BIJLAGE Q

The Document Style Designer as a separate Entity

Victor Eijkhout & Andries Lenstra

Abstract

An argument for the need for a programmable meta format: a format that introduces a new syntactic level in TEX for document style designers.

TEX has a number of characteristics that set it apart from all other text processors. Its unsurpassed quality of text setting and its capabilities for handling mathematics are some of the more visible aspects. On a deeper level, however, the extreme programmability of TEX is just as big an asset. Any layout can be automated to an arbitrary extent. (It is strange that *The TEX book* gives almost no hint of this.)

The form such automation usually takes is what is called 'generalized markup'. The person who keys in the text has at his or her disposal commands that describe the logical structure of the document, and as little as is possible of the visual structure.

Document styles as they appear in LATEXare examples of this. Here the layout is not merely automated, it is completely hidden from the end user. In particular, the same input can produce widely different output by letting it be interpreted by different styles.

With LATEX, however, the problem is the production of document styles. This is a major task, and consequently most people use either the standard styles, or minor modifications of these. Furthermore, LATEX does not offer sufficient tools to produce even moderate variations on the layout of the standard styles.

For scientific use of TEX one can become reconciled to this situation. A scientist should be more concerned with the contents than with the looks of a document, so if there is a format that offers all the tools to get those contents across to the reader, the visual appearance is of secondary concern. We may conclude that, for scientists, LATEX fits the bill.

When a document is not a scientific article, however, the inflexibility of LATEX renders it useless. The alternative would seem to be plain TEX, but there the objection is the long and slow learning curve.

One way out of this dilemma is the 'front end to TEX' approach taken in *The Publisher* from Arbortext and *Grif* from Gipsi. Both systems present almost a 'wysiwyg' (what you see is what you get) interface to the user (the term 'wysipn', what you see is pretty neat, has been used), and allow altering style parameters via dialog boxes and menus. In both cases, however, programming the basic style structure and appearance is still less than trivial.

In this article we describe an approach which brings down the complexity of programming a style to that of using it. We propose a front end programming language for style design, which is itself implemented in TEX.

The 'Checklist' Approach to Style Definition

If one compares TEX to mouse-and-menu text processor packages, one runs into two basic characteristics of programming that constitute a disadvantage when learning TEX.

The obvious first point is that TEX has a *syntax*. Every TEX programmer knows that you can have no end of fun with missing or mismatched braces. Mouse riders are not bothered by this. One can, at most, click the wrong item, but one cannot click in the wrong way.

The second point is more subtle: programming involves and imposes algorithmic thinking. Consider the ordinary loop statement

```
for $i$:=$1$ to $n$
```

In many cases, all that is meant is

```
for all $i$ between $1$ and
        $n$ inclusive
```

Thus, the syntactic formulation contains a sequentiality that may semantically not be present. Similarly, the statements in a macro definition are sequentially ordered, even when the corresponding actions are in no such relation.

But even when actions are sequentially ordered, it should not always be necessary for a style designer to specify them in that same order. For instance, in the design of a list environment the amounts of white space above and below the list, and the amount of indentation of the list, should be specifiable in any order, even though they correspond to sequentially ordered actions. Also, the decision whether or not to break a page above a certain type of heading should be specifiable at any point in the definition.

Such actions do not really correspond to design decisions, rather they are the specific incarnations of general parameters and switches. Thus, it would be a valid approach to style design to offer the person implementing the style a small number of basic constructs (these could for instance be headings, lists, and page layouts), each of which has a 'checklist' of parameters and switches controlling the final layout.

Abandoning sequentiality, and making most specifications optional with default values, is then a sensible way to facilitate TEX programming. An implementation of these demands could take the form of lists of keyword-value pairs. Such lists can be presented in extremely simple syntax, and as matching is performed by keyword instead of by position, such an approach would meet some of the criticism of algorithmic specification above.

As an example, the layout of unnumbered section headings could be given in Lisp 'property list' syntax as,

```
\defineheading:section
   (white_above 6pt plus 2pt minus 1pt)
   (white_below 6pt plus 2pt minus 1pt)
   (caption (size 12pt) (style bold)
            (shape ragged_right))
```

but any other syntax is just as valid. It is advisable, however, to steer clear of the idiosyncrasies of the pure TEX syntax.

