Model-based analysis of printed tables.

Conference Paper · January 1995

DOI: 10.1109/ICDAR.1995.598979 · Source: DBLP

Mukkai S. Krishnamoorthy
Rensselaer Polytechnic Institute
168 PUBLICATIONS **1,896** CITATIONS

Model-based Analysis of Printed Tables

Edward A. Green* and Mukki S. Krishnamoorthy

Department of Computer Science
Rensselaer Polytechnic Institute, Troy, NY 12180
greene@cs.rpi.edu

We discuss our system of model-based analysis of printed tables. The goal of our system is to extract and associate parts of a table's image into related segments. For example, we can locate columns, rows, column and table headings of a table's image. The location of these segments are based on features of the table image and on a model of the table. This is a stepwise, top-down approach to table image analysis; thus, for example, the body of the table is located before rows or columns, or individual table cells. The algorithms we discuss involve the stepwise analysis of the image and the grouping of these segments into larger structures (columns, rows, etc.).

Keywords: tables, model-based analysis, top-down analysis.

1 Introduction

Tables may be printed in many different styles. Aside from the various devices used to distinguish one cell from another (white space, ruling lines of varying widths, for example) a table may have different orientations. For example, tables may be organized such that each row in the body represents one item, with the items' attributes specified in the columns. Another typical arrangement transposes this table, so that items are arranged in columns. A third common organization has each cell (rather than rows or columns of cells) representing an item (for example, the Periodic Chart of the Elements).[1] At the most abstract level, of course, a table is a relation. A system that attempts to understand a table is actually attempting to derive the relational information stored in the printed manifestation of the table.

Most previous systems for analyzing tables have concentrated on analyzing tables with some specific format or restricted set of formats.[1][2][3] In this paper we describe a system that can analyze a wide variety of printed table formats. The adaptability of this system is realized by a model of the table's organization.

Printed representations of relational data rely on several kinds of visual clues for imparting the table's logical structure to the reader. For example, ruling lines of various widths might indicate a grouping of consecutive items or attributes. A system reading a table must make deductions based on these visual devices before it is able to specify the relational organization of the table. Some of the visual clues used to logically organize the physically structured information in the table are:

. *ruling line locations and attributes (e.g., thickness)*
. *font characteristics of cell contents (e.g., bold-face type)*
. *the semantics of the cell text*
. *format characteristics of cell data (e.g., integer vs. real)*
. *the shape of the cells (i.e., the cells aspect ratio)*

The approach asserted by this research is that the table analysis task can be done in steps, from analyzing the table's image to the logical interpretation of the table. There are two phases. The first phase generates a set of markers from the table image which would serve to logically organize the space occupied by the table; in other words, the first phase segments the table image. This segmentation would be expressed in a manner to allow efficient extraction of the logical relationships represented by the table. The second phase takes the segmented table and extracts the logically related data.

This paper discusses the algorithms used in these two phases. Section 2 of this paper diagrams the overall system design. Section 3 describes cell labels. Cell labels are used in each cycle of the analysis and for matching cells with templates to extract logic. An algorithm is provided which generates a labeling of individual cells. Section 4 describes how logically related cells may be grouped using cell labels. Section 5 describes the components of a table model. Section 6 describes some experimental results and Section 7 concludes the paper.

2 System Design

The overall design of our system is diagramed in figure 1:

