数式フォントについて、ちょっとだけ

私立文系🔰初級ユーザー

2022 年 12 月 4 日

🎄TEX & LATEX Advent Calendar 2022🎅

あらかじめのお詫び

⚠ 無保証です、すいません
私は一介の私立文系エンドユーザーであって、これまで LATEX で数式を使ったことがありません。また、飽くまでフォントの話が中心ですので、「正しい数式の書き方」ですとか「美しい数式とは」みたいなこともまったく分かりません。数学用語の邦訳も調べてないので、英語のママです。
この文書は、末尾に掲げた参考資料 2 点に目を通したうえで、手元でチョコチョコと試してみて、それで、こういう仕組みになってるのかなぁ、と推測できたことを適当にまとめてみたに過ぎないものです（なので、実用性も怪しいです）。LATEX の実装をきちんと追いかけて、正確に理解した上で書いているわけではありませんので、話半分くらいでお読みいただければと思います。

⚠ 最新の情報ではありません、すいません
私の手元のシステムは W32TEX [2020/07/19] です。しかも、私は pdfLATEX+article.cls と pLATEX+jarticle.cls しか使ったことがありません。そのため、最新の TEX Live ですとか、LuaTEX なんかでは違った結果になることもあるかも知れません。その点も、何卒ご注意、ご容赦を願えましたら幸いです。

はじめに、LATEX 2ε におけるテキストフォント指定の仕組みについて簡単に確認をして、それから、数式フォントの設定について見てみたいと思います。

1 まずは、LATEX 2ε の NFSS のおさらい

NFSS (New Font Selection Scheme) というのは、一言で言うと、フォントの属性を ⟨*encoding*⟩、⟨*family*⟩、⟨*series*⟩、⟨*shape*⟩、⟨*size*⟩ という 5 つに分類して、ロードする `tfm` はこれら属性の組み合わせに応じて決定する、という仕組みです（plain TEX や LATEX2.09 の頃は、`tfm` に予め“決め打ち”で名前を付けておいて、その名前を明示的に指定してロードするしか出来なかったみたいです）。

この、属性の組み合わせと `tfm` とを結び付けているのは、（`\DeclareFontFamily` と）`\DeclareFontShape` というコマンドです。

`\DeclareFontFamily` と `\DeclareFontShape` とを使って NFSS による表現と `tfm` とを対応付けるのは、クラスファイルの中でも、パッケージの中でも、プリアンブルにおいてでも構わないのですが、タイプセットの際に当該 ⟨*encoding*⟩ と ⟨*family*⟩ の組み合わせを宣言する `\DeclareFontFamily` がこれらのどこにも見当たらない場合には、⟨*encoding*⟩⟨*family*⟩`.fd` という名前のファイルが探されて[1]、その `fd` ファイルの中で `\DeclareFontFamily` と `\DeclareFontShape` とが宣言されています。

具体例を考えてみますと：

- 属性の組み合わせが `OT1/cmr/m/n/10` であるなら、`ot1cmr.fd` の中の `\DeclareFontShape` の定義に従って“`cmr10.tfm`”が `10pt` でロードされ、
- `T1/lmss/bx/sl/12` という組み合わせなら、`t1lmss.fd` の中の `\DeclareFontShape` の定義に従って“`ec-lmssbo10.tfm`”が `12pt` でロードされます。

属性を変更するコマンドを使って組み合わせを変えると、それに伴って、ロードされる `tfm` も自動的に変更されます。

例えば、属性の組み合わせが `OT1/cmr/m/n/10` であるときに `\fontshape{it}\selectfont` とか `\itshape` を使って、⟨*shape*⟩ を“`n`”から“`it`”に変更すると、`OT1/cmr/m/it/10` という組み合わせになるので、これに対しては、（`ot1cmr.fd` を経て）“`cmti10.tfm`”が `10pt` でロードされることになります。

続いて、`\fontfamily{ptm}\selectfont` として、⟨*family*⟩ を“`cmr`”から“`ptm`”に変更すると、今度は `OT1/ptm/m/it/10` という組み合わせとなるので、これに対しては、（`ot1ptm.fd` を経て）“`ptmri7t.tfm`”が `10pt` でロードされます。

このようにして、ユーザーは、実際にロードされる `tfm` がどれなのかをいちいち意識することなしに、NFSS の規則に則ってフォントの属性を指定するだけで、フォント（`tfm`）をいろいろと変更できるわけです[2]。

ここで、実際の例で、原稿ファイルから `pdf` へと至るフォント指定の流れを辿ってみましょう。一例として：

```
\documentclass{article}
\pagestyle{empty}
\begin{document}
a
\end{document}
```

というファイル `test_text.tex` を、私の手元で：

```
prompt> pdflatex test_text
```

のように pdfLATEX で処理すると、出来上がった `test_text.pdf` の中味は、「a」の一文字となります。`pdf` のプロパティを見て使われているフォントを確認しますと、“`CMR10`”という名前の“Type 1”フォントとなっています。

この場合、まず、LATEX のカーネル（`fonttext.ltx [2021/01/15 v3.0i]`[3]）では、以下のように定義がなされています：

```
\newcommand\familydefault{\rmdefault}
\newcommand\seriesdefault{\mddefault}
\newcommand\shapedefault{n}
```

[1] `fd` ファイル名では ⟨*encoding*⟩（と ⟨*family*⟩）は小文字化されることが多いです（`ltfssbas.dtx` での“`\try@load@fontshape`”の定義とその説明を参照）。

[2] TEX によるタイプセットが無事成功した後の、`tfm` と実フォントとの対応関係は、DVIware の `map` ファイルで指定します。TEX が使った `tfm` と、`map` で指定されている `tfm` 名とが異なっている場合は、その差は `vf` が埋めています。

[3] 私の手元の `[2020/02/11 v3.0g]` では“`\ifx\Umathchar\@undefined`”となってたのですけど。

```
\newcommand\mddefault{m}

\ifx\Umathcode\@undefined
\newcommand\encodingdefault{OT1}
\newcommand\rmdefault{cmr}
\newcommand\sfdefault{cmss}
\newcommand\ttdefault{cmtt}
\else
\newcommand\encodingdefault{TU}
\newcommand\rmdefault{lmr}
\fontfamily{\rmdefault}
\newcommand\sfdefault{lmss}
\newcommand\ttdefault{lmtt}
\fi
```

pdfTEX には \Umathcode というプリミティブはないので、

```
\encodingdefault/\familydefault/
                \seriesdefault/\shapedefault
```

はそれぞれ：

- \encodingdefault -> OT1
- \familydefault -> \rmdefault -> cmr
- \seriesdefault -> \mddefault -> m
- \shapedefault -> n

ということになります。

⟨*size*⟩ については、article.cls が読み込む size10.clo で：

```
\renewcommand\normalsize{%
   \@setfontsize\normalsize\@xpt\@xiipt
   .....
   \let\@listi\@listI}
\normalsize
```

として \normalsize が実行されており、その内部の \@setfontsize の定義は ltfssini.dtx で：

```
\def\@setfontsize#1#2#3{\@nomath#1%
    \ifx\protect\@typeset@protect
      \let\@currsize#1%
    \fi
    \fontsize{#2}{#3}\selectfont}
```

となっているので、結局：

```
\fontsize{10}{12}\selectfont
```

と同じことになります。したがって、デフォルトのフォント属性の組み合わせは：

OT1/cmr/m/n/10

になっています（\Umathcode というプリミティブが存在する LuaTEX や XETEX の場合には、属性の組み合わせは TU/lmr/m/n/10 となって、つまりは Latin Modern になります）。

この組み合わせについては、上述したとおり、ot1cmr.fd の中で：

```
\DeclareFontShape{OT1}{cmr}{m}{n}%
    {<5><6><7><8><9><10><12>gen*cmr%
     <10.95>cmr10%
     <14.4>cmr12%
     <17.28><20.74><24.88>cmr17}{}
```

