Factoring Web Tables

David W. Embley[1], Mukkai Krishnamoorthy[2],
George Nagy[2], and Sharad Seth[3]

[1] Brigham Young University, Provo, UT, USA
embley@cs.byu.edu
[2] Rensselaer Polytechnic Institute, Troy, NY, USA
mskmoorthy@gmail.com, nagy@ecse.rpi.edu
[3] University of Nebraska, Lincoln, Lincoln, NE, USA
seth@cse.unl.edu

Abstract. Automatic interpretation of web tables can enable database-like semantic search over the plethora of information stored in tables on the web. Our table interpretation method presented here converts the two-dimensional hierarchy of table headers, which provides a visual means of assimilating complex data, into a set of strings that is more amenable to algorithmic analysis of table structure. We show that Header Paths, a new purely syntactic representation of visual tables, can be readily transformed ("factored") into several existing representations of structured data, including category trees and relational tables. Detailed examination of over 100 tables reveals what table features require further work.

Keywords: table analysis, table headers, header paths, table syntax, category trees, relational tables, algebraic factorization.

1 Introduction

The objective of this work is to make sense of (analyze, interpret, transform) tables the best we can without resorting to any external semantics: we merely manipulate symbols. Remaining firmly on the near side of the semantic gap, we propose a canonical table representation based on *Header Paths* that relate column/row headers and data cells. We show that this "canonical" representation is adequate for subsequent transformations into three other representations that are more suitable for specific data/information retrieval tasks. The three targets for transformations are

Visual Table (VT). The VT provides the conventional two-dimensional table that humans are used to. Header and contents rows and columns may, however, be jointly permuted in the VT generated from Header Paths, and it does not preserve style attributes like typeface and italics.

Category Tree (CT). The CT is an "abstract data structure" proposed by X. Wang in 1996[1]. It is a convenient format for storing tables in an internal XML format and thus can also be used for information exchange.

K.G. Mehrotra et al. (Eds.): IEA/AIE 2011, Part I, LNAI 6703, pp. 253–263, 2011.

Relational Table (RT). RT provides the link to standard relational databases and their extensive apparatus for storing and retrieving data from a collection of tables.

Header Paths are a general approach to tables and capture the essential features of two-dimensional indexing. A *Well Formed Table* (*WFT*) is a table that provides a unique string of Header Paths to each data cell. If the Header Paths of two distinct data cells are identical, then the table is ambiguous. Although the concept of Header Paths seems natural, we have not found any previous work that defines them or uses them explicitly for table analysis. Our contribution includes extracting Header Paths from web tables, analysis of Header Paths using open-source mathematical software, and a new way of transforming them into a standard relational representation.

We have surveyed earlier table processing methods--most directed at scanned printed tables rather than HTML tables--in [2]. An excellent survey from a different perspective was published by Zanibbi et al [3]. More recent approaches that share our objectives have been proposed in [4], [5], and [6]. Advanced approaches to document image analysis that can be potentially applied to tables are reported in [7] and [8].

After providing some definitions, we describe a simple method of extracting Header Paths from CSV versions of web tables from large statistical sites. Then we show how the structure of the table is revealed by a decomposition that incorporates algebraic factoring. The next section on analysis illustrates the application of Header Paths to relational tables. Our experiments to date demonstrate the extraction of header paths, the factorization, and the resulting Wang categories. Examination of the results suggests modifications of our algorithms that will most increase the fraction of automatically analyzed tables, and enhancements of the functionality of the interactive interface necessary to correct residual errors.

