Table Diversions*

Kees van der Laan

Hunzeweg 57,
9893 PB Garnwerd, The Netherlands
`cgl@rug.nl`

September 1992

Abstract

Characteristics of existing table macro collections are given. A kaleidoscope of tables—as next best to a taxonomy—is presented. Newly introduced is the class of bordered tables. Variations in print—ruled, nonruled, framed, nonframed, dotted, centered, flushed—can be obtained via the invoke of parameter setting macros; no modification of user mark up. Simultaneous row and column spans, partial rules, and dotted lines are dealt with. The listing of the macro `\btable`, with auxiliaries, is included.

Keywords: (Bordered) Tables, framed and non-framed, ruled and non-ruled, simultaneous row and columns spans, partial rules, connected cells, plain TEX, SGML, education.

1 Introduction

The formatting of tables is considered complex, time consuming and expensive. Table making via TEX, or LATEX, needs too complex descriptions for simple tables, except when the table can be formatted via `\settabs`. For `\halign` use, there is no default template provided by plain. The $\langle X \rangle$TEX macros I have seen don't reflect the logical structure of the table. An apparently simple change of representation, for example from a ruled into a non-ruled, or from a framed into a non-framed table, requires in TEX, and LATEX, some non-negligible effort.

Therefore, there is a need for an easy to use macro on top of plain's powerful `\`$\langle X \rangle$`halign`-s, which accounts for the structure of the table.

Characteristics of existing table macros are given in the Existing TEX Table Packages section. In the Kaleidoscope section an anthology of tables is presented. Peculiar are the deterministic tables, and the tables which update memory. The section ends with the bordered table model. In the Bordered Table section the macro and examples of use are given. In the Blocks section blocks and connected cells are dealt with.

No landscape vs. portrait issues are discussed, nor the general use of rotated fonts. Tables which extend the page will not be discussed either. See for the latter for example supertabular.sty, [7], or longtable.sty, [4]. I also refrained from the issues which come from mapping the 'main vertical list' onto pages, under the restriction not to split tables over page boundaries.

2 Existing TEX table packages

Cowan, [5], provided a nice package in order to facilitate the formatting of simple tables. A main feature of his macros is that no template and no explicit number of columns, have to be provided. The template is created dynamically from the data and the 'attributes.' This is powerful and Hi-TEX on the one hand, but restrictive and time consuming on the other. The automatism can't be easily superseded, as opposed to a default repetitive template, which I have adopted. The functionality as demonstrated by his examples can be easily obtained via the bordered table macro.

Khanh, [8], introduced table macros which mainly deal with complex header rows. Because I consider those complicated header rows the exception rather than the rule, and because the material is not in the public domain, I will not consider it further.

Spivak, [23], provided some powerful macros for formatting tables separately from the main document. His merging, at the dvi level, of the formatted table into the main document, is Hi-TEX. He also provided some interesting examples of simultaneous row and column spans. Good, but complex, and perhaps too advanced for daily use. It is in the public domain.

*Presented at EuroTEX '92, September 14–18, Prague, Czechoslovakia; however different from proceedings EuroTEX '92 in the encoding of FIFO.

Lamport, [19], mainly provided facilities for placing a table appropriately within the context.[1] He also developed `\cline`, for partial horizontal rules, which is a powerful and a handy feature. I don't consider his way of framing nice, nor his way of specifying the ruling.

Hendrickson, [8], has an approach similar to mine. No bordered table model, however, and less pronunciated 'attributes.'

There exist more table macros. Of those I had access to, the ones discussed above seemed relevant to me. Now and then visual mark up packages appear, like Type and Set, [2]. With bars and visual mark up the table is specified on the screen, and from that TEX code is generated. However useful this might be within the 'conversion' of wordwhatever into TEX area, I don't like the resulting TEX encoding. It is clumsy. If such simple tables generate such an encoding then we are lost with really something a little more complex. Besides, the purpose of this paper is to introduce some new ways of formatting *simple* tables, next to the general accepted formatting TEXniques. The TEXbook considers alignment the general issue. I like to approach the issue from the structure point of view, and to couple it to what is already available via plain.[2] This work is a continuation of my earlier work on the SGML-TEX relation, [13].

3 Kaleidoscope

Some examples of special tables are given in order to illustrate the diversity. With respect to programming, the deterministic tables and the tables which update the computer memory, deserve special attention. The latter category is induced by *computer* typography.

- Pascal triangle, a deterministic table, [14]

$$\displaylines{1\cr 1\quad1\cr 1\quad2\quad1\cr 1\quad3\quad3\quad1\cr}$$

via

```
$$\displaylines{1\cr
  1\quad1\cr
  1\quad2\quad1\cr
  1\quad3\quad3\quad1\cr}$$
```

- Tower of Hanoi, deterministic, [15]. The process of replacement of the disks will be printed by the invoke of `\hanoi`$\langle n\rangle$, n an integer. No user mark up is needed.

- Young tableaux, irregular shape, [9], [25]

1	2	3	4
5	6	7	
8			

7	8	9	10
11	13		
16			

via

```
\def\young#1{\hbox{\vrule\vtop{\hrule\offinterlineskip\halign
{&\vbox{\hbox to\csize{\strut\hss##\hss\vrule}\hrule}\cr#1\crcr}}}}
%
\def\data{'&'&'&'\cr'&'&'\cr'}    \young\data\qquad
\def\data{7&8&9&10\cr11&13\cr16\cr}\young\data
```

with auxiliaries

```
\newdimen\csize\csize=3ex \newcount\cnt\catcode`'=13
\def'{\global\advance\cnt1 \the\cnt}
```

- Much alignment occurs in typesetting mathematics: aligned equations in display, matrices, and the complex commutative diagrams. This has been dealt with elsewhere, for example [13], [14], and [23].

- Crosswords: puzzle, clues and solution, [16]

1	■	2	
3	4		■
■		■	5
6			

Across
2 Switch mode
3 Knuth
6 Prior to TEX

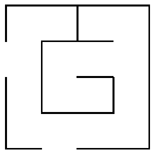
Down
1 Public domain
2 All right
4 All comes to it
5 Atari type

P	■	O	N
D	E	K	■
■	N	■	S
E	D	I	T

[1] As floating body, also called island by Spivak (1989).

[2] Jackie Damrau compiled a collection of table macros/styles resulting from the Portland workshop on the issue, which contains most of the works I mentioned above.

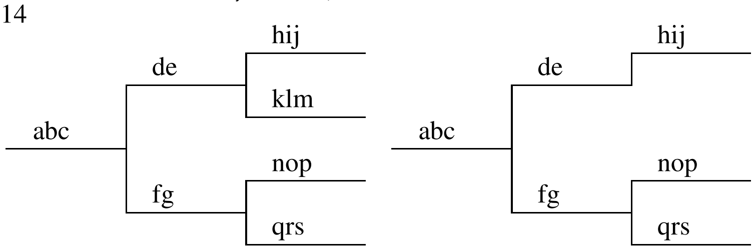
- Maze



via

```
\newdimen\csize \csize=3ex
\catcode`|=13 \def|{\vrule}
\catcode`_=13 \def_{\hrulefill}
\hbox{\vtop{\offinterlineskip\hrule
\halign{&\hbox to\csize
{\vrule height\csize width0pt#\hfil}\cr
         |&_&|_& \cr
          &|&_&  \cr
         |&|_&_&|\cr
         |_& &_&_\cr}%end \halign
 }\vrule%end \vbox
 }%end \hbox
```

- Chart, TEXbook ex22.14



via the following alternative code without `\halign`

```
\newdimen\dist \dist=2\baselineskip
\def\ent#1{\hbox{\vbox to0pt{\vss\hbox to10ex{\quad\strut#1\hss}\hrule}}}
\def\vl#1{\hbox{\vrule height#1\dist depth0pt}}
\def\bl#1{\kern#1\dist}%BLank of proper size
%Balanced
$$\offinterlineskip
 \vcenter{\ent{abc}}
 \vcenter{\ent{de} \vl2\ent{fg}}
 \vcenter{\ent{hij}\vl1\ent{klm}\bl1\ent{nop}\vl1\ent{qrs}}
\quad%Nearly balanced tree; pruning a branch
 \vcenter{\ent{abc}}
 \vcenter{\ent{de} \vl2\ent{fg}}
 \vcenter{\ent{hij}\vl{.5}%\bl{.5}\ent{klm}
                              \bl{1.5}\ent{nop}\vl1\ent{qrs}}$$
```

There exist some special tree packages, see [3], or consult Beebe's TUGlib server, especially the directory trees.bib.

- Row and/or column spans, partial horizontal rules[3]

Pair No	Contract		Re-sults	Scores		MP s
	N–S	E–W		N–S	E–W	
1.						
2.						
et cetera						

- Bridge pair match scheme, with special 'first' element

Spel / Ronde	1–3	4–6	7–9	10–12	13–15	16–18	19–21
1.	1: 1–2			2: 3–4		3: 5–6	4: 7–8
2.	4: 8–3	1: 1–6			2: 5–7		3: 2–4
. . .							

