An Abstract Model for Tables

Xinxin Wang
Department of Computer Science, University of Waterloo
Waterloo, Ontario N2L 3G1, Canada
internet: wang@watdragon.uwaterloo.ca.

Derick Wood
Department of Computer Science, University of Western Ontario
London, Ontario N6A 5B7, Canada
Internet: dwood@csd.uwo.ca.

Abstract

We present a tabular model that abstracts a wide range of tables. We abstract the logical structure of tables, rather than their presentational form. The model can be used to guide the design and implementation of tabular editors and formatters. In addition, the model is formatter independent; it can be used to direct the formatting of tables in many typesetting systems, including TEX.

Introduction

Although tables are widely used in daily life to convey information in a compact and convenient form, tabular processing is one of the most difficult parts of document processing, because tables are more complex than other textual objects. The separation of the logical and layout structures of documents is widely used in many document formatting systems (Lamport (1985); Quint and Vatton (1986); and Reid (1980)). It enables authors to focus on the manipulation of the logical structure of a document. The layout structure is determined by the formatting systems based on style specifications; thus, high quality typeset documents can be produced with little or no help from typographers. Tabular formatting is, however, the weak link in most formatting systems. The main reason is that the tabular models used in many systems (Beach (1995); Biggerstaff et al. (1984); Cameron (1989); Lamport (1985); and Lesk (1979)) are **presentation dependent;** that is, the models describe tables based on their presentational form. In other words, it is the user's responsibility to design the geometric arrangement of tabular components. Some systems (Improv Handbook (1991) and Vanoirbeek and Coray, eds. (1992)) use presentation-independent models for tables that are based on their logical structure; however, the models fall short in that they are made with specific environments in mind. The strength of our model is that it is not tied to any specific realization and it can be viewed as an abstract data type. One other drawback of most tabular systems is that the tabular operations that are provided are too weak to manipulate tables based on the logical relationships among tabular components.

We are currently developing a tabular composition system based on this model, which can be used as a front end for LATEX tables.

In this paper, we first summarize the main characteristics of tables, and then present our model. To conclude the presentation, we compare our model with Vanoirbeek's model and also discuss the influence of our model on the design and implementation of a tabular composition system.

The Characteristics of Tables

The Oxford English Dictionary defines a table as: "an arrangement of numbers, words or items of any kind, in a definite and compact form, so as to exhibit some set of facts or relations in a distinct and comprehensive way, for convenience of study, reference, or calculation". This definition summarizes the characteristics of a table using three different aspects: content, form and function.

The content of a table. The content of a table is a collection of interrelated items, which can be numbers, text, symbols, figures, mathematical equations, or even other tables. In most tables, these items can be divided into two classes based on their function in the table: **entries,** which are facts of any kind that we present in a table, and **labels,** which we use to locate the entries. The logical relationships among the items of a table are the associations among labels and entries. Each entry is associated with a set

Table 1: The average marks of CS351(1991-1992).

Year / Mark / Term	Assignments		Exams		Final
	Ass1	Ass2	Midterm	Final	marks
1991					
Winter	85	80	70	73	74
Summer	78	79	65	70	70
Fall	82	80	–	80	80
1992					
Winter	83	78	72	75	75
Summer	80	76	–	78	78
Fall	76	74	60	80	72

Table 2: The average marks of CS351(1991-1992).

Term / Year / Mark	Winter		Summer		Fall	
	1991	1992	1991	1992	1991	1992
Assignments						
Ass1	85	83	78	80	82	76
Ass2	80	78	79	76	80	74
Exams						
Midterm	70	72	65	–	–	60
Final	73	75	70	78	80	80
Final Marks	74	75	70	78	80	72

of labels; for example, in Table 1, entry 85 is associated with labels 1991, Winter, Assignments and Ass1. The items and the logical relationships among them provide the **logical structure** of a table, which is the primary information conveyed by the table and which is independent of its presentational form.

