The interaction of LaTeX and PostScript Type 3: The underlying principles of dynMath

Abdelouahad BAYAR

Abstract

The \special command allows different ways of interacting between LaTeX and PostScript. The use of PostScript literal options such as \special{! ⟨*PostScript code*⟩ } and \special{" ⟨*PostScript code*⟩ } are the underlying means to implement the support for dynamic mathematical symbols in dynMath package. Thanks to these facilities, a PostScript Type 3 font is accessed and used inside LaTeX to draw mathematical symbols able to vary in size with respect to the context. In this talk, the basics of dynMath implementation and functionality based on PostScript Type 3 fonts are presented.

1 Introduction

Electronic documents, especially scientific ones, are typeset using static and/or dynamic characters. The mathematical formula is always the most suitable example for highlighting the subject. Mathematical variable-sized symbols, such as delimiters (brackets, braces, radicals, etc.), are a good way to make concrete the matter.

When we talk about scientific document processing, we think first and foremost of (La)TeX in its various implementations, i.e. TeX[5], LaTeX [6], luaTeX [7], ... etc. Dynamic symbols such as delimiters and others are supported by (La)TeX but in some cases the *optical scaling*, *uniformity of shape*, *right-sizing* and *metal likness* properties are not respected. The dynMath package [4] has been developed with the aim of supporting such characteristics and thus enhancing and improving the typesetting qualities of (La)TeX.

(La)TeX offers the possibility of interacting with PostScript [1] via the macro-primitive \special. The latter makes it possible to insert and manipulate PostScript code in (La)TeX through the dvips driver [8] while generating PostScript from the dvi files. We have used this mechanism to handle a dynamic Type 3 [1] font in TeX, thus enabling dynamic mathematical symbols to be supported by the dynMath system. The way in which this means of interaction is used is not usual in the development of (La)TeX packages. For this reason, it would be interesting to present details of the implementation process. We note that the same work was done when the development of dynMath was launched in 2016 [2]. The resumption of such work is justified by the change that has taken place in the implementation.

This preprint is organized as follows. In Section 2, the layout of the dynMath system is given. In Section 3, some details of dynMath in terms of the Type 3 font are presented. In Section 4, the way in which dynMath supports dynamic mathematical symbols in terms of TeX programming and interaction with the dynamic font Type 3 is studied. The paper ends with a conclusion and perspectives.

2 dynMath : the layout

The layout of dynMath is based mainly on two files, dynMath.sty and dynMath.tps.

- dynMath.sty: this is the LaTeX package itself. It contains the definition of the macros required to support the mathematical variable-sized symbols.
- dynMath.tps : this is the specification of a PostScript Type 3 font parameterized to draw mathematical symbols with dimensions and shapes satisfying given contexts.

Some details of the two files will be seen below to give an idea on how they work. A part highlights the interaction between LaTeX and PostScript Type 3.

3 dynMath: the font

3.1 PostScript inside LaTeX

The PostScript Type 3 font is specified in the media dynMath.tps. It is a font which respects the Type 3 specification but it is included in the macro primitive \special and having a global PostScript scope:

```
% Content of ‘‘dynMath.tps’’
\special{!
⟨PostScript Type 3 Specification of dynamth font⟩
}
```

This is an interaction between LaTeX and PostScript in which the Type 3 font is inserted and seen throughout the document generated by LaTeX via the dvips driver.

3.2 Symbols in Table

The Type 3 font in dynMath.tps is called dynMath. Its encoding table is constituted by referring to the table for the character font cmex10 [5, p 432]. The font layout for dynMath is on Table 1.

Because dynMath is dynamic, the symbol appears only once in the table. However, it is parameterized to respond to the dimensions requested in a given context.

