Kenneth J. Turner. A Structural Comparison of LaTeX and FrameMaker. Submitted to Electronic Publication - Origination, Dissemination and Design, April 1997.

A Structural Comparison of LaTeX and FrameMaker

Kenneth J. Turner

Computing Science and Mathematics, University of Stirling, Stirling FK9 4LA, UK (Email kjt@cs.stir.ac.uk)

SUMMARY

The general characteristics of LaTeX and FrameMaker as text markup languages are presented. This is followed by a structural comparison that looks at progressively finer features: documents, pages, text and floating objects. Supporting facilities and utilities are then discussed, including work by the author to support mixed LaTeX and FrameMaker environments.

1 Introduction

LaTeX [1,2] and FrameMaker^{TM} (e.g. [3]) both employ text markup, i.e. they use annotations rather than actual format to indicate the layout of text. Markup languages belong to a family that includes HTML (HyperText Markup Language [4]), derivatives of *roff* (Run-Off Language) and SGML (Standard Generalized Markup Language [5]). Although LaTeX and FrameMaker superficially share the same approach, their philosophies and details are very different. The aim of this paper is to examine the two from a structural point of view. The goal was essentially practical, namely to derive techniques and tools for supporting both LaTeX and FrameMaker in a mixed environment.

LaTeX is to some extent oriented towards the sciences, although it is a general-purpose language. LaTeX is widely used for scientific articles, papers, books and theses. LaTeX derives many of its benefits from the underlying TeX system [6], for example its excellence in typesetting mathematics. Since LaTeX is 'merely' a macro package written in TeX and since TeX is effectively a programming language, it is possible to extend LaTeX for almost any typesetting task. For this reason, it is difficult to define the exact capabilities of LaTeX. This article concentrates on LaTeX as it might be obtained in a standard distribution. Programmability makes LaTeX very flexible, but unfortunately it can be tricky to enhance or modify existing LaTeX behaviour. Some stylistic changes are easy (e.g. alphabetic section identification), others require a detailed knowledge of how LaTeX is implemented (e.g. altering the layout of a section heading). Changing LaTeX is probably beyond the interest of ordinary users, who can rely on styles defined by others.

The original LaTeX was developed largely by Leslie Lamport. Subsequently Michel Goossens, Frank Mittelbach, Alexander Samarin, Rainer Schöpf and others cooperated on its refinement. The current version is LaTeX2$_\varepsilon$, on its way to becoming LaTeX3. Major reasons for the popularity of LaTeX include its widespread distribution, its portability and the fact that it is free.

FrameMaker has been slanted more towards commercial publishing. It is entirely visual with WYSIWYG support, which makes it more approachable for those who are uninterested in program-like markup. Despite this, FrameMaker is surprisingly flexible. Attributes are

set through graphical selection in a variety of menu panels, so even non-programmers can achieve sophisticated results. Compared to LaTeX, FrameMaker is somewhat closed though there are utilities to manipulate FrameMaker files for programmable effects. FrameMaker shows its commercial origins in features such as the integrated drawing package, control of text-flows, colour separations, crop marks and change bars. However, it also includes support for typesetting mathematics – including the delightful ability to evaluate formulae. FrameMaker was originally developed by Frame Technology Corp. (now absorbed into Adobe Inc.). FrameMaker is sold commercially at a price that reflects its expected usage. The latest version is FrameMaker 5.

The most fundamental difference between LaTeX and FrameMaker is that LaTeX is batch-oriented but FrameMaker is interactive. It would have been possible for LaTeX to be interactive or FrameMaker to be batch-oriented. This difference permeates the design of both text-formatters. LaTeX is in a sense constructive, in that it is built from a hierarchy of facilities for typesetting with TeX. FrameMaker is essentially flat, consisting of a flat set of functions for setting text. From a programming standpoint, LaTeX might be regarded as procedural while FrameMaker is declarative.

The comparison of LaTeX and FrameMaker in the following sections takes a structural approach. For those who have not seen LaTeX before, Figure 1 shows what a user might write. Figure 2 is actually literal FrameMaker output, but serves two purposes. The figure can be regarded as the result of processing the LaTeX in Figure 1. Figure 2 also shows how text attributes are marked up 'invisibly' in FrameMaker.

The rest of the paper is organised as follows. In general the support of some typesetting aspect is presented first for LaTeX and then for FrameMaker. In a number of cases, opinions are expressed based on the author's experience.[1] These are necessarily generalisations and may not apply to particular classes of documents. Sections 2 to 5 deal with progressively finer structures within LaTeX and FrameMaker: whole documents, pages, text and floating objects. Section 6 looks at supporting facilities provided by tools. Section 7 explains how the foregoing analysis led to tools developed mainly by the author for supporting a mixed LaTeX and FrameMaker environment.

2 Documents

The main differences in document handling are summarised in Table 1. LaTeX files are more convenient because they are textual, so editing and file inclusion for example are easy. However, FrameMaker also permits inclusion of graphics and sound. LaTeX styles are generally more standard and readily usable for scientific publication. However, writing class and style files for LaTeX needs not only knowledge of TeX but also a detailed understanding of how LaTeX has been implemented. Creating FrameMaker templates is much easier, though by no means trivial. Because all stylistic aspects are menu-driven in FrameMaker, no programming is needed.

