2011 International Conference on Document Analysis and Recognition

# Data Extraction from Web Tables: the Devil is in the Details

George Nagy
Electrical, Computer, and Systems Engineering
Rensselaer Polytechnic Institute
Troy, NY, USA 12180
nagy@ecse,rpi.edu

Sharad Seth, Dongpu Jin
Computer Science and Engineering Department
University of Nebraska – Lincoln
Lincoln, NE, USA 68588
seth@cse.unl.edu, jindongpu@hotmail.com

David W. Embley, Spencer Machado
Computer Science Department
Brigham Young University
Provo, UT, USA 84602
embley@cs.byu.edu, admiralmachado@gmail.com

Mukkai Krishnamoorthy
Computer Science Department
Rensselaer Polytechnic Institute
Troy, NY, USA 12180
mskmoorthy@gmail.com

*Abstract*— **We present a method based on header paths for efficient and complete extraction of labeled data from tables meant for humans. Although many table configurations yield to the proposed syntactic analysis, some require access to semantic knowledge. Clicking on one or two critical cells per table, through a simple interface, is sufficient to resolve most of these problem tables. Header paths, a purely syntactic representation of visual tables, can be transformed ("factored") into existing representations of structured data such as category trees, relational tables, and RDF triples. From a random sample of 200 web tables from ten large statistical web sites, we generated 376 relational tables and 34,110 subject-predicate-object RDF triples.**

***Keywords-visual table, relational table, RDF, header-paths***

## I. INTRODUCTION

In the first decade of table processing, researchers concentrated on finding the underlying grid structure of scanned tables from rulings or text alignments [1,2], and of ASCII tables in email [3]. In the second decade the emphasis was on locating and bounding HTML pages in web tables. Reviews of earlier work can found in [4,5].

We target here the efficient extraction of the relations of header cells to content cells and the representation of these relations in appropriate data structures. Such multi-dimensional indexing is a prerequisite for understanding individual tables and for combining their contents into a queryable database or populated ontology, as we proposed for TANGO [6]. Similar goals are addressed in [7,8].

The foundations of syntactic table analysis were laid by X. Wang in her 1996 PhD dissertation [9]. Although she was interested primarily in reformatting tables for various media and page sizes, her definition of *categories* is equally suitable for layout analysis. Simple tables have only two categories defined by their row and column hierarchies, but more complex tables, such as her prime grade book example of *Year, Term* and *Mark*, require multidimensional indexing.

In [10] we presented our methods for extracting paths through the header hierarchy to content cells, and for decomposing these paths into orthogonal categories. Here we propose procedures for more complex table layouts, report additional experiments, and present a new tool for rapid interactive correction. We demonstrate the transformation of tables meant for human reading into a relational database accessible by formal languages like SQL [11] for relational tables or SPARQL [12] for RDF triple stores [13]. Although many of the tables available at large sites of statistical information—our primary focus—are generated dynamically from databases, often no direct public access is provided to the databases themselves. Individual users must therefore reconstruct fragments of the database of each source, or possibly of databases of multiple sources, by harvesting and analyzing individual tables. The proposed methods are intended to accelerate this process.

Unlike most published work to date, we present an end-to-end solution from HTML to SQL/SPARQL. Our starting point is a collection of tables selected randomly from ten large statistical web sites [14]. HTML tables can readily be exported to Excel, which provides all the necessary VBA primitives for manipulating grid cells in our GUI. Comma Separated Value (CSV) files are easy to parse with Python. Although the transformation from HTML to CSV loses some formatting, the standard CSV format suffices for a broad range of applications, including intermediate states of table processing.

In Section II we discuss common table formatting conventions that must be accommodated by automated table analysis. Section III describes our heuristics for finding header paths given these conventions and presents our new GUI for interactive corrections. Section IV reviews the extraction of Wang category trees using mathematical software designed for the synthesis of logic circuits. Section V demonstrates the transformation to relational tables and RDF triples. Section VI reports our experimental results.