Table 1: `dynMath` font layout

	0	1	2	3	4	5	6	7
′00x	(0	) 1	[2	] 3	⌈ 4	⌉ 5	⌊ 6	⌋ 7
′01x	⦃ 8	⦄ 9	⟨ 10	⟩ 11	| 12	‖ 13	/ 14	\ 15
′02x	16	17	18	19	20	21	22	23
′03x	24	25	26	27	28	29	30	31
′04x	32	33	34	35	36	37	38	39
′05x	40	41	42	43	44	45	46	47
′06x	48	49	50	51	52	53	54	55
′07x	56	57	58	59	60	61	62	↕ 63
′10x	64	65	66	67	68	69	70	71
′11x	72 ∮	73	74	75	76	77	78	79
′12x	80	81	82 ∫	83	84	85	86	87
′13x	88	89	90	91	92	93	94	95
′14x	96	97	⌢ 98	99	100	∼ 101	102	103
′15x	104	105	106	107	108	109	110	111
′16x	112 √	113	114	115	116	117	118	⇕ 119
′17x	↑ 120	↓ 121	122	123	124	125	⇑ 126	⇓ 127

3.3 Parameterizing

Dynamic symbols are parameterized in the font to meet extension requirements. Two categories of characters are identified. This depends on whether the dynamic parts are delimited by straight-lines or curved-lines. Two types of stretching are identified:

1. Line-based extension: this type of extension is easy and straightforward to support. Examples include the bracket symbol "[", "↑",... etc.
2. Curved-based extension: this type of extension concerns symbols whose dynamic parts have curved-lines. Support for dynamism has necessitated the development of a mathematical stretching model (to be published) and an interpolation method that respects obliquity and convexity [3]. Examples include the parenthesis "(", "⦇", ... etc.

A dynamic symbol is characterized by three parameters: *height* (including *depth*), *width* and *thickness*. The thickness is in some way linked to the characteristics of the writing instrument (pen) or drawing instrument (brush).

It should be noted that the stretching undergone by a dynamic symbol is partly supported by the `dynMath.sty` package and the `dynMath.tps` font. Consider the dynamic symbol S. Let H_S, W_S and E_S be its *height*, *width* and *thickness* respectively. If the symbol is to be stretched by the amount h vertically and w horizontally, then the features in the stretched state will be H_S+h, W_S+w and E_S as its *height*, *width* and *thickness* respectively. Thickness is not affected by the extension. It should be noted that the stretching supported by the font is not linear. We'll call it semi-optical because the thickness remains unchanged. Globally speaking, the thickness also changes, but this is the work of LaTeX and the PostScript interpreter (see below).

4 `dynMath`: the style package

4.1 Useful macros and conventions

The `dynMath.sty` style package defines all the variables useful for internal operation, as well as defining others used as an interface for interaction with the PostScript Type 3 font `dynMath`. It also defines macros for managing mathematical formulas based on extensible symbols. We have followed a particular way of naming the macros relating to dynamic symbols in LaTeX. Without doubt, the most interesting of the macros is the primitive `\left` and its counterpart `\right`. `dynMath` defines a similar macro which does the same job as `\left` but operates with the dynamic symbols defined in the PostScript Type 3 font. The general syntax of this macro is:

`\meLeft`⟨*delim*$_1$⟩⟨*formula*⟩`\meRight`⟨*delim*$_2$⟩

We refer to the normal-LaTeX macros to name the `dynMath` ones in order to make it easier to use for those accustomed to using (La)TeX . The same names are used, beginning with a capital letter and

preceded by "me" meaning "metal". Another example concerns \overbrace to which corresponds \meOverBrace in dynMath.

4.2 Dynamism management steps

In this section, the important steps in dynamism management are presented. It should be noted that each macro relating to the extension phenomenon is responsible for managing the relative extension parameters. The need may differ from one macro to another. Consideration of one of them highlights the general concept. The macro used as an example is \meLeft. One of the steps in the extension process is interaction with the Type 3 font. We are not going to talk about the \meLeft macro in programming terms, but only in an algorithmic sense and in a language as natural and abstract as possible. The definition of this macro is:
\def\meLeft#1#2\meRight#3{⟨ *macro definition*⟩}
With:
#1: left delimiter.
#2: formula to be delimited.
#3: right delimiter.
We assume that:
ldel: represents #1,
formula: represents #2,
rdel: represents #3.

