

1. In order for the lower limit to be set close enough to the integral sign, as in \int_a^b , the integral sign would have to be set on a non-rectangular block, like



An even more complicated block would be needed in order to stack two integral signs next to each other to form a double integral sign, although there could just be special symbols for those, as the *MathTime* fonts now have.

2. The largest parentheses in the Computer Modern fonts (and the largest that are chosen by `\left... \right` in the *MathTime* fonts) have a height plus depth of 30pt, while the next largest parentheses, totaling 36pt, are made of top and bottom pieces, and it's a matter of terminology whether the latter should be considered extensible or not! If we assume our formula is vertically centered, as in the example given, then, roughly speaking, we will get extensible parentheses when $\Delta =$ the height plus depth of the formula is greater than 30pt, or greater than 36pt, depending on which view we take.

It's actually quite a bit more complicated, however, depending on certain T_EX parameters: the `\delimiterfactor` f , which plain T_EX sets to 901, and the `\delimitershortfall` l , which plain T_EX sets to 5pt. Instead of requiring that the formula be completely covered by the parentheses, T_EX only requires that the height plus depth of the parentheses should be at least $\Delta - l$ and at least $(f/1000)\Delta$; thus, to get the 30pt parentheses, we need $30 \geq \Delta - 5$ and $30 \geq .901\Delta$, or $\Delta \leq 33.296337$.

That, at any rate, is what should be the case, but in my experiments I found that in plain T_EX, with the Computer Modern fonts, one got the 30pt parentheses for $\Delta = 33.3$, although not for $\Delta = 33.4$. No doubt Knuth would know the precise reason for this (apparent) discrepancy.

The largest radical symbol in the Computer Modern fonts likewise has a height plus depth of 30pt. The rules for square root symbols are even more complicated than for parentheses; I found by experiment that if the formula has zero depth, then the switch from the largest symbol to an extensible one occurred somewhere between a height of 28.1pt and 28.2pt.

3. PostScript sets a limit of 80pt for the height of characters, so parentheses over this height couldn't be on the font. One might try to make a font with parentheses going up to 1 inch which would then be used at 4 times the design size, but in that case the smaller characters wouldn't involve enough PostScript units to be drawn well.

4. $\partial^2 f$ looks so bad because the f slants so much; $\partial^2 |f|$ doesn't have this problem and also points out that the f requires so much space to its left so that it won't overlap the $|$ that precedes it. The combination ∂f doesn't have this problem because f and ∂ are on the same font, and can therefore be kerned. But ∂^2 doesn't "inherit" this kerning. I use a special control sequence `\psqf`, defined as `\partial^2 \kern-1pt f`.

I don't see any way to have T_EX fix this automatically, though perhaps an extension of T_EX might allow kerns to be "inherited" in simple cases.

5. In the Computer Modern formula M^i the superscript i is extremely close to the M . (I find it a strain to look carefully at this formula, though it's OK at a quick glance.) But with the *MathTime* fonts, where the strokes are thicker, having the superscript so close would be bad (at least to my eyes). This means that the italic correction of the M , which determines how far to displace the superscripts, is larger for the *MathTime* fonts. But T_EX always adds the italic correction after a character in math, so a combination like MK has the characters too far apart (the *MathTime* upper-case letters also slant more than the Computer Modern ones, which exacerbates the problem), and kerning is used to get MK (versus MK in Computer Modern). Some people might want the M and K to be kerned even more,

but I felt that in circumstances like this, where multiplication $M \cdot K$ is normally implied, it was better to have some separation between the characters, so that they don't appear too connected.

6. $\mathcal{A}\mathcal{M}\mathcal{S}$ - TEX has `\sptilde` so that one can type $(A^1+ \cdots+ a^n)\sptilde$ to get $(A_1 + \cdots + A_n)\sim$.

