With LaTeX into the Nineties

FRANK MITTELBACH

Fachbereich Mathematik
Universität Mainz
Staudinger Weg 9
D-6500 Mainz
schoepf@dmznat51.bitnet

Electronic Data Systems
(Deutschland) GmbH
Eisenstraße 56
D-6090 Rüsselsheim
qzdmgn@drueds2.bitnet

RAINER SCHÖPF

Institut für Physik
Universität Mainz
Staudinger Weg 7
D-6500 Mainz
schoepf@dmznat51.bitnet

ABSTRACT

During the last three years, LaTeX has spread widely, even into such new fields as business applications. The fact that there are new classes of users forces one to reconsider the LaTeX implementation and some of its features. Within a few years, LaTeX 2.09 alone will not be sufficient to satisfy the increasing needs of its users. As a consequence one of the important characteristics of the LaTeX concept — the possibility of exchanging documents — is in danger of being sacrificed on the altar of local changes and enhancements.

Starting from these considerations and from our experiences of several years of LaTeX support, we will present a proposal for a re-implementation of LaTeX. This new version would not only preserve the essential features of the present user interface (in order to be compatible with old LaTeX files), but also take into account already formulated requests, as well as future developments.

1. Introduction

During the last several years, LaTeX has become a widely used tool for typesetting documents. Its advantages over WYSIWIG systems as well as over plain TeX — logical design and high-level commands for formatting issues — allow even the inexperienced typesetter (author) to easily produce readable output. Especially for author groups, the concept of logical design is essential to ensure uniformity of their work.

At least in theory, LaTeX is a markup language which enables the users to specify their input as logical parts, e.g., to mark a text fraction as a "quotation" or as a "labeled list" instead of supplying formatting commands such as "indent the next three lines and start with a bullet".

The translation into formatting commands, i.e., TeX primitives, is restricted from use (again, at least in theory) in the source file. It will take place instead, unnoticed by the user, in one of several style files which should be applicable to the same input source producing different layouts. LaTeX itself provides four different prototype styles (`article`, `report`, `book` and `letter`) which should be used for different types of input sources, i.e., should not be interchanged. This contradiction at a first glance (one can't switch between `article` and `report` for example) is the first (of many) misunderstandings for LaTeX users and amateur style designers.

Since there is only one officially supported document style for each type of input, TeX users with plain TeX experience often find themselves unable to produce a desired layout with LaTeX. So, after some unsuccessful attempts, they return to plain TeX even if they then have to face other problems (such as cross-references, etc.) which are easily provided in a markup language such as LaTeX.

In this case the root of the problem is the fact that most of the LaTeX interfaces are poorly documented so that even TeX experts might have trouble designing an offset layout style for example, which could be used instead of the standard report style. The result is that LaTeX documents all look alike because all existing document styles are either unusable (because of many bugs), or they are only variants of the prototype styles, without noticable differences.

2. The LaTeX User Interface — or Essential Features of LaTeX

For every type of document LaTeX provides a set of high-level formatting commands which themselves are defined with the help of internal macros. This internal code is highly modular, often ingenious,[1] and allows a wide variety of layouts if the style designer is sufficiently familar with the interfaces.

On the surface, LaTeX demonstrates a thought-out concept too: similar situations require similar syntax, unusual cases are hidden behind optional arguments ... But that isn't all. As a standard in all applications (but with different layouts) LaTeX provides:

- automatical generation of contents listings
- a powerful cross-referencing mechanism with symbolic names. This allows insertion, deletion and movement of text blocks without re-arranging the equation numbers, etc.
- the possibility of typesetting only parts of the document without losing cross-references, counter-values, etc.
- a general float mechanism for automatic placement of figures, tables, etc. independently of each other (but with each class preserving its order)
- in conjunction with BibTeX and MakeIndex, it has powerful tools for automatic creation of index and bibliography entries[2]
- fully implemented size changing commands for several types of fonts
- a general mechanism for switching page layouts (running headings, etc.)
- ...