We can describe the logical structure of a wide range of tables in this way: first, we group the labels into n **categories** such that in each category labels are organized in a tree structure, and then we associate each entry with one, or more, n-element sets of label sequences where each label sequence is the catenation of labels on the path from the root to a leaf in a category. For example, the labels of Table 1 can be grouped into three categories:

Year = {1991, 1992},
Term = {Winter, Summer, Fall}, and
Mark = {Assignments, Exams, Final marks}.

In category Mark, there are two subcategories:

Assignments = {Ass1, Ass2} and
Exams = {Midterm, Final}.

Entry 85 is associated with a 3-element set of label sequences: {Year.1991, Term.Winter, Mark.Assignments.Ass1}; Entry 76, which appears in the table twice, is associated with two 3-element sets of label sequences: {Year.1992, Term.Fall, Mark.Assignments.Ass1} and {Year.1992, Term.Summer, Mark.Assignments.Ass2}.

The form of a table. The content of a table must be presented in some form and on some medium. Usually, tables are presented as a row-column structure on a two-dimensional plane, such as paper or screen. The presentational form of a table consists of two components: the topological arrangement and the typographic specification. The topological arrangement is an arrangement of the table components in two-dimensional space such that the logical structure of the table is clearly conveyed; for example, where to put the labels and entries and how to order the labels in a category. The typographic specification is a group of formatting attributes for rendering tabular data and the graphic objects that are used to outline the topological arrangement, such as the font type for entries, the line style for rules, and so on. The content of a table can be presented with different topological arrangements and different typographic specifications. For example, Tables 1 and 2 are two different presentations for a three-category table. Although the row-column structure is a familiar and natural form for tabular presentation, tables may also be presented in other forms, such as the bar graph, the pie graph, and so on.

The function of a table. The main function of a table is to convey data and its relationship in a compact and convenient way.

The Tabular Model

In our opinion, a tabular composition system should allow users to be mainly concerned about the logical structure of tables; they should leave the presentational form to a high-quality tabular formatting system that requires little or no user intervention. A tabular model for such a system should possess the following characteristics:

- it can be used to abstract a wide range of tables;
- it is presentation independent; that is, it captures the logical structure of tables and ignores any topological and typographic attributes; and
- it includes a group of operations that support tabular manipulation.

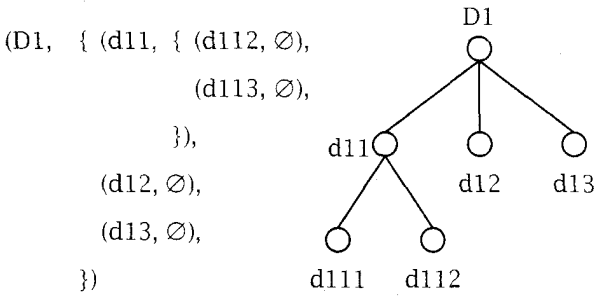

Figure 1: The relationship between a labeled domain and its tree.

We specify our tabular model with mathematical notions so as to avoid the representational structure and the implementation details. Therefore, the model can also be viewed as an abstract data type; that is, an abstract table and a set of operations.

Terminology. We first define some terminology before we give the definition of an abstract table.

A **labeled set** is a label together with a set. We denote a labeled set as $(label, set)$.

A **labeled domain** is defined inductively as follows:

1. A labeled empty set $(L, \emptyset)$ is a labeled domain.
2. A labeled set of labeled domains is a labeled domain.
3. Only labeled sets that are obtained with rules 1 and 2 are labeled domains.

A labeled domain can be represented by an unordered tree of labels. Figure 1 presents the relationship between a labeled domain and its tree. Each node in the tree represents a labeled domain. For convenience, we will use the tree of a labeled domain to explain some concepts that are related to labeled domains. If a labeled domain $D = (L, S)$, we use $L[D]$ to denote the label L, of D, and $S[D]$ to denote the set S of D.

