

MODELS AND ALGORITHMS FOR REGULARIZING HETEROGENEOUS WEB TABLES

David W. Embley, Brigham Young University, Provo, UT
embley@cs.byu.edu

Mukkai Krishnamoorthy, Rensselaer Polytechnic Institute, Troy, NY
moorthy@cs.rpi.edu

George Nagy, Rensselaer Polytechnic Institute, Troy, NY
nagy@ecse.rpi.edu

Sharad Seth, University of Nebraska-Lincoln, Lincoln, NE
seth@cse.unl.edu

Abstract

Despite decades of research, fully automated end-to-end table processing—converting tables intended for human comprehension into a machine representations intended for query processing—remains elusive. Based on a formalization of well-formed tables, we proffer an algorithmic solution to end-to-end table processing for a large class of human-readable tables. The proposed algorithms transform well-formed tables to a canonical table format that maps easily to a variety of industry-standard data stores for query processing. The algorithms segment table regions based on the unique indexing of the data region by header paths, classify table cells, and factor header category structures of two-dimensional as well as the less common multi-dimensional tables. Experimental evaluations substantiate the algorithmic approach to processing heterogeneous tables. As demonstrable results, the algorithms generate queryable relational database tables and semantic-web triple stores. Application of our parameter-free algorithms to 200 web tables that have resisted alternative approaches shows that the algorithmic solution automates end-to-end table processing.

Tri-University Consortium for Table Research Report #1001

March 1, 2015

MODELS AND ALGORITHMS FOR REGULARIZING HETEROGENEOUS WEB TABLES

TABLE OF FIGURES	2
MODELS AND ALGORITHMS FOR REGULARIZING HETEROGENEOUS WEB TABLES	3
Abstract:	3
1. INTRODUCTION	3
2. PRIOR WORK	6
2.1 Wang Tables	6
2.2 Physical Structure Extraction (Low-level Table Processing).	7
2.3 Logical Structure Extraction (High-level Table Processing)	10
2.4 Our earlier work	12
3. HUMAN READABLE TABLES	12
3.1 What is a table?	13
3.2 Well Formed Tables: Formal Characterization	14
4. SEGMENTATION AND CELL CLASSIFICATION	18
4.1 Header Segmentation	18
4.3 Auxiliary Regions	21
4.4 Classification Table	21
5. COMPLEX HEADER STRUCTURES	22
5.1 Category Analysis	22
5.2 Factorization Algorithm	23
5.3 Canonical Tables	24
6. EXPERIMENTAL RESULTS	26
7. APPLICATION QUERIES	27
7.1 Relational Database Queries	27
7.2 Semantic Web Queries	31
8. CONCLUSION	33
ACKNOWLEDGEMENTS	34
REFERENCES	34

Table of Figures

Fig. 1.1. An exemplary table, used as a running example throughout the paper.....	4
Fig. 2.1. Source code of the HTML table in Fig. 1.1. Less than one fifth of the 446 lines are shown.....	8
Fig. 2.2 Text (Notepad) display of CSV file after Import from the HTML in Fig. 2.1.....	9
Fig. 2.3 Spreadsheet (Excel) display of CSV file after Import from the HTML in Fig. 2.1.	9
Fig. 3.1. Genetic coding tables.....	14
Fig. 3.2. Visual WFT model: (a) with and (b) without blank rows and columns.....	15
Fig. 3.3. The relations of Allen's interval algebra.....	16
Fig. 3.4. Network representation of the spatial constraints of the table model in Fig. 4(b).....	17
Fig. 4.1. Part of the table of Fig. 1.1 in CSV format.....	19
Fig. 4.2. The MIPS algorithm.4.2 Prefixing.....	20
Fig. 4.3. A web table that requires prefixing.....	21
Fig. 5.1. Canonical table for the table in Fig. 1.1 (first 30 of 240 rows).	25
Table 6.1: Profile of our random sample of 200 tables.	26
Table 6.2: Joint distribution of row and column header sizes in the indexable tables.....	27
Fig. 7.1. MS-Access screenshot of Query 1 results (partial).....	28
Fig. 7.2. International land trade with the U.S. through Detroit, Michigan.....	29
Fig. 7.3. International trade with the U.S. (with surface trade header fully shown in edit box).	29
Fig. 7.5. Results of Total-check query.	31
Fig. 7.6. Generated RDF triples.	31
Fig. 7.7. SPARQL query.....	32

MODELS AND ALGORITHMS FOR REGULARIZING HETEROGENEOUS WEB TABLES

David W. Embley, Mukkai Krishnamoorthy, George Nagy, Sharad Seth

Abstract:

Despite decades of research, fully automated end-to-end table processing—converting tables intended for human comprehension into a machine representations intended for query processing—remains elusive. Based on a formalization of well-formed tables, we proffer an algorithmic solution to end-to-end table processing for a large class of human-readable tables. The proposed algorithms transform well-formed tables to a canonical table format that maps easily to a variety of industry-standard data stores for query processing. The algorithms segment table regions based on the unique indexing of the data region by header paths, locate header paths, classify table cells, and factor header category structures of two-dimensional as well as the less common multi-dimensional tables. Experimental evaluations provide evidence to substantiate the algorithmic approach to processing heterogeneous tables currently found on the web. As demonstrable results, the algorithms generate queryable relational database tables and semantic-web triple stores. Application of our parameter-free algorithms to 200 heterogeneous tables that have resisted alternative approaches shows that the algorithmic solution automates end-to-end table processing.

Keywords: document analysis, table segmentation, table analysis, table header factoring, relational tables, table headers, table queries

1. INTRODUCTION

Tables provide a convenient and succinct way to communicate data of interest to human readers. Tables are not, however, inherently amenable to machine-based search and query. Automating the process of converting the content of human-readable tables to a queryable store of machine-manipulatable assertions has been and remains an important goal.

Research in document image analysis suggests that there is a natural progression from source document images to a searchable database via “physical” and “logical” layout analysis. In the case of tables, physical analysis must assign literal content to cells laid out on a grid. Logical analysis determines the indexing relationship between header cells and data cells. The indexing structure can be readily converted to any appropriate machine-queryable representation such as relations in a relational database or subject-predicate-object fact assertions in a semantic web triple store. We propose here a complete and coherent table-processing framework to accomplish all of these tasks. The exemplary table in Fig. 1.1 will serve to illustrate the analysis of physical and logical layout and the assertion of facts in machine-queryable form.

1 Official development assistance.

Country	Million dollar					Percentage of GNI				
	2007	2008	2009	2010*	2011*	2007	2008	2009	2010*	2011*
Norway	3 735	4 006	4 081	4 580	4 936	0.95	0.89	1.06	1.10	1.00
Denmark	2 562	2 803	2 810	2 871	2 981	0.81	0.82	0.88	0.91	0.86
Finland	981	1 166	1 290	1 333	1 409	0.39	0.44	0.54	0.55	0.52
Sweden	4 339	4 732	4 548	4 533	5 606	0.93	0.98	1.12	0.97	1.02
Belgium	1 951	2 386	2 610	3 004	2 800	0.43	0.48	0.55	0.64	0.53
France	9 884	10 908	12 602	12 915	12 994	0.38	0.39	0.47	0.50	0.46
Greece	501	703	607	508	331	0.16	0.21	0.19	0.17	0.11
Ireland	1 192	1 328	1 006	895	904	0.55	0.59	0.54	0.52	0.52
Italy	3 971	4 861	3 297	2 996	4 241	0.19	0.22	0.16	0.15	0.19
Luxembourg	376	415	415	403	413	0.92	0.97	1.04	1.05	0.99
Netherlands	6 224	6 993	6 426	6 357	6 324	0.81	0.80	0.82	0.81	0.75
Portugal	471	620	513	649	669	0.22	0.27	0.23	0.29	0.29
Spain	5 140	6 867	6 584	5 949	4 264	0.37	0.45	0.46	0.43	0.29
United Kingdom	9 849	11 500	11 283	13 053	13 739	0.36	0.43	0.51	0.57	0.56
Switzerland	1 685	2 038	2 310	2 300	3 086	0.38	0.44	0.45	0.40	0.46
Germany	12 291	13 981	12 079	12 985	14 533	0.37	0.38	0.35	0.39	0.40
Austria	1 808	1 714	1 142	1 208	1 107	0.50	0.43	0.30	0.32	0.27
Canada	4 080	4 795	4 000	5 209	5 291	0.29	0.33	0.30	0.34	0.31
United States	21 787	26 437	28 831	30 353	30 745	0.16	0.18	0.21	0.21	0.20
Japan	7 697	9 601	9 457	11 021	10 604	0.17	0.19	0.18	0.20	0.18
South Korea	696	802	816	1 174	1 321	0.07	0.09	0.10	0.12	0.12
Australia	2 669	2 954	2 762	3 826	4 799	0.32	0.32	0.29	0.32	0.35
New Zealand	320	348	309	342	429	0.27	0.30	0.28	0.26	0.28
OECD/DAC¹ countries total	104 206	121 954	119 778	128 465	133 526	0.27	0.30	0.31	0.32	0.31

¹ DAC-countries are members of OECD's Development Assistance Committee.

Source: OECD.

Fig. 1.1. An exemplary table, used as a running example throughout the paper.

- Physical Layout.** Tables have a grid structure. Every literal (word, phrase, or numerical value) has a row and a column coordinate. In Fig. 1.1 as in most tables, the data values form a natural grid. When spanning header labels (*Country*, *Million dollar*, and *Percentage of GNI* in Fig. 1.1) are replicated into the cells they span, header labels together with data values also form a grid of cells. Because we also process table titles, footnotes, and other notes associated with tables, we assign them to the grid of cells too. We treat these auxiliary components as spanning cells and replicate them across the row in which they appear. Our processing chain starts with a grid, as described here, because satisfactory methods have already been developed for converting scanned, ASCII, HTML, and searchable PDF tables to a grid of cells in spite of the variety of framing, partial ruling, typeface, color scheme, and cell formatting details. Explicit distinctions between cells containing table title, data values, row and column headers, and footnotes, however, are totally absent in our initial grid representation. Furthermore, there are no rulings that might indicate divisions between data values and other parts of a table, and cell content is just text without color or font formatting. Surprisingly, this lossy representation of an original table suffices to automatically extract the fact assertions stated in a table.
- Logical Layout.** Starting with a table as a grid of text-filled or empty cells, we locate its data region, header regions, and auxiliary regions containing its title and any footnotes and other

associated commentary. We then reveal its indexing structure in terms of categories and an ordered list of category paths for each data cell. The table in Fig. 1.1 has three hierarchical header categories: (*Country (Norway, Denmark, ...)*), (*Year (2007, 2008, ...)*), and (*Development Assistance (Million dollar, Percentage of GNI)*). The index for each data value comprises one header path from each category tree. The upper-left data value in the table in Fig. 1.1, 3 735, for example, is indexed by: (*Country.Norway, Year.2007, Development_Assistance.Million_dollar*). This representation mirrors Wang’s formalization of indexing in tables^[1], which maps a 2-D grid constituting a table into a multi-dimensional array with coordinates corresponding to the categories, i.e., a data cube.

- **Fact Assertions.** The final output of our table-processing work is a collection of fact assertions, represented as relational-database tables and also as subject-predicate-object triples in a semantic-web standard. Each data value in a table makes a fact assertion. The assertion for the data value 3 735 in the table in Fig. 1.1, is: The *Country Norway* in *Year 2007* provided *Development Assistance* in the amount of 3 735 *Million dollars*. Our table-processing system yields these assertions in a form that can be queried with standard query languages—SQL for relational-database tables and SPARQL for semantic-web triples. Our table-processing system also identifies auxiliary information, comprising titles, footnotes, footnote markers and references, and notes, and turns their existence into fact assertions, which can then be queried as such.

Physical/logical-layout and fact-assertion representations are appropriate for almost all common tables. Whereas most previous work addresses only specific types of tables (scanned hardcopy tables or ASCII tables or spreadsheets or web tables) and generally ignored embedded auxiliary data (titles, footnotes, and notes), we cover all of these. A source table may be any file representation that allows rendering (printing or displaying) the essential characteristics of a source table in a form suitable for a human reader, where layout, rulings and typesetting are often used to reveal the intrinsic relationship between headers and content cells. We do not deal here with concatenated (composite) tables, nested tables (tables whose data cells may themselves be tables), tables containing graphic data, or “egregious” tables (those not laid out on a grid with headers above and to the left).

Although most research in document processing is experimental, our table-processing work makes several theoretical contributions that have immediate practical applications. We provide

- (1) a formal (block grammar) definition of well-formed tables that can be used for analysis of most human-readable tables;
- (2) an automatic transformation of well-formed tables to a new canonical table format via:
 - a. segmenting table regions by algorithmic data cell indexing,
 - b. factoring header paths into categories by algorithmic header analysis, and
 - c. generating queryable canonical relational tables and semantic-web triple stores.

After reviewing relevant prior research in Section 2, we present in Section 3 classical (printing and publishing) table terminology and formalize well-formed tables in terms of a block grammar. We explain how our table-processing software segments and classifies cells in Section 4 and finds categories, assigns indexes for data cells and produces canonical tables in Section 5. In Section 6, we validate our work by

comparing a ground truth over a collection of tables with (1) automatic table-region segmentation and cell classification in Section 4 and (2) category-tree indexing in Section 5. Section 7 shows SQL and SPARQL queries to demonstrate that the human readable tables are indeed converted into data stores of machine-queryable fact assertions. In Section 8, we draw conclusions and point to further research opportunities.

