Standard dtd's and Scientific Publishing*

N.A.F.M. Poppelier†

and

E. van Herwijnen‡

and

C.A. Rowley§

August 1992

Abstract

This paper has two parts. In the first part we argue that scientific publishing needs *one* standard dtd for each class of documents that is published, for example one for all research papers and one for all books. In the second part we apply this reasoning to mathematical formulas, and we outline some design requirements for a document type definition for mathematical formulas. In the appendices we discuss and compare existing document type definitions for mathematical formulas.

1 Introduction

In the preface to [1] Charles Goldfarb wrote that the Standard Generalized Markup Language can be described as many things, and that SGML is all that – and more. In the introduction to [1] Yuri Rubinsky wrote:

> *ISO 8879 never describes SGML as a meta-language, but everything about its system of declarations and notations implies that a developer has the tools to build exactly what is required to indicate the internal structure of any type of information in a common tool-independent manner.*

Indeed, a strong point of SGML is that it can be regarded as a meta-language, a tool with which one can define the syntax of many languages, very much similar to context-free grammars. In SGML terminology these 'languages' are called document type definitions, called *dtd's* for short. Dtd's can be written for any type of information, e.g. research papers, books and music. A dtd can be used for many purposes, of which two important ones are storage and exchange of information coded according to this dtd.

The premise of this paper is that the exchange of information, if it is based on SGML, needs a single common dtd, agreed upon by all parties involved, for each class of documents that is exchanged.

Suppose two parties, A and B, exchange information in the form of one class of documents, and that they each have a dtd, D(A) and D(B), with D(A) not identical to D(B). If A sends a document to B then A can include the document type definition, D(A), for that document (instance) at the beginning of the document. This enables B to use an SGML parser to check the validity of the document he received. However, there is nothing more B can do with the document: the dtd D(A) contains no information about the *meaning* of the coding scheme that D(A) defines, and a mapping of the document from D(A) to D(B) is a procedure that cannot be automated. The problem becomes even more difficult when a third party, C, is introduced, who accepts material from both A and B. How is C going to handle material with two different coding schemes?

This is where we encounter one of the weaknesses of SGML *as it is being used currently*, namely that it enables every party involved in this process to define and use a different dtd.

2 Scientific publishing

In the rest of this paper we concentrate on the exchange of information that occurs in scientific publishing, in particular on the exchange of papers that contain math-

*Presented at the 9^e NTG meeting, June 4, 1992, Amsterdam. To be published in EPSIG news.

†Elsevier Science Publishers, P.O. Box 2400, 1000 CK Amsterdam, the Netherlands.

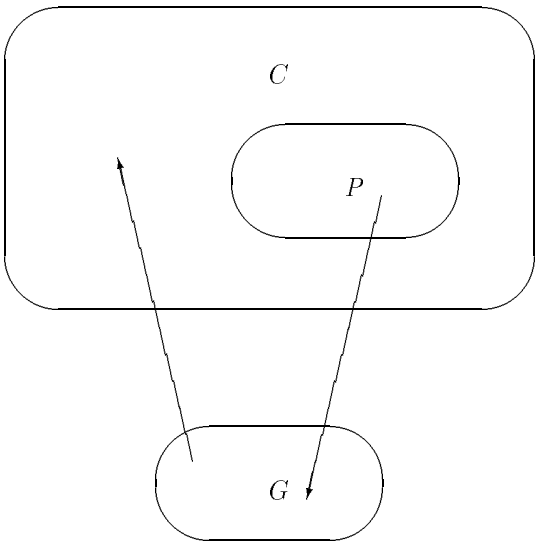
‡CERN, 1211-CH Geneva 23, Switzerland.

§Open University, 527 Finchley Road, London NW3 7BG, UK.

ematical formulas and are published in research journals. Recent developments in this area formed the main reason for writing this paper.

A few standards for encoding of mathematical formulas have already emerged, of which a well-known one is the AAP Standard or Electronic Manuscript Standard [2]. A dtd for mathematical formulas accompanies this standard, but it is not part of it. Another standard for mathematical formulas is the one adopted by CALS [3], and others are under development [4, 5].

The handling of mathematical formulas in scientific publishing is part of the bigger whole of information exchange within a (the) scientific community, with the publisher as intermediary, as is shown in figure 1.



Figuur 1: *Information exchange within a (the) scientific community.*

The authors of research papers are the providers, P. The publishers are the gatherers of information, G. They accept information from many providers, gather this in the form of a journal issue, and distribute this. In this process, the publisher provides a quality check via the system of peer reviewing, makes notation consistent, and in some cases improves the prose. The information is distributed to a group of consumers, C, with the set C a superset of the set P. In this process, two sorts of information can be exchanged:

- material that is structured in the sense of being encoded according to, and checked against, some formal structural specification such as a dtd;
- material that is not structured.

