BLUe-2-LaTeX—expansion and some more

Kees van der Laan

Abstract

Conversion should not be a problem. It is best to use a general accepted tool, which comes with a preprint format. Until BLUe's format is generally accepted a BLUe script has to converted to comply with publishers' formats to get it out. Conversion via TeX's expansion is exercised in this note, with as result a plain TeX Convertor Assistant, in the spirit of AWK. In netland it does not matter because BLUe's format system is available from CTAN and NTG's 4allTeX CD-ROM, and therefore everybody can format BLUe scripts, and no conversion is needed.

1 Introduction

Conversion is important and will be with us for a long time, because TeX is a fixed-point and the world around is changing. The conversion problem treated is relatively simple because it is between marked up scripts of the same kind, but nevertheless hard in its full generality.

Suppose that you have adopted BLUe's format as your personalized system. Suppose further that you submit articles to a journal—for example to MAPS or AMS. How to achieve this with as little conversion hazzle as possible? Below the solution BLUe-2-LaTeX, biased by `MAPS.sty` has been discussed.

To start with the bad news. I think it is undoable and a waste of energy, to provide a fully-automated conversion tool, unless there is a real demand, for example when a publisher adopts BLUe's format system.[1]

The good news is that in the hands of a disciplined, knowledgeable user and restricted to canonical BLUe scripts we can get by with a little care.

I'll sum up the various ways I can think of and exercise the expansion approach via TeX. For the target I assume old LaTeX, that is the version prior to 1994, especially to `MAPS.sty`. I hope,[2] before losing myself into details, that

> this note will spark discussions towards THE approach.

At the end there is the Columbus' egg solution from the point of view of authors.[3]

2 Why?

Of course I have touched upon the conversion problem before while submitting various BLUe's notes to MAPS. However, only after Irina Makhovaya asked me

> 'And what about a BLUe-2-AMS convertor?'

I started experiencing how far we can get by with TeX alone. The purpose is a 'little tool in plain TeX' to assist the conversion of a canonical BLUe script into a MAPS submission.

I don't use MAPS, TUGboat, or `<YouNameIt>` styles because I like to keep my scripts consistent, to use minimal markup, and because I prefer TeX, the 'fixed-point.'

3 What are the problems?

In essence we have a mapping problem between *incompatible* sets. There are incompatibilities at various levels. At the functionality level BLUe's format lacks explicit markup for the author, makes use of selective loading from databases, allows generating a ToC, and an index on the fly, and uses `\this<tag>{<options>}` and `\every<tag>{<options>}` to handle options, and the option of file verbatim inclusion in particular. Moreover BLUe's verbatims are semi-transparant, meaning an escape character is there to yield special effects. MAPS—LaTeX-biased—allows for switching from 1-column into 2-column and vice versa anywhere on the page,[4] next to having adopted NFSS.[5] Moreover, LaTeX comes with BiBTeX, and Makeindex. The difference in spelling is the least of all problems. Because of the above

> I really don't see how to provide for a general, automatic and *complete* convertor.

I would classify this conversion problem as *ill-posed*.

> The problem has to be simplified and confined to the tags which provide similar functionalities.

[1]A script of a dozen of pages contains 50 odd different BLUe markup tags. Hand conversion takes an hour.

[2]My experience so far is that not many TeXies—actually hardly none—take time for a proper discussion, alas.

[3]If a publisher adopts BLUe's format system, the default format, can act as a preprint style to suit the author and the publisher, with as result that an author is releaved from the conversion problem.

[4]It is a public secret that switching from 1-column into 2-column can't be done completely automatically.

[5]The New Font Selection Scheme.

3.1 Outer level and inner level markup

My a priori simplification is to separate outer level markup tags —which determines the look-and-feel of the typeset document—from inner level markup tags. The first are the subject of the simplified conversion problem, while the latter markup is left invariant, that is excluded from conversion.

Indispensible is a table of BLUe's format markup tags and the corresponding tags for the journal.

Inner level markup[6] has to do with the details of mathematics, tables and graphics, the common teasers. Look upon these as (black)boxes, with the detailed markup left at the discretion of the author, IMHO, with all respect.