2. PRIOR WORK

Tables are a well-established means of presenting semi-structured information where related values occupy the same row or columns. Tables are prevalent in newspapers (weather, financial reports, polls, sports statistics), in technical journals, and in scientific text. They have also migrated to the web and can be accessed through browsers in a variety of sizes and formats. It has been a persistent goal of research to make the information contained in human-readable tables accessible to algorithmic queries. Different methods have been found appropriate for bitmapped images of scanned, digitally photographed or faxed hardcopy tables, ASCII tables found in email messages or in early computer-generated documents, searchable or raw PDF files, and both manually coded and automatically generated spreadsheet and HTML tables. We describe previous table models and summarize published methods of transformation between representations (often called *table recognition*, *table interpretation*, *table understanding*, or *table data extraction*).

This literature review has four parts. We first review X. Wang's pioneering research which has long guided our approach to table understanding. In the second subsection we point out research that justifies our claim that table spotting, table isolation and conversion of source tables to grid tables are no longer major obstacles to table understanding. Next we review research that aims, like ours, at higher-level, logical analysis of tables. Finally, we summarize our own previous work that underlies our current endeavors.

2.1 Wang Tables

X. Wang regarded tables as an abstract data type [1]. She formalized the distinction between physical and logical structure in the course of building X-Table for practical table composition in a Unix X-Windows environment. She defined *layout structure* as the presentation form of a table, and *logical structure* as a set of labels and entries. Labels are assigned to hierarchies of categories and sub-categories, and each entry is associated with one label from each of the categories. The number of categories defines the dimensionality of the abstract table.

Wang formulated the logical structure of a table in terms of category trees corresponding to the header structure of the table [1]. "Wang categories," a form of multidimensional indexing, are defined implicitly by the 2-D geometric indexing of the data cells by row and column headers. The index of each data cell is unique (but it may be multidimensional and hierarchical in spite of the flat, two-dimensional physical layout of the table). She used the object-oriented dot notation, *label1.label2.label3.entry*, to represent a path in the category tree from headers to data cells. Thus, for example, Wang would identify the three

category trees in Fig. 1.1 for countries, years, and development assistance, and index each data cell as a triple of paths, one for each category tree.

2.2 Physical Structure Extraction (Low-level Table Processing).

In printed tables, boxing, rules, or white space alignment are used for separating cell entries. In one of the earliest works, Laurentini and Viada extracted cell corner coordinates from the ruling lines [2]. Image processing techniques for the extraction of physical structure from scanned tables include Hough Transforms [3], run-length encoding [4], word bounding boxes [5], and conditional random fields (CRF) [6]. Hirayama presented an algorithm for segmenting partially-ruled tables into a rectangular lattice [7]. Handley's method of iterative identification of cell separators successfully processed large, complex, fully-lined, semi-lined, and unruled tables with multiple lines of text per cell [8]. Zuyev used connected components, and projection profiles to identify the cell contents for an OCR system [9]. The notion of converting paper tables into Excel spreadsheets dates back at least to 1998 [10]. Early research in table processing suffered from the isolation of the graphics research community from the OCR community. Our methods are applicable to scanned and segmented printed tables even without accurate OCR. Current OCR products can locate tables on a printed page and convert them into a designated table format. Most desktop publishing software has provisions for the inter-conversion of tables and spreadsheets.

Less attention has been focused on ASCII table analysis, where the structure must often be discovered from the correlation of text blocks on successive lines. Grid structure is preserved by spacing, although vertical separators ("|") and extra New Line symbols for blank rows or rows filled with dashes are sometimes used. Pyreddy and Croft demonstrated results on over 6000 tables from the Wall Street Journal [11]. T-Recs clustered words for bottom-up structural analysis of ASCII tables [12]. Hu et al. explored row and column alignment via directed acyclic attribute graphs [13]. Work on such tables has diminished since the development of XML for communicating structured data without sacrificing ASCII encoding.

The web contains many millions of tables encoded in HTML like the exemplary table in Fig. 1.1. Its underlying HTML is in Fig. 2.1, which shows that the cell contents are listed between <th> tags for table headers and <td> tags for table data which are included between <tr> table-row tags. Table captions, titles and headers are also explicitly tagged. The tagging makes the extraction of the table's underlying grid structure from its customary HTML representation relatively simple. Figs 2.2 and 2.3 show the limited information retained when Excel converts the HTML table in Fig. 1.1 and Fig. 2.1 into CSV format. In the CSV file (1) the labels of spanning cells are followed by delimiters (here commas) that form a full grid of cells; and (2) all type and cell formatting and ruling lines are removed. Excel displays files with an equal number of delimiters between new-line symbols as a table. Changes to the display, like the increased width of the first column in Fig. 2.3, are not retained when the file is stored in CSV format.

```
<html>
...
```

```
<!-- START TABELL -->
```



```

<tableborder=0width=98%>

<captionclass=tabellttittel>
<spanclass=tabellnummer>1</span>
  Official development assistance.
</caption>

<thead>
<tr>
<tdclass=sepcolspan=80></td>
</tr>

<tr>
<th rowspan=2class=level11>Country</th>
<thclass=multispancolspan=5style=border-bottom:
1px #000000 solid; >Million dollar</th>
<thclass=multispancolspan=5style=border-bottom:
1px #000000 solid; >Percentage of GNI</th>
</tr>

<tr>
<th>2007</th>
<th>2008</th>
<th>2009</th>
<th>2010*</th>
<th>2011*</th>
<th>2007</th>
<th>2008</th>
<th>2009</th>
<th>2010*</th>
<th>2011*</th>
</tr>

<tr>
<tdclass=sepcolspan=80></td>
</tr>
</thead>

<tbody>

<tr>
<tdclass=level11>Norway</td>
<td>3&nbsp;735</td>
<td>4&nbsp;006</td>

<td>4&nbsp;081</td>
<td>4&nbsp;580</td>
<td>4&nbsp;936</td>
<td>0.95</td>
<td>0.89</td>
<td>1.06</td>
<td>1.10</td>
<td>1.00</td>
</tr>
... MANY MORE COUNTRIES AND THEIR DATA
<tr>
<tdclass=label>OECD/DAC<sup>1</sup> countries
total</td>
<tdclass=sum>&nbsp;104&nbsp;206</td>
<tdclass=sum>&nbsp;121&nbsp;954</td>
<tdclass=sum>&nbsp;119&nbsp;778</td>
<tdclass=sum>&nbsp;128&nbsp;465</td>
<tdclass=sum>&nbsp;133&nbsp;526</td>
<tdclass=sum>0.27</td>
<tdclass=sum>0.30</td>
<tdclass=sum>0.31</td>
<tdclass=sum>0.32</td>
<tdclass=sum>0.31</td>
</tr>

<tr>

</tbody>
</table>

<tableborder=0width=98%>
<tr><tdclass=footnotevalign=topwidth=1%><sup>1</s
up>&nbsp;</td><tdclass=footnotevalign=top> DAC-
countries are members of OECD's Development
Assistance Committee.</td></tr>
<tr><tdclass=kildecolspan=2><b>Source:&nbsp;</b>0
ECD.</td></tr>
</table>

<!-- SLUTT TABELL -->
...
</html>

```

Fig. 2.1. Source code of the HTML table in Fig. 1.1. Less than one fifth of the 446 lines are shown.

(URL: http://www.ssb.no/a/english/kortnavn/uhjelpoecd_en/tab-2012-05-15-01-en.html, accessed Jan. 2015).

```

Expenditures on development aid in the OECD countries.....
1 Official development assistance.....
Country,Million dollar,.....Percentage of GNI
,2007,2008,2009,2010*,2011*,2007,2008,2009,2010*,2011*.....
Norway,3 735,4 006,4 081,4 580,4 936,0.95,0.89,1.06,1.1,1.....
Denmark,2 562,2 803,2 810,2 871,2 981,0.81,0.82,0.88,0.91,0.86.....
Finland,981,1 166,1 290,1 333,1 409,0.39,0.44,0.54,0.55,0.52.....
Sweden,4 339,4 732,4 548,4 533,5 606,0.93,0.98,1.12,0.97,1.02.....
Belgium,1 951,2 386,2 610,3 004,2 800,0.43,0.48,0.55,0.64,0.53.....
France,9 884,10 908,12 602,12 915,12 994,0.38,0.39,0.47,0.5,0.46.....
Greece,501,703,607,508,331,0.16,0.21,0.19,0.17,0.11.....
Ireland,1 192,1 328,1 006,895,904,0.55,0.59,0.54,0.52,0.52.....
Italy,3 971,4 861,3 297,2 996,4 241,0.19,0.22,0.16,0.15,0.19.....
Luxembourg,376,415,415,403,413,0.92,0.97,1.04,1.05,0.99.....
Netherlands,6 224,6 993,6 426,6 357,6 324,0.81,0.8,0.82,0.81,0.75.....
Portugal,471,620,513,649,669,0.22,0.27,0.23,0.29,0.29.....
Spain,5 140,6 867,6 584,5 949,4 264,0.37,0.45,0.46,0.43,0.29.....
United Kingdom,9 849,11 500,11 283,13 053,13 739,0.36,0.43,0.51,0.57,0.56.....
Switzerland,1 685,2 038,2 310,2 300,3 086,0.38,0.44,0.45,0.4,0.46.....
Germany,12 291,13 981,12 079,12 985,14 533,0.37,0.38,0.35,0.39,0.4.....
Austria,1 808,1 714,1 142,1 208,1 107,0.5,0.43,0.3,0.32,0.27.....
Canada,4 080,4 795,4 000,5 209,5 291,0.29,0.33,0.3,0.34,0.31.....
United States,21 787,26 437,28 831,30 353,30 745,0.16,0.18,0.21,0.21,0.2.....
Japan,7 697,9 601,9 457,11 021,10 604,0.17,0.19,0.18,0.2,0.18.....
South Korea,696,802,816,1 174,1 321,0.07,0.09,0.1,0.12,0.12.....
Australia,2 669,2 954,2 762,3 826,4 799,0.32,0.32,0.29,0.32,0.35.....
New Zealand,320,348,309,342,429,0.27,0.3,0.28,0.26,0.28.....
OECD/DAC1 countries
total,104 206,121 954,119 778,128 465,133 526,0.27,0.3,0.31,0.32,0.31.....
1 DAC-countries are members of OECD's Development Assistance
Committee.....
Source: OECD.....
Explanation of symbols.....

```

Fig. 2.2 Text (Notepad) display of CSV file after Import from the HTML in Fig. 2.1.

Expenditures on development aid in the OECD countries										
1 Official development assistance.										
Country	Million dollar					Percentage of GNI				
	2007	2008	2009	2010*	2011*	2007	2008	2009	2010*	2011*
Norway	3 735	4 006	4 081	4 580	4 936	0.95	0.89	1.06	1.1	1.1
Denmark	2 562	2 803	2 810	2 871	2 981	0.81	0.82	0.88	0.91	0.86
Finland	981	1 166	1 290	1 333	1 409	0.39	0.44	0.54	0.55	0.52
Sweden	4 339	4 732	4 548	4 533	5 606	0.93	0.98	1.12	0.97	1.02
Belgium	1 951	2 386	2 610	3 004	2 800	0.43	0.48	0.55	0.64	0.53
France	9 884	10 908	12 602	12 915	12 994	0.38	0.39	0.47	0.5	0.46
Greece	501	703	607	508	331	0.16	0.21	0.19	0.17	0.11
Ireland	1 192	1 328	1 006	895	904	0.55	0.59	0.54	0.52	0.52
Italy	3 971	4 861	3 297	2 996	4 241	0.19	0.22	0.16	0.15	0.19
Luxembourg	376	415	415	403	413	0.92	0.97	1.04	1.05	0.99
Netherlands	6 224	6 993	6 426	6 357	6 324	0.81	0.8	0.82	0.81	0.75
Portugal	471	620	513	649	669	0.22	0.27	0.23	0.29	0.29
Spain	5 140	6 867	6 584	5 949	4 264	0.37	0.45	0.46	0.43	0.29
United Kingdom	9 849	11 500	11 283	13 053	13 739	0.36	0.43	0.51	0.57	0.56
Switzerland	1 685	2 038	2 310	2 300	3 086	0.38	0.44	0.45	0.4	0.46
Germany	12 291	13 981	12 079	12 985	14 533	0.37	0.38	0.35	0.39	0.4
Austria	1 808	1 714	1 142	1 208	1 107	0.5	0.43	0.3	0.32	0.27
Canada	4 080	4 795	4 000	5 209	5 291	0.29	0.33	0.3	0.34	0.33
United States	21 787	26 437	28 831	30 353	30 745	0.16	0.18	0.21	0.21	0.2
Japan	7 697	9 601	9 457	11 021	10 604	0.17	0.19	0.18	0.2	0.18
South Korea	696	802	816	1 174	1 321	0.07	0.09	0.1	0.12	0.12
Australia	2 669	2 954	2 762	3 826	4 799	0.32	0.32	0.29	0.32	0.35
New Zealand	320	348	309	342	429	0.27	0.3	0.28	0.26	0.28
OECD/DAC1 countries total	104 206	121 954	119 778	128 465	133 526	0.27	0.3	0.31	0.32	0.31
1	DAC-countries are members of OECD's Development Assistance Committee.									
Source: OECD.										
Explanation of symbols										

Fig. 2.3 Spreadsheet (Excel) display of CSV file after Import from the HTML in Fig. 2.1.