[1] The 'author' means the writer of this paper; the term 'user' is applied to the person who writes a LaTeX or FrameMaker document.

```
\section{Text Markup}

\subsection{Tables}

{\bf Tables} have the following features:

\begin{itemize}
  \item {\em not} more than one page
  \item labelled with captions
\end{itemize}

\begin{table}
  \begin{center}
    \begin{tabular}{| l | c | r |} \hline
      Feature    & Property                 & Support \\ \hline \hline
      Cell       & Alignment                & Yes \\ \cline{2-3}
                 & Width                    & If 'Left' \\ \hline
      \multicolumn{2}{| l |}{Border}        & Yes \\ \hline
    \end{tabular}
  \end{center}
\caption{Sample Table}
\end{table}

\noindent
where \defn{Width} may be given in a number of units.

\subsection{Mathematics}

The electromagnetic field strength $\cal E$ in scalar and vector potential
fields $\phi$ and {\bf A} is:

\[ {\cal E} = -\nabla\phi - \frac{1}{c} \frac{\partial {\bf A}}{\partial t} \]
```

Figure 1. Sample LaTeX Markup

2.1 File Facilities

LaTeX files contain only printable characters. This means that standard facilities can be used for editing, transmission by electronic mail, processing by scripts, etc. LaTeX files are usually quite compact – kbytes for a few pages. Subsidiary files can be imported using *input* (usually small portions of LaTeX) or using *include* (usually large pieces like chapters). A book might be built from a set of chapters as separate files. The author has written a utility to automate the process of including files in a larger document. Standard LaTeX style files like *epsf* or the *graphics* package allow EPS (Encapsulated PostScript) graphics to be

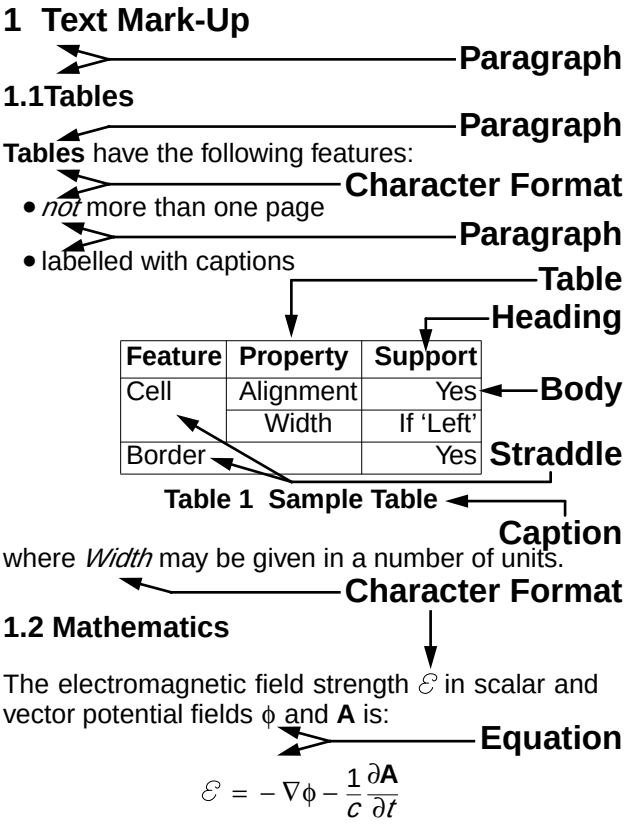

Figure 2. Sample FrameMaker Markup

included – at least if PostScript is being produced from the LaTeX.

A LaTeX file has to be processed before the resulting output can be seen. The *latex* tool invokes TeX with the LaTeX macro package (though this may be preloaded). The output is a DVI (Device Independent) file of typesetting commands. Screen previewers exist for DVI files. To print the document it is normal to convert the DVI file into another format like PostScript, and a variety of such translators exist. Converting LaTeX input to printed output thus requires several steps. A fair amount of processor time may be needed (some minutes for a large or complex document like a thesis). If errors are found in the formatting, the whole edit-process-view cycle must be repeated. Some typesetting issues (e.g. figure placement) are hard to get right and can require many iterations.

FrameMaker files are binary and so cannot be edited or processed so readily by standard tools. Of course, FrameMaker is an editor and previewer for its own files. However, FrameMaker files can be turned into alternative forms that are purely textual. MIF (Maker Interchange Format) conveys exactly the same information as a FrameMaker file, but textually. MML (Maker Markup Language) is a simplified textual markup, so it does not support the full range of FrameMaker facilities. Being text, both MIF and MML are amenable to use with standard tools.

File Facilities	LaTeX	FrameMaker
text formatting	one or more processing steps	immediately visible to user
	when document is processed	as document is entered
	more under tool control	more under user control
file contents	printable text	binary (native)
		printable text (MIF, MML)
file inclusion	by *input* or *include*	by paste from other document
		include (MML)
file size	compact	large (native)
		very large (MIF)
		compact (MML)
multi-file books	by file inclusion	special book file
graphics import	PostScript, using style file	PostScript, TIFF, ...
sound import	no	local platform formats

Document Styles	LaTeX	FrameMaker
standard styles	yes	rather limited
style usage	when document processed	when document created
style definition	relatively hard	relatively easy
	by programming	by attribute selection

Table 1. LaTeX and FrameMaker Documents

Because FrameMaker is WYSIWYG the user can immediately see the effect of any style setting. This is particularly beneficial for layout that the user wishes to control directly, such as placement of figures. However, there is a balance to be struck between allowing the tool or the user to take control. Both LaTeX and FrameMaker are intended for markup – the user should indicate logical formatting and leave it up to the tool to implement the markup effectively. LaTeX tends to take the reins, and can be quite wilful about how it wishes to lay out a document. Attempts by the user to interfere can be counterproductive. FrameMaker tries to give the user more control. But the penalty is that fewer things are done invisibly (though ordinary tasks such as line-breaking are automatic).

FrameMaker files are often rather large, mainly because they include full template information. Even a one-line document in FrameMaker might occupy several kbytes. Of course this overhead becomes less significant as the text contents grow, but it is a disadvantage if FrameMaker is used for many small files (like memos or letters). MIF files are rather verbose and can occupy four or five times the space of a normal FrameMaker file. MML files are generally compact because they contain limited markup in addition to the text.