Representative outer tags

BLUe script	$\longrightarrow$	MAPS.sty script
`\blueabstract`… …	$\longrightarrow$	`\begin{abstract}`… … `\end{abstract}`

`\beginscript`	$\longrightarrow$	`\begin{document}`
		`\maketitle`

`\bluehead ...`	$\longrightarrow$	`\section{...}`

`\bitem`
…
`\bitem`
`\smallbreak`
	$\longrightarrow$	`\begin{itemize}`
		`\item`…
		`\item`…
		`\end{itemize}`

`\begin` `quote`
`\end`
	$\longrightarrow$	`\begin{quote}`
		`\end{quote}`

`\ftn`	$\longrightarrow$	`\footnote`

`\begin` `verbatim`
`\<escape char>end`
	$\longrightarrow$	`\begin{verbatim}`
		`\end{verbatim}`

`\endscript`	$\longrightarrow$	`\end{document}`

Whatever the method applied for handling the outer level markup tags, lower level (plain TEX) macros have to accompany the converted script for processing the inner level markup tags.

3.1.1 Prelude matter

Let me summarize some conventions of BLUe's format prelude. In BLUe's format system the order of specification of the prelude[7] items is immaterial. Author and affiliation details are known by the system. References are specified in the prelude and set in a box to be pasted up at a place at will, allowing as side-effect refereeing to the literature items in one-pass.[8] Pictures can also be specified in the prelude and selectively loaded from the `pic.dat` database. In general these must be adapted to LATEX's picture environment, or perhaps recasted into POSTSCRIPT. Acknowledgements is something which does not belong to the script proper, similar for keywords and abstract, IMHO, with all respect. The non-proper script issues have to be supplied in the prelude part in BLUe's format system, but if you don't that is fine too.

All the above aspects differ with MAPS style —especially with LATEX which is underneath—in philosophy or markup conventions, and usually both.

Moving document elements such as the abstract part after the `\maketitle` is considered as fine-tunings.[9]

3.2 MAPS submission

Jos Winnink—one of MAPS editors—commented that the requirements for MAPS are that scripts should *not*

- exercise their own layout[10]
- interfere with existing control sequences; `\global`s especially are nasty.

MAPS processed scripts marked up in *old* LATEX up till 1995.

3.3 Canonical BLUe scripts

Plain TEX contains $O(1K)$ control symbols and sequences. BLUe's format adds $O(.1K)$ to that. My 'little' convertor does not account for all those. With a canonical script I mean a minimal marked up script, restricted to the use of the common high-level `blue.tex` markup tags —the so-called FUTs Frequently Used tags—as enumerated in the Appendix 'Canonical BLUe Tags.' The bonus of this disciplined markup is a clean script, hardly inflicked by markup. Conversion of this class of scripts shouldn't halt and shouldn't leave you with incomprehensible error messages.

TEX is extensible. An author can define special purpose lower level macros with names I can't foresee. To account for these I have provided a toks variable `\invariantconvertor` with the purpose that the provided tags won't be expanded during conversion, that is remain invariant.

A BLUe script lacks explicit markup for the author because it is a personalized system and therefore BLUe knows its author. The convertor must be assisted by prompting the author. An example of prompts which should be supplied in the file `aidsconvertor.tex` reads as follows.

```
\authorconvertor{<YourName>}
\invariantconvertor{\<tagone>\<tagtwo>...
   \<tagn>}
```

Mark up for affiliation in LATEX is included in `\author`.

[6]Where do outer level markup tags end and inner markup tags start? What is the borderline between them? As argued before—see the PWT users' guide—this is context dependent, and has to be discussed casu quo agreed upon by all parties involved. Allow the use of `\btable` or Turtle Graphics. An editor must always allow lower level tags. Would you replace fancy commutative diagram markup by your markup?

[7]Also called front matter.

[8]That I don't supply references any longer in MAPS submissions is an aside, which does not matter for the general problem. I urge readers to browse separately `lit.dat` by keyword, to spot the details of the references.

[9]BLUe's format boxes prelude elements. To move the abstract part I have to borrow a little more from BLUe's format system.

[10]IMHO, the question is to what level?

3.4 And what about the active parts?

Active documents are all about things happening behind the scenes, like automatic distilling information from the static elements, to be used for example in headers, footers, table of contents, and index. For the moment I consider these as fine-tunings. Tokens like `\loadtocmacros` and ilks are suppressed during conversion.

Cross references occur within math formulas via `\ref` and `\crsref`. Optionally `\ref` is followed by a symbolic name. For the moment this name has to be supplied in the invariant set. The cross reference macros from BLUe's format have to accompany the converted script.

4 The challenge

Given the above discussed complexities and restrictions the challenge is to develop a trustworthy Convertor Assistant which works the way you expect.

Coding the convertor in plain TeX is a non-trivial exercise in macro writing.

