Dozenal Mathematical Displays Using LaTeX

by Donald P. Goodman III

Prior to the advent of computers, dozenalists were easily able to use new symbols for ten and eleven simply by putting new symbols on their typewriters. Since computers became standard, however, dozenalists have been increasingly confronted with the fact that computers are designed around interacting with human beings in decimal. The two character encodings in common use today, ASCII and Unicode, both presume the decimal system, providing only ten slots for numeric digits. Since ASCII and the first 127 slots of Unicode are identical, in the future a slight reform of them, perhaps removing some rarely used unprinted characters from the 0-32 range to include digits for ten and eleven, is clearly necessary; until that time, however, the discipline-standard method of typesetting mathematics, LaTeX, provides an easy method for the use of whatever symbols are desired in a way mostly transparent to the author.

TeX and LaTeX 1

For those readers who are not familiar with it, TeX is a typesetting engine designed by famous mathematician and computer scientist Donald E. Knuth. TeX is capable of typesetting arbitrary text; however, Knuth specifically designed it to be good at typesetting mathematics, and since its release it has been a de facto standard among many mathematicians.

LaTeX is a layer of macros running on top of TeX, designed by Leslie Lamport. It automates most of the nitty-gritty of typesetting in TeX; these days, very few people typeset directly in TeX, and most people use LaTeX to provide a layer of abstraction over the powerful but difficult TeX typesetting language. The American Mathematical Society also provides additions to LaTeX to facilitate the typesetting of high-order mathematics; the results produced by LaTeX are generally agreed to be unequalled by other digital typesetters.

The concept of TeX and LaTeX is simple. Most of the work of typesetting can be automated by the computer. Why not, then, let the computer do it, and make the information given to the computer (the user input) as simple and portable as possible? Most computer users these days are accustomed to a "word processing" model; that is, they type their words into a special program which displays them on the screen (theoretically) the same way that they will appear in the final product, saving the words in a usually proprietary binary file that is largely inaccessible to any program but that in which it was written.

TeX and LaTeX, on the other hand, accept the user's input (the document to be typeset) in a plain text file, which includes "markup" which provides instructions to the computer when it is necessary. This provides a file in which the data is always accessible and which can be edited and manipulated by any program

capable of dealing with plain text input streams—that is, pretty much any program worth using.

For example, in a word processing program, when the user wants to emphasize a word, he highlights it, selects "italics" from whatever menu it might happen to be in, and then moves on. In LaTeX, however, the author must think not how he wants the word to appear, but rather what he wants it to do. If the italicization is meant to emphasize the word, then, the user will type into his file "\emph{word}", which instructs LaTeX to typeset the word with emphasis. A document class file (programmed by somebody else, and specified by the user) will tell LaTeX how emphasized text should appear. Normally, of course, this will be in italics.

While this system presents a higher learning curve than conventional word processors, it also produces better results and more portable input files. For this reason, LaTeX is extremely common in the academic community. Furthermore, its basis in TeX means that LaTeX, too, is extremely powerful when it comes to typesetting mathematics, bringing it prominence in the mathematical field, as well.

2 Macro Indirection

TeX, and those systems built upon it (like LaTeX), is a macro language, and thus provides an easy way to supply extra symbols for non-decimal number systems mostly transparently. This macro system allows new symbols to be seamlessly entered into sequences of numbers, providing minimal distraction to the author and being produced perfectly by the user:

`\x44 = 1492` yields 𐅂44 = 1492

While still easy to read in the LaTeX code, this markup still produces the appropriate typeset result.

There are two main ways of achieving this result, both extremely simple to implement. The first would be to simply define the macro to be used (in this case, "\x") to produce an image, which contains the symbol one wishes to use for that digit. However, this solution presents some problems. Most specifically, it's difficult to make it flexible. Producing the symbol "𐅂" from "\x" in normal text would be fine; however, if the number appears in italic text, or in a different size from normal, the code to implement this solution begins to become overly complicated. While still easy to read in the LaTeX code, this markup still produces the appropriate typeset result.

The other solution is to design special fonts, which contain only two characters each; specifically, the characters DIGIT-TEN and DIGIT-ELEVEN. These characters can, of course, assume whatever shape one desires. Once these fonts have been created, TeX will automatically select the correct size, style, slant, shape, and whatever other attributes have been selected. This solution is so simple that the entirety (not including the font files themselves, of course) can be implemented, with the help of a simple helper package, in a mere six lines.

The advantage of using LaTeX with either of these methods is that any symbol can be substituted for DIGIT-TEN and DIGIT-ELEVEN. So, for example, if one

wished to use a simple "χ" and "ε", one simply redefines the macro in order to indicate this. As we shall see shortly, redefinition of the macro is trivial. Thus, we have a truly easy yet powerfully flexible system for producing professionally typeset dozenal documents in the customary professional mathematicians' language.