Metaformats

Checklists may capture a large part of the variation in basic constructs, but for any set of parameters there will be a layout that eludes classification in terms of these parameters. Thus, it seems inevitable that a style implementer needs to do some programming. However, it is possible to make a very smooth transition between marking a checklist and programming macros. For this, we need the distinction between formats and 'metaformats'.

Let us denote by the term 'format' any collection of macros that gives the end user commands of a higher level than those of pure TEX. By 'metaformat,' we will mean a format such that the commands for the end user are in majority not defined in the format. Metaformats offer the tools with which a style implementer constructs the commands for the end user.

In a restricted sense, the LATEX format is a metaformat. A good example here is the `\@startsection` macro, which is basic to LATEX, and in terms of which commands like `\section` are defined in the style files. The parameters of this command are mainly numeric parameters determining the layout. One parameter functions as a switch.

As another example, the `\list` command, which is the basis for various environments in LATEX, offers the style designer the possibility for programmable extension. Such extensions are specified by passing a piece of TEX code as the second parameter, that is, LATEX does not really provide the style implementer with a separate syntactic level.

Calling LATEX a 'parameterized metaformat,' we can also envision 'programmable metaformats.' There is no objection against implementing a new programming language in TEXwhich would be easier to learn and to use.

The challenge is then to find primitives that allow formulation in a simple syntax and that are sufficiently powerful for producing a wide range of layouts.

By any other name . . .

One choice for the primitives of a metaformat is immediately clear: the boxes and glue of TEX itself. Any TEX programmer knows that you can do all typesetting with boxes and glue, so why not give them to a style designer? The problem of course is how to simplify their declaration. It is here that the writer of the metaformat takes certain decisions.

Consider, as an example, section headings such as they appear in the LATEX article style.

1. Section title
1.1. Subsection title

These take the form (but not the implementation) of two boxes on the same line: one containing the number, the other containing the text. Also the headings of the 'artikel' style can be described thus.

1.   Section title
1.1. Subsection title

But the box of the number then has a prescribed width. In both cases, however, the sum of the width is the text width. We can therefore imagine that these two layouts were specified as

```
\defineheading:section
    (inline
       ((value sectionnumber)
        (spaces 2))
       (title))
```

for the standard LATEX heading, and

```
\defineheading:section
    (inline
       ((value sectionnumber)
        (FillUpTo labelwidth))
       (title))
```

Here the metacommand 'inline' is just an `\hbox to \hsize` in disguise. It takes all boxes and sets them at natural or specified width, and the width of the box containing the actual heading is calculated to fill the remainder of the text width.

Headlines and footlines provide a nice example of how a programmable metaformat can make intelligent use of

TEX's glue. Like section headings, footlines are a disguised `\hbox to \hsize`. A footline with just a left aligned page number can be specified like

```
(footline (value pagenumber))
```

where the format supplies a trailing `\hfil` to prevent an underfull box. Cases where the number should be right aligned can be specified as

```
(footline (whitespace fillerup)
          (value pagenumber))
```

where the 'whitespace' is an `\hfill`, squashing the trailing `\hfil` at the end of the line.

A syntax and instruction set, such as sketched in these examples, tax the programming capabilities of the format designer, but not those of the style designer. In effect, the format designer implements a new programming language on top of TEX with a simple syntax, a small instruction set, and at the same time, sufficient generality to produce a wide range of layouts.

Obviously, a smaller instruction set makes learning the format easier. Another advantage is less immediately clear: it reduces the chance of errors. More compact instructions will likely have a more defined function; so on the one hand the style implementer need not specify those actions that must be taken anyway, and on the other hand the format can perform some consistency checking on the intentions of its user.

Conclusion

We have argued the need for a 'programmable metaformat:' a format that introduces a new syntactic level in TEX for document style designers. Such a level should probably have a different syntax from pure TEX, and it should contain a relatively small instruction set in which the actual user macros are written. We have indicated that elements of such a format can, to a large extent, be captured in 'checklists.' Others can take the form of intelligently disguised TEX primitives. We believe that a format incorporating these principles is possible, and that it can be taught to people with little knowledge of TEX.

As an illustration of these ideas we present the style definition of this article in the 'Lollipop' format.

