Directions in Recognizing Tabular Structures of Handwritten Mathematics Notation

Richard Zanibbi, Dorothea Blostein, and James R. Cordy
Department of Computing and Information Science,
Queen's University, Kingston, Ontario, Canada
{zanibbi,blostein,cordy}@cs.queensu.ca

Abstract

Mathematics notation contains a variety of tabular structures, including matrices, lists of expressions in a derivation, stacked limit expressions in a summation, and conditional expressions in a function definition. Tabular structures are particularly difficult to recognize in handwritten mathematics notation, due to irregular placement and sizing of symbols. In this paper we briefly survey prior work, describe a representation for tabular structures in mathematics notation, and present possible directions for recognizing tabular structures in handwritten notation.

1 Introduction

Tabular structures are a prominent part of the syntax of mathematics notation (see Figure 1). At the highest level, a proof may be viewed as a table of expressions with a single column. At the level of individual expressions, tables are used in matrices, for multiple indexes (e.g. Figure 1b), and for listing alternatives (e.g. Figure 1d). Matrices are probably the most complex tabular structure used in mathematics notation, as they may have irregular row and/or column structure (e.g. due to shorthand symbols, as in Figure 1e). In this paper we describe some preliminary work into creating general, robust recognition methods for the tabular structures in mathematics notation.

Our work is motivated by a desire to extend our existing prototype for recognizing handwritten mathematics notation [9, 10] to handle tabular structures (particularly matrices and proofs). A logical first step is to examine the table recognition literature. One key point seems to be that it is important to separate the physical structure of a table (e.g. layout and appearance) from its logical structure, which specifies the type and contents of each of the table's elements [5]. In our math recognition system we first describe symbol layout independently

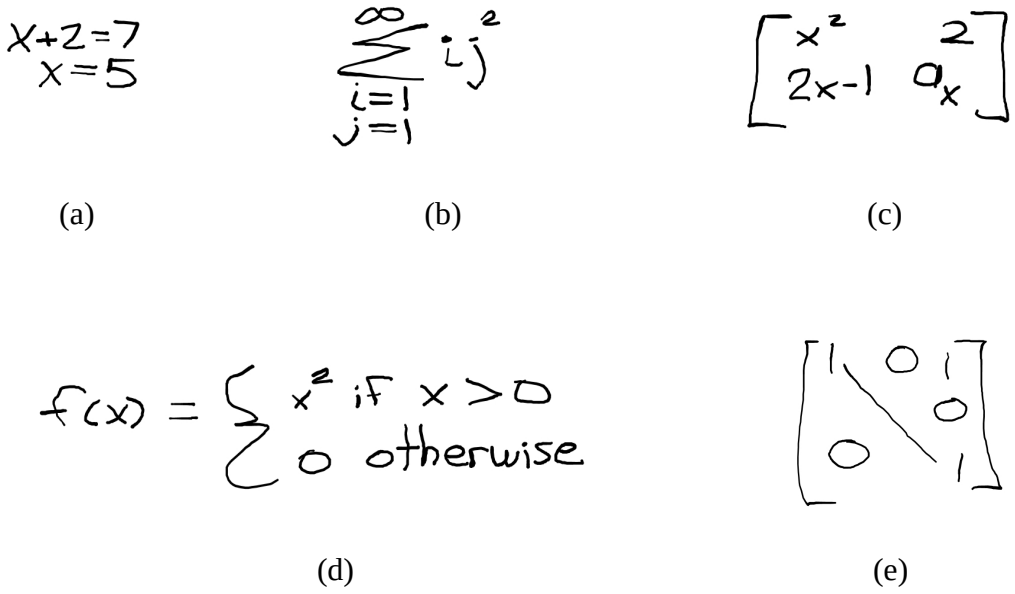


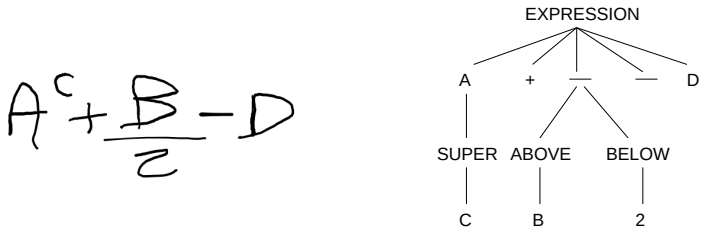
Figure 1: Examples of expressions with tabular structures.

of the syntax and semantics of an expression using *Baseline Structure Trees* (see Figure 2); we have found that this increases the extensibility of our system. A Baseline Structure Tree (BST) describes the hierarchical nesting of *baselines* in an expression, where we define a baseline as a horizontal list of symbols that are intended to be adjacent. The baseline below the root of a BST is the *dominant* baseline of the expression, from which interpretation begins. In section 3 we present a simple extension to Baseline Structure Trees which allows them to describe tabular structures.

