Math notation used in The Netherlands: quick fix or lasting solution?

Drs Dorine in ‘t Veld[1] and Davy Kager[1]

[1] Dedicon Educational, The Netherlands, dorineintveld@dedicon.nl and davykager@dedicon.nl.

Abstract

Some twenty years ago in The Netherlands math had become practically inaccessible, the primary problem being the lack of a braille math code. At DEIMS 2012 Dorine in ‘t Veld explained how this situation came to be[i]. A linear notation was introduced to do math on the computer with braille display support. In the notation the braille output is not defined; it depends on the table that the user selected in their screen reader or braille display. It is not a ‘math braille code’, but the notation worked better than initially supposed, also on academic level. We are now working on generating this notation directly from MathML. Our ‘quick fix’ might prove a lasting solution, maybe even with international importance.

1 Introduction

One of the biggest obstacles in learning math twenty to fifteen years ago was the combination of blind students not learning a math braille code and books being transcribed differently per transcriber. Students integrated rapidly into mainstream schools and worked with a computer with a braille display and speech output. Printed braille was hardly used anymore, especially in mainstream secondary schools.

Books were delivered to the students in WordPerfect and later in Microsoft Word format. For the mainstream teachers – and thus for integration – this had big advantages; they didn’t have to learn braille, since text could be read on the computer screen. Contracted braille was abandoned. So the display of text was ‘one-to-one’ on an 8-dot braille display. Since a Dutch 8-dot table was not available, some students used the American and others the German 8-dot table.

Support by TVIs (teachers of the visually impaired) was minimal. Students received their books digitally, i.e. they received the text version that would produce the right braille when printed. To sighted peers and teachers the notation looked ‘odd’. E.g.: 5 ü2 would mean ‘5 over 2’. Blind students didn’t learn this either, but could look up the meaning in a symbols list in the book. What ü would look like on their braille display, as said, depended on the braille table in use. It likely looked very different from the corresponding 6-dot character on paper.

For notation, they would use : or /, since this is much easier to type, especially on a laptop keyboard. However: 5 /2 in their books would mean ‘5 to the power of two’. This was very confusing for blind students as well as for their sighted peers and teachers who were more used to linear notation as in calculators, where 5/2 definitely means ‘5 over 2[ii]’.

With many more of this type of ‘odd’ and confusing notations, many students and their teachers gave up. Since math was not compulsory for the final exams, braille students for some ten years were advised not to choose math, but to choose text oriented studies instead. Only very few youngsters will choose math when their teachers emphasize that it is impossible for a blind student to do math. But then legislation changed. Math became compulsory again. A quick fix was needed! It was not feasible to choose a braille code and train all TVIs. Besides: should we choose a code from our English, German or Belgian neighbors? Both Germany and Belgium at the time had several codes and what would be better: the English braille code or the Nemeth Code?

In order to make math accessible again, starting around the year 2000 in The Netherlands a notation was developed for braille students working with a braille display, that was a mix of the calculator notation mentioned above and linear notations as in Microsoft Excel or in the command line mode of Mathematica, Maple and other math software. This notation was standardized by a taskforce of math teachers for the VI in primary and secondary education (age 6-18). It required no conversion software. A sighted person could read and write it without much training.

At the start there were doubts if this notation would work, since in advanced formulas many parentheses may be needed. But it looks very much like 'we can train the brain'; students who learned to work with this notation are quite happy with it.

Due to privacy legislation it is very hard to come up with exact numbers. There are about 40 braille students in mainstream secondary education; and in all probability an equivalent amount of low vision students (seeing too little to read normally and who cannot read formulas or graphics). Around the age of 20 we see an increase of blind and low vision students due to accidents, diseases or hereditary illnesses that manifest themselves. Again we don't have exact figures. What we can say is, that nowadays all VI students in The Netherlands do math in secondary school again and we have a steadily increasing number of STEM students, especially in the field of computer science.

Since 2005 Dedicon transcribes the math books in the linear notation mentioned above (that is documented at http://braille.dedicon.nl/wiskunde) and students learn to work with it in the schools.

We recently upgraded the website, replacing the examples that were in .gif format with interactive MathML, allowing zooming without visual quality loss for partially sighted readers. We added a lot of notation that we had in the mean time developed for – and in cooperation with – academic students, requesting books with mathematical content, that we didn't have a fixed linear notation yet.

You might say we have a pragmatic approach. We have to, since math notation is very extensive and we are not an institute that specializes in math. We have to transcribe books for students on demand. Our goals is to give them access to those books and thus the education they have the capacities for. Lately, due to the success of our work and more academic students choosing math, we get more and more demand for more advanced math notations.