と宣言されていますので、"cmr10.tfm" が 10pt でロードされることになります[4]。

tfm さえロードできれば TEX によるタイプセット自体は可能となります。その後で DVIware による処理で必要となる、タイプセットに使われた tfm と実フォントとの対応関係は、DVIware の map ファイルで指示されています。今の場合ですと、私の手元では、pdftex.map に以下のようなエントリが含まれています：

```
cmr10    CMR10    <cmr10.pfb
```

pdfTEX のマニュアルによれば、pdftex.map の書式は：

⟨*tfmname*⟩ ⟨*psname*⟩ ⟨*fontflags*⟩ ⟨*special*⟩ ⟨*encodingfile*⟩ ⟨*fontfile*⟩

となっていて、この "cmr10.tfm" のエントリについては、⟨*fontflags*⟩、⟨*special*⟩、⟨*encodingfile*⟩ の指定はなくて[5]、

- ⟨*tfmname*⟩ = cmr10
- ⟨*psname*⟩ = CMR10
- ⟨*fontfile*⟩ = cmr10.pfb

ということになります（"<" はサブセット埋め込みという指定です）。

以上、NFSS のデフォルトの組み合わせ OT1/cmr/m/n/10 から始まって、ot1cmr.fd ファイル内の \DeclareFontShape の設定による "cmr10.tfm" のロードを経て、最終的には、pdfTEX によるタイプセットの場合には、pdftex.map の中で "cmr10.tfm" と cmr10.pfb とが結びつけられている、ということが判りました。

これを踏まえれば、手持ちの欧文フォントを LATEX で使うには、何らかの方法で tfm を作って[6]、fd ファイルと、使いたい DVIware 用の map ファイルとを書いて、原稿でそれに合わせた NFSS の属性指定をすればよい、ということになります。

いろいろと端折ってしまいますが、仮に、GoodFont.pfb という Type 1 フォントがあるとして、その afm ファイルから、GoodFont.tfm、GoodFont.vf という T1 の vf と、その vf の参照先である base-GoodFont.tfm という 8r の tfm が作れたとするならば、原稿ファイルのプリアンブルに：

```
\AtBeginDvi{%
  \special{pdf:mapline base-GoodFont 8r.enc GoodFont.pfb}}

\usepackage[T1]{fontenc}

\DeclareFontFamily{T1}{GoodFnt}{}
\DeclareFontShape{T1}{GoodFnt}{m}{n}{<-> GoodFont}{}

\renewcommand{\rmdefault}{GoodFnt}
```

と書くだけで、GoodFont が dvipdfmx で使えるようになります（fd と map を別に用意する場合には、t1goodfnt.fd と GoodFnt.map とかにして、dvipdfmx の実行時に "-f GoodFnt.map" を追加する必要があります）。

---

[4] \DeclareFontShape の書式の詳細については、The LATEX Companion, 2nd ed. [1] の "7.10.3 Declaring new font families and font shape groups" や LATEX 2ε font selection [2] の "4.2 Font definition file commands" 等をご覧ください。また、この ot1cmr.fd では、ビットマップフォントを用いていた頃との互換性を保持するため、サイズの選択肢は 5pt から 24.88pt の 12 種類に固定されています。

ot1cmr.fd では 12 種類の固定サイズに対して対応させている tfm は 8 種類ですが、t1cmr.fd の場合はサイズが 14 種類の固定になっており、対応させている tfm も 14 種類に増えているので、Computer Modern はサイズによっては OT1 と T1 とでデザインが異なります。Computer Modern で任意のサイズが指定できるようにしたパッケージとして type1cm があり、任意サイズ指定可能且つ、OT1 と T1 とでデザインが変わらないようにしたパッケージとして fix-cm というのがあります。

[5] Adobe StandardEncoding（8a）を TeX Base 1 Encoding（8r）に re-encoding しているような場合には、例えば、"ptmr8r Times-Roman "TeXBase1Encoding ReEncodeFont" <8r.enc <ptmr8a.pfb" みたいな感じになります。"8r.enc" が ⟨*encodingfile*⟩ で、ダブルクォーテーションで囲まれている部分が ⟨*special*⟩ に当たりますが、pdfTEX のマニュアルによりますと、pdfTEX の場合は "TeXBase1Encoding ReEncodeFont" という instruction は "just ignored" とのことです。

[6] Type 1 フォントなら、簡単には afm2tfm や afm2pl があり、TrueType フォントなら、ttf2tfm が使え、OpenType フォントだと、otftotfm というツールがあります。また、TrueType フォントの場合は ttf2afm を使うと afm ファイルを得ることができます。そして、Type 1 にせよ TrueType にせよ、afm ファイルがあるならば、fontinst を使うことも可能となります。

2　それで、数式フォントだとどうなってるの？

ちょっと前置きが長くなり過ぎましたけれど、ここからが本題の数式フォントの設定の話になります。

最初に、結論的なことを掲げておきますと、**The LaTeX Companion, 2nd ed.** [1] の "7.10.7 Declaring new fonts for use in math" からの以下の引用部分が、なんか、ポイントっぽいです：

- To summarize: to introduce new symbol fonts, you need to issue a small number of `\DeclareSymbolFont` and `\SetSymbolFont` declarations and a potentially large number of `\DeclareMathSymbol` declarations. (p. 436)
- Because `\DeclareMathSymbol` is used to specify a position in some symbol font, it is important that all external fonts associated with this symbol font via the `\DeclareSymbolFont` and `\SetSymbolFont` commands have the same character in that position. The simplest way to ensure this uniformity is to use only fonts with the same encoding. (p. 435)

うんとザックリ言いますと、まず、`\DeclareSymbolFont`、`\SetSymbolFont`、`\DeclareMathSymbol` を使えば、数式フォントの設定がおおよそ出来て、それで、`\DeclareSymbolFont` と `\SetSymbolFont` で宣言するフォントのエンコーディングを揃えておけば、バージョンを変えるだけで部分的に数式フォントを変更するようなこともうまくいく、みたいなことが書かれています。

具体例を見てみましょう。今度は：

```
\documentclass{article}
\pagestyle{empty}
\begin{document}
\[(a+bc)\]
\end{document}
```

というファイル `test_math.tex` を考えます。これを私の手元で：

```
prompt> pdflatex test_math
```

のように pdfLaTeX で処理すると、出来上がった `test_math.pdf` の中味は：

$$(a + bc)$$

となっています。`pdf` のプロパティでフォントを確認しますと、"`CMMI10`" と "`CMR10`" という "Type 1" フォントとあります。

数式フォントの設定も NFSS に沿っているのであれば、上で見ましたように、「フォントの属性の組み合わせ → `fd` ファイル → `tfm` のロード → `map` ファイルでの実フォントとの対応づけ」という仕組みになってそうです。

実際、結論を先取りしてしまいますと、今の場合のフォント指定の流れは：

- `OML/cmm/m/it/10 -(omlcmm.fd)-> cmmi10.tfm`
  `-(pdftex.map)-> cmmi10.pfb`
- `OT1/cmr/m/n/10 -(see above)-> cmr10.pfb`

となっています。

NFSS による属性の組み合わせさえ決まれば、あとは、テキストフォントの場合と同じです。ところが、テキストフォントの場合には、属性変更コマンドを使ってフォントの属性を個々に変更することができましたが、数式フォントの場合には、"⟨*encoding*⟩/⟨*family*⟩/⟨*series*⟩/⟨*shape*⟩" をセットにして「シンボルフォント名（symbol font name）」や「数式アルファベットフォント名（math alphabet identifier）」を割り当てて、これらを使って属性の組み合わせをまとめて変更するようになってるみたいです。

また、`test_math.tex` では、何もフォントの変更を指示していませんのに、タイプセットした `test_math.pdf` では、"a"、"b"、"c" はイタリックになっていて、他方、"("、"+"、")" は立体となっています。更には、"+" の両脇にだけ、アキが入っています。

なぜこうなるのかについて、以下、順を追って見ていきましょう。

まず、`test_math.tex` の数式で使った 6 文字については、LaTeX のカーネル（`fontmath.ltx`）で、以下のように定義されています[7]：

```
\DeclareMathSymbol{a}{\mathalpha}{letters}{`a}
\DeclareMathSymbol{b}{\mathalpha}{letters}{`b}
\DeclareMathSymbol{c}{\mathalpha}{letters}{`c}

\DeclareMathSymbol{+}{\mathbin}{operators}{"2B}

\DeclareMathDelimiter{(}{\mathopen} {operators}{"28}
                                    {largesymbols}{"00}
\DeclareMathDelimiter{)}{\mathclose}{operators}{"29}
                                    {largesymbols}{"01}
```

