

Reasoning on Abstract Matrix Structures

Alan P. Sexton and Volker Sorge

School of Computer Science, University of Birmingham

A.P.Sexton|V.Sorge@cs.bham.ac.uk, www.cs.bham.ac.uk/~aps|~vxs

Matrices are often presented with symbolic dimensions and with a mixture of terms and ellipsis symbols to describe their structure. Such a representation effectively determines entire classes of matrices with respect to their common structural properties. When reasoning with such abstract matrices, one is often only interested in the interaction of structural properties of matrix classes under arithmetic operations, while ignoring the actual content of the matrices involved. For example, when showing that the product of two upper triangular matrices is an upper triangular matrix, one shows this property by considering the interaction of the zero and non-zero regions, while ignoring the actual composition of the latter. While this type of reasoning is everyday mathematical practice there is little automated support for this.

We present have developed an algebraic encoding of abstract matrices as linear combinations of basis functions that enables us to define addition and multiplication algorithms. Since recovering structural properties from the results of these operations is non trivial we have designed a rewrite system that normalises the results to a form from which the structural properties can be directly read. We can thus regain the resulting abstract matrix, which provides an effective procedure for reasoning about structural properties of classes of abstract matrices under arithmetic operations. The work is based on (1; 2).

Basis Function

Abstract matrices consist of three types of regions: single terms, single lines or ellipses, and closed convex polygons. These can all be represented via intersections of half planes that corresponds to the four different possible ellipses in the abstract matrix: vertical, horizontal, diagonal and anti-diagonal, the latter two at $\pm 45^\circ$ angles. Each half plane constrains the indices of the region. In order to capture the idea of half plane constraints algebraically, we define a basis function

$\sigma(x, y) = \begin{cases} 1 & \text{if } x \leq y \\ 0 & \text{otherwise} \end{cases}$ for $x, y \in \mathbb{N}$. We often write $\sigma_{x,y}$ as shorthand and can define the complement of $\sigma_{x,y}$ to be $\overline{\sigma_{x,y}} = \sigma_{y,x-1}$. As example consider:

$$M = \begin{bmatrix} a_{11} \cdots a_{1n} & & & & \\ & \ddots & \vdots & \mathbf{0} & \\ & & a_{nn} & & \\ & & & b_{11} \cdots b_{1m} & \\ \mathbf{0} & & & & \ddots & \vdots \\ & & & & & b_{mm} \end{bmatrix} = \begin{cases} [\sigma_{1,i} \sigma_{j,n} \sigma_{i,j} a_{ij} + \sigma_{n+1,i} \sigma_{j,n+m} \sigma_{i,j} b_{i-n,j-n}]_{i,j}^{m+n,m+n} \\ [\Gamma_1 T_1 + \Gamma_2 T_2]_{i,j}^{m+n,m+n} \\ [M_{ij}]_{i,j}^{m+n,m+n} \end{cases} \quad (1)$$

Here $[M_{ij}]_{i,j}^{m+n,m+n}$ denotes a matrix of entries indexed by variables i, j each ranging from 1 to $m+n$. We write regions as $\Gamma_i T_i$, where Γ_i is a product of σ terms and T_i is a product of matrix entries. If $\Gamma = \sigma_1 \sigma_2 \dots \sigma_n$, then its complement is defined as $\overline{\Gamma} = \overline{\sigma_1 \sigma_2 \dots \sigma_n} = \overline{\sigma_1} + \overline{\sigma_2} \sigma_1 + \overline{\sigma_3} \sigma_1 \sigma_2 + \dots + \overline{\sigma_n} \sigma_1 \sigma_2 \dots \sigma_{n-1}$

Abstract Matrix Addition

The reason why we can not analyse the sum of two abstract matrices immediately is that, while the regions of an individual matrix are disjoint, pairs of regions drawn from different matrices are not so in general. We overcome this by decomposing the regions of each matrix into compatible disjoint regions during the addition.

$$A = [(\Gamma_1^A T_1^A + \dots + \Gamma_1^A T_R^A)]_{i,j}^{m,n} \quad B = [(\Gamma_1^B T_1^B + \dots + \Gamma_S^B T_S^B)]_{i,j}^{m,n} \quad (2)$$

where Γ_i^A, Γ_i^B indicates that the respective Γ term originates from matrix A or B , respectively. We define the abstract matrix addition algorithm ADD as