To achieve the effect of including a file in FrameMaker it is necessary to copy the content of another file and paste this in. However, MML supports an *include* facility. FrameMaker has a special format to handle multi-file books: a master book file contains information on the subsidiary chapters. FrameMaker can import not only graphics but also other information such as formatted text and sounds. There is also the freedom to import

the entire contents of a file or to maintain only a link to the original. For certain kinds of documents, links can be a major space-saver and can automatically track changes to the source files.

2.2 Document Styles

LaTeX document styles are just macro packages that add to or modify the basic LaTeX macros. One of the major improvements in LaTeX2$_\varepsilon$ is that it is much easier to develop packages. A distinction is made between the root class of a document, an option within such a class, and additional stylistic features. LaTeX is supported by a widely used set of standard document classes for articles, books, letters, etc. Classes such as *article* and *report* are immediately suitable for scientific publication, and are widely used as perusal of any journal or proceedings will show.

There are also many style files that can be easily obtained. LaTeX thus has a set of effectively standard styles. When a LaTeX document is processed the class, option and style files it needs are read in (usually from a library). This makes it easy to maintain consistency among LaTeX documents of a particular type. It also allows a document written in one style (say, an in-house report) to be quickly re-cast in another style (say, for a journal).

The equivalent in FrameMaker is the template. This is like an empty document that contains defined styles for characters, paragraphs, tables, etc. A FrameMaker document is created as an instance of a template (usually held in a library). If a template contains many style definitions, every FrameMaker document derived from it carries these as a large overhead. Also if the template is changed after the document has been created, there is no way to update the document automatically. Instead FrameMaker requires the formats of an updated template to be applied manually to the document.

Although FrameMaker is distributed with sample templates to get users started, there does not appear to be the same degree of standardised use as with LaTeX. Interchange of documents between sites therefore requires each document to carry its own style information. The sample FrameMaker styles are also not so readily usable for scientific publication. To solve this problem and to give a greater degree of compatibility with LaTeX, the author has developed a set of LaTeX-like templates for use with FrameMaker. This is not as trivial as it sounds because it is necessary to recreate many features of LaTeX such as the formatting of a title page, contents list or index.

3 Pages

The main differences in page handling are summarised in Table 2. LaTeX page layout can be more awkward to control, and lacks the convenience offered by FrameMaker. In general LaTeX offers greater control over formatting text.

3.1 Page Layout

LaTeX page layout is controlled by about ten parameters, some of which interact. Certain aspects, such as setting up footers, are not part of the LaTeX facilities and require delving into the implementation. Laying out a page is therefore not very direct. Advanced features such as rotating a table or a page need additional style files that typically depend on

Page Layout	LaTeX	FrameMaker
user page layout	can be intricate	easy and flexible
	by parameter/macro	by drawing on master page
crop marks	with style file	yes
text-flows	just one	as many as required
figure/table placing	automatic, with user hints	automatic, with user hints
	needs processing to see	seen immediately
text/figure rotation	style file using PostScript	built-in
text/figure scaling	style file using PostScript	built-in

Frames	LaTeX	FrameMaker
text area	box	frame
text mode	left-to-right, maths, paragraph	text column, text line
text length	calculations possible	not visible to user
	rubber lengths for gaps	needs manual intervention
frame per paragraph	no	from reference page

Table 2. LaTeX and FrameMaker Pages

PostScript. Although hints about figure or table placement can be given to LaTeX, their actual positioning is up to the tool. For certain kinds of document it can be frustratingly difficult to put figures or tables where the user wishes. Nonetheless the LaTeX philosophy is that the user should not be concerned about physical layout.

By way of contrast, FrameMaker page layout is under direct visual control. Headers, columns and footers are drawn on a *master page* and used to format the *body pages* automatically. Documents may even have several master pages (e.g. cover sheet, left page, right page) that can be applied to selected body pages. FrameMaker offers slightly more flexible placement options than LaTeX for positioning figures and tables. The main advantage is that the user can view at once the effect of placement instructions. For example, positioning a figure immediately after a paragraph might be seen to cause a gap at the bottom of the page because the figure will not fit. But if a document changes significantly after manual positioning of figures, the placement may become unsatisfactory.

FrameMaker offers additional features that are not available with LaTeX. Whole pages can be rotated. As with PostScript figures for LaTeX, figures can be scaled. However, the effect of scaling is immediate and other forms of image such as TIFF can be handled. The major feature of FrameMaker not supported by LaTeX is the ability to define *text flows*. In a simple document there is just one flow of text from beginning to end. FrameMaker allows several flows of text, much as in a newspaper where a story may begin on the first page and finish on a later one.

3.2 Frames

LaTeX collects text in *boxes* that are processed in one of three modes: left-to-right, maths

or paragraph mode. Boxes can be raised or lowered, and can be saved for later re-use (to avoid repeated typesetting). Calculations of width and height are carried out using special length variables. Rubber lengths are an interesting concept, allowing stretchable spaces for equalisation of horizontal or vertical gaps.

FrameMaker collects text into *frames*. In fact as the name FrameMaker suggests, the frame is fundamental to the way that pages are constructed. There are flexible options for positioning frames. Most text is collected into *text columns*, but *text lines* are used for annotations in figures. *Anchored frames* are fixed relative to the text, and are convenient for floating objects like figures and tables. *Unanchored frames* are fixed in position on the page, and so are often used for background graphics. FrameMaker supports a third class of page called a *reference page*. This contains special frames that can be inserted automatically when a particular type of paragraph is written.

The calculation of lengths in FrameMaker is not visible to the user. The equivalent of rubber lengths is missing, so there is no way to achieve equal horizontal or vertical gaps without manual intervention.