ここで、`\DeclareMathSymbol` と `\DeclareMathDelimiter` の書式は、次のようになっています：

- `\DeclareMathSymbol{`⟨*symbol*⟩`}{`⟨*type*⟩`}`
  `{`⟨*sym-fnt-name*⟩`}{`⟨*slot*⟩`}`
- `\DeclareMathDelimiter{`⟨*delim*⟩`}{`⟨*type*⟩`}`
  `{`⟨*sym-fnt-name1*⟩`}{`⟨*slot1*⟩`}{`⟨*sym-fnt-name2*⟩`}{`⟨*slot2*⟩`}`

これらの宣言には、それぞれ 2 つの意味が含まれています。つまり：

- 数式モード内での当該 symbol や delimiter の ⟨*type*⟩ は青字で示されているものである、ということ[8]
- 数式モード内では当該 symbol や delimiter のグリフは、赤字で示されているように名付けられた「シンボルフォント[9]」の、赤字の番号のスロットから持って来る、ということ（delimiter については、大小 2 個のグリフが宣言されています）

例えば、"a" の場合ではあれば、このグリフの ⟨*type*⟩ は "`\mathalpha`" であって、"`letters`" と名付けられた「シンボルフォント」の `` `a `` (=97) 番のスロットの文字を持ってくる、という意味で、また、"(" の場合であれば、このグリフの ⟨*type*⟩ は "`\mathopen`" であって、小さい "(" なら "`operators`" と名付けられた「シンボルフォント」の `"28` 番のスロットの文字、大きい "(" だったら "`largesymbols`" と名付けられた「シンボルフォント」の `"00` 番のスロットの文字を持ってくる、ということを表わしています。

個々のグリフへの ⟨*type*⟩ の割り当ては数式内のスペーシング調整に関係していて、他方、「シンボルフォント名」は "⟨*encoding*⟩/⟨*family*⟩/⟨*series*⟩/⟨*shape*⟩" のセットに対して割り当てられていて、書体指定に関係しているのですが、順番が逆になりますけれど、まず後者のほうから見ていきます。

上の赤字の部分から見て取れますように、"a"、"b"、"c" の各グリフは "`letters`" という名前の「シンボルフォント」から持ってきていて、"+"、"("、")" は "`operators`" から持ってきているわけですが、これらの「シンボルフォント名」は、`fontmath.ltx` で、次のように割り当てられています：

```
\DeclareSymbolFont{operators} {OT1}{cmr} {m}{n}
\DeclareSymbolFont{letters}   {OML}{cmm} {m}{it}
\DeclareSymbolFont{symbols}   {OMS}{cmsy}{m}{n}
\DeclareSymbolFont{largesymbols}{OMX}{cmex}{m}{n}
```

---

[7] "`` `x ``" は当該文字の文字コードの数値に置き換える記法で、"`"` は 16 進表記を表わしています。また、`fontmath.ltx` と `fonttext.ltx` はどちらも `fontdef.dtx` から生成されるので、`fontmath.ltx` のバージョンや日付けは、`fonttext.ltx` と一緒です。

[8] LaTeX 2ε でいう "⟨*type*⟩" は、TeX の "class" に当たるそうです（☞補足 1）。

[9] この「シンボルフォント名」には "math group" に代入するための番号が割り振られますが、LaTeX 2ε の "math group" は TeX の "math family" に当たるとのことです（`ltfssbas.dtx` says: "We also give a new name to `\newfam` and `\fam` to avoid verbal confusion".）（☞補足 1）。

つまり、"OT1/cmr/m/n" というフォント属性のセットには、"operators" という「シンボルフォント名」が割り当てられていて、同様に、"OML/cmm/m/it" には "letters"、"OMS/cmsy/m/n" には "symbols"、そして "OMX/cmex/m/n" には "largesymbols" というシンボルフォント名が当てられています（命名法は、各種 operator のグリフを持ってくるフォントに対しては "operators"、各種 letter を持ってくるフォントには "letters"、といった具合です）。

これで、\DeclareMathSymbol や \DeclareMathDelimiter で宣言されているグリフについて、「シンボルフォント名」を介してフォント属性の組み合わせが決まりますので、その後のフォント指定の流れは、テキストフォントの場合と同じです。

\DeclareMathSymbol では「シンボルフォント名」とスロットだけを指定していて、属性のセットに対する「シンボルフォント名」の割り当ては \DeclareSymbolFont のほうで行っているので、\DeclareSymbolFont でその「シンボルフォント名」が割り当てられている（⟨*encoding*⟩ 以外の）属性の組み合わせを変更すると、\DeclareMathSymbol の宣言には手を加えることなく、当該「シンボルフォント名」のグリフの書体を変更することが可能ということになります。

また、数式フォントの書体を変更するには、\DeclareSymbolFont での属性の組み合わせを変更する以外に、当該「シンボルフォント名」のフォントの、別「バージョン」を用意するという方法もあります。そのためには、予め \DeclareMathVersion を用いてバージョン名を宣言しておいてから、\SetSymbolFont で、そのシンボルフォント名の当該バージョン用の属性のセットを設定します。

例えば、LATEX のカーネル（ltfssini.dtx）では：

```
\DeclareMathVersion{normal}
\DeclareMathVersion{bold}
```

という 2 つのバージョンが宣言済みで、fontmath.ltx では：

```
\SetSymbolFont{operators}{bold}{OT1}{cmr} {bx}{n}
\SetSymbolFont{letters}   {bold}{OML}{cmm} {b}{it}
\SetSymbolFont{symbols}   {bold}{OMS}{cmsy}{b}{n}
```

が設定されています。これら諸設定により、通常ではフォント属性の組み合わせが：

```
operators:   OT1/cmr/m/n    -> cmr10 (! = ?)
letters:     OML/cmm/m/it   -> cmmi10 (a b c)
symbols:     OMS/cmsy/m/n   -> cmsy10 (ℜ ℑ ℵ)
```

であるところ、\mathversion{bold}（または \boldmath）の宣言後は、フォント属性の組み合わせが：

```
operators:   OT1/cmr/bx/n   -> cmbx10 (! = ?)
letters:     OML/cmm/b/it   -> cmmib10 (a b c)
symbols:     OMS/cmsy/b/n   -> cmbsy10 (ℜ ℑ ℵ)
```

へと変更されます。

さて、続いて、前のページで青字で示されていた ⟨*type*⟩ について見てみたいと思います。\DeclareMathSymbol や \DeclareMathDelimiter では、各グリフに ⟨*type*⟩ を割り当てていました。この ⟨*type*⟩ の種類と「意味」については、**The LATEX Companion, 2nd ed.** [1] の "Table 7.30: Math symbol type classification" とかにもありますが、ここでは **LATEX 2ε font selection** [2] の "3.6 Declaring math symbols" の部分に載っている表をコピペさせてもらっちゃいます：

Type	*Meaning*	*Example*
0 or \mathord	Ordinary	α
1 or \mathop	Large operator	$\sum$
2 or \mathbin	Binary operation	$\times$
3 or \mathrel	Relation	$\leq$
4 or \mathopen	Opening	$\langle$
5 or \mathclose	Closing	$\rangle$
6 or \mathpunct	Punctuation	;
7 or \mathalpha	Alphabet character	A

⟨*type*⟩ の指定は、\mathord ～ \mathalpha のような descriptive なものによってでも、0 ～ 7 のような数値によってでも、どちらでも構わないそうです（内部では結局数値に変換されているのですけれど[10]）。「意味」はこれで分かりましたが、なんで各グリフに ⟨*type*⟩ なんていうのを割り当てるのかと言いますと、それは、TEX が数式内で隣り合うグリフ同士の ⟨*type*⟩ の組み合わせに応じて、その間のスペース量を決定しているからです。

test_math.tex の例で考えてみますと、"("、"a"、"+"、"b"、"c"、")" の ⟨*type*⟩ はそれぞれ、\mathopen、\mathalpha、\mathbin、\mathalpha、\mathalpha、\mathclose でしたよね。ここで、各グリフの間にネコの絵文字を入れて：

(🐱 a 🐱 + 🐱 b 🐱 c 🐱)

としてみますと：

- 🐱 は、\mathopen と \mathalpha の間
- 🐱 は、\mathalpha と \mathbin の間
- 🐱 は、\mathbin と \mathalpha の間
- 🐱 は、\mathalpha と \mathalpha の間
- 🐱 は、\mathalpha と \mathclose の間

