International Journal of Document Analysis (2006) 8(2) 66–86
DOI 10.1007/s10032-006-0017-x

ORIGINAL PAPER

David W. Embley · Matthew Hurst ·
Daniel Lopresti · George Nagy

Table-processing paradigms: a research survey

Received: 9 January 2005 / Accepted: 1 January 2006 / Published online: 9 May 2006

Abstract Tables are a ubiquitous form of communication. While everyone seems to know what a table is, a precise, analytical definition of "tabularity" remains elusive because some bureaucratic forms, multicolumn text layouts, and schematic drawings share many characteristics of tables. There are significant differences between typeset tables, electronic files designed for display of tables, and tables in symbolic form intended for information retrieval. Most past research has addressed the extraction of low-level geometric information from raster images of tables scanned from printed documents, although there is growing interest in the processing of tables in electronic form as well. Recent research on table composition and table analysis has improved our understanding of the distinction between the logical and physical structures of tables, and has led to improved formalisms for modeling tables. This review, which is structured in terms of generalized paradigms for table processing, indicates that progress on half-a-dozen specific research issues would open the door to using existing paper and electronic tables for database update, tabular browsing, structured information retrieval through graphical and audio interfaces, multimedia table editing, and platform-independent display.

Keywords Document analysis · Table recognition · Table understanding

D. W. Embley
Computer Science Department, Brigham Young University,
Provo, UT 84602, USA

M. Hurst
Intelliseek Applied Research Center, Pittsburgh, PA 15213, USA

D. Lopresti (✉)
Department of Computer Science and Engineering, Lehigh University,
Bethlehem, PA 18015, USA

G. Nagy
Department of Electrical, Computer, and Systems Engineering,
Rensselaer Polytechnic Institute, Troy, NY 12180, USA

1 Introduction

1.1 Why tables?

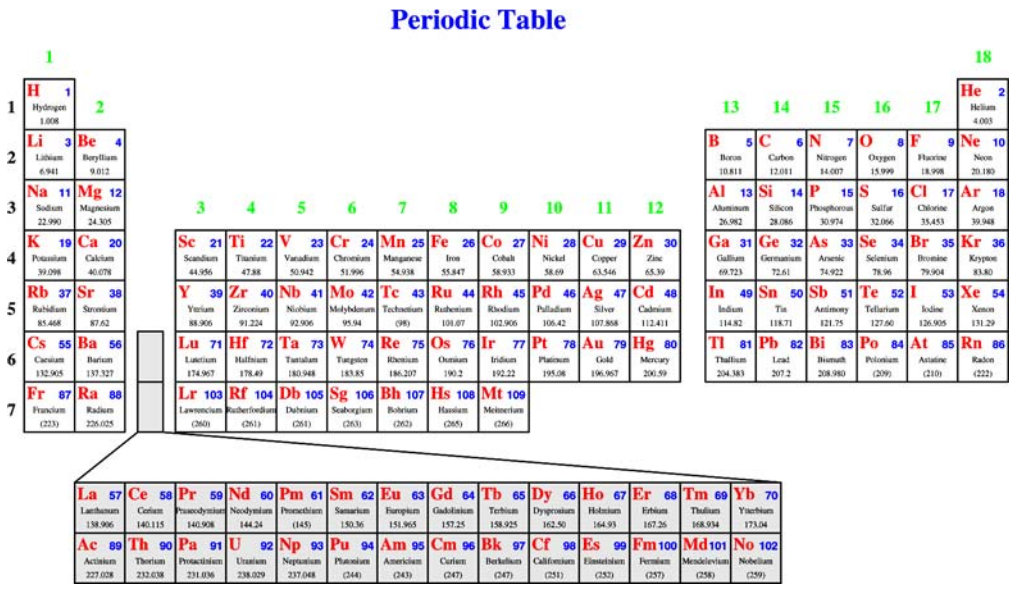
Tables are the prevalent means of representing and communicating structured data. They may contain words, numbers, formulae, and even graphics. Developed originally in the days of printed or handwritten documents (indeed, tables may pre-date sentential text [47]), they have been adapted to word processors and page composition languages, and form the underlying paradigm for spreadsheets and relational database systems.

Some common examples of data usually presented in the form of tables are calendars, rail and flight schedules, financial reports, experimental results, and grade reports. It is worth noting that the need to reformat and analyze the 1890 U.S. Census forms launched the punched-card "tabulator" industry. Electronic computers were commissioned during WWII for computing ballistic tables. The major commercial applications envisioned for computers in the 1950s centered on database manipulation, which remains the mainstay of business data processing.

The other common representation for structured data is a list. If we consider ordered lists analogous to vectors, then we can think of tables as analogous to matrices. Unlike vectors and matrices, lists and tables may contain non-numeric data items. Graphs are required for relationships more complex than can be represented by tables and are used primarily for *inter*-document structure. Trees are often used to represent *intra*-document structure.

Note that not all tables can be easily interpreted using only common sense: consider, for instance, the Periodic Table of the Elements, which requires substantial domain knowledge to understand (see Table 1). It is exceedingly easy to come up with other examples that are challenging even from a human perspective. Rather than belabor this point, for the purposes of this survey, we choose instead to focus on the kinds of tables that researchers have addressed with some degree of success.

Table 1 The periodic table of the elements



Periodic Table

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	H 1 Hydrogen 1.008																	He 2 Helium 4.003
2	Li 3 Lithium 6.941	Be 4 Beryllium 9.012											B 5 Boron 10.811	C 6 Carbon 12.011	N 7 Nitrogen 14.007	O 8 Oxygen 15.999	F 9 Fluorine 18.998	Ne 10 Neon 20.180
3	Na 11 Sodium 22.990	Mg 12 Magnesium 24.305											Al 13 Aluminum 26.982	Si 14 Silicon 28.086	P 15 Phosphorous 30.974	S 16 Sulfur 32.066	Cl 17 Chlorine 35.453	Ar 18 Argon 39.948
4	K 19 Potassium 39.098	Ca 20 Calcium 40.078	Sc 21 Scandium 44.956	Ti 22 Titanium 47.88	V 23 Vanadium 50.942	Cr 24 Chromium 51.996	Mn 25 Manganese 54.938	Fe 26 Iron 55.847	Co 27 Cobalt 58.933	Ni 28 Nickel 58.69	Cu 29 Copper 63.546	Zn 30 Zinc 65.39	Ga 31 Gallium 69.723	Ge 32 Germanium 72.61	As 33 Arsenic 74.922	Se 34 Selenium 78.96	Br 35 Bromine 79.904	Kr 36 Krypton 83.80
5	Rb 37 Rubidium 85.468	Sr 38 Strontium 87.62	Y 39 Yttrium 88.906	Zr 40 Zirconium 91.224	Nb 41 Niobium 92.906	Mo 42 Molybdenum 95.94	Tc 43 Technetium (98)	Ru 44 Ruthenium 101.07	Rh 45 Rhodium 102.906	Pd 46 Palladium 106.42	Ag 47 Silver 107.868	Cd 48 Cadmium 112.411	In 49 Indium 114.82	Sn 50 Tin 118.71	Sb 51 Antimony 121.75	Te 52 Tellurium 127.60	I 53 Iodine 126.905	Xe 54 Xenon 131.29
6	Cs 55 Caesium 132.905	Ba 56 Barium 137.327	Lu 71 Lutetium 174.967	Hf 72 Halfnium 178.49	Ta 73 Tantalum 180.948	W 74 Tungsten 183.85	Re 75 Rhenium 186.207	Os 76 Osmium 190.2	Ir 77 Iridium 192.22	Pt 78 Platinum 195.08	Au 79 Gold 196.967	Hg 80 Mercury 200.59	Tl 81 Thallium 204.383	Pb 82 Lead 207.2	Bi 83 Bismuth 208.980	Po 84 Polonium (209)	At 85 Astatine (210)	Rn 86 Radon (222)
7	Fr 87 Francium (223)	Ra 88 Radium 226.025	Lr 103 Lawrencium (260)	Rf 104 Rutherfordium (261)	Db 105 Dubnium (261)	Sg 106 Seaborgium (263)	Bh 107 Bohrium (262)	Hs 108 Hassium (265)	Mt 109 Meitnerium (266)									

La 57 Lanthanum 138.906	Ce 58 Cerium 140.115	Pr 59 Praseodymium 140.908	Nd 60 Neodymium 144.24	Pm 61 Promethium (145)	Sm 62 Samarium 150.36	Eu 63 Europium 151.965	Gd 64 Gadolinium 157.25	Tb 65 Terbium 158.925	Dy 66 Dysprosium 162.50	Ho 67 Holmium 164.93	Er 68 Erbium 167.26	Tm 69 Thulium 168.934	Yb 70 Ytterbium 173.04
Ac 89 Actinium 227.028	Th 90 Thorium 232.038	Pa 91 Protactinium 231.036	U 92 Uranium 238.029	Np 93 Neptunium 237.048	Pu 94 Plutonium (244)	Am 95 Americium (243)	Cm 96 Curium (247)	Bk 97 Berkelium (247)	Cf 98 Californium (251)	Es 99 Einsteinium (252)	Fm 100 Fermium (257)	Md 101 Mendelevium (258)	No 102 Nobelium (259)

Table-processing paradigms are part of the older field of document image analysis [67]. A common objective of finding and delimiting tables, equations and illustrations, is to clear the path for optical character recognition (OCR) or, if the document is already in electronic form, for text analysis. Tables are between text and graphics with regard to the relative proportion of alphanumeric symbols, linear components and white space. If the text is sufficient for the purpose at hand, then all other document components can simply be either eliminated or preserved only in some image format. That is indeed adequate for many keyword-based document retrieval applications.

However, if some or all of the essential information resides in tables, then the tables themselves must be processed. There is little published research on OCR for tables. Current commercial OCR systems are hampered by the non-uniform spacing, multiple typefaces and sizes, rulings, and lack of language context found in tables. Without special provisions for tables, the OCR format preprocessor may simply attempt to decolumnize them. Some OCR products offer a *zoning* feature that marks and avoids table regions. More sophisticated systems attempt to transform tables, without further analysis, into the table format of the target representation (like Microsoft Word), but the results tend to be unreliable. None of these alternatives are satisfactory for table-rich documents: hence this special issue.

1.2 Rationale for this review

It appears likely that the automated or semi-automated interconversion of tables from one medium to another (e.g., from paper or electronic text to a spreadsheet, database, or query–answer system), or from one format to another in the same medium (e.g., for different display sizes) will prove desirable in a variety of computing environments. In some applications, it may be advantageous to query and reference tabular data without regard to the underlying medium or form.

Reflecting this growing interest, a number of surveys on table processing have appeared over the past several years [37, 47, 62, 63, 97]. The review by Lopresti and Nagy [62] aims at exploring the diversity and extent of the table world, and the many areas where further progress is needed to make the transition between traditional tables and digital presentation of structured information. A large collection of examples is included to illustrate the difficulty of both human and machine understanding of many tables. As an experiment, the entire survey was converted into tabular form for the version of the GREC proceedings later published as a separate book [63].

Handley's survey on document recognition [37] has sections on table recognition and forms recognition with accurate and detailed descriptions of many previously published algorithms.

The recent survey by Zanibbi et al. [97] includes references to much new material, organized according to a view of table recognition as the interaction of "models, observations, transformations, and inferences." Work in the area is partitioned according to the methods used for classification (e.g., decision trees and neural networks) and segmentation (e.g., clustering and grammars). This paper also has a useful section on performance evaluation.

Surveys of the information-organizational aspects of tables are included in Hurst's Ph.D. thesis [47] and a recent paper by Embley et al. [28]. At the time of writing the paper, another extensive bibliography, compiled by Price, could be found on-line.[1]

[1] See http://iris.usc.edu/Vision-Notes/bibliography/char966.html

The present paper is an attempt to review not only what people have actually done with tables but also what they would like to do with them, what they cannot do, and what and how they think about them. Our object is to collect information about the composition, use, interpretation, and understanding of tables that may prove useful in the development of tools for manipulating tables presented in a variety of media.

1.3 Guide to the remainder of this paper

Our organizing principle is to attempt to orthogonalize the various issues, so as to be able to make independent decisions regarding algorithm development.

We begin by first considering the fundamental question: "What is a table?" Rather than get hung up on the complexities of what is certainly a deep and debatable issue, we take a pragmatic approach to our definition of tabularity, informed largely by what researchers have already been able to accomplish in the area of table understanding.

