Arithmetic on Matrices with Blocks of Symbolic Size

Alan Sexton and Volker Sorge
University of Birmingham, UK

Stephen M. Watt
University of Western Ontario

I. Abstract Matrices

How to interpret matrices of the form:

$$\begin{bmatrix} a_1 & b & \cdots & b \\ & \ddots & \ddots & \vdots \\ & & \ddots & b \\ \mathbf{0} & & & a_n \end{bmatrix}$$

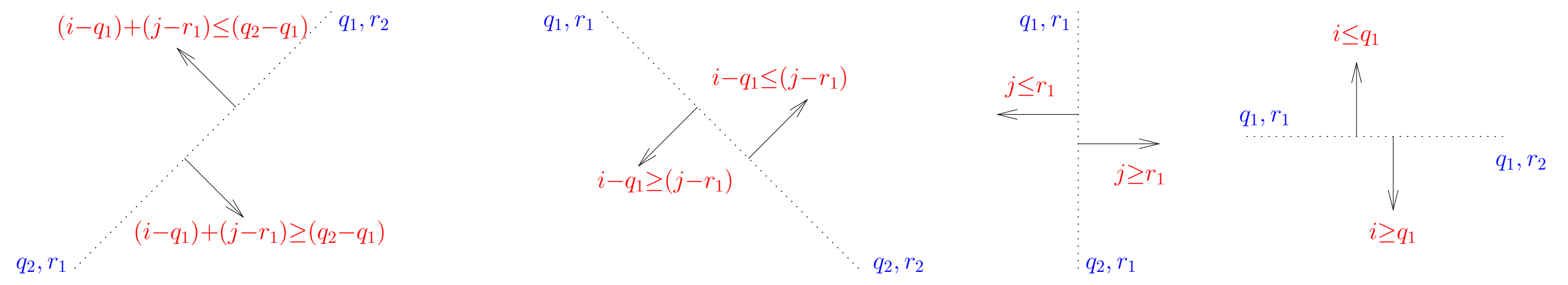
Interpretation as class of concrete matrices, e.g.:

$$\begin{bmatrix} a_1 \end{bmatrix} \begin{bmatrix} a_1 & b \\ 0 & a_2 \end{bmatrix} \begin{bmatrix} a_1 & b & b \\ 0 & a_2 & b \\ 0 & 0 & a_3 \end{bmatrix} \begin{bmatrix} a_1 & b & b & b \\ 0 & a_2 & b & b \\ 0 & 0 & a_3 & b \\ 0 & 0 & 0 & a_4 \end{bmatrix} \quad \begin{bmatrix} a_1 & b & b \\ 0 & a_0 & b \\ 0 & 0 & a_{-1} \end{bmatrix} \begin{bmatrix} a_1 & b & b \\ 0 & a_1 & b \\ 0 & 0 & a_1 \end{bmatrix} \begin{bmatrix} a_1 & b & b & b \\ 0 & a_1 & b & b \\ 0 & 0 & a_1 & b \\ 0 & 0 & 0 & a_1 \end{bmatrix}$$

⇒ Capture semantics as a data type of abstract matrices

III. Symbolic Representation

- Interpolation Function and Generalised Term gives region contents
- Multiply by Basis Function to capture containment in a region.
- Basis functions derived from conjunction of equations of the half-planes defining a convex polygon.



- Result is a Symbolic Term for a region.
- Add symbolic terms for all regions to get symbolic term for matrix

$$\begin{bmatrix} a_{1,1} & \cdots & a_{1,m} & \\ \vdots & \iddots & & \\ a_{n,1} & & & \end{bmatrix} \qquad \begin{aligned} &\text{basis}_1(i,j) \times a\left(\frac{in-i-n+q}{q-1}, \frac{jm-j-m+q}{q-1}\right) \\ &+\text{basis}_2(i,j) \times \ldots \end{aligned}$$

VI. Interval Signed Basis Function Primitives

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

$$\text{basis}_1(i,j) = \xi(i,1,q-j+2) \times \xi(j,1,q-i+2)$$

Algebraic identities: $\begin{cases} \xi(i,y,z) = -\xi(i,z,y) \\ \xi(i,y,x) + \xi(i,x,z) = \xi(i,y,z) \end{cases}$

$$U = \left[u_1, \ldots, u_{h-1}, u'_1, \ldots, u'_{n-h}\right], \quad V = \left[v_1, \ldots, v_{k-1}, v'_1, \ldots, v'_{n-k}\right]$$

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

(*) reduces to three cases:

$$\begin{array}{r|r|r} (1) \quad h<k & (2) \quad h=k & (3) \quad h>k \\ \xi(i,1,h) \times (u_i+v_i) & & \xi(i,1,k) \times (u_i+v_i) \\ +\xi(i,h,k) \times (u'_i+v_i) & \xi(i,1,h) \times (u_i+v_i) & +\xi(i,k,h) \times (u_i+v'_i) \\ +\xi(i,k,n) \times (u'_i+v'_i) & +\xi(i,h,n) \times (u'_i+v'_i) & +\xi(i,h,n) \times (u'_i+v'_i) \end{array}$$