Implementation: The Dozenal Package 3

The author has used these ideas to produce the dozenal package, a LaTeX package which utilizes a conversion macro written in eTeX by David Kastrup (a coder well-known and respected in the LaTeX community). Naturally, this required the production of font files for the symbols DIGIT-TEN and DIGIT-ELEVEN to match the default LaTeX font, Computer Modern; these files were produced, in both Metafont and Postscript Type 1 forms (the latter produced by computerized tracing of the former). The Metafont fonts are called "`dozchars`"; the Postscript Type 1 fonts, in order to conform to the Berry Type 1 naming scheme, are named "`fdz`". These fonts approximate the Pitman characters {ↂ ↈ}, well-known throughout the dozenal community.

The conversion macro designed by Kastrup permits the easy redefinition of all automatic LaTeX counters to produce dozenal output. Therefore, when the package is selected (the user does this simply by including the line:

```
\usepackage{dozenal}
```

in his document's preamble), all the counters will be produced in dozenal. This means that page numbers, footnote numbers, section numbers, chapter numbers, and so on will all be printed in their normal places unchanged, except that they will be printed in dozenal, without the user having to do anything at all.

The characters which will be used for DIGIT-TEN and DIGIT-ELEVEN in those numbers, of course, depend upon the definitions of the macros. By default, the macros will select the "`dozchars`" or "`fdz`" fonts, which are visually identical and differ only relative to internal font definitions, and print them wherever "ↂ" and "ↈ" should be printed. However, if the user prefers to use different transdecimal characters (those which symbolize digits greater than or equal to ten), he or she need merely include his or her own font, or select his own symbols, and redefine the macros to use them instead. As mentioned earlier, this process is nearly trivial. To employ the Greek characters chi (χ) for "ↂ" and epsilon (ε) for "ↈ", which the author generally did before his Pitman fonts were production-ready, one can simply redefine the macros thus:

```
\renewcommand\x{\ensuremath{\chi}}
\renewcommand\x{\ensuremath{\epsilon}}
```

A better solution would be to use special fonts which contain the characters desired to be used for DIGIT-TEN and DIGIT-ELEVEN. For example, suppose there were a font containing Dwiggins transdecimals { χ ε } designed to blend with the Computer Modern Fonts. Because this solution has greater capability, it also involves greater complication; however, it is still quite simple, and implementing the new symbols is a simple matter of changing a few lines of LaTeX code. Assuming that the font is called "`dwig`" and the symbol for DIGIT-TEN is

contained in slot 88 (the slot for uppercase letter "X") and the symbol for DIGIT-ELEVEN is contained in slot 69 (the slot for uppercase letter "E"), and assuming the font definitions (which themselves consist of only nine very simple lines) are already available, all the user need do is enter this into his or her document:

```
\renewcommand\doz[1]{{\fontfamily{dwig}\selectfont #1}}
```

For the rest of the document, "`\x`" and "`\e`" can be used to refer to DIGIT-TEN = "χ" and DIGIT-ELEVEN = "ε", and the final document will be produced containing the Dwiggins numerals { χ ε }, rather than the Pitman numerals { ↄ ε } included by default with the dozenal package.

Of course, even this isn't something we should expect the typical LaTeX user to do. Such wizardry should normally be included in a style file, or as an option in the dozenal package itself, and implemented by the user simply by referring to that style file or option. For example, if the Dwiggins characters were defined in a new package ("style file"), the user would simply enter

```
\usepackage{dwiggins}
```

and his solution would be automatically implemented. If, on the other hand, the dozenal package were to include an option for using Dwiggins rather than the default Pitman characters, the user would instead enter

```
\usepackage[dwiggins]{dozenal}
```

and Dwiggins numerals would be used in place of the default Pitman characters throughout the document.

4 Work to Be Done

Naturally, the true solution to this problem is for dozenal characters, agreed upon by the vast majority of the dozenal community, to be included in every font and encoded by default into ASCII and Unicode. However, while this is the perfect solution, it's a solution that remains far in the future. Until then, easy ways of using dozenal characters in typeset works, particularly typeset mathematical works, need to be designed and pursued. The dozenal package for the widespread and popular LaTeX system thus fulfills a longstanding need.

Further work needs to be done, however. Most especially, the author of the dozenal package believes that he has produced credible Pitman characters for the package; however, he is not a type designer, and is well aware that these characters could be considerably improved by a competent font artist. Furthermore, not everyone prefers the Pitman numerals. Unless everyone were to accept the Pitman numerals, a solution the author would admittedly prefer, fonts containing other characters would be appropriate. We have already seen how such solutions might be implemented and how easy they would be for the LaTeX user.

The dozenal package has laid the groundwork for this future work. It is now possible to produce consistently dozenal documents in LaTeX, the typesetting language of mathematics and science. Maintaining and further extending this capability will greatly advance the cause of dozenalism in the world.