Although table recognition has been studied extensively, little work has targeted the recognition of handwritten tables [5]. The symbols and layout of handwritten tables are of course far more irregular than in typeset tables. We summarize existing research into recognizing tabular structures in math notation in Section 2, and in Section 4 describe research directions that we are considering for recognizing tabular structures in mathematics notation after symbols have been correctly recognized.

2 Previous Work in Recognizing Tabular Structures

Up to the present, only a small amount of work has been devoted to recognizing tabular structures in mathematics notation [2, 3]. Impressive early work was done by Anderson in the late 1960's [1]. Anderson described the syntax of a subset of mathematics notation using a *coordinate grammar* (an attributed context-free grammar with constraints on attributes).



(a) (b)

Figure 2: Baseline Structure Tree (b) for a simple expression (a). The dominant baseline of the expression is (A, +, -, -, D). There are three nested baselines relative to the symbols of the dominant baseline, each comprised of a single symbol: the superscripted C, and the B and 2 above and below the fraction line.

The grammar is capable of describing row vectors and square matrices, and can handle line shorthands used to indicate repeated elements. The grammar is limited in that matrix elements have to be single symbols, and restrictive assumptions are made about symbol placement.

Roughly twenty years later, Okamoto and Twaakyondo addressed recognition of matrices from scanned images of typeset mathematics [6, 7, 8]. They use recursive projection profile cutting to obtain expression structure and connected components *before* recognizing symbols. In this approach, a tree describing expression structure is constructed as pixels are recursively projected onto the x and then y axes. After each projection, groups of pixels separated by whitespace on the projection axis are split into separate components, and projections are then applied recursively to each of the components until no further segmentation is possible. Symbol recognition is then applied to the indivisible components, which are assumed to be symbols. If a pair of delimiters (e.g. brackets) is detected, then horizontal projections are used to estimate the number and height of rows of the potential matrix. If horizontal projection reveals two or more rows of similar height, then vertical projection is used to locate the number of columns in the matrix and the regions of each matrix element. Recursive projection profile cutting is then re-applied to the matrix element regions, the results of which are associated with the delimiters row-by-row. This projection profile approach is also capable of recognizing multiple indexes, as in Figure 1b. However, this approach is limited to non-skewed typeset notation, where the necessary whitespace information used for segmentation after each projection may be obtained reliably.

Fateman et. al. [4] created a mathematics recognition system that is capable of recognizing tables of integrals. It locates vertical white-space between expressions and uses contextual clues to locate expressions. Contextual clues used by the system include the facts that in the table of integrals to be recognized, all expressions begin with an integral on the left, while any continuations of an expression begin with an operator on the left.

3 Representing Tabular Structures

A key component in our prior research on mathematics notation recognition is a simple, readable data structure for describing the layout of symbols in a mathematical expression called a *Baseline Structure Tree* (see Section 1 and Figure 2). Baseline Structure Trees aid understanding of recognition results, and can be used with different syntactic and semantic definitions to obtain multiple interpretations of the same layout description [10].

To describe tabular structures, we extend Baseline Structure Trees to have *TABLE*, *ROW* and element (written *ELEM*) nodes. To keep the trees small, a row containing a single element is represented using a single *ELEM* node, and a table containing a single row with a single element is also represented using a single *ELEM* node. A *TABLE* node has one or more child *ROW* nodes, each of which in turn have one or more *ELEM* nodes. *ROW* nodes are ordered left-to-right, corresponding to their appearance top-to-bottom. *ELEM* nodes are ordered left-to-right to reflect the left-to-right ordering of elements in a row.

Tables with unequal row or column lengths may be represented using empty elements, in effect giving a representation with regular row and column length. Line shorthands for repeated elements in matrices may also be represented, by treating these lines as spatial operators that divide an image into regions. For example, the diagonal line shorthand in Figure 1e may be represented as shown in Figure 3e. In Figure 3e the diagonal line has four child tables: these represent the matrix elements to the top-left and bottom-right of the line, and the upper and lower diagonal matrices (filled out with an empty element in the case of the upper diagonal).

To reflect the fact that we are now dealing with tables of expressions, the root of a Baseline Structure tree is labeled *TABLE* or *ELEM* (for a single expression, e.g. a table with a single element). As shown in Figure 3, this extended BST definition allows us to describe the symbol layout of the expressions in Figure 1.

