G. Nagy, D. W. Embley, D. P. Lopresti, Boxy, semi-structured document elements, GREC'13 Record, Bethlehem, PA August 2013.

Boxy, semi-structured document elements

G. Nagy[1], D. W. Embley[2], D. P. Lopresti[3]

[1]Rensselaer Polytechnic Institute, [2]Brigham Young University, [3]Lehigh University
nagy@ecse.rpi.edu

Abstract. The characteristics of lists, forms and tables are compared from the perspective of layout and indexing. Examples of ambiguous document elements reveal barriers to interpreting them. As a common denominator, the collection of facts in lists, forms, and tables all constitute first-order logic theories and can be represented as machine-queriable relations in a relational database.

Keywords: Table, form, list, relational database

1 Introduction

Conflicting or inconsistent definitions of lists, forms, and tables hamper the development and application of specialized methods for the interpretation of such "boxy," semi-structured document elements. Our objective is to determine their commonalities and differences to the extent possible using only their structure, without semantics or context. Although strictly structural criteria suffice to classify many of them into exactly one category, others span two or even all three categories.

Differences aside, commonalities of lists, filled-in forms, and tables are that they all serve to convey facts—assertions in first-order logic. Thus, in addition to facilitating understanding and analysis by humans, these concise, boxy representations of fact collections can become machine "understandable" (e.g., queriable as relations in a relational database). An acceptable set of definitions, which we offer here for consideration, may accelerate research toward automating the recognition and ultimately the machine "understanding" of graphic document elements, which has been a longstanding objective of the GREC community.

In Section 2 we describe the structure and layout of these related families of semi-structured document elements. In Section 3 we illustrate why it is difficult to classify boxy, table-like document elements to everyone's satisfaction and demonstrate common ambiguities. In Section 4 we address one goal of recent research: populating databases or ontologies with data extracted from lists, filled-in forms, and tables. Since our proposals are intended to stimulate discussion, we present no conclusion.

2 Observations on boxy document elements

Similarities and differences between lists, forms and tables are due to their purpose, appearance, and embedded search mechanisms (i.e., *indexability*). The discussion below is couched in the traditional printing and publishing vocabulary and may be easier to follow with a glance at the figures in Section 3. Our presentation is abstract in the sense that it is not tied to any data structure, algorithm or heuristic for analysis or interpretation.

2.1 Lists

Lists can be used either to collect or to disseminate information: *List your favorite radio stations* vs. *List of NYC Radio Stations*.

Lists contain related items, but the nature of the relationship may be implicit. They rarely have headers designating the types of items in the list (e.g., the list of authors of a paper does not have a header, such as *authors:*). Lists are searched over data items rather than headers. A list may be *ordered*, to facilitate search, or *unordered*. Ordering may be implicit and possibly obscure (authors of a paper ordered by contribution to the research or a grocery list ordered by aisle order in a store). A list can be laid out like a table, but without rulings. Lists seldom contain checkmarks or only numerical values. Vertical lists are often aligned, but items in horizontal lists are usually separated only by punctuation.

In a *single-item list* each entry has only one item (an author name or the name of a grocery item). Each entry in a *multi-item list* has many items (a student's list of courses currently being taken along with meeting times and places). Lists may be juxtaposed. List entries may be complex (a list of box scores in the sports section of a newspaper where the boxy elements themselves consist of lists and tables and even forms filled in by the reporters). Occasionally, multi-item lists have in-line item designators ("John Jones born 1856 died 1907" where "born" and "died" are not list items but serve to distinguish the two years). Most often, however, items in lists are understood inherently within context and without item designators.

Lists may be *nested* (e.g., a list of children born to a family within a list of families); *mixed*, interleaving different kinds of list entries (a list of references at the end of a paper where entries for journals, books, theses, etc. are all intermixed); *factored* (surnames in a telephone directory for each person listed below until the next surname is encountered); or *split* (by page boundaries with footers and headers between, or even within, list entries).

2.2 Forms

Forms are used for collecting information. They are also called *bureaucratic forms*, *office forms*, *official forms* or, more specifically, *tax forms*, *claim forms*, *betting forms*.