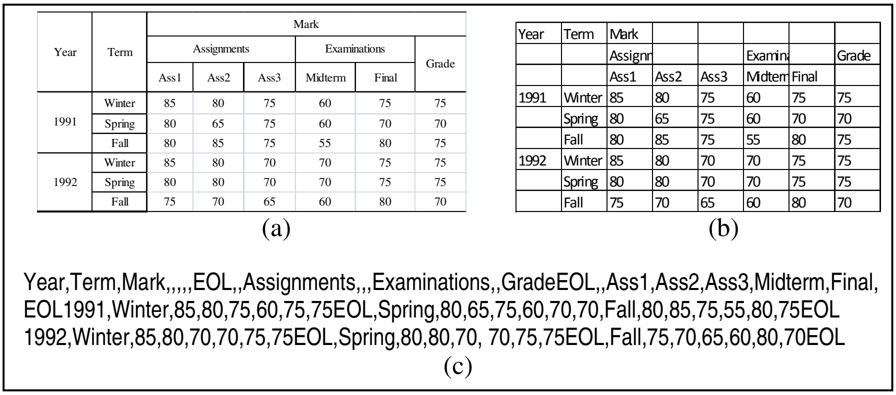
2 Header Paths

We extract Header Paths from CSV versions of web tables imported into Excel, which is considerably simpler than extracting them directly from html [9]. The transformation into the standard comma-separated-variables format for information exchange is not, however, entirely lossless. Color, typeface, type size, type style (bold, italics), and layout (e.g. indentations not specified explicitly in an HTML style sheet) within the cells are lost. Merged table cells are divided into elementary cells and the cell content appears only in the first element of the row. Anomalies include demoted superscripts that alter cell content. Nevertheless enough of the structure and content of the web tables is usually preserved for potentially complete understanding of the table. Some sites provide both HTML and CSV versions of the same table.

An example table is shown in Fig. 1a. The (*row-*) *stub* contains Year and Term, the *row header* is below the stub, the *column header* is to the right of the stub, and the 36 *delta* (or *data,* or *content) cells* are below the column header. Our Python routines convert the CSV file (e.g. Figs. 1b and 1c) to Header Paths as follows:

1. Identify the stub header, column header, row header, and content regions.
2. Eliminate blank rows and almost blank rows (that often designate units).
3. Copy into blank cells the contents of the cell above. } *reverse for rows*
4. Copy into blank cells the contents of the cell to the left.

5. Underscore blanks within cells and add quote marks to cell contents.
6. Mark row headers roots in the stub with negative column coordinates.
7. Add cell identifiers (coordinates) to each cell.
8. Trace the column and row Header Paths.



Year	Term	Mark					
		Assignments			Examinations		Grade
		Ass1	Ass2	Ass3	Midterm	Final	
1991	Winter	85	80	75	60	75	75
	Spring	80	65	75	60	70	70
	Fall	80	85	75	55	80	75
1992	Winter	85	80	70	70	75	75
	Spring	80	80	70	70	75	75
	Fall	75	70	65	60	80	70

(a)

Year	Term	Mark					
		Assignm			Examina		Grade
		Ass1	Ass2	Ass3	Midterm	Final	
1991	Winter	85	80	75	60	75	75
	Spring	80	65	75	60	70	70
	Fall	80	85	75	55	80	75
1992	Winter	85	80	70	70	75	75
	Spring	80	80	70	70	75	75
	Fall	75	70	65	60	80	70

(b)

```
Year,Term,Mark,,,,,EOL,,Assignments,,,Examinations,,GradeEOL,,Ass1,Ass2,Ass3,Midterm,Final,
EOL1991,Winter,85,80,75,60,75,75EOL,Spring,80,65,75,60,70,70,Fall,80,85,75,55,80,75EOL
1992,Winter,85,80,70,70,75,75EOL,Spring,80,80,70, 70,75,75EOL,Fall,75,70,65,60,80,70EOL
```

(c)

Fig. 1. (a) A table from [1] used as a running example. (b) Its CSVversion. (c) CSV file string.

Step 1 can partition a table into its four constituent regions only if either the stub header is empty, or the content cells contain only digits. Otherwise the user must click on the top-left (*critical*) delta cell. Fig. 2 shows the paths for the example table.