5 Conversion approaches

Stripped from incompatibilities the conversion has been reduced to transforming the various begin and end tags, blue headings, quotations, loops, if clauses, verbatims and ilks.[11]

The possible ways I can think of are

a Demarkup a BLUe script and insert the publisher's markup tags

b Leave the script as is, but provide
  - **1** `\maps` command, to yield the MAPS look-and-feel via plain TeX
  - **2** macros which invoke the MAPS markup commands via LaTeX

c Convert BLUe's outer level markup tags into MAPS markup tags, via
  - **1** an editor
  - **2** an AWK, Perl, or …script[12]
  - **3** TeX itself; the conversion is done via expansion of `\write`, mainly.

5.1 Method a: demarkup first

Roughly speaking method a is in use for example while transforming a Wordperfect script into (La)TeX.[13] To demarkup a script is very close to tag replacement and therefore this approach is close to method c1. The advantage is that it works at the expense of some tedious editing and trial-and-error formatting runs. When (minimal) mark is inserted in the final stages of the document preparation process this method is feasible.

5.2 Method b1: maps format

This method has all to do with acceptance of BLUe's format. The function of `\maps` is similar to say `\report`, but with as result of the BLUe script a document with the MAPS look-and-feel, processed by plain TeX. Similar to `\maps` one could provide `\ams`, or `\YouNameIt`.[14] Typical for this approach is that for example `\bluehead` remains as is, but `\prehead`, `\posthead`, and `\headfont` are adapted. Because MAPS editors have adopted to process MAPS completely via LaTeX, it is unlikely that `\maps` will ever see the light of day.

5.3 Method b2: BLUe as user interface

The idea is that `blue.tex` is seen as a user interface to LaTeX. This method redefines BLUe tags with as replacement text invokes of MAPS commands. A prototype `\bluemaps` for BLUe script markup tags to expand into the LaTeX-biased MAPS tags. This method respects MAPS to be processed completely by LaTeX.

Example *(Representative definitions)*

```
\def\bluehead#1\par{\section{#1}}
%
\def\bitem{\bgroup\begin{itemize}%
   \let\bitem\item
   \def\smallbreak{\end{itemize}\egroup}%
   \item}
```

5.4 Method c1: hand replacement

Little knowledge is required. Just a lot of work each time. Single shot contributions are handled this way in practice.

5.5 Method c2: converter scripts

I suspect MAPS editors to use method c2 when I submitted notes together with the `bluemaps` macros. The advantage of their approach is that they don't end up with a mess of macro sets.

5.6 Method c3: conversion via expansion

This is an unusal approach, and induced by TeX's mouth processing capabilities.[15] If it is true that TeX is all about text replacement this approach deserves attention, if not for finding out its limitations.

[11]However, in practice I stumbled upon another difference in style. BLUe's format system allows for running-in heads which is absent in MAPS as such, yielding not nice opening sentences when conversion is done more or less automatically. I like running-in heads especially for `\subsubhead`. The way out is to replace these `\subsubhead`s by `\(sub(sub))paragraph`s.

[12]Not further elaborated here, because of lack of hands-on for these tools. Maybe, I'll come back on this.

[13]I know of automatic convertors, and I'm also aware of their limitations. The activity at the moment is to start from the dvi file, to provide a `dvi2<YouNameIt>`, similar to `dvi2ps` and such like.

[14]As volunteer this development work is not so interesting for me. An organization can hire me to do the job.

[15]Bernd Raichle communicated that he has already applied the basic idea in his CTAN convertor programs: tex-archive/supportconvert/…

5.7 Method x: conversion via modified TEX

At TUG 95 Sebastian Rahtz has reported about the Elsevier project LATEX-2-SGML. They envision to modify TEX such that an appropriate dvi file will emerge, an 'intermediate,' to be processed further. The method builds upon the TEX parser.

Given their mass 'audience' a foolproof, cost-effective tool is needed. My convertor is personalized, small scale, and expects a little TLC, Tender Loving Care.

However, one never can tell what future has in store for us.

6 Conversion via expansion

The basic idea of the method elaborated in this note is to read and write each line[16] with the conversion done while writing. The convertor is table-driven. This basic loop reads in its simplest form as follows.

```
\loop\ifeof\bluesrc \break\fi
   \read\bluesrc to\inp
   \expandafter\lop\inp\eol
   \immediate\write\maps{\inp}
\pool
%with auxiliaries
\def\loop#1\pool{#1\loop#1\pool}
\def\break#1\pool{\fi}
```

Crucial is the inspection of `\inp`. The *first* token is lopped off and depending on its value non-expandable actions are performed, for example redefining `\inp`.

