cdots & db \\ \vdots & & \vdots \\ \vdots & & \vdots \\ db & \cdots & db \end{bmatrix}}^{m}$$

$\mathbf{n < m}$:
$$\overbrace{\begin{bmatrix} ca & \cdots & ca \\ & \ddots & \vdots \\ & & ca \\ cb & & \vdots \\ & \ddots & \vdots \\ & & cb \\ \multicolumn{3}{c}{\mathbf{0}} \end{bmatrix}}^{n} \overbrace{\begin{bmatrix} cb & \cdots & cb \\ \vdots & & \vdots \\ \vdots & & \vdots \\ \vdots & & \vdots \\ cb+db & \cdots & cb+db \\ db & \cdots & db \\ \vdots & & \vdots \\ db & \cdots & db \end{bmatrix}}^{m-n} \overbrace{\begin{bmatrix} cb+db & \cdots & cb+db \\ \vdots & & \vdots \\ \vdots & & \vdots \\ \vdots & & \vdots \\ cb+db & \cdots & cb+db \\ db & \cdots & db \end{bmatrix}}^{n}$$

Since we get three different explicit cases that are implicitly included in the closed form of the product in (6), we need to extract all possible disjoint regions, as we did for matrix addition. However, now we have the added problem that the shapes of the regions are not obvious as the σ terms describing them are obscured by Σ expressions that involve the index variable k of the sum. These terms describe the multiplicity of a region entry for a particular index pair (i,j).

We therefore also have to find the tightest possible boundary for each of the regions by extracting the most general product of σs from under each sum. We do this by computing the product of all the σs that are implied by each summand but are independent of the summation variable. We generate this σ-expression by completing the partial order on the σ limits, and display them as a set of Hasse diagrams [2][1].



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

$\sigma_{m+1,j}\sigma_{m+1,n}\sigma_{j,n}$ $\sigma_{m+1,j}\sigma_{n+1,j}$

In practice, we only need a minimal set of inequalities that imply all the inequalities in the Hasse diagram. For example, we can eliminate $\sigma_{i,n}$ for the summand of the ca-region: $\sigma_{i,m}\sigma_{i,j}\sigma_{j,n}\sum_{k=1}^{n+m}(\sigma_{k,m}\sigma_{i,k}\sigma_{k,j}\ ca)$. Observe that we have removed all σ-expressions not involving k from inside the sum. The remainder can not be removed.

In the next step we separate out overlapping regions by generating all possible disjoint regions. Unlike the case of addition, we can no longer exploit disjointness of regions in the original matrices as the row column multiplication cuts through all regions. Thus we have to look at all possible combinations of summands, i.e., 15 altogether. We use again the set view. Let $\mathcal{CA}$, $\mathcal{CB}$, $\mathcal{DA}$, $\mathcal{DB}$, be the sets representing the regions containing ca, cb, da, db respectively. Then $\mathcal{CA} \setminus \mathcal{DA} \setminus \mathcal{CB} \setminus \mathcal{DB}$ represents the region containing only ac, $(\mathcal{CA} \cap \mathcal{DA}) \setminus \mathcal{CB} \setminus \mathcal{DB}$ the one containing $ca+db$ and so on. We can compute intersections as products of σs and set difference by intersection with a region's complement. Some of the simplified regions are $\sigma_{i,m}\sigma_{i,j}\sigma_{j,n}\sigma_{j,m}ca$, $\sigma_{i,m}\sigma_{i,j}\sigma_{j,n}\sigma_{m+1,j}(ca + da)$, $\sigma_{i,m}\sigma_{i,j}\sigma_{j,n}\sigma_{m+1,j}\sigma_{n+1,m}(ca + da + cb)$, etc. In the last expression we have $\sigma_{j,n}, \sigma_{n+1,j}$, which are obviously inconsistent, meaning that the region vanishes. Equation (6) can thus be rewritten with simplified regions as:

$$\begin{bmatrix} \sigma_{i,m}\sigma_{i,j}\sigma_{j,n}\sigma_{j,m}ca + \sigma_{i,m}\sigma_{i,j}\sigma_{j,n}\sigma_{m+1,j}(ca + da) \\ +\sigma_{m+1,j}\sigma_{j,n}\sigma_{m+1,i}da + \sigma_{i,j}\sigma_{i,m}\sigma_{n+1,j}\sigma_{j,m}cb \\ +\sigma_{i,j}\sigma_{i,m}\sigma_{n+1,j}\sigma_{m+1,j}\sigma_{j,m}(cb + db) + \sigma_{m+1,j}\sigma_{n+1,j}db \end{bmatrix}_{i,j}^{N,N}$$