A **label sequence** is either one label or the catenation of multiple labels separated with the symbol '.'. With the tree of Figure 1, $D1$ and $D1.d11.d111$ are two examples of label sequences. Operation $\odot$ takes a *label* and a set of label sequences $\{s_1, \cdots, s_n\}$ as operands and its result is a set of label sequences such that

$$label \odot \{s_1, \cdots, s_n\} = \{label.s_1, \cdots, label.s_n\}.$$

The **frontier** of a labeled domain D is the set of external nodes of the tree of D. It is denoted by $F[D]$ and is defined inductively as

1. $\{label\}$, if $D = (label, \emptyset)$;
2. $label \odot (\bigcup_{x \in S} F[x])$, if $D = (label, S)$.

With the labeled domain of Figure 1, $F[D1] = \{D1.d11.d111, D1.d11.d112, D1.d12, D1.d13\}$.

A **frontier item** of label domain D is any member of F[D]. It is the label sequence on a path from the root to an external node in the tree of labeled domain D.

An **item** of a labeled domain D is any prefix of a frontier item of D; it is the label sequence on a path from the root to a node in the tree of D. With the labeled domain of Figure 1, $D1$, $D1.d11$, $D1.d11.d111$, $D1.d11.d112$, $D1.d12$ and $D1.d13$ are all items of the labeled domain $D1$. An item is actually the label sequence on the path from the root to a node in the tree of the labeled domain D. We use an item to identify its associated node.

If i is an item, we use $LD[i]$ to denote the labeled domain represented by the associated node of i and $P[i]$ to denote the item that identifies the direct parent of the associated node of i. For example, with the labeled domain of Figure 1, if $i = D1.d11.d112$, then $LD[i]$ is the labeled domain $(d112, \emptyset)$ and $P[i] = D1.d11$. If $LD[i] = (L, S)$, we also use $L[i]$ to denote the label L, $S[i]$ to denote the set S, and $F[i]$ to denote the frontier of $LD[i]$.

The **dimension** of a labeled domain $D = (L, S)$ is denoted by $Dim[D]$; it is the number of elements in S. With the labeled domain of Figure 1, $D1 = (D1, \{d11, d12, d13\})$; thus, $Dim[D1] = 3$. We say that two items i and j are in the same dimension of D if and only if both the associated nodes of i and j are in a child subtree of the tree of D. For example, with the labeled domain of Figure 1, $D1.d11$ and $D1.d11.d111$ are in the same dimension of $D1$, but $D1.d11$ and $D1.d12$ are not.

For $n > 1$, an **n-set** is a set of n elements. For two sets A and B, $A \otimes B$ is the set of all 2-sets that consists of one element of A and one element of B. $A \otimes B$ is similar to, yet different from, $A \times B$, the Cartesian product of A and B. It is similar in that we take all pairs of elements, one from A and one from B; it is different because we obtain unordered pairs, rather than ordered pairs. It is **unordered Cartesian product**.

We now apply $\otimes$ to labeled domains as follows. It takes a labeled domain $D = (L, S)$ as operand and it results in a set in which each element contains $Dim[D]$ frontier items, each of which identifies an external node of a labeled domain in S; that is,

$$\begin{aligned} \otimes D &= \emptyset, \quad \text{if } S = \emptyset \\ &= \{L \odot \{t_1, \cdots, t_n\} \mid t_i \in F[D_i], 1 \le i \le n\}, \\ &\quad \text{if } S = \{D_1, \cdots, D_n\}. \end{aligned}$$

Table 3: A three-category table.

D1 \ D2 \ D3		d31		d32
		d311	d312	
d11	d21	e1	e2	e5
	d22	e3		
	d23	e4		
d12	d21	e6	e7	e9
	d22			
	d23		e8	

Table 4: A two-category table.