The paths to the body of the table (delta cells) are traced in the same way. The combination of Header Paths uniquely identifies each delta cell. The three Wang Categories for this table are shown in Fig. 3. The algorithm yielded paths for 89 tables on a random sample of 107 tables from our collection of 1000 web tables. Most of the rejected tables had non-empty stub headers and blank separators or decimal commas.

```
colpaths =
 (("<0,2>Mark"*"<1,2>Assignments"*"<2,2>Ass1")
+("<0,3>Mark"*"<1,3>Assignments"*"<2,3>Ass2")
+("<0,4>Mark"*"<1,4>Assignments"*"<2,4>Ass3")
+("<0,5>Mark"*"<1,5>Examinations"*"<2,5>Midterm")
+("<0,6>Mark"*"<1,6>Examinations"*"<2,6>Final")
+("<0,7>Mark"*"<1,7>Grade"*"<2,7>Grade"));

rowpaths =
(("<-2,3>Year"*"<-1,3>1991"*"<0,3>Term"*"<1,3>Winter")
+("<-2,4>Year"*"<-1,4>1991"*"<0,4>Term"*"<1,4>Spring")
+("<-2,5>Year"*"<-1,5>1991"*"<0,5>Term"*"<1,5>Fall")
+("<-2,6>Year"*"<-1,6>1992"*"<0,6>Term"*"<1,6>Winter")
+("<-2,7>Year"*"<-1,7>1992"*"<0,7>Term"*"<1,7>Spring")
 +("<-2,8>Year"*"<-1,8>1992"*"<0,8>Term"*"<1,8>Fall"));
```

Fig. 2. Column Header and Row Header Paths for the table of Fig. 1

- Year
  - 1991
  - 1992
- Term
  - Winter
  - Spring
  - Fall

- Mark
  - Assignments
    - Ass1
    - Ass2
    - Ass3
  - Examinations
    - Midterm
    - Final
  - Grade*Grade

Fig. 3. Wang Categories for the table of Fig. 1

3 Algebraic Formulation and Factorization

The column and row Header Paths can be thought of as providing *indexing* relationships respectively, to the set of columns and rows in the table; their cross-product narrows these relationships to individual entries in the table. In this section, we develop an algebra of relations for Header Paths, which allows interpreting them in terms of categorical hierarchies using standard symbolic mathematical tools. As the essential formalism for row Header Paths is the same as it is for column Header Paths, we limit the exposition below to column Header Paths. We will use the example table shown in Fig. 1a with its column and row Header Paths shown in Fig. 2.

In our algebra of relations for column Header Paths, the domain of the relations is the set Γ of all columns in the table. Each cell c in the column header covers a subset of the columns in Γ. By identifying cells with their labels, the column-covering relation can be extended to cell labels.

In the example, the label Assignments covers columns 1, 2, and 3, and the label Mark covers all of Γ. These cell-label relations can be combined, using the union and intersection operations, to define new relations. We use the symbols + and *, respectively, for the union and intersection operations. Thus, the expression Assignments+Grade covers columns 1, 2, 3, and 6 and the expression Examinations*Midterm covers just column 4. We call the first a sum and the second a product relation. It will be seen that the collection of Header Paths, described in the previous section, can be represented as a sum of products (SOP) expression in our algebra. We will call this SOP an indexing relation on Γ because it covers all columns in Γ and each product term covers one column in Γ. This SOP is also an irredundant relation because each column is uniquely covered by one product term.

The notions of indexing and irredundancy can be extended to arbitrarily nested expressions by expanding the expression into its SOP-equivalent. For example, Mark*(Assignments*Ass1+Examinations*(Midterm+Final)) is an irredundant indexing relation on the column set {1,4,5}.

We will use the term *decomposition* to refer to converting a relational expression into a hierarchy of sub-expressions that are joined together by the union and intersection operations. Equivalent terms, *simplification* (in contrast to expansion) and *Horner nested representation* [10], are also used in the literature to denote the same concept. Now, we can formulate the problem of recovering a categorical structure of the column header as the decomposition of the column Header Paths expression into an equivalent expression *E* satisfying the following constraints:

(a) *E* is an indexing relation on Γ;
(b) *E* is irredundant ;
(c) *E* is minimal in the occurrence of the total number of labels in the expression.

A benefit of this formulation is that the decomposition problem for algebraic expressions has attracted attention for a long time from fields ranging from symbolic mathematics [11,12,13] to logic synthesis [14,15] and programs incorporating the proposed solutions are widely available [16,17]. In this work, we adopt Sis, a system for sequential circuit synthesis, for decomposition of header-path expressions [16]. Sis represents logic functions to be optimized in the form of a network with nodes representing the logic functions and directed edges (*a,b*) to denote the use of the function at node *a* as a sub-function at node *b*. An important step in network optimization is extracting, by means of a division operation, new nodes representing logic functions that are factors of other nodes. Because all good Boolean division algorithms are computationally expensive, Sis uses the ordinary algebraic division. The basic idea used is to look for expressions that are observed many times in the nodes of the network and extract such expressions.