So, why not use LaTeX? This question leads us to our next topic.

3. Limitations of LaTeX Version 2.09

Limitations to the current LaTeX version can be divided into several groups, which we will discuss separately, giving examples as we go along.

3.1 Implementation Disasters

This is where most of our examples are located. In a way it is also the group of problems which are the easiest to solve: just learn from the mistakes Leslie Lamport made but use all his good ideas.

Fragile commands

Maybe the most troublesome concept in this category is the classification of commands into *robust* and *fragile* ones. At the bottom of page 23 the LaTeX book says:

> The argument of a sectioning command may be used for more than just producing the section title; it can generate a table of contents entry and a running head at the top of the page. [...] When carried from where it appears in the input file to the other places it is used, the argument of a sectioning command is shaken up quite a bit. Some LaTeX commands are *fragile* and can break when they appear in an argument that is shaken in this way. Fragile commands are rarely used in the argument of a sectioning

[1] Clearly not all parts of LaTeX are well implemented. But the overall concept is wisely chosen and this isn't affected by design or implementation bugs in its modules.

[2] At a site with an up-to-date LaTeX installation, both programs should be available. Unfortunately, this is often not the case.

> command. [...] On the rare occasions when you have to put a fragile command in a section title, you simply protect it with a \protect command.

And later on (p. 24):

> An argument in which fragile commands need \protect will be called a *moving* argument. Commands that are not fragile will be called *robust.* For any command [...] I will indicate whether it is robust or fragile. Except in special cases [...] a \protect command can't hurt, so it is almost always safe to use one when you're not sure if it's necessary.

Isn't that easy? Unfortunately not, because a broken command will produce a totally unintelligible error message and, to make the chaos perfect, not only could this error occur at a different location, it is also possible that the error will not vanish when the missing \protect is finally discovered. This is hell for novice LaTeX users, but even experts are shaken quite a bit if they commit this sin.

The \protect is used to suppress unwanted expansions which are the reasons for the errors mentioned above. This is the wrong design decision: it would be better to suppress all expansions by default, and allow the user to expand single macros if there is need for it.

LaTeX *error recovery*

This part of the implementation can be summarized in a single statement: there is none. Actually there are several situations where the LaTeX error routine is triggered but the recovery mechanism isn't very powerful. On its own, this poses no problems because one can adopt the philosophy "understand the error, correct the source and re-run", but unfortunately the error help messages aren't very enlightening:

```
You're in trouble here.  Try typing <return> to proceed.
If that doesn't work, type X <return> to quit.
```

Well, we certainly know we're in trouble when we see a whole page of error messages coming from TeX's stomach,[3] and instructions on how to turn off the computer and go home: that isn't what the user expects to get when he or she enters 'H' (for help) after a LaTeX error.

The error messages themselves are often simply wrong; for example, the input

```
\begin{center}
␣
\end{center}
```

produces the error

```
Something's wrong--perhaps a missing \item
```

In nine out of ten cases this error isn't caused by a missing \item so the user doesn't know what to correct — in this case the center environment expects text, i.e., something in horizontal mode, and not just a blank line. Looking in the manual helps a little bit, because there the error messages are explained in greater detail, but all in all the error messages produced by LaTeX are a mess from the user's point of view.

The generic list environment

The generic list environment is one of the central modules of the LaTeX implementation. It is used internally by most standard environments provided by LaTeX; even environments such as center are handled as a special kind of list (with an empty \item command).

To allow for such a variety of applications, the list environment has nearly 20 parameters and switches which can be manipulated to change the layout. Additionally the default values for these parameters (as defined in the document style file) depend on the level of nesting; that is, the document style may provide different default spacing for lists within lists.

[3] LaTeX's tendency to produce horrible-looking error listings is actually a TeX problem which should be listed under "TeXnical limitations". A large macro package such as LaTeX is bound to have many expansion levels and there is no possibility of suppressing the intermediate part in the stack history when TeX spots an error. In our opinion there should be a TeX \tracing... command to control the amount of history listing.