Forms were sometimes published in newspapers with a request for reader feedback. Currently many organizations maintain websites with downloadable forms, but computer-fillable web forms are replacing typed and handwritten forms. All computer-fillable forms can be printed. Security measures may, however, prevent filled-out forms from being downloaded or saved by the client.

In addition to a *Form Name* ("Application for Driver's License"), professionally designed forms usually have a *Form Number*, *Version Number*, or *Date of Issue*. Forms may also show instructions, organizational affiliation (including logos), source, signature lines, spaces for stamps, and advertising. The preprinted instructions may include lists or tables: e.g., state sales tax rates.

Data from individual forms is often manually or automatically entered into a database. The aggregated data may be presented or published in table format.

The principal elements of a form are *labeled fields* demarcated by line art or color. The blank spaces for entering information may include horizontal lines, combs, or other aids to separate characters. Fields may be grouped by line art or color at one or more levels to facilitate entering the required information.

The labels are preprinted in or near the blank space where the information is to be entered. Labels may range from a single word to an entire paragraph, possibly in several languages. Forms may also contain check boxes. Line art and labels may be printed in a *drop-out color* invisible to the designated scanner. *Mark sense forms* represent an extreme combination of drop-out ink and check boxes.

In some web forms, the amount of space per entry expands as needed, up to a set maximum. Others impose a strict word-count limit. Web forms, like payment forms listing purchased items and prices and requesting credit card information, can be created dynamically. Forms often solicit redundant information for error detection.

Form configurations may range from the very simple, like forms for recording tournament chess games, to the outright recondite (like some tax forms). Even simple forms may have dozens of repetitive fields and continuation pages.

Most filled-in forms cannot be confused with tables because they are not constructed on an underlying uniform grid and form fields are distinguished by field labels rather than horizontal and vertical headers. However, any table can be converted to a form by deleting the contents of the value cells and adding a request to fill them.

Currently the conversion of forms to computer databases is far more important commercially than that of tables or lists.

2.3 Tables

Tables are universally used for presenting data logically organized into two or more *categories*. Their *value cells* (*data cells*) are laid out on a uniform grid. Each value cell is *indexed* by its row and column headers. The 2-D indexing distinguishes tables from multi-item lists. In conventional printing terminology, the principal regions of a table are called *stub head*, *stub*, *column header*, and *data*.

An $M \times N$ table has M rows and N columns of value cells, in addition to one or more columns of row headers and one or more rows of column headers. Some authors include the header rows and columns in their counts.

A single category (*Country*) can be indexed by a flat header, or by a hierarchical header (*Africa/Chad,Tunisia*; *Asia/China,India,Japan*) laid out in several rows or columns or designated by indentations or font characteristics. Hierarchical headers also allow 2-D display of more than two categories (column: *Country/France;* row: *Gender/Male//Year/2000*); value: *81,200.* Values may be conceived as populating a multi-dimensional array where each dimension corresponds to a distinct category.

Since horizontal and vertical table organization is symmetric and permutable, the number of possible table layouts increases combinatorially with the number of categories and their membership. The choice may be guided by the aspect ratio of the available page or display space, preference for horizontal or vertical labels, compatibility with existing tables, and expected reader interests. Larger tables tend to be laid out with more rows than columns. Thus Canadian provinces often appear as column headers, while US states are typically row headers.

The order of rows and columns does not affect indexing: When row order is significant, the leading column may be populated with integers denoting rank. Since these uniquely index all the remaining rows, they logically suffice for row headers in spite of their descriptive poverty.

1x*N* or *M*x1 *tables* are *degenerate*. A single-row table requires one or more column-header rows, but the row header for its single row is optional. Conversely, a single-column table, unlike a single-column list, requires row headers, but no column header. A degenerate table may have an arbitrary number of category dimensions. The smallest non-degenerate table has 2x2 value cells.

Every category should be a *rooted tree*. Its root serves as its *Category Name*. In practice, it is often omitted because it is obvious to the reader. For instance, a row or a column consisting of a list of countries need not be designated by the root header *Country*. When a category root is missing, an arbitrary string (e.g., *RootHeader#2*) may be inserted to complete the category structure. Assigning a meaningful name would require semantic analysis of the contents of the table, table title, notes, or of the surrounding text. For instance a column showing hours and minutes could denote arrival or departure times.