2.3 Logical Structure Extraction (High-level Table Processing)

Gatgebauer et al. presented a geometric approach to table extraction from arbitrary web pages based on the spatial location of table elements prescribed by the DOM tree [14]. They formulated a “visual table model” of nested rectangular boxes derived from Cascading Style Sheets. They applied spatial reasoning—primarily based on adjacency topology and Allen interval relations—to their visualization model in order to determine the final box structure. They also conducted some semantic analysis with a known or assumed list of keywords. Their interpretation consists of XML-tagged generalized n-tuples. They evaluated several steps of their process on 269 web pages with 493 tables and reported 48% precision and 57% recall.

Amano and Asada have published a series of papers on graph grammars based on box adjacency for “table-form” documents [15]. Their grammars encode the relationship between “indicator,” “example,” and data boxes. Similarities between table and form processing were already emphasized by Bing et al. [16] and Kieninger and Dengel [17]. General grammar-based approaches that can be specialized to forms and tables have been demonstrated on large data sets [18, 19, 20].

A group headed by T. Watanabe aimed at learning the various types of information necessary to interpret a ruled scanned table [21]. They used a training set of diverse tables to populate a “Classification Tree.” The nodes of the tree are “Structure Description Trees” that can interpret a specific family of tables. In the operational phrase, new classification nodes and tree structure descriptions are added for unrecognized tables.

Shamalian et al. demonstrated a model-based table reader for reading batches of similar tables [22]. Their model specifies the location of the data cells, thus obviating the need to interpret headers either syntactically or semantically.

In the last several years, an active and inventive group at Google, possibly inspired by Halevy, Norvig, and Pereira [23], collected and analyzed millions of tables harvested from the web [24, 25, 26]. Their general approach has been to treat table rows as tuples with attributes specified by the top row. Visual verification of their results has necessarily been restricted to much smaller samples. Extending this work to tables more complex than simple relational tables, Adelfio and Samet leveraged the principles of table construction to generate interpretations for spreadsheet and HTML tables²⁷. Using Conditional Random Fields, they classified each table row as: *header*, *data*, *title*, *group header*, *aggregate*, *non-relational metadata*, or *blank*. With their test set of 1048 spreadsheet tables and 928 HTML tables, they achieved an accuracy of 76.0% for classifying header and data rows for spreadsheet tables and 85.3% for HTML tables, and for classifying all rows, 56.3% and 84.6% respectively. In contrast to the work of the Google group and of Adelfio and Samet, we treat row headers as first-class objects on a par with column headers and depend on indexing properties rather appearance features for further analysis.

A series of papers culminating in V. Long’s doctoral thesis [28] analyzes a large sample of tables from Australian Stock Exchange financial reports. An interesting aspect of this work is the detection and verification of the scope and value of *aggregates* like totals, subtotals, and averages. The analysis is

based on a blackboard framework with a set of cooperating agents. This dissertation has a good bibliography of table papers up to 2009. Other work dealing with aggregates in tables includes [29].

Already in 1997, Hurst and Douglas advocated converting tables into relational form: “Once the relational structure of the table is known it can be manipulated for many purposes.” [30]. Hurst provided a taxonomy of category attributes in terms of *is-a*, *part-of*, *unit-is*, *quantity-is*. He pointed out that the physical structure of a table is somewhat analogous to syntax in linguistic objects. He also emphasized the necessity and role of natural language analysis for table understanding, including the syntax of within-cell strings [31]. Hurst’s dissertation contains a wealth of interesting examples of tables [32].

Hurst’s work was reviewed and augmented in Costa e Silva et al. [33], who analyzed prior work in detail in terms of contributions to the tasks of *table location*, *segmentation*, *functional analysis* (tagging cells as data or attribute), *structural analysis* (header index identification), and *interpretation* (semantics). Costa E Silva’s research group also provides a clear distinction between tables, forms, and lists. The ultimate objective of this group is the operational analysis of financial tables with feedback between the five tasks based on confidence levels.

Kim and Lee reviewed web table analysis from 2000 to 2006 and found logical hierarchies in HTML tables using cell formats and syntactic coherency [34]. They extracted the table caption and divided spanning cells correctly. Like us, but in contrast to many other researchers, they handled vertical and horizontal column headers symmetrically.

The TARTAR (*Transforming ARbitrary Tables into fRames*) system developed by Pivk et al. has objectives similar to ours: “The input to the system is semi-structured information in the form of *arbitrary* (HTML, PDF, EXCEL, etc.) tables.” [35]. However, in the cited paper, the authors demonstrated their work only on HTML tables. Their “cleaned and canonicalized” matrix representation is similar to our grid table. Downstream analysis and region segmentation proceeded, however, on the basis of cell formats (letters, numerals, capitalization, and punctuation) rather than indexing properties. The cells were functionally labeled in a manner similar to Hurst as *access* or *data* cells and assembled into a *Functional Table Model*. An attempt was made for semantic interpretation of strings using WordNet. The final output was a semantic (F-logic) frame. The complex evaluation scheme that was presented and applied to 158 HTML tables was hampered by human disagreement over the description of the frames.

Chen and Cafarella recently presented a table-processing system that transforms spreadsheet tables into relational database tables³⁶. Like Adelfio and Samet[27] and Pinto et al.[6], they adapt a CRF to label each row with one of four labels: title, header, data, and footnote, using similar row features. (Rows labeled as “data” also include the cells in the row header, hence to distinguish between the two, they must assume, unlike us, that the data region is purely numeric.) Their hierarchy extractor builds parent-child candidates of cells in the header region using formatting, syntactic, and layout features. The candidate list is pruned by an SVM classifier that enforces the resulting set of candidate pairs to be

cycle-free. In our algorithmic approach to table processing, the resulting structure is guaranteed to be cycle-free by construction.

2.4 Our earlier work

A review of early work on table processing, a catalog of obstacles, and a collection of tables that stretch the very definition of *table* was presented in [37]. Examples of human ambiguity in table interpretation were discussed in [38]. The extent to which semantic information is revealed by table structure was explored in [39]. We compiled a comprehensive survey of table processing in 2006 [40]. The notion of a WoK (Web-of-Knowledge) similar to the Yahoo researchers' web-of-concepts [41] was proposed in [42]. Input tables were matched with known conceptualizations in an attempt to interpret them in [43]. Information extraction from *sibling tables* with identical headers was demonstrated in [44]. A taxonomy of tables based on the geometric relationship of tabular structures to isothetic tessellations and to X-Y trees was proposed in [45]. A machine learning approach to segmentation of a grid table appeared in [46]. Algorithms for turning web tables into relational tables by recovering and factoring header paths were presented in [47]. Experience with VeriClick, an interactive tool for table segmentation, was described in [48]. Algorithmic table segmentation, based on the fundamental indexing property, was demonstrated in [49]. Several other papers, mostly reporting experiments on various aspects of table processing, are referenced in these earlier publications of our work.

In addition to the IEA/AIE'11 [47] and ICDAR'13 [49] papers mentioned above, three precursors to this paper have recently appeared in conference proceedings. At the 2014 Document Analysis Systems workshop, we reported on our initial, automatic end-to-end conversion of web tables to relational databases [50]. We showed SQL queries on HTML tables imported into MS-Access at ICPR 2014 [51]. At the 2015 Conference on Document Analysis and Recognition, our focus was on comparing the headers of category hierarchies to reveal commonalities among tables [52].

The current paper combines and significantly expands these precursors: (1) Our literature review (Section 2) more specifically compares prior work to our efforts. (2) We describe well-formed tables in terms of a block algebra that formalizes the conventional typesetting practices of the printing and publishing industry that underlie web tables [53] (Section 3). (3) We present our MIPS (Minimum Indexing Point Search) algorithm (Section 4) and our category-tree extraction algorithm (Section 5) in terms of the new well-formed table formalization. (4) Exercising these algorithms on a collection heterogeneous table, we present detailed analyses of the required header modifications (Section 6). We transform algorithmically-discovered table content to semantic-web triple stores and to relational databases, and we execute both SQL and SPARQL queries over several hundred automatically processed HTML tables (Section 7).

3. HUMAN READABLE TABLES

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*. In this section, we first informally

present the generally accepted view of tables. We then specify a visual schematic model of the well-formed tables that we can process. The model is formalized in a 2-D interval algebra in terms of the inherent spatial constraints.

3.1 What is a table?

Tables are universally used for presenting data logically organized into two or more *categories*: *Country*, *Year*, and *development assistance* in Fig. 1.1. Their *data cells* are laid out on a uniform grid. Each data cell is indexed by its row and column headers. In conventional printing terminology, the *principal zone* of a table comprises regions called *stub* (or, *stub head*), *row header*, *column header*, and *data*. Auxiliary information, such as the table title, notes, and footnotes appear outside this principal zone. Most authors consider an $m \times n$ table to have m rows and n columns of data 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. The stub may be empty or augment information carried by row or column header, or the table title. In Fig. 1.1, the stub contains *Country*, a row header.

A single category (*Country*) can be indexed by a flat header (*Chad, Tunisia, China, India, Japan ...*), 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 by repeated labels. In Fig. 1.1 two categories, *development assistance* and *Year* have the hierarchical display: (*Million dollar (2007, ..., 2011*)*, *Percentage of GNI (2007, ..., 2011*)*).

Since horizontal and vertical table organization is symmetric and permutable, the number of possible table layouts increases combinatorially with the number of categories and the number of their content labels. 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.

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. In Fig. 1.1, for example, the label *Year* does not appear (and could offend some readers if it did). 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. In Fig. 1.1, *Country*-category root is in the stub header, and the root for the *development assistance* category is in the table title.

In a *well-formed table (WFT)*, every data cell is uniquely indexed by its row and column *header paths*, which are respectively left of and above the data region. 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. The concatenated path *development_assistance.Million_dollar.Year.2007* is the column header for the first data column in Fig. 1.1. When tables are well-formed, they are generally amenable to automated data extraction using only structural information.

Egregious tables (those that are not well-formed) may not puzzle human readers, but they challenge algorithms and require external context to extract values with their applicable indexes. The genetic code tables in Fig. 3.1, for example, may have a much better layout for human understanding than if they were laid out as WFTs. Nevertheless, it is easy for humans to recast such tables as WFTs, although the task is far from trivial for machines. In Fig. 3.1a, we can move the rightmost column of codon labels and the *3rd base in codon* spanning label to the left, making its layout conform to the layout of a WFT. Fig. 3.1b requires recasting the information in a form similar to the reformulated table of Fig. 3.1a and making explicit the implicit understanding of which codon is first, second, and third. The periodic table is another classic example: its layout succinctly captures element properties for an informed human reader, which can be cast into the layout of a WFT by listing the element symbols as row headers and providing column header labels for each of the depicted element properties (Atomic Number, Group, Period, ...). Further samples of egregious tables appear in [37].

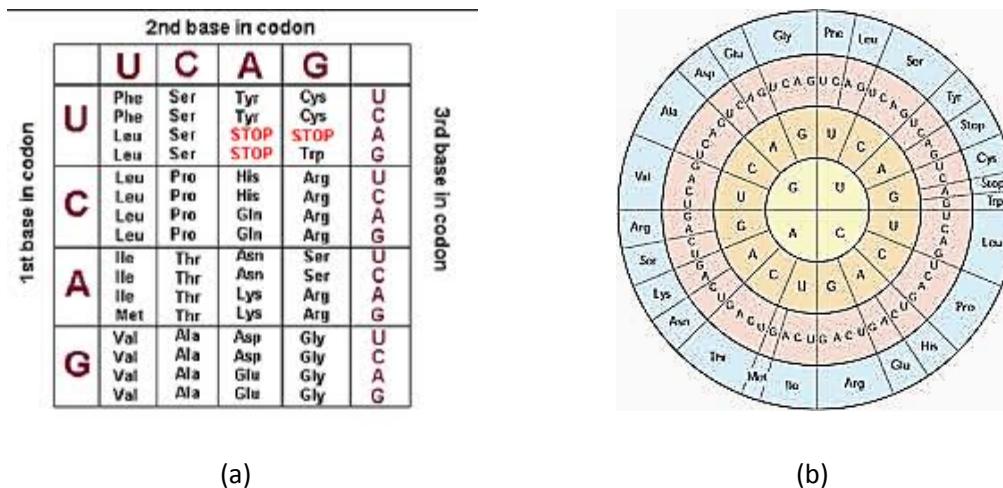


Fig. 3.1. Genetic coding tables.

3.2 Well Formed Tables: Formal Characterization

Fig. 3.2a shows a visual model of the WFTs we process. The principal spatial constraints are that the *RowHeader* must be to the left and aligned with the *Data* region, and that the *ColumnHeader* must be above and also aligned with the *Data* region. The *TableTitle* is in the top row. *Footnotes* along with their preceding footnote marker must be below the *RowHeader* and the *Data* and cannot share their row with anything else. The corresponding reference to the footnote, matching the footnote marker, may occur in any cell above the footnote. *Notes* provide information about the source or dissemination of the data (e.g., *Source: OECD* in Fig. 1.1). *Notes* may occur above or below the *StubHeader-ColumnHeader* or below the *Footnotes*. Empty rows or columns may occur anywhere except at the top or left (a CSV

requirement), but are most common on the far right or below the table. They can be deleted without loss of information, yielding the simplified model in Fig. 3.2b.