But there are also a few implementation problems:

- An actual conceptual bug was the decision to add the value of `\parskip` to all vertical spacing parameters, even when it is used in places where no paragraph ends.
  This means that changing this parameter influences the layout in unexpected places, which in turn means that other parameters must be adjusted unnecessarily to compensate for this undesired side effect.
- There is also the problem that the `\parskip` parameter is user-accessible while the affected parameters are only changeable in a style file. The user can change the `\topsep` parameter, for example, but its default value, define d in the style file, will be restored later on.
- The resetting of parameters to their default values if nothing else is specified is somewhat arbitrary. Some of the important parameters (i.e., the penalty values for page breaking before and after the list) get their values from the surrounding list which is more than a nuisance for a style designer.[4]
- Another implementation decision makes it impossible to define lists with page-wide labels, i.e., with labels placed on a line by themselves. Those lists cannot be nested properly (see, for example, comments in the article about the implementation of the extended theorem environment [8]).

As LaTeX is used in more and more fields the need for an even more general list environment is increasing every day.

3.2 Design Limitations

It is certainly not possible to draw a sharp line between implementation disasters and design limitations. The former are problems introduced when the macros were written, the latter come from decisions or omissions made during the design phase. Nevertheless, these two phases often go with each other.

Font selection
One of these limitations was Leslie Lamport's decision to take over from plain TeX the method of selecting different typefaces: he arranged fonts in a two-dimensional grid, one dimension specifying the size, the other one the typeface. This was reasonable at the time it was implemented as there were only a few different typefaces for use with TeX. In the meantime, however, more and more fonts have become available, rendering the above design decision inadequate. For example, it is not possible for the user to switch between Knuth's fonts and the resident fonts of a PostScript printer in a transparent way.

The list environment
In spite of its generality, the list environment has some conceptual weaknesses:

- There is no possibility of generating a run-in list, i.e., a list where the first item runs into the preceding text. This feature is provided in AMS-TeX.
- More generally, since the layout of the standard list types is fixed by the document style selected, there is no way for the user to select different kinds of layouts for the same type of lists (e.g., enumerated or itemized lists) without defining his own environments. It would be better to provide a way to specify attributes such as "compact", "stream" (see, for example [17]), "run-in", etc.
- The vertical spacing before and after lists is controlled by the same parameter.
- The vertical space preceding the first item does not depend on the length of the last line of the preceding paragraph (as is the case for displayed equations).

Attribute concept
LaTeX does not allow the user to specify attributes as, for example, Script-DCF does. However, this concept is worth being considered for at least partial inclusion in LaTeX (see previous subsection).

[4]As an unpleasant result the LaTeX `fleqn` style option (which causes displayed equations to be typeset flush left instead of the usual centering) favours pagebreaks just before displayed equations. (This bug is corrected in the LaTeX version of 24 May 89.)

Text-producing arguments
The TEX mechanism for scanning macro arguments fixes the `\catcodes` of the token it reads. As a result, certain LATEX commands (such as `\verb`) cannot be used inside `\parboxes`, `\footnotes`, etc. This can be avoided using a more elaborate scheme; see for example the `\footnote` implementation in plain TEX [3, p. 363] or the article about chapter mottos [18].

Support for mathematics
In addition to the math features provided by plain TEX, LATEX offers only the `eqnarray` and `array` environments. For typesetting mathematical papers with LATEX this is certainly not sufficient. AMS-TEX provides the necessary features; however, inclusion of these into LATEX has not been done yet.[5] Instead, various people have written unpublished style options, each implementing limited subsets.

`array` *and* `tabular`
LATEX's `tabular` environment is a sophisticated tool for typesetting alignments. That means that you need not be a TEX master to "know how to make ruled tables" [3, p. 245]. On the other hand several easy-to-implement features are not provided. See for example the new implementation described in [5, 6].

