Paradigms: Parameterization I

Options

Kees van der Laan

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

1 BLUe's Design IV

Hi folks. Much parameterization in TEX has already been taken care of via the various modes of TEX, that is by the states script elements can be processed in. How ingenious and handy that may be, it is not sufficient.
Knuth also provided switching commands for fonts, formats and language.[1]
Then there are the various parameters—integer, dimen, glue, muglue—as listed in *The TEXbook* 272–274.

Markup tags can be parameterized via (at most 9) arguments. However, there exist also for this purpose (global) token variables—the `\every<tag>`s—with `\output` and `\errhelp` as special cases.

To remind the reader of Knuth's `\every`s, I have enumerated them below.

```
\everypar
\everymath, and \everydisplay
\everyhbox, and \everyvbox
\everyjob
\everycr
```

For using them, it is important to know where the token variables are precisely inserted.[2]

For my taste Knuth's `\every<tag>`s have not been appreciated as they should have been, the more so when extended by the analogons `\this<tag>`s. This in contrast with the abundant use of arguments, especially the so-called optional arguments.

1.1 Why?

The purpose of this note is to show that optional arguments have led to cumbersome TEX coding, while the same functionality can be attained more easily via the use of `\this<tag>`s. In general one can devote a monograph to markup language parameterization, I guess.

2 Examples of use of `\every`s

I looked through the *The TEXbook* for examples of use by Knuth himself.

2.1 `\everypar`

An example of use is given in Appendix D 381, about verbatim listings with line numbers. In blue.tex I enriched this approach by allowing selective line numbering. I mean by the latter that I can number parts, starting with any suitable number. This is handy when macro sets are accompanied by a table of contents (as should always be the case), which contains references to the line numbers of the macro set. From blue.tex the following.

```
\def\setupverbatim{\makeactive\'%
  \let\!=!\makeescape\!%Knuth&Levy
  \def\par{\leavevmode\endgraf}%381
  \obeylines \uncatcodespecials
  \obeyspaces}
%
\def\numvrb{\vrblin0
   \everypar{\advance\vrblin1
    \llap{\sevenrm\the\vrblin\quad}}}
%
\def\nonum{\everypar={}}
!endverbatim
%
```

Note that the line numbers can be adjusted via modifying the counter `\vrblin`.

Another example is provided on 393, in Paragraph maneuvers. It is about automatically inserting `\hangindent` and `\hangafter`.

In manmac it is used in

- `\endchapter`, as `\everypar{\sl}`, to set the quotations slanted.
- `\beginlines`, as `\everypar{\strut}`, to insert a strut.

2.2 `\everydisplay`

This is used in the solution of exercise 19.4 to obtain non-centered displays.

```
\def\leftdisplay#1$${\leftline
 {\indent$\displaystyle{#1}$}$$}
\everydisplay{\leftdisplay}
```

This is refined in Appendix D 376, to allow for equation numbers as well. In 'Math into BLUes' I have worked on the Appendix D version, and also indicated how a single formula can be non-centered. This is also included in blue.tex.

[1]For example `\eightpoint`, `\report` (well in blue.tex), and `\language1`. Similar to the font switching macros I used `\english`, `\dutch` and the like, to activate all language dependent parameters for typesetting bridge with the right words. Early LATEX had several words 'hardwired' in the code in English. The latter was the reason for Johannes Braams to give birth to Babel, to parameterize and concentrate language specific issues, and to allow easy switching from one language to another.

[2]For example, in a setbox the right-hand side of `\afterassignment` is inserted after the opening brace of the box and followed by the tokens of `\everyhbox` or `\everyvbox`.

2.3 `\everyjob`

The example is exercise 24.5. In Salomon's courseware an example is given to open automatically files at the beginning of each TEX job.