Let us discuss representative situations, for example how to handle BLUe's `\blueabstract`, `\bluepictures`, `\beginscript` and `\endscript`, `\bluehead`, `\bitems`, `\ftn`, `\beginverbatim` and closing, comments, and `\this<tag>s`.

Some of the problems are induced by BLUe's minimal markup spirit,[17] such as explicit opening tags and implicit closing tags and as few curly braces as possible. This opposed to as well explicit opening as closing tags and to surround arguments by braces. Because conversion via expansion is unusual I have exercised it a little, to illustrates the idea and to stirr up discussions.[18]

Dislaimer. Beware there is no complete solution, no finetunings, no general production tool, and definitely no foolproof version as yet. (In-line) Verbatims have to be verified for sure. However, one never can tell whether the used TEXniques will turn out to be useful some day.

6.1 Abstract

The minimal markup in a BLUe script is to precede the abstract by `\blueabstract` and end it by the implicit markup of a blank line. The abstract text can extend several lines. The convertor encloses the abstract text by `\begin{abstract}` and `\end{abstract}`. `\bluekeywords` goes similar.

Restricted to the handling of `\blueabstract` only the coding reads as follows.

```
\def\blueabstract#1{\begingroup\abstractcnt1
   \def\inp{\string\begin{abstract}\gobble#1}}
\def\par{\ifnum\abstractcnt>0 \endgroup
   \def\inp{\string\end{abstract}}\fi}
```

`\inp` contains the line read. The first token is `\blueabstract` and the rest of the line is supplied as argument to the invoke of `\blueabstract` by the main processing loop. The blank line yields `\inp{\par}` and `\par` is also included in `\convertorset` meaning that '`\par`' is executed, that is the above replacement text is executed.

6.2 Pictures

The codings as such are already separated from the script and stored in `pic.dat`. This `\thispicture...` is left invariant. The picture names should be added to `\invariantconvertor`. The same holds for `\bluereferences`. The macros should accompany the converted script.

6.3 Beginscript and endscript

The conversion of `\beginscript` goes via the following redefinition. Note that LATEX needs an explicit `\maketitle`. BLUe sets the title automatically when `\beginscript` is encountered.

```
\def\beginscript{\string\begin{document}
                 \string\maketitle}
```

`\endscript` is similar in principle. A little more has to be done, however, to suppress the `\par` at the end of the file. When `\endscript` is encountered, `\end{document}` is written to the file and the conversion is stopped.[19]

6.4 Headings

The problem is to recognize the situation, to get hold of the header text—the argument—and to insert the latter within braces after `\section`. I assume that the header text is a 1-liner, and that the markup tag is the first tag on the line. A nuisance is to get rid of inserted spaces.[20]

```
\def\bluehead#1{\def\inp{%
   \string\section{\gobble#1}}}
```

Similar to headings is `\blueexample`. A macro `\example` must accompany the converted script because LATEX lacks `\example`.

[16]Do remember that either a line is read, or more than one line until the braces match.
[17]In pursuit of Knuth.
[18]Have a glance at the included toy example to get an idea of what can be handled automa*g*ically, already.
[19]Be aware of the difference beteen `\string` and `\noexpand`. The latter inserts a space after the non-expanded token.
[20]The inserted space before the closing brace does not harm most of the time. `\example` must be defined appropriately, however.

6.5 Items

First of all there are the following varieties in BLUe: `\item`, `\bitem`, `\nitem`, `\aitem`, and `\Aitem`. BLUe's format assumes that a sequence of item tags is terminated by `\smallbreak`.

LATEX provides the bulleted, enumerated and descipted items, all surrounded by proper opening and closing tags.

The `\bitem`s have been treated in the example below, to show one way of doing. I consider the handling of the * option of LATEX's sectioning tags as fine-tunings. `\input` goes similar. Restricted to the case of `\bluehead` only the coding reads as follows. `\remove` removes its argument from `\setconvertor` with the effect that in the main processing loop `\bitem` is treated differently for the second, third etc. time, untill the group is closed that is when `\smallbreak` is encountered.

```
\def\bitem#1{\begingroup
   \def\bitem{\string\item}
   \immediate\write\bluemaps
      {\string\begin{itemize}}
   \advance\bitemcnt1 \remove\bitem}
\def\smallbreak{\ifnum\bitemcnt>0
   \endgroup
   \def\inp{\string\end{itemize}}\fi}
```

The `\item`s go similar. The redefinition of `\inp` must insert square brackets around the argument of the `\item`. The `\nitem`s are slightly different. Nested items have been accounted for too. `\aitem`s and `\Aitem`s are considered as fine-tunings.