1520-5363/11 $26.00 © 2011 IEEE
DOI 10.1109/ICDAR.2011.57

## II. TABLE FORMATTING CONVENTIONS

Since the invention of printing, many table formatting conventions have been developed and refined for ease of human access to tabulated data (cf. *The Chicago Manual of Style* or the *US Government Printing Office Style Manual*). Although tables can be prepared with any text editor, most document preparation systems (e.g. *Office, Latex*) offer elaborate provisions for constructing tables.

A table contains a rectangular configuration of data cells, each of which can be uniquely designated by a named row and a named column. More formally, a *table structure* is a discrete rectilinear tessellation, or a rectangular tiling, based on the partition of an isothetic rectangle into rectangles defined on a $r_T$ x $c_T$ lattice [15]. The bottom-right region of a *well-formed table* (*WFT*) contains a set of $r_d$ x $c_d$ *content-* (aka *value-, data-, or delta-* in [9]) *cells* that can be uniquely specified by a *column-header path* and a *row-header path*. Fig. 1 explains our notation and shows how four *critical cells* define the segmentation of the table into *stub, row header, column header*, and *delta* regions. (Although our observations are drawn from a collection posted on the project website [ 16 ], we present simplified examples because most real tables are too large for the ICDAR page allotment.)

The extraction of header paths must take into account the commonly occurring configurations discussed below.

### A. Virtual Headers

Wang defines rooted category trees. Category roots, however, are often missing in real tables. In the table of Fig. 1, which has two column categories A and B and one row category C, it is necessary to add a virtual root header CH1 (for Category Header 1) to the category A paths.

| | a | b | c | d | e | f | g | h |
|---|---|---|---|---|---|---|---|---|
| 1 | | | **B** | | | | | |
| 2 | | | **B1** | | | **B2** | | |
| 3 | | | **A1** | **A2** | **A3** | **A1** | **A2** | **A3** |
| 4 | **C** | **C1** | *D11* | *D12* | *D13* | *D14* | *D15* | *D16* |
| 5 | | **C2** | *D21* | *D22* | *D23* | *D24* | *D25* | *D26* |

Figure 1. Table Notation. Lower-case letters above and numerals to the left are not part of the table, just Excel-style cell addresses. Cells a1:b3 are the stub header (stub). The delta-cell region is c4:h5. The critical cells are a1, b3, c4, and h5, which also uniquely define the header regions. The column header path to cell f4 (which has cell-content *D14*) is **B-B2-A1**. The corresponding row header path is **C-C1**.

### B. Headers in the Stub

In the simple example of Fig. 2a it is not obvious whether "A" in cell a1 (the only cell in the stub here) is the root for category B or C. A more realistic example is shown in Figs. 2b. The appropriate category trees cannot be determined without semantic considerations. In the large majority of the tables we have seen, however, the contents of the stub are *row* headers: we would expect AGE instead of VERMONT in merged-header-cell b1:e1.

| A | B1 | B2 |
|---|---|---|
| C1 | *D11* | *D12* |
| C2 | *D21* | *D22* |

(a)

| | VERMONT | | | |
|---|---|---|---|---|
| AGE | <25 | | >25 | |
| XXX | M | F | M | F |
| HIGH SCHOO | *50%* | *65%* | *44%* | *78%* |
| COLLEGE | *20%* | *23%* | *22%* | *25%* |

(b)

Figure 2. (a) Is "A" a row or column header? In (b), if XXX is GENDER, then it is is a column-root-header, but if XXX is EDUCATION then it is a row-root-header.This is a three-catgory table.

### C. Multi-row/column Indexing

In a table with $r_d$ rows and $c_d$ columns of delta-cells, there must be $r_d$ row-header leaf-cells and $c_d$ column-header leaf-cells. In the table of Fig. 3, one might at first consider State to be the row-root-header, and all but the first column as delta-cells. There are, however, multiple rows with the same entry (AL, MI, MN). Even adding the second column is insufficient because Minnesota Power Inc appears twice. The row-header here consists of the first three columns.