This matrix now only contains disjoint, convex regions and the shapes of the regions are obvious. Moreover, the relationships between the sigma terms of the different summands contain information that yield the correct cases with respect to the relationship of n and m. For instance, $\sigma_{j,n}$ and $\sigma_{n+1,j}$ in the third and fourth summand, respectively, determine that there cannot exist an instance of the matrix that contains both an da and cb region.

We now present our general algorithm for multiplication, which is based on the standard dot product of rows with columns. However, this is carried out pairwise on regions, and the resulting terms, for any particular cell of the product matrix, may be from intersecting or overlapping regions. To end up with a disjoint sum of regions, we need to deal with these overlaps in a manner similar to the case of addition, via calculating intersections and differences of regions. As opposed to our example we calculate this before splitting the sum into separate regions. While this is slightly less intuitive it yields the same result and has the benefit that we can cleanly separate the pure multiplication from rewriting the result into its regular form.

Let A, B be abstract matrices of the form

$$A = \left[\Gamma_1^A T_1^A + \cdots + \Gamma_1^A T_R^A\right]_{i,j}^{m,p} \quad B = \left[\Gamma_1^B T_1^B + \cdots + \Gamma_S^B T_S^B\right]_{i,j}^{p,n}$$

The abstract matrix multiplication algorithm MULT is defined as follows:

$$\begin{aligned} &\mathsf{MULT}(A,B) := [(AB)_{i,j}]_{i,j}^{m,n} = \left[\sum_{k=1}^{p} A_{i,k}B_{k,j}\right]_{i,j}^{m,n} \\ &= \left[\sum_{k=1}^{p}\left(\Gamma_1^A T_1^A + \cdots + \Gamma_R^A T_R^A\right)\left(\Gamma_1^B T_1^B + \cdots + \Gamma_S^B T_S^B\right)\right]_{i,j}^{m,n} \\ &= \left[\sum_{k=1}^{p}\left(\Gamma_1^A\Gamma_1^B T_1^A T_1^B + \cdots + \Gamma_R^A\Gamma_S^B T_R^A T_S^B\right)\right]_{i,j}^{m,n} \\ &= \left[\sum_{k=1}^{p}\left(\Gamma_1 T_1 + \Gamma_2 T_2 + \cdots + \Gamma_N T_N\right)\right]_{i,j}^{m,n} \end{aligned}$$

In the last step we have performed an obvious renaming, as the distinction between terms originating from A and B is no longer necessary. While the Γ^A and Γ^B terms describe convex regions, the product of a Γ^A and a Γ^B term does not describe a simple intersection of the two convex regions as one might expect. This is because the half-planes described by the $\sigma_{x,y}$ terms in Γ^A describe half-planes with respect to i, k index variables and those of Γ^B describe half-planes with respect to k, j index variables. Their product, however, is being interpreted with respect to an i, j indexed space. Thus the k variable, in this context, no longer has the special role of an index variable and, instead, acts like an integer constant in the same way that variables like the matrix width or height do. Hence terms such as $\sigma_{k,3}$ are not half-plane constraints, but merely ordering constraints that link the real index variables to other constants to form the half-plane constraints (e.g. $\sigma_{i,k}\sigma_{k,3}$), or simple control conditionals that lead to different region shapes depending on relative size of the underlying region. It should be noted, however, that the product of a Γ^A and a Γ^B is just a product of individual $\sigma_{x,y}$ terms, even if only some of those correspond to half-plane constraints, and thus still denotes a convex region.