Pictures
To draw picture diagrams LATEX offers mostly basic positioning commands that should actually be used internally to define high-level interfaces. Examples of such interfaces are given in [13, 16].

The output routine
The LATEX output routine is a very complex and sophisticated module that offers a wide variety of possibilities for page make-up. However, certain layout decisions, such as special placement of footnotes and floats, cannot be implemented in style files without a re-definition of LATEX's internal macros. This poses problems of compatibility. See also Section 4.

3.3 Unknown Interfaces

As we have already noted, many interfaces are not properly documented. This results in unnecessarily complicated solutions to certain layout problems. Even worse: sometimes people are led to the conclusion that there is no solution at all!

Such problems can often be solved easily by the internal LATEX macros used in the right way. As an example, consider an offset layout where all section headers are to extend into the left margin that itself is wide enough to hold them. This is provided by the generic sectioning command `\@startsection` that allows an explicit indentation parameter to be specified. If you give a negative value to it, you end up with section headers that stick out to the left.

3.4 TEXnical Limitations

In this group we gather limitations caused by the TEX program itself. Probably the most severe limitation comes from TEX's insert mechanism. After prematurely ejecting a page (so that the output routine can look at its contents), inserted material is removed from the main vertical list and gathered into certain boxes. It is therefore not possible to re-insert the page contents unchanged. This makes it nearly impossible to balance the height of several columns when insertions such as footnotes or floats are involved [9]. Other TEXnical problems arise from the way TEX inserts penalties, or breaks paragraphs into lines.

4. New Demands

When asked their opinion, many LATEX users reply:

> "LATEX is a very fine thing but this (...) and that is missing."

[5]This situation has now been remedied; the American Mathematical Society will be releasing an AMS-TEX style file which will be an option for use with any LATEX style file. There will also be versions of `book.sty` and `article.sty`, implementing book and journal styles defined by the AMS. This information comes courtesy of Regina Girouard, Composition Services Manager of the AMS –Ed.

If we ignore those requests that can be satisfied by reading the manual, we are left with three groups of wishes:

- Features that can be implemented in the current version of LaTeX. Here we have the problem that there are too few people who know the internal structure and interfaces sufficiently well to do it. These interfaces are poorly documented, and misunderstandings and misuse of these can lead to catastrophic results. Even if this is done correctly there remains the problem of compatibility of the different style options.
- Features where the necessary interfaces are missing. Here one needs to change the internal macros of LaTeX to implement them.
- Totally new requirements that lead to large-scale changes to the code or even to the concepts. Even worse: some things cannot be implemented at all because TeX itself is not able to handle them.

Instead of listing the numerous requests we will give a few examples.

4.1 Easy Implementable Features

We have already talked about the offset style problem. Other examples for requests which can be solved easily in a style file are special `pagestyles` (like the one Leslie Lamport used in his book), numbering conventions (e.g., equations within theorems ...), and special heading layouts (e.g., centered headings or `\chapter` without the word "Chapter").

A very important issue is the support of national languages. Standard LaTeX does not offer anything in this regard. However, this is easily implemented as a style option as the file `german.sty` shows. This file constitutes the German standard to which all German language sites agreed in 1987. See [12] for a description of its features.

Certain demands arise in business applications; for example, the need to stamp every page of the document with date, time, name of the input file, and possibly security information.

4.2 Features Implementable with Moderate Effort

As an example of this second group, take the new implementation of the `array` and `tabular` environments described in [5, 6]. In addition to the possibility of creating beautiful ruled tables, this new implementation allows the user to specify the layout of a column in one place.

