Preparing readable documents containing mathematical expressions is a job for professionals. The conventions that mathematicians follow are intricate and exacting. A small change in the shape or position of a symbol or the font used to typeset a variable sometimes constitutes the difference between sense and nonsense or between a true proposition and a false one. Not surprisingly, the authors of mathematical papers and books care a great deal about these small details of formatting and spacing.

Computer software has made it somewhat easier for those authors to prepare their own documents, getting all the little details right. The conventions that mathematicians follow can be programmed into the software that carries out the formatting. Much of the knowledge and experience of human typographers, font designers, typesetters, and printers can also be made explicit and formal and programmed into the software, modularized and packaged so that the document author can take advantage of all that hard-won knowledge without having to possess it.

To the users of software tools for creating mathematical documents, the most striking difference among them is the kind of interface they provide. The designers of some such tools prefer graphical user interfaces that display the document as the author prepares it, showing what it will look like when it is printed and asking the author to enter and position the elements of mathematical expressions by selecting them from menus, moving around inside complicated expressions using arrow keys or the mouse, and tweaking the results until they look right.

The alternative approach is for the author to use a conventional text editor such as vim or Emacs to create text files with embedded commands to direct the software. For example, to command the software to typeset the somewhat exotic punctuation mark known as a dagger, ‘†’, in the middle of a paragraph, the author might type the command ‘\dag’ into the file.

Files that authors create in this way are metadocuments. They are in some ways like text documents and in some ways like programs. Depending on the intentions of the software designer, the notation used for the embedded commands can be simple and straightforward, but limited in flexibility, or it can be as elaborate, subtle, and extensible as a high-level programming language. In fact, these notations are special-purpose programming languages designed for mathematical typesetting. The formatting software can interpret the letter T at the beginning of this sentence as a (trivial) command: “Typeset the letter T.” But the author/programmer can also give commands (like ‘\dag’) that have less trivial effects, including side effects on the state of the virtual typesetting machine that the software implements.