Example

The following piece of code is the style definition for this article given in the 'Lollipop' format.

```
\typeface:Computer
\fontsize:10 \fontstyle:roman

%declare \parindent
\distance:indentation=20pt
%white space around text elements
\distance:whiteline=6pt plus 2pt
         minus 2pt
\distance:leadingwhite=whiteline
\distance:trailingwite=0pt

%lots of defaults used here
\defineheading:section
         fontstyle:bold stop

%very simple page layout
\definelayout:twocolumn
    height:22.5cm width:16.5cm
    band:start column
         whitespace:0.6cm
         column
         band:end
    stop
\twocolumn %install output routine

%paragraph shape
\defineparagraph:flushright indent:yes
    rightjustified:yes stop
\flushright %install paragraph shape

%bibliography
\definelist:literature counter:1
    item:left
         litteral:[ value:itemnumber
         litteral:]
    item:end stop
\externalreferences:yes
```

Bibliography

Braams, Johannes, Victor Eijkhout, and Nico Poppelier, "The development of national LATEX styles," *TUGboat*, Vol 10(4), 1989.

Grif manual / Les languages de Gipsi, Gipsi 1989

Knuth, Donald E. *The TEX book*, Addison-Wesley Publishing Company, 1984.

Lamport, Leslie. *LATEX, a document preparation system*, Addison-Wesley Publishing Company, 1986.

BIJLAGE R

The Dutch national LATEX effort

Johannes Braams, Victor Eijkhout & Nico Poppelier

Abstract

In this article an overview is given of the activities of working group 13 (WG13) of the "Nederlandstalige TEX Gebruikersgroep" (Dutch TEX Users Group). This working group is also called "Neerlandica", and is interested in anything that has something to do with using LATEX (and TEX) in a non-American environment. The topics tackled so far range from the design of a page layout suitable for A4 paper by adapting the American layout of `article.sty` to Dutch typographical tastes, to the implementation of a new letter style called "brief".

Introduction

At its founding meeting the NTG decided to establish a number of "working groups", which would tackle some of the problems encountered by members of the NTG. Some of the subject are "education", "drivers", "TEX for personal computers". At the second NTG meeting another working group was started. The task of this group was to investigate the TEXnical problems Dutch-speaking LATEX users encountered and to suggest solutions to these problems.

The first activity of this group was to decide on which problems would be tackled and in what order. We had a number of subjects:

- The original LATEX document styles are designed for American-sized paper. The dimensions of this paper differ from A4 paper, which is used most commonly in Europe. The available style option files at the time were not very satisfactory.
- In Dutch texts, commands that produce text, like `\chapter` or `\abstract`, should produce *Dutch* texts instead of *English*. A way to solve this problem had already been pointed out by Hubert Partl *et al* [5].
- The design of the standard document styles provided by Leslie Lamport is very "American" and, at least to our Dutch eyes, a bit "loud". We decided to develop replacement styles that would be more adapted to Dutch typographical standards.

After identifying these tasks we set ourselves a short-term and a long-term goal. The short-term goal was to provide the members of the NTG with an acceptible page-layout, adapted to A4 paper and to provide a document-style option that redefines commands like `\chapter` to use macros like `\chaptername`. This seemed a reasonably simple task that would be welcomed by a lot of Dutch LATEX users. The long-term goal was to develop new document styles, plug-compatible with the styles provided by Leslie Lamport. To this we added the need for the implementation of a document style for letters that follows Dutch guidelines for the design of letters.

In the following sections we will discuss these topics in more detail. Some of this work has led to articles published or to be published in *TUGboat*.

Page layout for A4 paper

As described before one of the first problems tackled was that of adapting the standard document styles to A4 paper. The solution we sought was to provide a document style option file, and not to modify document styles for this. This was done mainly to avoid having to maintain extra document styles that differ only marginally from the originals.

Obviously, we were not the first LATEX users to identify the problem and we knew that document-style options were available in the international TEX community. However, as we were not satisfied with the results these options gave us, we decided to adapt them. This has led to the file `A4.sty` described in [9]. This file started as a combination of two options that were already available. The combination of these two files didn't satisfy our demands, which were:

1. The width of the text should be chosen in such a way that no more than sixty to seventy characters appear on a line of text;
2. When a document is printed two-sided, the texts printed on both sides of one sheet of paper should overlap;
3. The "inner" margin of the document should be wide enough to allow for the binding of the document;
4. The "outer" margin should be wide enough for marginal notes.