4 Text

The main differences in text handling are summarised in Table 3. In principle LaTeX allows greater flexibility with sectioning provided the user is willing to write TeX macros. In practice, paragraph attributes in FrameMaker allow nearly every desirable effect. The analogue of the built-in LaTeX commands for sections and lists is the FrameMaker paragraph catalogue. Defining paragraph formats is rather detailed and is perhaps best left to a specialist in template design. For fixed layouts, FrameMaker generally offers a wider range of features than LaTeX. However, some limitations in LaTeX *tabbing* layouts are overcome in LaTeX tables. LaTeX line and page breaks can be harder to control than in FrameMaker. When it comes to special characters, LaTeX wins on the range of offered – especially for typesetting mathematics.

4.1 Text Units

LaTeX is provided with a rich set of commands for sectioning documents (*chapter*, *section*, *subsection*, ...) and for different kinds of lists (bulleted, enumerated, definitions, ...). The built-in commands are sufficient for most documents, but it is possible to define new ones (e.g. for a special kind of list). Changing the standard way of producing sections or lists requires TeX programming and patient investigation of the existing definitions. There are non-English language versions of LaTeX that respect national conventions such as hyphenation. The *babel* package gives some support for a large number of languages.

FrameMaker paragraphs are defined in a parameterised way, and are easy to alter through menu panels. Despite the fact that FrameMaker is not programmable, its paragraph attributes cover almost everything that would be required (e.g. positioning, indentation, font, alignment, tabs and relationship between paragraphs). A useful feature is direct control of widows and orphans, which are handled more at arms length by LaTeX. Perhaps the trickiest FrameMaker paragraph attribute concerns automatic ‘numbering’ (which includes the use of special characters such as bullets).

There are non-English versions of FrameMaker. A neat feature of FrameMaker is the

Text Units	LaTeX	FrameMaker
sectioning	built-in command	paragraph catalogue
	changes need programming	changes use menus
non-English use	with *babel* package	yes
	language per paragraph	language per paragraph
conditional section	with *ifthen* style	by condition tag
change bars	semi-automatic, with style file	automatic
rule	built-in	line from drawing tool
strut	built-in	by setting line height
widow/orphan	hint of given strength	automatic/manual

Fixed Layouts	LaTeX	FrameMaker
varbtim layout	built-in environment	fixed-width font
justification	left/centre/right	left/centre/right
layout with tabs	automatic/manual tab setting	manual tab setting
	left/right	left/right/centre/decimal
	no filler	filler between fields

Mathematics	LaTeX	FrameMaker
formula	box, automatic size	frame, size set by user
formula entry	algebraic	graphical
symbol set	very rich	missing specialised symbols
equation numbers	automatic, manual	paragraph format
equation array	*array* environment	table
extensible	yes	no

Breaks	LaTeX	FrameMaker
page breaks	hints of various strength	manual control
line breaks	hints of various strength	manual control
non-breaking space	yes	yes
hyphenation	automatic/manual	automatic/manual
	for paragraph language	for paragraph language
kerning	automatic	automatic, parameterised

Character Styles	LaTeX	FrameMaker
native font	Computer Modern	according to platform
PostScript fonts	through virtual font set-up	through configuration file
standard styles	yes	rather limited
non-English	extensive list	limited in most fonts
	accent commands consistent	commands per platform
maths symbols	extensive list	more limited

Table 3. LaTeX and FrameMaker Text

ability to state the language of a paragraph. For computer scientists, this feature is ideal for suppressing spelling checks on paragraphs containing program code.

Although FrameMaker allows for several numbering series, certain effects (such as numbering figures and tables separately within chapters) require ingenuity. By default a new paragraph is given the same format as the previous one. However, paragraph styles can be stored in a catalogue for quick selection by the user. Typically, paragraph styles are defined by a template designer and need not be set up by the user. Because FrameMaker lacks the equivalent of an *environment* in LaTeX, handling the start and finish of lists is clumsy. It is a nuisance that a paragraph number cannot be set in FrameMaker without printing it (though there is a work-around). Surprisingly, FrameMaker does not recognise appendixes specially. This means that appendixes have to be created as different paragraph types, making it harder to move sections between the body of a document and its appendixes. Footnotes are not numbered per chapter unless each chapter is put in a separate book file.

For the scientific user at least, LaTeX offers a range of standard section and list styles that are unlikely to need much extension. This facilitates exchange of documents between co-authors. FrameMaker, however, comes with a fairly limited range of document styles. The author has therefore developed a set of FrameMaker templates that emulate the appearance of LaTeX documents.

LaTeX *rules* are effectively filled boxes used for drawing horizontal (or vertical) lines. Zero-width boxes called *struts* ensure that a certain amount of vertical space is taken in a line. In FrameMaker, the equivalent of a rule is a line produced with the drawing tools. The effect of a strut in FrameMaker can be achieved by manually setting the height of a paragraph line.

FrameMaker supports some convenient features that are not so directly available in LaTeX. Text and diagrams can be governed by condition tags, allowing selective inclusion in a document; the LaTeX equivalent is the *ifthen* style. Conditions are useful where the same basic document has to be subset in different ways for different readers (e.g. to remove technical material for a management overview). FrameMaker can also be set up to insert change bars in the margin automatically where a document is altered. (The LaTeX equivalent is a shell script that compares two files and generates input for the *changebar* style by Johannes Braams.) If necessary, page numbering can be frozen when new text is to be entered.

4.2 Fixed Layouts

The LaTeX *verbatim* environment respects the format of the original text, except that tabs (as usual) are ignored. Normal paragraphs may be left-justified, centred or right-justified. LaTeX also offers a *tabbing* environment for typesetting with fixed tab positions. This is made more sophisticated through setting tab positions from a sample text line. To handle indentation, the initial tab stop can be incremented or decremented. Another variation allows text to be positioned hard right against a tab stop. More sophisticated tabs (e.g. for alignment on a decimal point) can be simulated using the *tabular* environment. It is irritating that some of the LaTeX accent commands are redefined inside the *tabbing* environment.