To lay the groundwork for the kinds of tables we shall consider, we then proceed to describe half-a-dozen applications that would result from new developments in table processing. This is followed by a brief overview of existing commercial approaches to the problems of table and forms processing.

We then discuss input media under the headings of "electronic" and "paper." The former can be further subdivided into plain ASCII and page-descriptor representations. Electronic tables such as those found in word processing documents, e-mail, Portable Document Format (PDF) files, and the Web, already have the content of the leaf cells in symbolic form, so OCR is not necessary, but the structure is seldom available in a convenient form. Tables on paper must be optically scanned for any type of automated processing.

The bulk of our paper is structured in terms of *paradigms for table processing*. Here we outline the major steps as a series of generic tasks to be performed. Most, but not all, past work can be cast in this framework, which provides a foundation for describing and discussing research results to this point in time.

In the final section, we summarize potential research directions.

2 Tables

2.1 What is a table?

Although many consider the idea of a table to be simple, careful study (e.g., [63]) reveals that the question "What constitutes a table?" is indeed difficult to answer. Several researchers have provided definitions. Peterman et al. [71], for example, state that "tables have a regular repetitive structure along one axis so that the data type is determined either by the horizontal or vertical indices." These definitions, however intuitive, do not provide a theoretical basis from which to work.

As recognized by Tijerino et al. [84], relational tables [21] do provide a theoretical basis for tables. Axiomatically, relations in a relational database can be considered to be tables in a canonical form. Using a standard, formal definition of a relational table [66, 84] shows how to define a canonical table as follows. A *schema* for a canonical table is a finite set $L = \{L_1, \ldots, L_n\}$ of label names or phrases, which are simply called *labels*. Corresponding to each label L_i, $1 \le i \le n$, is a set D_i, called the *domain* of L_i. Let $D = D_1 \cup \cdots \cup D_n$. A *canonical table* T is a set of functions $T = \{t_1, \ldots, t_m\}$ from L to D with the restriction that for each function $t \in T$, $t(L_i) \in D_i$, $1 \le i \le n$.

As is common for relational databases, we can display tables in two dimensions. When we display a table two dimensionally, we fix the order of the labels in the schema for each function and factor these labels to the top as column headers. Each row in the table constitutes the domain values for the corresponding labels in the column headers. Thus, for example, we can display the canonical table:

```
{{(LAST NAME, Smith), (INITIAL, J),
   (BIRTH DATE, 12/3 1988)},
{(LAST NAME, Barr), (INITIAL, K),
   (BIRTH DATE, 25/5 1975)}}
```

as Table 2 shows. Displayed in this form, a canonical table is simply called a *table*. Whether any format in which this same information may be displayed (e.g., as the set of sets illustrated earlier) should be called a "table" may be debatable. To avoid the argument, whenever there may be doubt, we can refer to the information as *table-equivalent data* [84]. Displayed in its usual way as depicted in Table 2, this information would certainly be called a table.

One consequence of this definition is that we can formally investigate the boundary conditions constituting degenerate table-equivalent data. When there is only one column, the table is more commonly called a *list*. When there are no domain-value rows, we may think of the empty table as a *form* with slots to be filled in. When there is only one row, we may think of the table as a *filled-in form*. If a label is missing (e.g., if either *LAST NAME* or *INITIAL* is missing), we may think of the label as being implicit. Common sense (e.g., the names look like names and the initials look like initials) and context (e.g., *BIRTH DATE* usually implies people with names) allow us to reconstruct missing labels, or at least synonymously equivalent missing labels. If all labels are missing, self-identifying data may allow us to reconstruct all implicit labels. If all labels are missing and all domain values are numbers, we think of the table as a *matrix*. Further, if a matrix has only one row or one column, we think of it as a *vector*, as noted previously.

Table 2 A simple canonical table

LAST NAME	INITIAL	BIRTH DATE
Smith	J	12/3 1988
Barr	K	25/5 1975

2.2 What is table understanding?

Another consequence of the formal definition for tables is that it leads directly to a formal definition of table understanding. A table is *understood* if we can recover the set of labels $L = \{L_1, \ldots, L_n\}$, the set of domains $D = \{D_1, \ldots, D_n\}$, and the set of functions $T = \{t_1, \ldots, t_m\}$ that each maps L to D. Often, we use a less-inclusive definition that does not require us to identify D and the individual domains, $D_1, \ldots, D_n$, that constitute D. In this case, recovering the label–value pairs for each function $t_i \in T$ is sufficient. Thus, for example, Table 2 is understood if we can recover the set of functions:

```
{{(LAST NAME, Smith), (INITIAL, J),
   (BIRTH DATE, 12/3 1988)},
{(LAST NAME, Barr), (INITIAL, K),
  (BIRTH DATE, 25/5 1975)}}
```

Experiments have shown that even human "experts" do not always agree on the sets of label–value pairs for a table [41]. Thus, we should not be surprised that automating table understanding is difficult.

Although the task may be challenging, the formal definition does tell us exactly what we have to do to automate table understanding: we must recover the label–value pairs from the representation of a given table. To formalize this process, we can adopt the ideas from [36], which proposes the use of an ontology for automated table understanding. Since "an ontology is a formal, explicit specification of a shared conceptualization" [34], a table understanding ontology formally and explicitly specifies a shared conceptualization of table understanding. Basically, the idea is to ontologically capture all the relevant representational knowledge about a table (the input ontology) and transform it algorithmically to sets of label–value pairs (the output ontology).

Figure 1 shows a graphical depiction of the output ontology for canonical tables. Later in this paper, when we describe table-processing paradigms, we will show how to represent input tables ontologically and explain how the paradigms can all be thought of as transforming an input table captured ontologically into an output ontology such as the one in Fig. 1. In the diagram, we use boxes to represent object sets, solid boxes for abstract items represented by object identifiers, and dotted boxes for concrete items represented by value strings. Thus, for example, *Table* in Fig. 1 is a set of table identifiers representing the tables of interest, and *Label* is a set of labels such as *LAST NAME* or *BIRTH DATE*. Hyperedges connecting object sets represent relationship sets. N-ary edges have a diamond; binary edges do not have a diamond. Edges may be functional, denoted by an arrowhead on their range side. Thus, the relationship between tuples and tables is functional: each tuple (identified by a tuple identifier) belongs to one and only one table. The absence of an arrowhead allows an object to participate with many other objects. Thus, in the n-ary relationship set, a tuple may have many label–value pairs. Additional constraints may further restrict object- or relationship sets. Thus, we can force the conceptualization to correspond to the formal definition of a table, which requires a distinct and equal set of labels for every tuple belonging to a particular table.

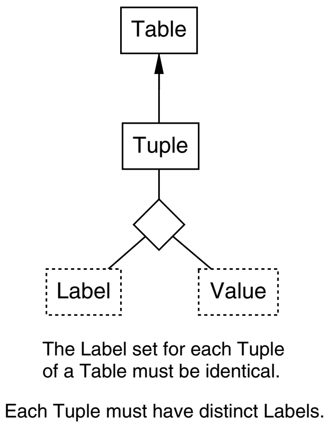


Table
Tuple
Label
Value
The Label set for each Tuple of a Table must be identical.
Each Tuple must have distinct Labels.

Fig. 1 An output ontology for tables [36]

2.3 Generalizing tables and table understanding

A further consequence of the formal definition for tables is generalizations of tables and table understanding. Indeed, some researchers have offered generalizations [47, 84]. One way we can formally extend the definition is by defining nested labels [47]. We can alter Table 2, for example, by nesting *LAST NAME* and *INITIAL* under *EMPLOYEE* and *BIRTH DATE* under *PENSION STATUS*. For nested tables, we can use a nesting structure to describe the more complex attribute value pairs.

In the following discussion we show the nested augmentation of Table 2 using the notation of [47]. Categories are made up of minimal sequences of dependent cells. Thus, in the extended version of Table 2, as EMPLOYEE has no discriminative power other than defining LAST NAME and INITIAL, it forms part of the values in that category. The reading set describes the subset of the cartesian product of the categories that the syntax of the table allows. A reading path, then, is a subset of the reading set with the specific data category removed.

Categories:

```
{EMPLOYEE.LAST NAME, EMPLOYEE.INITIAL},
  {PENSION STATUS.BIRTH DATE}, {Smith,
  Barr}, {J, K}, {12/3 1988, 25/5 1975}
```

Reading Set:

```
{{EMPLOYEE.LAST NAME, Smith,
  PENSION STATUS.BIRTHDATE,
  12/3 1988}, ...}
```

Table 3 shows a more complex kind of nesting. Two dimensions are used to index a third data category.[2] The two dimensions are *Position* {*First*, *Second*, *Third*} and

[2] Here the first and third categories are laid out vertically, and the second category horizontally. Many other possible permutations exist.

nucleotide {*U*, *C*, *A*, *G*}. Understanding the full label (or reading path) *Second Position C* as a complex object, not a simple string, is required if we are to interpret the table against a model of the domain. This table is described as follows:

Categories:

```
{First Position, Second Position,
   Third Position},
{Nucleotide.U, Nucleotide.C,
    Nucleotide.A, Nucleotide.G}
{Phe, Ser, Tyr, Cys, Leu, Stop, Trp,
   Pro, His, Arg, Gln, Ile, Thr, Asn,
   Lys, Met, Val, Ala, Asp, Gly, Glu}
```

Reading Set:

```
{{First Position, Nucleotide.U,
   Second Position, Nucleotide.U,
   Third Position Nucleotide.U, Phe},
    ...}
```

Another way we can formally extend the definition is by defining collections of tables, in which case we have the equivalent of a relational database [66]. Further, we can reverse engineer (e.g., [20]) a collection of tables into a conceptual model (e.g., the entity–relationship model [18]). In a similar vein, we can consider reverse engineering a single table or a group of related tables into an ontology and populate them as described in [84]. Using this technique, Fig. 2 shows the ontological representation for Table 2 that has *LAST NAME* and *INITIAL* nested under *EMPLOYEE* and *BIRTH DATE* nested under *PENSION STATUS*, and Fig. 3 shows the ontological representation for the nested genetic code in Table 3.

In Fig. 3, the black triangle denotes an aggregation of the elements connected to its base into the aggregate connected to its apex, while the open triangles denote ISA relationships between specialization elements connected to their bases and generalization elements connected to their apexes. The symbol '⊎' in an open triangle denotes a partition among the specialization elements with respect to the generalization element. The large black dots represent individual elements, thought of as singleton sets in the partition.

It is important to realize that we are using ontologies in two different ways in this discussion: (1) ontologies to represent understood tables (Figs. 2 and 3) and (2) ontologies to represent input and output descriptions of table knowledge (Fig. 1). Although much too complex to depict and describe in this survey, [27] gives an ontology that formally describes ontologies that represent understood tables. In this sense, ontologies that represent output descriptions are meta-ontologies. Indeed, the output ontology in Fig. 1 is a meta-description of a canonical relational table.

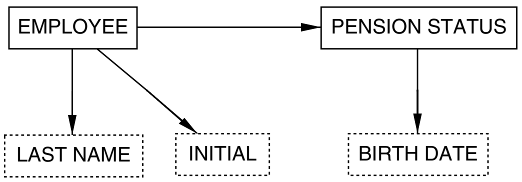


Fig. 2 An ontology for the nested EMPLOYEE table

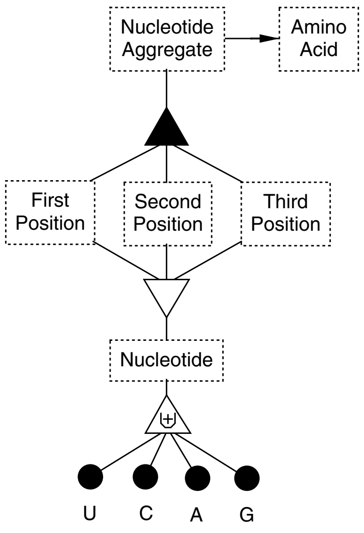


Fig. 3 An ontology for Table 3

Table 3 The genetic code: a mapping from triples of nucleotides (*codons*) to the amino acids they encode

	Second position				
First position	U	C	A	G	Third position
U	Phe	Ser	Tyr	Cys	U
	Phe	Ser	Tyr	Cys	C
	Leu	Ser	Stop	Stop	A
	Leu	Ser	Stop	Trp	G
C	Leu	Pro	His	Arg	U
	Leu	Pro	His	Arg	C
	Leu	Pro	Gln	Arg	A
	Leu	Pro	Gln	Arg	G
A	Ile	Thr	Asn	Ser	U
	Ile	Thr	Asn	Ser	C
	Ile	Thr	Lys	Arg	A
	Met	Thr	Lys	Arg	G
G	Val	Ala	Asp	Gly	U
	Val	Ala	Asp	Gly	C
	Val	Ala	Glu	Gly	A
	Val	Ala	Glu	Gly	G

2.4 Models of tables

Not all table-processing research aims at table understanding. For example, some may just want to convert a scanned table into an editable Microsoft Excel or Word table that has no meaning except to a human. A substantial amount of table processing to date has not attempted to interpret tables; rather, recovering the grid and cell contents are considered the target. The researchers in question did formulate adequate models based on their intended goals and their views of what constitutes a table even though these models are largely insufficient for the task of table understanding.