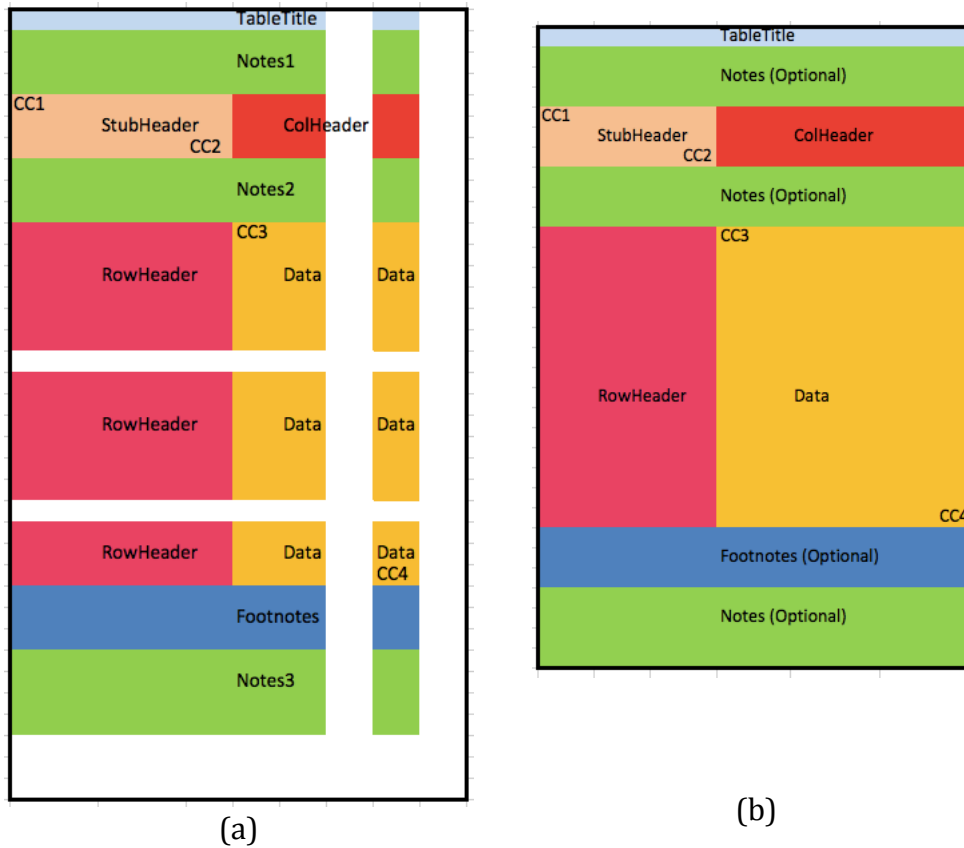


Fig. 3.2. Visual WFT model: (a) with and (b) without blank rows and columns.

Critical cells ($CC1$, $CC2$, $CC3$, $CC4$) delineate regions. In a WFT every critical cell must appear in the grid. As Fig. 3.2 shows, $CC1$ and $CC2$ demarcate the *StubHeader* and $CC3$ and $CC4$ demarcate the *Data* region. Furthermore, in combination with one another, these critical cells also demarcate both the *ColHeader* and *RowHeader* regions. Letting row r_i and column c_j be the coordinates of critical cell CC_i , a WFT satisfies the following constraints: $r_1 \leq r_2 < r_3 \leq r_4$ and $c_1 \leq c_2 < c_3 \leq c_4$. These constraints guarantee that the *ColHeader* and *RowHeader* regions properly align with the *Data* region and that the *Data* region is not degenerate. A single row or column of data is acceptable, provided both row and column headers exist. To complete our formalization of a WFT, we add “region-level constraints,” which further constrain how regions align, and “cell-level constraints,” which constrain how cells within and across regions align.

Region-level Constraints

The region-level spatial constraints can be formalized using a block algebra ^[54], which is a spatial application of Allen’s interval algebra ^[55]. We derive the constraints for the simplified visual table model of the simplified model of Fig. 3.2b.

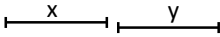
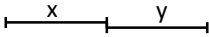
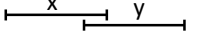
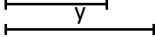
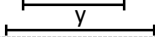

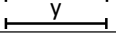
Relation	Graphic Illustration
$x \text{ } b \text{ } y$ (before)	
$x \text{ } m \text{ } y$ (meets)	
$x \text{ } o \text{ } y$ (overlaps)	
$x \text{ } s \text{ } y$ (starts)	
$x \text{ } d \text{ } y$ (during)	
$x \text{ } f \text{ } y$ (finishes)	
$x \text{ } eq \text{ } y$ (equals)	

Fig. 3.3. The relations of Allen's interval algebra.

Fig. 3.3 shows the 7 basic relations of interval algebra. When applying the algebra to the constraints on the blocks of Fig 3.2b, the intervals are derived independently for the projections of the blocks along the rows and columns. Hence constraints are expressed as pairs, with the two elements of each pair corresponding to the two projections. To illustrate, consider the spatial constraints between the blocks, *TableTitle* and *ColHeader* in Fig. 3.2b. Horizontally, the *ColHeader* block *finishes* the *TableTitle* block, which can be stated as the relation *ColHeader* *f* *TableTitle* or as its inverse *TableTitle* *fi* *ColHeader*. Vertically, *TableTitle* appears *before* *ColHeader*, but if the optional *Notes* block is missing, the two blocks can also *meet*. Together, the horizontal and vertical relationships can be expressed as a disjunction of (horizontal, vertical) relation pairs: *TableTitle* (*fi*, *b*) *ColHeader* \vee *TableTitle* (*fi*, *m*) *ColHeader*. In Allen's *network representation*, the two relation pairs become labels on the edge directed from the *TableTitle* node to the *ColHeader* node, with the interpretation that the spatial constraints between the two blocks can be satisfied by *any* of the relations on the edge label.

The network representation applies to every pair of blocks in Fig 3.2b and is shown in a matrix form in Fig. 3.4. In this matrix, for example, the relation pairs (*fi*, *b*), (*fi*, *m*) appear in the column of *TableTitle* and the row of *ColHeader*. Further, the entry in the symmetrical cell (column: *ColHeader*, row: *TableTitle*) will be its inverse, i.e. (*f*, *bi*), (*f*, *mi*). Because of this symmetry relationship, the cell entries in the gray region are not shown.

	TableTitle	StubHeader	ColHeader	RowHeader	Notes	Footnotes	Data
TableTitle	(eq, eq)						
StubHeader	(si, b), (si, m)	(eq, eq)					
ColHeader	(fi, b), (fi, m)	(m, eq)	(eq, eq)				
RowHeader	(si, b)	(eq, b), (eq, m)	(mi, b), (mi, m)	(eq, eq)			
Notes	(eq, m), (eq, b)	(s, mi), (s, m), (s, b)	(f, mi), (f, m), (f, b)	(s, bi), (s, mi), (s, b), (s, m)	(eq, eq), (eq, b)		
Footnotes	(eq, b)	(s, b)	(f, b)	(s, m)	(eq, b), (eq, mi)	(eq, eq)	
Data	(fi, b)	(m, b), (m, m)	(eq, b), (eq, m)	(m, eq)	(fi, b), (fi, m), (fi, bi), (fi, mi)	(fi, mi)	(eq, eq)

Fig. 3.4. Network representation of the spatial constraints of the table model in Fig. 4(b).

Cell-level Constraints

Apart from the among-region structural constraints, a WFT also satisfies the following cell-level constraints related to data cells, header cells, categories, and auxiliary cells comprising titles, notes, and footnotes.

Data Cells

1. Each *DataCell* in a grid table is atomic.
2. Every *DataCell* is indexed by the header cells from every category.

Header Cells

1. Every *HeaderCell* belongs to at least one *HeaderPath*.
2. *DataCell* (r, c) has *RowHeaderPath* $Cell(r, c_1), \dots, Cell(r, c_2)$, where c_1 and c_2 are the column coordinates of CC1 and CC2, i.e., the sequence of horizontal cells in the *RowHeader* region in row r , and has *ColumnHeaderPath* $Cell(r_1, c), \dots, Cell(r_2, c)$, where r_1 and r_2 are the row coordinates of CC1 and CC2, i.e., the sequence of vertical cells in the *ColumnHeader* region in column c .
3. Column (Row) *HeaderPaths* (concatenations of *HeaderPaths* for multi-category headers) uniquely identify a column (row) of data cells.

Auxiliary Cells

1. A *footnote marker* and its associated *footnote* may appear in a single cell or in two adjacent cells.
2. Every *footnote marker* has a corresponding *footnote reference* that may appear in the table title, header or data region.

As WFTs cover most printed, web, and spreadsheet tables, as well as relational database tables displayed in standard form with keys on the left, it is worthwhile to characterize this class of tables in terms of its spatial and logical constraints. The WFT formalization corresponds to the visual model of Fig. 3.2a and characterizes the class of table that we can process with the methods presented below.

4. SEGMENTATION AND CELL CLASSIFICATION

Segmentation consists of locating the critical “corner” cells *CC1* and *CC2* of the stub-header, and *CC3* and *CC4* of the data region, as well as the rows or elementary cells containing the embedded table title, footnotes, footnote marks, footnote references, and miscellaneous notes. Our MIPS (Minimum Indexing Point Search) algorithm finds *CC1* and *CC2*. The underlying assumption is that the row headers and column headers index the data cells. Header indexing requires header cells to be aligned with the data cells they index, as is also required of WFTs. Therefore MIPS transforms near WFTs into WFTs by straightening out any “crooked” header paths.

Although *CC1* and *CC2* are found algorithmically, heuristics are needed to demarcate the top and bottom of the *data* region (indicated by *CC3* and *CC4*) from its surrounding regions. As shown in Section 4.4, the output of the segmentation and cell classification stage is a CSV table in a uniform format with one row for each cell of the source table.

4.1 Header Segmentation

The input to the MIPS algorithm is a CSV table, converted from a web table. Fig 4.1 shows the first seven and last five rows of the exemplary table of Fig. 1.1 converted to CSV format and rendered as a table. We explain MIPS using the pseudo-code of Fig. 4.2 and the exemplary table. As shown in the WFT model (Fig. 3.2), the data region extends to the right of the table. Because the footnote region and bottom notes region need not be indexed, MIPS operates on the portion of the table above the bottom of the data region whose rightmost bottom cell is indicated by *CC4*. This critical cell is found before MIPS is launched by searching from the bottom of the original table for the last filled row with a minority of empty cells (in Fig. 1.1, it is Row 30, with *OECD/DAC* in its first cell). Filled rows are assumed to be part of the data region rather than notes or footnotes rows (which usually have only one or two filled cells). The MIPS algorithm operates on the table from the top row to the row of the *CC4* cell.

In the first step of the algorithm, empty columns and rows beyond the table’s rightmost column and bottom row are deleted. Also, empty rows and columns are labeled as *EMPTY* to indicate that these rows and columns can be ignored during segmentation and classification. They are not deleted because that would interfere with referencing the original cell coordinates and because they sometimes serve as visual clues to focus on certain aspects of the table (e.g., Nordic countries Fig. 1.1).

Separate tables are generated to process row-spanning cells and column-spanning cells. In the second step of the algorithm, the empty cells resulting from splitting spanning cells are filled with the label of the parent spanning cell. The second step also resolves “crooked header paths” by “straightening them out” in a process we call “prefixing,” but we defer a description of this prefixing process until Section 4.2.

The third step finds the Minimum Indexing Point. The algorithm determines *CC1* and *CC2* using the spanning-cell-distributed row- and column versions of the input table, which are also prefixed if necessary. In Fig. 4.1, for example, the spanning-cell headers, *Million dollar* and *Percentage of GNI*, will have been distributed to the empty cells to their right, and *Country* will have been distributed to the empty cell beneath it (although not to the second initially empty cell beneath it since that cell will have

been replaced by *EMPTY* because it is part of a full row of empty cells). The title will also have been distributed to each empty cell to its right, but neither the footnote nor the note about the source will have been distributed since they fall below *CC4*. (No prefixing is required for the exemplary table.) In the first while loop, the algorithm scans from the first row down until it finds unique columns (consisting of vertically aligned cell strings concatenated from the top down). In the grid table in Fig. 4.1 (modified by distributing spanning cells), this does not occur until Row 4, where the vertically aligned cell string sequences *1 Official development assistance – EMPTY – Country – Country, ..., 1 Official development assistance – EMPTY – Percentage of GNI – 2011** for all columns are unique. Rows 1 through 4, thus, become a candidate for the column header.

In the next while loop, the algorithm applies the same procedure to all the rows (columns in the transposed table) below the current column-header candidate. For the table in Fig. 4.1, the first column indexes all the rows to its right and is therefore a row-header candidate. Then, the column(s) of the row-header candidate are removed from the column-header candidate (here, *1 Official development assistance – EMPTY – Country – Country*). The truncated column header could be indexed, possibly, with a fewer number of rows. After the third while loop checks for this eventuality, which does not happen in our example since the years are needed to index the columns, the minimum indexing point is found as the bottom-right corner of *CC2*, the junction point of the two headers.