The next step, we hope, will be to generate the 'universal linear notation' in the textbooks for students directly from MathML. The reason we choose MathML is obvious: it is at the core of accessibility, compatible with DAISY4 and EPUB3 (the format we will adopt for production), already allows TTS and it will allow our notation to be built into many programs.

2 Design

The linear math notation comprises two independent parts that map directly to tasks for transcription, one to linearize the visual notation, the other to substitute plain text descriptions for the specialized math symbols. Because of their independence these steps can be applied in any order, assuming there exists an appropriate way of storing the intermediate, not yet linearized representation. The aim of the notation is to convey the visual appearance of mathematics. Because of this the semantics of most of the 'code' are undefined. That is to say, there is a definition for the notation of 'something in superscript', e.g. x^2 means 'x with superscript 2'. But there is nothing in the notation to indicate whether this means 'x to the power of 2' or if it is just a superscript, e.g. to indicate a footnote. Similarly, 1 + 2 is nothing more than two numbers separated by a symbol (the plus sign).

The notation doesn't tag this as a summation, and indeed in more advanced – or shall we say abstract – areas of science the plus sign may very well be used like this but with a different meaning than summation. This is analogous to MathML's presentation mode, versus its 'storage-proof' content mode. One example of an exception is the notation of roots, such as sqrt(4) = 2 or root_3(27) = 3. Just as in MathML presentation mode we use special elements for these. More or less the same goes for fractions, but these are more complicated (see section 4).

2.1 Linearization

In this step the visual representation is transformed into a linear form. This is often a very natural process that is also used outside of the field of accessible math. For example, most people will understand what is meant by 3/8, (2 + 1)/(4 * 2), 5^2 and x_1.

Although the notation was not explicitly developed based on the MathML specification, it turns out that a lot of concepts from MathML presentation mode apply to the notation as well. For example, the following MathML fragment can directly be transformed into the linear form 5^2 without adding or removing information:

```
<math> <msup> <mn>5</mn> <mn>2</mn> </msup> </math>
```

2.2 Symbol substitution

In this step mathematical symbols – and sometimes other symbols too – are replaced with a description in plain text. The biggest challenge is to find as short as possible a notation that is still easily – and preferably internationally – understandable. We'll explain this in this subsection.

As it is, the notation is easy to read and write (i.e. to type on a PC), both for sighted and blind people, while the use of most graphical interfaces (like MathType or dialog boxes and toolbars in programs like Microsoft Word) is inaccessible for blind users and often slower for sighted users. Generally it is considerably easier and faster to type plain text than it is to find the appropriate Unicode characters for these specialized symbols in graphical interfaces, e.g. pi instead of the Unicode symbol π (the Greek letter pi).

The Dedicon notation thus defines descriptions for the infinity sign (inf), but also for derivatives, integrals (Intg), logic, set theory, and so on. As an example of the latter, one might write 3 element {1, 2, 3, 4}. The word 'element' simply replaces the symbol $\in$ ('is element of'). This is true even for the Greek letters that are so ubiquitous in mathematics.

At present, Dutch is the predominant language in the Dedicon notation for these descriptions, but a few are in English. For example: 'inf' for infinity – instead of the Dutch 'oneindig'. In many cases it is simple to replace the Dutch with English and often it has advantages since many English terms are conveniently short. E.g. 'not' is shorter than the Dutch 'niet'. Probably for Dutch students doing this level of math it would not be a problem to work with English abbreviations and it might prove a big advantage to be used as an internationally oriented notation on academic level.

3 Strong points

One of the very strong points of the Dutch notation is that it remains consistent notwithstanding in what application it is used. It can be converted from Microsoft Word or HTML to plain text or Markdown without a problem. That is because the notation only uses characters that can be typed with a qwerty keyboard. It can be used for direct communication without requiring additional assistive or conversion software.

As explained in the introduction, ever since the notation was implemented we have an increasing number of braille students passing their final exams in mathematics and science successfully. So our concern, that the notation might not work in more advanced situations that require many parentheses and brackets, was denied. Obviously – or maybe apparently is a better word, since there is no scientific evidence yet – one can train the brain. In other words, the brain adapts to what it is trained to do. In fact when asked, students mostly state that they prefer the Dedicon notation over learning a braille math code. Of course the students cannot really compare the two, since they were never trained in using a braille math code. But still this is a significant finding.

Looking abroad, we see similar problems. Even students who learn a code often write in a calculator notation in their documents. They learned a math code and got their books transcribed in that code, but obviously many students think it cumbersome to write in code. And not only that, many teachers and peers prefer a way to notate math in such a way that both parties can understand and manipulate it. One approach is to provide automatic translation between a braille code such as the Nemeth code and a visual representation, which is what HumanWare now offers in their notetakers[iii].