Or consider the challenge posed by David F. Rogers in [15]. He asked for a figure placement macro that would fulfill at least three of five requirements, and states that "LaTeX's floating insert commands also do not work". This is only partly true. Two requirements are automatically fulfilled in standard LaTeX; the remaining one (numbered figures must be inserted after the first reference to the figure) can be easily implemented by changing only *one* line of code in *one* internal macro of the LaTeX output routine: in the macro `\@addtocurcol`, the call to `\@addtotoporbot` has to be changed to a call to `\@addtobot`. Of course, a style option which implements this change has to take care of the float parameters too, since their default settings tend to discourage bottom floats.

As another example take a two-column layout where all footnotes appear at the bottom of the right column. Again this can be solved by re-defining *only* the internal macro `\@makecol`.

Support for PostScript printing devices

"We have to acknowledge the importance of the *de facto* standard, POSTSCRIPT. [...] We must be aware of the way in which PostScript compatibility is crucial if we are to be taken seriously by the rest of the world" [2, p. 153]. The question of converting the `.DVI` file contents to PostScript has already been taken care of, but here we are concerned with the problem of incorporating PostScript fonts. It is easy to change `lfonts.tex` to use the fonts built into PostScript printers instead of the ones by Knuth. However, this must be done at dump time, and is therefore not selectable by the user. Instead, one needs a font selection scheme that allows dynamic switching of fonts during a TeX run, as outlined in [10].

4.3 Hard Problems

Now that we have seen that complex page make-up can be handled easily in LaTeX, one might wonder which demands (in our opinion) belong to the third group. Well, page make-up for example; *really* complex demands such as the layout used by *Scientific American*: three columns, figures spanning one to three columns with captions placed in the neighbouring column if necessary. As already mentioned, TEX's `\insert` primitive cannot be used for such a task. Doing *everything* by hand is possible (TEX is a Turing engine, as Leslie Lamport remarked) but there are limitations in space and time.[6]

As we have already noted, we think that balancing of columns belongs to the third group if footnotes are involved, despite the fact that the *TEXbook* [3, p. 417] implicitly says that this is possible. We would be only too happy to see an algorithm which did *not* break under normal (i.e., unrestricted) conditions.

5. Proposal for a New Implementation

The current LATEX version essentially consists of the following files:

`lplain.tex`	— plain TEX features used by LATEX
`lfonts.tex`	— font definitions
`hyphen.tex`	— hyphenation patterns
`latex.tex`	— most LATEX features or the internal macros to build them in a style file
`*.sty`	— document style files and document style options

The first four of these files are all loaded when a format file is dumped. This means that a large amount of TEX's internal memory is used to store the definitions contained in these files, even though there are only very rare occasions when they are all used together. If we consider adding more and more features to LATEX, we are led to ask: which parts of LATEX are essential for most document types (called the "kernel") and which are only used by special applications (called "the peripheral part").

5.1 The LATEX Kernel

Before we can talk about re-writing the kernel, we have to separate it from the peripheral parts. The following modules are considered part of the kernel:

- Basic features from plain TEX (defined in `lplain.tex`)
- Font selection
- Constants used by the rest of the program
- Program control structure
- Semi-parameter concept
- Basic error handling routines
- Spacing, line and page breaking
- File handling
- Counter management
- Cross-referencing
- Environment handling
- Basic math commands
- Generic list environment and their basic applications
- Box making commands (including `parbox` and `minipage`)
- `array` and `tabular`
- The interface for the `theorem` environment
- Generic sectioning commands
- The interface to generate tables of contents, lists of figures, etc.
- Bibliography
- Floats
- Footnotes

[6]However, we are working on this project at Mainz, hoping that we can prove that this problem belongs to the second group.

- The output routine

Some of these parts need to be revised, others must be re-implemented completely: the list environment, the writing to `.aux` files to remove the `\protect` feature (already done), the font selection code (already done), the counter mechanism (done), hierarchical references, `array` and `tabular` (done), or a conceptionally new output routine (certainly the hardest part). All the above parts should be better modularized to obtain a higher degree of flexibility.

To provide better control by means of the styles or the extensions, a number of hooks, such as `\everypar`, will have to be introduced: think of `\everylist`, `\everysection`, `\everybegindocument`, `\everyenddocument`, etc.

