Computing with Abstract Matrix Structures

Alan P. Sexton, Volker Sorge
University of Birmingham
UK

Stephen M. Watt
University of Western Ontario
Canada

Motivation

What is the shape of the results of the following expressions:

$$\begin{bmatrix} a & \cdots & a \\ & \ddots & \vdots \\ \mathbf{0} & & a \end{bmatrix} \begin{bmatrix} b & \cdots & b \\ & \ddots & \vdots \\ \mathbf{0} & & b \end{bmatrix} = \quad ?$$

Motivation

What is the shape of the results of the following expressions:

$$\begin{bmatrix} a & \cdots & a \\ & \ddots & \vdots \\ \mathbf{0} & & a \end{bmatrix} \begin{bmatrix} b & \cdots & b \\ & \ddots & \vdots \\ \mathbf{0} & & b \end{bmatrix} = \begin{bmatrix} [a, b] & \cdots & [a, b] \\ & \ddots & \vdots \\ \mathbf{0} & & [a, b] \end{bmatrix}$$

Motivation

What is the shape of the results of the following expressions:

$$\begin{bmatrix} a & \cdots & a \\ & \ddots & \vdots \\ \mathbf{0} & & a \end{bmatrix} \begin{bmatrix} b & \cdots & b \\ & \ddots & \vdots \\ \mathbf{0} & & b \end{bmatrix} = \begin{bmatrix} [a, b] & \cdots & [a, b] \\ & \ddots & \vdots \\ \mathbf{0} & & [a, b] \end{bmatrix}$$

$$\begin{bmatrix} a & \cdots & a & & & \mathbf{0} \\ & \ddots & \vdots & & & \\ & & a & & & \\ & & & b & \cdots & b \\ & & & & \ddots & \vdots \\ \mathbf{0} & & & & & b \end{bmatrix}^2 = \quad ?$$

Motivation

What is the shape of the results of the following expressions:

$$\begin{bmatrix} a & \cdots & a \\ & \ddots & \vdots \\ \mathbf{0} & & a \end{bmatrix} \begin{bmatrix} b & \cdots & b \\ & \ddots & \vdots \\ \mathbf{0} & & b \end{bmatrix} = \begin{bmatrix} [a, b] & \cdots & [a, b] \\ & \ddots & \vdots \\ \mathbf{0} & & [a, b] \end{bmatrix}$$

$$\begin{bmatrix} a & \cdots & a & & & \mathbf{0} \\ & \ddots & \vdots & & & \\ & & a & & & \\ & & & b & \cdots & b \\ & & & & \ddots & \vdots \\ \mathbf{0} & & & & & b \end{bmatrix}^2 = \begin{bmatrix} [a] & \cdots & [a] & & & \mathbf{0} \\ & \ddots & \vdots & & & \\ & & [a] & & & \\ & & & [b] & \cdots & [b] \\ & & & & \ddots & \vdots \\ \mathbf{0} & & & & & [b] \end{bmatrix}$$

Motivation

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

$$=?$$

Motivation

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

$$\mathbf{m} = \mathbf{n} : \begin{bmatrix} [ac] & \cdots & [ac] & & \mathbf{0} & \\ & \ddots & \vdots & & & \\ & & [ac] & & & \\ & & & [bd] & \cdots & [bd] \\ & & & & \ddots & \vdots \\ \mathbf{0} & & & & & [bd] \end{bmatrix}$$

Motivation

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

$$
\mathbf{m} < \mathbf{n} : \quad
\begin{bmatrix}
[ac] & \cdots & [ac] & [ad] & \cdots & \cdots & \cdots & \cdots & [ad] \\
 & \ddots & \vdots & \vdots & & & & & \vdots \\
 & & [ac] & \vdots & & & & & \vdots \\
 & & & [ad] & & & & & \vdots \\
 & & & & \ddots & & & & \vdots \\
 & & & & & [ad] & \cdots & \cdots & [ad] \\
 & & & & & & [bd] & \cdots & [bd] \\
 & & & & & & & \ddots & \vdots \\
\mathbf{0} & & & & & & & & [bd]
\end{bmatrix}
$$

Motivation

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

$$\mathbf{n} < \mathbf{m} : \begin{bmatrix} [ac] & \cdots & \cdots & \cdots & [ac] & & & & \mathbf{0} \\ & \ddots & & & \vdots & & & & \\ & & [ac] & \cdots & [ac] & & & & \\ & & & [bc] & \cdots & [bc] & [bd] & \cdots & [bd] \\ & & & & \ddots & \vdots & \vdots & & \vdots \\ & & & & & [bc] & \vdots & & \vdots \\ & & & & & & [bd] & & \vdots \\ & & & & & & & \ddots & \vdots \\ \mathbf{0} & & & & & & & & [bd] \end{bmatrix}$$