Many different models of tables have been proposed. This variation is generally in line with the particular tasks addressed by the systems described or the particular philosophy of document encoding. Low-level models use line-art [31], white space [78], and character distributions [51] as key features to drive analysis. Grid-based models include [60] and [80].

Another class of table model is that which describes a specific table. This is not only a syntactic constraint but also a semantic constraint—the model fixes the labels of the positions in the data area of the table, thus obviating the need to interpret the labels either syntactically or semantically. The application context is one in which a known table (or small set of tables) is input many times and requires interpretation [80].

It is also possible to devise more general systems that can be customized to specific families of recursively or iteratively generated structured documents, including tables. Enhanced Position Formalism (EPF) is a 2-D grammar able to describe document layouts for sheet music, mathematical equations, and tables [24, 25]. Although the examples presented in this paper are forms rather than tables, it should be possible to compile EPF grammars for specific families of tables. The authors claim that the grammars can be readily combined, for example to recognize tables of mathematical equations. The method was applied to several thousand degraded 19th century military documents with similar layout.

The database model is an appealing analogy to human-authored tables. Green uses this analogy to refer to the printed table as *printed manifestations of relational information* [30]. He then continues by describing the complexities of the relational model in terms of joining and merging multiple relations. This effectively appeals to the implicit semantics behind the join as an analogy to the complex categorical structure present in all but the most trivial uniform grid table.

The Semantic and Representation Detection (SRD) framework is proposed for combining information from a domain ontology, and standard-unit ontology, and table metadata (possibly derived from surrounding context) into relational tables for populating a database [4]. It is not clear from the paper whether the SRD has been implemented.

Often the input format to a table-processing system limits the complexity of the tables, requiring a model of a suitably limited scope. Pyreddy and Croft [73] characterize tables in a typed-line-manner due to the limitations of an ASCII representation.

Moving from low-level structural models to more abstract models, we can see the influence of table editing systems. Wang's is perhaps the most well-known model which captures both logical and physical aspects [88]. She describes the Improv system [23] as being perhaps the first system which provided a clear separation of logical and physical aspects.

Extending some of the concepts presented in [88], Hurst [47] develops a characterization of tables as document objects in context, recognizing the potential for surrounding text to impact the understanding of the table. If the related text informs the reader that "the values in the second column are the median value," we read the data quite differently to the case where we have been told "the values in the second column are the maxima."

Finally, an account of table models is not complete without mentioning research in the field of psycholinguistics. Wright [95] describes the understanding of the *organizational principles* in tables, and Guthrie et al. [35] consider the nature of categories.

3 Applications of table processing

In this section, we separate applications into as many discrete categories as possible. It may or may not be advantageous to develop a table-processing framework that can handle several of these applications in a unified way.

3.1 Large-volume, homogeneous table conversion

An example of an application in this area is the work done at AT&T/Lucent on the conversion of telephone billing statements to a usable form [80]. Although the tables may vary in format and content, all contain similar types of data that are compatible with an existing database. The database itself can be used to facilitate and validate data extraction from the tables [29]. This application is very similar to forms processing and could probably make use of advanced existing commercial software developed for this purpose.

The authors of the aforementioned paper emphasize the importance of a well-designed graphical user interface (GUI) to allow customization of the table-processing tools for specific formats. The use of table templates eliminates the need for elaborate structure hypotheses, and the success of the approach depends mainly on thorough preprocessing and accurate OCR.

3.2 Large-volume, mixed table conversion

This is a preliminary step for data mining from sources that are available only as paper or electronic tables. This application may require table spotting and table-similarity detection in addition to content and structure extraction.

Note that a successful approach to table understanding could be used to facilitate what is regarded as traditional information retrieval. The answers to certain kinds of queries seem most naturally expressed in tabular form. Consider, for example, the following ad hoc topic (#219) from the TREC 4 evaluation [85]:

> How has the volume of U.S. imports of Japanese autos compared with exports of U.S. autos to Canada and Mexico?

A document relevant to such a query will likely contain a table comparing auto imports/exports over time or by country.

3.3 Individual database creation

This is a filing application for data that arrives in e-mail, by post, or is discovered on the Web [94]. The individual sets up some goal-oriented digital filing system and populates it with items that arrive at unpredictable times. The tables are processed either as they arrive, or batched for more convenient interactive processing. An important consideration here is minimization of the original set-up time and level of skill required.

3.4 Tabular browsing

Interactively extracting specific information from a large table is somewhat similar to addressing queries to a database with a language like SQL. Wang gives examples where the results of a query consist of highlighting specific cells in a table. She also mentions the possibility of creating subtables in response to a query, which is similar to view generation in a database [88].

3.5 Audio access to tables

In the EMU project [81], it may be desirable to detect and access newly received tables in e-mail by telephone. Access may take the form of an abbreviated reading or summarization of the table, a query–answer interface directly to the table, or conversion of the table to a database and access through an existing audio-database interface (if one were to exist). A protocol for direct access to tables was devised for "talking books" for the blind [76]. It requires repeating the appropriate table heading before each content cell is voiced, which can be a slow and painful process.

3.6 Table manipulation

Existing tables often need to be reformatted, combined, or modified for specific target audiences. Such manipulation may take place at the level of format, using a word processor, page-composition language, or spreadsheet, or at the deeper level of the underlying database. The latter can use independently-developed facilities for view generation and database output formatting. This application is mentioned in [48, 88].

3.7 Table modification for display

Retargeting Web page displays for small-screen devices like personal digital assistants (PDAs) and cell phones has assumed increased urgency and importance with the deployment of fast wireless connectivity. A recent review [2] lists four alternative techniques: hand recoding, trans-coding (automatic replacement of HTML tags by device- and target-specific tags), re-authoring based on automatic layout analysis, and re-authoring based on natural language processing (NLP). Re-authored pages can be presented hierarchically, with a root node consisting of a table-of-contents with links to detailed content. Although no table-specific techniques are given, some of the methods we describe are referenced.

Interestingly, many if not most Web pages are constructed with the HTML `<table>` construct (just as figures in Microsoft Word are often laid out using its table facility). The real problem with layout analysis on Web pages is that everything floats. The geometry is not fixed until the page is displayed by a particular browser, with specific settings and window size. Nevertheless, HTML preserves some relative ordering. This is exploited in [3] to re-author an HTML list. Further suggestions for generalizing the notions of precedence, proximity, prominence, and preference for interpreting content flow in HTML documents are presented in [2]. As seen in the following discussion, PDF documents share this problem of lack of association between content and form, therefore some of the same techniques may be useful for retargeting them to different formats.

In addition to accommodating small-format displays such as a PDA, one may wish to modify a page-width table to single-column width. Additional headers must be inserted to divide long tables to fit pages. A research issue here that may draw on database concepts is the division of one or more tables into a set of equivalent tables (cf. "Large Tables" in [88]).

We believe that the extraction of tables from HTML documents is evanescent compared to the conversion of paper documents because XML-based schemes are conceived with the goal of assuring machine interpretability [75].

3.8 Information extraction from tables

Information extraction from tables is perhaps analogous to the task of the same name applied to sentential text. The narrow definition requires a target schema and requires that arbitrary input (generally of some standard encoding) be transformed into instances of the schema. Examples of systems that fit this definition include [22, 28, 31, 60, 80]. Each of these systems works with varying definitions of tables and varying data formats.[3]

A wider definition without a fixed schema may be analogous to message understanding and represents perhaps the ultimate goal of table understanding.

It is perhaps too early to report any statistics representing the state of the art for this task. Part of the challenge is to provide a standard data set against which systems may be tested and results compared.

An interesting approach adopted by many industrial solutions uses the schema to drive the segmentation of the document and the recognition and interpretation of the tables. Knowing that one is looking for a financial table of a certain

[3] One challenge that document-based tasks face is the added burden of standardizing input into analytical components. One could argue that information extraction from sentential text must only standardize on the object language.

sort, and the form of likely labels and values, is invaluable knowledge even at the OCR and text blocking stage.

3.9 Ontology learning from tables

Ontology learning (e.g., [65]) has recently received considerable attention because of the emergence of the Semantic Web. The Semantic Web requires an abundance of ontologies, and creating them by hand is seen as a barrier preventing widespread use of the Semantic Web. In an attempt to break through this barrier, researchers have begun to build systems to "learn" ontologies from existing documents. Learning ontologies from sentential text, however, has proven to be difficult. Learning ontologies from tables may be more fruitful.

Thus, a relatively new application for table processing is the consolidation of information from multiple tables (usually downloaded from the Web) to generate domain-specific ontologies. The TANGO project [84] is an initial effort to use table analysis for generating ontologies. At least partially automating the preparation of such bodies of factual information may help pave the way towards a realization of the Semantic Web.

4 The commercial landscape

The majority of current table applications, as described in this paper, can be found in academic and other research contexts. However, like any advanced technology, a number of commercial systems are now available that either offer direct table-processing capabilities, or which rely to some extent on table understanding technology.

Low-end OCR systems, such as ScanSoft's OmniPage [79], provide table location and segmentation features. These are generally targeted at explicitly gridded tables (with some packages permitting user-guided analysis of non-gridded tables). Although the location of tables in such systems is generally adequate, market forces are such that the appearance of high-quality table segmentation features for arbitrary document input is unlikely.

Companies providing archival and document conversion services, such as XML Cities [96], recognize the importance of capturing table data—as well as the need to index this data appropriately. The work-flow around these services permits the creation of new matching rules, as well as the validation and correction of conversion by a human operator, thus providing the required quality level demanded by the customer.

In application services environments, where table understanding can be customized by domain to include constraints that enable high-quality results with almost complete automation, the medical insurance domain is perhaps one of the most successful. Insiders Information Management GmbH [49] and TCG Informatik AG [83], for example, both adopt this approach.

Information on the quality of commercial systems is not generally available. In the low-end OCR market, the input is so varied that claims—if available—would be hard to interpret. Where the work-flow involves a human, the quality is generally controlled according to individual customer needs through customization and/or validation processes.

A *form*, as opposed to a table, is a sheet of paper with labeled boxes used for information collection: the items specified by the labels are written or typed into the boxes, then the form is returned to the originator and the relevant information is extracted. Common examples of forms are tax returns and catalog order forms. The advent of graphic printers allowed printing forms on demand: forms intended for the same purpose became diversified. The rulings and boxes lost their importance. In document analysis, the distinction between forms, invoices, and business letters is fading.

Forms processing is now a major industry. Large applications, such as medical claims processing, state income tax, insurance, and retirement systems require conversion of several hundred thousand forms per day. In many such applications, most forms are filled out by hand. The similarities between table and form processing are emphasized in [87] and [13]. Other notable work on forms includes [5] and [69]. Continuing efforts to pass processing costs down to the end users will cause many of these mass form-processing applications to be migrated to the Web. Electronic forms are based on HTML, JAVA, Active-X, or XML.

Few forms processing systems used in production environments are described in the research literature. An exception is smartFIX, which evolved from 10 years of research at the German Artificial Intelligence Research Center (DFKI), and is now used by a dozen medical insurance companies to process tens of thousands of bills daily [54–56]. The system is able to classify about 60 types of documents (hospital bills, prescription drug bills, dentist's bills), and extracts over 100 different types of information from them (about 20 items per document on average). It relies on large databases of customers, products, and price schedules, and has elaborate models of the each customer's information flow, accuracy requirements, audit practices, training schedules, and distributed computational resources. Although constraint satisfaction methods are incorporated, every extracted field is subject to human verification. About 75% of the fields are labeled "safe," with less than 1 error per 1000. The major source of inadequately processed fields is OCR error. It is reported that the system saves 65–75% time over conventional manual data entry.