D1 \ D2	S1				S2
	S11		S12	S13	
	S111	S112			
L1	e1	e2	e3	e4	e5
L2	e6	e7	e8	e9	e10
L3	e11	e12	e13	e14	e15

For example, with the labeled domain of Figure 1,

$$\otimes D1 = \{\{D1.d11.d111,\ D1.d12,\ D1.d13\},\ \{D1.d11.d112,\ D1.d12,\ D1.d13\}\}.$$

The definition of an abstract table. An **abstract table** consists of three elements: a labeled domain, a set of entries, and a function from a set of n-element sets of frontier items (n is the dimension of the labeled domain) to the set of entries. It can be formally defined by a tuple (D, E, δ), where

- D is a labeled domain
- E is a set of entries
- δ is a partial function from $\otimes D$ onto E

We use a labeled domain D to describe the category structure of a table, the dimension of a labeled domain corresponds to the number of categories of the table, and each labeled domain in $S[D]$ corresponds to a category. We use a function to describe the logical associations among labels and entries. Using this model, Table 3 can be abstracted by (D, E, δ), where

$$\begin{aligned} D = (D,\ &\{(D1,\ \{(d11, \varnothing),\ (d12, \varnothing)\}\ ),\\ &(D2,\ \{(d21, \varnothing),\ (d22, \varnothing),\ (d23, \varnothing)\}),\\ &(D3,\ \{(d31,\ \{(d311, \varnothing),\ (d312, \varnothing)\}),\\ &\qquad (d32, \varnothing)\ \}\\ &)\\ &\}\\ ),& \end{aligned}$$

$E = \{e1, e2, e3, e4, e5, e6, e7, e8, e9\}$, and

$$\begin{aligned}
&\delta(\{D.D1.d11,\ D.D2.d21,\ D.D3.d31.d311\}) = e1;\\
&\delta(\{D.D1.d11,\ D.D2.d21,\ D.D3.d31.d312\}) = e2;\\
&\delta(\{D.D1.d11,\ D.D2.d22,\ D.D3.d31.d311\}) = e3;\\
&\delta(\{D.D1.d11,\ D.D2.d22,\ D.D3.d31.d312\}) = e3;\\
&\delta(\{D.D1.d11,\ D.D2.d23,\ D.D3.d31.d311\}) = e4;\\
&\delta(\{D.D1.d11,\ D.D2.d21,\ D.D3.d32\}) = e5;\\
&\delta(\{D.D1.d11,\ D.D2.d22,\ D.D3.d32\}) = e5;\\
&\delta(\{D.D1.d11,\ D.D2.d23,\ D.D3.d32\}) = e5;\\
&\delta(\{D.D1.d12,\ D.D2.d21,\ D.D3.d31.d311\}) = e6;\\
&\delta(\{D.D1.d12,\ D.D2.d21,\ D.D3.d31.d312\}) = e7;\\
&\delta(\{D.D1.d12,\ D.D2.d22,\ D.D3.d31.d312\}) = e7;\\
&\delta(\{D.D1.d12,\ D.D2.d23,\ D.D3.d31.d312\}) = e8;\\
&\delta(\{D.D1.d12,\ D.D2.d21,\ D.D3.d32\}) = e9;\\
&\delta(\{D.D1.d12,\ D.D2.d22,\ D.D3.d32\}) = e9;\\
&\delta(\{D.D1.d12,\ D.D2.d23,\ D.D3.d32\}) = e9.
\end{aligned}$$

Basic operations for abstract tables. We define a basic set of operations for tabular editing. These operations are divided into four groups: operations for categories, items, labels, and entries. For each operation, we first give its name and the types of its operands and result, and then explain its semantics informally.

Category operations. There are two operations for categories.