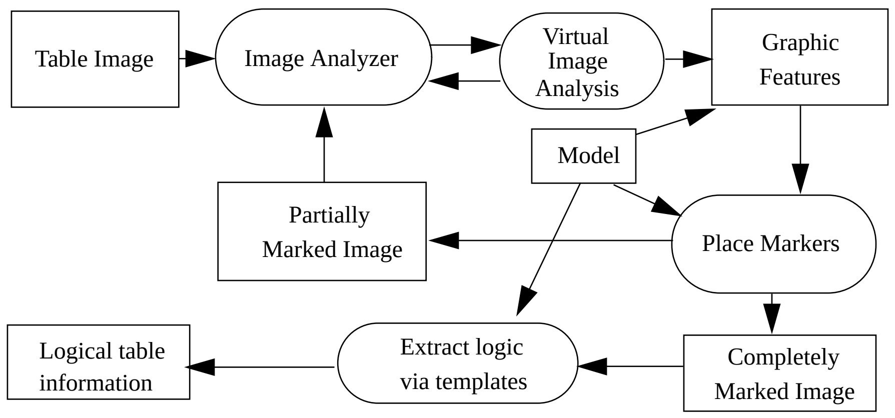


Figure 1: Table analysis system

The features of the table image are found by the image analyzer. The virtual image analysis provides a "front end" to the image analyzer: it translates commands to find features in specific areas and translates the results into a form that the rest of the system can use. Since the actual image analyzer retains independence from the rest of the system, new feature extraction programs can be substituted without affecting the rest of the system. The features are simply characterized by their bounding rectangle and a label: all separator structures will have rectangular shapes if the image is (as we are assuming) initially deskewed.

Currently, only features based on white space and solid ruling lines are found and used. This is sufficient for analyzing many types of tables. Attributes include the width of white space and whether ruling lines are single or double, etc. The image analysis is based on connected components and is susceptible to noise and skew.

The image analysis and virtual image analysis locate the graphic features which form the basis of segmenting or "marking" the table. Marking the table is a cyclic process; first the table is found on the page, then the table heading, column headings, and table body are segmented (for example), and finally individual cells are recognized. Each cycle requires that features be found on which to base the segmentation. The features which will be used at each cycle of the process and in each location of the table are specified in the model (the topic of section 5).

The final marked image is then examined for logical features (also specified in the model). Thus, the area of the table corresponding to (for example) "tuple 1" or "column B" can be recognized. Templates are provided by the model for this purpose. Thus, the model coordinates feature extraction and logical assignment of the resulting segments.

3 Characterizing Table Cells

The characterization of physical cells is hierarchical: at the top level, all table cells exist in the table. Individual cells belong to distinct (but possibly overlapping) logical classes: table header, column headings, and table body, for instance. The table body, in turn, would be characterized as collections of rows and columns; an individual cell might therefore belong to more than one class. The task of elementary cell characterization, then, is to label these cells in some way such that the cells belonging to the underlying nesting and overlapping of logical units can be properly extracted. In this section we first consider the one-dimensional analog of the problem, and then show a practical extension of this solution to two dimensions.

3.1 Cell Hierarchies

Consider a one dimensional, say vertical, profile of a table. The following diagram gives such a vertical profile, which is a sequence of cells and separators with corresponding weights.

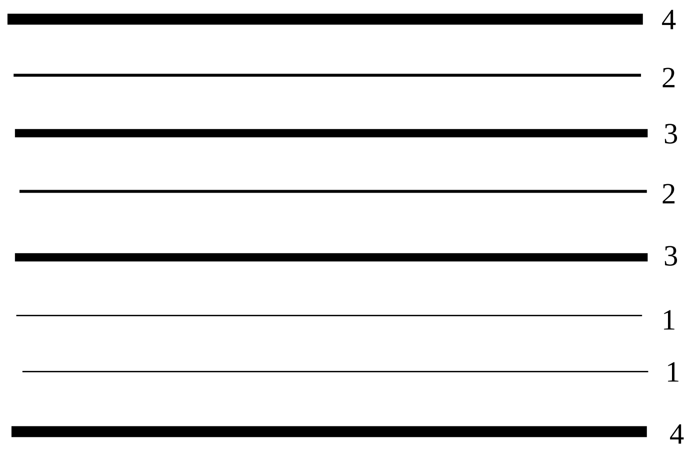


Figure 2: A one-dimensional cell structure

This sequence of cells (represented by c) and separators might be represented by the string:

4c2c3c2c3c1c1c4

or, if []=4, {}=3, ()=2, and ε=1, then the string is represented in a more familiar notation:

[c)(c}{c)(c}{ccc]

For a "proper" nesting, the brackets, braces, and parenthesis should be balanced, and each cell should be at the bottom-most level (within parenthesis), so the string should be converted to:

[{(c)(c)}{(c)(c)}{(ccc)}]

This is a straight-forward transformation in one dimension; since parenthesis, braces, and brackets are not nested in this formulation, it is a regular transformation. A regular grammar which generates this "balanced cell language" is:

S -> *[A*
A -> *{B*
B -> *(C*
C -> *cD*
D -> *cD* | *)E* | *)B*
E -> *}F* | *}A*
F -> *]*

The extension to two dimensions is straightforward. for example, consider figure 3.

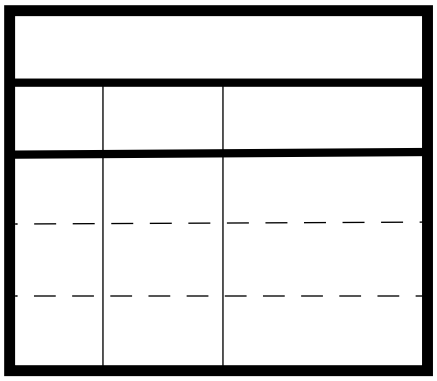


Figure 3: A two-dimensional cell structure (i.e., a table).

The natural approach for handling the two-dimensional case is to consider regions of cells in the two- dimensional grid, and to group those cells separated by the lower weight separators within the compound cells surrounded by heavier weight separators. Note that this is identical to nesting each cell surrounded by light-weight separators inside cells surrounded by heavier weight separators, resulting in a contour map of nested cell hierarchies.

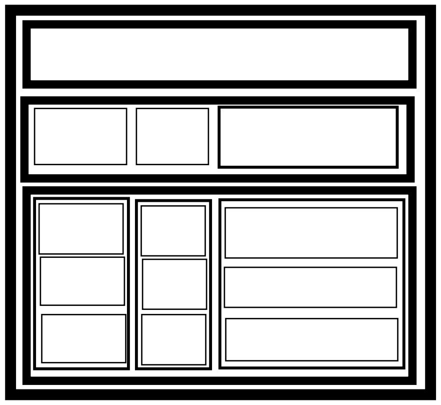


Figure 4: Contour map of cell hierarchies.

Given this natural hierarchical structuring of a table, we now address the problem of referencing individual cells in the table in a way that reflects the cell's position in the hierarchy.

3.2 Cell labeling format

The notation used for our cell labeling is an extension of the familiar notation for cell addresses used in many popular spreadsheets. Cells are addressed based on its column and row location. The upper left (“home”) address is A1. Columns run from addresses A through Z, then AA through AZ, then BA through BZ, etc. The rows are numbered from 1. The two cells which are “knight moves” from the home address are, therefore, C2 and B3. C2 is a diagonally adjacent cell to (up and to the right of) cell B3.

The labeling system developed here extends the normal spreadsheet labeling by concatenating spreadsheet cell labels, one label for each level in the hierarchy: for example, a typical address of an individual cell in a five level hierarchy would be A1C1A3B1A1. A cell group at some level in the hierarchy is addressed relative to its position in the enclosing next higher level. Since groups of cells exist in each level of the hierarchy (except the bottom level), a group of cells are referenced relative to the position of the top left-most cell in a group (see figure 5).