So far, there is no comparable table-processing industry, but some service bureaus do offer conversion of printed tables to electronic form.

5 Input media and formats

We consider tables that are presented in two different media: electronic and paper. We further subdivide the former based on encoding schemes. The net result is three (broad) classes of input tables:

1. ASCII file with only "pure" linguistic content and character-level spacing.
2. Page-descriptor file (Word, LaTeX, HTML, PostScript, PDF) with linguistic content, and refined formatting.
3. Bitmap file of an image of a table with white space around it.[4]

5.1 Tables presented in electronic format

Tables in plain text format may appear in e-mail or on certain kinds of Web pages. The structure of the table is represented only by ASCII symbols for space (blanks), tab characters, and carriage returns. Occasionally, printable ASCII symbols are used to show horizontal and vertical rules.

Electronic tables not intended for printing tend to be smaller and simpler than paper tables. The amount of detail that can be displayed on a typical monitor is less than one-tenth of what can be seen on a typeset page.

Mark-up languages like SGML, HTML, and XML have special conventions for tables, but there is no assurance that table tags are not abused or misused. Page composition languages have elaborate facilities for formatting tables, like TROFF Tbl [61] and the LaTeX table and array environments [59]. Many other table composition systems are surveyed in [88].

Microsoft Word has a table formatting subsystem and provides interconversion between tables in plain-text, Word-table, Rich Text Format (RTF), and Excel spreadsheets. FrameMaker offers PDF for posting tables on the Web in non-editable form, and XML for applications where the structure needs to be accessible. VXML is a proposed general-purpose format for audio access to Web documents.

Tables may also be reproduced in any raster image format, such as TIF or GIF, or rendered directly in PostScript [74]. Although directly-generated tables in image format may look superficially like scanned paper tables, they are not affected by noise or skew.

The Portable Document Format (PDF) is one of the most widely used formats for document representation. PDF files can be readily transformed to and from PostScript, and are relatively small due to embedded compression. PDF can be used for both computer-generated documents (conversion options are built into many word processing systems) and for scanned pixel maps in black-and-white, gray scale, or color. It also has facilities for searching, indexing, annotation and limited editing, but does not encode document structure below the page level: the file is simply a list of low-level objects like groups of characters, curves, and blobs, with associated style attributes like font, color, and shape. While there are several on-going research projects on recovering logical structure from PDF documents, we have found no research specifically on PDF table recognition.

PDF encodes a document as four types of graphics rendering instructions: (1) control instructions produce no output; (2) text instructions render glyphs of symbols; (3) graphics instructions render line art; (4) image instructions map bitmapped images [6]. It is therefore possible to apply directly the methods developed for hard-copy table recognition, but this requires error-prone image processing and OCR, the results of which are already explicitly and unambiguously provided in the PDF representation of computer-generated documents.

The AIDAS project converts industrial technical manuals into an indexed database of training material. The manuals are annotated according to a domain ontology. An important step is the extraction of logical structure from PDF files. This is accomplished by assigning logical functions (section header, text paragraph) to each layout object and refining the assignment as more evidence (bullets, boldface) becomes available. A shallow grammar is implemented for recognizing each function: tables are recognized as a proximate set of "floating" text [6]. The approach is based on the notion that layout objects do not explicitly represent logical structure but contain cues about their role in the structure [82].

The goal of a project at Hewlett-Packard Laboratories is to reuse the layouts of existing PDF documents as templates for creating new pages. This necessitates the identification of logical components and the extraction of the content of each component. The procedure first separates into text, image, and vector graphics layers. Compound objects are reduced to simple objects. Each component block is represented as a polygonal outline, a set of style attributes, and content. Text word, line, and segment (paragraph block) analysis is performed on the text layer, taking into account style attributes such as type size and italics. The contents are transformed to XML format. Bitmap analysis of the graphics layer was, perhaps surprisingly, found easier than performing segmentation following drawing paths. The segmented graphic objects are eventually converted to SVG format. Based on the analysis of the 18 page-segmentation errors that arose in processing 200 test pages, the development of specific table and map recognition modules is suggested to reduce errors further. It is clear that the combination of the current text layer and vector graphics layer analysis provides the necessary foundations [16, 17]. Among references that address electronic tables are [26, 48, 71, 73].

5.2 Tables presented on paper

Paper tables are usually typeset, typewritten, or computer-generated. In principle, they can also be hand-printed or drafted (like telephone-company drawings [7–11, 15, 19], and the header-blocks of old engineering drawings), but we deem such hand-drawn tables as more akin to forms and exclude them from consideration here.

Paper tables are converted to digital form by optical scanning. Printed tables are typically scanned at sampling rates of 200–600 dpi, but for some applications facsimile

[4] We assume that dynamic binarization, deskewing, and noise removal have already been accomplished by standard image processing methods, and also that a black-box OCR system—for print, handprint, or handwriting as required—is available.

scans (100×200 dpi) may be important. High-speed duplex scanners have a throughput of 100 pages per minute at 300 dpi and 24-bit color depth. Bilevel scanners, which are suitable for most tables, are even faster.

Copying and scanning may introduce noise and skew. Both of these are more effectively corrected on a gray-level representation of the page. Image-reparation software is available from many vendors, including Lead Technologies, Mitek, Visual Image, Cardiff, and Captiva. The majority of the published work on table processing deals with the extraction of structure from scanned paper tables [1, 10, 15, 32, 39, 50, 60, 87, 93, 98].

5.3 Table detection

Conceptually, table processing can be broken into two logical steps: table detection and table recognition. Much existing work on tables described in the literature addresses the latter step and assumes that the table has already been identified and segmented out from the input (or that identifying the table is trivial—e.g., the whole document is the table). While this is, in fact, the focus of our survey, we digress briefly to consider the table detection problem.

Most prior research on the problem of table detection has concentrated on detecting tables in scanned images, and the vast majority depends on the presence of at least some ruling lines (e.g., [60]). Hirayama [39] uses ruling lines as initial evidence of a table or figure and then further refines this decision to distinguish tables from figures by a measure based on such features as the presence of characters. There is, of course, no guarantee that such lines will be present in printed tables. Notable exceptions to this assumption include work by Rahgozar and Cooperman [74], where a system based on graph-rewriting is described and work done by Shamalian et al. [80] in which a system based on predefined layout structures in given.

There is much less prior art in the case of symbolic tables, though they are becoming increasingly important. As noted earlier, these may originate either in ASCII form (e.g., as part of an e-mail message), or as the result of saving a "richer" document (e.g., an HTML page) in "text-only" format. They may also be encoded in a page-descriptor language such as PDF or PostScript, or in an electronic file format such as the one used by Microsoft Word. More often than not, ASCII tables contain no ruling lines whatsoever, depending only on the 2-D layout of the cell contents to convey the table's structure. Little of the past research on printed tables is applicable in such cases.

Hu et al. [42] describe a technique for detecting tables that does not rely on ruling lines and has the desirable property that an identical high-level approach can be applied to tables expressed as ASCII text (or any other symbolic format) and those in image form. This general framework is based on computing an optimal partitioning of a page column into some number of tables. A dynamic programming algorithm is presented to solve the resulting optimization problem.

Three different heuristics to enable a production system to detect tables in incoming documents are discussed by Klein et al. [57]. The first, based on searching OCR results for predefined table headers, was found to be too susceptible to a variety of real-world complications and hence unacceptable from a user standpoint. More sophisticated techniques based on detecting column structure and inter-textline similarities proved to be more robust.

Lastly, in a recent paper, Pinto et al. [72] describe an approach for locating and extracting tables based on conditional random fields. Applied to plain-text government statistical reports, they report a detection accuracy of 92%.

5.4 Simplifying assumptions

We note that in order to focus on a core set of issues, we have been forced to omit numerous important problems relating to the processing of tables, including plausible sources of tables, table similarity detection, and human–machine interfaces (graphical and spoken) to tabular data. For these, we refer the reader to the previously mentioned surveys [62, 63, 97].

For the purposes of the present study, we exclude from consideration the following concerns.

1. Information external to the table proper, including titles and captions; footnotes; relevant passages from nearby narrative text; information from related tables; and domain-specific table conventions.
2. Tables outside our restrictive definition, including folded and nested tables; tables with spanning cells in the table body; tables with both horizontal and vertical text; tables with domain-specific symbols, foreign script, or out-of lexicon text; tables containing graphics; skewed tables; and sparse tables.
3. Expandable (clickable) Web tables and Web tables employing hypertext links (embedded URLs).
4. Multidimensional data arrays ($D > 2$).
5. Tables that may or may not be tables, including matrices; tables used for formatting text, equations, or graphics; tables of contents; and artistic, multi-color, and sloppy tables.

6 Table-processing paradigms

Tables may be encoded in many different input formats. However, in this section we take the view that a table is a table if and only if it appears as such when presented in its intended visual form to the end user. Hence, the concept of a 2-D rendering is central to our discussion of table-processing paradigms.

On the one hand, renderings of tables encoded as ASCII text are so self-evident that it is easy to forget that they are still based on a set of underlying assumptions (e.g., what is connoted by a carriage return and, in most cases, that the rendering will use a monospaced font). While other encoding schemes such as PostScript and HTML have the potential to be much more complex, the simple fact is that such documents are rendered all the time, and developing systems to perform this function is not considered a particularly daunting task. The former are known as PostScript interpreters (e.g., Ghostscript), while the latter are referred to as Web browsers (e.g., Mozilla).

Although we shall at times strive for maximum possible generality, from a pragmatic standpoint, the vast majority of table-processing research to date has focused on two specific classes of inputs. Tables encoded in ASCII format are a canonical instance of rendering on a coarse (i.e., character-level) grid, while scanned bitmap tables are a canonical instance of rendering on a fine (i.e., pixel-level) grid. Hence, these are the concrete examples we turn to most frequently in the following exposition.

The first group of paradigms associates cell content with row and column numbers. Logically, their output is a list: `(i,j) cell-content`, etc. Top–down methods recover the underlying grid structure, then find the content of each cell. Bottom–up methods first delimit cell contents, then construct the grid.

The input to the second group of paradigms is the above list. These paradigms associate cell contents with row and/or column headers. If row and column headers are absent, virtual headers are assigned. This requires some renumbering. The most complex algorithms target nested headers.

The third paradigm level extracts high-level (semantic) information from the output of the earlier paradigms, i.e., row and column numbered headers and cell contents. Its output is suitable for downstream applications like SQL, PROLOG, XML or other logic-based schemata. While this is of increasing interest, especially arising out of the Semantic Web, there has been more work on the earlier aspects.

6.1 Simple tables

In the simplest case, it is possible to determine the cell structure of the table using purely geometric cues from the 2-D rendering. If it is known that the maximum intra-cell horizontal spacing is strictly less than the minimum inter-column horizontal spacing, and that the maximum intra-cell vertical spacing is strictly less than the minimum inter-row vertical spacing, the table can be parsed into columns and rows by using these parameters to determine whether a given "gap" represents a continuation of the current cell or the start of a new cell.

Note that this paradigm can be implemented independently of the input format of the table because it is defined in terms of the intended 2-D rendering of the tabular information. All we need is an understanding of the way the file is to be rendered, a way to identify the basic "unit" in the input under study (i.e., character strings in the case of ASCII and connected components in the case of bitmaps), and a way to measure distances between these basic units.

This paradigm is too simple by itself to suffice for many tables, but the notion of thresholds that allow merging intra-cell constituents without merging the contents of separate cells is subsumed in many of the paradigms mentioned later.