7. The problem here is that TEX likes to lower the radical sign to cover the depth of the y . You can use `\smash` to get rid of the height and depth of the formula, and then add a "strut" to get enough height

```
\sqrt{\vrule width0pt height8.8pt \smash{x^2+y^2}}
```

($\mathcal{A}\mathcal{M}\mathcal{S}$ - TEX has `\botssmash` to get rid of only the depth, but in this case it turned out that a strut was still needed to keep the top of the radical sign from being too close to the formula.)

8. You need to type `'If\/'` because the slanted f is followed by a straight character.

If you are using $\text{L}\text{A}\text{T}\text{E}\text{X}$ and typing the math formula as $\langle 0 < k < p \rangle$, then you can set things up for TEX to do this automatically as follows.

```
\let\oldstartmath=\(  
\newskip\saveskip  
\def\langle{\saveskip=\lastskip\ifdim\saveskip>0pt\unskip\/\hskip\saveskip\fi\oldstartmath}
```

If you are using plain TEX , or using $\text{L}\text{A}\text{T}\text{E}\text{X}$ but still like to type $\$0 < k < p\$$, then things are more complicated.

```
\let\mathdollar=$  
\catcode'\$=\active  
\def$\{\ifhmode [code from before] \mathdollar\else\mathdollar\fi}
```

If you are using plain TEX and typing $\$\$$ for displayed math, then you need something like

```
\def$\{\futurelet\next\dodollar}  
\def\dodollar{\ifx\next$\def\DoNext$\{\mathdollar\mathdollar}\else  
  \def\DoNext{\ifhmode ... \fi}\fi  
  \DoNext}
```

Notice that in the first case we define `\DoNext` to have the next $\$$ as part of its syntax, so that it will be swallowed up. Another way would be to

```
\def\DoNext##1{\mathdollar\mathdollar}
```

9. (a) There will be only an ordinary interword space after FORMS. because TEX will assume the period after the uppercase S indicates an abbreviation.

(b) The hyphen will look bad, because it is designed for use with lower-case letters and will be too low for the upper-case letters. Also, the spacing after the ; won't be so good.

(c) You can easily add a higher hyphen to the base font, by creating a virtual font using the text font as base, and adding the hyphen, together with a vertical change, from a copy of the base font to it. The semi-colon could probably best be added by combining the comma with a period placed higher above it than in the standard semi-colon. (Instead of using an exact copy of the base font as the second font, it might look better to use a slightly enlarged version, etc.)

You could then make the `\uccode` of $-$ be the position of the new hyphen, and the `\uccode` of $;$ be the position of the new semi-colon, etc. It would even be useful to have another period, with the `\uccode` of $.$ being its position. Then one could assign different `\sfcode`'s to these new positions.

(d) If you added specially designed accents to the virtual font, and you had a new `\ifupper` and redine `\uppercase` to make `\uppertrue`, then the definitions of various accents could be changed to give the new accent `\ifupper` and the old one otherwise.

10. I didn't word this problem correctly. I should have asked how you can get T_EX to get around this problem automatically, assuming that the comma used in math is the same shape as the text comma, and how virtual fonts could be used if they aren't.

Assuming for the moment that we don't have to worry about the case where the comma in the text font is noticeably different from that in the math font, we need to have the `\)` or `$` used to end a formula look at the next token and see if it is a period or comma—and if so, that token must be appended to the formula before the formula is ended.

```
\let\oldendmath=\)
\def\){\futurelet\next\endthemath}
\def\endthemath{\ifx\next,%
  \def\DoNext,{,\oldendmath}\else
  \def\DoNext{\oldendmath}\fi
\DoNext}
```

(we really need another clause similar to the first in case the next token is a period). See the remark in answer 8. concerning the definition of `\DoNext`.

If the comma in text has a different shape, then we could make a virtual font using the math font as the base, and adding a copy of the text comma somewhere. We would also have to add new kerns: if `M` and `,` have a kern in the math font, then `M` and the added “text comma” needs to have the same kern, and similarly for all other characters in the font. Then in the above code, we need to have

```
\def\DoNext,{\code for added comma}\oldendmath}
```

with corresponding adjustments for periods.