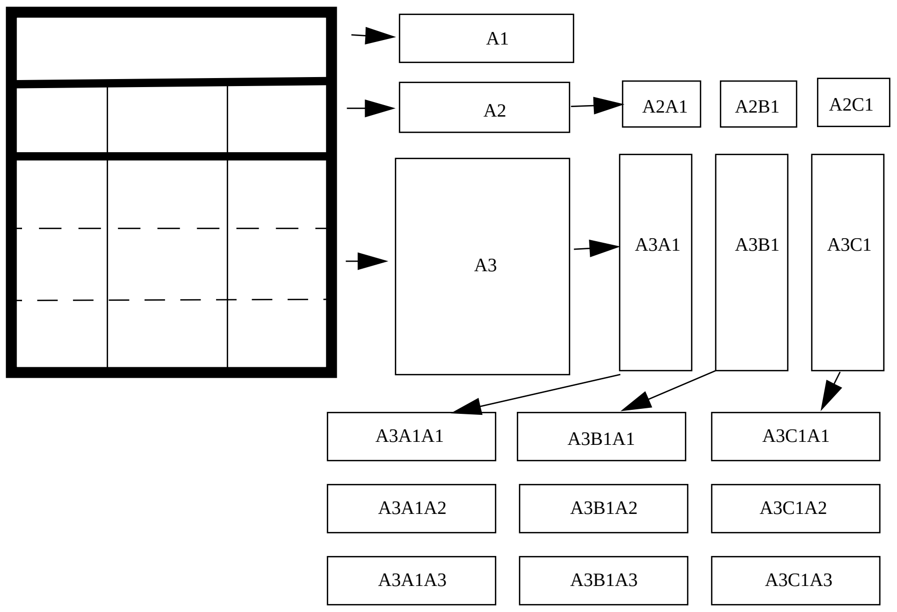


Figure 5: Labeling a table

3.3 Algorithm to Label Cells

Cells are grouped by building a general tree with the assistance of a minimum priority queue.

1. Start with any individual cell in the table. This is the first “current” cell.

2. Initialize a tree to be n nodes in height, with a single leaf, where n is the number of separator weights. Attach the current cell to the leaf of this tree.

3. Mark the current cell (so it will no be selected again).

4. Place each unmarked neighbor of the current cell into a priority queue. The key is the separator weight between the current cell and the neighbor.

5. Pull the minimum weight cell from the queue. Let this be the “new” cell. (If the queue is empty, exit.)

6. If the new cell is marked, goto 5.

7. The current cell belongs to n levels. Let k be the key of the new cell. Perform the following operations on the tree:
   ascend the tree n-k levels.
   add a new subtree to this level. This subtree will be a single leaf tree of n-k height.
   attach the new cell to the new leaf.

8. The new cell becomes the current cell.

9. Goto 3.

Once the tree is built, (recursively) sort the child nodes of each level according to the upper-left most position of the levels beneath. Each level of the tree then represents one part of the label of the child nodes beneath it, and labels can be generated by traversing the tree.

4 Templates

At this point, the labeling of individual cells relative to a separator structure has been achieved. The goal is to match this raw data to a model such that relational information in the table can be extracted.

A template is a set of logical specifications or assignments. These assignments map a logical label to a set of cells in the table: all logical entities in a table will be represented by some set of individual cells. A logical specification in this project has the following format:

<logical label> = *<cell label template>*

The explanation of templates is easier with reference to an actual example.

A1		
A2A1	A2B1	A2C1
A3A1A1	A3B1A1	A3C1A1
A3A1A2	A3B1A2	A3C1A2
A3A1A3	A3B1A3	A3C1A2

Figure 6: An example labeled table.

There are 3 levels of separator hierarchy in this example. If the first row of is considered to be a table heading, then the logical assignment would be:

table heading = A1

Wild-carding allows multiple cells to be assigned to a single logical entity:

column headings = A2??

Where "?" is the wild card heading. This matches cells A2A1, A2B1, and A2C1, the column headings.

A simple extension allows a multiple assignment, based on a wild card reference in the logical label:

col?2 = A3??

groups all elementary cells matching the template by the 2nd coordinate("?2"), numbering from the 0th cell label template position (the "A"). This specification would be the same as the three specifications:

colA = A3A????
colB = A3B????
colC = A3C????