In the ASCII domain, this corresponds to character strings delimited by column separators (e.g., consecutive spaces) and row separators (e.g., carriage returns). For example, we might have the following input, where spaces are indicated by a dash symbol, -, and carriage returns are indicated by a new paragraph symbol, ¶:

```
LASTNAME----INITIAL----BIRTHDATE¶Smith--
-----J----------12/3/1988¶Barr--------K-
---------25/5/1975¶
```

which would be rendered (on a coarse grid) as:

```
LASTNAME    INITIAL    BIRTHDATE
Smith       J          12/3/1988
Barr        K          25/5/1975
```

On the other hand, in PostScript a similar table would be represented, thus:

```
/Courier-New findfont 8 scalefont setfont
0 100 moveto (LASTNAME) show
60 100 moveto (INITIAL) show
120 100 moveto (BIRTHDATE) show
0 90 moveto (Smith) show
60 90 moveto (J) show
120 90 moveto(12/3/1988) show
0 80 moveto (Barr) show
60 80 moveto (K) show
120 80 moveto (25/5/1975) show
showpage
```

with a rendering (on a fine grid) like this:

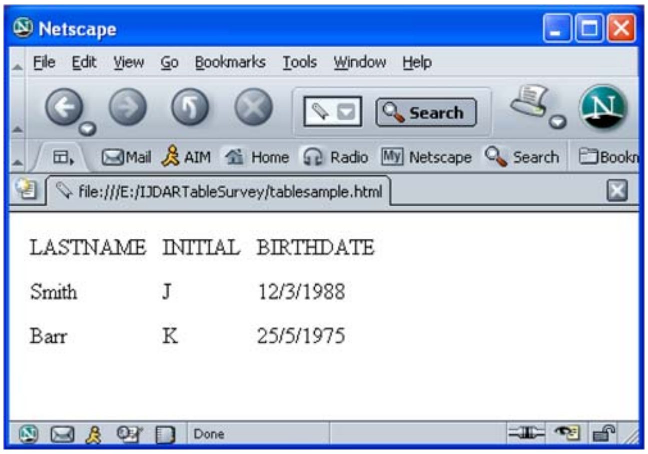
```
LASTNAME  INITIAL BIRTHDATE
Smith     J       12/3/1988
Barr      K       25/5/1975
```

Lastly, in HTML, for an input like this:

```
<html><body><table cellpadding="5">
<tr><td>LASTNAME</td>
<td>INITIAL</td>
<td>BIRTHDATE</td></tr>
<tr><td>Smith</td>
<td>J</td>
<td>12/3/1988</td></tr>
<tr><td>Barr</td>
<td>K</td>
<td>25/5/1975</td></tr>
</table></body></html>
```

the rendering (on a fine grid) might appear as in Fig. 4.

Independent of how the table is stored or rendered, we can capture the table in an ontology that formally describes the observable input. Figure 5 shows an ontology describing the observable input for the table described by the ASCII


Netscape
File Edit View Go Bookmarks Tools Window Help
Search
Mail AIM Home Radio Netscape Search Bookm
file:///E:/IJDARTableSurvey/tablesample.html
Done

LASTNAME	INITIAL	BIRTHDATE
Smith	J	12/3/1988
Barr	K	25/5/1975

Fig. 4 Screen snapshot of the rendering of an HTML table

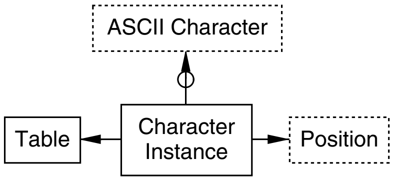


Fig. 5 Ontology of observable features of an input ASCII table

character sequence above. In Fig. 5 the *ASCII Character* object set consists of all ASCII characters. The *Character Instance* object set consists of object identifiers, one for each instance of an ASCII character in a table at a particular position. Thus, we can capture each ASCII character (including each space and carriage-return character) and its position in the sequence of ASCII characters representing the table.[5]

Although our example here is particularly simple, we observe that it is possible to model ontologically all observable input features. Different features would be captured for alternative input media. For PostScript, for example, the input ontology would capture bounding boxes for strings along with string content, and for HTML, the input ontology would capture the table-row and table-data structure as provided by the $\langle tr \rangle$ and $\langle td \rangle$ tags. For tables whose input media is an image, we can model the input down to the pixel level if we wish. Our ASCII example here only indicates the possibilities, and we will not attempt in this paper to produce a full ontology of all features of interest. Our desire here is only to indicate that it is possible to create such ontological models as suggested in both [36] and [4].

In any case, independent of the form of the input (whether it is a simple string of ASCII characters, a PostScript file, an HTML file, or an ontologically described sequence of ASCII character instances), the goal is to obtain the following kind of output:

[5] The 'o' at the base of the arrowhead connected to *ASCII Character* in Fig. 5 denotes "optional participation." The meaning here is that although *ASCII Character* contains the full set of ASCII characters, some of the characters may not appear in the tables under consideration and thus their participation in the relationship set between *Character Instance* and *ASCII Character* is optional.

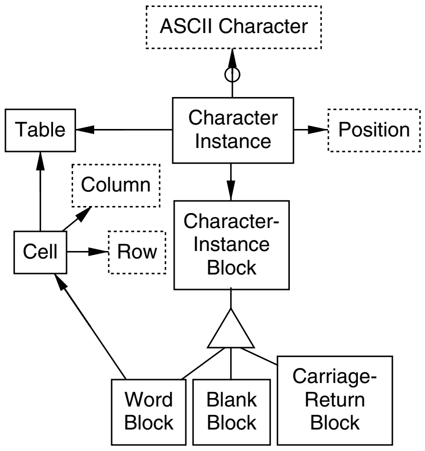


Fig. 6 Ontology with *Row* and *Column* of cells derived

```
<cell row="1" col="1">LASTNAME</cell>
<cell row="1" col="2">INITIAL</cell>
<cell row="1" col="3">BIRTHDATE</cell>
<cell row="2" col="1">Smith</cell>
<cell row="2" col="2">J</cell>
<cell row="2" col="3">12/3/1988</cell>
<cell row="3" col="1">Barr</cell>
<cell row="3" col="2">K</cell>
<cell row="3" col="3">25/5/1975</cell>
```

which is a logical representation of the cell structure of the table, with row and column indices assigned to each cell.

Considering table processing from an ontological point of view, we can see this paradigm as generating object and relationship sets in an intermediate, derived ontology that includes a derived object set *Cell* with associated *Row* and *Column* object sets. Figure 6 shows an ontology with these derived object sets with their derived relationships among each other and their relationship to *Table*. As part of the derivation, the paradigm would have recognized character-instance blocks of words, spaces, and carriage returns. An intermediate ontology can represent these derived object sets as they are created as Fig. 6 shows.

The paradigm for processing such tables, which corresponds to transforming a list to an array, is as follows:

1. Render table on logical 2-D grid.
2. Parse 2-D representation. If necessary (i.e., if table is in bitmap format), OCR cell contents. Call consecutive character strings cell "word blocks" or "phrase blocks" or "cell contents."
3. Advance column count for each column separator.
4. Advance row count for each row separator and reset column count to 1.

In [75], table structure is viewed as a perfectly regular isothetic tessellation of a rectangular region into virtual cells, and a superimposed partitions of the virtual cells with which cell content is associated. The authors propose to link together cells in the same row or column with a text-block-adjacency graph reminiscent of DocStrum [70].

A proposal to combine page analysis and table structure analysis by seeking regions with horizontally and vertically aligned word bounding boxes is advanced in [89]. No experimental results are presented because the test data had not yet been collected when the paper was written.

6.2 Compound tables with blank lines

In this case, the input is character strings delimited by column-separators and row-separators. Each cell may have multiple components on the same or different logical text lines. For example, in ASCII this might be:

```
LAST----INITIAL----BIRTH¶NAME-----------
----DATE¶¶Smith---J----------12/3-1988¶¶
Barr----K----------25/5-1975¶
```

which would be rendered (on a coarse grid) as:

```
LAST     INITIAL     BIRTH
NAME                 DATE

Smith    J           12/3 1988

Barr     K           25/5 1975
```

The paradigm in this case, which is top–down, is as follows:

1. Render table on logical 2-D grid.
2. Parse 2-D representation. If necessary (i.e., if table is in bitmap format), OCR cell contents. Project characters horizontally and vertically.
3. Make horizontal cuts at the end of groups of blank text lines and vertical cuts at the end of space sequences.
4. Then consider any text within a cell delimited by horizontal and vertical cuts as a phrase block.
5. Assign row and column numbers.

Among the earliest researchers to tackle table recognition were Laurentini and Vida [60]. Their objective was to transform tables found on scanned pages into electronic form, rather than extract the table structure for further analysis. They find rulings by run-length analysis, and then check if tight groups of character-sized connected components fall within the resulting cells. Groups with large gaps in their projection profiles are subdivided by invisible virtual rulings.

A method based on a similar view of tables, using projection profiles and aligned spaces between word bounding boxes, was applied to convert Japanese tables into HTML [86]. This paper has a very concise review of previous work, but only examples and no statistical results.

Another early paper by Chandran and Kasturi recognizes the lines in partially ruled tables as successions of adjacent black runs. Missing demarcations are found by an analysis of white streams. After a set of horizontal and vertical demarcations are obtained, individual blocks are labeled as heading, subheading, or entry. A block is labeled as a heading if it has more than one child, and as a subheading if it is the first block with a single child, followed by a similar single-child pattern. Only column headings are considered.

Abu Tarif finds and vectorizes any rulings, and adds "virtual lines" that separate aligned components of text. He converts the resulting "table skeleton" first into an X–Y tree [68] and to Microsoft Excel spreadsheets using Excel macros [1]. He did not OCR the text itself.

Cesarini et al. search Modified X–Y Tree descriptors of documents to find tables consisting of clusters of horizontal and vertical cuts [14]. The algorithm has five thresholds, which are optimized to obtain the maximum value of a "Table Location Index" on training documents. The method locates correctly over 58 of 75 tables in almost noise-free IEEE-PAMI pages, and 22 of 58 tables in U. Washington test images. They report that this performance is far superior to that of two leading commercial OCR systems that also find tables.

John Handley uses both rulings and word bounding boxes to separate the cells and construct cell separators for the table frame. His method handles large, complex, fully-lined, semi-lined, and line-less cell tables with multiple lines of symbols per cell by iteratively identifying all cell separators and cells. Although spanning header cells are found, their relationship to the leaf cells is not determined [38].

Hirayama proposes a sophisticated algorithm for segmenting a partially-ruled table into a lattice composed of a grid of rectangles [39]. Lines are grouped when they intersect, are close and nearly parallel, or if their endpoints are close. Rulings are extended by virtual lines to the outermost ruling. Eventually, rectangles separated only by virtual lines are joined. The resulting polygons form cells only if they are rectangular, contain only character strings, or are empty. Alignment is performed left-to-right with a string-correction algorithm (the DP—dynamic programming—that appears in the title of the paper) where the weights for substitution are the differences between the baselines of two text strings. This method can find the cell structure even when the cell contents are of unequal size or when there are many empty cells.

6.3 Compound tables without blank lines

Here we have no guarantee that there is vertical separation between logically distinct rows in the table. For example, input in the ASCII case might consist of character strings delimited by column separators and row separators. Each cell may have multiple word blocks on the same or different text lines. The contents of cells overlap both horizontally and vertically.

```
LAST--INITIAL-BIRTH¶NAME----------DATE¶S
mith--J-------12/3-1988¶Barr--K-------25
/5-1975¶
```

which would be rendered (on a coarse grid) as:

```
LAST   INITIAL BIRTH
NAME           DATE
```

```
Smith J        12/3 1988
Barr  K        25/5 1975
```

Unlike Paradigm 2, there are no blank lines, so we cannot find the horizontal separators directly. The paradigm here combines top–down and bottom–up processing:

1. Render table on logical 2-D grid.
2. Parse 2-D representation. If necessary (i.e., if table is in bitmap format), OCR cell contents. Work bottom–up by grouping horizontally and vertically adjacent word blocks by some linguistic association measure (cohesion).
   (a) Association may be based on language models, typeface and size, indentation.
   (b) Language model includes conventions for dates, currencies and prices, telephone numbers, units, etc.

   Here "LAST" and "NAME" go together, as do "BIRTH" and "DATE," and perhaps "12/3" and "1988."
3. Group word blocks that have high association (or cohesion) into phrase blocks.
4. Find and use bounding box of phrase blocks (cell overlap) to construct grid. (Graph representation may be appropriate.)
5. Finally, assign row and column numbers to phrase blocks.

A notation and an inference algorithm to identify conceptual cell commonalities, such as "amount fields" and "dates," was developed by Bayer [12]. While the work does not specifically address tables, it offers a toolbox of syntactic, lexical, and geometrical properties in a manner suitable for a table ontology.