[3] How many rows does the header have? Descriptive 1, visual 2, and for plain TEX formatting 3? See Mark up of Bridge Form subsection.

- Alignment at decimal points. Generally this is done by introducing an extra column for the dot, and flushing right the digits left and flushing left the digits right of it. It comes from the wishes not to print non-significant leading zeroes, and to suppress zeroes behind the point for exact numbers. 3.5 means exactly $3\frac{1}{2}$. 3.500 means accurate to three digits. I would mark up the numbers separately and insert Knuth's '?', TEXbook, p.240–241, for non-signicant zeroes, automatically. The latter can also be done by programming the editor by a template consisting of sufficient ?-s followed by the decimal point. The required numbers can be brought in by overtyping the template. Related to alignment at the decimal point is alignment at number signs,[4] see the AAP table.
- Fill-in forms, especially the registration forms for the various TEX conferences, are captivating. Sometimes I ponder about the e-mail equivalents of the traditional snail forms, especially what and how they should be filled in. Redefine a list of empty definitions?
- Time-tables (railway, bus, . . .), and the use of rotated fonts.
- Nested, and interrupted[5] table with updating of memory, [14]

Puzzle	♠ KQ76	6NT,
	♡ J98	by East
	♢ J942	
	♣ 65	
♠ AJ3		♠ T9
♡ K653	N	♡ A2
♢ AK3	W E S	♢ T5
♣ AQT		♣ KJ9xxxx
	♠ 8542	
	♡ QT74	
	♢ Q876	
	♣ 2	

Trick					NS	EW
1	♡4!	♡K	♡8	♡2	–	1
2	♣A	♣5	♣x	♣2	–	2
3	♣Q	♣6	♣x	♠2	–	3
4	♣T	♡9	♣K	♠4	–	4
5	♣J	♠5	♠3	♠6	–	5
6	♣9	♠8	♡5	♠7	–	6
7	♣x	♢6	♠J	♢2	–	7

Puzzle	♠ KQ	NS squeezed on
	♡ J	♣ continuation?
	♢ J94	
	♣ –	
♠ A		♠ T9
♡ 63	N	♡ A
♢ AK3	W E S	♢ T5
♣ –		♣ ⓧ
	♠ –	
	♡ QT7	
	♢ Q87	
	♣ –	

8	♣x	♡7	♡6	♡J	–	8

et cetera

Tables which also require updating of memory occur with typesetting of chess, for example [24], or GO, [11].

---

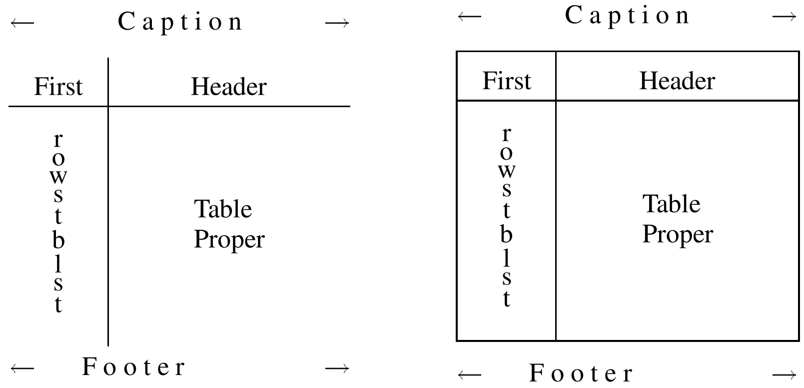
[4] Another column for the signs offends the structure. At a lower level one could think of `\scs`, a Sign Column Separator, but that is not nice either.

[5] From the user level one would say: aligned material connected by text, where the latter in the case at hand is the status of the play.

4 Bordered table

A bordered table consists of a caption, a header, row stubs, a table proper and a footer, all with appropriate separators. Special cases occur when some of the elements are absent. Abstraction from ruled and non-ruled is realized at the user level. Ruled can be specified either as horizontal, vertical or both. Dotted rules, also the partial ones, can be specified.[6] No variation in the line thicknesss has been encoded.[7] Framing is treated just as if a general document element has to be framed. I chose to leave the caption and the footer outside the frame. The positioning of the elements can be specified as centered, flushed left, respectively flushed right. A user may supply his own template. Caption, border (first, header, row stubs), and footer can be specified; all mutual orthogonal and independent from `\data`.

- Bordered table model



via

```
\def\caption{$\leftarrow$\hfill C a p t i o n\hfill$\rightarrow$}
\def\first{\hbox to6ex{\hss First\hss}}
\def\header{\hbox to18ex{\hss Header\hss}}
\def\rowstblst{{\hfil$\vcenter to20ex{\vss\offinterlineskip
            \fifo rowstblst\ofif\vss}$}}
\def\data{$\vcenter{\offinterlineskip\hbox{Table}\kern1ex\hbox{Proper}}$}
\def\footer{\hbox to\thsize
{$\leftarrow$\hss{\small F o o t e r}\hss\hss\hss$\rightarrow$}}
$$\vcenter{\btable\data}\qquad\qquad
  \vcenter{\framed\btable\data}$$
```

with, [17], [19]

```
\def\fifo#1{\ifx\ofif#1\ofif\fi\process{#1}\fifo}\def\ofif#1\fifo{\fi}
\def\process#1{\hbox to0pt{\hss#1\hss}\kern.5ex}
```

Rules can be specified via for example `\ruled`, `\hruled`, `\vruled`, `\nonruled`, respectively `\dotruled`.[8] The rules that separate the border can be eliminated, by redefinition of the macros `\headersep` and `\rowstbsep`. Furthermore, there are `\ctr`, `\fll`, and `\flr`, for controlling the positioning of the elements of the table. The framing can be controlled by `\framed`, `\dotframed`, or `\nonframed`.

4.1 Encoding

```
%btable.tex version 1, 17/7/92                          author: cgl@rug.nl
\newbox\tbl\let\ea=\expandafter
%Cell vertical size, row height and depth (separation implicit),
\newdimen\cvsize\newdimen\tsht\newdimen\tsdp\newdimen\tvsize\newdimen\thsize
%Parameter setting macros:   Rules
\def\hruled{\def\lineglue{\hrulefill}\def\colsep{}      \def\rowsep{\hrule}
   \let\rowstbsep=\colsep\let\headersep=\rowsep}
\def\vruled{\def\lineglue{\hfil}     \def\colsep{\vrule}\def\rowsep{}
   \let\rowstbsep=\colsep\let\headersep=\hrule}
\def\ruled {\def\lineglue{\hrulefill}\def\colsep{\vrule}\def\rowsep{\hrule}
   \let\rowstbsep=\colsep\let\headersep=\rowsep}
\def\nonruled{\def\lineglue{\hfil}   \def\colsep{}      \def\rowsep{}
   \def\rowstbsep{\vrule}\def\headersep{\hrule}}
\def\dotruled{\def\lineglue{\dotfill}\def\rowsep{\hbox to\thsize{\dotfill}}
```

---

[6] The chameleon sized `\hrule` and `\vrule`, don't have dotted analogues, however.

[7] It is not difficult to introduce a parameter `\linethickness`.