Motivation

- *Abstract Matrices* are matrices that contain unspecified components, such as ellipses or symbolic dimension
- We developed a parsing algorithm to make them available as templates for symbolic computation [ISSAC 06]
- This talk: doing arithmetic with them

Motivation

- *Abstract Matrices* are matrices that contain unspecified components, such as ellipses or symbolic dimension
- We developed a parsing algorithm to make them available as templates for symbolic computation [ISSAC 06]
- This talk: doing arithmetic with them
- Choice of internal representation makes naïve evaluation straightforward

Motivation

- *Abstract Matrices* are matrices that contain unspecified components, such as ellipses or symbolic dimension
- We developed a parsing algorithm to make them available as templates for symbolic computation [ISSAC 06]
- This talk: doing arithmetic with them
- Choice of internal representation makes naïve evaluation straightforward
- Challenge is to recover region shapes of result
  - Expose interactions between regions in the arithmetic
  - Support investigation of properties of shaped matrices

Motivation

- *Abstract Matrices* are matrices that contain unspecified components, such as ellipses or symbolic dimension
- We developed a parsing algorithm to make them available as templates for symbolic computation [ISSAC 06]
- This talk: doing arithmetic with them
- Choice of internal representation makes naïve evaluation straightforward
- Challenge is to recover region shapes of result
  - Expose interactions between regions in the arithmetic
  - Support investigation of properties of shaped matrices
- Compute with AMs, specialize results as unspecified elements become known

Abstract Matrix

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

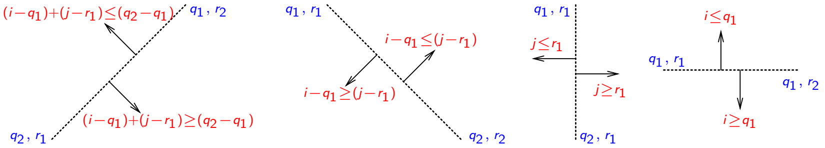
- Abstract matrices are composed of a set of regions
- Region shape is a closed convex polygonal area
- Region content is interpolated from a generalised term (a term with unification variables)

$$\begin{cases} a_{ij} & \text{if } i, j \in \{1 \ldots n\} \\ b_{(i-n),(j-n)} & \text{if } i, j \in \{n+1 \ldots n+m\} \\ 0 & \text{otherwise} \end{cases}$$

Half Plane Constraints

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

- Convex polygons represented as intersections of half planes
- Region shapes represented as conjunction of half plane constraints
- Concave regions have to be decomposed into convex regions



Support Function

- Use support (or characteristic) functions to represent half-plane contraints:

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

  We write $\sigma_{x,y}$ for convenience
- Conjunction of constraints $\equiv$ product of support functions
- Restrict terms to regions by multiplying by products of σs

Expressing Abstract Matrices in σs

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

- Every region can be expressed as a product of σs and its generalised term
- An Abstract Matrix is a sum of region terms

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

where i, j are the index variables

Expressing Abstract Matrices in σs

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

- Every region can be expressed as a product of σs and its generalised term
- An Abstract Matrix is a sum of region terms

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

Expressing Abstract Matrices in σs

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

- Every region can be expressed as a product of σs and its generalised term
- An Abstract Matrix is a sum of region terms

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

- Note: region structure can be read directly (in this case)

Some properties of σ

$$\sigma_{x,x} = 1 \tag{1}$$

$$\sigma_{x,y} = \sigma_{x+z,y+z} \tag{2}$$

$$\sigma_{x,y} = \sigma_{-y,-x} \tag{3}$$

$$\sigma_{x,y}\sigma_{y,x} = 1 \Leftrightarrow x = y \tag{4}$$

$$\sigma_{y+1,x}\sigma_{x,y} = 0 \tag{5}$$

Complement of a half-plain constraint:

$$\overline{\sigma_{x,y}} = \sigma_{y,x-1}$$

Complement of a region:

$$\overline{\sigma_1\sigma_2\ldots\sigma_n} = \overline{\sigma_1} + \overline{\sigma_2}\sigma_1 + \overline{\sigma_3}\sigma_1\sigma_2 + \cdots + \overline{\sigma_n}\sigma_1\sigma_2\ldots\sigma_n$$