From the WFT model (Fig. 3.2), it can be seen that the critical cell *CC1* always appears in the first row-header column but not in the first row, which is occupied by the title of the table. Furthermore, there may be additional rows of notes above the *StubHeader*. The column header candidate, however, includes all of these rows. Hence, in order to determine *CC1*, the algorithm scans the column header candidate in the reverse direction to determine the minimum number of rows that are sufficient for indexing. In the exemplary table, it finds that Rows 3 and 4 in Fig. 4.1 constitute a column header with unique (two-row) columns that index all the columns below. In Fig. 4.1 *Million dollar – 2007, ..., Percentage of GNI – 2011** are unique.

1 Official development assistance.										
Country	Million dollar					Percentage of GNI				
	2007	2008	2009	2010*	2011*	2007	2008	2009	2010*	2011*
Norway	3 735	4 006	4 081	4 580	4 936	0.95	0.89	1.06	1.1	1
Denmark	2 562	2 803	2 810	2 871	2 981	0.81	0.82	0.88	0.91	0.86
...										
New Zealand	320	348	309	342	429	0.27	0.3	0.28	0.26	0.28
OECD/DAC1	104 206	121 954	119 778	128 465	133 526	0.27	0.3	0.31	0.32	0.31
1 DAC-countries are members of OECD's Development Assistance Committee.										
Source: OECD.										

Fig. 4.1. Part of the table of Fig. 1.1 in CSV format.

MIPS Algorithm

Input: CSV table with ASCII cell strings

Output: critical cells CC1 and CC2

All rows below CC4 are assumed to have been removed

Remove empty rows below and empty columns to the right of the table

Fill any remaining empty rows and columns with *EMPTY*

Fill all other empty cells with *BLANC*

Call this table *Table_1*; *# it has Nrows rows and Ncols columns*

Table_2 = Table_1; # Table_1 used for column indexing, Table_2 for row indexing

the following loop prefixes duplicate labels

for every row of *Table_1* and every column of *Table_2*:

 Make a list of unique labels and a list of duplicate labels, except *BLANCs* and *EMPTYs*

 Copy the label of every spanning cell into each of its elementary cells

 Prepend the label of the preceding unique label, if any, to every duplicate label;

preceding is left in Table_1 and above in Table_2

now the labels of Table_1 and Table_2 may no longer be identical

Top = Row = Col = 1; # initial stub is the top left cell

Set ColHeader with rows *Top* to *Row* and columns *Col + 1* to *Ncols* of *Table_1*

Set RowHeader with columns *1* to *Col* and rows *Row + 1* to *Nrows* of *Table_2*

while ColHeader does not index columns of *Table_1*: *Row = Row + 1*;

while RowHeader does not index rows of *Table_2*: *Col = Col + 1*;

the following loop ensures minimality of indexing

while both columns of *Table_1* and rows of *Table_2* are indexed: *Row = Row - 1*;

Row = Row + 1; # restore to the last value for both row and column indexing

CC2 = (Row, Col) # == MIP (the Minimal Indexing Point)

while ColHeader indexes columns of *Table_1*: *Top = Top + 1*;

CC1 = (Row, Top - 1) # eliminate redundant rows above column header

Fig. 4.2. The MIPS algorithm.4.2 Prefixing

In the table of Fig. 1.1 all the headers are properly aligned, so all that is required is distributing the labels into the atomic cells resulting from fragmented spanning cells. But Fig. 4.3 shows an example where it is necessary to modify the labels of some header cells. This table is not a WFT because it violates the header-cell-uniqueness constraint of a WFT. One way to process such tables is to first convert them into WFTs. We handle a common case, where labels in a previous cell, like *Short messages, thousands 1)* and *Multimedia messages, thousands* in Fig. 4.3, disambiguate the duplicate label, *Change, %*. The solution is to prefix duplicate labels with unique predecessor labels. Here the two *Change, %* labels become *Short messages, thousands 1)&&Change, %* and *Multimedia messages, thousands&&Change, %*. (We use “&&” to mark the junction of the concatenated strings.) At this stage, we don’t yet know the extent of

the headers, so prefixing is done on a temporary copy of the table. Over 15% of the tables in our collection require prefixing (almost all in row headers) to turn them into WFTs.

Table 9. Numbers of outgoing short messages and multimedia messages from mobile phones in 2002-2008					
Year	Short messages, thousands 1)	Change, %	Short messages/ subscription	Multimedia messages, thousands	Change, %
2003	1 647 218	24,3	347	2 314	
2004	2 193 498	33,2	439	7 386	219,2

Fig. 4.3. A web table that requires prefixing.

After this prefixing step and the analogous step on the transposed rows, the MIPS algorithm proceeds as explained. MIPS finds *CC1* and *CC2* first. Then the program checks the original table under the column header candidate to find *CC3* as the leftmost cell of the first filled row of data region. *CC4*, was already located earlier as the rightmost cell of the last filled row. The cells in the corresponding regions are then labeled *StubHeader*, *RowHeader*, *ColHeader*, or *Data*.

4.3 Auxiliary Regions

To construct the Classification Table (Section 4.4), it is also necessary to analyze the auxiliary regions. Table titles are almost invariably in the top row of a table, therefore all the cells of the top row are labeled *TableTitle*. Footnote markers, if present, are found by searching below the data region for a list of common footnote-mark symbols (*, #, . °, †, etc.) and for single digits and letters (possibly followed by a period or a parenthesis). They are labeled *FNprefix*. All the cells following a footnote marker in the same row are marked *FNtext*. A cell containing both a *FNprefix* and a *FNtext* is marked *FNprefix&FNtext*. The program searches the entire table above the footnotes for the already detected and isolated footnote markers. If the footnote reference is found, the cell is labeled *FNref* (if the footnote reference is in a cell by itself) or *X&FNref*, where *X* can be any of the table regions above the footnote region, e.g., *RowHeader&FNref* for the footnote reference in cell A3 in Fig. 4.1. Our program missed this footnote reference because it is embedded in the middle of a header label, *OECD/DAC1 countries total*, and of course its superscript formatting disappeared in CSV.

Finally, every cell in a row that contains only non-empty cells that have not been otherwise classified is labeled *Note*.

4.4 Classification Table

The output of this stage is a *Classification Table*, e.g., Fig. 4.4 for the table in Fig. 1.1. This table is in a five-column format, with a row entry (after the header row) for each cell of its source table. The first column is a unique cell identifier with the file name of the CSV table and the cell coordinates. The second and third row give the numerical cell coordinates separately for ease of handling. The fourth column is the content of the cell in the original table, and the last column is its assigned class. Section 7 contains some examples of the application of this table.

Cell_ID	Row	Column	Content	Class
ODA_R1_C1	1	1	1 Official development assistance.	tabletitle
ODA_R1_C2	1	2		tabletitle
ODA_R1_C3	1	3		tabletitle
ODA_R1_C4	1	4		tabletitle
ODA_R1_C5	1	5		tabletitle
ODA_R1_C6	1	6		tabletitle
ODA_R1_C7	1	7		tabletitle
ODA_R1_C8	1	8		tabletitle
ODA_R1_C9	1	9		tabletitle
ODA_R1_C10	1	10		tabletitle
ODA_R1_C11	1	11		tabletitle
ODA_R2_C1	2	1		EMPTY
ODA_R2_C2	2	2		EMPTY
ODA_R2_C3	2	3		EMPTY
ODA_R2_C4	2	4		EMPTY
ODA_R2_C5	2	5		EMPTY
ODA_R2_C6	2	6		EMPTY
ODA_R2_C7	2	7		EMPTY
ODA_R2_C8	2	8		EMPTY
ODA_R2_C9	2	9		EMPTY
ODA_R2_C10	2	10		EMPTY
ODA_R2_C11	2	11		EMPTY
ODA_R3_C1	3	1	Country	stubheade
ODA_R3_C2	3	2	Million dollar	colheader
ODA_R3_C3	3	3		colheader
ODA_R3_C4	3	4		colheader
ODA_R3_C5	3	5		colheader
ODA_R3_C6	3	6		colheader
ODA_R3_C7	3	7	Percentage of GNI	colheader

Fig. 4.4. First 30 rows of the 408-row ClassificationTable for the table of Fig. 4.1.

5. COMPLEX HEADER STRUCTURES

Among our 200 tables, over 30% have complex header structures—multiple-row column headers, multiple-column row headers, and single row (column) headers that require prefixing. We analyze these headers to discover their category structure, and we use the discovered structure to create canonical relational tables that index individual data cells, making them searchable with standard database query languages.

5.1 Category Analysis

We define a simple algebra over the set of header labels. Each label appearing in a header is said to cover a subset of the cells in a table’s data region. For example, in Fig. 1.1 the label *Million dollar* covers

the first five columns of data cells and the label 2007 covers the first and the sixth columns. We define two binary operations, \times (intersection) and $+$ (union) over the header labels with respect to their covering properties. For example, the expression *Million dollar* + *Percentage of GNI* covers all the columns of the data region, while *Million dollar* \times 2007 covers only the first column. In this formulation, each header path can be equated with the product of labels appearing in it, and the set of all header paths can be equated with a sum of products (SOP) expression, in which each product term corresponds to a unique header path.

To determine the number of categories and their hierarchical structures, a factorization of an SOP expression E is carried out under the following constraints:

1. Only the distributive law and the associative laws of \times and $+$ are used in factorization. The commutative law is disallowed, so that ordering is maintained both among header paths for $+$ and within header paths for \times . (The $+$ ordering is left-to-right among column-header paths and top-to-bottom among row header paths, and the \times ordering is top-to-bottom within a column-header path and left-to-right within a row-header path).
2. The \times operation has higher precedence than $+$.
3. The factorization preserves the unique indexing property of E .
4. The factorization is complete in the sense that no term in it can be further factored.

As an illustration, consider the SOP expression corresponding to the ten column header paths of the exemplary table in Fig. 1.1 (where *Million\$* is *Million dollar* and *%GNI* is *Percentage of GNI*):

$$\begin{aligned} & \text{Million\$} \times 2007 + \text{Million\$} \times 2008 + \text{Million\$} \times 2009 + \text{Million\$} \times 2010^* + \text{Million\$} \times 2011^* \\ & + \%GNI \times 2007 + \%GNI \times 2008 + \%GNI \times 2009 + \%GNI \times 2010^* + \%GNI \times 2011^* \end{aligned}$$

The resulting complete factorization for this expression is:

$$(\text{Million\$} + \%GNI) \times (2007 + 2008 + 2009 + 2010^* + 2011^*)$$

The two categories, (*Million\$*, *%GNI*) and (2007, 2008, 2009, 2010*, 2011*), are revealed by the two top-level sum terms joined by \times in this expression. The product expression also indicates that the column header paths define the cross product of the individual category label sets associated with each sum term. This relationship of a multi-category header to the cross-product of category sets holds, in general.

5.2 Factorization Algorithm

We first factored headers with a public domain program [⁵⁶], which uses absorption and commutativity in addition to the distributive and associative laws. It yields an incorrect interpretation for header analysis because it fails to preserve the reading order and the distinction between identical labels in product terms. This motivated us to develop our own factorization algorithm based on the four constraints stated earlier. The explanation below is for column headers; the explanation for row headers is analogous.

Algorithm $Fact(E)$: As each header path includes every (cell) label of a header-path column, the initial SOP expression E has as many labels in each product term as there are header rows. This property is maintained for each subsequent call to the recursive factorization algorithm with a new argument expression. During a call the algorithm looks for several disjoint cases to identify the factors.

The simplest case is of a singleton factor S as the argument, where S is either just a simple sum $(a_1+a_2+...+a_n)$ or a simple product $(a_1 \times a_2 \times ... \times a_n)$ of the labels for $n > 0$. The algorithm outputs the singleton factor and returns from the call.

Otherwise, the argument expression E is an SOP with two or more terms of the same length greater than one. The algorithm analyzes the form of E based on factoring out the leading labels from the leftmost terms. In the limiting case, all the terms are included. If the *residues* after factoring out the leading labels are identical, i.e. E is of the form $S \times F$; otherwise, E must of the form $S \times F + G$. In all cases, S is identified as a factor and recursive calls are made to expressions F and G that have either shorter or fewer product terms. In summary, the three cases and the corresponding actions can be described as follows:

Case 1. E is of the form S ; $Fact(S) = S$.

Case 2. E is of the form $S \times F$, where F is the sum of two or more product terms; $Fact(E) = Fact(S) \times Fact(F)$.

Case 3. E is of the form $S \times F + G$, where F and G are non-null; $Fact(E) = S \times Fact(F) + Fact(G)$.