However, if these later specifications were specified in a model file, the model would only be able to characterize 3 column tables. Using multiple assignments allows for a varying number of columns to be handled by a model.

5 The Table Model

The core of our system is the table model. The table model supplies the essential characteristics that a table will exhibit if the table can be recognized, and coordinates the mapping of graphical attributes to logical structure. The model is divided into 4 main parts: feature codes, image isolation, phases, and templates.

This section describes each section, and shows an example of analyzing a table using the model. The example table is shown in figure 7.

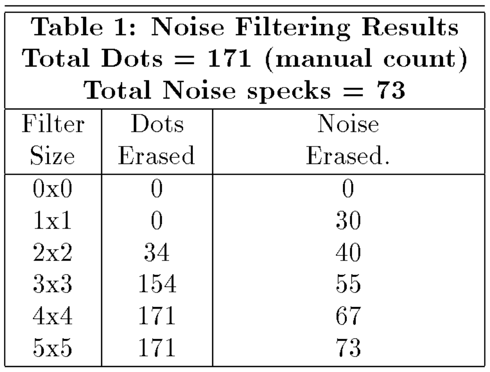


Table 1: Noise Filtering Results
Total Dots = 171 (manual count)
Total Noise specks = 73

Filter Size	Dots Erased	Noise Erased.
0x0	0	0
1x1	0	30
2x2	34	40
3x3	154	55
4x4	171	67
5x5	171	73

Figure 7: Example table to illustrate table models.

5.1 Feature codes

Feature codes indicate all of the graphical features that are required to separate each cell. A few examples of features and feature codes are horizontal lines (HL), vertical lines (VL), horizontal space (HS), and vertical space (VS). Feature codes must be recognized by the virtual image analyzer. Other codes may be added as required. The entry in the model file for this table is:

[feature-codes]

HL VL HS HL-M

where HL-M means that analyzing the table requires recognition of multiple horizontal lines.

5.2 Image isolation

These are the features of a table which separate the table from the remainder of the image. In this work, we assume that the table image lies on a white background. Thus, white space features (HS and VS) can be used. If the table is surrounded by lines (as in a box), then horizontal and vertical lines can be used. As shown in section 3, this will form the “heaviest weight” separator.

For the example table (which is in a box), the entry for image isolation would be:

[image-isolation]

* top bottom left right (comments start with *)

HL-M HL VL VL

5.3 Phases

As we proceed in a top down fashion, each phase of marker placement subdivides the areas marked in the previous phase. For a given phase, each cell of the previous phase may be subdivided, and different image features may used to subdivide different cells. All separators placed in a given phase will have the same weight, and each separator (which is to say, the underlying graphical feature) must span the entire cell to cause a separator to be placed. Between phases cell labels are calculated anew, and are used to specify the cells to be subdivided.

For the example table, there are 3 phases used to separate the table. The entry is:

```
[phase 0]
        ?: HL    * ? indicates the entire table image.
[phase 1]
        A2: VL
        A3: VL
[phase 2]
        A2??: HS
```

This decomposes the table in accordance with figures 5 and 6.

5.4 Templates

Templates were discussed in section 4. A set of templates which can be used to characterize the table in figure 7 might be:

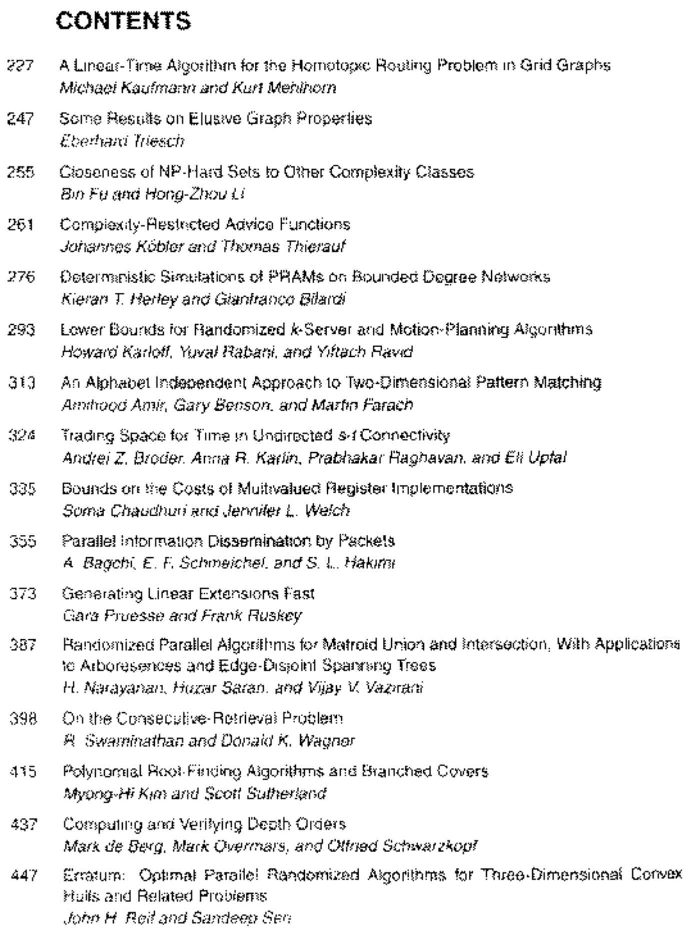
```
[templates]
tablehead = A1
colhead?2 = A2??
col?2 = A3????
tuple?5 = A3????
* mark end of model file
[end]
```

6 Results

A table analysis system with an X Window System graphical interface and based on the notions described above was written. We present in this section preliminary results of using this system on a set of tables; the system was used to analyze the table of contents of the SIAM Journal on Computing; the model was developed manually using one table (the April 1994 issue) and the model was tested on 9 other tables. A sample of the table image is shown in figure 8, and the model file is shown in figure 9.

The separators are all based on white space between entries: for example, the space between title and author is narrower than the spaces between articles; this is one basis for drawing markers. Since the feature extraction only looks at white space and ruling lines, errors will occur where there is more than one line of title or authors; extraneous marks will be drawn in this case. If feature extraction sensitive to typeface were substituted for the one used in this implementation such an error could be avoided.

With the graphical user interface used by the system these extra marks can be deleted (or missing marks inserted) during the analysis process and an accurate result can be obtained. The number of such corrections are tabulated in terms of mouse clicks needed to make the corrections: this is presented in table 1.



CONTENTS

227 A Linear-Time Algorithm for the Homotopic Routing Problem in Grid Graphs
Michael Kaufmann and Kurt Mehlhorn

247 Some Results on Elusive Graph Properties
Eberhard Triesch

255 Closeness of NP-Hard Sets to Other Complexity Classes
Bin Fu and Hong-Zhou Li

261 Complexity-Restricted Advice Functions
Johannes Köbler and Thomas Thierauf

276 Deterministic Simulations of PRAMs on Bounded Degree Networks
Kieran T. Herley and Gianfranco Bilardi

293 Lower Bounds for Randomized *k*-Server and Motion-Planning Algorithms
Howard Karloff, Yuval Rabani, and Yiftach Ravid

313 An Alphabet Independent Approach to Two-Dimensional Pattern Matching
Amihood Amir, Gary Benson, and Martin Farach

324 Trading Space for Time in Undirected *s-t* Connectivity
Andrei Z. Broder, Anna R. Karlin, Prabhakar Raghavan, and Eli Upfal