The operation **Add_Category** adds a new category to a table. It takes a table $T = (D, E, \delta)$ and a labeled domain D_m as operands, and returns a new table $T' = (D', E', \delta')$ such that:

(1) D' is produced by inserting D_m into the set of D;

(2) the entry set E' is the same as E;

(3) δ' maps any $fs \in \otimes D'$, which contains an element $L[D].f$ such that f is a frontier item of D_m, to $\delta(fs - \{L[D].f\})$. For example, if T is Table 4, Add_Category$(T, D3)$, where $D3 = (D3, \{(T1, \varnothing), (T2, \varnothing)\})$, produces Table 5.

The operation **Remove_Category** removes a category from a table. It takes a table $T = (D, E, \delta)$ and an item d_i (which must identify an element of the set of labeled domain D) as operands, and returns a new table $T' = (D', E', \delta')$ such that:

(1) D' is produced by deleting the labeled domain $LD[d_i]$ from the set of D;

Table 5: After adding a new category to Table 4.

D3	D2 / D1	S1				S2
		S11		S12	S13	
		S111	S112			
T1	L1	e1	e2	e3	e4	e5
	L2	e6	e7	e8	e9	e10
	L3	e11	e12	e13	e14	e15
T2	L1	e1	e2	e3	e4	e5
	L2	e6	e7	e8	e9	e10
	L3	e11	e12	e13	e14	e15

Table 6: After removing a category from Table 5.

D3	D2	S1				S2
		S11		S12	S13	
		S111	S112			
T1		e1	e2	e3	e4	e5
		e6	e7	e8	e9	e10
		e11	e12	e13	e14	e15
T2		e1	e2	e3	e4	e5
		e6	e7	e8	e9	e10
		e11	e12	e13	e14	e15

(2) E' is a set in which each element is $\{\delta(fs \cup \{L[D].k_1\}), \cdots, \delta(fs \cup \{L[D].k_m\})\}$ where fs is any element of $\otimes D'$ and $k_1, \cdots, k_m$ are all frontier items of $LD[d_i]$;

(3) δ' maps any $fs \in \otimes D'$ to set $\{\delta(fs \cup \{L[D].k_1\}), \cdots, \delta(fs \cup \{L[D].k_m\})\}$. For example, if T is Table 5, then Remove_Category$(T, D1)$ produces Table 6.

Item operations. There are four operations for items.

The operation **Insert_Item** inserts a labeled tree to a category. It takes a table $T = (D, E, \delta)$, one of its items p (which cannot be D) and a labeled domain C as operands and returns a new table $T' = (D', E', \delta')$ such that:

(1) D' is produced by inserting C into the tree of D such that C will be a child of $LD[p]$;

(2) E' is the same as E;

(3) if p is a frontier item of D, then δ' will map every element $fs \in \otimes D'$ which contains $p.f$ such that f is a frontier item of C, to $\delta((fs - \{p.f\}) \cup \{p\})$; otherwise, δ' on these elements is undefined; for other $fs \in \otimes D'$, $\delta'(fs)$ is the same as $\delta(fs)$. For example, if T is Table 4, Insert_Item$(T, D.D2.S1, C)$, where $C = (S14, \varnothing)$, produces Table 7.

Table 7: After inserting an item to Table 4.

D1	D2	S1					S2
		S11		S12	S13	S14	
		S111	S112				
L1		e1	e2	e3	e4		e5
L2		e6	e7	e8	e9		e10
L3		e11	e12	e13	e14		e15

Table 8: After deleting an item from Table 4.

D1	D2	S1			S2
		S11		S13	
		S111	S112		
L1		e1	e2	e4	e5
L2		e6	e7	e9	e10
L3		e11	e12	e14	e15

The operation **Delete_Item** deletes a labeled tree from a category. It takes a table $T = (D, E, \delta)$ and one of its items i (which cannot be D or any item that identifies a child of D) as operands, and returns a new table $T' = (D', E', \delta')$ such that:

(1) D' is produced by removing the labeled domain $LD[i]$ from D;

(2) E' is produced by removing all entries that are not mapped from any element in $\otimes D'$ by δ;

(3) if the old parent of i, i.e, $P[i]$, becomes a frontier item, then δ' on any $fs \in \otimes D'$, which contains $P[i]$, is undefined; for other $fs \in \otimes D'$, $\delta'(fs)$ is the same as $\delta(fs)$. For example, if T is Table 4, Delete_Item$(T, D.D2.S1.S12)$ produces Table 8.