Some caution is in order for this formulation because seemingly identical labels in two cells may carry different meaning. In our experience with many web tables, however, identical labels in the same row of a column header (or the same column of a row header) do not need to be differentiated for our Boolean-algebraic formulation. This is because either they carry the same meaning (e.g. repeated labels like Winter in the example table) or the differences in their meaning are preserved by other labels in their product terms. To illustrate the last point, assume ITEMS*TOTAL and AMOUNT*TOTAL are two product terms in a Header Paths expression, where the two occurrences of TOTAL refer to a quantity and a value, respectively. In the factored form, these two terms might appear as TOTAL*(ITEMS+AMOUNT), where the cell with label TOTAL now spans cells labeled ITEMS and AMOUNT, thus preserving the two terms in the Header Paths expression. On the other hand, suppose the column HEADER PATHS expression included TOTAL*MALES + TOTAL*FEMALES + TOTAL*TOTAL as a subexpression, where the last term spans the two rows corresponding to the first two terms. In Boolean algebra, the sub-expression could be simplified to TOTAL and the resulting expression would no longer cover the two columns covered by the first two terms. With the row indices attached to labels, the subexpression becomes: TOTAL1*MALES + TOTAL1*FEMALES + TOTAL1*TOTAL2, which cannot be simplified so as to eliminate one or more terms.

We illustrate the decomposition obtained by Sis for the column Header Paths of the example table:

```
Input Order = Mark Assignments Ass1 Ass2 Ass3 Examinations Midterm
              Final Grade
colpaths = Mark*[Examinations*[Final + Midterm]
              + Assignments*   [Ass3 +  Ass2 + Ass1] + Grade]
```

Note that Sis does not preserve the input order in the output equations, but because the order is listed in the output, we can permute the terms of the output expression according to the left-to-right, top-to-bottom order of the labels occurring in the table:

```
colpaths=Mark*[Assignments*[Ass1+Ass2+Ass3]                +
        [Examinations*[Midterm + Final]] + Grade]
```

Similarly, the rowpaths expression produces the following output:

```
rowpaths = Year*[ 1991+1992]*Term*[Winter+Spring+Fall]
```

We apply a set of Python regular expression functions to split each equation into a list of product terms and convert each product term into the form of [root, children]. A virtual header is inserted whenever a label for the category root is missing in the table, e.g. if the right-hand side of colpaths were missing "Mark*", we would insert a virtual header for the missing root. Then, the overall equation of the table can be printed out recursively in the form of Wang categories trees, as illustrated in Fig. 3. Further, we produce a canonical expression for the three categories in the form in Fig. 4, which is used in the generation of relational tables:

Mark*(Assignments*(Ass1+Ass2Ass3)+Examinations*(Midterm+Final)+Grade)
+ Year*(1991+1992)
+ Term*(Winter+Spring+Fall)

Fig. 4. Canonical expression for the table of Fig. 1

Currently, we don't have an automated way of verifying the visual table (VT) in CSV form against the category tree (CT) form obtained by factorization. Instead, we do the verification by comparing the following common characteristic features of the two forms: the number of row and column categories, and for each category: fanout at the root level, the total fanout, and whether the root category has a real or virtual label. The data for CT are unambiguous and readily derived. For VT, however, we derive it by labor-intensive and error-prone visual inspection, which limits the size and the objectiveness of the experiment. Still, we believe, the result demonstrates the current status of the proposed scheme for automated conversion of web tables.

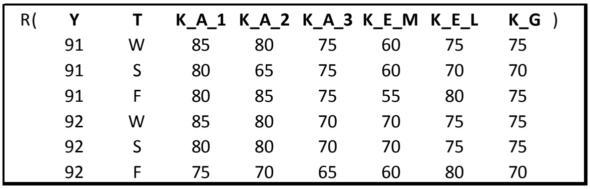
Of 89 tables for which Header Paths were generated, 66 (74%) have correct row and column categories. Among the remaining 23 tables, 18 have either row *or* column category correct, including two where the original table contained mistakes (duplicate row entries). Both human and computer found it especially difficult to detect the subtle visual cues (indentation or change in type style) that indicate row categories. Since in principle the decomposition is error-free given the correct Header Paths, we are striving to improve the rowpath extraction routines.