**Table 1.9 Net Summer Capacity of Plants Cofiring Biomass and Coal, 2007 (Megawatts)**

| State | Company Name | Plant I.D. | Plant Name | County | Biomass / Coal Cofiring Capacity | Total Plant Capacity |
|---|---|---|---|---|---|---|
| AL | DTE Energy Services | 50407 | Mobile Energy Services LLC | Mobile | 91 | 91 |
| AL | Georgia-Pacific Corp | 10699 | Georgia Pacific Naheola Mill | Choctaw | 31 | 78 |
| AL | International Paper Co | 52140 | International Paper Prattville Mill | Autauga | 49 | 90 |
| AR | Domtar Industries Inc | 54104 | Ashdown | Little River | 157 | 157 |
| AZ | Tucson Electric Power Co | 126 | H Wilson Sundt Generating Station | Pima | 173 | 559 |
| | | | ROWS OMITTED | | | |
| MI | S D Warren Co | 50438 | S D Warren Muskegon | Muskegon | 51 | 51 |
| MI | TES Filer City Station LP | 50835 | TES Filer City Station | Manistee | 70 | 70 |
| MN | Minnesota Power Inc | 10686 | Rapids Energy Center | Itasca | 27 | 28 |
| MN | Minnesota Power Inc | 1897 | M L Hibbard | St Louis | 73 | 123 |
| MO | University of Missouri-Columba | 50969 | University of Missouri Columbia | Boone | 6 | 91 |
| MS | Weyerhaeuser Co | 50184 | Weyerhaeuser Columbus MS | Lowndes | 123 | 123 |
| NC | Carlyle/Riverstone Renewable Ei | 10381 | Coastal Carolina Clean Power | Duplin | 44 | 44 |
| | | | ROWS OMITTED | | | |

Figure 3. Three columns are required here to index columns.

### D. Row/Column Order

In tables meant for humans, order can be suggestive. In Fig. 4, the Rank column may at first appear inferior to the State column as a row-header. But the table designer would have put State on the left if the table were meant to find the rank of various states instead of the states with various ranks (the title of this table was "*Top 3 States for Trade via Port Huron, MI: 2008*").

| Rank | State | Total | Exports | Imports |
|---|---|---|---|---|
| 1 | Michigan | 24,266 | 3,992 | 20,274 |
| 2 | Illinois | 8,259 | 4,669 | 3,590 |
| 3 | Texas | 7,001 | 4,635 | 2,366 |

Figure 4. Row order is important here, but column order is less so.

Even if a numerical index satisfies our header path uniqueness requirement, it is often of little value for querying table data. Nevertheless omitting the Rank column would entail loss of information because we don't currently add explicit order information (and preserve the original CSV cell addresses only as meta-data).

### E. Degenerate Tables and Lists

Our fundamental requirement is that each data cell can be indexed uniquely. We call a table with $r_d$ =1 or $c_d$ =1 *degenerate (*such a structure can have multiple categories). A structure missing any row or column header necessary for indexing every data cell is a *list*. Classification of multi-row and multi-column lists is addressed in [17].

### F. Aggregates

Tables often contain row or column totals that can be recovered from the rest of the table, as shown by Long on Australian financial reports [18]. Other aggregates, like median, standard deviation, or truncated mean, could also be identified when the span of the aggregation operation is clear. Sums are often identified by keywords like *Sum* or *Total* which apply to the whole, or part of a, row or column. In other cases more complicated semantic processing may be required. In the *Canada Statistics* tables, totals for all the provinces and territories appear under CANADA. But CANADA also often appears as a row-header root for the provinces, without any aggregate data.