[8] The latter requires some extra work: table measurement via 2 passes.

```
\def\colsep{\lower1.5\tsdp\vbox to\cvsize{%
\leaders\hbox to0pt{\vrule height2pt depth2pt width0pt\hss.\hss}\vfil}}
\let\rowstbsep=\colsep\let\headersep=\rowsep}
%Parameter setting macros:   Controling positioning
\def\ctr{\def\lft{\hfil}\def\rgt{\hfil}}%Centered
\def\fll{\def\lft{}     \def\rgt{\hfil}}%Flushed left
\def\flr{\def\lft{\hfil}\def\rgt{}}     %Flushed right
%Parameter setting macros:   Framing
\def\framed{\let\frameit=\boxit}
\def\nonframed{\def\frameit##1{##1}}
\def\dotframed{\let\frameit=\dotboxit}
%
\def\btable#1{\vbox{\let\rsl=\rowstblst%Copy
\ifx\empty\template\ifx\empty\rowstblst
    \def\template{\colsepsurround\lft####\rgt&&\lft####\rgt\cr}
    \else\def\template{\colsepsurround####\hfil&&\lft####\rgt\cr}\fi
   \fi
\tsht=.775\cvsize\tsdp=.225\cvsize
\def\tstrut{\vrule height\tsht depth\tsdp width0pt}
%Logical mark up of column and row separators, via use of
\def\cs{&\colsepsurround\colsep\colsepsurround&}
\def\prs{&\colsepsurround\lineglue&}   \def\srp{&\lineglue\colsepsurround&}
\def\rs{\colsepsurround\tstrut\cr
        \ifx\empty\rowsep\else\noalign{\rowsep}\fi
        \ifx\empty\rowstblst\else\ea\nxtrs\fi}
\def\grs{\colsepsurround\tstrut\cr\ghostrow}
\def\rss{&\colsepsurround\rowstbsep\colsepsurround&}
\def\hs{\colsepsurround\tstrut\cr
       \ifx\empty\headersep\else\noalign{\headersep}\fi
       \ifx\empty\rowstblst\else\ea\nxtrs\fi}
\preinsert                                                      %User action
\setbox\tbl=\vbox{\tabskip=0pt\relax\offinterlineskip
\halign{\span\template\ifx\empty\first\ifx\empty\rowstblst\else
\ifx\empty\header\else\ea\rss\fi\fi\else\first\ea\rss\fi
\ifx\empty\header\ifx\empty\first\if\empty\rsl\else\ea\nxtrs\fi
                 \else\ea\hs\fi
\else\header\ea\hs\fi
#1\colsepsurround\tstrut\crcr}    }                              %end \setbox
\postinsert%Pick up what is needed, \thsize,...
\ifx\caption\empty\else\hbox to\thsize{\strut\hfil\caption\hss}\captionsep\fi
\frameit{\copy\tbl}
\ifx\footer\empty\else\footersep\hbox{\vtop{\noindent\hsize=\thsize%
\footer}}\fi                 }}                                 %end \btable
%Defaults
\cvsize=4ex\tsht=.775\cvsize\tsdp=.225\cvsize\def\colsepsurround{\kern.5em}
\def\caption{}\def\first{}\def\header{}\def\rowstblst{}\def\footer{}\def\data{}
\def\captionsep{\medskip}    \def\headersep{\hrule}
\def\footersep{\smallskip}   \def\rowstbsep{\vrule}
\def\preinsert{}
\def\postinsert{\global\thsize=\wd\tbl
                \global\tvsize=\ht\tbl\global\advance\tvsize by\dp\tbl}
\ctr\nonruled\nonframed\def\template{}\def\ghostrow{}           %end Defaults
%Auxiliaries
\def\boxit#1{\vbox{\hrule\hbox{\vrule\vbox{#1}\vrule}\hrule}}
\def\dotboxit#1{\vbox{\offinterlineskip\hbox to\thsize{\dotfill}%
\hbox{\lower\tsdp\vbox to\tvsize{%
\leaders\hbox to0pt{\hss\vrule height2pt depth2pt width0pt.\hss}\vfil}%
\vbox{#1}\lower\tsdp\vbox to\tvsize{%
\leaders\hbox to0pt{\hss\vrule height2pt depth2pt width0pt.\hss}\vfil}}%
\hbox to\thsize{\dotfill}}}
%And to account for logical columns with \logmsp
\def\spicspan{\span\omit}
\def\logmsp#1{\omit\mscount=#1\multiply\mscount by2 \advance\mscount by-1
\loop\ifnum\mscount>1 \spicspan\advance\mscount by-1 \repeat}
```

```
%To handle the row stub list: \rsl
\def\nxtrs{\ifx\empty\rsl\else\def\nxtel{\ea\nrs\rsl\srn}\ea\nxtel\fi}
\def\nrs#1#2\srn{\gdef\rsl{#2}#1\rss}                     %end btable.tex
```

`\btable` Implements the bordered table model. Note that the table is measured in order to set the width of the header and the footer. The measurement is also useful for dotted horizontal lines. When no template is provided, then a default will be created, automatically.

\\$\langle X\rangle$`sep` Parameters which govern the kind of separators.

`\nxtrs` Yields the next element of the row stub list. *The last element of the list has to be enclosed by an extra pair of braces.* I could have introduced an explicit list terminator instead, but that would have created incompatibilities with the termination in other defs. Weird?

`\cvsize, \tsht, \tsdp, \thsize, \tvsize` These dimension variables stand for vertical cell size, table strut height, table strut depth, table horizontal size, and table vertical size.

`\preinsert, \postinsert` For flexibility. Default `\preinsert` is empty. `\postinsert` delivers by default the table sizes globally into `\thsize`, and `\tvsize`.

4.2 Examples of use

Before the invocation of `\btable\data` do:
Redefine `\data` with column and row separators, `\cs`, respectively `\rs`.
Choose from the 'attributes'
– `\ruled`, `\hruled`, `\vruled`, `\dotruled` Default: `\nonruled`
– `\framed`, `\dotframed` Default: `\nonframed`
– `\fll`, `\flr` Default: `\ctr`
Supply the caption, header, first element, row stubs, or footer via redefinitions of the corresponding macros.

- Just data, [14]

2	7	6
9	5	1
4	3	8

16	3	2	13
5	10	11	8
9	6	7	12
4	**15**	**14**	1

via

```
$$\def\data{2\cs7\cs6\rs 9\cs5\cs1\rs  4\cs3\cs8}
  \vcenter{\framed\ruled\btable\data}\qquad\qquad\qquad
  \def\data{16\cs    3\cs    2\cs13\rs
             5\cs   10\cs   11\cs 8\rs
             9\cs    6\cs    7\cs12\rs
             4\cs\bf15\cs\bf14\cs 1}
\ruled\framed\setbox0=\btable\data%for measuring sizes
\vcenter{\dotruled\dotframed\btable\data}\qquad$$
```

- Data plus header, TEXbook p.246

Year	World Population
8000 B.C.	5,000,000
50 A.D.	200,000,000

Year	World Population
8000 B.C.	5,000,000
50 A.D.	200,000,000

via the simpler nearly descriptive mark up

```
\def\header{\hfill Year\hfill\cs\hfill World Population\hfill}
\def\data{8000 B.C.\cs  5,000,000\rs 50 A.D.\cs200,000,000}
$$\flr\vruled\vcenter{\btable\data}\qquad
\ruled\framed\vcenter{\btable\data}$$
```

- Data with row stubs, TEXbook p.232

Horizontal lists	Chapter 14
Vertical lists	Chapter 15
Math lists	Chapter 17

via

```
\def\rowstblst{{Horizontal lists}
  {Vertical lists}{{Math lists}}}
\def\rowstbsep{}%Default a \vrule
\def\data{Chapter 14\rs
Chapter 15\rs Chapter 17}
$$\btable\data$$
```

- Data, row stubs, header, caption and footer

11	12
21	22

	Header	
1^{st} row	11	12
2^{nd} row	21	22

Caption

	Header	
1^{st} row	11	12
2^{nd} row	21	22

Footer

Caption

	Header	
1^{st} row	11	12
2^{nd} row	21	22

Footer

after data definition, `\def\data{11\cs12\rs21\cs22}`, via[9]

```
$$\vcenter{\framed\btable\data}      \qquad
  \def\header{\logmsp2\hfill Header\hfill}
  \def\rowstblst{{$1^{st}$ row}{{$2^{nd}$ row}}}
  \vcenter{\btable\data}             \qquad
  \def\caption{Caption}\def\footer{Footer}
  \vcenter{\dotruled\btable\data}    \qquad
  \vcenter{\ruled\framed\btable\data}$$
```

- Walter's spreadsheet, TEXbook, p.244. No preamble has to be defined. It makes ex22.10 superfluous. How about that?

1	Adjusted gross income		$4,000
2	Zero bracket amount for		
	a single individual	$2,300	
3	Earned income	1,500	
4	Substract line 3 from line 2		800
5	Add lines 1 and 4. Enter here		
	and on Form 1040, line 35		$4,800

The above is encoded, as a 4-column table, via

```
\def\data{1\cs\logmsp2Adjusted gross income     \dotfill\cs\$4,000\rs
         2\cs Zero bracket amount for                            \rs
          \cs a single individual \dotfill\cs\hfill\$2,300\cs    \rs
         3\cs Earned income       \dotfill\cs
                                   \hfill\underbar{ 1,500}\cs    \rs
         4\cs\logmsp2Substract line 3 from line 2
                                  \dotfill\cs\hfill\underbar{ 800}\rs
         5\cs Add lines 1 and 4. Enter here                      \rs
          \cs\logmsp2and on Form 1040, line 35 \dotfill\cs\$4,800}
\def\colsepsurround{\kern.5ex}\cvsize=3ex
$$\fll\btable\data$$
```

[9] Note the invariance of the `\data` specification.