At present most of the material exchanged in the process of scientific publishing is of the unstructured type. We expect that this will remain the situation in the near future. As soon as authors get the possibility of using more sophisticated tools, we expect that publishers will receive increasing numbers of papers of the structured type.

Several scientific publishers, among whom Elsevier Science Publishers, have adopted SGML as the future main tool for the process of publishing scientific articles [6], and several other publishers have made, or are expected to make, the same choice. The European Laboratory for Particle Physics (CERN), a large community of information providers, are using SGML to automate the loading of bibliographic information in their library's database [7]. For both authors and publishers it would be advantageous to agree on one dtd for the encoding of research papers. There are several reasons for this:

- Most authors do not submit all their articles to one and the same publisher every time. At present they are confronted with 'Instructions to Authors' that differ significantly from publisher to publisher.
- A recent trend is that authors prepare their papers with text-processing software on some computer. This enables them to send the paper in electronic form (electronic manuscript or 'compuscript') to the publisher. Publishers are confronted with a variety of text-processing software on a variety of computer systems [8, 9]. Moreover, every field of science appears to have its own 'Top Ten' of most used text processing packages.
- Bibliographic information about all research papers in all (or most) scientific journals is stored in bibliographic databases.

In an ideal world, authors would still be able to use their favourite text-processing system, which would generate SGML 'behind the screens', so to speak. All publishers would accept one standard dtd, and all text-processing systems would be able to generate documents prepared according to this dtd, and all bibliographic databases would be able to store this material.

An example of activities towards achieving this ideal situation: the European Working Group on SGML (EWS) and the European Physical Society (EPS) have taken the Electronic Manuscript Standard and are trying to develop it into a complete dtd, which should be acceptable to information providers, information gatherers and information consumers. The Electronic Manuscript Standard is now a Draft International Standard, ISO/DIS 12083. The EWS and EPS hope that the final standard will include their work.

3 Encoding of mathematical formulas

In Annex A of ISO 8879 [10] we find the following:

> *Generalized markup is based on two novel postulates:*
> 1. *Markup should describe a document's structure and other attributes rather than specify processing to be performed on it, as descriptive markup need be done only once and will suffice for all future processing.*
> 2. *Markup should be rigorous so that the techniques available for processing rigorously-defined objects like programs and databases can be used for processing documents as well.*

There is no reason why this should not be valid for mathematical formulas. We need to delimit the kind of mathematical formulas we are trying to describe if we want an unambiguous structure. The field of mathematics is so vast, that it may be impossible to design a single dtd that covers every kind of mathematical formula. If we concentrate on those sciences which use mathematics as a tool, for example physics, we see that the mathematics used in many physics papers can be described as "advanced calculus". This definition can be made more precise by referring to some standard textbooks containing these types of formulas, e.g. *Handbook of Mathematical Functions* [11] and the *Table of integrals, series and products* [12].

If we aim for rigorous encoding of mathematical formulas (the second postulate), we must develop a system of descriptive markup of mathematical formulas that enables us to:

- convert the formulas between different word processors;
- store the formulas in and extract them from a database;
- allow programs to input or output formulas in descriptive markup.

An example of the first application would be the conversion of mathematical formulas coded in LaTeX to, say, Word[1] via SGML. The benefits of using SGML as an intermediate language for conversion are described in [13]. Note, for example, that the number of programs required for pairwise conversion between n languages is proportional to $n^2 - n$ without an intermediate language, but to $2n$ with an intermediate language.

An example of the second application would be encoding and storing the complete contents of the above mentioned *Handbook of Mathematical Functions* [11] and *Table of integrals, series and products* [12] in a database, so that this information can be accessed on-line by, say, mathematicians and physicists. Many articles have mathematical formulas in their titles, so any program that extracts bibliographic data should be able to handle mathematics as well.

An example of the third application would be the extraction and subsequent use in a computer program, written in an ordinary programming language or, for example, in Mathematica.[2]

At this point we come back to the ideal world for scientific publishing we sketched earlier. In this world, publishers would use one standard dtd for scientific papers, which enables them to prepare a primary publication – in paper and (or) in some electronic form – and to store the information in databases for various secondary purposes.

The question now is: what should a dtd for mathematical formulas look like, if it is going to be used for these purposes?

There are two choices for a dtd for mathematics:

- P-type: the dtd reflects the Presentation or visual structure, Examples of this type are discussed in the appendices.
- S-type: the dtd reflects the Semantics or logical structure At present no dtd's of this type exist.

The quotation from Annex A of ISO 8879 [10] indicates the preference of the creator(s) of SGML: markup of a formula should be of S-type, it should describe the logical structure of the formula, rather than the way it is represented on a certain medium, say the page of a traditional (non-electronic) book.