After the publication of our article on `A4.sty`, we received a comment that it doesn't handle texts with more than one column correctly. We still have to look into this, but it is a nice example of Gödels principle [11][1].

The "loud" American design of `article.sty`

As discussed before, the modification of the design of the standard document styles was a long-term project. As a first attempt a document style option "`sober.sty`" was developed. This file modifies the amount of white space around section heads and lists. It also modifies the fonts used for the various levels of section heads.

A more fundamental approach was the development of new document styles, derived from `article.sty` and `report.sty` [7]. The design of these "new" styles was based on discussions with a Dutch typographical designer and some books about typographical design [3, 4]. One of the ideas behind the redesign of `article.sty` was to minimize the number of "implied left margins". By an implied left margin we mean a non-zero distance from the actual left margin that is used in more than one element of the document. Examples of implied left margins are:

- the paragraph indentation,
- the left margins of items in an "itemize" or "enumerate" list construct,
- the left (or right) sides of the numbers and labels in such list constructs and
- the left side of the text of a section heading.

In the standard styles of LaTeX all of these four distances are independent and are different from one another. In the style we have developed it was decided to strengthen the visual coherence of the layout by taking the same value for each of them whenever possible.

Another idea that was implemented with `artikel1.sty` is that the white space separating a section heading and the text following it should bear some simple relation to the baselineskip, and should not have any stretch.

A third major modification in the design of the article style is a new layout of the table of contents. Both Treebus [3] and Miles [4] are very explicit in their opinion about the layout of a table of contents. They find a layout like the one implemented in `article.sty` "old fashioned" and even confusing. So, a completely different layout was implemented.

As the new layout of the table of contents met with some reservations from users, the old layout is still available through an option.

Besides `artikel1.sty` we also implemented two other article styles that have a layout that differs from the layout of `artikel1.sty`. In `artikel2.sty` we implemented a layout that was designed to show what can be accomplished by modifying a few parameters in a document style. The third article style is like `artikel1.sty`, but with one major design decision changed: in `artikel3.sty` paragraphs are not indented, but they are separated by vertical space.

The design of the document styles `artikel1` and `artikel3` is also implemented for reports, as `rapport1.sty` and `rapport3.sty` respectively, and there is a `boek` style based on the design of `artikel1`.

Modifying the standard styles without modifying them

As discussed in the introduction, one of the topics WG13 should be working on was a solution to the problem of English terms appearing in Dutch texts. The basic idea behind our first solution was already implemented in `german.sty` [5]. We adopted the idea and created `dutch.sty`. In some ways this file is much simpler than `german.sty`, in another way it is more complex. A major difference between the two files is that `german.sty` just provides parameters and parameter values for the various terms and states in the comment that the user should provide modified document style files that make use of these parameters. The file `dutch.sty` also contains the necessary redefinitions for the various LaTeX macros. This implies that `dutch.sty` can be used with the standard *unmodified* document style files as they are included in every LaTeX distribution.

While implementing `dutch.sty` it occurred to us that it had some code in common with `german.sty`. This, combined with several discussions at the 1989 EuroTeX meeting in Karlsruhe led to the idea of building a more universal system of style option files, called the babel-system [10].

The babel system consists of one file, with macro definitions common for all languages and a language-specific file for every language that is to be part of the system. It offers the possibility to switch between languages while processing a multilingual document. Because we wanted this system of style option files to be compatible with the original `german.sty` the files are implemented in such a way that they can also be used with plain TeX. This is useful because the language-specific files can (and do) contain more than just parameters for LaTeX terms. For instance, for the German language as well as for the Dutch language we have an extra active character. This active character is, among other things, used for controlling the hyphenation of words containing accented letters or explicit hyphens by inserting `\discretionary` commands.

All the user needs to specify is the (main) language used in his document as an option to the `\documentstyle`

[1] See the chapter about the *Contracrostipunctus*.

command. This will instruct TeX to read the appropriate language-specific file. This file checks whether the core of the babel system, `babel.sty`, has been read before. If this turns out not to have happened it `\inputs` `babel.sty`.