## III. CONSTRUCTION OF HEADER PATHS

### A. CSV Version of the HTML table

The CSV text file is parsed by a Python program [10]: The CSV structure must be modified because merged cells, like the column-header root containing B in Fig. 5, are unmerged. For the table of Fig. 1, B, B1, and B2 are copied into all the empty cells, represented by null strings, to their right. C is copied into the first empty cell of the fifth row. Blanks within cell contents are replaced by underscores, and some characters with special meanings in downstream programs ("\", "+", "*", "(", ")", etc.) are replaced by ASCII strings like "backslashtoken".

```
,,B,,,,,
,,B1,,B2,,
,,A1,A2,A3,A1,A2,A3
C,C1,D11,D12,D13,D14,D15,D16
,C2,D21,D22,D23,D24,D25,D26
```

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 0 | | | B | | | | | |
| 1 | | | B1 | | | B2 | | |
| 2 | | | A1 | A2 | A3 | A1 | A2 | A3 |
| 3 | C | C1 | D11 | D12 | D13 | D14 | D15 | D16 |
| 4 | | C2 | D21 | D22 | D23 | D24 | D25 | D26 |

Figure 5. CSV file and table for the Excel table of Fig. 1. Excel cell addresses changed to x-y coordinates to allow negative indices.

Both the header paths and the paths through the delta cell region consist of asterisk-separated-sequences of cell contents in double quotes, with the sequences separated by plus signs (Fig. 6). After the paths are extracted, the original cell coordinates, enclosed by angle brackets <>, are added to each path element. If needed, the program creates negative x-coordinates for roots virtually moved from the stub to left of the leftmost row-header column. The stub and delta region are not necessarily contiguous.

### B. Path Extraction Heuristics

The critical cells are located automatically if either of two conditions holds: (1) the stub header is empty, and (2) the delta cell region consists of numerical information. If neither of these conditions holds, the table is tagged for interactive identification of the critical cells.

Row header roots in the stub are added as roots of the row headers below them. In the table of Fig. 2a, A would be added to the row header paths C1 and C2 (*if D11, D12, D21, D22* were numerical).

```
rowpaths =
(("<0,3>C"*"<1,3>C1")
+("<0,4>C"*"<1,4>C2"));

colpaths =
(("<2,0>B"*"<2,1>B1"*"<2,2>A1")
+("<3,0>B"*"<3,1>B1"*"<3,2>A2")
+("<4,0>B"*"<4,1>B1"*"<4,2>A3")
+("<5,0>B"*"<5,1>B2"*"<5,2>A1")
+("<6,0>B"*"<6,1>B2"*"<6,2>A2")
+("<7,0>B"*"<7,1>B2"*"<7,2>A3"));
```

Figure 6. Row and column paths for table of Figs. 1 and 5. The delta paths are simimlar but longer.

### C. Interactive verification

VeriClick is an interactive Excel file with VBA macros that sequentially presents for verification the CSV tables in a given directory. It highlights the stub and delta-cell region recognized by Python table parser. If they are deemed correct, a click outside the table triggers the display of the next table. If wrong, as in Figure 7, the operator corrects the error by clicking first on the wrong cell then on the right cell. At most 8 clicks are required.

| 3 | Plastic waste by method of treatment or disposal. 1995-1997. Per cent | | | | | | |
|---|---|---|---|---|---|---|---|
| | 1995 | 1996 | 1997 | | | | |
| Material re | 0 | 2 | 2 | | | | |
| Incineratio | 15 | 15 | 14 | | | | |
| Landfill | 63 | 66 | 60 | | | | |
| Export | 2 | 3 | 3 | | | | |
| Other/unkr | 20 | 14 | 20 | | | | |
| 1 | Source: Plastretur AS and Statistics Norway's manufacturing statistics. | | | | | | |

Figure 7. VeriClick GUI for critical cells. In this table, the Python program misconstrued cell a1 for the stub, which should be only a2. Here the cause was poor table layout: "3" is part of the table title.

