

Reasoning with Generic Cases in the Arithmetic of Abstract Matrices

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

¹ School of Computer Science, University of Birmingham
www.cs.bham.ac.uk/~aps|~vxs

² Department 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 [8,9,10] and abstract matrices [5,6,7]. 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