- Part of AAP's table, [1], or [13]

AAP's table: Job Changes 1973–1980

	Gain/Loss of hospitals since 1973	Total No of CEO Job Changes 1973–90	Survival Rate of CEO's
Texas	+20	—	22%
Maryland	+ 5	42	24%

Source: David Kinzer, 'Turnover Of Hospital Chief Executive Officers: A Hospital Association Perspective,' Hospital and health Services Administration May-June 1982.

obtained via

```
\def\caption{AAP's table:\qquad Job Changes 1973--1980\hfill}
\def\rowstblst{{Texas}{{Maryland}}}
\def\header{\copy1\cs\copy2\cs\copy3}
\def\data{$+20$\cs---\cs22\%\rs
         $+?5$\cs 42\cs24\%}
\def\footer{Source: David Kinzer, 'Turnover Of Hospital Chief
Executive Officers: A Hospital Association Perspective,'
Hospital and health Services Administration May-June 1982.}
$$\btable\data$$
```

with auxiliaries

```
\def\nl{\hfil\strut\break\strut}\catcode`?=\active \def?{\kern1.1ex}
\setbox1=\vtop{\noindent\hsize14ex
    Gain/Loss of\nl hospitals\nl since 1973}
\setbox2=\vtop{\noindent\hsize13ex
    Total No\nl of CEO Job\nl Changes\nl 1973--90}
\setbox3=\vtop{\noindent\hsize10ex
    Survival\nl Rate of\nl CEO's}
```

- AT&T table diversions, TEXbook p.247, [20]

```
\def\caption{AT\&T Common Stock}
\def\header{Year\cs Price\cs Dividend}
\catcode`?=\active \def?{\kern1.1ex}
\def\data{1971\cs41--54\cs\llap{\$}2.60\rs
             2\cs41--54\cs         2.70\rs
             3\cs46--55\cs         2.87\rs
             4\cs40--53\cs         3.24\rs
             5\cs45--52\cs         3.40\rs
             6\cs51--59\cs         ?.95\rlap*}
\def\footer{* (first quarter only)}
$$\framed\vcenter{\vbox{\small\btable\data}}\qquad\qquad
\def\caption{}\def\header{\logmsp3 \hfil AT\&T Common Stock\hfil\rs
            \hfill Year\hfill\cs\hfill Price\hfill\cs \hfill Dividend\hfill}
\vcenter{\flr\ruled\btable\data}$$
```

yields

AT&T Common Stock

Year	Price	Dividend
1971	41–54	$2.60
2	41–54	2.70
3	46–55	2.87
4	40–53	3.24
5	45–52	3.40
6	51–59	.95*

* (first quarter only)

AT&T Common Stock		
Year	Price	Dividend
1971	41–54	$2.60
2	41–54	2.70
3	46–55	2.87
4	40–53	3.24
5	45–52	3.40
6	51–59	.95*

* (first quarter only)

It remains a matter of choice what should be considered as header and what as caption. A footnote facility, which will append the footnote to a footnote list similar to the row stub list, could have been implemented. I refrained from implementing it, for the moment.

- Pittman's, [22], deterministic multiplication table. Typographically and in other programming languages a trifle. In TEX the encoding matters.

1	2	3
2	4	6

×	1	2	3
1	1	2	3
2	2	4	6

×	1	2	3	4
1	1	2	3	4
2	2	4	6	8
3	3	6	9	12

via

```
\let\nx=\noexpand \let\ea=\expandafter
\newcount\rcnt \newcount\ccnt \newcount\tnum
\newcount\mrow \newcount\mcol
\rcnt1 \ccnt1 \mrow2 \mcol3
\def\rows{\global\ccnt1 \cols \global\advance\rcnt1
   \ifnum\rcnt>\mrow\swor\fi\rs\rows}
\def\swor#1\rows{\fi}
\def\cols{\tnum\rcnt \multiply\tnum\ccnt \the\tnum
   \global\advance\ccnt1 \ifnum\ccnt>\mcol\sloc\fi\cs\cols}
\def\sloc#1\cols{\fi}
\vcenter{\btable\rows}\qquad\qquad
%
\global\ccnt=1 \global\rcnt=1 \mrow2 \mcol3
\def\first{$\times$}  \def\header{\row}
\def\row{\the\ccnt\global\advance\ccnt1
   \ifnum\ccnt>\mcol\wor\fi\cs\row}
\def\wor#1\row{\fi}
\def\rowstblst{\ifnum\rcnt=\mrow\gdef\rowstblst{}\fi}
\def\nxtrs{\the\rcnt\rss}
\vcenter{\btable\rows}\qquad\qquad
%
\global\ccnt=1 \global\rcnt=1 \mrow3 \mcol4
\def\rowstblst{\ifnum\rcnt=\mrow\gdef\rowstblst{}\fi}
\framed\vcenter{\btable\rows}$$
```

4.3 No mark up

For simple tables the data can be specified without `\cs`-s, and `\rs`-s, that is without explicit markers for column and row separators. These can be inserted by TEX, see TEXbook p.249, [17], or [22]. For special cases, for example crosswords, [16], this is handy.
Pondering about lists, make you realize that the table proper data forms a nested list with `\rs` and `\cs`, respectively first level and second level separators. One could also adopt $\langle cr\rangle$ and ␣ for that, in other words allow natural data provision.

5 Simultaneous row and column spans

When one kind of spans is the issue one can use `\logmsp`—for logical columns, as an extension to `\multispan`— either within `\halign`, for column spans, or within `\valign`, for row spans. Simultaneous row and column spans require more work.
First, we have to account for the blocks proper. This is done via leaving space open, overprinting the left open space, and the use of a *Ghost Row Separator*. For a block of even *row* size, I chose to insert lines of height zero. I call these ghost lines. For a block with an odd number of rows, I chose to modify the middle line.
Next, when we are dealing with *ruled* tables we have to hide the rules that should otherwise traverse the block. This can be done for partial vertical rules via `\logmsp{n}\hfil`, with n an integer. For partial horizontal rules there are two aspects. First, the table-wide rules are not typeset, and second, partial horizontal rules have to be typeset around the blocks. I supplied the macros `\prs`, Partial Rule Separator, and `\srp`, *mirror* Separator Rule Partial, to enclose a partial rule. The kind of partial rule is determined by the replacement text of `\lineglue`.

Row spans in the row stub list can be handled similarly. For even spans define an appropriate `\ghostrow`, with a copy of the block in it. For odd spans provide a copy of the block in the row stub list. Provide suitable empty row stub elements, { }, in both cases.

5.1 Partial rules

For a horizontal rule over one (logical) column use
`...\prs\lineglue\srp...` instead of
`...\cs...\cs....`
`\lineglue` is defined by the $\langle X\rangle$`ruled` macros.

The `\prs`, Partial Rule Separator, accounts for that part of the rule, which extends into the hidden column for the separator. `\srp` is its mirror terminator. At the line-ends of the table one can simply use `\cr`, as mirror terminator. The respective (symmetric) definitions are

```
\def\prs{&\colsepsurround\lineglue&}   \def\srp{&\lineglue\colsepsurround&}
```

For a horizontal rule over $\langle n\rangle$ (logical) columns use

```
...\prs\logmsp{n}\lineglue\srp...%n logical columns
```

5.2 Mark up bridge form

The header row of the bridge form can be characterized by: one 3-row span, two 2-column spans, and two 2-column partial horizontal rules. Furthermore, the repetition of the number of rows is interesting.

```
\def\header{\cs\logmsp2\hfil Contract\hfil\cs
                          \cs\logmsp2\hfil Scores\hfil\cs\grs
            \cs N--S\cs E--W\cs\cs N--S\cs E--W\cs}
\def\data{\lines}   \framed\ruled\btable\data
```

with auxiliaries

```
\newcount\bcnt
\def\lines{\global\advance\bcnt by 1 \ifnum\bcnt=3 et cetera\hidewidth
   \senil\fi\the\bcnt.\cs\cs\cs\cs\cs\cs\rs\lines}
\def\senil#1\lines{\fi}
%
\def\ghostrow{\omit\colsepsurround%
  \vbox to 0pt{\vss\hbox to 5ex{\hss Pair\hss}\vskip.5ex
          \hbox to 5ex{\hss No\hss}\vss}\prs\logmsp2\lineglue\srp
  \vbox to 0pt{\vss\hbox to 5ex{\hss Re-\hss}\vskip.5ex
        \hbox to5ex{\hss sults\hss}\vss}\prs\logmsp2\lineglue\srp
  \vtop to 0pt{\vss\hbox to 5ex{\hss MP$\,$s\hss}\vss}\colsepsurround\cr}
```

The row span extends implicitly into the first and third row. The last row of the data does not take vertical rules, automatically.[10]

[10] For a real form—24 or more rows—just modify `\lines`.

5.3 Blocks; connected cells

In order to specify spans of *logical* columns, we need an adapted version of `\multispan`, TEXbook p.354, as given in the Encoding subsection.
Spans form a block. Around the blocks partial rules emerge.

- **Odd blocks**, 1-by-2 etc. Use `\logmsp`, `\srp`, `\lineglue` and the like. Define appropriate -s.

$1*2$		13
21	22	23
31	32	33

A $\{$		13
		23
		33
41	42	43

Caption

	Header $\Downarrow$			
row 1	$3*3$			14
row 2				24
row 3				34
$\Rightarrow$	41	42	43	44

Footer

can be obtained via

```
$$\setbox\block=\hbox{$1*2$}
\def\data{\logmsp2\colsepsurround\hfil\copy\block\hfil\cs13\rs
21\cs22\cs23\rs
31\cs32\cs33}                                               %end \data
{\ruled\setbox0=\btable\data}%For measurement
\vcenter{\dotruled\btable\data}\qquad\qquad
%
%3-by-2, and connected cells
\def\data{\logmsp2\hfil\cs13\grs
\logmsp2\colsepsurround\hfil\vbox to0pt{\vss
                                     \copy\block\vss}\hfil\cs23\grs
          \logmsp2\hfil\cs33\rs
                   41\cs42\cs43}                            %end \data
\setbox\block=\hbox{{\large A}\enspace%
             $\left\{\vrule height3ex depth3ex width0pt\right.$}
\def\ghostrow{\ifx\empty\rowstblst\else\omit\lineglue\ea\srp\fi
\logmsp2\hfil\prs\lineglue\cr
\ifx\empty\rowstblst\else\ea\nxtrs\fi}
\vcenter{\ruled\btable\data}\qquad\qquad
%
%3-by-3 block and all around
\def\data{\logmsp3\hfil\cs14\grs
\logmsp3\hfil\vbox to 0pt{\vss\copy\block\vss}\hfil\cs24\grs
          \logmsp3\hfil\cs34\rs
              41\cs42\cs43\cs44}                            %end \data
\setbox\block=\hbox{\Large$3*3$}
\def\ghostrow{\ifx\empty\rowstblst\else\omit\lineglue\ea\srp\fi
\logmsp3\hfil\prs\lineglue\cr\ifx\empty\rowstblst\else\ea\nxtrs\fi}
\def\caption{{\Large Caption}}
\def\header{\logmsp4 Header\hfil$\Downarrow$\ }
\def\rowstblst{{row 1}{row 2}{row 3}{{\hfil$\Rightarrow$}}}
\def\footer{Footer}
\vcenter{\ruled\framed\btable\data}$$
```

- **Even blocks**, 2-by-1 etc. Leave the cells for the block open in `\data`. Use `\grs`, `\prs`, `\lineglue`, and the like. Define `\ghostrow`.