The operation **Move_Item** moves a subtree to a new place within a category. It takes a table $T = (D, E, \delta)$ and two of its items c and p that

Table 9: After moving an item in Table 4.

D2 / D1	S1		S11		S2
	S12	S13	S111	S112	
L1	e3	e4	e1	e2	e5
L2	e8	e9	e6	e7	e10
L3	e13	e14	e11	e12	e15

are in the same dimension of D (p cannot be a descendant of c) as operands, and returns a new table $T' = (D', E', \delta')$ such that:

(1) D' is produced by moving labeled domain $LD[c]$ to be a child of labeled domain $LD[p]$;

(2) E' is the same as E;

(3) δ' maps any $fs \in \otimes D'$ which contains item $p.t$ where t is a frontier item of $LD[c]$ to $\delta((fs - \{p.t\}) \cup \{P[c].t\})$, and if the old parent of c , i.e, $P[c]$, become a frontier item of D', then δ on any $fs \in \otimes D'$, which contains $P[c]$, is undefined; for other $fs \in \otimes D'$, $\delta'(fs)$ is the same as $\delta(fs)$. For example, if T is Table 4, Move_Item$(T, D.D2.S1.S11, D.D2)$ produces Table 9.

The operation **Copy_Item** duplicates a subtree in a category. It takes a table $T = (D, E, \delta)$, two of its items c and p that are in the same dimension of D, and a label l as operands, and returns a new table $T' = (D', E', \delta')$ such that:

(1) D' is produced by copying labeled domain $LD[c]$ to be a child of labeled domain $LD[p]$ and assigning label l to the new labeled domain copied from $LD[c]$;

(2) E' is the same as E;

(3) if c is a frontier item of D, then δ' maps any $fs \in \otimes D'$ which contains item $p.l$ to $\delta((fs - \{p.l\}) \cup \{c\})$, otherwise, δ' maps any $fs \in \otimes D'$ which contains item $p.l.t$ such that $c.t$ is a frontier item of D to $\delta((fs - \{p.l.t\}) \cup \{c.t\})$; for other $fs \in \otimes D'$, $\delta'(fs)$ is the same as $\delta(fs)$. For example, if T is Table 4, Copy_Item$(T, D.D2.S1.S112, D.D2.S1, S14)$ produces Table 10.

Label operations. There are two operations for labels.

The operation **Put_Label** assigns a new label to a labeled domain. It takes a table $T = (D, E, \delta)$, one of its items i, and a label l as operands, and returns a new table by assigning the label l to labeled domain $LD[i]$.

Table 10: After copying an item in Table 4.

D2 / D1	S1					S2
	S11		S12	S13	S14	
	S111	S112				
L1	e1	e2	e3	e4	e2	e5
L2	e6	e7	e8	e9	e7	e10
L3	e11	e12	e13	e14	e12	e15

The operation **Get_Label** takes a table $T = (D, E, \delta)$ and one of its items i and returns the label of i.

Entry operations. There are two operations for entries.

The operation **Put_Entry** associates a new entry with a set of frontier items. It takes a table $T = (D, E, \delta)$, an entry e and a number of frontier items $f_1, \cdots, f_{Dim[D]}$ such that $\{f_1, \cdots, f_{Dim[D]}\}$ must be an element of $\otimes D$. It returns a new table by putting entry e into table T such that the new function maps $\{f_1, \cdots, f_{Dim[D]}\}$ to e. If the old entry mapped from $\{f_1, \cdots, f_{Dim[D]}\}$ is not mapped from any other element in $\otimes D$, it will be deleted from E.