ということになります。そして、それぞれのスペーシングは：

- 🐱 = no space
- 🐱 = medium space
- 🐱 = medium space
- 🐱 = no space
- 🐱 = no space

となっています。これらがどうやって決まっているのかというと、⟨*type*⟩ の組み合わせに応じたスペースの挿入量について、TEX が内部で以下のような規則を持っているということらしいです（The TEXbook, p. 170 より）[11]：

		Right atom							
		Ord	Op	Bin	Rel	Open	Close	Punct	Inner
Left atom	Ord	0	1	(2)	(3)	0	0	0	(1)
	Op	1	1	*	(3)	0	0	0	(1)
	Bin	(2)	(2)	*	*	(2)	*	*	(2)
	Rel	(3)	(3)	*	0	(3)	0	0	(3)
	Open	0	0	*	0	0	0	0	0
	Close	0	1	(2)	(3)	0	0	0	(1)
	Punct	(1)	(1)	*	(1)	(1)	(1)	(1)	(1)
	Inner	(1)	1	(2)	(3)	(1)	0	(1)	(1)

この表で、0 ～ 3 は、以下のスペース量を表わしています：

0 = no space
1 = thin space (\,)
2 = medium space (\>, \:)
3 = thick space (\;)

括弧が付いているものは、text style と display style のときにだけその量のスペースが挿入されて、script style と scriptscript

(10) ⟨*type*⟩ は ltfssdcl.dtx で定義されている \mathchar@type に渡されて、数値に変換されます。ちなみに、\mathalpha 以外の ⟨*type*⟩ については、同名のプリミティブが存在しますが、ここではプリミティブとしてではなく単に ⟨*type*⟩ のマーカーとして使われています。プリミティブとしての \mathord ～ \mathpunct は、その引数に対して ⟨*type*⟩ の機能を与えたり、⟨*type*⟩ を変更したりするものです。例えば、$\tau\epsilon\chi$ だと "$\tau\epsilon\chi$" ですが、$\tau\mathbin{\epsilon}\chi$ とすると "$\tau\,\epsilon\,\chi$" となります。

(11) スペーシングについては、\mathalpha は \mathord と同じ扱いになります。"Inner" はブレース（"{ }"）で囲まれた "subformula" のことです。

style のときにはスペースが入らないということを意味しているのだそうです。また、"*" の組み合わせのときには、\mathbin が \mathord に変換されるとのことです[12]。

ところで、各種 ⟨*type*⟩ は数式内のスペーシング調整のために割り当てるわけですが、\mathalpha は、スペーシングに関しては \mathord と同じ扱いになっています。それでは、なぜわざわざ \mathalpha を \mathord とは別に割り当てるのかといいますと、それは、\mathalpha は書体の変更に関係しているからです。

\DeclareMathAlphabet という宣言を使うと、"⟨*encoding*⟩/⟨*family*⟩/⟨*series*⟩/⟨*shape*⟩" というフォントの属性のセットに対して「数式アルファベットフォント名（math alphabet identifier）」を割り当てることができて、この数式アルファベットフォントコマンドは、⟨*type*⟩ が \mathalpha であるようなグリフにだけ作用します。

fontmath.ltx では、以下の 4 つの数式アルファベットフォントコマンドが宣言されています[13]：

```
\DeclareMathAlphabet    {\mathbf}{OT1}{cmr}{bx}{n}
\DeclareMathAlphabet    {\mathsf}{OT1}{cmss}{m}{n}
\DeclareMathAlphabet    {\mathit}{OT1}{cmr}{m}{it}
\DeclareMathAlphabet    {\mathtt}{OT1}{cmtt}{m}{n}
```

また、\DeclareSymbolFontAlphabet を使うと、既に \DeclaerSymbolFont で宣言済みの「シンボルフォント名」に割り当てられたフォント属性のセットを、「数式アルファベットフォント名」の属性のセットとしても指定することができます（昔ながらの TEX では、使える math group 数の上限が 16 個なので、既に \DeclareSymbolFont で math group に割り当てた属性のセットと同じセットを再度 \DeclareMathAlphabet で別の math group に割り当てるような無駄遣いをしないための方策です[14]）。それで：

```
\DeclareSymbolFontAlphabet{\mathrm}    {operators}
\DeclareSymbolFontAlphabet{\mathnormal}{letters}
\DeclareSymbolFontAlphabet{\mathcal}   {symbols}
```

という 3 つの数式アルファベットフォントコマンドも併せて宣言されています。

更には、\SetMathAlphabet を使って、数式アルファベットフォント名についても別バージョンの設定ができるので、\mathsf と \mathit については、以下も宣言されています：

```
\SetMathAlphabet\mathsf{bold}{OT1}{cmss}{bx}{n}
\SetMathAlphabet\mathit{bold}{OT1}{cmr}{bx}{it}
```

fontmath.ltx で ⟨*type*⟩ が \mathalpha となっているグリフは、小文字と大文字のアルファベットと、0 ～ 9 の数字、それから大文字のギリシャ文字です[15]。なので、例えば：

```
\newcommand{\sample}
           {abAB12\gamma\delta\Gamma\Delta\sum\int}
```

として、このサンプル文字列に対して数式アルファベットフォントコマンドを適用すると、以下のようになります[16]：

`$\mathbf{\sample}$`	$\mathbf{abAB12\gamma\delta\Gamma\Delta\sum\int}$
`$\mathsf{\sample}$`	$\mathsf{abAB12\gamma\delta\Gamma\Delta\sum\int}$
`$\mathit{\sample}$`	$\mathit{abAB12\gamma\delta\Gamma\Delta\sum\int}$
`$\mathtt{\sample}$`	$\mathtt{abAB12\gamma\delta\Gamma\Delta\sum\int}$
`$\mathrm{\sample}$`	$\mathrm{abAB12\gamma\delta\Gamma\Delta\sum\int}$
`$\mathnormal{\sample}$`	$\mathnormal{abAB12\gamma\delta\Gamma\Delta\sum\int}$
`$\mathcal{\sample}$`	$\mathcal{abAB12\gamma\delta\Gamma\Delta\sum\int}$

また、\mathversion{bold}（または \boldmath）を宣言すると：

`$\mathsf{\sample}$`	$\mathsf{abAB12\gamma\delta\Gamma\Delta\sum\int}$
`$\mathit{\sample}$`	$\mathit{abAB12\gamma\delta\Gamma\Delta\sum\int}$
`$\sample$`	$\mathbf{abAB12\gamma\delta\Gamma\Delta\sum\int}$

という風になります。

以上で大体数式フォントの設定は判ったような気がしますが[17]、最後に、数式フォントのサイズ設定について触れておきます。

数式フォントのサイズは、本文のテキストサイズに対して、text style 及び display style、script style、scriptscript style の 3 つがセットになっていますが、これらを \DeclareMathSizes で宣言しておくことができるみたいです：

```
\DeclareMathSizes{⟨text⟩}{⟨math_text⟩}{⟨script⟩}{⟨scriptscript⟩}
```

この宣言をしておかないと \defaultscriptratio(=.7) と \defaultscriptscriptratio(=.5) に従って自動で計算されるらしいのですけれど、そうなると、フォントによっては、小さいサイズではかなり見辛くなってしまいます[18]。

そのため、fontmath.ltx のデフォルトでもちゃんと：

```
\DeclareMathSizes{5}{5}{5}{5}
\DeclareMathSizes{6}{6}{5}{5}
\DeclareMathSizes{7}{7}{5}{5}
\DeclareMathSizes{8}{8}{6}{5}
\DeclareMathSizes{9}{9}{6}{5}
\DeclareMathSizes{\@xpt}{\@xpt}{7}{5}
\DeclareMathSizes{\@xipt}{\@xipt}{8}{6}
\DeclareMathSizes{\@xiipt}{\@xiipt}{8}{6}
\DeclareMathSizes{\@xivpt}{\@xivpt}{\@xpt}{7}
\DeclareMathSizes{\@xviipt}{\@xviipt}{\@xiipt}{\@xpt}
\DeclareMathSizes{\@xxpt}{\@xxpt}{\@xivpt}{\@xiipt}
\DeclareMathSizes{\@xxvpt}{\@xxvpt}{\@xxpt}{\@xviipt}
```

---

(12) The TEXbook, Appendix G, pp. 442 f. に、次のように説明されています：
"**5.** If the current item is a Bin atom, and if this was the first atom in the list, or if the most recent previous atom was Bin, Op, Rel, Open, or Punct, change the current Bin to Ord and continue with Rule 14. Otherwise continue with Rule 17."
"**6.** If the current item is a Rel or Close or Punct atom, and if the most recent previous atom was Bin, change that previous Bin to Ord. Continue with Rule 17."