4 Generation of Relational Tables

Given the headers of a table, factored into canonical form as explained in Section 3 along with their accompanying data (the delta cells), we can transform the table into a relational table for a relational database. We can then query the table with SQL or any other standard database query language. Given a collection of tables transformed into relations, we may also be able to join tables in the collection and otherwise manipulate the tables as we do in a standard relational database.

Our transformation of a factored table assumes that one of the Wang categories provides the attributes for the relational table while the remaining categories provide key values for objects represented in the original table. Without input from an oracle, we do not know which of the Wang categories would serve best for the attributes. We therefore transform a table with *n* Wang category trees into *n* complementary relational tables—one for each choice of a category to serve as the attributes.

The transformation from a factored table with its delta cells associated with one of its categories is straightforward. We illustrate with the example from Wang [1] in Fig. 1a This table has three categories and thus three associated relational tables (attributes can be permuted). We obtain the relational table in Fig. 5 using the factored expression in Fig. 4 and the data values from the original table in Fig. 1.



R(	Y	T	K_A_1	K_A_2	K_A_3	K_E_M	K_E_L	K_G	)
	91	W	85	80	75	60	75	75	
	91	S	80	65	75	60	70	70	
	91	F	80	85	75	55	80	75	
	92	W	85	80	70	70	75	75	
	92	S	80	80	70	70	75	75	
	92	F	75	70	65	60	80	70	

Fig. 5. Relational table for the table of Fig. 1

In Fig. 5, R is the title of the table ("The_average_marks_for_1991_to_1992"), Y is Year, T is Term, K is Mark, A is Assignment, 1–3 are Assignment numbers, E is Examinations, M is Midterm, L is Final, G is Grade, 91 and 92 are year values, and the terms are W for Winter, S for Spring, and F for Fall.

The key for the table is {Y, T}, a composite key with two attributes because we have two categories for key values. Since YT is the key, the objects for which we have attribute values are terms in a particular year, e.g., the Winter 1991 term. The attributes values are average marks for various assignments and examinations and for the final grade for the term.

In general, the algorithmic transformation to a relational table is as follows: (1) Create attributes from the first category-tree expression by concatenating labels along each path from root to leaf (e.g., Mark*(Assignments*(Ass1+ ... becomes the attribute Mark_Assignments_Ass1). (2) Take each root of the remaining category-tree expressions as additional attributes, indeed as (composite) primary-key attributes (e.g., Year and Term as attributes with the composite primary key {Year, Term}). (3) Form key values from the (cross product of) the labels below the root of the category-tree expressions (e.g., {1991, 1992} × {Winter, Spring Fall}). (4) Fill in the remaining table values from the delta-cell values in the original table as indexed by the header labels (e.g. for the attribute Mark_Assignment_Ass1 and the composite key value Year = 1991 and Term = Winter, the delta-cell delta-cell value 85).

Given the relational table in Fig. 5, we can now pose queries with SQL:

Query 1: What is the average grade for Fall, 1991?

```
select K_G
from R
where Y = 91 and T = “F”
```

With abbreviated names spelled out, the query takes on greater meaning:

```
select Mark_Grade
from The_average_marks_for_1991_to_1992
where Year = 1991 and Term = “Fall”
```

The answer for Query 1 is:

Mark_Grade
75

Query 2:

What is the overall average of the final for each year?

```
select Year, avg(Mark_Examinations_Final)
from The_average_marks_for_1991_to_1992
group by Year
```

The answer for Query 2 is:

Year	Avg(Mark_Examinations_Final)
1991	75.0
1992	76.7

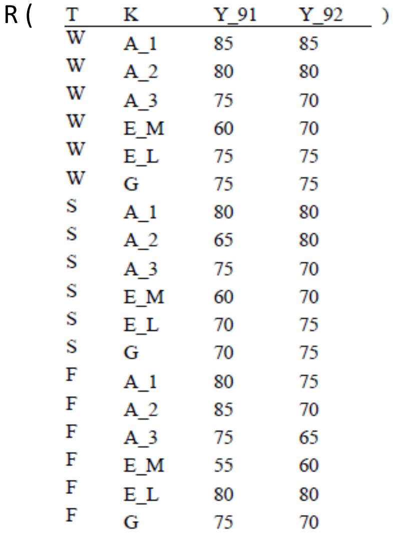
Commuting the expression in Fig. 4 to let Term*(Winter+Spring+Fall) be first and stand for the attributes, and to let the Year expression be second and the Mark expression be third, this yields the relational table in Fig. 6a.