## IV. EXTRACTION OF CATEGORY TREES

With the asterisks and plus signs added in the row and column header paths, both resemble sum-of-products algebraic expressions. This convention simplifies extracting category trees from these expressions through an algebraic factorization process. As the formulation is completely symmetric for column and row header paths, we describe below only the column header path analysis.

The algebraic interpretation is based on a *covering* relation defined between each product term in the header paths expression with the column covered by it. The covering relation can be extended to sums of products by taking the union of individually covered columns. For example, the sum-of-products:

("<2,0>B"∗"<2,1>B1"∗"<2,2>A1")+("<4,0>B"∗"<4,1>B1"∗"<4,2>A3") (1)

covers the first and the third columns of values in the table in Fig. 5. Note that the cell labels <2,0>B and <4,0>B in this example are identical in the original table. To enforce this constraint, we drop the second (column) coordinate in the header paths. As a consequence, the labels <2,2>A1 and <5,2>A1 in the colpaths expression are also treated as being the same, in accordance with the normal conventions of table layout.

The covering relation can be further extended to *factored* forms, as long as the inverse *multiplying-out* process can recover the original product terms from it. For example, the whole colpaths expression can be factored as:

<0>B*[<1>B1+<1>B2]*[<2>A1*<2>A2*<2>A3] (2)

from which the category trees for the headers are derived by our Python program, as shown in Fig. 8. Here, the program has stripped the indices and added the virtual root header CH1 to represent the missing header for the second column category.

| B | CH1 | C |
|---|---|---|
| B1 | A1 | C1 |
| B2 | A2 | C2 |
| | A3 | |

Figure 8. Wang Categories for the table of Fig. 1

To facilitate the generation of relational tables (see Section V), the program combines the category-tree structures for the row and column headers into a canonical form, as shown below for the example:

C*(C1+C2)+B*(B1+B2)+CH1*(A1+A2+A3) (3)

It also generates multiple views of the table resulting from the different choices of the category chosen to provide the attributes.

Another output produced by the program are the values of the table features used in verification of the result against visual inspection of the CSV table, as illustrated below for the example of Fig. 1:

Row categories: 1; Leaf nodes for row categories: 2
Col categories: 2; Leaf nodes for col categories: 2, 3

A benefit of this formulation is that the algebraic factorization problem has been studied extensively in the past in fields ranging from symbolic mathematics [19] to logic synthesis [20], and we gain leverage from the sophisticated strategies developed in these fields. We map the header paths expression to Boolean algebra and adapt the logic synthesis tool *Sis* [21] for factorization. Interested readers can find further details in our earlier work [10].

## V. RELATIONAL TABLES & RDF TRIPLES

Given a factored expression for a table in canonical form along with the table's data indexed by the header paths, we can generate a corresponding relational table and populate it with the data. We can then query the table with SQL and otherwise manipulate it along with other tables in a standard relational database.

We assume that one of the category terms provides the attributes for the relational table while the remaining category terms provide key values for objects represented in the original table. We do not know which category would serve best for the attributes. We therefore transform a table with *n* categories into *n* complementary relational tables—one for each possible choice.

For the canonical expression (3) with the choice of the term CH1*(A1+A2+A3) for the attributes along with the header paths and data for the table of Fig. 5, Fig. 9 shows the generated SQL create statement and SQL insert statements for one of the relational tables. In general, we

(1) transform each header path of the term chosen to represent attributes into an attribute name (e.g., CH1_A1 in Fig. 9),

(2) transform the root node of the remaining header paths into attribute names (e.g., C and B in Fig. 9),

(3) declare the attributes of these remaining header paths to be the primary key (e.g., PRIMARY KEY (C, B) in Fig. 9), and