In the ASCII domain, Pyreddy and Croft report on a table extraction and retrieval experiment involving 6,509 tables from a corpus consisting of 6 years of text from the Wall Street Journal [73]. This data, professionally written and from a single source, is likely to be unrealistically uniform, however. Pyreddy and Croft have elaborate heuristics to separate leaf cells from other table content but do not differentiate between table captions and headings because they are used in a similar way in their information retrieval system.

Peterman et al. consider a table a collection of five types of cells: *data*, *vertical indices*, *horizontal indices*, *title*, and *footnotes*. They present a detailed analysis of "table topology," i.e., the conventions governing the layout of cells, and of the placement of data within the cells. The contents of each cell are analyzed by string matching to discover cells with similar letter syntax. The resulting rules for determining the "reading order" of the table are embodied in a PERL script. They present experimental results on a heterogeneous corpus of 100 electronic tables that they suggest mimic the results of processing typeset paper tables with 99% accurate OCR. It is clear that even aside from possible OCR and image processing errors, manual editing would be required for most applications [71].

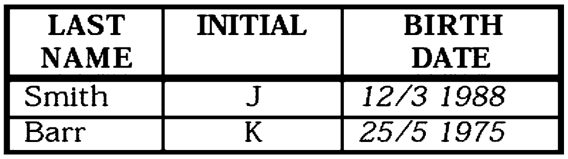
Building on extensive previous work, Rus and Subramanian [77] offer an interactive method of building models consisting of modular interactive agents for information access and capture in distributed databases. They give examples of structure detectors and segmentation modules for both paper and electronic tables. These modules subdivide documents according to prevalent white spaces and match table rows by syntactic string matching. In an interesting digression, they predict the probability of incidental white streams from word length statistics.

In a series of papers [41, 42, 44], Hu et al. describe a medium-independent approach to table detection and structure recognition based on a dynamic programming algorithm that computes the optimal partitioning of the input into some number of tables, uses hierarchical clustering to determine the column structure, and then applies heuristics to determine table headers and row segmentation. They also present evaluation measures for quantifying the performance of such algorithms [45]. One targeted application is automatically reformulating tables found in email for user access over the telephone [43].

6.4 Tables with rules

Previously, we considered the table cell contents to be delimited by white space. Now we turn to the scenario where cells are delimited by ruling lines. Such situations are more likely to arise in the case of scanned tables, so our examples will now refer to that mode of input.

The input, for example, might be a 300 dpi scanned bitmap of a ruled table:



LAST NAME	INITIAL	BIRTH DATE
Smith	J	*12/3 1988*
Barr	K	*25/5 1975*

The paradigm in this case is:

1. Process image to find and assemble line segments to determine frame of this table.
2. If necessary (i.e., if table is in bitmap format), OCR cell contents.
3. Use frame for row and column numbering.

Order of line finding and OCR, if it is necessary, may be interchanged.

Image processing techniques for the extraction of line segments include the Hough Transform [87], thinning, vectorization [1] and projection profiles [50]. Turolla et al. succeed in detecting 95% of 11,513 lines in 114 tables. They located cell entries of fully boxed tables by finding the minimal cycle of the graph corresponding to the frame. The lines are found using the Hough Transform. The system was developed primarily for French tax forms.

Itonori [50] combines text-block information with ruled lines. He expands the text bounding boxes until they meet either rulings or other text. Then he aligns cell boundaries in partially ruled tables with projection profiles. The method is applied to tables scanned at 400 dpi. The method attempts to extract spanning vertical and horizontal header cells, but sometimes fails on multiline header cells because of inaccurate local text-block extraction.

Box-driven reasoning is proposed in [40] to mitigate content-separator overlaps. Instead of seeking the intersection of horizontal and vertical lines, inner (white) and outer (black) bounding boxes constitute the lowest-level structure analyzed. The proposed underlying model is described only as follows: "A table-form document is a type of form composed of strings and cells made from vertical and horizontal lines." The system was tried only on 10 fairly complex forms, and only the timing results are given in detail.

The primary goal of this research was the recovery of cell structure from fully lined but highly degraded tables with broken rulings and overlapping cell contents [40]. The main contributions are the use of fine and coarse scans, and a separate set of bounding boxes in both for white spaces and for foreground connected components. The boxes in the coarse and fine images are reconciled according to the expected grid layout, and converted into a cell structure that corresponds to an idealized version of the scanned table.

T-Recs (Table REcognition System), an elaborate program for the structural analysis of ASCII tables based on bottom–up clustering of words, is described in [53]. The method works on both electronic and paper tables, starting with word bounding boxes. It can handle very narrow gaps, misaligned cells, and cells that span more than one printed line. It ignores ruling lines completely because it was designed for blocked text structures not only regular tables. A more flexible approach, T-Recs++, that can detect and analyze less regular tables as well as business letters, was subsequently reported [52].

6.5 Tables with simple headers

The input to this stage of processing is the output from the previous paradigms, i.e., phrase blocks with row and column numbers. For the example we have been using thus far, the output from the previous stage might be:

```
<cell row="1" col="1">LAST NAME</cell>
<cell row="1" col="2">INITIAL</cell>
<cell row="1" col="3">BIRTH DATE</cell>
<cell row="2" col="1">Smith</cell>
<cell row="2" col="2">J</cell>
<cell row="2" col="3">12/3 1988</cell>
<cell row="3" col="1">Barr</cell>
<cell row="3" col="2">K</cell>
<cell row="3" col="3">25/5 1975</cell>
```

The output from this stage should be:

```
<colhead col="1">LAST NAME</colhead>
<colhead col="2">INITIAL</colhead>
<colhead col="3">BIRTH DATE</colhead>
<cell row="1" col="1">Smith</cell>
<cell row="1" col="2">J</cell>
<cell row="1" col="3">12/3 1988</cell>
```

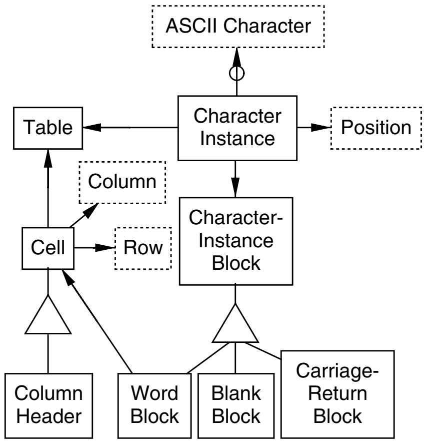


Fig. 7 Ontology with *Column Header* derived

```
<cell row="2" col="1">Barr</cell>
<cell row="2" col="2">K</cell>
<cell row="2" col="3">25/5 1975</cell>
```

The paradigm (recover header–cell relations) is as follows:

1. Determine whether rows or columns or both are homogeneous (using language model, typeface/size, spacing).
2. If a row or column is homogeneous, it may have a header.
   (a) Determine if top/leftmost cell is distinguished from others (using language model, typeface/size, spacing, rule if present).
   (b) If yes, call this top/leftmost phrase block vertical/horizontal header.
   (c) If not, add virtual/row column header, and assign unique constants as headers, and renumber.

Seen ontologically, we can consider the input for this paradigm as the derived ontology in Fig. 6. Using algorithms to derive headers, produces the ontology in Fig. 7, where the only change to the diagram in Fig. 6 is the object set *Column Header*, which marks certain cells as header cells. From this information it should be clear that we can derive the label–value pairs needed for the output ontology in Fig. 1.

Rahgozar applies rewriting rules in a graph language to parse a table. The sequence of productions reproduces the table structure of rows and columns [74].

6.6 Tables with nested headers

The input in this case is the intermediate output of Paradigm 6.3: phrase blocks and phrase block bounding boxes that do not constitute a uniform grid. For example:

```
        EMPLOYEE        PENSION STATUS
LAST NAME   INITIAL     BIRTH DATE
Smith       J           12/3 1988
Barr        K           25/5 1975
```

The output in this case should be:

```
<colhead col="1">EMPLOYEE LAST NAME
  </colhead>
<colhead col="2">EMPLOYEE INITIAL
  </colhead>
<colhead col="3">PENSION STATUS BIRTH
   DATE </colhead>
<cell row="1" col="1">Smith</cell>
<cell row="1" col="2">J</cell>
<cell row="1" col="3">12/3 1988</cell>
<cell row="2" col="1">Barr</cell>
<cell row="2" col="2">K</cell>
<cell row="2" col="3">25/5 1975</cell>}
```

The paradigm is as follows:

1. Determine top/leftmost spanning cells (using language model, typeface/size, spacing).
2. Create virtual cells by subdividing spanning cells.
3. Determine whether elements of next row or column distinguished from rest of row/column.
4. If yes, distribute (inverse of "factoring") contents of spanning cells over next row/column. Rename spanning headers accordingly.
5. If no, create virtual row/column headers with unique constant names and rename spanning headers.

This paradigm may become immensely complex with multiply nested row and column headers. The paradigm may also include analysis of stub (top-left cell), which is often the header for the row headers.

Formal paradigms for describing the structure of tables are the Table Syntax [32, 58], the Structure Description Tree [93], and the Cohesion Domain Template [48]. All three model only local horizontal and vertical adjacency relationships between cells. They aim at finding an appropriate tiling of the table. The foundations for a more sophisticated scheme are laid in [46].

Known (model-based) domain dependency relationships between cells can be exploited for validating an interpretation. Some examples are given in [93].

In a series of papers [31, 32, 33], Green and Krishnamoorthy apply a compiler design approach to parsing scanned ruled tables. The analysis consists of lexical, syntactic, and semantic steps starting at the pixel level and ending up with an EXCEL-like cell enumeration scheme suitable for multiple levels of spanning headers. Although the method is quite general, a model must be defined for every new family of tables.

Toyohide Watanabe and coworkers [64, 92, 91, 90, 93] aim at a complete description of the various types of information necessary to interpret a ruled scanned table. A training set of diverse tables is used to populate a classification tree, and each node of the classification tree contains information, in the form of a Structure Description Tree (SDT), to interpret a specific family of tables. In the operational phase, unrecognized documents are added to the classification tree, and a new STD is created for them.

The SDT represents generalized composition rules for horizontally and vertically repeated structures. It is both a logical layout representation and a syntactic description. Single and multiple horizontal and vertical location dependence relations are defined. These relations allow the analysis of rectangular substructures (called "structure fragments") of cells with spanning cells (usually headers) to the left or above related to content cells below and to the right. The semantic properties of individual table entries (city, zip-code) are expressed as item frames. Item fields may be name fields or data fields. The authors view the SDT as 2-D information, item sequence rules as 1-D, and a pattern dictionary as 0-D.

The image-processing components (for scanned tables) include extraction of horizontal and vertical line segments and corners. Image processing errors may be recovered in the course of subsequent analysis. The final output, aside from the meta-information used to process the tables, is the grid outlay and a set of interpreted name and data fields. Recognition of the table type assumes that the relationship between these fields is already known, hence high-level interpretation is moot.

Konstantin Zuyev [98] converts scanned ruled tables into a grid structure by finding horizontal and vertical "split points" using connected components, projection profiles, and gap thresholds. The method was developed for a multilingual FineReader OCR product. He also suggests using a high-level declarative definition of possible table layouts in the form of a grammar, with the extracted table grid and its cells as the terminal symbols. Heuristics are provided for common layouts of simple and compound (header) cells, where allowable layouts are specified by a "style" variable. The examples in the paper show successful segmentation of quite complex and dense tables.

6.7 Nested tables with row and column headers

Table 4 shows a table with nested headers for both rows and columns. The spanning header for the rows, which is *Char.* in Table 4, is in the stub of the table.[6] In general, a table with row headers may have none, one, or several spanning headers. When there are several, they typically appear to the left of the row labels in spanning boxes.

Since Table 4 is a table with rules, the input for this example is the output of Paradigm 6.4. Since there is a row spanning header, we should distribute it over the rows in the same way we distribute column spanning headers over columns as explained in Paradigm 6.6. The output in this case should be:

```
<rowhead row="1">Char. A</rowhead>
<rowhead row="2">Char. B</rowhead>
...
<colhead col="1">Binary Zone</colhead>
<colhead col="2">Binary Numeric</colhead>
```

[6] Note that it is sometimes ambiguous even to a trained human eye whether the stub is a spanning header for a column of row labels or a simple header for a column of values. Depending on the objectives for table processing, this may or may not matter.