6.6 Footnotes

As such there are no problems. However, the footnote text is read completely—remind the The TEXbook217 'Additional lines are read if necessary, until an equal number of left and right braces has been found.'

LATEX won't bark but the converted script might contain long lines.

6.7 Verbatim tags

`\thisverbatim`s are preceded by % because they are incompatible. However, the contents is copied into `\thisverbatimconvertor` and used during the conversion.[21]

This enables flexibility with respect to the escape character which precedes the `endverbatim`. `\everyverbatim`s are just preceded by %.

`\beginverbatim` token is redefined straightforwardly. The exclamation mark ! is the default escape character to end a verbatim. Its catcode is set to 0. The catcodes of the braces are changed in order to process the verbatim line-by-line.

The in-line verbatims need (pre)insertion of `\verb`. Because these can occur on any position of the line no catcodes are changed. So the material inside is expanded during conversion.

6.8 Verbatim texts

Handling verbatim opening and closing tags is the top of the iceberg. The verbatim text must be handled appropriately. Like in ordinary verbatim we have to switch off temporarily the backslash as escape character.[22] A beneficial side-effect is that TEX does not continue reading lines until braces match. Verbatim texts after `\begindemo` and Knuth's demos are handled similarly.

6.9 Comments

The catcode of the comment symbol, %, is set to 12, with as result that comment lines won't disappear. Be aware that tokens in there are expanded.

6.10 Explicit skips

I decided to replace `\bigskip` and ilks by a blank line. It is descriptive markup we are after, aren't we?

6.11 Newcounts and such like

It can happen that counters and such like will be used in the script. Not only do we have to take care of `\newcount`—an outer def—but also of the counter name, which can be an arbitrary control sequence.

```
\def\newcount#1{\def#1{\string#1}\def\inp
   {\ea\string\csname newcount\endcsname#1}}
```

It is assumed that each `\newcount\<countername>` occurs as such on a line.

I chose to neglect LATEX's `\newcounter`.

Remark. It is tempting to treat defs in the same way. I refrained because defs can occur more than once with the same name for example `\def\data`.

6.12 Facts from real life

In converting a `\btable` and its data, I stumbled upon that read does not per se read a line but, if the case arises, it reads up to and including the line with the closing brace.[23]

6.13 Conversion table

The table consists of redefinitions of BLUe's format tags. First the simple ones which when expanded yield the required substitution. Next there are definitions which should remain invariant, but should not be expanded during the write. Third there are (empty) definitions which omit the tag during the write. Finally, we have headings and ilks which next to changing the tag name should insert braces *around* the head text. Below representative parts of the table have been selected to convey the idea.

```
%Conversion table
%1. Cs-s which redefine \inp, or do
%   some other inexpandable process.
%The cs-s are  also collected in the
```

[21]This implies that `\catcode` isn't converted as such.

[22]Still a problem with in-line verbatims.

[23]If we wish to keep the layout invariant we could change the catcodes of the braces before reading a line. However, this conflicts with other situations, like the argument of an `\item`.

```
%set \setconvertor for identification.
\def\bitem#1{\begingroup
   \def\bitem{\string\item}
    \immediate\write\bluemaps
      {\string\begin{itemize}}
    \advance\bitemcnt1 \remove\bitem}
%...
\def\yields{\endgroup\def\inp{!yields}}
%2. Cs-s which expand irregularly
\def\beginscript{\string\begin{document}
   \string\maketitle}
%...
\def\ftn{\string\footnote}
%3. CS-s which should disappear
\long\def\process#1{\let#1\empty}
\expandafter\fifo\disappearset\ofif
%4. CS-s which should remain invariant
\long\def\process#1{\def#1{\string#1}}
\expandafter\fifo\invariantset\ofif
%5. To be commented out
{\catcode`\%=12\gdef\percent{%}}
\long\def\process#1{\def#1{\percent\string#1}}
\expandafter\fifo\commentset\ofif
%end conversion table
```