Let us suppose, for the sake of the argument, that an information gatherer, a publisher, chooses a dtd of S-type. This raises two further questions:

1. Is descriptive markup of mathematical material possible?
2. If it is possible, who can use it and for which purposes?

The second question needs some explanation. As discussed in section 2, in the process of scientific publishing two sorts of information can be exchanged: mathematical material that is structured according to

[1] Word is a registered trademark of MicroSoft.

[2] Mathematica is a registered trademark of Wolfram Research, Inc.

a formal structural specification, and material that is not structured. This means that there are two possible scenarios.

Scenario 1: an author submits a paper in the form of a manuscript (paper), i.e. with unstructured formulas, or a compuscript with mathematical formulas in P-type notation (TEX, WordPerfect, ...).

Scenario 2: an author submits a paper with mathematical formulas in S-type notation.

In scenario 1 it is the task of the publisher to convert from paper or P-type notation to S-type notation. Before we discuss the feasibility of this conversion, we will first look at some characteristics of mathematical notation.

3.1 Characteristics of mathematical notation

Mathematical notation is designed to create the correct ideas in the mind of the reader. It is *deliberately* ambiguous and incomplete: indeed, it is almost meaningless to all other readers. Or, more technically: the intrinsic information content of any mathematical formula is very low. A formula gets its meaning, i.e. its information content, only when used to communicate between two minds which share a large collection of concepts and assumptions, together with an agreed language for communicating the associated ideas.

The ambiguity encountered in mathematical notation can be of two types [14]:

1. A generic notation uses the same symbols to represent similar but different functions, for example '$+$' or '$\times$'. In the case of addition this is not really a problem, but multiplication *is* a problem since, multiplication of numbers is commutative, whereas matrix multiplication is non-commutative!
2. A more fundamental ambiguity is posed by the same notation being used in different fields in different ways. For example: f' stands for the first derivative of f in calculus, but can mean 'any other entity different from f' in other areas.

More examples of ambiguity are:

- Does $\bar{x}$ represent a mean, a conjugation or a negation?
- Is i an integer variable, e.g. the index of a matrix, or is it $\sqrt{-1}$?
- The other way around: is $\sqrt{-1}$ denoted by i or by j?[3]
- What is the function of the 2 in SU_2, $\log_2 x$, x_2, x^2, T_2^2?[4]
- Is $|X|$ the absolute value of a real (complex) number X or the polyhedron of a simplicial complex X [15]?

The inverse problem, which is equally common, arises when different typographical constructs have the same mathematical meaning. For example, the meanings of both the following two lines would be coded identically

$$3 + 4 \pmod 5$$
$$3 +_5 4$$

and this would lead to great difficulty if an author wanted to write:

> We shall often write, for example, $3 + 4 \pmod 5$ in the shorter form $3 +_5 4$, or even as simply $3 + 4$ when this will not lead to confusion.

Of course, natural languages are similarly ambiguous and incomplete, but no one we know is suggesting that in an SGML document each word should be coded such that it reflects the full dictionary definition of the meaning which that particular use of the word is intended to have!

3.2 Who performs the markup of math?

How does one convert P-type mathematical material, which an author has produced, to S-type notation, which the publisher uses?

In [1, p. 9] Goldfarb gives a three-step model for document processing:

1. recognition of part of a document (adding a generic identifier for the appropriate element);
2. mapping (associating a processing function with each element);
3. processing (e.g. translating elements into word processor commands).

In the publishing of scientific papers and books steps 2 and 3 are the responsibility of the publisher. Traditionally, step 1 was also their responsibility: the technical editor adds markup signs in the margin of the manuscript, depending on the text and the visual representation that the house style dictates. It is, however, unlikely that a technical editor is capable of identifying the precise function of every part of a mathematical formula, for several reasons, most of which were discussed in the previous subsection, namely that mathematical notation:

- is not unambiguous,
- is not completely standardized,
- is not a closed system.

Even if the technical editor were capable of identifying every part of a formula, this would be too time-consuming – and therefore too costly. However, under certain conditions [16], automatic translation from

[3] There are examples of authors actually writing something like $[L_i, L_j] = \frac{i}{2} L_k$, where the first i is an index, and the second i stands for $\sqrt{-1}$.

[4] In SU_2 it is the number of dimensions of the Lie group; in $\log_2 x$ it is the base of the logarithm; if x is a vector, the $_2$ in x_2 is an index; the ^{2} in x^2 could be a power, but if T is a tensor, the ^{2} in T_2^2 is a contravariant tensor index.

visual structure to logical structure of mathematical material is simplified greatly.