Verbatim layout in FrameMaker requires a fixed-width font. Tabs can be set in a paragraph for something similar to *tabbing*. FrameMaker allows tabs stops that align on the centre of text or on decimal points. Gaps between tab fields can also be filled, for example with full stops.

$$\left\| \begin{bmatrix} \int_1^4 (3x^2 - 2)dx & \frac{2}{3} + \frac{1}{6} \\ \frac{d}{dy}(ye^y) & \sum_{z=1}^{5}(z^3 + z) \end{bmatrix} \right\|_{y=0}^{1}$$

$$\left\| \begin{bmatrix} \int_1^4 (3x^2 - 2)\,dx & \frac{2}{3} + \frac{1}{6} \\ \frac{d}{dy}(ye^y) & \sum_{z=1}^{5}(z^3 + z) \end{bmatrix} \right\|_{y=0}^{1}$$

Figure 3. Comparison of LaTeX and FrameMaker Mathematics Typesetting

4.3 Mathematics

LaTeX sets formulae in boxes whose size is calculated automatically. Thanks to TeX, LaTeX has a rich set of mathematical symbols and delimiters of varying sizes (e.g. braces, matrixes and integrals). LaTeX also supports standard mathematical requirements such as sub/superscripts and under/overlining. Equations can be automatically or manually numbered, and can be grouped in arrays. Typesetting mathematics in LaTeX can be intricate, although the commoner features are quickly learned. TeX embodies considerable expertise in how to set mathematics effectively.

FrameMaker has an integrated feature called FrameMath that sets formulae in frames. Navigating parts of a formula using graphical editing requires some getting used to. It is often necessary to 'shrink wrap' a frame after setting a formula, and then to expand it during later editing. FrameMaker lacks a number of the special mathematical symbols in LaTeX, largely because it is used with standard fonts like *Symbol*. Nonetheless, FrameMaker comes with the kind of built-in delimiters that LaTeX does.

It is rather subjective whether LaTeX or FrameMaker handles mathematics 'better'; there is often little to differentiate the results. The programmability of LaTeX means that new notations like operators and formats can be introduced, whereas FrameMaker is limited to its built-in capabilities. As an example of formatting mathematics, compare the LaTeX typesetting in the upper half of Figure 3 to that of FrameMaker in the lower half. Remarkably, FrameMaker can carry out symbolic and numeric evaluation of expressions. For example the formula shown in Figure 3 can be evaluated in about four (manual) steps. To help with difficult integrals, FrameMaker is provided with on-line tables that contain standard rewrite rules.

4.4 Breaks

Control of line breaks and page breaks in LaTeX is essentially advisory. The user provides hints of various strength to LaTeX (actually TeX), and the breaks may or not be implemented as desired. Again this is a case where LaTeX takes the view that it and not the user should be in control of physical layout. LaTeX supports a non-breaking space between adjacent words, and an explicit page break command can be used where required.

Since what FrameMaker does to break lines and pages is seen directly, it is usually easier to obtain the required result. A *soft return* can be used to prematurely end a line within a paragraph, while a *hard space* can be used to ensure that adjacent words are not

split. Curiously, FrameMaker does not have a page break command; instead the paragraph that should start on a new page has a *top of page* attribute set.

Although TEX is generally good at hyphenation and it is possible to give advice about this, it is sometimes necessary to use an empirical approach. Perhaps the commonest complaint from LATEX is an under/overfull horizontal or vertical box. TEX is extremely fussy about such things, although it is possible to define the degree of looseness with which lines or pages are broken.

FrameMaker offers a fine degree of control over kerning and hyphenation. This is especially useful with narrow columns or to fit text well into a given space. For multi-column work, FrameMaker deals nicely with balancing flows between columns. It can also feather columns, adjusting the text length to fit the column size exactly.

4.5 Character Styles

LATEX is often used with just the *Computer Modern* type family designed by Donald Knuth. In fact this is a comprehensive set of fonts covering various character styles and weights, non-English and accented characters, mathematical and special symbols. TEX fonts are stored as pixels, so one font definition often leads to many variants at different basic font sizes and magnifications (*magsteps*). Setting up fonts for TEX can be a black art, and is complicated by the availability of different font encodings (e.g. *OT1* and *T1*). Font installation may also require knowledge of METAFONT. It is possible to use PostScript fonts with TEX through the *virtual font* mechanism. Unfortunately setting these up needs specialist knowledge, so most users will make use of ready-made definitions (say, for the PostScript fonts found on many printers).

FrameMaker uses fonts native to the platform it runs on, such as PostScript^{TM} or TrueType^{TM}. A configuration file establishes the mapping between font names and how they are known to FrameMaker. Non-English and accented characters depend on the original fonts, so unusual accents like Turkish ğ and İ may not be available. Character formats can be stored in the *character catalogue* – often derived from the template.

The difference between LATEX and FrameMaker is more profound when it comes to typesetting mathematics. FrameMaker will normally use something like the *Symbol* font for mathematics, but this is somewhat deficient compared to *Computer Modern*. Although FrameMaker adds a variety of special symbols such as integrals and summations, it cannot fully compete with LATEX. Accented characters in LATEX use a fairly consistent set of commands, but the control sequences in FrameMaker may be platform-dependent.

Because FrameMaker does not have standard character styles in the way that LATEX does, the author has defined a LATEX-compatible set of formats that control size, weight and style. Font selection in FrameMaker is like that of the original LATEX: it is not compositional, i.e. all attributes are set at once. For example, it is not possible to set bold and italics separately since a character format defines weight and angle together. LATEX2$_\varepsilon$ has introduced a greater level of independence in selecting font attributes.