6.14 Toy test

A toy test program to demonstrate representative cases.

```
\bluetitle BLUe-2-MAPS

\bluesubtitle expansion and some more

\blueabstract BLUe-2-MAPS is all
              about conversion.

\newcount\cglcnt
\beginscript

\bluehead head text

Text
\bitem first bitem
\itemitem- first subitem
\bitem second bitem
\smallbreak
\item- first item
\item- second item
\smallbreak
\nitem first nitem
\nitem second nitem
\itemitem- first subitem
\itemitem- second subitem
\smallbreak
\item- first item
\itemitem- first subitem
\itemitem- second subitem
\item- second item
\smallbreak
\begindisplay
...&...\cr
\enddisplay
Math
$$a^2+b^2=c^2\ref\pyth$$
A famous truth from antiquity \crsref\pyth.

\thisverbatim{\catcode`\|=12 }
\beginverbatim
  \def\cs#1{{\tt\char`\\ #1}}
  \beginscript
   \buehead blah-blah

   ...& $ | ^ _ %
  \endscript
!endverbatim

\begindemo
...
!yields
...
\enddemo

\loop
...
\ifnum\cglcnt=0
...
\advance\cglcnt-1
\repeat

\mycs{abc}

\def\btablecaption{$\leftarrow$\hfill
 C a p t i o n\hfill$\rightarrow$}
\def\first{\hbox to6ex{\hss First\hss}}
\def\header{\hbox to18ex{\hss Header\hss}}
\def\rowstblst{{\hfil$\vcenter to20ex{\vss
\offinterlineskip
\fifo rowstblst\ofif\vss}$}}
\def\process#1{\hbox to0pt{\hss
#1\hss}\kern.5ex}
\def\data{$\vcenter{\offinterlineskip
\hbox{Table}\kern1ex\hbox{Proper}}$}
\def\footer{$\leftarrow$\hfill{\small
 F o o t e r}\hfill\hfill\hfill
 $\rightarrow$}
$$\vcenter{\btable{\data}}$$
\endscript
```

6.15 User guidance: aidsconvertor.tex

The above example was converted with the following `aidsconvertor.tex`.

```
%Auxiliary for BLUe-2-MAPS: aidsconvertor.tex
%Personalized info
\authorconvertor{Kees van der Laan}
%To assist conversion
\invariantconvertor{\mycs\a\b\pyth}
```

Other token variables available are `\commentconvertor` to precede the tags by comment symbol, `\deleteconvertor` to neglect the tags.

6.16 Test result

```
\documentstyle{article}
\author{Kees van der Laan}
\input{bluemaps.tex}
\input{bluemaps.xtr}

\title{BLUe-2-MAPS }
%\subtitle{expansion and some more }

\begin{abstract}BLUe-2-MAPS is all
              about conversion.
\end{abstract}
\newcount\cglcnt
\begin{document} \maketitle

\section{head text }

Text
\begin{itemize}
\item first bitem
\begin{description}
\item[-] first subitem
\end{description}
\item second bitem
\end{itemize}
\begin{description}
\item[-] first item
```

```
\item[-] second item
\end{description}
\begin{enumerate}
\item first nitem
\item second nitem
\begin{description}
\item[-] first subitem
\item[-] second subitem
\end{description}
\end{enumerate}
\begin{description}
\item[-] first item
\begin{description}
\item[-] first subitem
\item[-] second subitem
\end{description}
\item[-] second item
\end{description}
\begindisplay
...&...\cr
\enddisplay
Math
$$a^2+b^2=c^2\ref\pyth$$
A famous truth from antiquity \crsref\pyth.

 \begin{verbatim}
  \def\cs#1{{\tt\char'\\#1}}
  \beginscript
   \buehead blah-blah

   ...& $ | ^ _ %
  \endscript
 \ end{verbatim}%the space is a kludge

\begindemo
...
!yields
...
\enddemo

\loop
...
\ifnum\cglcnt=0
...
\advance\cglcnt-1
\repeat

\mycs{abc}

\def\btablecaption{$\leftarrow$\hfill
    C a p t i o n\hfill$\rightarrow$}
\def\first{\hbox to6ex{\hss First\hss}}
\def\header{\hbox to18ex{\hss Header\hss}}
\def\rowstblst{{\hfil$\vcenter to20ex
   {\vss\offinterlineskip\fifo rowstblst\ofif
   \vss}$}}
\def\process#1{\hbox to0pt{\hss
     #1\hss}\kern  .5ex}
\def\data{$\vcenter{\offinterlineskip
   \hbox{Table}\kern  1ex\hbox{Proper}}$}
\def\footer{$\leftarrow$\hfill{\small
 F o o t e r}\hfill\hfill\hfill$\rightarrow$}
$$\vcenter{\btable{\data}}$$
\def\data{1\cs2\rs...\cs...\rs}
$$\btable\data$$
\end{document}
```

Remarks. I don't know how to cope with nested verbatims in LATEX, so I inserted a space to fool the pattern matching.