Our question – and we are still trying to answer it as this paper shows – is whether people can be equally as productive and gain an equally deep understanding of mathematics if they rely solely on a notation primarily designed for linear notation with a computer. Maybe we are taking too much of a shortcut here?

4 Weak points

This notation initially only covered primary and secondary education, not higher education. And it was good for writing, but not good for automated conversion. One of the problems we found was the use of spaces. For example, there is a rule that spaces terminate superscript and subscript text, which is usually very convenient but looks 'wrong' when you combine the two, as in x_2 ^2 (versus x_2^2 which equates to x_4). Note that it is the space between x_2 and ^2 that makes the difference. Mistakes are easily made! In MathML these examples can be written as follows, respectively:

<math> <msup> <msub> <mi>x</mi> <mn>2</mn> </msub> <mn>2</mn> </msup> </math>

<math> <msub> <mi>x</mi> <msup> <mn>2</mn> <mn>2</mn> </msup> </msub> </math>

Consider the previous example again. It can be rewritten as (x_2)^2, which is arguably more readable. The same goes for fractions, e.g. x/2 is clear, but in 4/x/2 we need parentheses or spaces as it could mean (4/x)/2 or 4/(x/2).

The use of parentheses seems a great solution in these relatively simple cases. It helps to identify chunks quickly. But in more complex formulas this approach can lead to a confusing jumble of parentheses and it is not always obvious at once where to place them, e.g. ((a+b)/c)*(x-y^b)/sqrt(a-b). On the other hand: even in spatial notation this formula requires a good display strategy.

Another aspect of using parentheses is that users must understand their meaning and have some basic knowledge of operator precedence. But then, they also need to know this for spatial notation.

Our present notation is convenient enough for daily use by humans. Someone may make a mistake and forget a space or a parenthesis of some kind, or type a space or a parenthesis too much, and one will mostly still understand what a formula was meant to be written like. In that respect we humans are still smarter than conversion programs – albeit much slower. But our 'linear universal math notation' needs to be expanded and improved for automation. And on an academic level, omitting or misplacing a symbol could be considered to be an actual error. The biggest challenge we face in this respect is to create a set of rules that is specific enough to result in a concise and unambiguous notation – and not resulting in a flood of parentheses in relatively simple formulas.

Probably the most intuitive part of the notation is the linearization. As becomes evident from reading our discussion of spaces and parentheses, there are still some problems to work out. But in general, the underlying concepts are clear. The primary weak spot in this area is the notation of text directly under or over other text, e.g. as marked up in MathML with the <munder> and <mover> elements – text under *and* above other text is not a problem. An example is the vector arrow:

<math> <mover> <mo>→</mo> <mi>T</mi> </mover> </math>

We currently notate this as {-->T, where the curly bracket indicates that the next symbol is written above the text that follows it. The obvious flaw in this rule is that it is not immediately clear where the 'over-text' stops and the regular text begins. As a more advanced example, think about how to write complex (imaginative) numbers with this notation.

A final problem is that the notation can become quite verbose, though we try to keep descriptions short by using abbreviations where possible (see also section 2.2). There are a few reasons for our wish for brevity. Long descriptions take longer to type. And there is only so much that fits on a (portable) braille display. Authors of mathematical content use single-space symbols – assuming a monospaced font – so we shouldn't assume we can replace these with lengthy descriptive terms without a loss of readability. In the future we might also investigate embossing the notation on paper in 6-dot or 8-dot braille. This introduces yet more (physical) limits on the length of these descriptions.

5 Improvements: MathML-proof

In 2015 we set out to find MathML notations for all examples that were illustrating our rules and descriptions of symbols that were displayed on our website. This site was created in 2009, when the Dedicon notation was accepted in The Netherlands. Only in the case of the notation for logarithmic formulas we couldn't find MathML that would render the Dutch (European) way of notating this. However, the American notation, e.g. log_10 100 = 2 or log_10(100) = 2, is perfectly understandable by sighted Dutch students and the linear universal notation comes closer to the American than the Dutch notation. In the Dutch print notation the 10 in this example is written to the upper left of 'log' instead of bottom right of it.

Writing the examples in MathML was preparation for rendering the Dedicon notation automatically. There was also an immediate advantage: partially sighted readers can zoom in without losing sharpness of the script.

The process is not yet finished. We are still working on improvements. If we succeed in automatic transcription, the advantages are obvious. Our students will be able to profit from TTS, our notation can be implemented in software like Infty Reader, NVDA, MathType and so on, and others may use it as well. After all, 1/2 is more readable with TTS than 1 ü2, but the natural speech feedback and interactive navigation options of MathML are a clear benefit.