(13) 数式アルファベットフォントコマンドは "\math..." で始まることが多いです。LATEX 2ε font selection [2], "6.4 Naming conventions" says: "Math alphabet commands all start with \math...: examples are \mathbf, \mathcal, etc."

(14) ltplain.dtx の \e@mathgroup@top では、\Umathcode の有無で場合分けをしていて、"The upper limit of extended math groups (\fam) 16 in classic TEX and e-TEX, but 256 in Unicode TEX variants; classic and e tex have 16 fam (0–15); xetex and luatex have 256 fam (0–255)." と説明されています。

(15) mathpple パッケージや mathpazo パッケージでは、小文字のギリシャ文字も \mathalpha になっています。

(16) \mathnormal（= letters）が割り当てられている cmmi10.tfm（-> cmmi10.pfb）に入っている数字は old-style になっており、また、\mathcal（= symbols）が割り当てられている cmsy10.tfm（-> cmsy10.pfb）に入っている calligraphic letters は大文字のみで、小文字のアルファベットや数字は入っていないので、このような結果となります。

(17) あとは、\DeclareMathAccent とか \DeclareMathRadical というコマンドもあるのですけれど、これらについては \DecalreMathSymbol や \DeclareMathDelimiter からの類推で分かると思います。ちなみに The LATEX Companion, 2nd ed. [1] でも "Besides \DeclareMathSymbol, LATEX knows about \DeclareMathAccent, \DeclareMathDelimiter, and \DeclareMathRadical for setting up math font support. Details about these slightly special declarations can be found in [109], which is part of every LATEX distribution." (p. 435) として、これらの説明は LATEX 2ε font selection [2] に丸投げされています。

(18) 昔ながらの設定であれば、文字サイズは 5pt よりも小さくはなりませんし、Computer Modern ならオプティカルにデザインされているので小さな字でも読み易いのですが、スケーラブルなフォントで自動的に縮小されると、読めないくらい小さくなっちゃいますよね。

となっていますが、例えば、mathptmx パッケージなんかでは：

```
\def\defaultscriptratio{.74}
\def\defaultscriptscriptratio{.6}
\DeclareMathSizes{5}{5}{5}{5}
\DeclareMathSizes{6}{6}{5}{5}
\DeclareMathSizes{7}{7}{5}{5}
\DeclareMathSizes{8}{8}{6}{5}
\DeclareMathSizes{9}{9}{7}{5}
\DeclareMathSizes{10}{10}{7.4}{6}
\DeclareMathSizes{10.95}{10.95}{8}{6}
\DeclareMathSizes{12}{12}{9}{7}
\DeclareMathSizes{14.4}{14.4}{10.95}{8}
\DeclareMathSizes{17.28}{17.28}{12}{10}
\DeclareMathSizes{20.74}{20.74}{14.4}{12}
\DeclareMathSizes{24.88}{24.88}{17.28}{14.4}
```

というように、（概ね）少し大き目に調整されています。

ここまで縷々見てきたところを踏まえますと、数式フォントの書体を変更する方法としては、`\DeclareSymbolFont` の属性のセットを変える、バージョンを設定する、それと、数式アルファベットフォント名の設定をする、というやり方があるようです。

しかし、`\DeclareSymbolFont` を変更すると、そのシンボルフォント名の属性変更は文書全体に及んでしまいます。他方、数式アルファベットフォントコマンドは ⟨*type*⟩ が `\mathalpha` であるようなグリフにしか効きません。そうなりますと、文書中で部分的に数式フォントの書体を変えたいような場合には、バージョンを設定するのがよさそうです。

というわけで、ちょっと試してみます：

```
\documentclass{article}
\pagestyle{empty}

% altered from: mathptmx.sty
\DeclareMathVersion{ptmx}
\SetSymbolFont{operators}{ptmx} {OT1}{ztmcm}{m}{n}
\SetSymbolFont{letters}{ptmx}  {OML}{ztmcm}{m}{it}
\SetSymbolFont{symbols}{ptmx}  {OMS}{ztmcm}{m}{n}
\SetSymbolFont{largesymbols}{ptmx}{OMX}{ztmcm}{m}{n}

% altered from: mathpazo.sty
\DeclareMathVersion{pazo}
\SetSymbolFont{operators}{pazo} {OT1}{ppl}{m}{n}
\SetSymbolFont{letters}{pazo}  {OML}{zplm}{m}{it}
\SetSymbolFont{symbols}{pazo}  {OMS}{zplm}{m}{n}
\SetSymbolFont{largesymbols}{pazo}{OMX}{zplm}{m}{n}

% altered from: sansmathfonts.sty
\DeclareMathVersion{sans}
\SetSymbolFont{operators}{sans} {OT1}{cmsmf} {m}{n}
\SetSymbolFont{letters}{sans}  {OML}{cmssm} {m}{it}
\SetSymbolFont{symbols}{sans}  {OMS}{cmsssy}{m}{n}
\SetSymbolFont{largesymbols}{sans}{OMX}{cmssex}{m}{n}

% altered from: eulervm.sty
\DeclareMathVersion{euler}
% \SetSymbolFont{operators}{euler}{OT1}{myzeu}{m}{n}
\SetSymbolFont{letters}{euler}  {U}{zeur}{m}{n}
\SetSymbolFont{symbols}{euler}  {U}{zeus}{m}{n}
\SetSymbolFont{largesymbols}{euler}{U}{zeuex}{m}{n}
\DeclareSymbolFontAlphabet{\matheuler}{letters}

% The TeXbook, Excercise 18.37
\newcommand{\mathsample}{%
   \prod_{j\geq 0}
     \left(\sum_{k\geq 0}a_{jk} z^k\right)
   = \sum_{n\geq 0} z^n
     \left(\sum_{\scriptstyle k_0,k_1,\ldots\geq 0\atop
                \scriptstyle k_0+k_1+\cdots=n}
          a_{0k_0}a_{1k_1}\ldots\right)
}
\newcommand{\mathsampleA}{\[\mathsample\]}
\newcommand{\mathsampleB}{\[\matheuler{\mathsample}\]}

\begin{document}

\mathsampleA

{\mathversion{ptmx}
\mathsampleA}

{\mathversion{pazo}
\mathsampleA}

{\mathversion{sans}
\mathsampleA}

{\mathversion{euler}
\mathsampleB}

\end{document}
```

というファイル `math_sample.tex` を私の手元で：

```
prompt> pdflatex math_sample
```

のように pdfLaTeX で処理すると、以下のようになります：

$$\prod_{j\geq 0}\left(\sum_{k\geq 0}a_{jk}z^k\right) = \sum_{n\geq 0} z^n \left(\sum_{\substack{k_0,k_1,\ldots\geq 0 \\ k_0+k_1+\cdots=n}} a_{0k_0}a_{1k_1}\ldots\right)$$

$$\prod_{j\geq 0}\left(\sum_{k\geq 0}a_{jk}z^k\right) = \sum_{n\geq 0} z^n \left(\sum_{\substack{k_0,k_1,\ldots\geq 0 \\ k_0+k_1+\cdots=n}} a_{0k_0}a_{1k_1}\ldots\right)$$

$$\prod_{j\geq 0}\left(\sum_{k\geq 0}a_{jk}z^k\right) = \sum_{n\geq 0} z^n \left(\sum_{\substack{k_0,k_1,\ldots\geq 0 \\ k_0+k_1+\cdots=n}} a_{0k_0}a_{1k_1}\ldots\right)$$

$$\prod_{j\geq 0}\left(\sum_{k\geq 0}a_{jk}z^k\right) = \sum_{n\geq 0} z^n \left(\sum_{\substack{k_0,k_1,\ldots\geq 0 \\ k_0+k_1+\cdots=n}} a_{0k_0}a_{1k_1}\ldots\right)$$

$$\prod_{j\geqslant 0}\left(\sum_{k\geqslant 0}a_{jk}z^k\right) = \sum_{n\geqslant 0} z^n \left(\sum_{\substack{k_0,k_1,\ldots\geqslant 0 \\ k_0+k_1+\cdots=n}} a_{0k_0}a_{1k_1}\ldots\right)$$

Euler 以外のフォントでは、数字と "=" は "`operators`" から、アルファベットと "`\ldots`" は "`letters`" から、"`\geq`" と "`\cdots`" は "`symbols`" から、"`\prod`"、"`\sum`"、"("、")" は "`largesymbols`" から取ってきていて、それぞれの ⟨*encoding*⟩ も `OT1`、`OML`、`OMS`、`OMX` のママなので、この例の範囲では単純に `\SetSymbolFont` でバージョンを設定するだけでうまくいっているように見えます。（Euler だけ `\matheuler` という数式アルファベットフォント名を割り当ててますけれど、その辺りの事情については ☞ 補足 2）

それでは、Merry TeXmas & Happy TeXing! 🎉

参考資料

[1] Frank Mittelbach and Michel Goossens (with Johannes Braams, David Carlisle, and Chris Rowley), *The LaTeX Companion, second edition*, Addison-Wesley, 2004.