4 Possible Approaches to Recognizing Tabular Structures

The projection-based of approach of Okamoto et. al. is an intuitive, appealing method for locating tabular structure in typeset notation, but is probably not a robust approach for messy handwritten expressions which may contain significant skew. As a first step we are interested in studying approaches to recognizing tabular structures after symbols have been recognized. Two approaches we are considering are the following:

1. Use a simple form of clustering. Assign "gravitational pulls" to classes of symbols, so that for instance $\sum$ attracts symbols more strongly than a digit. Then define a set of rules specifying how "gravity" segments symbols into table elements. The gravitational model might use trees or directed acyclic graphs to represent the relationships

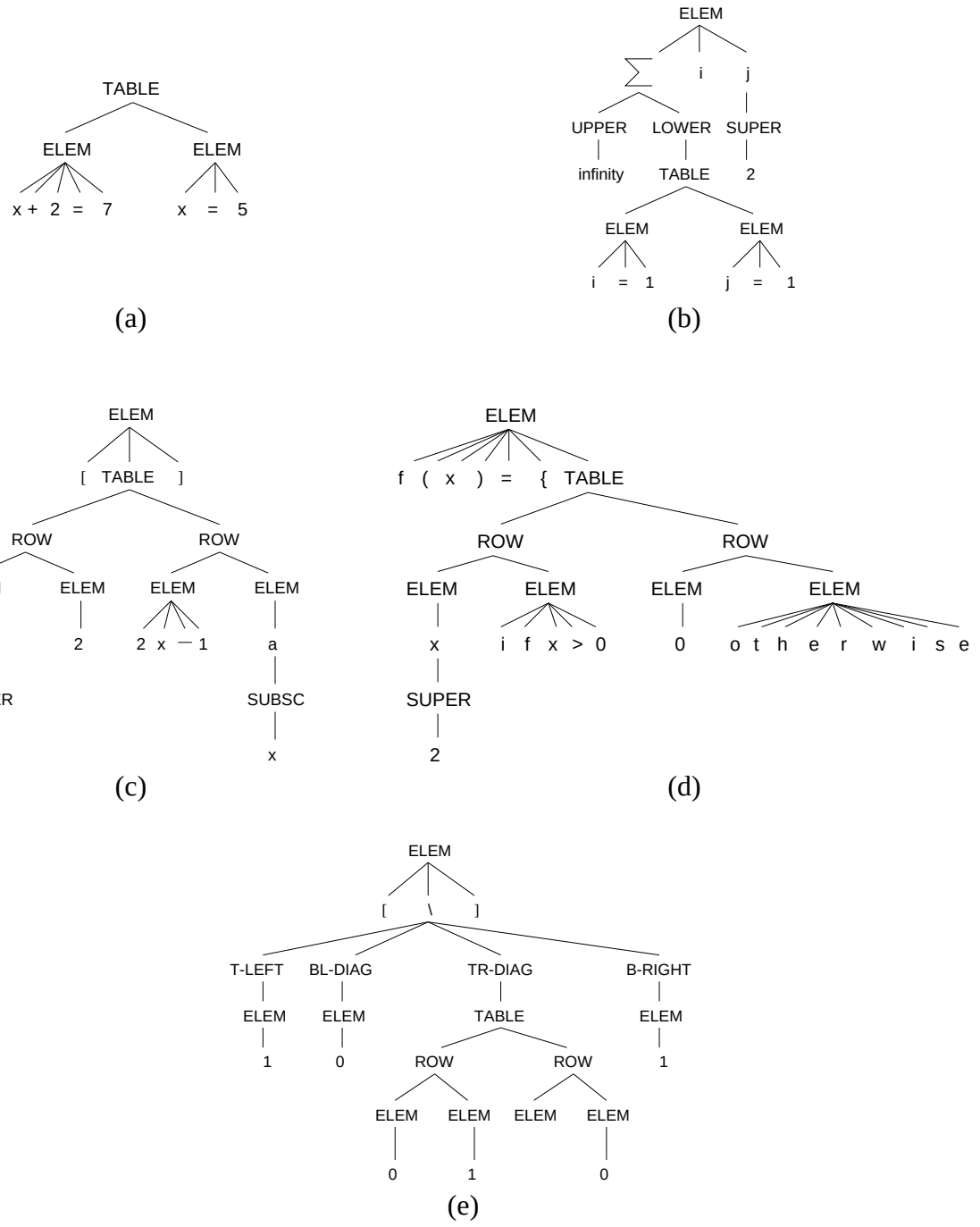


Figure 3: Baseline Structure Trees for the expressions in Figure 1. In (b), *UPPER* and *LOWER* indicate the upper and lower limit regions of the summation. In (e), *T-LEFT, B-LEFT, TR-DIAG* and *BL-DIAG* represent top-left, bottom-left, top-right diagonal and bottom-left diagonal, respectively.

between symbols. Tree or graph transformation systems may be used to build an initial gravitational model and then apply rules to transform the initial model into a table.