To factor these overlapping regions into disjoint ones, we need to apply some elementary set manipulations. If there are N potentially overlapping regions, then these can be decomposed into $2^N - 1$ disjoint component regions. Each such component region is formed by constructing a region R as the intersection of some subset $\mathcal{R}$ of the regions, and removing any parts of R that are in any of the other regions, i.e., intersecting R with the intersection of the complements of the remaining non-$\mathcal{R}$ regions. The content of these regions then consists of the sum of the elements from the intersected

[1]Note that we abuse the Hasse diagram notation by using non-strict rather than the usual strict inequalities.

regions. We then compute the $2^N - 1$ regions as follows:

Let $\mathcal{N} = \{1, \ldots, N\}$ and $\mathcal{P}(\mathcal{N})$ its powerset. For each $\mathcal{I} \in \mathcal{P}(\mathcal{N}) \backslash \emptyset$ with $\mathcal{I} = \{s_1, \ldots, s_m\}$ and $\mathcal{N} \backslash \mathcal{I} = \{s_{m+1}, \ldots, s_N\}$, where $1 \leq m \leq N$ and $s_i \in \mathbb{N}$, $i = 1, \ldots, N$ compute

$$R_{\mathcal{I}} = (\Gamma_{s_1} \cdots \Gamma_{s_m} \overline{\Gamma_{s_{m+1}} \cdots \Gamma_{s_N}})(T_{s_1} + \cdots + T_{s_m}) \quad (9)$$

If we enumerate the subsets we can consequently combine the expressions from (9) as matrix:

$$\left[\sum_{k=1}^{p} \left(R_1 + \cdots + R_{2^N-1}\right)\right]_{i,j}^{m,n}$$

Since the R_i still contain complemented σ, which can be expanded out fully into the appropriate sum of σ products, and this sum is distributed across the non-complemented σ product and the remaining T-expressions, we finally get a simple sum of ΓT expressions of the form

$$\left[\sum_{k=1}^{p} \left(\Gamma'_1 T'_1 + \Gamma'_2 T'_2 + \cdots + \Gamma'_{N'} T'_{N'}\right)\right]_{i,j}^{m,n}$$
$$= \left[\sum_{k=1}^{p} \left(\Gamma'_1 T'_1\right) + \sum_{k=1}^{p} \left(\Gamma'_2 T'_2\right) + \cdots + \sum_{k=1}^{p} \left(\Gamma'_{N'} T'_{N'}\right)\right]_{i,j}^{m,n}$$

The final result of the MULT algorithm has the following properties:

THEOREM 11. *Let A, B be abstract matrices, and let $C = \mathsf{MULT}(A, B) = [\Gamma_1 T_1 + \Gamma_2 T_2 + \cdots + \Gamma_N T_N]_{i,j}^{m,n}$, then the regions in C are disjoint and convex.*

PROOF. Disjointness of the result regions is guaranteed by the set construction. And since each Γ_i, $i = 1, \ldots, N$ is a product of single σ terms the regions are also convex. □

We observe that the result of MULT is not Γ-Partitioned, non-empty or Γ-Minimal. To regain these properties, we use a rewrite system for normalisation presented in the next section.

7. NORMALISATION

The results of the algorithms ADD and MULT are generally not in the desired regular form, i.e. the regions of the resulting matrices are not in a discernible form. To regularise terms after each execution of ADD or MULT we employ a rewrite system NORM. The advantage of using a formal rewrite system is that it allows more convenient reasoning on theoretical properties such as termination and confluence[2]

NORM takes terms that are not in regular form as input. The only assumption we make is that all regions in the input term are convex and disjoint. But this is guaranteed by the output forms of the two algorithms. Since the output of the rewrite system is an abstract matrix in the regular form, it can again be used as input for the addition or multiplication algorithm and is thus amenable for further arithmetic manipulations.

We define the rewrite system NORM with the rules given in Table 1. For these rules we require the following notational conventions. We define a semantic partial order on σ subscript terms for use in the side conditions of the rules:

[2]For an introduction to basic terminology of term rewriting systems, see [1].

Contraction Rules

$$\frac{\sigma_{x,y}}{0}\ 0\text{-}\sigma,\ \text{if } y \preccurlyeq x \qquad \frac{0T}{0}\ 0\text{-}T \qquad \frac{\sum_{k=1}^{p}(0)}{0}\ 0\text{-}\Sigma$$

$$\frac{0\sigma_{x,y}}{0}\ 0\text{-}\sigma_r \qquad \frac{\sigma_{x,y}0}{0}\ 0\text{-}\sigma_l \qquad \frac{0+F}{F}\ 0\text{-}F_r \qquad \frac{F+0}{F}\ 0\text{-}F_l$$