5.2 Peripheral Features

The peripheral parts should not be included in the format file. They can easily be moved to a number of files that are loaded on demand during a LaTeX run.

We consider the following parts to be peripheral:

- Higher math. The only features currently available are `eqnarray` and `array`. A new implementation should provide the same features as AMS-TeX, each of them loaded separately.
- New `verbatim` with unlimited size of verbatim text and a command to input verbatim text from a file (done).
- Enhanced `picture` environment (conceptionally done, partly implemented: epic, PICTeX)
- Tabbing. Improvements should be discussed, e.g., push/pop over several tabbing environments.
- Support for special draft options showing symbolic labels, index entries where they appear, time and date stamps, etc.
- Index: more exactly, the interface to an index program such as MakeIndex.

As we mentioned earlier, a LaTeX installation is complete only if it provides BIBTeX and MakeIndex. The integration of these programs, especially of MakeIndex, i.e., the interfaces, should be improved.

5.3 Document Styles

A standard mark-up concept (SGML)

For a re-implementation of LaTeX one also has to reconsider the standard document styles. As described in the introduction, there are different types of documents, e.g., books, manuals, articles, reports, letters, proceedings, etc. These types cannot be interchanged since each has its own logical concepts: letters do not possess chapters whereas there is no need for an opening or a signature in books. Given one type of document (e.g., report), there are many different ways to do the formatting. The `report` style of current LaTeX is therefore only one out of many instances of the "meta" report style.

The important point here is that all report-type styles must use the same logical concepts so that a properly written LaTeX document is independent of the specific style instance used. A German site (say, the University of Mainz) may decide to provide a special style (called, say, `uni-mainz-report`) to format reports according to its own conventions. But a report written at Stanford can then be typeset at Mainz using this style *without* changing the LaTeX input file — but of course the document will come out with different spacing, etc.

Therefore an important task of a new implementation is to reconsider the logical concepts used so far, and then decide to drop or change some, or to add new ones. For example, currently there isn't any difference between the prototype styles `report` and `book`. Obviously this cannot be correct. This is of course a question being debated by the experts.

International language support

There is one point to make here: from what we have said above one must not infer that the textual representation of headers (e.g., "Contents", "Litteratura") must be fixed by the style. In this respect we disagree with Leslie Lamport. To the contrary: since it may well be that a site wants to typeset documents written in different languages, but all in the same style, it is only reasonable to provide a way to change header names. Actually we propose to make this a logical concept of the specific meta style. This means that there must exist commands such as `\chaptername` and `\refname` that can be changed by the document (using `\renewcommand`) or by style options. It is perfectly allowed (though not necessarily reasonable) for a specific style to ignore these commands, and to typeset all headers in the same way.

6. Institutional Considerations

Leslie Lamport has copyrighted LaTeX and fixed the status quo. While this is the best way to ensure uniformity over a large group of installations and users, one runs the risk that further developments and enhancements will become incompatible. Therefore our proposal for a new implementation is bound to fail if it becomes only one more among many others (with only a new name). We feel that such a project should only be undertaken if it is supported by an institution which can channel future developments.

6.1 Maintenance

From the size of the LaTeX source code, it is clear that it must be maintained. This not only means that there must exist a person who will correct any bugs found; this is only part of the story, and not necessarily the most important one. Another necessary task is to develop the software as new demands arise. Software design is a difficult job. It is even more difficult to revise design decisions made earlier, because in addition to software development aspects, one has to consider questions of compatibility.

For the maintainance of a package the size of LaTeX, one needs a group of people who can respond to demands and wishes, and decide what can and should be done. This means that this group must include document style as well as LaTeX specialists. These people need not necessarily be the implementors themselves. They have to set the standards, not write the macros. Nevertheless, they need to be sufficiently familiar with LaTeX.