In some CSV tables it is difficult to distinguish between a column root header and the title of the table. In others, the expected location of the root column header (above the leaf column headers) contains the source of the data or the year when the data was collected. The stub head may also contain root headers. Most of the data (value) region should be populated and may contain duplicate rows or columns. Sparse data can be presented more compactly as a list.

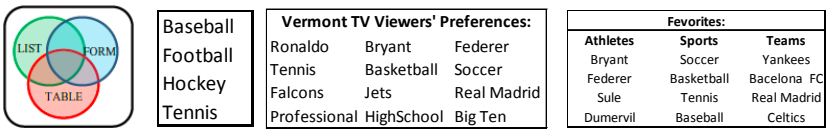
In a *Well-Formed Table (WFT)*, every value cell is uniquely indexed by its row and column *header paths*. A hierarchical (row or column) header may index one or more categories. A single-category header path consists of the root-to-leaf path of the corresponding category tree. A multi-category header path consists of concatenated category paths (*Year, 2000; Gender, Male*) or *(Year, 2001; Gender, Female*). WFTs are generally amenable to automated data extraction using only structural information.

Egregious tables may not puzzle human readers, but they challenge algorithms and require external context to extract values with their applicable indexes. A *concatenated table* merges distinct tables with identical or similar row or column headers. In order to keep headers close to values of likely reader interest, an egregious table may have headers elsewhere than only at the top or left edge. *Nested tables*, tables with *graphic cell contents*, *matrices*, and tables with row and column headers that are *implicit* or *incomplete* may also be considered egregious. However, *right-to-left* and *top-to-bottom* scripts require only minor modifications of table layout expectations.

Good table layout is an art described in several books and in lengthy sections of the *US Government Printing Office Style Manual* and in the *Chicago Manual of Style*.

3 List, Form, or Table?

The possible overlap between tables, forms, and lists can be visualized as a three-variable Venn diagram with eight regions. We present boxy document elements that fall into each of these regions (except in the null region). We use contrived examples to keep the illustrations small and, we hope, easily understood.



(a) **(b)** **(c)** **(d)**

Fig. 1.(a) Venn diagram of boxy document elements (b) a simple untitled list (c) titled list of lists ((Ronaldo, Bryant, Federer),(Tennis, Basketball, Soccer),...,…) (d) three juxtaposed lists.

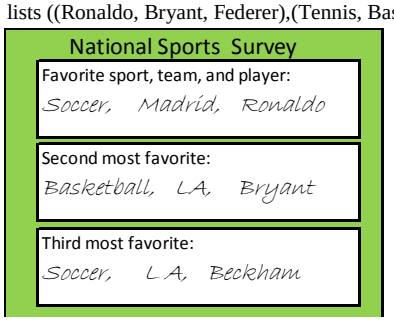


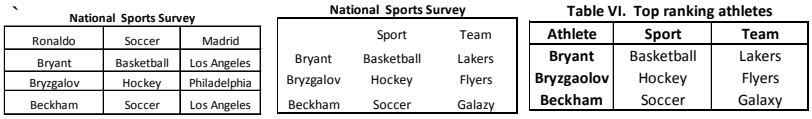
Fig. 2. A survey form.

Table V. Most watched athletes in Vermont

Rank	Sport	Athlete	Event
1	Tennis	Federer	Wimbledon
2	Soccer	Beckham	World Cup
3	Basketball	Bryant	NBA Finals

Fig. 3. A table with row and column headers.

Classifying ambiguous document elements requires some context. For example in Fig. 4a we have to know that Soccer is not a heading for Basketball, Hockey, and Soccer, and that Bryant is not a kind of Ronaldo.