Naïve Matrix Addition

- Abstract Matrix Addition is now simple (symbolic) addition

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

- Example:

$$U^T = \left[u_1, u_2, \ldots, u_h, u'_1, u'_2, \ldots, u'_{n-h}\right] = \left[\sigma_{j,h}\ u_j + \sigma_{h+1,j}\ u'_{j-h}\right]_{i,j}^{1,n}$$

$$V^T = \left[v_1, v_2, \ldots, v_k, v'_1, v'_2, \ldots, v'_{n-k}\right] = \left[\sigma_{j,k}\ v_j + \sigma_{k+1,j}\ v'_{j-k}\right]_{i,j}^{1,n}$$

$$U^T + V^T = \left[\sigma_{j,h}\ u_j + \sigma_{h+1,j}\ u'_{j-h} + \sigma_{j,k}\ v_j + \sigma_{k+1,j}\ v'_{j-k}\right]_{i,j}^{1,n}$$

Naïve Matrix Addition

- Abstract Matrix Addition is now simple (symbolic) addition

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

- Example:

$$U^T = \left[u_1, u_2, \ldots, u_h, u'_1, u'_2, \ldots, u'_{n-h}\right] = \left[\sigma_{j,h}\ u_j + \sigma_{h+1,j}\ u'_{j-h}\right]_{i,j}^{1,n}$$

$$V^T = \left[v_1, v_2, \ldots, v_k, v'_1, v'_2, \ldots, v'_{n-k}\right] = \left[\sigma_{j,k}\ v_j + \sigma_{k+1,j}\ v'_{j-k}\right]_{i,j}^{1,n}$$

$$U^T + V^T = \left[\sigma_{j,h}\ u_j + \sigma_{h+1,j}\ u'_{j-h} + \sigma_{j,k}\ v_j + \sigma_{k+1,j}\ v'_{j-k}\right]_{i,j}^{1,n}$$

- **BUT: region structure is lost** *(because summands no longer describe disjoint regions)*

Matrix Addition: Recovering Structure

- Decompose into set of convex region terms so that every pair is either disjoint or has the same shape and location

Matrix Addition: Recovering Structure

- Decompose into set of convex region terms so that every pair is either disjoint or has the same shape and location
  - use set theory (and region complement definition) to construct a disjoint cover, and project each region onto the elements of the cover

Matrix Addition: Recovering Structure

- Decompose into set of convex region terms so that every pair is either disjoint or has the same shape and location
  - use set theory (and region complement definition) to construct a disjoint cover, and project each region onto the elements of the cover
- Rewrite each region term into a simple structure so that the full region shape information is easily extractible from the term
- Remove superfluous regions
- Simplify the region description part of the term

Matrix Addition: Recovering Structure

- Decompose into set of convex region terms so that every pair is either disjoint or has the same shape and location
  - use set theory (and region complement definition) to construct a disjoint cover, and project each region onto the elements of the cover
- Rewrite each region term into a simple structure so that the full region shape information is easily extractible from the term
- Remove superfluous regions
- Simplify the region description part of the term

Matrix Multiplication

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

Note: Multiplication introduces a **syntactic** summation operator

- Example:

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

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

Matrix Multiplication Example

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

$$A = [\sigma_{j,n}\sigma_{i,j}\ a + \sigma_{n+1,i}\sigma_{i,j}\ b]_{i,j}^{n+m,n+m} \qquad B = [\sigma_{j,m}\sigma_{i,j}\ c + \sigma_{m+1,i}\sigma_{i,j}\ d]_{i,j}^{m+n,m+n}$$

$$AB = \left[\sum_{k=1}^{n+m} \begin{pmatrix} \sigma_{k,n}\sigma_{i,k}\sigma_{j,m}\sigma_{k,j} & ac+ \\ \sigma_{k,n}\sigma_{i,k}\sigma_{m+1,k}\sigma_{k,j} & ad+ \\ \sigma_{n+1,i}\sigma_{i,k}\sigma_{j,m}\sigma_{k,j} & bc+ \\ \sigma_{n+1,i}\sigma_{i,k}\sigma_{m+1,k}\sigma_{k,j} & bd \end{pmatrix} \right]_{i,j}^{n+m,n+m}$$

Key insight: σs involving the summation variable k do not represent half plane constraints

Matrix Multiplication Example (cont.)

$$= \begin{bmatrix} \sum_{k=1}^{n+m} \sigma_{k,n}\sigma_{i,k}\sigma_{j,m}\sigma_{k,j}\ ac + \sum_{k=1}^{n+m} \sigma_{k,n}\sigma_{i,k}\sigma_{m+1,k}\sigma_{k,j}\ ad + \\ \sum_{k=1}^{n+m} \sigma_{n+1,i}\sigma_{i,k}\sigma_{j,m}\sigma_{k,j}\ bc + \sum_{k=1}^{n+m} \sigma_{n+1,i}\sigma_{i,k}\sigma_{m+1,k}\sigma_{k,j}\ bd \end{bmatrix}_{i,j}^{n+m,n+m}$$