5 Floating Objects

The main differences in handling floating objects are summarised in Table 4. FrameMaker has the general concept of frame to deal with floating objects, though specific uses of

Tables	LaTeX	FrameMaker
format catalogue	no	yes
multi-page tables	manual splitting, or package	automatic
header/footer	user convention only	built-in
table to/from text	no	yes
column width	automatic/manual	semi-automatic/manual
row height	automatic	automatic
straddling	yes	yes
table/cell border	yes	yes
colour/shading	no	yes
caption layout	no	yes

Figures	LaTeX	FrameMaker
figure construct	built-in	simulated by one-cell table
picture drawing	limited, helped by front-ends	straightforward line art
clip-art	any external PostScript	supplied and external

Table 4. LaTeX and FrameMaker Floating Objects

frames need to be defined. FrameMaker deals better with tables than LaTeX in nearly all respects, except that semi-automatic or manual calculation of column widths is necessary. Apart from the fact that FrameMaker lacks a figure construct, it is preferable to LaTeX for producing diagrams.

5.1 Tables

LaTeX has the concept of *table* as a floating object that could contain almost anything. However, by convention (and captioning) it is used for tabular information. Table information is often defined using the *tabular* environment that defines what would ordinarily be called a table. LaTeX support for tables is very flexible. Row and column sizes are determined automatically, though fixed width columns are also possible (but only left-justified). Columns can be left-justified, centred or right-justified. It is also possible for table cell contents to span several rows or columns. The so-called *@-specifier* can be used to place fixed text between columns (e.g. a period, thus achieving alignment on decimal points). Once a LaTeX table has been typeset it behaves like a single box and so can be processed in other ways such as horizontal centering. However a box cannot be split across pages, so multi-page tables need user intervention or the *longtable* package by David Carlisle.

FrameMaker has essentially the same facilities for formatting tables (except for the LaTeX @-specifier, which corresponds to an explicit column of values in FrameMaker). FrameMaker uses the term *straddling* for contents that occupy more than one logical cell. As usual, FrameMaker allows the user to see how table layout is progressing. A range of options allows columns to be resized, for example to a fixed width or to the width of another column. FrameMaker also offers more freedom when it comes defining the layout of the

whole table or a single cell. A table has parameters describing its borders. Cell borders, colour and shading can also be set as differences from the standard table settings, making it easy to highlight certain cells. Tables may have defined headers and footers, permitting automatic multi-page tables. The biggest advantage is that table formats are stored in a catalogue (typically derived from a template). This ensures consistency and saves effort for a document that uses many tables of the same basic structure (like this paper). Tables can be converted to text with separators and vice versa. This is useful when importing data from a spreadsheet for example. The position and layout of a table caption can be controlled through menus.

5.2 Figures

Figures in LaTeX are much like tables – floating objects with a different series of captions. Figures contain diagrams only by convention. LaTeX provides primitive drawing facilities for lines, arrows, (rounded) rectangles, circles and text. There are severe limitations on these (e.g. the slope of a line or the size of a circle). Nonetheless, a very patient user can produce acceptable line art. A useful feature is the ability to replicate a picture object at a number equally spaced points. There are higher-level drawing languages and tools that can generate LaTeX picture commands, making the process of producing diagrams less fraught. An alternative solution is to use one of the many drawing packages that can produce PostScript. If LaTeX is translated into PostScript for printing, such figures can be included. This also allows LaTeX to exploit separate clip-art libraries.

FrameMaker does not have the structural concept of a figure. Instead a figure must be simulated as, say, a one-cell table. (The table catalogue simplifies this to some extent.) FrameMaker has a built-in drawing package for straightforward line art. One of the nicer features is *gravity* whereby lines are attracted to nearby objects. This allows lines to be neatly terminated on object boundaries. FrameMaker also allows import of files in PostScript, TIFF, etc. according to the platform on which it runs. This is commonly the preferred approach to diagram production in professional publishing, since it allows the use of a dedicated drawing editor. FrameMaker comes with a small clip-art library produced using its own drawing package.

6 Supporting Facilities

The main differences in supporting facilities are summarised in Table 5. Both LaTeX and FrameMaker can generate the usual kinds of list automatically from a document. Cross-referencing is also possible with both. LaTeX uses external utilities for citations and indexing, while FrameMaker needs an external utility for citations only. LaTeX creates indexes and similar lists from embedded commands that can make the main text unreadable. FrameMaker embeds special markers that do not obscure the main text.

6.1 Generated Lists

LaTeX has built-in support for generating lists of contents, figures and tables automatically. Cross-references to sections and pages are also automatic. LaTeX itself offers only basic support for citations and indexes, but post-processing is possible with BibTeX for citations

Generated Lists	LaTeX	FrameMaker
content/figure/table	automatic	automatic, invoked by user
index	embedded commands	embedded markers
	makeindex utility	automatic, invoked by user
	text can become unreadable	text remains readable
citations	embedded commands	embedded markers
	BibTeX utility	*bibframe* or *fbib* utility
glossary	embedded commands	special markers
	no known utility	no direct support
special lists	macro programming	special markers

Slides	LaTeX	FrameMaker
formatting	*slide* class	no external utility
	slide styles by author	slide template by author
colour slides	*colordvi* or *colourslide* style	slide template by author
overlays	*slide* class	no
separations	*slide* class	built-in generic feature

Dictionaries	LaTeX	FrameMaker
spelling checker	external utility	built-in
error checking	external utility	built-in
thesaurus	external utility	built-in (some versions)

Filters	LaTeX	FrameMaker
conversion	HTML, RTF, ...	HTML, PDF, Word, ...
utilities	mainly public domain	mainly commercial