Example: To illustrate the algorithm, we consider our exemplary CSV table in Fig. 4.1. Before the factorization algorithm is called the fill-with-preceding-label operation described in the Section 4.2, restores the spanning-cell labels *Million dollar* and *Percentage of GNI* in the blank cells to their right. Thus, the factorization algorithm starts with the initial SOP expression:

$$E = MD \times Y7 + MD \times Y8 + MD \times Y9 + MD \times Y10 + MD \times Y11 + PGNI \times Y7 + PGNI \times Y8 + PGNI \times Y9 + PGNI \times Y10 + PGNI \times Y11$$

where we have abbreviated the long cell labels for convenience. Below, we show the execution trace of the factorization algorithm:

$$\begin{array}{lcl} Fact(E) & = & Fact(MD+PGNI) \times Fact(Y7+Y8+Y9+Y10+Y11) \\ \text{Case 2} & \downarrow & \downarrow \\ & \text{Case 1} & \text{Case 1} \\ & \rightarrow = (MD+PGNI) & \rightarrow = (Y7+Y8+Y9+Y10+Y11) \end{array}$$

5.3 Canonical Tables

The table designer's choice of rows or columns for laying out the categories depends primarily on the number of items in the category and on the size and aspect ratio of the available space. In relational tables, however, rows are tuples (records in Access), while columns are attributes (fields in Access). The database schema immutably assigns the values of each category to either a record or a field. We introduce canonical tables to represent the data elements in "ordinary" tables within the constraints of relational tables. Our canonical table is an $M \times 1$ relational table where each row comprises the indexing

header paths and the corresponding indexed data value. Therefore the number of rows in the canonical table equals the number of data cells in the original table (plus one for the relational table's field names in a header row). The number of columns is one (for the cell ID) plus one (for the data value) plus the number of categories (the dimensionality of the table). Hierarchical headers within a category also induce multiple columns in the category table. For our exemplary table, the canonical table has 240 rows and 5 columns.

To form the $M \times 1$ canonical table, each cell label in the original header paths becomes a key field value in the composite key comprising all key fields, and the data becomes a non-key field value. Fig. 5.1 shows part of the canonical table for the exemplary table. The first column references the original (table, row, and column) location of each data cell. The row headers in Fig. 1.1 are values in the RowCat_1.1 column in Fig. 5.1 and the column headers are distributed as values in the ColCat_1.1 and ColCat_2.1 columns according to their factorization—values in ColCat_1.1 from the factor (*Million dollar + Percentage of GNI*) and values in ColCat_2.1 from the factor (*2007 + 2008 + 2009 + 2010* + 2011**).

Cell_ID	RowCat_1.1	ColCat_1.1	ColCat_2.1	DATA
ODA_R6_C2	Norway	Million dollar	2007	3735
ODA_R6_C3	Norway	Million dollar	2008	4006
ODA_R6_C4	Norway	Million dollar	2009	4081
ODA_R6_C5	Norway	Million dollar	2010*	4580
ODA_R6_C6	Norway	Million dollar	2011*	4936
ODA_R6_C7	Norway	Percentage of GNI	2007	0.95
ODA_R6_C8	Norway	Percentage of GNI	2008	0.89
ODA_R6_C9	Norway	Percentage of GNI	2009	1.06
ODA_R6_C10	Norway	Percentage of GNI	2010*	1.1
ODA_R6_C11	Norway	Percentage of GNI	2011*	1
ODA_R7_C2	Denmark	Million dollar	2007	2562
ODA_R7_C3	Denmark	Million dollar	2008	2803
ODA_R7_C4	Denmark	Million dollar	2009	2810
ODA_R7_C5	Denmark	Million dollar	2010*	2871
ODA_R7_C6	Denmark	Million dollar	2011*	2981
ODA_R7_C7	Denmark	Percentage of GNI	2007	0.81
ODA_R7_C8	Denmark	Percentage of GNI	2008	0.82
ODA_R7_C9	Denmark	Percentage of GNI	2009	0.88
ODA_R7_C10	Denmark	Percentage of GNI	2010*	0.91
ODA_R7_C11	Denmark	Percentage of GNI	2011*	0.86
ODA_R8_C2	Finland	Million dollar	2007	981
ODA_R8_C3	Finland	Million dollar	2008	1166
ODA_R8_C4	Finland	Million dollar	2009	1290
ODA_R8_C5	Finland	Million dollar	2010*	1333
ODA_R8_C6	Finland	Million dollar	2011*	1409
ODA_R8_C7	Finland	Percentage of GNI	2007	0.39
ODA_R8_C8	Finland	Percentage of GNI	2008	0.44
ODA_R8_C9	Finland	Percentage of GNI	2009	0.54
ODA_R8_C10	Finland	Percentage of GNI	2010*	0.55
ODA_R8_C11	Finland	Percentage of GNI	2011*	0.52

Fig. 5.1. Canonical table for the table in Fig. 1.1 (first 30 of 240 rows).

The combined row and column headers that uniquely index each data value in the DATA column also index the data values in the original table. Because “ordinary” tables can always be recast as canonical tables, the formulation of the canonical table format and the automated reformulation of WFTs as canonical tables make a significant contribution to importing tabular web content into structured and searchable relational data structures. Moreover, as we show in Section 7, canonical tables also provide a direct path to the formulation of RDF triples and thus to searchable semantic-web content.

6. EXPERIMENTAL RESULTS

For the experiments, 200 tables were randomly drawn from a set of tables collected earlier from large statistical websites in the US and abroad ^[57]. The geopolitical and research sources included Statistics Canada, Science Direct, The World Bank, Statistics Norway, Statistics Finland, US Department of Justice, Geohive, US Energy Information Administration, and US Census Bureau. Table 6.1 shows the basic profile of the principal regions of these tables.

Table 6.1: Profile of our random sample of 200 tables.

	#Rows	#Cols	#Cells
Total	4,558	1,357	30,795
Average	23	7	154
Maximum	77	20	693
Minimum	5	2	20

The ground truth for the 200 tables consists of the four Critical Cells that demarcate the minimum indexing headers and the data region. The CCs can be easily verified against a ground truth obtained with VeriClick[48]. Different ground truth could be formulated to include some redundant rows above this minimal column header. For example, a row spanning the width of the column header could be considered either the table title or the label of the root-category. One could perhaps justify including in the column header some redundant rows (for example, units) above the data region. Readers could also differ on whether a first column containing only ordinal row numbers constitutes an acceptable row header. However, we chose the minimal column header for the ground truth as it does not depend on subjective interpretation of the table.

The performance of our Python segmentation program on the 200 tables can be summarized as follows: *100% correct segmentation*, including identification of the two non-indexable tables. The two non-indexable tables had duplicate columns, which we consider a source error.

All of the footnotes were found in the 33% of the tables that had them. The program detected 218 reference marks to the footnotes within the body of the tables (some had more than a dozen). It missed them in three tables where the footnote reference marks were not near the end of the cell text.

Tables 6.2 shows the distributions of the row and column header sizes. The data shows that multi-row column headers occur with much higher frequency (32) than multi-column row header (4). Prefixing was

applied to 4 column headers and 29 row headers. There are 15 two category row headers and six two-category column headers. We factor all of these correctly.

Table 6.2: Joint distribution of row and column header sizes in the indexable tables

RH	CH					
	1	2	3	4	5	Totals
1	162	28	4	0	0	194
2	0	0	0	0	0	0
3	4	0	0	0	0	4
4	0	0	0	0	0	0
5	0	0	0	0	0	0

The entire processing, including writing the 198 files for input to Access and the 198 Classification files, required 14 seconds on a 3.4 GHz Dell Optiplex 7010 running Python 2.7 under Windows 7.0.

7. APPLICATION QUERIES

Having shown how to transform a human-readable table to a machine-readable table, we now demonstrate that the transformations yield directly useable information for formal queries in widely available application software. Such a “proof of the pudding” is seldom offered in prior work where the table processing results are usually retained only in an ad hoc format.

We process queries using industry standards—Microsoft Access for SQL queries over a generated relational database and the OpenLink Virtuoso semantic-web endpoint and Protégé for SPARQL queries over a generated triple store represented in the semantic-web languages RDF^[58] and OWL^[59]. In all cases the generated, canonical M×1 tables and the generated classification tables are automatically imported into an appropriate data store where their content can be queried directly. Before importing them, an automated editing pass over cell content replaces decimal commas with periods and deletes thousands-separator blanks and commas. To accommodate syntax requirements, the dots in *RowCat* and *ColCat* identifiers are also removed.

The three Access queries—one on a single table, one on a pair of tables, and a meta-data query over all the tables—are followed by two SPARQL queries—one using Virtuoso that duplicates the second Access query and another using Protégé that runs globally over all generated M×1 tables.

7.1 Relational Database Queries

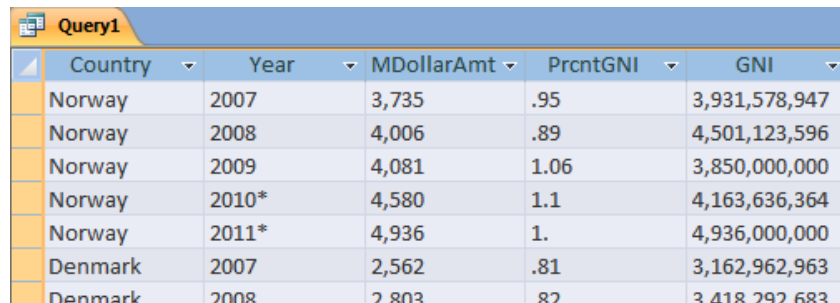
The first query addresses the table in Fig. 1.1, from which the MIPS and Fact algorithms generated the M×1 canonical table is in Fig. 5.13:

Query 1: Compute the GNI for every country for every year.

```

SELECT MDollarTbl.RowCat_11 AS Country, MDollarTbl.ColCat_21 AS Year,
       FORMAT(MDollarTbl.DATA, '#,###') AS MDollarAmt,
       FORMAT(PrcntTbl.DATA, '#.##') AS PrcntGNI,
       FORMAT(1000000*MDollarTbl.DATA/PrcntTbl.DATA, '###,###,###,###') AS GNI
FROM ODA_Mx1 AS MDollarTbl, ODA_Mx1 AS PrcntTbl
WHERE MDollarTbl.RowCat_11 = PrcntTbl.RowCat_11
      AND MDollarTbl.RowCat_11 <> 'EMPTY' AND MDollarTbl.RowCat_11 NOT LIKE 'OECD*'
      AND MDollarTbl.ColCat_11 = 'Million dollar' AND PrcntTbl.ColCat_11 LIKE 'Percentage*'
      AND MDollarTbl.ColCat_21 = PrcntTbl.ColCat_21;

```



Country	Year	MDollarAmt	PrcntGNI	GNI
Norway	2007	3,735	.95	3,931,578,947
Norway	2008	4,006	.89	4,501,123,596
Norway	2009	4,081	1.06	3,850,000,000
Norway	2010*	4,580	1.1	4,163,636,364
Norway	2011*	4,936	1.	4,936,000,000
Denmark	2007	2,562	.81	3,162,962,963
Denmark	2008	2,803	.82	3,418,292,683

Fig. 7.1. MS-Access screenshot of Query 1 results (partial).

Fig. 7.1 gives the first seven rows of the result. The query creates two intermediate tables, one for the million-dollar amounts and one for the percentage of GNI, and then aligns the values by country and year and computes the GNI. Because the column categories have the factorization

$$(\text{Million dollar} + \text{Percentage of GNI}) \times (2007 + 2008 + 2009 + 2010^* + 2011^*)$$

the SQL statements can tease apart the canonical table, collect the million-dollar data cells into the table called *MDollarTbl* and the percentage-GNI data cells in to the table called *PrcntTbl*, align the cells by year and country, and compute the GNI.

Whereas the first query illustrates the use of categories and factorization in query formulation, the second query illustrates combining disparate, but semantically overlapping tables. The table in Fig. 7.2 quantifies international trade by land through Detroit, Michigan, and the table in Fig. 7.3 quantifies and compares U.S. trade with its NAFTA partners, Canada and Mexico, against U.S. international trade for all countries.

Query 2. Find the percent of U.S. land trade through Detroit vs. the U.S. surface trade with NAFTA partners for all years the two tables have in common, 1999–2003.

```

SELECT FORMAT(T028Mx1.ColCat_11, '####') AS Year
       FORMAT(T028Mx1.DATA, '#,###') AS MillionDollarAmt,
       FORMAT(T079Mx1.DATA, '###') AS BillionDollar Amt,
       FORMAT(100*T028Mx1.DATA/(T079Mx1.DATA*1000), '##.##') AS PercentDetroitLandTrade
FROM T028Mx1, T079Mx1
WHERE T028Mx1.RowCat_11 = 'Total'
      AND T079Mx1.ColCat_11 Like "U.S. surface trade*current U.S. dollars*"
      AND T028Mx1.ColCat_11 = T079Mx1.RowCat_11
      AND 1999 <= T079Mx1.RowCat_11 AND T079Mx1RowCat_11 <= 2003;

```

Canonical Mx1 tables characterize cells. The highlighted cells in Figs. 7.2 and 7.3 mark the combination of cells that satisfy the constraints for the year 1999, yielding the 18.5% in Fig. 7.4, which is computed from the 92,583 million in the *Total* row of the table in Fig. 7.2 and the 501 billion, the *U.S. surface trad...* column of the table in Fig. 7.3.

	A	B	C	D	E	F	G	H	I	J	K	L
1	TABLE 4. Value of International Land Trade via Detroit, MI, by Mode: 1999-2003											
2												
3	(\$ millions)											
4												
5	Excel CSV											
6												
7		1999	2000	2001	2002	2003						
8	Truck	83,889	85,468	79,762	85,062	84,811						
9	Rail	8,343	8,598	11,909	15,607	16,723						
10	Pipeline	45	78	67	50	92						
11	Other and	306	297	244	172	263						
12	Total	92,583	94,441	91,982	100,891	101,890						
13												
14	SOURCE: U.S. Department of Transportation, Bureau of Transportation Statistics, Transborder Surface Freight Data, 1999-2003.											

