Progress in LaTeX typesetting

Cezary Kaliszyk
Universität Innsbruck

Vincent van Oostrom
Universität Innsbruck

***Abstract*—The Mizar Mathematical Library is a one of the largest collections of machine understandable formal proofs encompassing many areas of today mathematics including results from algebra, analysis, topology, and lattice theory. The Mizar system has so far been the only tool able to completely process, certify, and make use of these developments.**

I. Introduction

COMPUTER certified formal proofs are today one of the most important techniques used in formal methods. They are used to guarantee the correctness of compilers [1], operating systems [2], hardware [3], as well as to certify mathematical results that involve computation [4] [1].

- We provide an infrastructure for more elegant proofs (section II);
  section II.
- We formalize all basic algebraic Mizar structures in Isabelle (Figure 1).

II. Structures

Every Mizar structure signature is defined as a as set of assignments. Each assignment is of the form *sel* → *spec*, where *sel* is a unique structure element label (called selector in the Mizar language) and *spec* is the specification of the type of the respective element of the structure. The signature of a group is the addLoopStr structure. It is specified in MML as follows:

```
struct (ZeroStr,addMagma) addLoopStr (#
      carrier -> set,
      addF -> BinOp of the carrier,
      ZeroF -> Element of the carrier #);
```

And example Isabelle formalization is:

```
definition TheSelectorOf (the - of -  190) where
  func the sel of Term → object means λit.
    for T be object st [sel, T] in Term holds it = T
```

References

[1] X. Leroy, "Formal verification of a realistic compiler," *Commun. ACM*, vol. 52, no. 7, pp. 107–115, 2009.
[2] G. Klein, J. Andronick, K. Elphinstone, T. C. Murray, T. Sewell, R. Kolanski, and G. Heiser, "Comprehensive formal verification of an OS microkernel," *ACM Trans. Comput. Syst.*, vol. 32, no. 1, p. 2, 2014.
[3] J. Harrison, "Floating-point verification," *J. UCS*, vol. 13, no. 5, pp. 629–638, 2007. doi: 10.3217/jucs-013-05-0629
[4] G. Gonthier, "The four colour theorem: Engineering of a formal proof," in *Computer Mathematics, 8th Asian Symposium, ASCM 2007*, ser. LNCS, D. Kapur, Ed., vol. 5081. Springer, 2008, p. 333.

[1] Part of the EwA course.

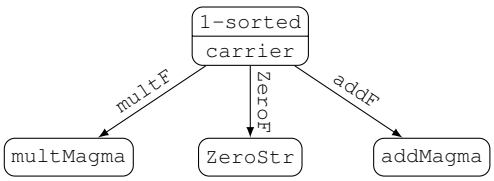

Figure 1. Lattice of the basic algebraic structures

APPENDIX

A. File processing

TEX produces output by glueing together boxes. Each letter yields a box, and boxes for letters are glued together horizontally to generate the box for a word, etc.. In turn, lines are glued together vertically to form bigger boxes, until the page is full, the big box is output, and the process starts anew.

To add some detail to this (but omitting more advanced features), a box has essentially three dimensions: height, depth, and width. The height is for 'ordinary' letters like a and k, and the depth specifies how much a letter extends below the base line. Height and depth sum up to total height.

When TEX processes a file it always is in one of three processing modes, each of which comes in two flavours: (restricted) horizontal, (restricted) vertical, and (display) math. When processing a paragraph TEXgoes/is in horizontal mode glueing letters together, aligning them along their base line, into words and in the end deciding on how to break the lines such that the formatted paragraph looks nice, using stretchable and shrinkable glue between words for the formatting/outlining. TEX is in vertical mode when stacking such paragraphs on top of each other, again with shrinkable and stretchable glue between then. Entering math mode (see below) makes TEX forget about the notion of word and processes letters instead as individual variables (with spacing appropriate for those, but inappropriate for words).

The above process corresponds quite closely to how typesetters would set type manually.

B. Math mode

The (false) equation $\sum_{i=1}^{n} i^2 = \frac{(n+1)n}{2}$ using LATEX (inline) math mode. The same but using TEX math mode $\sum_{i=1}^{n} i^2 = \frac{(n+1)n}{2}$ (Same output, different way to generate.)

Now in display math mode

$$\sum_{i=1}^{n} i^2 = \frac{(n+1)n}{2}$$

in LATEX, and the same in TEX

$$\sum_{i=1}^{n} i^2 = \frac{(n+1)n}{2}$$

As a rule of thumb: use the LATEX-commands when available; TEX-commands may easily mess up the 'invariants' and 'data structures' LATEX-processing depends on. (For instance, TEX's double-dollar to enter display math mode, does not work well with the fleqn option to amsmath; that is meant to display math aligned to the left, but only affects LATEX-display math.)

C. Bibliography

TEX produces its output after a single pass. To allow for (forward) references, labels (generated by the label command) are stored in an auxiliary file (extension aux), which serves as (additional) input for the next time TEX is run. Similarly, citations (generated by the cite command) are stored in the auxiliary file. This can then serve as input for the `bibtex` command which takes that file, a file of bibliography data (extension bib), and a bibliography style file (extension bst), as input, to generate the bibliography (extension bbl) containing the cited works, as found in the bibliograph data, and formatted according to the bibliography style file. This bibliography is then incorporated the next time TEX is run.

D. Commands

TEX allows you to define your own commands, simplifying both reuse and uniform change in case of expressions occurring more than once. For instance, $\sum_{i=1}^{n} i^2 = \frac{(n+1)n}{2}$ and $\sum_{i=1}^{n} i^2 = \frac{(n+1)n}{2}$ are obtained by using the same LATEX definition twice (see the source).

Commands/definitions can be parametrised: $\sum_{i=1}^{n} i^2 = \frac{(n+1)n}{2}$ and $\sum_{j=1}^{k} j^2 = \frac{(k+1)k}{2}$ are both obtained by instantiating the same TEX definition, with i, n, respectively j, k.

E. Some further links

- The definitive source on TEX is the TEX-book (paper).
- Detexify for searching for symbols you can draw.
- The Comprehensive LATEX Symbol List. More symbols.

Stackexchange has answers to many common TEX and LATEX questions. For instance,

- On the modes TEX is in while processing a file;
- On positioning of floats; and
- On the seven classes of math symbols TEX has.
  (Was not presented in class, but useful to know. Each class comes with its own spacing. For instance, if you want to use the letter R as a relation, then you should assign 'relation-class' status to the symbol resulting in this (see the source): $1\ R\ 2$, to express that R relates 1 to 2. Otherwise spacing is wrong: $1R2$, and output ugly.