The last lines about information for `\btable` have been edited for 2-column format. TEX reads either a line or more than one line to balance explicit braces. In general I would suggest to take this part out and include it in `bluemaps.xtr` under an appropriate name.

7 Convertor Assistant Program

The considerations in the beginning of this note led to a prototype. After a couple of months I returned to the subject and pushed it to its current state.

7.1 Design

A to be converted script has the following classes of markup tags, which should

- do more than expansion
- be expanded into
  - the control sequence itself
  - nothing
  - the control sequence itself preceded by %.

The documentation in this note was developed simultaneously,[24] and functioned also as design specifications.

In working with the Convertor Assistant fine-tunings had to be accounted for.

7.2 Coding

The main loop consists of processing the input line-by-line. Each line is split in two parts: the first token and the rest. The first token is checked whether it is in the set of tokens which do more than expansion.

The control sequences take the suffix `convertor`.[25] The educational pieces of code enclosed in this note have not been cluttered by the suffix.

User guidance is expected in the file `aidsconvertor.tex`.

7.3 Availability

The Convertor Assistant, this note as well as the kernel `bluemaps.tex` can be obtained from the CTAN.

7.4 Robustness

The convertor is not robust It will go astray when control sequences which do more than expansion during conversion are used in positions different from the first of a line. In short use THE markup tags only in the first position of a line.

8 bluemaps.tex accompanying macros

A converted script is accompanied by lower level macros which have been left invariant and can better be processed by this suite.

The kernel `bluemaps.tex` contains macros for

- btable
- Pascal fragments
- cross-referencing for math

[24]My PM-LP—Poor Man's Literate Programming—way of working.
[25]The control sequences of the program itself should not occur in the scripts.

- turtle graphics, and
- Knuth's gkppic 'picture' subset.

Next to this kernel I supply extra macros `bluemaps.xtr` specific for the document at hand.

9 Push the button

There are two phases: running the CA via plain TEX, and verifying the result by LATEX.

9.1 Convertor Assistant processing

Alas, the use of the CA is not that simple. Some preprocessing and postprocessing has to be done. The preprocessing consists of

- verify whether the used markup tags belong to the FUT, Frequently Used Tags (see the Appendix), and whether they occur on the first position of a line
- eliminate catcode changes, except for those in `\thisverbatim`
- eliminate Index Reminder markup
- assign control sequences which should remain invariant to `\invariantconvertor`, and similarly for `\deleteconvertor` and `\commentconvertor`
- extract detailed markup such as needed for a `\btable` from the script and accumulate this in `<jobname>.xtr`[26]
- look for use of ! (escape character in the convertor) and adapt this
- change markup for space after control sequences into `\cs{}␣`
- remove `\input blue.tex`, `\loadtocmacros`, `\loadindexmacros`, `\pasteuptoc`, `\sortindex`, `\pasteupindex`, and the overfull circumventing `\tolerance=500 \hbadness=499 \hfuzz=5pt`[27] and such like
- move the keywords and abstract part after `\beginscript` if they are supported by the style file
- make fileverbatims harmless by removing `\input` from `\thisverbatim`.[28]

Postprocessing has all to do with verifying the result and supplying pictures and other macros needed additional to `bluemaps.tex` to be sent along with the converted script. I called this extra file `bluemaps.xtr`. Tedious it is and beware name clashes.

While the CA is running keep an eye on the transformed file, and invariantly you will stumble upon unknown control sequences. Don't panic, just go on, and add these to the toks variable `\invariantconvertor`.

9.2 Problems left

Whenever I stumbled upon something I could not solve I could get around it after a night of sleep. For example testing for `\input blue.tex` and changing this into `\input{bluemaps.tex}`. The Columbus's egg solution I adopted was to let the author delete `\input blue.tex`[29] and let the Convertor Assistant automatically insert `\input{bluemaps.tex}`. I also decided to let the CA automatically insert `\input{bluemaps.xtr}`, the default extra lower-level macros.

10 Columbus' egg

What if …publishers adopt BLUe's format system with for example the default format as the preprint format?