Before presenting the steps of the \meLeft macro, the meanings of some keywords used are given in the table below:

Keyword	Meaning
ldel	left delimiter
rdel	right delimiter
mAxis	mathematical Axis
fbox	formula box
fh	formula height
fd	formula depth
fw	formula width
hm	height mathematical
lth	left thickness
fs	font size
symWidth	symbol Width
fdelb	formula delimiter box

The main steps of \meLeft are:

1. Determine the current mathematical style: style
2. In style
   - Determine the height of the mathematical axis: mAxis.
   - Put formula in fbox.
3. Determine the dimensions of fbox :
   - Height: fh
   - Depth : fd
   - Width : fw
4. Determine the mathematical height hm: $\texttt{hm} = \sup(\texttt{fh} - \texttt{mAxis}, \texttt{fd} + \texttt{mAxis})$
5. Based on hm, determine the thickness of the left dynamic symbol ldel: lth.
6. Based on lth, determine the size fs of the PostScript font dynMath to write the delimiter ldel.
7. In terms of fs determine:
   - The vertical stretching amount h.
   - The horizontal stretching amount w.
   - The delimiter width symWidth.
8. Process the box fdelb which will contain the extensible PostScript delimiter :
   - Write the special in fdelb:\special{" ⟨*leftSpecial*⟩}.
   - In ⟨*leftSpecial*⟩:
     – Align the mathematical axis of the symbol ldel according to the font dynMath at size fs with the mathematical axis mAxis of formula.
     – Write ldel with respect to the font dynMath at size fs from the coordinates $(0, 0)$.
9. Set the dimensions of fdelb:
   - Width at symWidth .
   - Height at (hm + mAxis).
   - Depth at (hm − mAxis).
10. Adjust the position of fdelb by kerning in order to adjust the left margin of ldel.
11. Put on the contents of the fdelb.
12. Adjust the right margin of ldel by kerning.
13. Put on formula.
14. Repeat steps 5 to 12 for the rdel delimiter.

5 Conclusions

We have given an idea on the principles of interaction between (LA)TEX and a PostScript Type-3 font. This is the basis for supporting dynamic math symbols. However, the idea is presented at a level of abstraction that does not allow the approach to be given clearly. This is the objective of the next article which will be an extension and at the same time an in-depth detail of the implementation of dynMath.

References

[1] Adobe Systems Incorporated. *PostScript Language Reference Manual*. Adobe Systems Incorporated, Addison-Wesley Publishing Company, Reading, Massachusetts, 1999. Available at https://www.adobe.com/jp/print/postscript/pdfs/PLRM.pdf.

[2] A. Bayar. Towards an operational (LA)TEX package supporting optical scaling of dynamic mathematical symbols. In *TUGboat, Volume 37, No. 2 - 2016 Conference Proceedings*, vol. 37, pp. 171–179, Toronto, Canada, 2016.

[3] A. BAYAR. C^1 interpolation of sequences of points preserving convexity and obliquity based on oblique convex two-dimensional cubic bézier splines. In *2024 IEEE International Conference on Signal, Image, Video and Communications (ISIVC 2024)*, p. To appear, Marrakech, Morocco, May 2024. IEEE.

[4] A. BAYAR. `dynMath`: A postscript type 3-based latex package to support extensible mathematical symbols. *TUGboat* 45(1):18–24, 2024. `https://tug.org/TUGboat/tb45-1`

[5] D.E. Knuth. *The TEXbook*. Computers and Typesetting. Addison-Wesley, Reading, Massachusetts, 1st ed., 1984.

[6] L. Lamport. *LATEX–A Document Preparation System*. Addison Wesley, USA, 1994.

[7] luaTEX development team. *LuaTEX Reference Manual*. LuaTEX development team, www.luatex.org, February 2024. Available at `https://mirror.marwan.ma/ctan/systems/doc/luatex/luatex.pdf`.

[8] T. Rokicki. *Dvips: A DVI-to-PostScript Translator*. TUG Group, https://tug.org/dvips, February 2024. Available at `https://tug.ctan.org/systems/doc/dvips/dvips.pdf`.

◇ Abdelouahad BAYAR
Cadi Ayyad University - Higher School of Technology of Safi, Sidi Aissa Road, PB 89
Safi, 46000
Morocco
`a.bayar (at) uca dot ma`
ORCID 0000-0002-3496-505X