6.2 Suggestion for a LaTeX Standardization Group

Certainly this is only possible if Leslie Lamport is willing to share control over LaTeX. Another important point is to guarantee that this group continues to perform its task even if the individual members change. The logical point to organize this would be the TeX Users Group. We think that TUG should form a committee to discuss these problems.

7. Update

During and after the conference at Stanford, discussions were held with Leslie Lamport concerning the issues raised in this paper. From these it became clear that he originally expected LaTeX to be superceded by some other document preparation system within a few years of its appearance. This has not happened and now he too sees a need for its further development.

He is in agreement with the principles contained in this paper concerning the future of LaTeX but does not wish to be directly involved in their implementation. During the discussions the following two-stage development plan was suggested:

1. Re-design the style file interface and document it: this would involve the publication of a manual on the design of style files. The resulting LaTeX version would be 2.10. This version may contain some minor enhancements visible to the user, but every input file that uses only features documented in the current LaTeX manual would work with version 2.10.
2. Produce a completely new implementation of LaTeX, version 3.0, according to the principles outlined in this paper. This would be upwardly compatible in the sense that it would be possible to process any existing document with the addition of a style option.

The timetable for this work cannot be fixed at present since it is not yet clear how much of our time we shall be able to spend on this project, nor what support will be forthcoming from various parties.

Bibliography

[1] Bartlett, Frederick H. "Automatic Page Balancing Macros Wanted." *TUGboat* 9(1):83, 1988.

[2] Clark, Malcolm. "International Standards and TeX." *TUGboat* 10(2):153–156, 1989.

[3] Knuth, Donald E. *The TeXbook. Computers and Typesetting* Vol. A. Reading, Massachusetts: Addison-Wesley, 1986.

[4] Lamport, Leslie. LaTeX: *A Document Preparation System. User's Guide and Reference Manual.* Reading, Massachusetts: Addison-Wesley, 1985.

[5] Mittelbach, Frank. "A new implementation of the array- and tabular-environments." *TUGboat* 9(3):298–314, 1988.

[6] Mittelbach, Frank. "A new implementation of the array- and tabular-environments — addenda." *TUGboat* 10(1):103–104, 1989.

[7] Mittelbach, Frank. "The doc option." *TUGboat* 10(2):245–273, 1989.

[8] Mittelbach, Frank. "An Extension of the LaTeX theorem environment." *TUGboat*, 1989 [forthcoming].

[9] Mittelbach, Frank. "An environment for multi-column output." *TUGboat*, 1989 [forthcoming].

[10] Mittelbach, Frank, and Rainer Schöpf. "A new font selection scheme for TeX macro packages — the basic macros." *TUGboat* 10(2):222–238, 1989.

[11] Mittelbach, Frank, and Rainer Schöpf. "A new font selection scheme for TeX macro packages — the LaTeX interface." *TUGboat*, [forthcoming].

[12] Partl, Hubert. "German TeX." *TUGboat* 9(1):70–72, 1988.

[13] Podar, Sunil. Enhancements to the Picture Environment of LaTeX. Dept. of Computer Science, S.U.N.Y. at Stony Brook, Technical Report 86-17, version 1.2, July 14, 1986.

[14] Price, Lynne A. "SGML and TeX." *TUGboat* 8(2):221–225, 1987.

[15] Rogers, David F. "A Page Make-up Challenge." *TUGboat* 9(3):292–293, 1988.

[16] Schöpf, Rainer. "Drawing histogram bars inside the LaTeX picture environment." *TUGboat* 10(1):105–107, 1989.

[17] Wonneberger, Reinhard. "Stream lists and related list types for LaTeX." *TUGboat* 6(3):156–157, 1985.

[18] Wonneberger, Reinhard. "Chapter Mottos and Optional Semi-Parameters in General and for LaTeX." *TUGboat* 7(3):177–185, 1986.

[19] Zalmstra, Joost, and David F. Rogers, "A Page Make-up Macro." *TUGboat* 10(1):73–81, 1989.