2.4 `\everycr`

A practical example is given in *The TEXbook* 140, to inhibit a page break.

```
$$\everycr{\noalign{\penalty10000}}
\halign{\indent#\hfil\qquad&...\cr
If a letter is in style&then...\cr
\noalign{\vskip 2pt}
$D,D',T,T'$&text size&\cr
$S,S'$&script size&\sevenrm\cr
$\SS,\SS'$&scriptscript size&
               \fiverm\cr}$$
```

It is also used in plain, Appendix B 362, in the macro `\displ@y`.

in *The TEXbook* file I did not find examples of use of `\everymath`, `\everyhbox`, and `\everyvbox`.

In blue.tex I introduced `\everyscript`. My use is to allow for more than one script to be processed, via `\everyscript{\notlastscript}`. I also introduced `\everyverbatim`, and some more.

3 Options via `\this`< *tag* >s

First an example. I used it for the first time with verbatims.

```
\thisverbatim{\emc}%enable metacode
\beginverbatim
\def\⟨tag⟩{...
   ...}
!endverbatim%! is the escape character
```

Not only can metalinguistic variables be handled nicely within verbatims, but also the changing of catcodes and file verbatim inclusion go easy with the use of these token variables.

The verbatim suite of blue.tex can be used with (LA)TEX, because the mechanism is simple, and not in conflict. File verbatim inclusion goes as follows[3]

```
\thisverbatim{\input ⟨file⟩}
\beginverbatim
Some text after the file.
!endverbatim
```

4 The parsing of options

Eijkhout in 'TEX by Topic' gives a template for coping with optional parameters.[4] The work is done by `\OptArgCom`, with as optional argument either the default or the one supplied. The markup starts with `\Com` followed either by a left bracket—a convention to start an option—or the argument.

```
\def\Com{\futurelet\testchar\MaybeOptArgCom}
\def\MaybeOptArgCom{\ifx[\testchar
   \expandafter\OptArgCom\else
   \expandafter\NoOptArgCom\fi}
\def\OptArgCom[#1]#2{...}
\def\NoOptArgCom{\OptArgCom[⟨default⟩]}
%with use
\Com[...]{...}    or   \Com{...}
```

With the concept of `\thisCom` the coding template and markup might look like the following.

```
\thisCom{⟨default⟩}
\def\Com#1{...\the\thisCom...
   \thisCom{⟨default⟩}}%restore default
%with use
\thisCom{...}%Provides option(s), if any
\Com{...}
```

Explanation

The contents of the token variable is available to the macro. It can be virtually anything. At the end of the macro the default is restored. IMHO, with all respect, the latter approach is simpler than the parsing of optional arguments.

5 Head example

blue.tex allows as markup

```
\beginhead...\endhead     or   \head{...}
```

And what about the coding? In the 'Paradigms: Headache?' I have shown the blue.tex coding

```
\def\beginhead{\the\prehead\bgroup\headfont}
\def\endhead{\egroup\the\posthead}
%with auxiliaries
\prehead{\vskip0pt plus2ex
   \penalty-250\vskip0pt plus1ex
   \bigskip\noindent}
\posthead{\medskip\nobreak
   \noindent\ignorewhitespace}
%and minimal variant
\def\head#{\beginhead\bgroup
   \aftergroup\endhead
   \afterassignment\ignorespaces
   \let\dummy=}
```

Note that in blue.tex, I also introduced the token variables `\pre<tag>`, and `\post<tag>`, to parameterize the placement within context. The advantage of doing so is that these tags have only one function and can therefore be customized easily.[5]

5.1 Relation with tugboat.sty

Submissions for TUGboat need as markup for headings

```
\head...\endhead     or   \head*...*
```

where in the short variant the spaces around the *s are neglected. TUGboat provides as toplevel coding

```
\def\head{\begingroup
   \def\CurrentTag{head}%
   \@allowindentfalse
   \@defaultoptions
   \@savingargumenttrue
   \def\\{\break}%
   \@checkoptions}
```

[3] It is true, that I need one escape character.

[4] I adapted the macros a little.