Fig. 7.2. International land trade with the U.S. through Detroit, Michigan.

D5 U.S. surface trade with NAFTA partners (Billions of current U.S. dollars)									
	A	B	C	D	E	F	G	H	I
1	Table 1: Value of U.S. Goods Trade with Canada and Mexico Compared with U.S. Merchandise Trade with All Countries: 1990-2004								
2									
3	Excel CSV								
4									
5	Year	Total U.S. inter	U.S. trade with N	U.S. surface trade	Ratio of U.S. trade	Ratio of U.S. surfac	Total U.S. interna	U.S. trade with N	U.S. surface trad
6	1990	889	233	204	26.2	87.5	822	215	188
7	1991	910	241	210	26.4	87.3	848	224	196
8	1992	981	264	232	27	87.7	924	249	218
9	1993	1,046	293	258	28	88	993	278	245
10	1994	1,176	343	312	29.2	90.9	1,107	323	293
11	1995	1,328	380	338	28.6	89.1	1,219	349	311
12	1996	1,420	421	377	29.7	89.5	1,338	397	355
13	1997	1,560	475	426	30.5	89.6	1,522	464	416
14	1998	1,594	503	452	31.5	89.9	1,630	514	462
15	1999	1,720	559	501	32.5	89.7	1,771	575	516
16	2000	2,000	653	576	32.7	88.1	2,000	653	576
17	2001	1,870	614	547	32.8	89.2	1,905	625	558
18	2002	1,857	604	541	32.5	89.6	1,915	622	558
19	2003	1,983	629	563	31.7	89.4	1,996	633	566
20	2004	2,288	712	634	31.1	89	2,207	687	611
21	Perce	157.4	205.7	211.2			168.3	218.7	224.4
22	1990-2004								
23	Annua	7	8.3	8.4			7.3	8.6	8.8
24									
25	NOTE: NAFTA = North American Free Trade Agreement								
26									
27	1 To compare economic changes over time, current or nominal values of currencies must be deflated or adjusted for inflation. In the United States								
28									
29	SOURCE: U.S. Department of Transportation, Research and Innovative Technology Administration, Bureau of Transportation Statistics, based on: cu								

Fig. 7.3. International trade with the U.S. (with surface trade header fully shown in edit box).

Query2				
	Year	MillionDollarAmt	BillionDollarAmt	PercentDetroitLandTrade
	1999	92,583	501	18.5
	2000	94,441	576	16.4
	2001	91,982	547	16.8
	2002	100,891	541	18.6
	2003	101,890	563	18.1

Fig. 7.4. Screenshot of the results of processing Query 2.

A last SQL query shows that one can obtain useful information from classification tables. The classification tables contain the meta-information needed for further downstream processing in automating table interpretation. Identifying aggregate operations such as sums and averages in tables is an example of the kind of additional processing that may be of interest. Query 3 checks for one of the most common aggregate-operation configurations: a row of data values labeled *Total* whose corresponding column data values sum to the total values.

Query 3. Do columns of data values sum to corresponding values with row header “Total”?

```

SELECT FORMAT(T1, Column, '#') AS Column, FORMAT(SUM(T1.Content), '###,###') AS
  ComputedTotal, "Total Equals Column Sum" AS Confirmation
FROM Classification_T028 T1
WHERE T1.Class = "data"
  AND T1.Row <> (SELECT T1.Row FROM Classification_T028 T1 WHERE T1.Content = "Total")
GROUP BY T1.Column
HAVING SUM(T1.Content) = (
  SELECT T2.Content
  FROM Classification_T028 T2
  WHERE T2.Class = "data"
    AND T2.Row = (SELECT T2.Row FROM Classification_T028 T2 WHERE T2.Content = "Total")
    AND T1.Column = T2.Column);

```

As noted in connection with the classification table in Fig. 4.4, all classification tables have the same field labels: {*Cell_ID*, *Row*, *Column*, *Content*, *Class*}. Since all classification tables have the same schema, we need write only one query (parameterized with the table to check) to discover a particular aggregate property for all tables. Fig. 7.5 shows the results of applying the SQL query to the classification table of table in Fig. 7.2 confirming that the *Total* values are indeed the aggregate sums of their respective columns. (Note that Column 6 does not appear in the result table in Fig. 7.5. The data values in Column 6 of the table in Fig. 7.2 sum to 101,889, not 101,890, as displayed in the table.)

Query3			
	Column	ComputedTotal	Confirmation
	2	92,583	Total Equals Column Sum
	3	94,441	Total Equals Column Sum
	4	91,982	Total Equals Column Sum
	5	100,891	Total Equals Column Sum

Fig. 7.5. Results of Total-check query.

7.2 Semantic Web Queries

To produce semantic-web data for queries, we create RDF triples—(*subject*, *predicate*, *object*) statements. Fig. 7.7 shows in triple-XML syntax the first six triples our Python program generates from the canonical M×1 table for the table in Fig. 7.2. As Fig. 5.1 illustrates, each row in a canonical table (beyond the header row) describes one data cell in a table, and the Python program generates a standard, fixed group of triples that describe each data cell—its ID, headers, and data value. The cell ID in the first column in an M×1 table identifies the cell being defined. Thus, the *subject* for each cell is its cell ID. In the XML syntax for triples in Fig. 7.7, the *about* attribute in an *rdf:Description* tag identifies the subject—a triple is “about” whatever its subject is. The *predicate* for a triple is in the tag of each nested XML statement, and the *object* is the value the tag encloses. The first triple in Fig. 7.7 is, therefore, (*C10028_R8_C2*, *RowCat_11*, *Truck*), the second is (*C10028_R8_C2*, *ColCat_11*, *1999*), and the third is (*C10028_R8_C2*, *DATA*, *83889*), which altogether means that the cell identified by *C10028_R8_C2* (the cell at Row 8 and Column 2 in Fig. 13, which displays table T028, whose internal object identifier is C10028) has as its row header *Truck*, as its column header *1999*, and its data value *83889*.

```

<rdf:RDF
  xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
  xmlns:T028Mx1='http://osm.cs.byu.edu/tables/T028.rdf#'
>
  <rdf:Description rdf:about='C10028_R8_C2'>
    <T028Mx1:RowCat_11>Truck</T028Mx1:RowCat_11>
    <T028Mx1:ColCat_11>1999</T028Mx1:ColCat_11>
    <T028Mx1:DATA>83889</T028Mx1:DATA>
  </rdf:Description>
  <rdf:Description rdf:about='C10028_R8_C3'>
    <T028Mx1:RowCat_11>Truck</T028Mx1:RowCat_11>
    <T028Mx1:ColCat_11>2000</T028Mx1:ColCat_11>
    <T028Mx1:DATA>85468</T028Mx1:DATA>
  </rdf:Description>
  ...
</rdf:RDF>

```

Fig. 7.6. Generated RDF triples.

Query Text

```

PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX xsd:<http://www.w3.org/2001/XMLSchema#>
PREFIX T028Mx1:<http://T028Mx1-rdf#>
PREFIX T079Mx1:<http://T079Mx1-rdf#>

SELECT xsd:int(?Year) AS ?Year,
       xsd:int(?Amount) AS ?MillionDollarAmt,
       xsd:int(?Value) AS ?BillionDollarAmt,
       100*xsd:float(?Amount)/(xsd:float(?Value)*1000) AS ?PercentDetroitLandTrade
FROM <http://osm.cs.byu.edu/tables/T028.rdf>
FROM <http://osm.cs.byu.edu/tables/T079.rdf>
WHERE {
  ?DetroitLandTradeCell T028Mx1:RowCat_11 ?TransportMode .
  ?DetroitLandTradeCell T028Mx1:ColCat_11 ?Year .
  ?DetroitLandTradeCell T028Mx1:DATA ?Amount .
  ?USGoodsTradeCell T079Mx1:RowCat_11 ?Year2 .
  ?USGoodsTradeCell T079Mx1:ColCat_11 ?TradeType .
  ?USGoodsTradeCell T079Mx1:DATA ?Value .
  FILTER (
    regex(?TradeType, "surface trade.*current.*dollars")
    AND regex(?TransportMode, "Total")
    AND ?Year = ?Year2
  )
}

```

Fig. 7.7. SPARQL query.

As an illustration of querying semantic-web data, Fig. 7.7 gives a SPARQL query for Query 2. The PREFIX statements define name spaces: *rdf* and *xsd* are W3C standards and the two *Mx1* prefixes are for the canonical tables of the two tables in Figs. 7.2 and 7.3, which we have stored on the web at <http://osm.cs.byu.edu/tables/T028.rdf> and <http://osm.cs.byu.edu/tables/T079.rdf>. SPARQL queries work by matching triples in the WHERE clause with triples in the RDF data in every combination that satisfies the constraints in the FILTER statement. Except for the formatting, the query results are the same as in Fig. 7.4.

The SPARQL query formulated above requires some knowledge of the queried table. In Fig. 7.7, for example, we see the line *?DetroitLandTradeCell T028Mx1:RowCat_11 ?TransportationMode*. Formulating this line (and some others) of the query requires understanding the structure of input tables.

To remove structure dependencies for global queries, we programmed the construction of a uniform set of triples based on the canonical *Mx1* tables. While in the triple construction described above the number of triples for each cell depends on the category structure of each table, the OWL-model triples do not. Instead, each cell is described by the same number of triples (based on the widest of the *Mx1* tables). Hence, all of our tables can be searched simultaneously with a single query, for example, to

determine in which tables *Exports* appears as a column category. Because of the uniformity of the model, the query (with prefix headers omitted) simplifies to:

```
Select distinct ?cell ?value where{
  ?cell table:hasColumn ?col filter regex(?col, Exports).
  ?cell table:hasValue ?value}
```

Run in Protégé on a Lenovo T61 laptop this query executed in less than a millisecond of CPU-time over a 104 megabyte triple store for the 200 tables.

8. CONCLUSION

The formalization of well-formed tables (WFTs) by means of block algebra captures and models the table layouts that cover the vast majority of tables encountered in print and on the web. It obviates previous attempts to recognize their infinite variety of framing, partial ruling, typeface, color scheme, or cell formatting details. The formalization serves as the basis for the algorithms described in Sections 4 and 5 that convert human-readable WFTs into a machine-processable form amenable to formal query processing. As shown in Section 7, the algorithms convert and import WFTs into both a relational database and an RDF/OWL triple store, enabling them to be queried with SQL or SPARQL. Moreover, our WFT formalization encompasses auxiliary information: table titles, footnotes, and miscellaneous notes, broadening previously reported work.

The WFT formalization not only engenders an algorithmic solution to discovering indexing headers and finding their multi-categorical indexing structure, it also provides a target for processing tables that do not strictly satisfy the WFT definition. As shown in Section 4, our MIPS algorithm converts prefix tables, in which header indexes are “crooked,” into bona fide WFTs.

The proposed algorithms are based on a formal definition of well-formed tables. Thus they need no statistically significant experimental validation, only a demonstration of implementability and applicability. Tables on the web, however, are not always well-formed. In our small but heterogeneous collection of 200 web tables, MIPS found all of the critical cells and correctly segmented the minimal table headers and the data regions. Fact(F) discovered all 21 multi-category headers. The heuristics for table titles, notes and footnotes probe the limits of purely syntactic table processing. The canonical and classification tables were imported and queried in Access, Virtuoso, and Protégé. The tables and the critical-cell ground truth, already in use by other researchers, will be posted at the IAPR TC-11 website.

This research also sets the stage for future work. In addition to enabling formal queries, the cell-classification table (Section 4) identifies each cell of every processed table as data, column header, row header, stub header, title, footnote marker, footnote, or miscellaneous note. Knowing the cell classification and the category-tree indexing structure are likely to aid in discovering aggregate operations (as suggested in Query 3 of Section 7), in resolving hierarchical headers with and without accompanying aggregate operations, in typing data values, and in sorting out and discovering implicit roots of category trees (e.g., *Year* and *Country* in Fig. 1.1). Resolving these issues will require matching

observable table characteristics with semantic resources, whereas our work here is based on syntactic analysis.

Further long-term research objectives include (1) ontologically formalizing WFTs as both input for table processing algorithms and as interpreted output tables whose syntax and semantics have been fully resolved, (2) turning egregious tables into input WFTs, (3) integrating interpreted tables into ontologies, and (4) automating free-form query processing over collections of interpreted and integrated table content. All of this will require continuing efforts to combine the perspectives of the document-processing, information-retrieval, database, and web-science communities.

ACKNOWLEDGEMENTS

Mukkai Krishnamoorthy acknowledges the help of Dr. Ravi Palla with Protégé.