The operation **Get_Entry** returns the entry that is associated to a set of frontier items. It takes a table T and a number of frontier items $f_1, \cdots, f_{Dim[D]}$ such that $\{f_1, \cdots, f_{Dim[D]}\}$ must be an element of $\otimes D$ as operands, and returns the entry that is mapped from $\{f_1, \cdots, f_{Dim[D]}\}$.

Conclusions

We have presented a tabular model that, although it is not a universal model, can be used to abstract a wide range of tables. This model is presentation independent because it abstracts only the logical structure of multi-dimensional tables and excludes any topological and typographic attributes. This characteristic makes it possible to design a tabular composition system in such a way that users are mainly concerned about the logical structure of tables, and the layout structure of a table is determined by the system based on style specifications. In this way, we can manipulate and format tables in a uniform way like other textual objects.

Our model is similar to Vanoirbeek's model (Vanoirbeek and Coray, eds., 1992), although we derived it independently. The major difference between these two models is the way to specify the logical structure of a table. In Vanoirbeek's model, the logical structure of a table is modeled by a tree with additional links: a table contains a set of logical dimensions and a set of items (entries); the logical dimensions include rubrics (labels) which may themselves contain subrubrics; links are used to represent the connections between items and rubrics. The main reason for this representation mechanism is to ensure that the table representation conforms with the hierarchical structured document representation used in the host system Grif (Quint and Vatton, 1986). In our model, the logical structure of a table is specified mathematically; it avoids the representational structure and implementation details. Our model is not tied to any specific environment; thus, we can develop a tabular composition system based on this model that can be used to direct the formatting of tables in different typesetting systems. Another difference is that the operations for rearranging the category structure and maintaining the logical relationships among labels and entries in Vanoirbeek's model and its Grif implementation are weaker than those in our model; for example, we can move and copy all labels in a subtree of a category and their associated entries.

Acknowledgements

We thank Darrell Raymond for his support and careful reading of a preliminary version of this paper.

Bibliography

Improv Handbook. Lotus Development Corporation, Cambridge, MA, 1991.

Beach, R. J. *Setting Tables and Illustrations with Style.* PhD thesis, University of Waterloo, Waterloo, Ontario, Canada, May 1985. Also issued as Technical Report CSL-85-3, Xerox Palo Alto Research Center, Palo Alto, CA.

Biggerstaff, Ted J., D. Mack Endres, and Ira R. Forman. "TABLE: Object Oriented Editing of Complex Structures". In *Proceeding of the 7th International Conference on Software Engineering,* pages 334-345, 1984.

Cameron, J. P. A Cognitive Model for Tabular Editing. Technical Report OSU-CISRC-6/89-TR 26, The Ohio State University, Columbus, OH, June 1989.

Lamport, Leslie. *LaTeX: A Document Preparation System.* Addison-Wesley, Reading, MA, 1985.

Lesk, M. E. "`tbl`—A Program to Format Tables". In *UNIX Programmer's Manual,* volume 2A. Bell Telephone Laboratories, Murray Hill, NJ, 7th edition, January 1979.

Quint, Vincent and Irène Vatton. "Grif: An Interactive System for Structured Document Manipulation". In *Text Processing and Document Manipulation, Proceedings of the International Conference,* pages 200-312, Cambridge, UK, 1986. Cambridge University Press.

Reid, Brian K. *Scribe: A Document Specification Language and its Compiler.* PhD thesis, Carnegie-Mellon University, Pittsburgh, PA, October 1980. Also issued as Technical Report CMU-CS-81-100, Carnegie-Mellon University.

Vanoirbeek, Christine. "Formatting Structured Tables". In C. Vanoirbeek & G. Coray, editor, *EP92(Proceedings of Electronic Publishing, 1992),* pages 291-309, Cambridge, UK, 1992. Cambridge University Press.