$$\text{ADD}(A, B) := \left[\begin{array}{l} \Gamma_1^A \Gamma_1^B (T_1^A + T_1^B) + \Gamma_1^A \Gamma_2^B (T_1^A + T_2^B) + \dots + \Gamma_R^A \Gamma_S^B (T_R^A + T_S^B) + \\ \Gamma_1^A \overline{\Gamma_1^B} \dots \overline{\Gamma_S^B} T_1^A + \Gamma_2^A \overline{\Gamma_1^B} \dots \overline{\Gamma_S^B} T_2^A + \dots + \Gamma_R^A \overline{\Gamma_1^B} \dots \overline{\Gamma_S^B} T_R^A + \\ \Gamma_1^B \overline{\Gamma_1^A} \dots \overline{\Gamma_R^A} T_1^B + \Gamma_2^B \overline{\Gamma_1^A} \dots \overline{\Gamma_R^A} T_2^B + \dots + \Gamma_S^B \overline{\Gamma_1^A} \dots \overline{\Gamma_R^A} T_S^B \end{array} \right]_{i,j}^{m,n} \quad (3)$$

Abstract Matrix Multiplication

Our multiplication algorithm 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. Letting A, B be as in (2), we define the abstract matrix multiplication algorithm MULT as:

$$\text{MULT}(A, B) := \left[\sum_{k=1}^p A_{i,k} B_{k,j} \right]_{i,j}^{m,n} = \left[\sum_{k=1}^p \left(\Gamma_1^A \Gamma_1^B T_1^A T_1^B + \Gamma_1^A \Gamma_2^B T_1^A T_2^B + \cdots + \Gamma_R^A \Gamma_S^B T_R^A T_S^B \right) \right]_{i,j}^{m,n} \quad (4)$$

$$= \left[\sum_{k=1}^p \left(\widehat{\sum}_{I \subseteq \mathcal{N}} \left(\widehat{\prod}_{q \in I} \Gamma_q \right) \left(\widehat{\prod}_{q \in \mathcal{N} \setminus I} \overline{\Gamma}_q \right) \left(\widehat{\sum}_{q \in I} T_q \right) \right) \right]_{i,j}^{m,n} = \left[\sum_{k=1}^p (\Gamma_1 T_1) + \sum_{k=1}^p (\Gamma_2 T_2) + \cdots + \sum_{k=1}^p (\Gamma_N T_N) \right]_{i,j}^{m,n} \quad (5)$$

Observe that in the formula above the sum symbol Σ is part of the term syntax, i.e., the actual terms in the matrix contain a syntactic summation. $\widehat{\Sigma}$ and $\widehat{\Pi}$ on the other hand meta-syntactic summation and product operators, which we use for the simplification of (4).

In (4) the distinction between terms originating from A and B is no longer necessary, because the σ terms involving the k index variable no longer describe half-plane constraints. The resulting Γ_i regions are still convex but also contain overlap. 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. Thus a single component region will have the form $\Gamma_{s_1} \dots \Gamma_{s_m} \overline{\Gamma_{s_{m+1}}} \dots \overline{\Gamma_{s_N}}$, where S is an indexing of $\{1, \dots, N\}$. This process is described by the formula on the left hand side of (5), while the result on the right hand side is the fully expanded version.

Normalisation

After addition and multiplication algorithm the resulting abstract matrices are generally no longer in a form from which we can easily infer the actual form and approximate content of the single regions. We therefore define a rewrite system that ensure to produce a desired *regular form* of an abstract matrix. We thereby define the *regular form* of $[M]_{i,j}^{m,n} = [\Gamma_1 T_1 + \cdots + \Gamma_k T_k]_{i,j}^{n,m}$ by requiring that the following holds:

- (i) $\Gamma_l \Gamma_{l'} = 0$ for each l, l' in $1, \dots, k$ with $l \neq l'$. (Disjointness)
- (ii) $\Gamma_l = \sigma_1 \cdots \sigma_p$, for each l in $1, \dots, k$ and $p \geq 0$. (Convexity)
- (iii) for each T_l, l in $1, \dots, k$ there does not exist a $\sigma \in T_l$ such that $\sigma T_l = T_l$. (Γ -Structured)

The rewrite system consists of four sets of rules that are applied in three different stages:

Stage 1: 2 Transitive Closure rules complete the partial order in a given Γ expression.

Stage 2: 2 Factoring rules pull single σ s that are independent from a summation variable in front of a sum expression.

Stage 3: 2 Reduction rules reduce 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, 7 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. We can then show for the combined system $\mathcal{R} = (\text{ADD}, \text{MULT}, \text{NORM})$ that it terminates, is confluent and correct.

References

- [1] A. P. Sexton and V. Sorge. Abstract matrices in symbolic computation. In *Proc. of ISSAC 2006*, p. 318–325. ACM Press, 2006.
- [2] A. P. Sexton, V. Sorge, S. M. Watt. Abstract matrix arithmetic. *ORCCA Technical Report TR-08-01*, University of Western Ontario, 2008.