National Sports Survey

Ronaldo	Soccer	Madrid
Bryant	Basketball	Los Angeles
Bryzgalov	Hockey	Philadelphia
Beckham	Soccer	Los Angeles

(a)

National Sports Survey

	Sport	Team
Bryant	Basketball	Lakers
Bryzgalov	Hockey	Flyers
Beckham	Soccer	Galazy

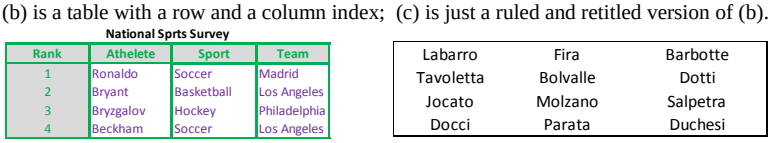
(b)

Table VI. Top ranking athletes

Athlete	Sport	Team
Bryant	Basketball	Lakers
Bryzgaolov	Hockey	Flyers
Beckham	Soccer	Galaxy

(c)

Fig. 4. List or table? (a) is a list because it has no column index; (b) is a table with a row and a column index; (c) is just a ruled and retitled version of (b).



National Sprts Survey

Rank	Athelete	Sport	Team
1	Ronaldo	Soccer	Madrid
2	Bryant	Basketball	Los Angeles
3	Bryzgalov	Hockey	Philadelphia
4	Beckham	Soccer	Los Angeles

(a)

Labarro	Fira	Barbotte
Tavoletta	Bolvalle	Dotti
Jocato	Molzano	Salpetra
Docci	Parata	Duchesi

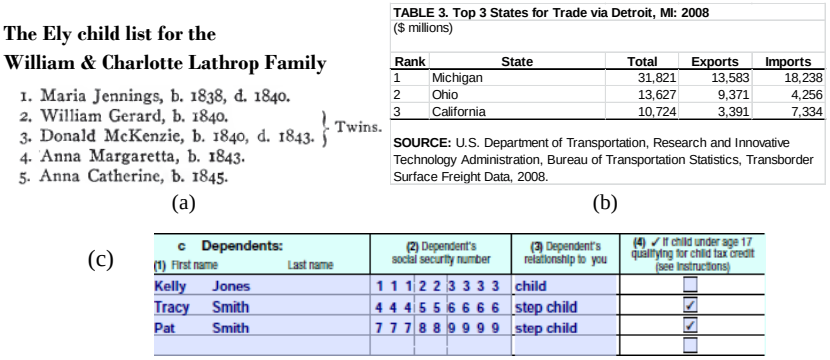
(b)

Fig. 5. List, form or table? (a) The font and background colors suggest that it is a filled-in form. (b) Could be any of the three: needs semantics to determine if it contains headers.

4 Queriable relational objects

A *relation* in a relational database is a set of *n*-tuples. An *n-tuple* functionally associates *n* attribute names with *n* values, forming a set of attribute-value pairs. Each *n*-tuple in a relation has the same set of attribute names. When displayed as a 2-D table, the attribute names appear as column headers and each row contains a tuple's values, positioned in the proper column. To illustrate the conversion of lists, tables,

and forms to database relations, and thus to machine "understandable" and queriable objects, we give a list, table, and filled-in form in Figure 7 and show their relational representation by giving the first two tuples of each relation:



The Ely child list for the
William & Charlotte Lathrop Family

1. Maria Jennings, b. 1838, d. 1840.
2. William Gerard, b. 1840.
3. Donald McKenzie, b. 1840, d. 1843. } Twins.
4. Anna Margaretta, b. 1843.
5. Anna Catherine, b. 1845.

(a)

TABLE 3. Top 3 States for Trade via Detroit, MI: 2008
($ millions)

Rank	State	Total	Exports	Imports
1	Michigan	31,821	13,583	18,238
2	Ohio	13,627	9,371	4,256
3	California	10,724	3,391	7,334

SOURCE: U.S. Department of Transportation, Research and Innovative Technology Administration, Bureau of Transportation Statistics, Transborder Surface Freight Data, 2008.

(b)

(c)

c Dependents: (1) First name	Last name	(2) Dependent's social security number	(3) Dependent's relationship to you	(4) ✓ if child under age 17 qualifying for child tax credit (see instructions)
Kelly	Jones	1 1 1 2 2 3 3 3 3	child	☐
Tracy	Smith	4 4 4 5 5 6 6 6 6	step child	☑
Pat	Smith	7 7 7 8 8 9 9 9 9	step child	☑
				☐

Fig. 7. (a) A printed and scanned multi-item list (b) A web table (c) Part of a filled-in form.