Practical Issues	LaTeX	FrameMaker
platforms	many	not as many as LaTeX
pricing	free, some commercial	commercial, some discounts
user interface	embedded textual commands	graphical, menu-driven
commands to learn	100–650	60 menus/700 key shortcuts
processing	concentrated, can be high	distributed during editing
virtual memory	moderate	can be high
robustness	high	reasonably high
recovery	not really required	automatic
guidance	public-domain tutorials	tutorials with manuals
	limited online help	online help
	textbooks available	textbooks available
	error diagnosis can be difficult	error diagnosis generally easy
archives	CTAN sites	FUN sites
newsgroup	for TeX generally	yes

Table 5. LaTeX and FrameMaker Supporting Facilities

```
$\#$\index{#@{\punc{\#}}}\indexz{#@{\punc{\#}}|mnem{hash}}
$\%$\index{%@{\punc{\%%}}}\indexz{%@{\punc{\%%}}|mnem{percent}}
$\&$\index{&@{\punc{\&}}}\indexn{&@{\string\punc{\&}}}{ampersand}
$@$\index{"@@{\punc{"@}}}\indexn{"@@{\string\punc{"@}}}{at}
$/$\index{/@{\punc{/}}}\indexn{/@{\string\punc{/}}}{slash,oblique,division}
$\backslash$\index{\@{\punc{$\backslash$}}}
\indexz{\@{\punc{$\backslash$}}|mnem{backslash}}
$<$\index{<@{\punc{$<$}}}\indexn{<@{\string\punc{$<$}}}{lessthan}
$>$\index{>@{\punc{$>$}}}\indexn{>@{\string\punc{$>$}}}{greaterthan}
$\{$\index{"|@{\punc{$\lbrace$}}}
\indexn{"|@{\string\punc{$\string\lbrace$}}}{openbrace}
$\}$\index{~@{\punc{$\rbrace$}}}
\indexz{~@{\punc{$\rbrace$}}|mnem{closebrace}}
```

Figure 4. Extreme Example of LaTeX *Indexing*

and *makeindex* for indexes. Although LaTeX has a *glossary* command, there does not appear to be a utility for organising glossaries. If the standard lists are not enough, TeX macros can be written to generate other kinds from commands embedded in the text.

FrameMaker has built-in support for generating lists of contents, figures and tables automatically; the user has to explicitly call a command to (re)generate such lists. The lists require special reference pages with slightly tricky characteristics. However, using definitions from an existing template is straightforward. Cross-references to sections and pages are also automatic; again the user has to call a command to (re)generate the references. Surprisingly, FrameMaker has no standard support for citations. The author developed the *fbib* utility to allow FrameMaker to work with normal BibTeX bibliographies. (This extends the earlier *bibframe* utility by Tommy Persson in a more LaTeX-like way.) FrameMaker indexes are generated from special markers in the text. Indexing also requires rather special reference pages. There is no direct support for glossaries or other kinds of list, but arbitrary markers can be placed in the text to generate them. In fact it is easy to produce arbitrary lists from selected items in a FrameMaker document (e.g. to check cross-references). FrameMaker can also generate hypertext links, which adds value to tasks like producing an index.

Since LaTeX uses embedded commands for indexes and the like, it can be hard to see the main text separately from the embedded commands. The FrameMaker embedded information is normally hidden from the user unless a special marker is selected. This makes reading very much easier. An extreme example of how unreadable indexed LaTeX can be is shown in Figure 4. This is taken literally from a genuine document by the author, and merely lists the punctuation marks `#%&@/\<>{}` using some author-defined macros for indexing.

6.2 Slides

LaTeX originally used an auxiliary program SliTeX for formatting slides, but this has now been superseded by the *slide* class. Although this has some advanced features such as use of overlays and support of colours through separations, the author finds it not so convenient. The author has therefore developed his own support for colours in LaTeX (an alternative approach to *colordvi* by Jim Hafner). This is integrated with a readily usable style file for producing coloured slides using predefined colour schemes. The author has also written a converter that turns such slide presentations into PowerPoint^{TM}. FrameMaker comes with some basic slide templates, but the author has implemented templates similar to those he developed for LaTeX.

6.3 Dictionaries

LaTeX does not have its own spelling checker. But with a utility to strip out LaTeX commands, it is possible to use a standard spelling checker. LaTeX commands can be removed with *detex* by Daniel Trinkle (or the similar *untex* by the author). FrameMaker has its own built-in spelling checker. However, this is most useful on platforms where FrameMaker shares its dictionary with other system utilities. In fact the FrameMaker spelling checker deals with several related issues such as incorrect hyphenation, repeated words and extra white space. FrameMaker can also spell check several languages within a document. LaTeX does not have a dedicated thesaurus tool, but any available one can be used separately. Some versions of FrameMaker have thesaurus support.

6.4 Filters

There are a number of public domain utilities for filtering LaTeX. Many of these convert LaTeX into other forms such as HTML (HyperText Markup Language) and RTF (Rich Text Format). There are public domain utilities for FrameMaker, but also a number of commercial offerings. In the main these are translators between FrameMaker and other publishing formats such as HTML, InterLeaf^{TM}, Microsoft Word^{TM}, PDF (Portable Document Format) and WordPerfect^{TM}. Some versions of FrameMaker are supplied with *fmbatch* for batch-processing of documents. The public domain utilities *MIFMucker* by Ken Harward and *MIFFed* by David Cortesi support systematic manipulations of FrameMaker files in MIF form.

LaTeX itself does not support hypertext, but *latex2html* by Nikos Drakos allows hypertext links to be embedded in the translation. FrameMaker has direct support for hypertext: embedded graphics and sounds, links within or between files, graphic buttons and pop-up menus.