Transitive Closure Rule

$$\frac{\Gamma_1\sigma_{w,x}\Gamma_2\sigma_{y,z}\Gamma_3 T}{\Gamma_1\sigma_{w,x}\Gamma_2\sigma_{y,z}\Gamma_3\sigma_{w+y,x+z}T}\ \text{Trans},\quad \text{if } \sigma_{w+y,x+z} \notin \Gamma_1\sigma_{w,x}\Gamma_2\sigma_{y,z}\Gamma_3$$

Factoring Rule

$$\frac{\Gamma \sum_{k=1}^{p} \Gamma_1\sigma_{x,y}\Gamma_2 T}{\Gamma\sigma_{x,y}\sum_{k=1}^{p}\Gamma_1\Gamma_2 T}\ \text{Fact}\ ,\ \text{if } k \notin \{x, y\}$$

Reduction Rules

$$\frac{\Gamma_1\sigma_{w,z}\Gamma_2}{\Gamma_1\Gamma_2}\ \text{Red}_1\ ,\ \text{if } \sigma_{w,x} \lhd \Gamma_1\Gamma_2 \wedge \sigma_{y,z} \lhd \Gamma_1\Gamma_2 \wedge x \preccurlyeq y$$

$$\frac{\sigma_{x,y}\Gamma}{\Gamma}\ \text{Red}_2\ ,\ \text{if } \sigma_{x,y} \lhd \Gamma \qquad \frac{\sigma_{x,y}T}{T}\ \text{Red}_3\ ,\ \text{if } x \preccurlyeq y$$

Table 1: The rules of the system NORM.

DEFINITION 12. *We define the partial order $\preccurlyeq$ on* iexp *terms to be $x \preccurlyeq y$ is true if for all possible bindings of variables,* var*, in x and y, to integers, then $x \leqslant y$ evaluates to true when terms x, y are evaluated as integer expressions.*

Evaluation of $x \preccurlyeq y$ is straightforward; rewrite as $x - y \preccurlyeq 0$, and simplify $x - y$ algebraically, which is straightforward due to the strict limitations on the form of iexp terms, c.f. Def 8. If the resulting expression contains any variables then $x \not\preccurlyeq y$. Otherwise, $x - y$ can be evaluated numerically and $x - y \preccurlyeq 0$ if and only if $x - y \leqslant 0$. In particular, this allows the side condition of 0-σ to be tested on an individual σ term without having to inspect any other terms in the whole expression.

DEFINITION 13. *We define the inclusion relation $\lhd$ to be*
(i) $\sigma_{w,x} \lhd \Gamma$ if there is a $\sigma_{y,z} \in \Gamma$ s.t. $y \preccurlyeq z \Rightarrow w \preccurlyeq x$,
(ii) $\sigma_{w,x} \not\lhd \Gamma$ if there is no $\sigma_{y,z} \in \Gamma$ s.t. $y \preccurlyeq z \Rightarrow w \preccurlyeq x$.

The basic idea of $\lhd$ is to capture all equivalent arithmetic reorderings of the arguments of a σ expression. For example, let $\sigma_{x,y+z} \in \Gamma$ then we have $\sigma_{x-z,y} \lhd \Gamma$. This enables us to state the rewrite system more compactly and thus to show its theoretical properties more easily. However, it will not actually change the power of the system as the condition could be expressed purely syntactically.

We have four sets of rules for contraction, transitive closure, factoring, and reduction, and rewriting occurs in three stages, where in each stage the rules are applied exhaustively before the next stage starts:

Stage 1: Transitive Closure completing the partial order in a given Γ expression. This corresponds to creating all possible σ expressions that can be read off the Hasse diagram.

Stage 2: Factoring pulls single σs that are independent of a summation variable in front of a sum expression.

Stage 3: Reduction reduces a Γ expression to a minimal form while retaining the maximum structural information on the overall partial order by removing all implied σ expressions.

Finally, the contraction rules can be applied during each stage to reduce the complexity of the entire operation, but always have to be exhaustively applied as well, before resuming the rewriting of the particular stage.