Table 4 ASCII Code for capital letters

	Binary		
Character	Zone	Numeric	Hex
A	1010	0001	A1
B		0010	A2
C		0011	A3
D		0100	A4
E		0101	A5
F		0110	A6
G		0111	A7
H		1000	A8
I		1001	A9
J		1010	AA
K		1011	AB
L		1100	AC
M		1101	AD
N		1110	AE
O		1111	AF
P	1011	0000	B0
Q		0001	B1
R		0010	B2
S		0011	B3
T		0100	B4
U		0101	B5
V		0110	B6
W		0111	B7
X		1000	B8
Y		1001	B9
Z		1010	BA

```
<colhead col="3">Hex</colhead>
<cell row="1" col="1">1010</cell>
<cell row="1" col="2">0001</cell>
<cell row="1" col="3">A1</cell>
<cell row="2" col="1"></cell>
<cell row="2" col="2">0010</cell>
<cell row="2" col="3">A2</cell>
...
```

The paradigm is a combination of Paradigm 6.6 along with a similar paradigm that produces row headers.

6.8 *N*-dimensional tables

Table 4 is a 2-D table; Table 3 is a 3-D table. In principle, we can have any number of dimensions, although higher dimensions are not typical because their layout becomes increasingly complex. Higher dimension tables, for example, may be recursively nested, broken into labeled groups of tables, or successively linked through hypertext in cells of HTML tables.

Paradigms to recognize and process high-dimensional tables generalize Paradigm 6.7, which in turn generalizes Paradigm 6.6. To generalize the output, we can use dimensions rather than rows and columns. The output for Table 3, for example, would be:

```
<header dimension="1", indexNr="1">
   Nucleotide First Position U</header>
<header dimension="1", indexNr="2">
   Nucleotide First Position C</header>
<header dimension="1", indexNr="3">
   Nucleotide First Position A</header>
<header dimension="1", indexNr="4">
   Nucleotide First Position G</header>
<header dimension="2", indexNr="1">
   Nucleotide Second Position U</header>
...
<header dimension="3", indexNr="1">
   Nucleotide Third Position U</header>

...
<cell
   dimension="1" indexNr="1"
      dimension="2" indexNr="1"
         dimension="3" indexNr="1">Phe</cell>
<cell
   dimension="1" indexNr="1"
      dimension="2" indexNr="1"
         dimension="3" indexNr="2">Phe</cell>
...
<cell
   dimension="1" indexNr="4"
      dimension="2" indexNr="4"
         dimension="3" indexNr="4">Gly</cell>
```

We are not aware of any work in this area and leave it as a future challenge for the community.

7 Conclusions

We have identified a number of potential applications for table processing and the corresponding research problems for which little work has been reported thus far. We have also expressed our opinions of the relative difficulties of the tasks involved. To recapitulate, the applications are as follows:

1. Large-volume, homogeneous table conversion.
2. Large-volume, mixed table conversion.
3. Individual database creation.
4. Tabular browsing.
5. Audio access to tables.
6. Table manipulation.
7. Table modification for display.

An obvious next step would be to analyze these applications further to determine their commonalities and differences.

The new research problems appear to us to be as follows:

1. Query mechanisms for freeform electronic tables.
2. Audio navigation and access to a gridded table.
3. Subdividing a table into a set of equivalent tables.
4. Spotting tables in electronic mail.
5. Clustering tables into similarity groups.
6. Converting a paper or electronic table into an abstract representation.
7. Effects of "noise" in tables and correction of errors introduced in processing.

Table 5 Interrelationships between applications and research problems in table processing

	Query mechanisms	Audio navigation	Table subdivision	Table spotting	Table clustering	Conversion to abstract form	Overcoming recognition errors	Performance evaluation
Large-volume, homogeneous conversion						●	●	●
Large-volume, mixed conversion				●	●	●	●	●
Individual database creation	●		●	●	●	●	●	●
Tabular browsing	●	●	●	●	●	●	●	●
Audio access to tables	●	●	●	●	●	●	●	●
Table manipulation			●		●	●	●	●
Table modification for display			●		●	●	●	●

8. Performance evaluation of both table conversion and table query.

The ways in which the applications and problems interrelate are depicted in Table 5. Unless we make headway on performance evaluation, including acquisition of statistically adequate test material, it will be difficult to evaluate progress on any of the other tasks.

Most work to date is based on table geometry, i.e., processing the graphic elements of the table. Very little has been reported on combining such image processing with the results of character recognition of the cell contents. Although the logical interpretation of paper and electronic tables is similar, the overhead of image processing and OCR makes the former a much more difficult task. Current OCR systems often de-columnize tables because superficially they look like multicolumn text. No test on a large, heterogeneous corpus has been reported, and few researchers have considered the need to provide a mechanism for the correction of residual errors from automated processing.

More recently, the trend has shifted to the apparently easier problem of electronic table conversion. Several commercial organizations advertise their capability of converting electronic tables to various forms, including spreadsheets. Some advertise conversion of tables presented in raster image form.

Simple electronic tables, whether ASCII, PDF, RTF, SGML, HTML, XML, LaTeX, Tbl, or other, can probably be converted with moderate effort to an abstract form with over 90% accuracy. Spotting large tables in electronic documents is relatively easy, but delineating them precisely is more difficult. A limit on achievable accuracy is imposed by the ambiguity inherent in these tasks.

We have offered a formal definition of table understanding in terms of relational tables and table ontologies. The derivation of information from a table could be accomplished by converting the table to a relational database or equivalent and formulating queries in SQL. Alternatively, queries can be answered by direct interactive access to a preprocessed table. Such preprocessing need not be much more elaborate than division into rows and columns.

However, tables do not generally contain sufficient information for conversion into a database, although they can be converted into an abstract table or spreadsheet. To add the necessary semantics, a model of the table is required. The model can be derived from an existing database corresponding to similar tables, or it can be provided by the user/operator. The user can either provide the model explicitly, or implicitly by correcting errors. Except for large volumes of similar tables, it appears sensible to take advantage of the user's understanding of the context of the table; endowing a table-understanding system with such context is difficult.

The economics of table processing is another important point that has often been ignored. Clearly, an investment in table processing must bring with it benefits that exceed the expenses involved. If it is always easier to recover the desired information through some other means (by browsing, say, or via a simple keyword query), then table processing serves no purpose. The formulation of such a model would be invaluable, and may very well provide insight into where we should apply our efforts to obtain the greatest possible return.

The vast majority of papers published to date have concentrated either on the problems associated with low-level analysis of printed tables, or on guidelines for table presentation, with comparatively little work on the topic of making tabular information *useful* (other than for highly specialized applications). What has changed to make this an interesting question to consider? The unprecedented explosion in the amount of information people are confronted with each day. Whereas large-scale databases were once the province of a select few, nowadays anyone with Internet access and an e-mail account is inundated with vast quantities of unstructured (or at best semi-structured) data. Automated table processing presents one promising way of recovering useful, familiar structure making it possible to realize more of the benefits of universal data access.

Industrial applications of table-processing technology also play a role in moving the state of the art forward. Such solutions have the potential to give data processing shops an advantage in throughput. These businesses service, for example, medical insurance companies (where tabulated claim data is verified) and telephone companies (where competitive analysis is performed on customer billing statements). Business intelligence and financial services companies have an interest in enhancing reaction speeds to dense information such as SEC filings, news wire items (which often contain tabular data) and other tabulated financial data. In certain domains, archival tasks are wholly enabled by table-processing systems, as are data conversion and storage solutions (e.g., conversion to XML).

It may be time for table-processing research to make the transition from pixel and cell level analysis to table interpretation in a multi-document context.

References

1. Abu-Tarif, A.: Table processing and table understanding. Master's thesis, Rensselaer Polytechnic Institute, May (1998)

2. Alam, H., Rahman, F.: Web document manipulation for small screen devices: A review. In: Proceedings of the Second International Workshop on Web Document Analysis (WDA2003) (2003) http://www.csc.liv.ac.uk/˜wda2003/Papers/Section_II/Paper_8.pdf
3. Alam, H., Rahman, F., Tarnikova, Y.: When is a list is a list?: Web page re-authoring for small display devices. In: Proceedings of the Twelfth International World Wide Web Conference, Budapest, Hungary, May (2003) http://www2003.org/cdrom/papers/poster/p054/p54-Alam.htm
4. Alrashed, S., Gray, W.A.: Detection approaches for table semantics in text. In: Lopresti, D., Hu, J., Kashi, R. (eds.) Document Analysis Systems V. Lecture Notes in Computer Science, vol. 2423, pp. 287–290. Springer-Verlag, Berlin, Germany (2002)
5. 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)
6. Anjewierden, A.: AIDAS: Incremental logical structure discovery in PDF documents. In: Proceedings of the Sixth International Conference on Document Analysis and Recognition, pp. 374–378. Seattle, WA (2001)
7. Arias, J.F., Balasubramanian, S., Prasad, A., Kasturi, R., Chhabra, A.: Information extraction from telephone company drawings. In: Proceedings of the Conference on Computer Vision and Pattern Recognition, pp. 729–732. Seattle, Washington, June (1994)
8. Arias, J.F., Chhabra, A., Misra, V.: Efficient interpretation of tabular documents. In: Proceedings of the International Conference on Pattern Recognition (ICPR'96), vol. III, pp. 681–685. Vienna, Austria (1996)
9. Arias, J.F., Chhabra, A., Misra, V.: Interpreting and representing tabular documents. In: Proceedings of the Conference on Computer Vision and Pattern Recognition, pp. 600–605. San Francisco, CA (1996)
10. Arias, J.F., Kasturi, R.: Efficient techniques for line drawing interpretation and their application to telephone company drawings. Technical Report CSE TR CSE-95-020, Penn State University (1995)
11. Balasubramanian, S., Chandran, S., Arias, J.F., Kasturi, R., Chhabra, A.: Information extraction from tabular drawings. In: Proceedings of Document Recognition I (IS&T/SPIE Electronic Imaging'94), vol. 2181, pp. 152–163. San Jose, CA (1994)
12. Bayer, T.A.: Understanding structured text documents by a model based document analysis system. In: Proceedings of the Second International Conference on Document Analysis and Recognition (ICDAR'93), pp. 448–453. Tsukuba Science City, Japan (1993)
13. 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)
14. Cesarini, F., Marinari, S., Sarti, L., Soda, G.: Traininable table location in document images. In: Proceedings of the International Conference on Pattern Recognition, vol. III, pp. 236–240 (2002)
15. 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)
16. Chao, H.: Background pattern recognition in multi-page PDF document. In: Proceedings of the Third International Workshop on Document Layout Interpretations and its Applications (DLIA2003), pp. 41–45 (2003) http://www.science.uva.nl/events/dlia2003/program/41-46-chao.pdf
17. Chao, H., Fan, J.: Layout and content extraction for PDF documents. In: Marinai, S., Dengel, A. (eds.) Document Analysis Systems VI. Lecture Notes in Computer Science, vol. 3163, pp. 213–224. Springer-Verlag, Berlin, Germany (2004)
18. Chen, P.: The entity-relationship model—toward a unified view of data. ACM Trans. Database Syst. **1**(1) (1976)
19. Chhabra, A.K., Misra, V., Arias, J.: Detection of horizontal lines in noisy run length encoded images: The FAST method. In: Kasturi, R., Tombre, K. (eds.) Graphics Recognition—Methods and Applications. Lecture Notes in Computer Science, vol. 1072, pp. 35–48. Springer-Verlag, Berlin, Germany (1996)
20. Chiang, R., Barron, T., Storey, V.: Reverse engineering of relational databases: Extraction of an EER model from a relational database. Data Knolw. Eng. **12**(1),107–142 (1994)
21. Codd, E.: A relational model for large shared data banks. Commun. ACM **13**(6),377–487 (1970)
22. Cohen, W., Hurst, M., Jensen, L.S.: A flexible learning system for wrapping tables and lists in HTML documents. In: Proceedings of the Eleventh International World Wide Web Conference WWW-2002 (2002)
23. Corporation, L.D.: Improv Handbook (1991)
24. Coüasnon, B.: DMOS: A generic document recognition method, application to an automatic generator of musical scores, mathematical formulae and table structures recognition systems. In: Proceedings of the Sixth International Conference on Document Analysis and Recognition. Seattle, WA (2001)
25. 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)
26. Douglas, S., Hurst, M., Quinn, D.: Using natural language processing for identifying and interpreting tables in plain text. In: Proceedings of the Symposium on Document Analysis and Information Retrieval (SDAIR'95), pp. 535–545. Las Vegas, NV (1995)
27. Embley, D., Kurtz, B., Woodfield, S.: Object-oriented Systems Analysis: A Model Driven Apprach. Yourdon Press (1992)
28. Embley, D., Tao, C., Liddle, S.: Automating the extraction of data from HTML tables with unknown structure. Data Knowl. Eng. (in press)
29. Gray, P., Embury, S., Gray, W., Hui, K.: An agent-based system for handling distributed design constraints. In: Proceedings of Agents'98 (1998)
30. Green, E.A.: Ph.d. research (1997). http://tardis.union.edu/greene/research-dir/research.html
31. Green, E.A., Krishnamoorthy, M.: Model-based analysis of printed tables. In: Proceedings of the First International Workshop on Graphics Recognition (GREC'95), pp. 234–242. PA (1995)
32. Green, E.A., Krishnamoorthy, M.: Model-based analysis of printed tables. In: Proceedings of the Third International Conference on Document Analysis and Recognition (ICDAR'95), pp. 214–217. Montréal, Canada (1995)
33. Green, E.A., Krishnamoorthy, M.: Recognition of tables using table grammars. In: Proceedings of the Symposium on Document Analysis and Information Retrieval (SDAIR'95), pp. 261–277. Las Vegas, NV (1995)
34. Gruber, T.: A translation approach to portable ontology specifications. Knowl. Acquis. **5**,199–220 (1993)
35. Guthrie, J.T., Britten, T., Barker, K.G.: Roles of document structure, cognitive strategy, and awareness in searching for information. Int. Read. Assoc. (1991)
36. Haas, T.: The development of a prototype knowledge-based table-processing system. Master's thesis, Brigham Young University, Provo, Utah (1998)
37. Handley, J.C.: Document recognition. In: Dougherty, E.R. (ed.) Electronic Imaging Technology, chapter 8. SPIE—The International Society for Optical Engineering (1999)
38. Handley, J.C.: Table analysis for multiline cell identification. In: Kantor, P.B., Lopresti, D.P., Zhou, J. (eds.) Proceedings of Document Recognition and Retrieval VIII (IS&T/SPIE Electronic Imaging), vol. 4307. San Jose, CA (2001)
39. 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. Montréal, Canada (1995)