If we write more rigorous rules the notation can be specified with the same techniques used to specify the syntax of programming languages and other (context-free) languages, e.g. grammars. We are therefore hopeful that an automated conversion from MathML will be possible and that the opposite, i.e. a parser for the notation, will be accomplished too.

5.1 Changes are not always welcome

As we said, we identified several inconsistencies, especially in the use of spaces and parentheses. Tightening the rules is necessary, but will this be welcomed by users or will it be rejected as 'yet another change' requiring much more discipline in applying strict rules? Do users even need to follow these strict rules in work that is intended only for private use? And can we simplify nested formulas by removing parentheses based on the rules of operator precedence? That is why we have a usergroup that discusses proposed changes. This may help greatly in understanding how to communicate about the changes in order to get them accepted.

6 Additions

For higher education the notation needs to be expanded. Step 1 (section 2.1) is as good as complete. But the work on step 2 (section 2.2) is far from finished. This is a rather difficult part. Not all cases are defined yet, there are many areas to explore, and sometimes notations in another context may have a different meaning. For the moment we are expanding the notation when a student requires a book where we find notation that is not yet covered. As we do not employ scientists at the highest level in every field – to begin with not in math – we struggle with this. What is the meaning of specific symbols, how do we describe and/or name them – visually and/or semantically? This process is further complicated by the fact that one symbol can mean many different things in different fields or even in different contexts within the same book.

A clear example is the use of arrows, which we describe and notate visually, e.g. → becomes -->. But how do we linearly write an upward-pointing arrow? And a left-pointing arrow above a right-pointing arrow? The same is true for symbols that authors may interpret and use differently, for example the distinction between a subset and a proper subset. We do not have the resources to guarantee that every symbol is described in a semantically correct way in every book. This is a problem primarily for material in higher education.

7 Following established and proven codes

Of course we encounter these challenges in other fields as well. It would be much easier if we could just follow established and proven codes. We did consider using the Nemeth Code or UEB. But yet another change is not welcomed in our country and it would mean that all teachers and itinerant teachers would have to be (re)educated. This may not go well in a country where the focus has shifted from braille on paper to braille as just another output method on the PC that should be readily understood. On top of this problem, we found that in many countries both itinerant and especially mainstream teachers are reluctant to learn a braille math code[iv].

Maybe the solution lies in adopting a notation like AsciiMath, which definitely provides the more stringent rules we are missing. But again, the ease of use and direct legibility of such a standard needs to be considered and tried in the field. In particular, AsciiMath will probably not have the symbol substitutions we require – and it remains to be seen to what extent Nemeth covers those hundreds of mathematical symbols defined in the Unicode standards. But other agencies have worked with AsciiMath and we think we should do more research in this area.

On the other hand, LaTeX is great for specifying symbols but might become tedious or difficult to write, especially for people in primary and secondary education. And it would put them in an isolated position again, since no other students or teachers know LaTeX. That having been said, we do produce books with LaTeX on request, that is: we use a form of ‘Human Readable LaTeX’, leaving out all unnecessary editing commands (like bold, italic, and so on). We only produce them for academic students in STEM-environments, where all other students and teachers ‘speak this language’ and papers are mostly written in LaTeX.

8 Internationalization?

Could our solution on the other hand, be an example for other countries? Or could our notation even be altered in such a way that it would be truly ‘universally’ usable? E.g. if we would use descriptions that are based on the English terms? When we observe how flexible young people are in switching between braille on the braille display and braille on paper and between different languages, we venture to suppose that this surely is an option – a chance even.

Our linear universal notation could be an enrichment even for students who do learn a math code but need a notation that is easy to communicate directly with sighted peers or teachers, allowing both parties to notate in their own preferred spatial or coded notation as well. We hope the quick fix will thus continue to be a useful tool.

---

[i] See Papers from the International Workshop on “Digitization and E-Inclusion in Mathematics and Science 2012″ (http://www.inftyreader.org/?p=315): Dorine in ‘t Veld: Promoting inclusion in mathematics and science in secondary mainstream and higher education

[ii] As you may have noticed, the use of spaces in 5 ü2 is different than in 5/2. Students in primary and secondary education easily adjust, but don’t know why this is so (because they never learnt the math braill code that is at the basis of this use of spaces.

[iii] http://www.humanware.ca/web/en/Newsletter/0312131432.html

[iv] On http://river-valley.zeeba.tv/mawen-mathematical-working-environment/ a nice overview is available on previous attempts to make math more accessible in the EU Micole program. It was in several conversations with project participants that we were confirmed about the resistance to braille math codes growing everywhere, the main reason being that codes complicate communication between blind and sighted people.