{ {(ChildNr, 1), (Name, Maria Jennings), (BirthYear, 1838), (DeathYear, 1840)},
{(ChildNr, 2), (Name, William Gerard), (BirthYear, 1840), (DeathYear, ⊥)}, …}
(The attribute names, which are only implicit or abbreviated in the list, are human provided.)

{ {(Rank, 1), (State, Michigan), (Total, 31821), (Exports, 13583), (Imports, 18238)},
{(Rank, 2), (State, Ohio), (Total, 13627), (Exports, 9371), (Imports, 4256)}, …}
(These tuples can be derived from the table without additional assumptions.)

{ {(FirstName, Kelly), (LastName, Jones), (SSN, 111223333), (Relationship, child), (U17, no)},
{(FirstName, Tracy), (LastName, Smith), (SSN, 444556666), (Relationship, step child), (U17, yes)}, …}
(Although perhaps derivable from the form, these attribute names are human provided.)

The conversion of tables to database relations differs from the conversion of lists or forms because tables have two or more indexing headers. When converting tables to relations, one of the indexing headers becomes the set of attributes for the relation and the others become key values. It does not matter which of the headers is chosen to constitute the set of attributes since the indexing headers are all transposable with one another. Key values in a table uniquely identify rows in a relation and thus serve the purpose of row headers. For example, the row headers in the table in Figure 7b are the rank values, and in the induced relation for the table, the rank values are keys.

Automating machine "understanding" for lists, forms, and tables consists of algorithmically transforming human readable images of these boxy components to relations for query by SQL, or more generally, to ontologies representing first-order theories. Challenges include discriminating values from labels, properly associating labels with values, replacing implicit headers with appropriate labels, and reformulating facts as tuples in relations or as predicate assertions for ontologies.

References

To be added from our bibliography of 200+ table papers if accepted for publication

Appendix: Notes on File Formats

Printed tables appear in books, newspapers and journals. Hardcopy tables are increasingly scanned rather than keyed-in for computer analysis. The DIA literature refers to scanned table in TIF, BMP or PNM files as *bitmapped tables.*.

ASCII tables occasionally appear in email. Their structure is indicated by character and line spacing, and a few printable symbols (_, -, ,=, |). They have been largely supplanted by more expressive representations like HTML that also use only ASCII (or equivalent) encoding.

HTML tables delimit rows and cells by tags. HTML also has tags for table titles and footnotes. Tags originally intended for tables are, however, often used in web pages to lay out non-table material.

Spreadsheets are WYSIWYG software for manipulating tables. The internal representation may be proprietary (XLSX) or open (CSV). XLS and XLSX preserve appearance, including merged cells and relative row heights and column widths. CSV tables must have the same number of cells in each row and in each column, so spanning cells are unmerged. CSV preserves the underlying uniform grid structure but it loses most appearance features like cell size, internal cell layout, font and background color, and ruling lines. Spreadsheets and CSV do not distinguish between header cells and value cells and contain no explicit category indexing. CSV tables imported from HTML include the table title and footnotes. In the original table they are usually positioned in a cell spanning several columns, but in CSV their value appears in the first (top-left) of the resulting atomic cells.

Most programming languages can import and export tables in spreadsheet formats. Off-the-shelf or built-in software is available for precise conversion between HTML, word-processing formats (e.g. RTF) and spread sheets. Conversion of tables from PDF is not necessarily lossless.

Many HTML and spreadsheet tables found on the web are automatically constructed (sometimes dynamically, on demand) from a database. Usually the viewer does not have access to the underlying database.

A computer file that contains a table must be distinguished from the *rendered* version of that table because the rendering program may use table formatting information that is not explicitly represented in the file. For instance, the program must know that in a particular ASCII CSV file TABs delimit table cells and LF-CR pairs delimit rows.

To the best of our knowledge, there are no generally accepted final formats that accommodate structural information like header paths in tables. There are, however, industry-specific conventions for associating form fields with HTML keywords.