In this relational table the key is YK, the composite key Year-Mark. Thus, we have Year-Mark objects with Term attributes. Although less intuitive than our earlier choice, it certainly makes sense to say that the Winter term average mark for the 1991 Assignment #1 is 85.

Commuting the expression in Fig. 4 to let Year*(1991+1992) be first and stand for the attributes, and to let the Term expression be second and the Mark expression be third, yields the relational table in Fig. 6b.In this relational table the key is TK, the composite key Term-Mark. Thus, we have Term-Mark objects with Year attributes. Here, again, although less intuitive than our first choice, it makes sense to say that the 1991 average mark for the Winter term Assignment #1 is 85.

R (	Y	K	T_W	T_S	T_F	)
	91	A_1	85	80	80	
	91	A_2	80	65	85	
	91	A_3	75	75	75	
	91	E_M	60	60	55	
	91	E_L	75	70	80	
	91	G	75	70	75	
	92	A_1	85	80	75	
	92	A_2	80	80	70	
	92	A_3	70	70	65	
	92	E_M	70	70	60	
	92	E_L	75	75	80	
	92	G	75	75	70	

(a)



R (	T	K	Y_91	Y_92	)
	W	A_1	85	85	
	W	A_2	80	80	
	W	A_3	75	70	
	W	E_M	60	70	
	W	E_L	75	75	
	W	G	75	75	
	S	A_1	80	80	
	S	A_2	65	80	
	S	A_3	75	70	
	S	E_M	60	70	
	S	E_L	70	75	
	S	G	70	75	
	F	A_1	80	75	
	F	A_2	85	70	
	F	A_3	75	65	
	F	E_M	55	60	
	F	E_L	80	80	
	F	G	75	70	

(b)

Fig. 6. (a) A second relational table for the table of Fig. 1. (b) A third relational table

5 Discussion

We propose a natural approach for table analysis based on the indexing of data cells by the column and row header hierarchies. Each data cell is defined by paths through every header cell that spans that data cell. Automated extraction of paths from CSV versions of web tables must first locate the column and row headers via identification of an empty stub or by finding the boundaries of homogeneous rows and columns of content cells. It must also compensate for the splitting of spanning cells in both directions, and for spanning unit cells. We define a relational algebra in which the collection of row or column Header Paths is represented by a sum-of-products expression, and we show that the hierarchical structure of the row or column categories can be recovered by a decomposition process that can be carried out using widely available symbolic mathematical tools.

We demonstrate initial results on 107 randomly selected web tables from our collection of 1000 web tables from large sites. The experiments show that in 83% of our sample the header regions can be found using empty stubs or numerical delta cells. We estimate that the header regions can be found in at least a further 10%-15% with only modest improvements in our algorithms. The remainder will still require clicking on one or two cells on an interactive display of the table.

Most of the extracted column Header Paths are correct, but nearly 25% of the row Header Paths contain some mistake (not all fatal). The next step is more thorough analysis of indentations and of header roots in the stub. Interactive path correction is far more time consuming than interactive table segmentation. An important task is to develop adequate confidence measures to avoid having to inspect every table.

In addition to algorithmic extraction of an index, we show how to manipulate sum-of-products expressions along with a table's delta-cells to yield relational tables that can be processed by standard relational database engines. Thus, for tables our algorithms can interpret, we automate the process of turning human-readable tables into machine-readable tables that can be queried, searched, and combined in a relational database.

Acknowledgments. This work was supported by the National Science Foundation under Grants # 044114854 (at Rensselaer Polytechnic Institute) and 0414644 (at Brigham Young University) and by the Rensselaer Center for Open Software. We are thankful to students Landy Zhang, Douglas Mosher, Stephen Poplasky (RPI), Dongpu Jin (UNL), and Spencer Machado (BYU) for collecting and analyzing data and for coding the transformation algorithms.