A	12	13
	22	23
31	32	33

$2*2$		13
		23
31	32	33

Caption

	Header ⇓			
row 1	$2*3$			14
row 2				24
⇒	31	32	33	34

Footer

can be obtained via

```
$$\setbox\block=\hbox{\large A}%2-by-1 block
\def\data{\cs12\cs13\grs
          \cs22\cs23\rs
        31\cs32\cs33}                                          %end \data
\def\ghostrow{\ifx\empty\rowstblst\else\omit\lineglue\ea\srp\fi%
\hfil\vbox to0pt{\vss\copy\block\vss}\hfil\prs
   \logmsp2\lineglue\cr\nxtrs}
{\ruled\framed\setbox0=\btable\data}%for measurement
\vcenter{\dotruled\btable\data} \qquad\qquad
%
\setbox\block=\hbox{\large$2*2$}
\def\data{\logmsp2\hfil\cs13\grs
          \logmsp2\hfil\cs23\rs
                   31\cs32\cs33}                               %end \data
\def\ghostrow{\ifx\empty\rowstblst\else\omit\lineglue\ea\srp\fi%
   \logmsp2\colsepsurround\hfil%
   \vbox to0pt{\copy\block\vss}\hfil\prs\lineglue\cr\nxtrs}
\vcenter{\framed\btable\data}    \qquad\qquad
%
\setbox\block=\hbox{\large$2*3$}
\def\data{\logmsp3\cs14\grs
          \logmsp3\cs24\rs
         31\cs32\cs33\cs34}                                     %end \data
\def\caption{{\Large Caption}}
\def\rowstblst{{row 1}{row 2}{{\hfil$\Rightarrow$}}}
\def\header{\logmsp4 Header\hfil$\Downarrow$\ }
\def\footer{Footer}
\def\ghostrow{\ifx\empty\rowstblst\else\omit\lineglue\ea\srp\fi
\logmsp3\hfil%
\vbox to0pt{\vss\copy\block\vss}\hfil\prs\lineglue\cr\nxtrs}
\vcenter{\framed\ruled\btable\data}$$
```

- **Multiple blocks**: in row stub and table proper

A	12	13
	22	23
B	32	33

Caption

	Header	
A	12	13
	22	23
B	32	33

Footer

via

```
$$\setbox\block=\vbox{\hbox{\large A}\kern3ex\hbox{\large B}}
\def\data{                           \cs12\cs13\grs
$\vcenter to0pt{\vss\copy\block\vss}$\cs22\cs23\grs
                                     \cs32\cs33\lr}
\vcenter{\btable\data}\qquad\qquad\qquad
```

```
\def\caption{{\Large Caption}}
\def\header{\logmsp3\hfill Header\hfill}
\def\rowstblst{{}{$\vcenter to0pt{\vss\Bigg\{\vss}$}{{}}} $$
\def\footer{Footer}
\vcenter{\ruled\btable\data}
```

with

```
\def\ghostrow{\rss\prs\logmsp2\lineglue\cr\nxtrs}
\def\lr{\tstrut\colsepsurround\cr\prs\logmsp3\lineglue\gobble}
\def\gobble#1#2{}%kludge to undo \tstrut, \colsepsurround
\def\preinsert{\def\rss{&\colsepsurround\hfil\colsepsurround&}
   \def\hs{\colsepsurround\tstrut\cr\prs\logmsp3\lineglue\cr\nxtrs}}
```

Not much is gained in clarity for complicated tables. The logical structure at the top level is there, however.

6 Conclusions

Tables are diverse, and a variety of tools is needed.

Some (deterministic) tables can be generated completely by computer, no detailed user mark up is needed, just the invocation of the macro.

Mark up of bordered scientific tables can be done via the provided bordered table macro `\btable`, in the SGML spirit.

Complex tables need detailed formatting commands and TEXnowledge, in agreement with DEK, TEXbook p.245

> 'People who know how to make ruled tables are generally known as TEX masters. Are you ready?'

Hard things: not to introduce parameters which are already available, and to avoid redundancy. However, I could not get around `\colsepsurround`, because of the impossibility to control the kind of glue inserted by `\tabskip`.
The lack of dotted equivalents of `\hrule`, and `\vrule`, as TEX primitives, is regrettable.

Typesetting tables requires a discipline to be adhered to. What about a discipline of TEXing?

References

[1] ASSOCIATION OF AMERICAN PUBLISHERS, "Markup of Tabular Material", Version 2.0 AAP Inc., 1989.

[2] Asher, "Inside Type & Set", TUGboat, 13.1, 1992, pp. 13–22.

[3] Anne BRÜGGEMAN-KLEIN, D. WOOD, "Drawing Trees nicely with TEX", EP-ODD, 2, 2, 1989, pp.101–115.

[4] David CARLISLE, "longtable.sty", from the fileserver, 1992.

[5] Ray F. COWAN, "tables.tex", from the fileserver, 1985. 7p.

[6] Amy HENDRICKSON, *MacroTEX*, version 0, 1989.

[7] Theo A. JURRIENS, "supertabular.sty", from the fileserver, 1989.

[8] Khanh HA, "Easy Table", TUGboat, 11.2, 1990, pp. 250–264.

[9] Donald E. KNUTH, *The Art of Computer Programming, 3. Sorting and Searching*, Addison-Wesley, Reading Mass., 1973.

[10] Donald E. KNUTH, *The TEXbook*, Addison-Wesley, Reading Mass., 1984.

[11] Hanna KOŁODZIEJSKA, "Go diagrams with TEX", MAPS 91.2, 1991, pp. 63–68.

[12] Kees VAN DER LAAN, "Typesetting Bridge via TEX", TUGboat, 11.2, 1990, pp. 265–276. (An earlier LATEX set-up appeared in TUGboat, 10.1, 1989, pp. 113–116.)

[13] Kees VAN DER LAAN, "SGML(, TEX and ...)", TUGboat, 12.1, 1991, pp. 90–104. Proceedings EuroTEX 90. Also MAPS 90.1.

[14] Kees VAN DER LAAN, "Math into BLUes, Part I", TUGboat, 12.4, 1991, pp. 485–501. (Part II has appeared in GUTenberg Cahiers 10&11, Proceedings EuroTEX 91.) Also MAPS 90.2.

[15] Kees VAN DER LAAN, "Tower of Hanoi, revisited", TUGboat, 13.1, 1992, pp. 91–94. Also MAPS 92.1, pp. 125–127.

[16] Kees VAN DER LAAN, "Typsetting Crosswords via TEX", This proceedings. Also MAPS 92.1, pp. 128–131.

[17] Kees VAN DER LAAN, "FIFO and LIFO incognito", This proceedings. Also MAPS, 92.1, pp. 121–124.

[18] Kees VAN DER LAAN, "FIFO and LIFO sing the BLUes", MAPS, 92.2.

[19] Leslie LAMPORT, *LATEX user's guide and reference manual*, Addison-Wesley, Reading, Mass., 1986.

[20] M. E. LESK, "Tbl – a program to format tables", UNIX manual, pp. 157–174, 1979.

[21] J.E PITTMAN, "Loopy.TEX", TUGboat, 9.3, 1988, pp. 289–291.

[22] David SALOMON, "Advanced TEX course: Insights & Hindsights," MAPS Special, 1992, 252p.

[23] Michael D. SPIVAK, *LAmSTEX—The Synthesis*. TEXplorators.

[24] Piet TUTELAERS, "A font and a style for typesetting chess using LATEX or TEX", TUGboat, 13.1, 1992, pp. 85–90.

[25] Paul E. S. WORMER, "Young tableaux", from the listserver tex-nl@nic.surfnet.nl.

Syntactic Sugar

Kees van der Laan

Hunzeweg 57,
9893 PB Garnwerd, The Netherlands
cgl@rug.nl

September 1992

Abstract

A plea is made for being honest with TEX and not imposing alien structures upon it, otherwise than via compatible extensions, or via (non-TEX) user interfaces to suit the publisher, the author, or the typist. This will facilitate the process to get (complex) publications out effectively, and typographically of high-quality.