(4) insert tuples into the generated table: the key values are a cross product of the header paths below the root from each category (e.g., {C1, C2} × {B1, B2} for the four tuples in Fig. 9), and the data values are inserted as directed by the header paths of each data value in the original table (e.g., "D11" for the CH1_A1 attribute of the tuple whose key is {C1, B1} as Fig.9 shows).

```
CREATE TABLE Fig_1(C varchar(2),B varchar(2),
   CH1_A1 varchar(3),CH1_A2  varchar(3),CH1_A3 varchar(3),
   PRIMARY KEY (C, B));
  INSERT INTO Fig_1 VALUES("C1", "B1", "D11", "D12", "D13");
  INSERT INTO Fig_1 VALUES("C1", "B2", "D14", "D15", "D16");
  INSERT INTO Fig_1 VALUES("C2", "B1", "D21", "D22", "D23");
  INSERT INTO Fig_1 VALUES("C2", "B2", "D24", "D25", "D26");
```

Figure 9. Tuple generation for the table of Fig. 1

For RDF triples we transform each data value in a relational table into a (subject, predicate, object)-triple. Fig. 10 shows the resulting RDF for the first tuple generated for the relational table in Fig. 9. For each data value (e.g., "D11"), the subject in the triple is an object identifier for the tuple (e.g., the object identifier C-B_0 in Fig. 10 obtained as a hyphenated concatenation of the attributes of the primary key along with a subscript 0 for the first tuple, 1 for the second, and so on). The predicate for the triple is the attribute for the value (e.g., CH1_A1 for the value "D11" and B for "B1" in Fig. 10). And the object in the triple is the value itself ("D11" and "B1" in Fig. 10).

```
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:Fig_1="mysql://localhost:3306/Fig_1#">

<rdf:Description
rdf:about="mysql://localhost:3306/Fig_1/C-B_0"
Fig_1:C="C1"
Fig_1:B="B1"
Fig_1:CH1_A1="D11"
Fig_1:CH1_A2="D12"
Fig_1:CH1_A3="D13"
/>
...
</rdf:RDF
```

Figure 10. RDF for the first tuple in Fig. 9

## VI. EXPERIMENTAL RESULTS

All the CSV tables we processed were interactively stripped of external metadata like table titles and footnotes. (We have, since then, automated these tasks). The Python segmentation routines found the critical cells on 197 of the 200 tables and generated path 197 files. The three that failed have *X* or *NA* in some delta cells and non-empty stubs. 26 of the 197 tables required interactive correction of one or more critical cells before path extraction. The interaction (up to eight mouse clicks) using VeriClick takes on average about 5 seconds per table that needs to be corrected, and under two seconds for confirmation.

If the header paths are correct, factorization produces the correct canonical form. The factorization program yielded the canonical expression for 196 of the 197 path files. Some cells contained MySQL-disallowed characters and some names exceeded the 64-character limit on attribute names. We enclosed such names in quotes and built correspondence tables to connect short names to full names.

Our implementation selects each term in a canonical expression to serve as the attributes of a generated relational table. The *to-relational-table* conversion program generated and populated 376 MySQL relational tables (2 tables for each of the two-category tables and 3 tables for each of the three-category tables). Of 196 tables, eight failed the parse: five tables had duplicate attribute names, and three had bad syntax.

To generate RDF triples, our implementation converts each value in a relational table to a triple. The *to-RDF-triple* conversion program generated 188 RDF files and 34,110 subject-predicate-object triples.


### ACKNOWLEDGMENTS

This work was supported by NSF Grants # 044114854 (at RPI) and # 0414644 (at BYU) and by the Rensselaer Center for Open Software. Mangesh Tamhankar (RPI) developed VeriClick.


## REFERENCES


[1] T.A. Bayer, Understanding Structured Text Documents by a Model based Document Analysis System,: Procs. ICDAR'93, pp. 448–453. Tsukuba Science City, Japan, 1993.

[2] J.C. Handley, Table Analysis for Multiline Cell Identification. In:Kantor, P.B., Lopresti, D.P., Zhou, J. (eds.) Procs. DRR VIII (IS&T/SPIE ElectronicImaging), vol. 4307. San Jose, CA, 2001.