2. Use a process similar to error correcting parsing. First use an existing single-expression recognition method to analyze symbol layout. Then search for structures that are unsyntactic in the context of a single expression (e.g. two or more digits directly above one another). If an unsyntactic structure is located, perform a local whitespace analysis and split the expression into table elements.

In each of these approaches stochastic methods could be employed. The gravitational model could be stochastic. In an error-correcting parsing approach a statistical weighting could be used determine the "best" way to split an expression into table elements. Either approach could be altered to provide a list of alternatives ranked by probability (e.g. an "n-best" list).

In both approaches, reading direction could be exploited to reduce computational complexity. Gravitational pulls for instance could act only in top-down and left-to-right directions, and any rules involving a search could use these directions. Similarly, in the error-correcting parse scenario we could search for unsyntactic structures left-to-right. Both processes are likely to be recursive, to take advantage of the recursive structure of mathematics notation [2].

Currently we are leaning towards the gravitational model, as at the outset it seems simpler. It may also be more general than the error-correcting parse approach by being less sensitive to dialect-dependent structures; some structures that are legal in one variant of mathematics notation may be illegal in another. If use of whitespace and other cues for tabular structure are reasonably consistent across dialects of mathematics notation, a general "gravitational model" may be adequate; we do not yet know whether this is the case.

5 Conclusion

We have presented a simple extension of Baseline Structure Trees [10] in order to represent tabular structures in mathematics such as matrices and lists of expressions in a simple, consistent fashion. Additionally, we have described a pair of approaches we are currently considering exploring for recognizing tabular structures in handwritten mathematics notation. The first uses "gravitational pulls" between symbols, modeling these as a directed graph or tree which is then transformed into a table of elements. The second is an error-correcting parse approach. We plan to explore the gravitational model first as it may provide a simple, general method for detecting tabular structures in handwritten mathematics notation, assuming that symbols have been recognized correctly.

References

[1] R.H. Anderson. Two-dimensional mathematical notation. In K.S. Fu, editor, *Syntactic Pattern Recognition*, pages 147–177. Springer-Verlag, New York, 1977.

[2] Dorothea Blostein and Ann Grbavec. Recognition of mathematical notation. In *Handbook of Character Recognition and Document Image Analysis*, pages 557–582. World Scientific Publishing Company, 1997.

[3] Kam-Fai Chan and Dit-Yan Yeung. Mathematical expression recognition: a survey. *International Journal on Document Analysis and Recognition*, 3(1):3–15, August 2000.

[4] Richard J. Fateman and Taku Tokuyasu. Progress in recognizing typeset mathematics. In *Proceedings of the International Society for Optical Engineering*, volume 2660, pages 37–50, 1996.

[5] Daniel Lopresti and George Nagy. A tabular survey of automated table processing. In A. Chhabra and D. Dori, editors, *Graphics Recognition – Recent Advances*, number 1941 in LNCS, pages 93–120. Springer-Verlag, 2000.

[6] Masayuki Okamoto and Bin Miao. Recognition of mathematical expressions by using the layout structures of symbols. In *Proc. First International Conference on Document Analysis and Recognition*, volume 1, pages 242–250, Saint-Malo, France, 1991.

[7] Masayuki Okamoto and Akira Miyazawa. An experimental implementation of a document recognition system for papers containing mathematical expressions. In H.S. Baird H. Bunke and K. Yamamoto, editors, *Structured Document Image Analysis*, pages 36–53. Springer-Verlag, New York, 1992.

[8] Hashim M. Twaakyondo and Masayuki Okamoto. Structure analysis and recognition of mathematical expressions. In *Proceedings of the Third International Conference on Document Analysis and Recognition*, volume 1, Montréal, Canada, 1995.

[9] Richard Zanibbi. Recognition of mathematics notation via computer using baseline structure. Technical Report ISBN-0836-0227-2000-439, Department of Comptuer Science, Queen's University, Kingston, Ontario, Canada, August 2000.

[10] Richard Zanibbi, Dorothea Blostein, and James Cordy. Baseline structure analysis of handwritten mathematics notation. To appear in *Proc. Sixth International Conference on Document Analysis and Recognition*, 2001.