[2] LaTeX Project Team, *LaTeX2ε font selection*, August 2022.

補足 1　Behind the Scenes

私は全然 “glutton for TEXnicalities” なんかじゃないのですけれど、今回初めて The TEXbook の 153〜159 頁を眺めてみて、TEX レベルでの数式フォントの設定ってこうなってるのかなというのを、判ったつもりの範囲でまとめてみます：

- 数式モード内の各グリフには、16 進表記 4 桁の “\mathcode” というのが振られている
  - 最上位の桁が ⟨*class*⟩ の値を表わし、その次の桁が “\fam” の値、そして下位の 2 桁が当該グリフの “スロット番号” を表わしている（⟨*class*⟩ の値を 4096 倍した数値と、\fam の値を 256 倍した数値を、当該グリフのスロットの番号に足して、それを 16 進で表記するだなんて！）
- TEX でいう ⟨*class*⟩ は LATEX 2ε でいう ⟨*type*⟩ に当たり、“\fam” は “\mathgroup” に当たる[1]
  - ⟨*class*⟩ は、⟨*type*⟩ 同様 0 〜 7 の 8 種類あって、意味や役割も LATEX 2ε の ⟨*type*⟩ と同じ
  - “\fam” は、数式フォントのファミリを表わすパラメータで、昔ながらの TEX では 0 から最大 15 までの整数値が割り当てられる
  - 1 つの数式ファミリは、“\textfont”、“\scriptfont”、“\scriptscriptfont” という 3 つのフォントから構成される
- ⟨*class*⟩ が 7（= \mathalpha）であるようなグリフのみ、\fam の値の変更によってフォントが変更され得る
- デリミタの場合には、\mathcode の他に、16 進表記 6 桁の “\delcode” というのも振られている（上位 3 桁が、小さい場合のグリフで、下位 3 桁が、大きい場合のグリフに相当．3 桁区切りの最上位がファミリの番号で、下位 2 桁がスロット番号）
- \radical と \mathaccent については、またまた割愛

\mathcode については、予め INITEX がすべての文字に対して文字コードの番号（0 〜 255）を \mathcode として振っているらしいのですが、その際 0 〜 9 の数字と、アルファベット 52 文字についてのみ、別の値が与えられています。つまり、数字とアルファベットの \mathcode に関しては、それぞれの文字のスロット番号に、"7000 を加えた数値と、"7100 を加えた数値にしてあるとのことです。例えば、“0” の \mathcode は "7030 で、“1” は "7031、“2” は "7032、…ということで、“a” の \mathcode は "7161 で、“b” は "7162、“c” は "7163、…ということになります。

1 桁目が ⟨*class*⟩ で、2 桁目が \fam、そして下位 2 桁がスロット番号を表わしているわけですから、これらはすなわち：

- 数字 “0” のグリフは、⟨*class*⟩ が 7（= \mathalpha）で、数式ファミリは 0、スロットの番号が "30 番（10 進表記だと 48 番）ということになり、同様に、
- “a” のグリフは、⟨*class*⟩ が 7 で、数式ファミリは 1、スロットの番号 が "61 番（10 進表記だと 97 番）

という意味です。

plain.tex の設定を見てみますと（以下では一部を省略し、改行も施していますが）：

```
\font\tenrm=cmr10 % roman text
\font\sevenrm=cmr7
\font\fiverm=cmr5
...
\textfont0=\tenrm
\scriptfont0=\sevenrm
\scriptscriptfont0=\fiverm
```

とか、

```
\font\teni=cmmi10 % math italic
\font\seveni=cmmi7
\font\fivei=cmmi5
...
\textfont1=\teni
\scriptfont1=\seveni
\scriptscriptfont1=\fivei
```

という定義がありますので、数式ファミリの 0 はつまり、cmr の 10、7、5 ポイントから構成されており、数式ファミリの 1 は cmmi の 10、7、5 ポイントから成っています。

通常の文字以外の沢山の数式用のグリフの \mathcode については、plain.tex で、\mathcode や \mathchardef 等を使ってひとつひとつ定義されています。

例えば、test_math.tex で使った、“+”、“(”、“)” について確認してみますと：

```
\mathcode`\+="202B

\mathcode`\(="4028
\mathcode`\)="5029

\delcode`\(="028300
\delcode`\)="029301
```

となっていますので、“+” のグリフは、⟨*class*⟩ が 2 で、\fam が 0、スロットが "2B 番で、“(” は、デリミタでないときには、⟨*class*⟩ が 4 で、\fam が 0、スロットが "28 番、デリミタのときは、大きさによって、\fam が 0 の "28 番スロットのグリフか、\fam が 3 の "00 番スロットのグリフということになります。

plain.tex には：

```
\font\tenex=cmex10 % math extension
...
\textfont3=\tenex
\scriptfont3=\tenex
\scriptscriptfont3=\tenex
```

とありますので、数式ファミリの 3 は cmex10 です。また、cmex10 には、"00 番以外にも、大きさが異なる “(” が含まれていて、これらは、必要に応じて適切に切り替わるらしいです。そして：

```
\font\tensy=cmsy10 % math symbols
\font\sevensy=cmsy7
\font\fivesy=cmsy5
...
\textfont2=\tensy
\scriptfont2=\sevensy
\scriptscriptfont2=\fivesy
```

なので、数式ファミリの 2 は cmsy の 10、7、5 ポイントから成っています。

plain.tex で数式ファミリの 2 と 3 に割り当てられている cmsy と cmex は特殊らしくて、これらのフォントには通常よりも多くの \fontdimen パラメータがないといけないとのことです：

> *The TEXbook*, p. 433:
> Math symbol fonts (i.e., fonts in family 2) are required to have at least 22 **\fontdimen** parameters instead of the usual seven; similarly, math extension fonts must have at least 13. The significance of these additional parameters is explained in Appendix G.
>
> *Id.*, p. 157:
> Plain TEX uses family 1 for math italic letters, family 2 for ordinary math symbols, and family 3 for large symbols. TEX insists that the fonts in families 2 and 3 have special **\fontdimen** parameters, which govern mathematical spacing according to the rules in Appendix G; the **cmsy** and **cmex** symbol fonts have these

[1] ltfssbas.dtx で “\let\mathgroup\fam” とされています。

parameters, so their assignment to families 2 and 3 is almost mandatory[2].