6.5 Practical Issues

LaTeX is widely available free of charge. LaTeX also runs on many platforms, in some cases as a commercial package. In the author's experience, LaTeX and its supporting TeX are very robust. About the worst that can go wrong is a LaTeX input error causing the contents of an auxiliary file (*.aux*) to be lost. Fortunately, recovery from this just requires running LaTeX again.

Large LaTeX documents can be time-consuming to process, and it is often necessary to run LaTeX several times to get things such as figure placement correct. LaTeX plus TeX makes for a large language: there are perhaps 650 commands, though novices might use only 100 or so of these. Because of its complexity, the learning curve for LaTeX is relatively steep for beginners. However, an experienced user can be very productive through using standard LaTeX or modifying it. LaTeX error messages can be very obscure, largely because they reflect what is happening in the underlying TeX. There is a large group of LaTeX users to ask for help, and a newsgroup *comp.text.tex* for TeX and its derivatives. CTAN (the Comprehensive TeX Archive Network) contains many public domain style files and utilities. There are several tutorials and textbooks (e.g. [1,2]) for LaTeX.

FrameMaker is sold commercially, bringing the benefits of professional support. In the author's experience, FrameMaker is not as robust as LaTeX. But even if FrameMaker crashes it saves a recovery file that minimises loss. FrameMaker can also make backup files automatically from time to time during editing.

Because changes to a FrameMaker document are processed 'on the fly', there is no real delay in seeing the results. Of course, this just means that the processing time is distributed over a longer period. Generated lists may have to be re-created several times. In fact the concentrated processing needed for a LaTeX document could well be less than that for FrameMaker over an extended period. FrameMaker can also place high demands on virtual memory. FrameMaker is generally easy to use because of its graphical interface, so initial learning is rapid. Although a typical FrameMaker implementation might have around 60 menu panels, learning the basics is straightforward. There are keyboard shortcuts for many options, but since there are about 700 shortcuts the average user may employ only a small number. There is a large group of FrameMaker users to ask for help, and a newsgroup *comp.text.frame*. There are product manuals for FrameMaker, several textbooks (e.g. [3]), FUN (the Frame Users' Network) and online help with the package itself.

7 LaTeX and FrameMaker Utilities

The goal of the structural comparison given in the preceding sections was to establish a systematic basis for translating between LaTeX and FrameMaker. The author, like many others, lives in a mixed LaTeX and FrameMaker environment. It is therefore highly desirable to be able to use both markup languages. For exchange of documents with other institutions (including publishers), the possibility of conversion between LaTeX and FrameMaker is attractive. The author has therefore developed a number of utilities that support both text formatters, including translation between them. In a few cases the author has extended an existing public domain solution. The utilities were implemented in a Unix environment with different languages according to the problem being solved: *C*, *perl*, *lex*, *yacc* and *sh* scripts.

Various LaTeX and FrameMaker utilities written by the author have been mentioned during the paper. The URL *http://www.cs.stir.ac.uk/~kjt/software/* gives public access to the following and others:

checktex: check of balanced pairs within a LaTeX file; this includes checks for brackets, braces, parentheses and matching **\begin**...**\end** environments
colours.sty: support of coloured text for LaTeX

colourslide.sty: support of coloured slides for LaTeX
fbib: allows use of BibTeX bibliographies with FrameMaker
fgrind: supports formatting of 'program' listings for FrameMaker; this is an adaptation of the *lgrind* (LaTeX) and *vgrind* (*troff*) family of translators
la_mml: converts LaTeX to MML format that can be read into FrameMaker
la_ppt: converts LaTeX using *slide.sty* or *colourslide.sty* to an outline file that can be read into PowerPoint
la_temp: LaTeX-like templates for FrameMaker
mif_la: converts FrameMaker MIF output to LaTeX
mktex: partially automates the inclusion of files in a multi-file LaTeX document
slide.sty: support of monochrome slides for LaTeX
untex: strips LaTeX commands to leave basic text

8 Conclusion

Both LaTeX and FrameMaker have large user communities, and neither is superior in any absolute sense. Each has strengths and weaknesses, and each will appeal to different users or for different purposes. It is perhaps surprising that two approaches of the same general type (markup) should differ so much, though this mainly reflects their respective batch-oriented and interactive natures. Although LaTeX and FrameMaker are in competition with each other and with a number of other commercial text formatters, it is clear that each has its own particular contribution to make.

Acknowledgements

The author is very grateful to the following for carefully reviewing a draft of the paper: David Cortesi (Silicon Graphics Inc.), Laurent Dami (Centre Universitaire d'Informatique, Geneva), Christoph Eyrich (Freie Universität, Berlin), Rick Schumeyer (University of Delaware), Joseph Slater (Wright State University, Dayton). Technical support staff of Frame Technology Corp. (now part of Adobe Inc.) kindly supplied the author with specifications of the MIF and MML formats.

REFERENCES

1. Leslie Lamport, *LaTeX: A Document Preparation System*, Addison-Wesley, Reading, Massachusetts, USA, 1994. second edition.
2. Michel Goossens, Frank Mittelbach, and Alexander Samarin, *The LaTeX Companion*, Addison-Wesley, Reading, Massachusetts, USA, 1994.
3. Linda Branagan and Micheal Sierra, *The Frame Handbook: Building FrameMaker Documents that Work*, O'Reilly and Associates, USA, 1994.
4. World-Wide Web Consortium, *Hypertext Markup Language 3.2 Reference Specification*, REC-html-32, World-Wide Web Consortium, USA, January 1997.
5. ISO/IEC, *SGML – Standard Generalized Markup Language*, ISO/IEC 8879, International Organization for Standardization, Geneva, Switzerland, 1992.
6. Donald E. Knuth, *The TeX Book*, Addison-Wesley, Reading, Massachusetts, USA, 1986.