335 Bounds on the Costs of Multivalued Register Implementations
Soma Chaudhuri and Jennifer L. Welch

355 Parallel Information Dissemination by Packets
A. Bagchi, E. F. Schmeichel, and S. L. Hakimi

373 Generating Linear Extensions Fast
Gara Pruesse and Frank Ruskey

387 Randomized Parallel Algorithms for Matroid Union and Intersection, With Applications to Arboresences and Edge-Disjoint Spanning Trees
H. Narayanan, Huzur Saran, and Vijay V. Vazirani

398 On the Consecutive-Retrieval Problem
R. Swaminathan and Donald K. Wagner

415 Polynomial Root-Finding Algorithms and Branched Covers
Myong-Hi Kim and Scott Sutherland

437 Computing and Verifying Depth Orders
Mark de Berg, Mark Overmars, and Otfried Schwarzkopf

447 Erratum: Optimal Parallel Randomized Algorithms for Three-Dimensional Convex Hulls and Related Problems
John H. Reif and Sandeep Sen

Figure 8: SIAM Journal on Computing table of contents

```
[feature-codes]
      HS VS HS-W.5 HS-W.99 VS-W.5 HS-W.0
[image-isolation]
      HS HS VS VS
[phase 0]
      ?: HS-W.90
[phase 1]
      A2:HS-W.5
[phase 2]
      A2:VS-W.5
[phase 3]
      A2??B?:HS-W.0
[templates]
Article_?3 = A2A?
Title_?3 = A2A?B1A1
Author_?3 = A2A?B1A2
Page_Num_?3 = A2A?A1
[end]
```

Figure 9: Model for SIAM Journal on Computing table of contents

Table 1: Results of table analysis of SIAM table of contents

Month of Issue	Clicks to change edit mode	Clicks on table image
Apr 1994*	1	2
Apr 1995	1	1
Aug 1993	1	2
Dec 1993	2	3
Dec 1994	4	6
Feb 1994	1	4
Feb 1995	1	1
Jun 1995	1	4
Oct 1993	3	3
Oct 1994***	5	9
Apr 1994**	15	99

*=table used to develop model
**=no model used to analyze table - analysis done entirely by hand on the image
***=poor image quality

7 Discussion and Conclusion

The data reflected in a table image may be the result of the merging or joining of several different relations or SQL tables. The ultimate goal of this work is the reconstruction of constituent relations from table images.

Deciding on sources of table input appropriate for this system is an interesting question. Tables which are static would not be of practical importance for this system; the Periodic Table is sufficiently static such that the information reflected in the table image would already be available in an existing database; it's not important that the Periodic Table be read and analyzed every time that it is encountered. Additionally, tables which have a very fixed format and which already have a system built explicitly for them are also not appropriate for this system; for example, tables for income tax forms (at least for a given year) do not vary at all in format, so recognition systems for these tables could be hard coded; these tables don't need the power of a modeling system.

Now that we have suggested some inappropriate tables, what would constitute an appropriate source of tables? Those tables which vary somewhat in format, that is, in number of columns, various column heading formats, etc. would be good candidates for this system. Currently, we are considering tables that appear in specific scientific journals or in textbooks, since these tables usually follow a standard set of layout rules, which would make models easier to develop. The question is still open, however.

Tables within a publication may follow one of a set number of layout patterns, instead of a single pattern. The plausibility of detecting one of several general table layouts will be studied.

Simplifying table model development is a key issue in this system. Two tasks related areas of improvement are an interface for the naive user, and fine-tuning old models or deducing new models by user corrections to the table analysis (using supervised learning techniques).

References

[1] Laurentini, A. and P. Viada, "Identifying and Understanding Tabular Material in Compound Documents," Proc. ICPR, 1992, pp. 405-409.

[2] Chandran, S. and R. Kasturi, "Structural Recognition of Tabulated Data," ICDAR Japan, October 1993, pp. 516-519.

[3] Watanabe, T., H. Naruse, Q. Luo, N. Sugie, "Structure Analysis of Table-form Documents on the Basis of the Recognition of Vertical and Horizontal Line Segments," ICDAR France, 1991, pp. 638-646.