Matrix Multiplication Example (cont.)

$$= \begin{bmatrix} \displaystyle\sum_{k=1}^{n+m} \sigma_{k,n}\sigma_{i,k}\sigma_{j,m}\sigma_{k,j}\ ac + \sum_{k=1}^{n+m} \sigma_{k,n}\sigma_{i,k}\sigma_{m+1,k}\sigma_{k,j}\ ad + \\ \displaystyle\sum_{k=1}^{n+m} \sigma_{n+1,i}\sigma_{i,k}\sigma_{j,m}\sigma_{k,j}\ bc + \sum_{k=1}^{n+m} \sigma_{n+1,i}\sigma_{i,k}\sigma_{m+1,k}\sigma_{k,j}\ bd \end{bmatrix}_{i,j}^{n+m,n+m}$$

Matrix Multiplication Example (cont.)

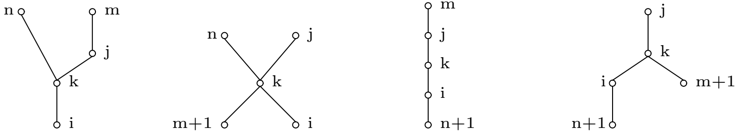
$$= \begin{bmatrix} \sum_{k=1}^{n+m} \sigma_{k,n}\sigma_{i,k}\sigma_{j,m}\sigma_{k,j}\ ac + \sum_{k=1}^{n+m} \sigma_{k,n}\sigma_{i,k}\sigma_{m+1,k}\sigma_{k,j}\ ad + \\ \sum_{k=1}^{n+m} \sigma_{n+1,i}\sigma_{i,k}\sigma_{j,m}\sigma_{k,j}\ bc + \sum_{k=1}^{n+m} \sigma_{n+1,i}\sigma_{i,k}\sigma_{m+1,k}\sigma_{k,j}\ bd \end{bmatrix}_{i,j}^{n+m,n+m}$$

Matrix Multiplication Example (cont.)

$$= \begin{bmatrix} \displaystyle\sum_{k=1}^{n+m} \sigma_{k,n}\sigma_{i,k}\sigma_{j,m}\sigma_{k,j}\ ac + \sum_{k=1}^{n+m} \sigma_{k,n}\sigma_{i,k}\sigma_{m+1,k}\sigma_{k,j}\ ad + \\ \displaystyle\sum_{k=1}^{n+m} \sigma_{n+1,i}\sigma_{i,k}\sigma_{j,m}\sigma_{k,j}\ bc + \sum_{k=1}^{n+m} \sigma_{n+1,i}\sigma_{i,k}\sigma_{m+1,k}\sigma_{k,j}\ bd \end{bmatrix}_{i,j}^{n+m,n+m}$$

For each summand we extract the full partial order defined by the σ product, excluding terms involving k

ac *ad* *bc* *bd*



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

Multiplication Example (cont.)

$\sigma_{j,m}\sigma_{i,n}\sigma_{i,j}$	$\sigma_{i,j}\sigma_{i,n}\sigma_{m+1,j}$	$\sigma_{n+1,i}\sigma_{i,j}\sigma_{j,m}$	$\sigma_{n+1,i}\sigma_{i,j}\sigma_{m+1,j}$
$[ac]$	$[ad]$	$[bc]$	$[bd]$

Multiplication Example (cont.)

$$\begin{array}{cccc} \sigma_{j,m}\sigma_{i,n}\sigma_{i,j} & \sigma_{i,j}\sigma_{i,n}\sigma_{m+1,j} & \sigma_{n+1,i}\sigma_{i,j}\sigma_{j,m} & \sigma_{n+1,i}\sigma_{i,j}\sigma_{m+1,j} \\ [ac] & [ad] & [bc] & [bd] \end{array}$$

$$\mathbf{m} = \mathbf{n} : \begin{bmatrix} [ac] & \cdots & [ac] & & \mathbf{0} & \\ & \ddots & \vdots & & & \\ & & [ac] & & & \\ & & & [bd] & \cdots & [bd] \\ & & & & \ddots & \vdots \\ \mathbf{0} & & & & & [bd] \end{bmatrix}$$

Multiplication Example (cont.)