40. Hori, O., Doermann, D.S.: Robust table-form structure analysis based on box-driven reasoning. In: Proceedings of the Third International Conference on Document Analysis and Recognition (ICDAR'95), pp. 218–221. Montréal, Canada (1995)
41. Hu, J., Kashi, R., Lopresti, D., Nagy, G., Wilfong, G.: Why table ground-truthing is hard. In: Proceedings of the Sixth International Conference on Document Analysis and Recognition, pp. 129–133. Seattle, WA (2001)
42. Hu, J., Kashi, R., Lopresti, D., Wilfong, G.: Medium-independent table detection. In: Lopresti, D.P., Zhou, J. (eds.) Proceedings of Document Recognition and Retrieval VII (IS&T/SPIE Electronic Imaging), vol. 3967, pp. 291–302. San Jose, CA (2000)
43. Hu, J., Kashi, R., Lopresti, D., Wilfong, G.: A system for understanding and reformulating tables. In: Proceedings of the Fourth IAPR International Workshop on Document Analysis Systems, pp. 361–372. Rio de Janeiro, Brazil (2000)
44. 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)
45. Hu, J., Kashi, R., Lopresti, D., Wilfong, G.: Evaluating the performance of table processing algorithms. Int. J. Doc. Anal. Recognit. **4**(3),140–153 (2002)
46. Hurst, M.: Layout and language: beyond simple text for information interaction—modelling the table. In: Proceedings of the Second International Conference on Multimodal Interfaces. Hong Kong (1999)
47. Hurst, M.: The Interpretation of Tables in Texts. Ph.D. thesis, University of Edinburgh (2000)
48. 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)
49. Insiders – knowledge-management specialists, February 2005. http://www.insiders.de/
50. Itonori, K.: A table structure recongnition 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)
51. Kieninger, T., Dengel, A.: A paper-to-HTML table converting system. In: Proceedings of Document Analysis Systems (DAS) 98. Nagano, Japan (1998)
52. Kieninger, T., Dengel, A.: Applying the T-Recs table recognition system to the business letter domain. In: Proceedings of the Sixth International Conference on Document Analysis and Recognition. Seattle, WA (2001)
53. 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)
54. Klein, B., Agne, S., Bagdanov, A.D.: Understanding document analysis and understanding (through modeling). In: Proceedings of the Seventh International Conference on Document Analysis and Recognition (ICDAR'03), pp. 1218–1222. Edinburgh, Scotland (2003)
55. Klein, B., Agne, S., Dengel, A.: Results of a study on invoice-reading systems in Germany. In: Marinai, S., Dengel, A. (eds.) Document Analysis Systems VI. Lecture Notes in Computer Science, vol. 3163, pp. 451–462. Springer-Verlag, Berlin, Germany (2004)
56. Klein, B., Dengel, A.R.: Problem-adaptable document analysis and understanding for high-volume applications. Int. J. Doc. Anal. Recognit. **6**(3),167–180 (2004)
57. Klein, B., Gökkus, S., Kieninger, T., Dengel, A.: Three approaches to "industrial" table spotting. In: Proceedings of the Sixth International Conference on Document Analysis and Recognition, pp. 513–517. Seattle, WA (2001)
58. Kornfeld, W., Wattecamps, J.: Automatically locating, extracting and analyzing tabular data. In: Proceedings of the 21st International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 347–348. Melbourne, Australia (1998)
59. Lamport, L.: $\LaTeX$: A Document Preparation System. Addison-Wesley, Reading, MA (1985)
60. 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)
61. Lesk, M.: Tbl—a program to format tables. In: UNIX Programmer's Manual, vol. 2A. Bell Telephone Laboratories, Murray Hill, NJ (1979)
62. Lopresti, D., Nagy, G.: Automated table processing: An (opinionated) survey. In: Proceedings of the Third IAPR International Workshop on Graphics Recognition, pp. 109–134. Jaipur, India (1999)
63. Lopresti, D., Nagy, G.: A tabular survey of automated table processing. In: Chhabra, A.K., Dori, D. (eds.) Graphics Recognition: Recent Advances. Lecture Notes in Computer Science, vol. 1941, pp. 93–120. Springer-Verlag, Berlin, Germany (2000)
64. Luo, Q., Watanabe, T., Yoshida, Y., Inagaki, Y.: Recognition of document structure on the basis of spatial and geometric relationships between document items. In: Proceedings of MVA '90, pp. 461–464 (1990)
65. Maedche, A., Staab, S.: Ontology learning for the semantic web. IEEE Intell. Syst. (2001)
66. Maier, D.: The Theory of Relational Databases. Computer Science Press Inc., Rockville, Maryland (1983)
67. Nagy, G.: Twenty years of document image analysis in PAMI. IEEE Trans. Pattern Anal. Mach. Intell. **22**(1),38–62 (2000)
68. Nagy, G., Seth, S.: Hierarchical representation of optically scanned documents. In: Proceedings the International Conference on Pattern Recognition (ICPR), pp. 347–349 (1984)
69. Nielson, H., Barrett, W.: Consensus-based table form recognition. In: Proceedings of the Seventh International Conference on Document Analysis and Recognition, pp. 906–910. (2003)
70. O'Gorman, L.: The document spectrum for structural page layout analysis. IEEE Trans. Pattern Anal. Mach. Intell. **15**(11):1162–1173 (1993)
71. Peterman, C., Chang, C.H., Alam, H.: A system for table understanding. In: Proceedings of the Symposium on Document Image Understanding Technology (SDIUT'97), pp. 55–62. Annapolis, MD (1997)
72. 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)
73. Pyreddy, P., Croft, W.B.: TINTIN: A system for retrieval in text tables. Technical Report UM-CS-1997-002. University of Massachusetts, Amherst (1997)
74. Rahgozar, M.A., Cooperman, R.: A graph-based table recognition system. In: Proceedings of Document Recognition III (IS&T/SPIE Electronic Imaging'96), vol. 2660, pp. 192–203. San Jose, CA (1996)
75. Ramel, J., Crucianu, M., Vincent, N., Faure, C.: Detection, extraction and representation of tables. In: Proceedings of the Seventh International Conference on Document Analysis and Recognition (2003)
76. Recording for the Blind and Dyslexic, Princeton, NJ. The 1.7 Tag Set Usage Guide (1994)
77. Rus, D., Subramanian, D.: Customizing information capture and access. ACM Trans. Inf. Syst. **15**(1),67–101 (1997)
78. Rus, D., Summers, K.: Using white space for automated document structureing. Technical Report TR94-1452, Cornell University, Department of Computer Science (1994)
79. ScanSoft OmniPage, February (2005) http://www.scansoft.com /omnipage/

80. 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)
81. Sproat, R., Hu, J., Chen, H.: EMU: an e-mail preprocessor for text-to-speech. In: Proceedings of the IEEE Workshop on Multimedia Signal Processing, pp. 239–244. Los Angeles, CA (1998)
82. Summers, K.: Automatic Discovery of Logical Document Structure. Ph.D. thesis, Cornell University (1998)
83. TCG Informatik AG—Data capture at its best (2005). http://www.tcginf.ch
84. Tijerino, Y., Embley, D., Lonsdale, D., Nagy, G.: Towards ontology generation from tables. World Wide Web J. (in press)
85. TREC Data—English Test Questions (Topics). http://trec.nist.gov/data/testq_eng.html
86. Tsuruoka, S., Takao, K., Tanaka, T., Yoshikawa, T., Shinogi, T.: Region segmentation for table image with unknown complex structure. In: Proceedings of the Sixth International Conference on Document Analysis and Recognition. Seattle, WA (2001)
87. 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)
88. Wang, X.: Tabular abstraction, editing, and formatting. Ph.D. thesis, University of Waterloo (1996)
89. Wasserman, H., Yukawa, K., Sy, B., Kwok, K.-L., Phillips, I.T.: A theoretical foundation and a method for document table structure extraction and decomposition. In: Lopresti, D., Hu, J., Kashi, R. (eds.) Document Analysis Systems V. Lecture Notes in Computer Science, vol. 2423, pp. 291–294. Springer-Verlag, Berlin, Germany (2002)
90. Watanabe, T., Fukumura, T.: A framework for validating recognized results in understanding table-form documents. In: Proceedings of the Third International Conference on Document Analysis and Recognition, pp. 536–539 (1995)
91. Watanabe, T., Luo, Q., Sugie, N.: Towards a practical document understanding of table-form documents: its framework and knowledge representation. In: Proceedings of the Second International Conference on Document Analysis and Recognition, pp. 510–515 (1993)
92. Watanabe, T., Naruse, H., Lou, Q., Sugie, N.: Structure analysis of table-form document on the basis of the recognition of vertical and horizontal line segments. In: Proceedings of the First International Conference on Document Analysis and Recognition, pp. 638–646 (1991)
93. Watanabe, T., Quo, Q.L., Sugie, N.: Layout recognition of multi-kinds of table-form documents. IEEE Trans. Pattern Anal. Mach. Intell. **17**(4),432–445 (1995)
94. Whittaker, S., Sidner, C.: Email overload: exploring personal information management of email. In: Proceedings of the Conference on Human Factors in Computing Systems (CHI), pp. 276–283. Vancouver, British Columbia, Canada (1996)
95. Wright, P.: A user-oriented approach to the design of tables and flowcharts. In: Jonassen, D.H. (ed.) The Technology of Text. Educational Technology Publications (1982)
96. XML Cities: XML Content for a New Era (2005) http://www.xmlcities.com
97. Zanibbi, R., Blostein, D., Cordy, J.R.: A survey of table recognition: Models, observations, transformations, and inferences. Int. J. Doc. Anal. Recognit. **7**(1),1–16 (2004)
98. Zuyev, K.: Table image segmentation. In: Proceedings of the International Conference on Document Analysis and Recognition (ICDAR'97), pp. 705–708. (1997)