Aha, then we don't have to muddle around any longer, because the author and the publisher use the same tags. The publisher is free to adjusts the tags to reflect the layout of the various journals, in short provide in-house production versions of the '`\maps`' tag.[30]

As far as I can see it, this is the way how publishers and authors cooperate nowadays with LATEX as formatting tool.

In netland—especially as *self-publishing author*—the problem is non-existant, because BLUe's format system is available from CTAN and NTG's 4AllTEX CD-ROM. Everybody can format BLUe scripts, and conversion is not needed.

11 Acknowledgements

Thank you Irina Makhovaya for challenging me to consider conversion of `BLUe-2-<YouNameIt>`. As usual Jos Winnink lend a helping hand in the finishing touch for MAPS inclusion.

12 Conclusion

Aren't we captivated by TEXnical solutions, while now and then (part of) the problem must be solved by other means? IMHO, with all respect, conversion is such a kind of problem. We should not convert but a `\maps`, `\ams`, or `\<YouNameIt>` tag should be provided, as variants on `\report`. The assumption is that general accepted tools or formats—standards so to say—have to be used. In short method b1 is the way to go.

Necessary conditions for general acceptance of a TEX format by users are

easy to use, well-documented, extensible.

BLUe's format system fulfills the necessary conditions, IMHO. I hope that `blue.tex` will enjoy general acceptance, to start in off-off-TEX netland.

[26] or enclose these (temporarily) by `\beginverbatim` and ⟨*escapechar*⟩`endverbatim`

[27] See Phil Taylor's Pragmatic TUGboat article on the issue.

[28] `\bluefileverbatim` is in the comment set.

[29] This is after conversion no longer appropriate, so why not just delete it as part of preprocessing phase?

[30] The latter should not be a too difficult job, because BLUe's format has been setup in a modular way, as a compatible extension of plain TEX, and manmac, the macros used by Knuth himself.

Conversion via TeX's expansion—or an AWK, Perl, … script—in short method c, is what I do in want for acceptance of BLUe's format by the MAPS editorial board. (MAPS) editors can forget about the format in which the document originated.

The toy example has shown the feasibility of the approach. It needs the Tender Loving Care of a knowledgeable user.

An early version of this note was converted with the following `aidsconvertor.tex`

```
%Personalized info
\authorconvertor{Kees van der Laan}
%To assist conversion
\invariantconvertor{\mycs\a\b\cglcnt
        \pyth\shiftright\this\every}
```

Conversion is an important issue, it deserves attention. Fine-tunings of the convertor will emerge if the need arises.

A real-life example would be BLUe-2-AMSPPT. I would embark such a project only with approval and support of the AMS.

My Convertor Assistant taught me to markup my scripts

simple, consistent, disciplined and above all minimal.

I expect my CA to become more and more useful because it will *learn* on-the-fly what TeX lingo I use.

My case rest. Have fun, and all the best.

13 Appendix: Canonical BLUe Tags

The follwing markup tags have been accounted for in the Plain TeX Convertor Assistant. These main tags—also called FUTs—have been borrowed from the PWT guide Appendix Formats. In the follwoing table the tags which take a ' as suffix have been used in this script.

```
Preliminary matter
\bluetitle'
\bluesubtitle'
\bluekeywords'
\blueabstract'
\blueissue'
\bluereferences
\bluepictures
\input'
Copy Proper
\beginscript \endscript'
\bluehead'
\thissubhead{...}
\bluesubhead'
\thissubsubhead{...}
\bluesubsubhead'

Inner level
\begindemo...<escapechar>yields...\enddemo
\everyverbatim{...}'
\thisverbatim{...}'
\beginverbatim'
<escapechar>endverbatim'
(and in-line \vrt\thinspace...\thinspace\vrt)'
\beginquote'
\endquote'
\em'
\dash, \Dash'
\dots'
\oldstyle'
\begindisplay
\enddisplay
(with & and \cr)
\displaycenterline{...}
\begincenterdisplay
\endcenterdisplay
(with & and \cr)
\btable \beginbtable \endbtable'
(with \data{...\cs...\rs...\cs...\rs...}
      \header{...}
      \footer{...}
      \rowstblst{...}
      \vruled{...} )'
\ref\<name>
\crsref\<name>
\ftn{...}'
\item{...}'
\itemitem{...}'
\aitem %to be fine-tuned
\Aitem %to be fine-tuned
\bitem
\nitem
\blueexample'
\thispicture{...}
(with \unitlength=...)
\cs{<tagname>}'

Back matter
\pasteupindex
\pasteupreferences
\pasteuptoc'
\sortindex
```