[3] D. Pinto, A. McCallum, Wei, W.B. Croft, Table Extraction using Conditional Random Fields, Procs. ACM SIGIR Conference on Research and Development in Inf'n Retrieval, pp. 235–242, 2003.

[4] R. Zanibbi, D. Blostein, J.R. Cordy, A Survey of Table Recognition: Models, Observations, Transformations, and Inferences, J. Doc. Anal. Recognit. 7(1), 1–16, 2004.

[5] D.W. Embley, M. Hurst, M. Lopresti, G. Nagy, Table Processing Paradigms: A Research Survey, J. Doc. Anal. Recognit. 8 (2-3), Springer, Heidelberg, 66-86, 2006.

[6] Y. A. Tijerino, D.W. Embley, D. W. Lonsdale, and G. Nagy, Towards Ontology Generation from Tables, World Wide Web Journal, vol. 6(3), 261-285, 2005.

[7] B. Krüpl, M. Herzog, W. Gatterbauer, Using Visual Cues for Extraction of Tabular Data from Arbitrary HTML Documents. Procs. of the 14th Int'l Conf. on World Wide Web, 1000-1001, 2005.

[8] A. Pivk, P. Ciamiano, Y. Sure, M. Gams, V. Rahkovic, R. Studer, Transforming arbitrary tables into logical form with TARTAR, Data and Knowledge Engineering 60(3), 567-595, 2007.

[9] X. Wang, Tabular Abstraction, Editing, and Formatting, Ph.D. Dissertation, U. Waterloo, Waterloo, ON, Canada, 1996.

[10] D.W. Embley, M. Krishnamoorthy, S. Seth, G. Nagy, Factoring WebTables, Procs. ACM EIA/AIE, Syracuse, NY: Modern Approaches in Applied Intelligence (Editors: .G. Mehrotra, C.Mohan, J. C. Oh, and P. K. Varshney), June 2011.

[11] D. Chamberlin, SQL, Encyclopedia of Database Systems, L. Liu and M. Tamer (eds.), Springer Verlag, 2009.

[12] http://www.w3.org/TR/rdf-sparql-query/

[13] http://www.w3.org/RDF/

[14] G. Nagy, R. Padmanabhan, M. Krishnamoorthy, R.C. Jandhyala, W. Silversmith, Table Metadata:Headers, Augmentations and Aggregates, Procs. DAS, pp. 507-510, Boston, MA, 2010.

[15] R.C. Jandhyala, G. Nagy, S. Seth, W. Silversmith, M. Krishnamoorthy, R. Padmanabhan, From Tessellations to Table Interpretation, L. Dixon et al. (Eds.): Calculemus/MKM 2009, Springer-Verlag, Berlin, vol. 5625 LNCS, 422-437, 2009.

[16] http://tango.byu.edu/data/

[17] M. Yoshida and K. Torisawa and J. Tsujii, A Method to Integrate Tables of the World Wide Web, "Proceedings of the International Workshop on Web Document Analysis (WDA 2001), pp. 31-34.

[18] V. Long. An RDF-based Blackboard Architecture for Improving Table Analysis, Procs. ICDAR 2009, Barcelona, Spain, 2009.

[19] D.E. Knuth, Factorization of Polynomials,.§4/6/2 in. Seminumerical Algorithms: The Art of Computer Programming. 2, Addison-Wesley, Reading, MA,. 1997.

[20] R.K. Brayton and C. McMullen. The Decomposition and Factorization of Boolean Expressions, Procs. International Symposium on Circuits and Systems, pages 49-54, May 1982.

[21] E.M. Sentovich E.M. Sentovich, K.J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P.R. Stephan, R.K. Brayton and A.L. Sangiovanni-Vincentelli, SIS: A System for Sequential Circuit Synthesis, University of California at Berkeley. downloaded 11/4/10 from: http://www.eecs.berkeley.edu/Pubs/TechRpts/1992/ERL-92-41.pdf.