REFERENCES

- 1 Wang, X.: Tabular abstraction, editing, and formatting. Ph.D. thesis, University of Waterloo (1996)
- 2 Laurentini, A., Viada, P.: Identifying and understanding tabular material in compound documents. In: Proceedings of the Eleventh International Conference on Pattern Recognition (ICPR'92), pp. 405–409, The Hague (1992)
- 3 Turolla, E., Belaid, Y., Belaid, A.: Form item extraction based on line searching. In: Kasturi, R., Tombre, K. (eds.) Graphics Recognition—Methods and Applications. Lecture Notes in Computer Science, vol. 1072, pp. 69–79. Springer-Verlag, Berlin, Germany (1996)
- 4 Chandran, S., Kasturi, R.: Structural recognition of tabulated data. In: Proceedings of the Second International Conference on Document Analysis and Recognition (ICDAR'93), pp. 516–519. Tsukuba Science City, Japan (1993)
- 5 Ittonori, K.: A table structure recognition based on textblock arrangement and ruled line position. In: Proceedings of the Second International Conference on Document Analysis and Recognition (ICDAR'93), pp. 765–768. Tsukuba Science City, Japan (1993)
- 6 Pinto, D., McCallum, A., Wei, X., Croft, W.B.: Table extraction using conditional random fields. In: Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 235–242 (2003)
- 7 Hirayama, Y.: A method for table structure analysis using DP matching. In: Proceedings of the Third International Conference on Document Analysis and Recognition (ICDAR'95), pp. 583–586. Montr' eal, Canada (1995)
- 8 Handley, J.C.: Document recognition. In: Dougherty, E.R. (ed.) Electronic Imaging Technology, chapter 8. SPIE—The International Society for Optical Engineering (1999)

- 9 Zuyev, K.: Table image segmentation. In: Proceedings of the International Conference on Document Analysis and Recognition (ICDAR'97), pp. 705–708. (1997)
- 10 Abu-Tarif, A.: Table processing and table understanding. Master's thesis, Rensselaer Polytechnic Institute, May (1998)
- 11 Pyreddy, P., Croft, W.B.: TINTIN: A system for retrieval in text tables. Technical Report UM-CS-1997-002. University of Massachusetts, Amherst (1997)
- 12 Kieninger, T.G.: Table structure recognition based on robust block segmentation. In: Proceedings of Document Recognition V (IS&T/SPIE Electronic Imaging'98), vol. 3305, pp. 22–32. San Jose, CA (1998)
- 13 Hu, J., Kashi, R., Lopresti, D., Wilfong, G.: Table structure recognition and its evaluation. In: Kantor, P.B., Lopresti, D.P., Zhou, J.(eds.) Proceedings of Document Recognition and Retrieval VIII (IS&T/SPIE Electronic Imaging), vol. 4307, pp. 44–55. San Jose, CA (2001)
- 14 Wolfgang Gatterbauer, Paul Bohunsky, Bernhard Krüpl, and Bernhard Pollak, Marcus Herzog, Towards Domain Independent Information Extraction from Web Tables. In: WWW 2007, May 8–12, 2007, Banff, Alberta, Canada.
- 15 Amano, A., Asada, N.: Graph grammar based analysis system of complex table form document. In: Proceedings of the Seventh International Conference on Document Analysis and Recognition (2003)
- 16 Bing, L., Zao, J., Hong, X.: New method for logical structure extraction of form document image. In: Proceedings of Document Recognition and Retrieval VI (IS&T/SPIE Electronic Imaging' 99), vol. 3651, pp. 183–193. San Jose, CA (1999)
- 17 Kieninger, T., Dengel, A.: A paper-to-HTML table converting system. In: Proceedings of Document Analysis Systems (DAS) 98. Nagano, Japan (1998)
- 18 Coüasnon, B., Camillerapp, J., Leplumey, I.: Making handwritten archives documents accessible to public with a generic system of document image analysis. In: Proceedings of the International Workshop on Document Image Analysis for Libraries, pp. 270–277. Palo Alto, CA (2004)
- 19 Isaac Martinat, Bertrand Coüasnon, Jean Camillerapp. An Adaptative Recognition System Using a Table Description Language for Hierarchical Table Structures in Archival Documents, In Graphics Recognition: Recent Advances and Perspectives, Vol. 5046, pp. 9-20, Lecture Note in Computer Science, Springer-Verlag, 2008
- 20 Aurélie Lemaitre, Jean Camillerapp, Bertrand Coüasnon. Multiresolution Cooperation Improves Document Structure Recognition, International Journal on Document Analysis and Recognition (IJ DAR), 11(2):97-109, Novembre 2008. [doi>](#)
- 21 Watanabe, T., Quo, Q.L., Sugie, N.: Layout recognition of multikinds of table-form documents. IEEE Trans. Pattern Anal. Mach. Intell. 17(4), 432–445 (1995)
- 22 Shamalian, J.H., Baird, H.S., Wood, T.L.: A retargetable table reader. In: Proceedings of the International Conference on Document Analysis and Recognition (ICDAR'97), pp. 158–163. (1997)
- 23 A. Halevy, P. Norvig, and F. Pereira, The Unreasonable Effectiveness of Data IEEE INTELLIGENT SYSTEMS. March/April 2009.

- 24 W.J. Cafarella, A. Halevy, D.Z. Wang, E. Wu, Y. Zhang, WebTables: Exploring the Power of Tables on the Web, VLDB '08 Auckland, New Zealand.
- 25 P. Venetis, A. Halevy, J. Madhavan, M. Pasca, W. Shen, F. Wu, G. Miao, C. Wu, Recovering Semantics of Tables on the Web, Proceedings of the LDB Endowment, Vol. 4, No. 9, 2011
- 26 Hector Gonzalez, Alon Y. Halevy, Christian S. Jensen, Anno Langen, Jayant Madhavan, Rebecca Shapley, Warren Shen, Jonathan Goldberg-Kidony, Google Fusion Tables: Web-Centered Data Management and Collaboration, SIGMOD'10, June 6–11, 2010, Indianapolis, Indiana, USA. 2010
- 27 M.D. Adelfio and H. Samet, Schema Extraction for Tabular Data on the Web, Proceedings of The 39th International Conference on Very Large Data Bases, (Proceedings of the VLDB Endowment, Volume 6, Number 6), Riva del Garda, Trento, Italy 26–30 August, 2013.
- 28 Vanessa Long, An Agent-Based Approach to Table Recognition and Interpretation, Macquarie University PhD dissertation, May 2010.
- 29 Nikita Astrakhantsev, Extracting Objects and Their Attributes from Tables in Text Documents IN Denis Turdakov, Andrey Simanovsky (Eds.): Proceedings of the Seventh Spring Researchers Colloquium on Databases and Information Systems, SYRCoDIS 2011, Moscow, Russia, June 2-3, 2011. CEUR Workshop Proceedings 735 CEUR-WS.org 2011 pp 24-37 OR pp 297-309? (from Moorthy)
- 30 Hurst, M., Douglas, S.: Layout and language: Preliminary investigations in recognizing the structure of tables. In: Proceedings of the International Conference on Document Analysis and Recognition (ICDAR'97), pp. 1043–1047 (1997)
- 31 Matthew Hurst, Towards a theory of tables, Int. J. Doc. Anal. Recognit. 8 (2-3), Springer, Heidelberg, 66-86, 2006
- 32 Hurst, M. 2000. The Interpretation of Tables in Texts. Ph.D. thesis, University of Edinburgh.
- 33 Ana Costa e Silva, A. M. Jorge and L. Torgo, Design of an end-to-end method to extract information from tables, Int. J. Doc. Anal. Recognit. 8 (2-3), Springer, Heidelberg, 66-86, 2006
- 34 Yeon-Seok Kim, Kyong-Ho Lee, Extracting logical structures from HTML tables, Computer Standards & Interfaces, Volume 30, Issue 5, July 2008, Pages 296-308
- 35 A. Pivk et al., Transforming arbitrary tables into logical form with TARTAR, Data & Knowledge Engineering 60 (2007) 567–595 // Aleksander Pivk, Philipp Cimiano, York Sure, Matjaz Gams, Vladislav Rajkovic, Rudi Studer
- 36 Z. Chen and M. Cafarella, Automatic Web Spreadsheet Data Extraction, Proceedings of the 3rd International Workshop on Semantic Search over the Web (SSW 2013), Riva del Garda, Trento, Italy, 30 August 2013.
- 37 D. Lopresti and G. Nagy, Automated table processing: an (opinionated) survey, Proceedings of IAPR Workshop on Graphics Recognition (GREC99), pp. 109-134, Jaipur, India, September 1999.
- 38 J. Hu, R. Kashi, D. Lopresti, G. Wilfong, and G. Nagy, Why table ground-truthing is hard, Proceedings of International Conference on Document Analysis and Recognition, pp. 129-133, Seattle, WA, IEEE Computer Society Press, September 2001

- 39 D.W. Embley, D. Lopresti, and G. Nagy, Notes on Contemporary Table Recognition, Document Analysis Systems VII, 7th International Workshop, Procs. DAS 2006, H. Bunke and A. L. Spitz, Eds., vol. 3872, LNCS, pp. 164-175, Springer, Nelson, New Zealand, February 13-15, 2006.
- 40 D.W. Embley, D. Lopresti, M. Hurst, and G. Nagy, "Table Processing Paradigms: A Research Survey," International Journal of Document Analysis and Recognition, vol 8, no. 2-3, pp. 66-86, Springer, June 2006.
- 41 Nilesh Dalvi, Ravi Kumar, Bo Pang, Raghu Ramakrishnan, Andrew Tomkins, Philip Bohannon, Sathiya Keerthi, Srujana Merugu, A Web of Concepts, PODS'09, June 29–July 2, 2009, Providence, Rhode Island, USA.
- 42 David W. Embley, Stephen W. Liddle, Deryle Lonsdale, George Nagy, Yuri Tijerino, Robert Clawson, Jordan Crabtree, Yihong Ding, Piyushee Jha, Zonghui Lian, Stephen Lynn, Raghav K. Padmanabhan, Jeff Peters, Cui Tao, Robby Watts, Charla Woodbury, and Andrew Zitzelberger, A Conceptual-Model-Based Computational Alembic for a Web of Knowledge, Procs. 27th International Conference on Conceptual Modeling (ER 2008), Oct 20-23, Barcelona.
- 43 Embley, D., Tao, C., Liddle, S. 2005. Automating the extraction of data from HTML tables with unknown structure. Data Knowl. Eng., 54(1), July 2005, 3–28.
- 44 C. Tao and D.W. Embley, Automatic Hidden-Web Table Interpretation, Conceptualization, and Semantic Annotation, Data & Knowledge Engineering, 68(7), July 2009, 683–703.
- 45 Ramana C. Jandhyala, Mukkai Krishnamoorthy, George Nagy, Raghav Padmanabhan, Shared Seth, and William Silversmith, From Tessellations to Table Interpretation, Proceedings of the 8th International Conference on Mathematical Knowledge Management, MKM 2009, Grand Bend, Ontario, in J. Carette et al. (Eds.): Calculemus/MKM 2009, LNAI 5625, pp. 422–437, 2009. Springer-Verlag Berlin Heidelberg 2009.
- 46 G. Nagy, Learning the Characteristics of Critical Cells from Web Tables, Procs. ICPR, Tsukuba, Japan, Nov. 2012.
- 47 D. W. Embley, M. Krishnamoorthy, G. Nagy, and S. Seth, Factoring Web Tables. K.G. Mehrotra et al. (Eds.): IEA/AIE 2011, Part I, LNAI 6703, pp. 253–263, 2011. © Springer-Verlag Berlin Heidelberg 2011]
- 48 G. Nagy, M. Tamhankar, VeriClick, an efficient tool for table format verification, Procs. SPIE/EIT/DRR, San Francisco, Jan. 2012
- 49 S. Seth, and G. Nagy, Segmenting Tables via Indexing of Value Cells by Table Headers, Proc. ICDAR 2013, Washington, D.C., August 2013.
- 50 G. Nagy, D. W. Embley, Seth Seth, End-to-End Conversion of HTML Tables for Populating a Relational Database, Proc. DAS 2014, Tours, France, 2014.
- 51 D.W. Embley, S. Seth, G. Nagy, Transforming Web tables to a relational database, Procs. ICPR 2014, Stockholm, Sweden, 2014.
- 52 D.W. Embley, S. Seth, M. Krishnamoorthy, G. Nagy, Clustering header categories extracted from web tables, Procs. SPIE/IST Document Recognition and Retrieval, San Francisco, CA, Feb. 2015.

- 53 U.S. Government Printing Office, Style Manual An official guide to the form and style of Federal Government printing, Section 13, 281-299, 2008. Also accessible at:
<http://www.gpoaccess.gov/stylemanual/index.html>.
- 54 P. Balbiani, J-F. Condotta, L. Farinas Del Cero, Tractability Results in the Block Algebra, J. Logic Computat. 12, 5, 885-909, 2002.
- 55 James F. Allen, Maintaining knowledge about temporal intervals, Communications of the ACM, v.26 n.11, p.832-843, Nov. 1983 [doi>10.1145/182.358434]
- 56 [1] 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, Memorandum No. UCB/ERL M92/41, Electronics Research Laboratory, College of Engineering, University of California, Berkeley, CA 94720, May 1992, downloaded 11/4/10 from: <http://www.eecs.berkeley.edu/Pubs/TechRpts/1992/ERL-92-41.pdf>.
- 57 Padmanabhan, R., R. C. Jandhyala, M. Krishnamoorthy, G. Nagy, S Seth, and W. Silversmith. "Interactive Conversion of Large Web Tables." GREC. 2009. 25-36.
- 58 www.w3.org/RDF/
- 59 www.w3.org/OWL