[5] A white lie. In `\report` format I also reused the headtitle in the ToC, ToE, and in the running header.

```
\def\endhead{\endgraf
   \ifcase\headlevel\or
      \@domainhead  \or
      \@dosubhead   \or
      \@dosubsubhead\fi
   \endgroup\@next}
```

What can be seen from the above is that both codings are based on two-part macros. The coding for TUGboat uses general mechanisms not restricted to `\head`, and is therefore difficult to understand.[6] Especially, when one realizes that next to the parsing of options, the minimal variant is also handled. (The parsing looks for *s). Clever, very clever. But the functionality can be attained simpler, with the extra bonus of easy maintenance and customization, IMHO, with all respect.

5.2 Relation with ams.ppt

Submissions to AMS (in ams.ppt) need as markup for headings

```
\head...\endhead
```

The coding reads

```
\outer\def\head#1\endhead{%
   \add@missing\endroster
   ...
   \add@missing\endproclaim
   \penaltyandskip@{-200}\aboveheadskip
   {\headfont@\raggedcenter@
    \interlinepenalty\@M
    #1\endgraf}\headmark{#1}%
   \nobreak\vskip\belowheadskip}
%
\let\headmark\eat@
```

The `\end@missing` checks whether the head occurs within the environment. `\eat@` gobbles its argument. There is no `\nofrillscheck`. This is done, however, in `\subhead`. The coding of the check is difficult to understand, not in the least because it allows for options.[7]

5.3 LaTeX's `\section`

The style defines `\section` as an invocation to `\@startsection`. From latex.doc the following.

```
%\@startsection{name}{level}{indent}
%  {beforeskip}{afterskip}{style}
%  optional * [altheading]%
%  {heading}
%Generic command to start a section.
%Name  : e.g. subsection
%Level : a number, denoting depth of
%        section e.g., chapter=1,
%        section=2 etc.
%Indent: Indentation of heading from
%        left margin
%Beforeskip: Absulute value is skip to
%            leave above the heading.
%            If negative, then paragraph
%            indent of text following
%            heading is suppressed.
%Afterskip : if positive then skip to leave
%            below heading, else negative of
%            skip to right
%            of run-in heading.
%Style     : commands to set style.
%If * missing then increments the counter. If
%it is present, then there should be no
%[altheading] argument. Use the counter
%‘secnumdepth’ whose value is the highest
%section level that is to be numbered.
```

This is just the top of the ice-mountain, but generic it is for sure. My choice would be the following. To go first for making it as simple as possible, and perhaps if the need is still there, use generic coding in order to save on development and maintenance costs.

6 Knuth's options after begin tag

There is a beautiful example of allowing for options in manmac. It is used in *The TeXbook* 29. All what follows after the opening tag `\begindisplay` up to the end-of-line is taken as (optional) argument, and inserted in the alignment display between $$ and `\halign`.

```
\begindisplay\hbadness10000
\hbox spread-.666667em{The badness
   of this line is 100.}&
     \quad(very tight)\cr
...
\enddisplay
```

This example shows what Knuth had on his mind with respect to options and explains why he did not introduce `\this<tag>`s. The coding is a litlle more complicated but systematic.

Can it be applied applied throughout a format? With `\beginverbatim` I stumbled upon problems in applying options to the in-line verbatim `|`...`|`.

7 Epilog

The paradigm is that too general, monolithic, codes are difficult to understand and to maintain. Borrowing macros from these collections is near to impossible. `\this<tag>` is a simple alternative for coping with optional arguments, and is used in blue.tex.

Knuth suggested a solid naming convention for two-part macros: `\begin<tag>`, `\end<tag>`, and for the second step macro `\start<tag>`, if any.

This naming convention has been adopted in blue.tex, next to `\<tag>` for one-part minimal variants.

Have fun, and all the best.

[6] An example is finding the answer to the question whether the ‘argument’ is processed on the fly. `\@savingstrue` suggests that it is stored. Try to confirm the assumption in the code, and you will agree that even reading the code is difficult, let alone to adapt it. To borrow macros for use in other collections is also inhibited, as Wietse Dol communicated to me. Of course, it is a good thing to go for general mechanisms in macro writing. But the right balance between metacodes and single shot coding is the royal road, IMHO, with all respect.

[7] I was told that AMS considers to abandon `\nofrills` altogether.