For the current system, $\mathcal{R}$, we are able to prove the following theorems:

THEOREM 14 (CORRECTNESS). *$\mathcal{R}$ always yields a term in regular form.*

PROOF SKETCH. We show that the output of the procedure is always a term in regular form. Theorems 10 and 11 show that ADD and MULT produce terms that are (1) disjoint and (2) convex. (3) To show they are Γ-Partitioned, we convince ourselves that (a) the transitive closure rule fully completes the partial order relation, (b) the factoring rule pulls out all relevant information from under the sum, (c) the reduction rules retain the full partial order and only eliminate implied σs, (d) the contraction rules only rewrite inconsistent σ to 0 thus dropping empty regions and ensuring that conditions (4) non-empty, and (5) Γ-Minimal are also met. □

THEOREM 15 (TERMINATION). *$\mathcal{R}$ terminates.*

PROOF SKETCH. It is obvious that both ADD and MULT terminate, as they are deterministic algorithms applied to finite expressions. To show that NORM terminates, we show that each stage of the rewrite process terminates. We define two reduction orderings: for the sum of regions and the size of the Γ terms. It is easy to see that the contraction rules always yield smaller terms wrt. reduction orderings on the overall sum. Likewise reduction and factoring rules strictly reduce the Γ terms on which they operate. Only the transitive closure rule increases Γ terms, and we show that they terminate due both to the finiteness of the Hasse diagram and since Γ terms strictly grow to the right. □

We can also show a confluence result, however only up to reordering and re-association of the elements of Γ terms. We define a equivalence relation $\approx$ such that $\Gamma_1 = \Gamma_2$ iff $\Gamma_1 = \sigma_1 \cdots \sigma_n$ and $\Gamma_2 = \sigma_{(1)\pi} \cdots \sigma_{(n)\pi}$ with $\pi \in S_n$, i.e. a permutation in n elements. Observe that this is sufficient for us since the product of σs are commutative anyway. We can then show

THEOREM 16 (CONFLUENCE). *$\mathcal{R}$ is confluent modulo $\approx$.*

PROOF SKETCH. Since $\mathcal{R}$ terminates, it only remains to show local confluence. We can show for the critical pairs 0-σ and 0-σ_r, 0-σ_l that they can always be joined by 0-T as can not have isolated Γ terms. The remaining critical pairs are copies of Trans and of Fact, respectively. They can be joined when exhaustively applied yielding terms containing Γ terms equivalent under $\approx$. □

8. CONCLUSIONS

We have presented procedures that enable the analysis of structural properties of results from arithmetic operations on abstract matrices. This allows us to show general structural results for infinite classes of matrices under arithmetic using a robust, computational approach. One example of the kind of structural result that we can directly obtain with this approach is that the product of upper triangular matrices is upper triangular. This is an important problem in practice but, to the best of our knowledge, ours is the first successful attempt to provide a comprehensive solution that can automatically solve all possible cases. Our approach uses a combination of procedural and rule based techniques. The advantage of using the rewrite system for the normalisation of results into regular forms is that we can more easily prove the relevant theoretical properties, in particular confluence and correctness, than we can for a procedural approach. The current form of the procedure is the first theoretical design of our approach and does not yet contain possible optimisations. For an efficient implementation, one will need to eagerly exploit information on regions, such as their incompatibility, to bring down complexity, in particular of the multiplication algorithm, which is currently exponential in the number of regions. The current version of the parsing algorithm and the simple arithmetic is implemented in Maple [5]. We intend to build an extension that allows for computing and reasoning on structural properties with the method described in this paper.

9. REFERENCES

[1] F. Baader and T. Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1999.

[2] B. Davey and H. Priestley. *Introduction to Lattices and Order*. Cambridge University Press, 2002.

[3] R. Fateman. Manipulation of matrices symbolically, 2003. `www.eecs.berkeley.edu/~fateman/papers/symmat2.pdf`

[4] M. Kauers and C. Schneider. Application of unspecified sequences in symbolic summation. In *Proc. of ISSAC 2006*, pages 177–183, 2006.

[5] Maplesoft. *Maple 12 User Manual*. Maplesoft, 2008.

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

[7] A. P. Sexton, V. Sorge, and S. M. Watt. Arithmetic on matrices with blocks of symbolic size. In *ISSAC 2007 poster abstracts*, 41(1-2):39–40 of *ACM Commun. Comput. Algebra*. 2007.

[8] A. P. Sexton, V. Sorge, and S. M. Watt. Abstract matrix arithmetic. In *Proc. of SYNASC-2008*. IEEE Computer Society Press, 2009.

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

[10] S. M. Watt. Functional decomposition of symbolic polynomials. In *Proc. of ICCSA 2008*, p. 353–362. IEEE Computer Society, 2008.

[11] S. Wolfram. *The Mathematica book*. Wolfram Media, Inc., 5^{th} edition, 2003.