References

1. Wang, X.: Tabular Abstraction, Editing, and Formatting, Ph.D Dissertation, University of Waterloo, Waterloo, ON, Canada (1996)
2. Embley, D.W., Hurst, M., Lopresti, D., Nagy, G.: Table Processing Paradigms: A Research Survey. Int. J. Doc. Anal. Recognit. 8(2-3), 66–86 (2006)
3. Zanibbi, R., Blostein, D., Cordy, J.R.: A survey of table recognition: Models, observations, transformations, and inferences. International Journal of Document Analysis and Recognition 7(1), 1–16 (2004)
4. Krüpl, B., Herzog, M., Gatterbauer, W.: Using visual cues for extraction of tabular data from arbitrary HTML documents. In: Proceedings. of the 14th Int'l Conf. on World Wide Web, pp. 1000–1001 (2005)
5. Pivk, A., Ciamiano, P., Sure, Y., Gams, M., Rahkovic, V., Studer, R.: Transforming arbitrary tables into logical form with TARTAR. Data and Knowledge Engineering 60(3), 567–595 (2007)
6. Silva, E.C., Jorge, A.M., Torgo, L.: Design of an end-to-end method to extract information from tables. Int. J. Doc. Anal. Recognit. 8(2), 144–171 (2006)
7. Esposito, F., Ferilli, S., Di Mauro, N., Basile, T.M.A.: Incremental Learning of First Order Logic Theories for the Automatic Annotations of Web Documents. In: Proceedings of the 9th International Conference on Document Analysis and Recognition (ICDAR-2007), Curitiba, Brazil, September 23-26, pp. 1093–1097. IEEE Computer Society, Los Alamitos (2007); ISBN 0-7695-2822-8, ISSN 1520-5363
8. Esposito, F., Ferilli, S., Basile, T.M.A., Di Mauro, N.: Machine Learning for Digital Document Processing: From Layout Analysis To Metadata Extraction. In: Marinai, S., Fujisawa, H. (eds.) Machine Learning in Document Analysis and Recognition. SCI, vol. 90, pp. 79–112. Springer, Berlin (2008); ISBN 978-3-540-76279-9
9. Jandhyala, R.C., Krishnamoorthy, M., Nagy, G., Padmanabhan, R., Seth, S., Silversmith, W.: From Tessellations to Table Interpretation. In: Carette, J., Dixon, L., Coen, C.S., Watt, S.M. (eds.) MKM 2009, Held as Part of CICM 2009. LNCS, vol. 5625, pp. 422–437. Springer, Heidelberg (2009)
10. `http://www.mathworks.com/help/toolbox/symbolic/horner.html`
11. Fateman, R. J.: Essays in Symbolic Simplification. MIT-LCS-TR-095, 4-1-1972, `http://publications.csail.mit.edu/lcs/pubs/pdf/MIT-LCS-TR-095.pdf` (downloaded November 10, 2010)

12. Knuth, D.E.: 4.6.2 Factorization of Polynomials". Seminumerical Algorithms. In: The Art of Computer Programming, 2nd edn., pp. 439–461, 678–691. Addison-Wesley, Reading (1997)
13. Kaltofen, E.: Polynomial factorization: a success story. In: ISSAC 2003 Proc. 2003 Internat. Symp. Symbolic Algebraic Comput. [-12], pp. 3–4 (2003)
14. Brayton, R.K., McMullen, C.: The Decomposition and Factorization of Boolean Expressions. In: Proceedings of the International Symposium on Circuits and Systems, pp. 49–54 (May1982)
15. Vasudevamurthy, J., Rajski, J.: A Method for Concurrent Decomposition and Factorization of Boolean Expressions. In: Proceedings of the International Conference on Computer-Aided Design, pp. 510–513 (November 1990)
16. Sentovich, E.M., Singh, K.J., Lavagno, L., Moon, C., Murgai, R., Saldanha, A., Savoj, H., Stephan, P.R., Brayton, R.K., Sangiovanni-Vincentelli, A.L.: SIS: A System for Sequential Circuit Synthesis. In: Memorandum No. UCB/ERL M92/41, Electronics Research Laboratory, University of California, Berkeley (May 1992), http://www.eecs.berkeley.edu/Pubs/TechRpts/1992/ERL-92-41.pdf (downloaded November 4, 2010)
17. (Quickmath-ref), http://www.quickmath.com/webMathematica3/quickmath/page.jsp?s1=algebra&s2=factor&s3=advanced (last accessed November 12, 2010)