H. Hagen PRAGMA ADE

MathML

It is a well known fact that TeX can do a pretty good job on typesetting math. This is one reason why many scientific articles, papers and books are typeset using TeX. However, in these days of triumphing angle brackets, coding in TeX looks more and more out of place.

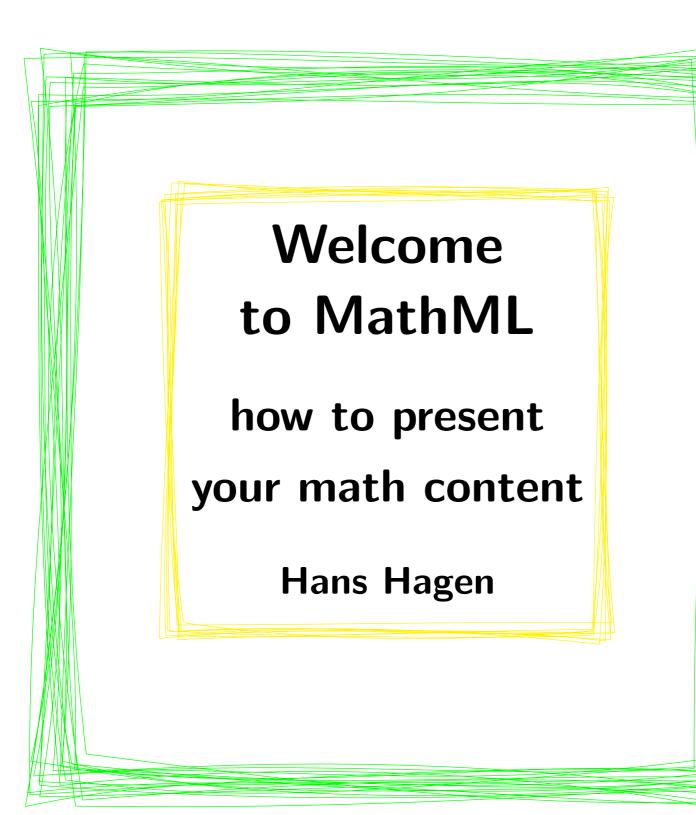
From the point of view of an author, coding in TeX is quite natural, given that some time is spent on reading the manuals. This is because not only the natural flow of the definition suits the way mathematicians think, but also because the author has quite some control over the way his thoughts end up on paper. It will be no surprise that switching to a more restricted way of coding, which also demands more keystrokes, is not on forehand considered to be better.

There are however circumstances that one wants to share formulas (or formula like specifications) between several applications, one of which is a typesetting engine. In that case, a bit more work now, later saves you some headaches due to keeping the different source documents in sync.

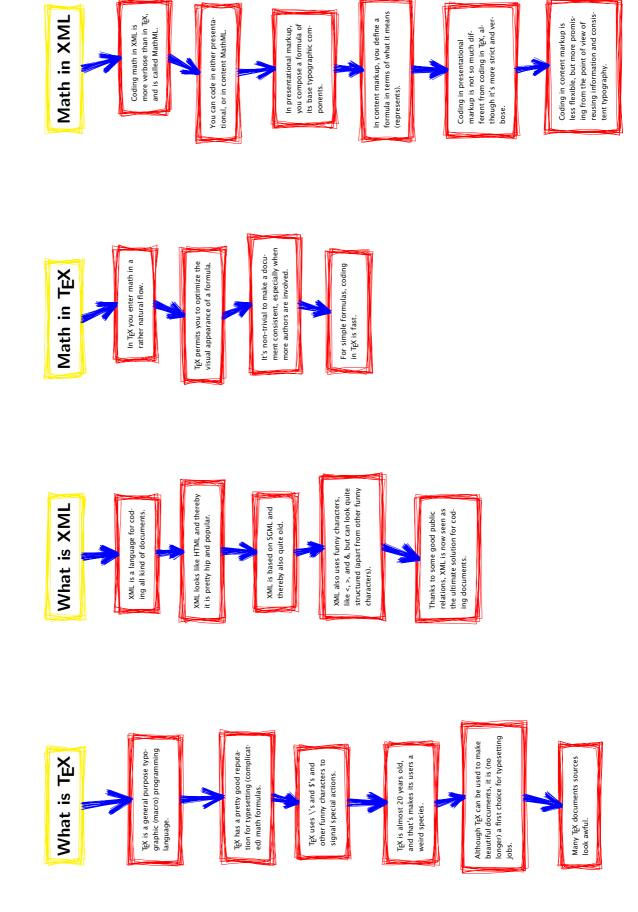
As soon as coding math in angle brackets is discussed, those in favour stress that coding can be eased by using appropriate editors. Here we encounter a dilemma. For optimal usage, one should code in terms of content, that is, the principles that are expressed in a formula. Editors are not that strong in this area, and if they would be, editing would be not that much different from traditionally editing formulas: just keying in ideas using code that at first sight looks obscure. A more graphical oriented editor can help authors to compose formulas, but the under laying coding will mainly be in terms of placing glyphs and boxes, and as a result the code will hardly be usable in other applications.

So either we code in terms of concepts, which permits sharing code among applications, and poses strong limitations on the influence of authors on the visual appearance. Or we use an interactive editor to fine tune the appearance of a formula and take for granted that reuse will be minimal or suboptimal.

In this presentation I will discuss the mathematical language MATHML in the perspective of typography as well as demonstrate how formulas coded in MATHML can be transformed into readable output.



Introduction



MathML support is supported by core xtag filters, to be loaded at runtime. You can embed MathML in normal ConTeXt documents: Such mixed documents can be converted to pure XML quite easily, which provides a nice migration path. There will be much more layout options, as well as support for units, chemistry, and complex formula building. Processing Instruction Given that the formulas are coded consistently, you can influence the layout by providing local or global processing instructions. Occasionally (or for some frequentity) MathML is not rich enough. In that case you add TeX code to your formula. Mixed Markup As an alternative, you can (mildly) enhance a formula marked up in content MathML with presentational elements. emaths eaply each ects at subsequence of the state of the secont of expositions of the secont of the second of the Content Markup You can summarize content markup as: you get typeset what you think. Presentational Markup You can summarize presentation-al markup as: what you key is what you get.

Exploration