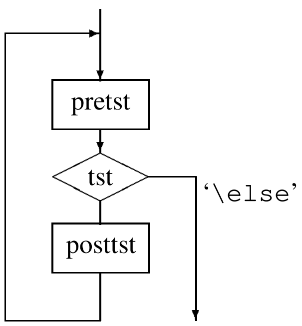
Keywords: (nested) loop, switch, array addressing, keyword and optional parameters, linear search, sorting, plain TEX, macro writing, education.

1 Introduction

TEX is a formatter and also a programming language. TEX is different from current high-level programming languages, but very powerful. A class on its own, and therefore unusual, and unfamiliar.
Because of TEX being different, macro writers propose to harnass TEX into a more familiar system, by imposing syntaxes borrowed from various successful high-level programming languages. In doing so, injustice to TEX's nature might result, and users might become intimidated, because of the difficult—at least unusual—encoding used to achieve the aim. The more so when functional equivalents are already there, although perhaps hidden, and not tagged by familiar names. This is demonstrated with examples about the loop, the switch, array addressing, optional and keyword parameters.
Furthermore, TEX encodings are sometimes peculiar, different from the familiar algorithms, possibly because macro writers are captivated by the mouth processing capabilities of TEX. Users who don't care so much about TEX's programming power, but who are attracted by the typesetting quality, which can be obtained with TEX as formatter, can be led astray when in search for a particular functionality they stumble upon unusual encodings. They might conclude that TEX is too difficult, too error-prone and more things like that and flee towards Word$whatever$, or embrace Desk Top Publishing systems.
The way out is education, next to the provision of compatible, well-documented and supported user interfaces, which don't act like syntactic sugar, by neglecting, or hiding, the already available functional equivalents. Neither the publication of encodings, nor the provision of encodings via file servers or archives—although a nice supporting feature for the TEX-ies—is enough. The quality, compatibility and the simplicity of the (generic) macros should be warranted too.
It is not the aim of this paper to revitalize a programming languages notation war, but to stimulate awareness, and exchange ideas.

2 Loops

DEK's loop, TEXbook p.219, implements the flow



with syntax[1]

`\loop`$\langle pretst\rangle$`\if...`$\langle posttst\rangle$`\repeat.`

Special cases result when either $\langle pretst\rangle$, or $\langle posttst\rangle$ is empty. The former is equivalent to for example PASCAL's **while**. . . **do** . . . , and the latter to **repeat**. . . **until**. With this awareness, I consider the variants as proposed by for example Pittman, [22], and Spivak, [26], as syntactic sugar.
If `\ifcase...` is used, then we have for $\langle posttst\rangle$ several parallel paths, of which one—determined

[1] No (user) '`\else`' is allowed in here, because it is already used in '`\iterate`,' TEXbook p219.

dynamically—will be traversed. Provide and choose your path! What do you mean by traversing the `\else`-path?

2.1 Why another loop?

Kabelschacht, [12], and Spivak, [26], favour a loop which allows the use of `\else`.[2] I have some objections to Kabelschacht's claim that his loop is a *generalization* of plain's loop.
First, it is not a generalization, just a clever, but variant, implementation of the loop flow chart.
Second, it is not compatible with plain's loop. His exit path is via the `\then` branch (or via any of the `\or`-s, when `\ifcase` is used), and not via the `\else` branch.
The reason, I can think of, for introducing another loop, while the most general form has been implemented already, is the existence of commands like `\ifvoid`, and `\ifeof`, and the absence of their negatives `\ifnonvoid`, respectively `\ifnoneof`. In those cases we like to continue the loop via the `\else` branch. For the latter case this means to continue the loop when the file is *not* ended. This can be attained via modifying the loop, of course, but I consider it simpler to use a `\newif` parameter, better known as 'boolean' or 'logical' in other programming languages. With the `\newif` parameter, `\ifneof`, the loop test for an end of file—functionally ¬`\ifeof`—can be obtained via

```
\ifeof\neoffalse\else\neoftrue\fi\ifneof
```

For an example of use, see the Sort It Out subsection. Related to the above encoding of the logical ¬, are the encodings of the logical and, ∧, and or, ∨, via

Functional code	TEX encoding
`¬\if...`	`\if...\notfalse\else`
	`\nottrue\fi\ifnot`
`\if...∧\if...`	`\andtrue\if...\if...`
	`\else\andfalse`
	`\else\andfalse\fi\fi`
	`\ifand`
`\if...∨\if...`	`\ortrue`
	`\if...\else\if...\else`
	`\orfalse\fi\fi   \ifor`

with the `\newif`-s: `\ifnot`, `\ifand`, and `\ifor`.

2.2 Nesting of loops

Pittman, [22], argued that there is a need for other loop encodings.

> 'Recently, I encountered an application that required a set of nested loops and local-only assignments and definitions. TEX's `\loop...\repeat` construction proved to be inadequate because of the requirement that the inner loop be grouped.'

If we take his (multiplication) table—I like to classify these as deterministic tables, because the data as such are not typed in—to be representative, then below a variant encoding is given, which does not need Pittman's double looping. The table is typographically a trifle, but it is all about how the deterministic data are encoded. My approach is to consider it primarily as a table, which it is after all. Within the table the rows and columns are generated, via recursion, and not via the `\loop`. Furthermore, I prefer to treat rules, a frame, a header and row stubs as separate items to be added to the table proper, [17]. The *creation* of local quantities is a general TEX aspect. I too like the idea of a hidden counter, and the next best TEX solution via the local counter. The local versus global creation of counters is a matter of taste, although very convenient now and then. The creation of local quantities is tacitly discouraged by DEK's implementation, because there is no explicit garbage collector implemented and therefore no memory savings can be gained. The only thing that remains is protection against programming mistakes, which is indeed important.
Pittman's table, focused at the essential issue of generating the elements, can be obtained via

```
$$\vbox{\halign{&\ \hfil#\hfil\strut\cr
\rows}}$$
```

with

```
\newcount\rcnt\newcount\ccnt\newcount\tnum
\newcount\mrow\newcount\mcol \mrow2 \mcol3
\def\rows{\global\advance\rcnt1
   \global\ccnt0 \cols\ifnum\rcnt=\mrow\swor
   \fi\rs\rows}
\def\swor#1\rows{\fi\crcr}
\def\cols{\global\advance\ccnt1
   \tnum\rcnt \multiply\tnum\ccnt \the\tnum
   \ifnum\ccnt=\mcol\sloc\fi\cs\cols}
\def\sloc#1\cols{\fi}
\def\rs{\cr}\def\cs{&}
```

The result is

1	2	3
2	4	6

The termination of the recursion is unusual. It is similar to the mechanism used on p.379 of the TEXbook, in the macro `\deleterightmost`. The latter TEXnique is elaborated in [6], and [19].

The above shows how to generate in TEX deterministic tables, where the table entries in other programming languages are generally generated via nested loops. One can apply this to other deterministic math tables—trigonometric tables for example— but then we need more advanced arithmetic facilities in TEX (or inputting the data calculated by other tools), not to mention the

[2] Their loops are equivalent to the general form of the loop with the execution of an extra part after the loop.

appropriate mapping of tables which extend the page boundaries.

For a more complete encoding see Table Diversions, [17]. The idea is that rules and a frame be commanded via `\ruled`, and `\framed`. The header via an appropriate definition of `\header`, $\times$, the indication that we deal with a multiplication table, in `\first`, and the row stubs via definition of the row stub list. All independent and separate from the table proper part.

2.3 Loops and novices

Novice TEXies find DEK's loop unusual, so they sugar it into the more familiar **while**, **repeat**, or **for** constructs, encouraged to do so by exercises as part of courseware. From the functionality viewpoint, there is no need for another loop notation.

With respect to the **for** loop, I personally like the idea of a hidden counter, [15], and [22]. The hidden counter has been used in an *additional* way to plain's loop in for example [15] (via `\preloop` and `\postloop`), and will not be repeated here. This way of doing is a matter of taste, which does not harm, nor hinder, because it is a compatible extension.

And, ...for the nesting of loops we need scope braces, because of the parameter separator `\repeat`. If braces are omitted, the first `\repeat` is mistaken for the outer one, with the result that the text of the outer loop will *not* become the first `\body`. The good way is, to make the inner `\repeat` invisible at the first loop level, by enclosing the inner loop in braces.

The point I like to get accross is, that there is no real need for another loop encoding. Syntactic sugar, yes.

3 Switches, is there a need?

Apart from the `\ifcase...` construct, TEX seems to lack a multiple branching facility with symbolic names. Fine, [6], introduced therefore

```
\def\fruit#1{\switch\if#1\is
a \apple
b \banana
c \cherry
d \date       \end}
```

I have 2, or rather 3, remarks to the above.
First, the 'switch'-functionality is already there.
Second, Fine's implementation is based upon

> 'It is clear that `\switch` must go through the alternatives one after another, reproducing the test...'