数式ファミリの 0 ～ 3 については直接数値が指定されていますが、最大 16 ファミリ使えるそれぞれを数字で覚えておくのは大変なので、plain.tex では 4 以降は \newfam を使って「数式ファミリ名」に自動的に順番に数値を割り振っています[3]：

```
\newfam\itfam \def\it{\fam\itfam\tenit} % \it is family 4
\textfont\itfam=\tenit

\newfam\slfam \def\sl{\fam\slfam\tensl} % \sl is family 5
\textfont\slfam=\tensl

\newfam\bffam \def\bf{\fam\bffam\tenbf} % \bf is family 6
\textfont\bffam=\tenbf \scriptfont\bffam=\sevenbf
\scriptscriptfont\bffam=\fivebf

\newfam\ttfam \def\tt{\fam\ttfam\tentt} % \tt is family 7
\textfont\ttfam=\tentt
```

（\bffam 以外では、\textfont しか割り当ててませんね。）

\mathcode の 2 桁目は数式ファミリなので、グリフごとにフォントは決まっているわけですが、⟨*class*⟩ が 7 のグリフについては、\fam の値を変えると、フォントを変更できます。例えば：

```
\nopagenumbers
1: $123abc$\par
2: $\fam\itfam 123abc$\par
3: $\fam\bffam 123abc$\par
4: $\fam\ttfam 123abc$
\bye
```

というファイル test_fam.tex を私の手元で：

```
prompt> pdftex test_fam
```

として試してみますと、結果は、次のようになります：

1: $123abc$

2: $\mathit{123abc}$

3: $\mathbf{123abc}$

4: 123abc

さて。ここまで見た TEX での設定と、本編のほうで見た LATEX 2ε での設定とを考えあわせますと、まず、fontmath.ltx における：

```
\DeclareSymbolFont{operators} {OT1}{cmr} {m}{n}
\DeclareSymbolFont{letters}   {OML}{cmm} {m}{it}
\DeclareSymbolFont{symbols}   {OMS}{cmsy}{m}{n}
\DeclareSymbolFont{largesymbols}{OMX}{cmex}{m}{n}
```

という設定は、TEX でいうところの数式ファミリの 0 ～ 3 に相当していそうです。実のところ、\DeclareSymbolFont は内部で：

```
\expandafter\new@mathgroup\csname sym#1\endcsname
```

を実行しているので、第 1 引数に "sym" という接頭辞を付けた「数式ファミリ名〔数式グループ名〕」に、順番に数値を割り振っていることになります。つまり、この 4 行からは：

- \symoperators (= 0)
- \symletters (= 1)
- \symsymbols (= 2)
- \symlargesymbols (= 3)

という数式ファミリ名〔数式グループ名〕が作られています。

ということは、TEX で例えば：

```
\fam\itfam
```

として数式ファミリを変更したのと同様のことが、LATEX 2ε では：

```
\mathgroup\symoperators
```

等としてできることになります。実際、fontmath.ltx には：

```
\def\operator@font{\mathgroup\symoperators}
```

という定義が含まれていますので、"\operator@font" を使うと、⟨*type*⟩ が \mathalpha であるようなグリフのフォントを operators に変更することができます。

次に、fontmath.ltx の：

```
\DeclareMathSymbol{a}{\mathalpha}{letters}{`a} % "61
\DeclareMathSymbol{b}{\mathalpha}{letters}{`b} % "62
\DeclareMathSymbol{c}{\mathalpha}{letters}{`c} % "63
```

において、\mathalpa は 7 で、letters は 1 なので、 これらは INITEX による \mathcode の設定：

```
\mathcode`\a="7161
\mathcode`\b="7162
\mathcode`\c="7163
```

と一致しています。同様に、fontmath.ltx の：

```
\DeclareMathSymbol{+}{\mathbin}{operators}{"2B}

\DeclareMathDelimiter{(}{\mathopen} {operators}{"28}
                                    {largesymbols}{"00}
\DeclareMathDelimiter{)}{\mathclose}{operators}{"29}
                                    {largesymbols}{"01}
```

において、\mathbin は 2 で、\mathopen は 4、\mathclose は 5 であって、operators は 0 で、largesymbols は 3 なので、こちらも上で見た plain.tex による \mathcode の設定：

```
\mathcode`\+="202B

\mathcode`\(="4028
\mathcode`\)="5029

\delcode`\(="028300
\delcode`\)="029301
```

と一致しています。

つまりは、LATEX 2ε の \DeclareMathSymbol や \DeclareMathDelimiter というのは、与えられた引数を元に、TEX の \mathcode に値を変換するコマンドなのでしょうね。

補足 2　Pis-aller

本文書執筆のそもそもの動機は、文書中で一部分だけ数式フォントを変えるにはどうしたらよいのかを知りたかったというものでした。それで、The LATEX Companion を眺めてみましたら、本編のほうにも書きましたように、\DeclareSymbolFont と \SetSymbolFont でエンコーディングを揃えれば、バージョンを切り替えるだけで数式フォントが変わるらしいということが判りました。

fontmath.ltx を覗いてみたところ、OT1 の "cmr"、OML の "cmm"、OMS の "cmsy"、そして OMX の "cmex" という 4 フォントについて \DeclareSymbolFont でシンボルフォント名が宣言されていましたので、それぞれについて、同一エンコーディングの別フォントを探してきて \SetSymbolFont すれば良さそうです。

---

[2] The LATEX Companion, 2nd ed. でも同じことが説明されています："Some fonts used in formulas need more than seven font dimensions – namely, the symbol fonts called "symbols" and "largesymbols" (see Section 7.10.7). TEX will not typeset a formula if these symbol fonts have fewer than 22 and 13 \fontdimen parameters, respectively. The values of these parameters are used to position the characters in a math formula. An explanation of the meaning of every such \fontdimen parameter is beyond the scope of this book; details can be found in Appendix G of The TEXbook [82]." (pp. 428 f.)

[3] ltplain.dtx では、新たに \new@mathgroup が定義されていて、それから "\let\newfam\new@mathgroup" とされています。

ところが、ちょっと探してみた限りでは、これらのエンコーディングをそのまま採用している数式フォントパッケージはあまり見当たりませんでした。多分、デフォルトよりも多くの数学記号類を使えるようにするために、独自エンコーディングを採用してるんだと思います。

それでも、本編末尾の例で試してみました mathptmx、mathpazo、sansmathfonts の各パッケージについては、それぞれのフォントのエンコーディングがデフォルトと同じになっていましたので、これらパッケージのフォントを \SetSymbolFont で設定したら、うまくいったように見えます。

で、問題は eulervm パッケージでして、このパッケージでは、設定されているフォントは letters、symbols、largesymbols の 3 種類で、且つ、それらのエンコーディングは "U" となっていて、そして、operators については \encodingdefault の \familydefault であるフォントが当てられていました。

まず、エンコーディングが "U" となっている点についてですが、幸いこれはあまり問題にならなさそうです。というのも、確認してみましたら、letters、symbols、largesymbols に割り当てられているフォントは、どれも 0 ～ 127 番の範囲のスロットでは OML、OMS、OMX と同じになるようにしてあったからです。OML、OMS、OMX と異なっている点は、128 番以降のスロットに、letters では "ħ"、"-"、"=" の 3 グリフが追加されており、symbols には "!"、"("、")"、"+"、":"、";"、"="、"?"、"["、"]"、"^" の 11 グリフが[4]、そして largesymbols では "∞" が 1 個、それぞれ追加されているところでした。

次に、operators についてなのですが、operators には普通、本文のテキストフォントが割り当てられていますけれど、fontmath.ltx で operators から持って来ているグリフは、0～9 の数字と、大文字のギリシャ文字、それから、"!"、"+"、":"、";"、"="、"?"、"("、")"、"["、"]"、"/" で、あとは、\DeclareMathAccent で宣言されている数式アクセント類の多くも operators から取って来られています。

これに対して、eulervm パッケージでは、これらのグリフのうち、大文字のギリシャ文字は letters から、"+" 等々は symbols から取ってくるように変更されていて、数字と数式アクセント類については operators からのママなのですが、"euler-digit" というパッケージオプションを指定すると：

```
\DeclareMathSymbol{0}\mathalpha{letters}{"30}
...
\DeclareMathSymbol{9}\mathalpha{letters}{"39}
```

という風に、数字を letters から持って来るように \DeclareMathSymbol が再設定されます。

つまり、何もオプションを指定しないと、数字は operators に当てられている本文のテキストフォントのものになり、オプションを指定して \DeclareMathSymbol による割り当てを letters に変えれば、例の特徴的な Euler の数字になるというわけです。

本文書の元々の目的が、fontmath.ltx で宣言されている沢山の \DeclareMathSymbol の設定には触らずに、バージョンを替えるだけで文書内の一部分の数式フォントを変更したいというものなのですから、数字を Euler にしたいからといって、数字を letters から持ってくるように \DeclareMathSymbol を書き換えるというわけにはいきません[5]。

でもでも、Euler の数字はステキなので、是非々々使いたいです。

それで、苦し紛れに思い付いたのが、以下のような方法です：

苦し紛れその 1　数字の \mathcode を一時的に書き換える

たった今「\DeclareMathSymbol の設定には触らずに」と言ったばかりですのに \mathcode を書き換えようだなんて、なんとも節操がないのですけれど、\DeclareMathSymbol はプリアンブルでしか使えませんし、その影響は文書全体に及んでしまいますが、\mathcode であれば、局所的に書き換えることができそうです。