When the user wants to add the definitions for another language to the environment in which his document is processed he can use the command `\addlanguage`[2] with the name of another language-specific file as an argument. So the preamble of his document might look like:

```
\documentstyle[11pt,dutch]{artikel1}

\addlanguage{germanB}

\begin{document}
```

So, this example document contains Dutch and German texts. Because the function of the extra active character (`"`) is different for `dutch.sty` and `german.sty` the user wants to switch this definition when he starts a German part of the document. This can be achieved by adding

```
\selectlanguage{germanB}
```

in front of the German text. When he wants the Dutch settings to be restored he simply uses the same command with the parameter `dutch`.

As the babel system has been developed in a pre-TeX 3.0 environment, it doesn't use any of the features of TeX 3.0. Perhaps the switching of hyphenation tables and maybe other TeX 3.0 features might be added to the definition of `\selectlanguage`.

The design of a new letter style

In one of its meetings the "Neerlandica" group decided to try to implement a document style for letters that should conform to Dutch standards for the layout of letters. We have consulted four such standards, from the Dutch standardization institute, the Nederlands Normalisatie Instituut (NNI). They were:

- NEN-1026 for letters,
- NEN-1025 for envelopes,
- NEN-3162 for the structure of documents and
- NEN-3516 for the design of forms.

The result is a design that can not be judged by taste: it just implements the standards. This seems quite rigid, but some freedom is left to the user to adapt certain parts of the layout to his own wishes.

This new style is *not* "plug compatible" with the LaTeX `letter` style, although equivalents of some of the macros from `letter.sty` have been provided. The main reason for its "incompatibility" with `letter.sty` is that we have quite a lot of new user commands to either modify some parts of the layout or to fill in some of the fields in the "reference lines". The reference lines contain fields such as "Your letter of" and "Date".

Some of the features of this document style are:

- If the user doesn't have printed letter paper he can provide his own letterhead by writing his own macro `\briefhoofd` or he can use the macro `\maakbriefhoofd` to adapt the default letterhead provided with the document style.
- An option is provided to print short horizontal rules on the sides of the paper as an indication where to fold the letter.
- The address is positioned such that it can be used with "window envelopes". The window can be either on the left side (default) or on the right side of the envelope.
- Provides a user-command `\voetitem` for providing information about the sender at the bottom of the letter.

An example of this document style can be found at the end of this article.

Conclusion

Working group 13 of the NTG has had a busy year. We have

- produced a number of replacement document styles for the standard LaTeX document styles,
- developed a new document style for letters that implements Dutch standards for the layout of letters,
- presented a new and improved document style option file for use with A4-size paper and
- produced a document style option file system that supports multiple languages in one document and provides additional features for specific languages.

In short: the category code of this working group has been `\active`.

References

[1] Donald E. Knuth, *The TeXbook*, Addison-Wesley, 1986.

[2] Leslie Lamport, *LaTeX, A document preparation System*, Addison-Wesley, 1986.

[3] K.F. Treebus. *Tekstwijzer, een gids voor het grafisch verwerken van tekst*, SDU Uitgeverij ('s-Gravenhage, 1988). A Dutch book on layout design and typography.

[4] John Miles, *Ontwerpen voor Desktop Publishing, Alles over layout en typografie op de personal computer*, Uitgeverij Mingus, Baarn 1988, Dutch translation of *Design for Desktop Publishing: a guide to layout and typography on the personal computer*, Fraser, London 1987.

[2] `\newlanguage` would be more appropriate, but this has become a TeX primitive, so the macro was renamed.

[5] Hubert Partl, *German TeX*, *TUGboat* 9 (1988) #1, p. 70–72.

[6] Leslie Lamport, in: TeXhax Digest, Volume 89, #13, 17 februari 1989.

[7] Johannes Braams, Victor Eijkhout and Nico Poppelier, *The development of national LaTeX styles*, *TUGboat* 10 (1989) #3, p. 401–406.

[8] Joachim Schrod, *International LaTeX Is Ready To Use*, *TUGboat* 11 (1990) #1, p. 87–90.

[9] Nico Poppelier and Johannes Braams, *A style option to adapt the standard LaTeX document styles to A4 paper*, *TUGboat* 11 (1990) #1, p. 98–103.

[10] Johannes Braams, *Babel, a multilingual style-option system for use with LaTeX's standard document styles*, *To be published in TUGboat*.

[11] Douglas R. Hofstadter, *Gödel, Escher, Bach: een eeuwige gouden band*, Uitgeverij Contact, Amsterdam 1985, Dutch translation of *Gödel, Escher, Bach: an eternal golden braid*, Basic Books, New York 1979.