Apply algebraic identities to (1):

$$(1) = \begin{array}{r} (\xi(i,1,k)+\xi(i,k,h)) \times (u_i+v_i) \\ -\xi(i,k,h) \times (u'_i+v_i) \\ +(\xi(i,k,h)+\xi(i,h,n)) \times (u'_i+v'_i) \end{array} = \begin{array}{r} \xi(i,1,k) \times (u_i+v_i) \\ +\xi(i,k,h) \times (u_i+v'_i) \\ +\xi(i,h,n) \times (u'_i+v'_i) \end{array} = (3)$$

Furthermore (1) and (3) both reduce to (2) if $h = k$.

Thus $(*) = (1) = (3)$ for all cases: only need to process 1 case

Sum of m vectors like U or V $\implies$ $m+1$ summands

II. Regions

- Region: closed polyline containing a homogeneous pattern of terms
- Generalised Term: found by anti-unification over concrete terms on region boundary to capture pattern of terms in region
- Interpolation Function: computes the concrete instantiation of unification variables in generalised terms
- Region locations change as they expand or shrink
- Generalised Position: coordinates are integer expressions in constants and constraint variables (e.g. $(0,0)$, $(0,q)$, $(q,0)$)

$$\begin{matrix} a_{1,1} & \cdots & a_{1,m} \\ \vdots & \iddots & \\ a_{n,1} & & \end{matrix} \quad \longrightarrow \quad a_{\alpha,\beta} \; \begin{matrix} [1,1] & \cdots & [1,m] \\ \vdots & \iddots & \\ [n,1] & & \end{matrix} \qquad \begin{aligned} \alpha &= \frac{in-i-n+q}{q-1} \\ \beta &= \frac{jm-j-m+q}{q-1} \end{aligned}$$

IV. Matrix Arithmetic

Most of the complexity is in the underlying Symbolic Term for the matrices. Matrix operations are standard:

- Addition: $(A+B)_{i,j} = A_{i,j} + B_{i,j}$
- Multiplication: $(A \times B)_{i,j} = \sum_{k=1}^{n} A_{i,k} \times B_{k,j}$ where n is the width of A (or height of B) and will normally be a symbolic term.

Challenge: avoid combinatorial explosion of terms

V. Simple Basis Function Primitives

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

$$\text{basis}_1(i,j) = \sigma(1,j-1) \times \sigma(1,i-1) \times \sigma(i+j-2,q-1)$$

Lack of useful algebraic identities means hard to manipulate (no transitivity, no anti-symmetry)

$$U = \left[u_1, \ldots, u_h, u'_1, \ldots, u'_{n-h}\right], \quad V = \left[v_1, \ldots, v_k, v'_1, \ldots, v'_{n-k}\right]$$

$$\begin{aligned} (U+V)_i = U_i + V_i = \quad & \sigma(i,h) \times u_i + \sigma(h+1,i) \times u'_{i-h} \\ & + \sigma(i,k) \times v_i + \sigma(k+1,i) \times v'_{i-k} \\ = \quad & \sigma(i,min(h,k)) \times (u_i+v_i) \\ & + \sigma(h+1,i) \times \sigma(i,k) \times (u'_i+v_i) \\ & + \sigma(k+1,i) \times \sigma(i,h) \times (u_i+v'_i) \\ & + \sigma(max(h,k)+1,i) \times \sigma(i,n) \times (u'_i+v'_i) \\ = \quad & \sigma(i,h) \times \sigma(i,k) \times (u_i+v_i) \\ & + \sigma(h+1,i) \times \sigma(i,k) \times (u'_i+v_i) \\ & + \sigma(k+1,i) \times \sigma(i,h) \times (u_i+v'_i) \\ & + \sigma(h+1,i) \times \sigma(k+1,i) \times \sigma(i,n) \times (u'_i+v'_i) \end{aligned}$$

Sum of m vectors like U or V $\implies$ 2^m summands

VII. Conclusion

- Symbolic matrix block representation by index intervals with negative lengths can treat all possible orderings of symbolic positions simultaneously, leading to elegant symbolic matrix computation algorithms.
- This technique is applicable over any free module with an ordered basis and ranges of terms being given by formulae. It thus can be applied to matrices and polynomials as well.

A. Sexton and V. Sorge, "Processing Textbook-style Matrices", MKM'05, pp 111–125

A. Sexton and V. Sorge, "Abstract Matrices in Symbolic Computation", ISSAC'06, pp 318–325

S. Watt, "Making Computer Algebra More Symbolic", Transgressive Computing'06, pp. 43–49

S. Watt, "Two Families of Algorithms for Symbolic Polynomials", Computer Algebra 2006, pp 193–219