$\sigma_{j,m}\sigma_{i,n}\sigma_{i,j}$	$\sigma_{i,j}\sigma_{i,n}\sigma_{m+1,j}$	$\sigma_{n+1,i}\sigma_{i,j}\sigma_{j,m}$	$\sigma_{n+1,i}\sigma_{i,j}\sigma_{m+1,j}$
$[ac]$	$[ad]$	$[bc]$	$[bd]$

$$\mathbf{m} < \mathbf{n}: \quad \begin{bmatrix} [ac] & \cdots & [ac] & [ad] & \cdots & \cdots & \cdots & \cdots & [ad] \\ & \ddots & \vdots & \vdots & & & & & \vdots \\ & & [ac] & \vdots & & & & & \vdots \\ & & & [ad] & & & & & \vdots \\ & & & & \ddots & & & & \vdots \\ & & & & & [ad] & \cdots & \cdots & [ad] \\ & & & & & & [bd] & \cdots & [bd] \\ & & & & & & & \ddots & \vdots \\ \mathbf{0} & & & & & & & & [bd] \end{bmatrix}$$

Multiplication Example (cont.)

$$\begin{array}{cccc} \sigma_{j,m}\sigma_{i,n}\sigma_{i,j} & \sigma_{i,j}\sigma_{i,n}\sigma_{m+1,j} & \sigma_{n+1,i}\sigma_{i,j}\sigma_{j,m} & \sigma_{n+1,i}\sigma_{i,j}\sigma_{m+1,j} \\ [ac] & [ad] & [bc] & [bd] \end{array}$$

$$\mathbf{n} < \mathbf{m} : \quad \begin{bmatrix} [ac] & \cdots & \cdots & \cdots & [ac] & & & & \mathbf{0} \\ & \ddots & & & \vdots & & & & \\ & & [ac] & \cdots & [ac] & & & & \\ & & & [bc] & \cdots & [bc] & [bd] & \cdots & [bd] \\ & & & & \ddots & \vdots & \vdots & & \vdots \\ & & & & & [bc] & \vdots & & \vdots \\ & & & & & & [bd] & & \vdots \\ & & & & & & & \ddots & \vdots \\ \mathbf{0} & & & & & & & & [bd] \end{bmatrix}$$

Grammar

$$
\begin{aligned}
M &::= [F]^{\text{iexp},\text{iexp}}_{\text{var},\text{var}} & \Gamma &::= \sigma_{\text{iexp},\text{iexp}}\,\Gamma \mid \epsilon \\
F &::= R \mid R + F & T &::= \text{exp} \mid \sum_{\text{var}=1}^{\text{iexp}} (F) \mid F \\
R &::= \Gamma\ T \mid T
\end{aligned}
$$

$$\text{iexp} ::= \text{var} \mid \text{int} \mid -\text{iexp} \mid \text{iexp} + \text{iexp} \mid \text{iexp} - \text{iexp}$$

- **var:** a single variable name
- **int:** a single, non-negative integer
- **exp:** an arbitrary functional expression, that can contain var and int terms.

The term for an Abstract Matrix thus has the form:

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

Regular Form

We define a set of formal properties that the Abstract Matrix term must have so that it the region structure is easily extractible from it, the term is appropriately minimal and it can be used as input arguments to further Abstract Matrix operations, hence ensuring closure in the arithmetic:

- **Disjoint**
- **Convex**
- Γ-**Partitioned**
- **Non-Empty**
- Γ-**Minimal**

These properties are formally specified in terms of the grammar and equations in the *sigma* functions.

Algorithms

ADD: Directly construct non-regularised addition result term from input Abstract Matrices

- Result is Disjoint, Convex and Γ-Partitioned, but not necessarily Non-Empty or Γ-Minimal

MULT: Directly construct non-regularised multiplication result term from input Abstract Matrices

- Result is Disjoint and Convex, but not necessarily Γ-Partitioned, Non-Empty or Γ-Minimal

NORM: Rewrite an Abstract Matrix term that is Disjoint and Convex into a Regular term

NORM

A 3-stage rewrite system with 4 sets of rules:

Stage 1: Transitive Closure creates all possible σ expressions that can be read from the Hasse diagrams

Stage 2: Factoring factors single σs out of summations

Stage 3: Reduction reduces Γ expression by removing superflous σ expressions.

The contraction rules can be applied during each stage for simplification, but must always be exhaustively applied

We can prove correctness, termination and confluence

Conclusions

- Useful representation for symbolic representation of AMs
- Algorithms for addition and multiplication of AMs
- AMs are closed under these operations
- Operations propagate region structure information
- Proofs of correctness, termination and confluence
- Makes a contribution to a previously under-explored area of symbolic and algebraic computation