なお、上で述べましたように、eulervm パッケージでは、fontmath.ltx だと operators から持ってきているグリフの一部を symbols の 128 番以降のスロットに移しています。ここでは eulervm では "(" と ")" が：

```
\DeclareMathDelimiter{(}{\mathopen}
                      {symbols}{168}{largesymbols}{"00}
\DeclareMathDelimiter{)}{\mathclose}
                      {symbols}{169}{largesymbols}{"01}
```

となっていることも考慮して、以下のようにしてみます：

```
\documentclass{article}
\pagestyle{empty}

\DeclareMathVersion{euler}
\SetSymbolFont{letters}{euler}  {U}{zeur}{m}{n}
\SetSymbolFont{symbols}{euler}  {U}{zeus}{m}{n}
\SetSymbolFont{largesymbols}{euler}{U}{zeuex}{m}{n}

\newcommand{\tempeuler}{%
   \mathcode`0="7130 \mathcode`1="7131 \mathcode`2="7132
   \mathcode`3="7133 \mathcode`4="7134 \mathcode`5="7135
   \mathcode`6="7136 \mathcode`7="7137 \mathcode`8="7138
   \mathcode`9="7139 \mathcode`\(="42A8 \mathcode`\)="52A9 }

\newcommand{\mathsampleC}{%
   b_k(n)=
   {2^{2n+1/2}\over e^{\sigma}\sqrt n}
    \,e^{-k^2\!/n}
}

\begin{document}

\[\mathsampleC\]

{\mathversion{euler}
\[\mathsampleC\]

\tempeuler
\[\mathsampleC\]}

\end{document}
```

この test_mathsampleC.tex を pdfLaTeX で処理しますと、次のようになります：

$$b_k(n) = \frac{2^{2n+1/2}}{e^{\sigma}\sqrt{n}}\, e^{-k^2/n}$$

$$\mathrm{b_k(n)} = \frac{2^{2n+1/2}}{e^{\sigma}\sqrt{n}}\, e^{-k^2/n}$$

$$\mathrm{b_k(n)} = \frac{2^{2n+1/2}}{e^{\sigma}\sqrt{n}}\, e^{-k^2/n}$$

苦し紛れその 2　\fam ないし \mathgroup の値を一時的に変更する

数字は ⟨*type*⟩ が \mathalpha なので、つまり、⟨*class*⟩ が 7 なので、"\fam" の値を変更すれば、フォントが変更できそうです。

LaTeX 2ε 風に "\mathgroup" のほうを使ってみます：

[4] あと、見えないのですけれど、176 番のスロットには Euler Script Medium の 48 番の "ghost" も追加されています。

[5] それに、これをプリアンブルで宣言してしまいますと、他の数式フォントの数字が old-style になっちゃいますし。と思ったのですけれど、確認してみましたら、デフォルトの Computer Modern と sansmathfonts パッケージでは letters の数字が old-style なのですが、mathptmx や mathpazo の letters の数字は old-style じゃなかったです…。

```
\documentclass{article}
\pagestyle{empty}

\DeclareMathVersion{euler}
\SetSymbolFont{letters}{euler}  {U}{zeur}{m}{n}
\SetSymbolFont{symbols}{euler}  {U}{zeus}{m}{n}
\SetSymbolFont{largesymbols}{euler}{U}{zeuex}{m}{n}

\newcommand{\mathsampleD}{%
   A_n=\sum_k{2n\choose k}}

\begin{document}

\[\mathsampleD\]

{\mathversion{euler}
\[\mathsampleD\]

\[\mathgroup\symletters
\mathsampleD\]}

\end{document}
```

この test_mathsampleD.tex を pdfLaTeX で処理しますと、次のようになります：

$$A_n = \sum_k \binom{2n}{k}$$

$$A_n = \sum_k \binom{2n}{k}$$

$$A_n = \sum_k \binom{2n}{k}$$

苦し紛れその 3　数式アルファベットフォント名を割り当てる

本編では \mathcode や \fam については何も言及してませんでしたので、最後になっていきなり“その 1”や“その 2”の方法を持ち出すわけにはいきません。

それで、\fam や \mathgroup の値の変更が効くのであれば、数式アルファベットフォントコマンドも効くんじゃないかなと思って、本編末尾の例では、\mathalpha のグリフを letters にするような “\matheuler” という数式アルファベットフォント名を導入しました。

苦し紛れその 4　operators 用のフォントを用意する

最後の方法は（ホントはこれが一番最初に思い付いたのですけれど）、Euler の数字を含んだ operators 用のフォントを用意して、それに \SetSymbolFont すればいいんじゃないかな、というものです。

繰り返しになりますが、eulervm パッケージでは、symbols のフォントの 128 番以降に、普通なら operators から持ってくるグリフの一部を移しています。なので、真面目にやるならこれらのグリフ群についても対処すべきなのですが、ここでは手抜きをして、0 ～ 9 の数字のグリフについてだけ考えることにします。

operators 用のフォントとしては、デフォルトと同じ OT1 の cmr10 を使うことにして、その数字を eurm10 の数字で置き換えた “mydigit10” という vf を作ることにしましょう。

これくらいなら手作業で頑張っても大した作業量ではないのですけれど、今回は fontinst を使ってみました：

```
\input fontinst.sty
\installfonts
\installfamily{OT1}{myzeu}{}
\installfont{mydigit10}
   {cmr10,unsetnum,eurm10}{OT1}
   {OT1}{myzeu}{m}{n}{}
\endinstallfonts
\end
```

というファイル make_vf.tex を作って[6]、適当な作業フォルダに、cmr10.afm、eurm10.afm と一緒に入れます。そして：

```
prompt> tex make_vf
```

とすると、“.mtx” や “.pl” といった中間ファイルもできますけれど、欲しいのは mydigit10.vpl と ot1myzeu.fd です。続けて：

```
prompt> vptovf mydigit10.vpl
```

とすると、mydigit10.vpl から、mydigit10.tfm と mydigit10.vf が得られます。

これらの mydigit10.tfm と mydigit10.vf、それから ot1myzeu.fd の 3 ファイルを TEX から見えるところに配置した上で、myzeu を operators に \SetSymbolFont します：

```
\documentclass{article}
\pagestyle{empty}

\DeclareMathVersion{euler}
\SetSymbolFont{operators}{euler}{OT1}{myzeu}{m}{n}
\SetSymbolFont{letters}{euler}  {U}{zeur}{m}{n}
\SetSymbolFont{symbols}{euler}  {U}{zeus}{m}{n}
\SetSymbolFont{largesymbols}{euler}{U}{zeuex}{m}{n}

\newcommand{\mathsampleE}{%
 A_n=\sum_k{2n\choose n+k}=\sum_k{(2n)!\over(n+k)!\,(n-k)!}
}

\begin{document}

\[\mathsampleE\]

{\mathversion{euler}
\[\mathsampleE\]}

\end{document}
```

この test_mathsampleE.tex を pdfLaTeX で処理しますと、次のようになります：

$$A_n = \sum_k \binom{2n}{n+k} = \sum_k \frac{(2n)!}{(n+k)!\,(n-k)!}$$

$$A_n = \sum_k \binom{2n}{n+k} = \sum_k \frac{(2n)!}{(n+k)!\,(n-k)!}$$

このサンプルでは、“=”、“+”、“!”、“(”、“)” が eufm10 ではなくて cmr10 なので、eulervm パッケージとまったく同じ結果にはなっていないのですけれど[7]、なんとなくそれっぽい雰囲気は出ているので、まぁひとまずはこれでよしとしましょう。

[6] ここでは簡単に unsetnum.mtx を使いましたが、Philipp Lehman, *The Font Installation Guide*, 2004 の “Tutorial III. Optical small caps and hanging figures” には、別解として、“suffix” という ⟨*optional modifier*⟩ を使う方法が載っています。

[7] eulervm パッケージが symbols に追加しているグリフ群は Euler Fraktur Medium から持ってきているので、fontinst を使ってそれらのグリフを cmr10 から一旦クリアしてから当該スロットを eufm10 で埋めれば、同じような感じになるんじゃないかなとは思われるのですが。