Well, ...going through the alternatives one after another is not necessary.
Third, his example, borrowed from Schwarz, [24], can be solved more elegantly without using a 'switch' or nested `\if`-s at all, as shown below.
The first two aspects are related. Fine's functionality can be obtained via

```
\def\fruit#1{\csname fruit#1\endcsname}
%with
\def\fruita{\apple}
\def\fruitb{\banana} %et cetera
```

With for example `\def\apple{{\bf apple}}`, `\fruit a` yields **apple**.

And what about the 'else' part? Thanks to `\csname`, `\relax` will return when the control sequence has not yet been defined. So, if nothing has to happen we are fine. In the other situations one could define `\def\fruitelse{...}`, and make the else fruits refer to it, for example `\def\fruity{\fruitelse}`, `\def\fruitz{\fruitelse}`, etc. When the set is really uncountable we are in trouble, but I don't know of such situations. And, ...the five letters 'fruit' are there only to enhance uniqueness of the names.

As example Fine gives the problem, treated by Schwarz, [24], to print vowels in bold face.[3]
The problem can be split into two parts. First, the general part of going character by character through a string, and second, to decide whether the character at hand is a vowel or not.
For the first part use for example, `\dolist`, TEXbook ex11.5, or `\fifo`, [19].

```
\def\fifo#1{\ifx\ofif#1\ofif\fi\process
#1\fifo}          \def\ofif#1\fifo{\fi}
```

For the second part, combine the vowels into a string, `aeiou`, and the problem is reduced to the question $\langle char\rangle \in$ `aeiou`? Earlier, I used the latter approach when searching for a card in a bridge hand, [14].[4] That was well-hidden under several piles of cards, I presume? Anyway, searching for a letter in a string can be based upon `\atest`, TEXbook, p.375, or one might benefit from `\ismember`, p.379. I composed the following

```
\def\loc#1#2{%locate #1 in #2
\def\locate##1#1##2\end{\ifx\empty##2%
\empty\foundfalse\else\foundtrue\fi}
\locate#2.#1\end}     \newif\iffound
```

Then `\fifo Audacious\ofif`
yields **Au**d**a**c**iou**s, with

```
\def\process#1{\uppercase{\loc#1}%
{AEIOU}\iffound{\bf#1}\else#1\fi}
```

[3] A bit misplaced example because the actions in the branches don't differ, except for the non-vowel part.
[4] The macro there was called `\strip`.

Note that en-passant we also accounted for uppercase vowels. By the way, did you figure out why a period—a free symbol—was inserted between the arguments for `\locate`? It is not needed in this example.[5] Due to the period one can test for substrings: $string_1 \in string_2$? Because, $\{string_1 \in string_2\} \wedge \{string_2 \in string_1\} \Rightarrow \{string_1 = string_2\}$, we also have the possibility to test for equality of strings, via `\loc`. Happily, there exists the following straightforward, and TEX-specific, way of testing for equality of strings

```
\def\eq#1#2{\def\st{#1}\def\nd{#2}
\ifx\st\nd\eqtrue\else\eqfalse\fi}
```

For lexicographic comparison, see [18], [19].

4 Array addressing

Related to the switch, or the old computed goto as it was called in FORTRAN, is array addressing. In TEX this can be done via the use of `\csname`. An array element, for example elements identified among others in PASCAL by `a[1]` or `a[apple]`, can be denoted in TEX via the control sequences

```
\csname a1\endcsname
%respectively
\csname aapple\endcsname
```

For practical purposes this accessing, or should we say 'reading,' has to be augmented with macros for writing, as given in [7], and [9]. Writing to an array element can be done via

```
\def\a#1#2{\ea\def\csname a#1%
\endcsname{#2}}\a{1}{Contents}
```

Typesetting (reading) via `\csname a1\endcsname` yields `Contents`, after the above.
The point I like to make is, that 'array addressing'—also called table lookup by some authors—is already there, although unusual and a bit hidden, but, . . . we are used to things like strong type-checking, isn't? Once we can do array addressing we can encode all kind of algorithms, which make use of the array data structure. What about sorting? See the Sort It Out subsection, for a glimpse, and the in depth treatment, [18], with O($n \log n$) algorithms, and application to glossary and index sorting.

5 Keyword parameters

In TEX literature the functionality of keyword parameters is heavily used. Some authors impose the syntax known from command languages upon TEX, for examples see [2], or [25]. In my opinion this is syntactic sugar, because of the following rhetorical question. What is essentially the difference between

```
\ref
\key W\by A. Weil
\paper Sur ...
...
\endref
```

as detailed in [25], and for example

```
{\def\key{W}\def\by{A. Weil}
\def\paper{Sur ...}...
\endref}%?
```

The typesetting is done in the cited case by `\ref...\endref`, and in the alternative case by `\endref`. The values for the keys are the background defaults and those temporarily redefined. Note that in both cases the order of the specifications is free and that defaults (empty) are used, for not explicitly specified values.

In my bordered table macro, [17], I could have introduced keyword parameters obeying the command languages syntax. Happily, I refrained from that. I needed several parameters. A parameter for framing, with functionalities nonframed, framed, and dotframed. A parameter for ruling, with functionalities nonruled, ruled, hruled, vruled, and dotruled. And a parameter for positioning of the elements, with functionalities centered, flushed left, and flushed right. (The first element of each enumerated list of values, acting as the default value.)
Furthermore, I decided to provide the user the possibility to *optionally* specify a caption, a header, a rowstub list, or a footer. If any of these is not explicitly specified, then the item will be absent in print too.[6] This resembles optional parameter behaviour, but has been realized by Knuth's parameter mechanism.
In following DEK's approach, I succeeded in keeping the encoding compact, and transparent. I experienced it as simple, direct, and serving extremely-well the purpose.[7]

5.1 Optional parameters

Among others, in LATEX, [20], the mechanism of optional parameters is used. Optional parameters are a special case of keyword parameters. Knuth used optional/keyword parameters abundantly, and called them just parameters, as opposed to arguments of macros. (Think for example of his various parameters and his `\every...`-s.) So it is already there, although in an unusual way.

Another example which illustrates the arbitrariness of the syntax choice with respect to optional/keyword parameters vs. Knuth's parameters is TUGboat's `\twocol` vs. LATEX's `twocolumn` style option.[8]

[5] If omitted the find of 'bb' in 'ab' goes wrong: abbb vs. ab.bb, will be searched.

[6] Another difficulty was to provide a default template, which can be overridden by the user. This was solved by the same approach.

[7] Earlier, I had a similar experience, van der Laan (1990).

[8] LATEX also provides `\twocolumn`.

5.2 Salomon's plain Makeindex

At NTG's 92 spring meeting David Salomon reported about his work in progress about adapting MakeIndex to work with plain. He used optional parameters, with the function as given in the following table

Source	Typeset	
	document	index
`^[abc]`	abc	abc
`^[xyz]{abc}`	abc	xyzabc
`^|abc|`	`abc`	`abc`
`^|\abc|`	`\abc`	`\abc`
`^{\abc}`	replacement text of `\abc`	same
`^[|\abc|!xyz]{}`	nothing	`\abc`, xyz

and combinations thereof.
The same functionality can be obtained via DEK's parameter mechanism. Only one parameter is needed. Let us call this the token variable `\p`. The idea is that the contents of `\p` has to be inserted before the index-entry in the index, and *not* in the text. Some symbols can be given a special meaning, like Salomon did, for example with ! (to denote a subentry).

Salomon's Source	Alternative
`^[abc]`	`^{abc}`
`^[xyz]{abc}`	`{{\p{xyz}^{abc}}}`
`^|\abc|`	`^{|\abc|}`
`^{\abc}`	`^{\abc}`
`^[|\abc|!xyz]{}`	`{\p{|\abc|!xyz}^{}}`

In the above | denotes TUGboat's verbatim delimiter. The macro for ^ has to be adapted accordingly. It is beyond the scope of this paper to work that out in detail. The point I like to make is, that the specification can be done, equally-well, if not simpler, via Knuth's parameter mechanism.

6 Mouth vs. stomach

When one starts with macro writing in TEX one can't get around awareness of TEX's digestive processing. Mouth processing is unusual. For the moment, I consider it as a special kind of built-in pre-processing, an unusual but powerful *generalization* of the elimination of 'dead branches.'[9]

Now and then encoding is published in TUGboat, and other sources as well, which looks difficult, and which does not seem to reflect the familiar algorithms. Sometimes, it has become difficult, because of the strived after processing in the mouth, see for example, [10], [21].[10] The latter author agrees more or less with what is stated above '. . . although the macros are hard to read. . . '.

What puzzles me, are the following questions.

> Why don't authors provide the straightforward TEX encoding, not restricted to mouth processing, as well?
> Why don't they make clear the *need* for mouth processing, or should I say mouth optimization?
> If so, why don't they start with the straightforward encoding and explain the adaptation steps?

Faced with the above questions myself, I would answer that it is apparently too difficult to do so.[11] Furthermore, I read and worked on the Math parts, the alignment parts, the macro chapter, and a substantial part of the dirty tricks Appendix D of the TEXbook, and did find until now only a comment about the *capability* of TEX's mouth processing along with the macro `\deleterightmost`. I know the argument that it is needed within an `\edef`, a `\write...`, and the like. I have heard of that, but from an application point of view, my obvious answer is: Isn't it possible to do the things outside those constructs, equally-well, and pass through the results?

> If authors don't help me out with the above, I consider the encoding as *l'art pour l'art*. Nothing wrong with that, on the contrary.
> The only thing against that is, that it will spread a negative image about TEX encoding, certainly not under the theoretical computer scientists, but under the day-to-day BLUe-type programmers, if not the authors who just use (La)TEX to get their work out, beautifully.

Agreed, Maus referred to the TEXbook, but Jeffrey could have provided a more intelligible solution, and should have refrained from burying his method under a sort of program correctness math. As it is at the moment, it is easier to start from scratch. I experienced that

[9] Knuth might forgive me my ignorance at this point. My brows are raised, when I see published code, restricted to mouth processing, which looks so verbose and unintelligible. I definitely turn my back on it, when the straightforward alternative encoding is familiar, compact, elegant and generic, despite rumour has it that TEX's mouth has the programming power of the Turing machine. As it is, that is something different from let us say literate programming, to indicate a broad stream of readable programs, in my opinion.

[10] By the way, when do we know that something is completely processed in the mouth? Is there a check on it? Or, . . . is it just an abstract part of the TEXnigma?

[11] And what about the efficiencies? From the viewpoint of the machine and with respect to human understanding? I have not seen the common and mouth versions of an algorithm published simultaneously, let alone have them compared with respect to timing.

already with the encoding of: the Tower of Hanoi, typesetting crosswords, generating n-copies, lexicographis comparison, and sorting. The published encodings inspired me to develop alternatives, that is true, but that should not be the aim, should it? Furthermore, I wonder how many *users* have been discouraged by those 'difficult to read' codes, especially when the familiar codes are straightforward?

6.1 n-copies

I needed Maus' functionality—avant la lettre—in typesetting a fill-in form, where a number of rows had to be repeated. Of course, my editor can do it—statically—and that served the purpose. It is easy for sure, but it does not look elegant. A straightforward use of tail recursion satisfied me better, because of the simplicity, the compactness and the elegance, at the expense of a negligible efficiency loss. See the example about the birdge form in Table Diversions, [17].[12] The tail recursion determines the number of copies dynamically, as do the other solutions given by DEK, for example the nice solution via the use of `\aftergroup`, TEXbook, p.374.

6.2 Sort it out

Jeffrey's problem is: given an unsorted list of (positive) integers via symbolic names, typeset the ordered list. In order to concentrate on the main issues, assume that his list adheres to Knuth's list structure, TEXbook, p.378. As example consider the list[13]

```
\def\lst{\\\ia\\\ib\\\ic}
\def\ia{314}\def\ib{27}\def\ic{1}
```

The sorted numbers 1, 27, 314, are obtained via

```
\def\\#1{\ifnum#1<\min\let\min=#1\fi}
\def\first#1{\def\lop\\##1##2\pol{%
        \let\min=##1}\ea\lop#1\pol}
\newif\ifnoe
\loop\ifx\empty\lst\noefalse\else
      \noetrue\fi
\ifnoe  \first\lst   \lst   \min,
   {\def\\##1{\ifx##1\min\else\noexpand\\
   \noexpand##1\fi}\xdef\lst{\lst}}
\repeat
```

The encoding implements the looping of the basic steps

- find minimum (via `\lst`, and suitable definition of the active list separator `\\`)
- typeset minimum (via `\min`)
- delete minimum from the list (again via an(other) appropriate definition of the active list separator).

For removing a typesetted element, I was inspired by `\remequivalent`, TEXbook, p.380.[14]

The above is effective for short lists, as was the case in Jeffrey's application.[15] For longer lists, techniques of order O($n \log n$) are more appropriate. For plain TEX encodings see [18].

6.3 Lexicographic comparison

Eijkhout, [5], provided macros—focused at mouth processing—for lexicographic ordering.
His `\ifallchars...\are...\before` made ample use of `\expandafter`, and is not easy accessible for somebody with say 2 years of TEX experience. Hackers might go into ecstasy, but application oriented users become discouraged. For a straightforward alternative, not restricted to mouth processing, see [19]. The point I like to make is, that I would have welcomed the familiar solution and the transformation steps as well.

7 Acknowledgements

Włodek Bzyl and Nelson Beebe are kindly acknowledged for their help in clearing up the contents and correcting my use of English, respectively.

8 Conclusion

It is hoped that authors who can't resist the challenge to impose syntaxes from successful programming languages upon TEX, also encode the desired functionality in TEX's peculiar way, and contrast this with their proposed improvements. The novice, the layman and his peers will benefit from it.

The difficulties caused by TEX's unusual encoding mechanisms, can best be solved via education, and not via imposing structures from other languages. The latter will entail confusion, because of all those varieties. Furthermore, it is opposed to the Reduced Instruction Set idea, which I like. For me it is similar to the axioms-and-theorems structure in math, with a minimal number of axions, all mutual orthogonal.

Publishing houses, user groups, and macro writers are encouraged to develop and maintain 'user interfaces,' which do justice to TEX's nature, and don't increase the complexity of TEX's components. Good examples are: TUGboat's sty files, AMS-LATEX & $\mathcal{AMS}$-TEX, and L$\mathcal{AMS}$-TEX. Macro-TEX and lxiii[16] are promising.

[12] The complexity is of order O(n), instead of O(log n), which is not important, because of the small number of copies involved.

[13] Equally-well, the comma could have been used as an active list separator, which looks more natural. I decided to adhere to Knuth's notation.

[14] I was not able to apply the parameter separator technique to locate the element to be removed.

[15] Remember, that sorting based on linear search has complexity O(n^2).

[16] The LATEX3 project.

File servers and archives are welcomed, but the compatibility, the simplicity and in general the quality, must be warranted too. Not to mention pleasant documentation and up-to-date-ness.
My wishful thinking is to have intelligent local archives, which have in store what is locally generally needed, and know about what is available elsewhere. The delivery should be transparent, and independent whether it comes from elsewhere or was in store.

References

[1] Appelt, W (1987): Macros with keyword parameters. *TUGboat* 8, no. (2), 182–184.
[2] Appelt, W (1988): TeX für Fortgeschrittene, Programmiertechniken und Makropakete. Addison-Wesley.
[3] Beebe, N.H.F (1991): The TUGlib server. MAPS 91.2, 117–123.
[4] Beeton, B.N, R. Whitney (1989): *TUGboat* 10, no. (3), 378–385.
[5] Eijkhout, V (1991): TeX by Topic. Addison-Wesley.
[6] Fine, J (1992): Some basic control macros for TeX. *TUGboat* 13, no. (1), 75–83.
[7] Greene, A. M (1989): TeXreation—Playing games with TeX's mind. TUG89. *TUGboat* 10, no. (4), 691–705.
[8] Hendrickson, A (1989): MacroTeX.
[9] Hendrickson, A (1990): Getting TeXnical: Insights into TeX macro writing techniques. *TUGboat* 11, no. (3), 359–370.
[10] Jeffreys, A (1990): Lists in TeX's mouth. *TUGboat* 11, no. (2), 237–244.
[11] Jensen, K, N. Wirth (1975): PASCAL user manual and report. Springer-Verlag. *TUGboat* 11, no. (2), 237–244.
[12] Kabelschacht, A (1987): `\expandafter` vs. `\let` and `\def` in conditionals and a generalization of plain's `\loop`. *TUGboat* 8, no. (2), 184–185.
[13] Knuth, D.E (1984): *The TeXbook*, Addison-Wesley.
[14] Laan, C.G van der (1990): Typesetting Bridge via TeX. *TUGboat* 11, no. (2), 265–276.
[15] Laan, C.G van der (1992a): Tower of Hanoi, revisited. *TUGboat* 13, no. (1), 91–94.
[16] Laan, C.G van der (1992b): FIFO & LIFO incognito. EuroTeX '92, 225–234. Also MAPS92.1. An elaborated version is FIFO & LIFO sing the BLUes.
[17] Laan, C.G van der (1992c): Table Diversions. EuroTeX '92, 191–211. A little adapted in MAPS92.2.
[18] Laan, C.G van der (in progress): Sorting in BLUe. MAPS93.1. Heap sort encoding is released in MAPS92.2.
[19] Laan, C.G van der (1992d): FIFO & LIFO sing the BLUes. MAPS92.2.
[20] Lamport, L (1986): LaTeX, user's guide & reference manual. Addison-Wesley.
[21] Maus, S (1991): An expansion power lemma. *TUGboat* 12, no. (2), 277.
[22] Pittman, J.E (1988): Loopy.TeX. *TUGboat* 9, no. (3), 289–291.
[23] Salomon, D (1992): NTG's Advanced TeX course: Insights & Hindsights. MAPS 92 Special. 254p.
[24] Schwarz, N (1987): *Einführung in TeX*, Addison-Wesley.
[25] Siebenmann, L (1992): Elementary Text Processing and Parsing in TeX. *TUGboat* 13, no. (1), 62–73.
[26] Spivak, M.D (1987): LAMS-TeX. TeXplorators.
[27] Youngen, R.E (1992): TeX-based production at AMS. MAPS92.2. 7p.