This, and what we discussed in section 3.1, leads us to conclude the following. A publisher has no choice but to use a P-type dtd for mathematical material that is submitted in unstructured form or in P-type notation. Even if S-type markup of a mathematical formula would be possible, conversion from P-type to S-type would be difficult or even impossible. Conclusion: the tags for S-type markup should not be added by the information gatherer, but by the information providers, i.e. the authors, who should be able to identify each part of their formulas.

3.3 Feasibility of S-type notation

In our second scenario, authors would submit papers with mathematical formulas in S-type notation. This would enable the publisher to 'down translate'[5] to any mathematics typesetting language (P-type notation). However, the same reasoning as in section 3.1 leads us to the following conjecture:

Conjecture. It is impossible to create an S-type dtd for *all* of mathematics.

Representing the "full meaning" of a mathematical formula, if such a notion exists, will almost certainly lead to attempts to pack more and more unnecessary information into the representation until it becomes useless for any purpose. This is rather like Russell and Whitehead reducing "simple arithmetic" to logic and taking several pages of symbols to represent the "true meaning of $2 + 2 = 4$".

Even if it were possible to define an S-type dtd for a certain branch of mathematics, this still gives problems. Supposing an S-type dtd contains an element for a "derivative" of a function. Since the S-type dtd will not contain any presentational attributes, a decision will have to be made to represent the derivative of $f(x)$ on paper as $f'(x)$ or $\frac{\mathrm{d}f(x)}{\mathrm{d}x}$. There are, however, times (such as in this article) that both representations are required for the same semantic object, and that the author will need other notation in addition to that defined by the S-type dtd.

A likely reason for the belief that an S-type dtd is possible, is that many people in the worlds of document processing or computer science are convinced that each symbol has at most a few possible uses and that mathematical notation is as straightforward to analyse as, for example, a piece of code for a somewhat complicated programming language. The reality is that mathematical notation is more akin to natural language: it is ambiguous and incomplete, as we pointed out earlier.

3.4 Some problems with existing languages

To show that it is not obvious to capture mathematical syntax in a dtd, let alone its semantics, consider the example of a limit

$$\lim_{x \to a} f(x).$$

The syntactical structure of a limit is:

- The limit operator
- The part containing the variable and its limit value
- The expression of which the limit is to be taken

The first part could:

- always be "lim", in which case it is just a part of the presentation of the formula and it should be left out.
- be one of a finite list of alternatives, indicating the type of limit (lim inf, sup, max etc.). In this case it should be an attribute.
- be any expression.
- be any text.

We think the second possibility comes closest to the syntax of the limit construct. The second and third parts can be any mathematical expression.

Now let's look at the way this formula is coded with the dtd's from ISO TR 9573, AAP math and Euromath respectively. Using the mathematics dtd from ISO TR 9573 there are three possibilities:

- `lim <sub pos=mid> x &rarr; a </sub> f(x)`
- `<plex><operator>lim</operator><from>x &darr; a</from> <of>f(x)</of></plex>`
- `<mfn name=lim><sub pos=mid>x &rarr; a</sub><of>f(x)</of></mfn>`

The AAP math dtd strongly suggests the following representation:

```
<lim><op><rf>lim</rf></op><ll>x \&rarr;
            a</ll><opd>f(x)</opd></lim>
```

whereas with the Euromath dtd we would have:

```
<lim.cst><l.part.c limitop=limm><range>
                  <relation>x \&rarr; a
</relation></range></l.part.c><r.part.c>
                 <textual>f(x)</textual>
```

We see that the AAP and Euromath expressions are closest to the limit syntax. The best solution from ISO TR 9573 involves a more general "plex" construct, which can be used for integrals, sums, products, set unions, limits and others. When the plex construct contains the actual lower and upper bounds it may even give semantic information.

Some mathematicians, however, are not satisfied with this solution [17]. The plex operation is probably a notation for an iterated application of a binary operation (e.g. sums and products), while limits are of a

[5] 'Down' because information is lost in the process; we borrowed the terminology of translating 'up' and 'down' from Exoterica OmniMark.

different nature. In many cases only the from part will be used, and there the whole range of the bound variable will be indicated, as an interval or a more general set. How does one go about extracting the bound variable?

This supports our conjecture from the previous section, namely that it is very hard to capture the semantics for all mathematics. It also suggests that some redundancy is required to select whichever notation is most appropriate in a certain context.

4 Re-using mathematical formulas

There are two important uses for a generically coded mathematical formula. The first one is in a mathematical manipulation – or computer algebra – system (MMS), such as Mathematica [18] or Maple [19]. Computer programs for the numerical evaluation of formulas, for example written in FORTRAN or Modula-2, can also be regarded as mathematical manipulation programs.

The second form of re-usage is in a mathematical typesetting system, for formatting the formula on paper or on screen; examples of this are TEX [20] and eqn/troff [21, 22].

For computer algebra systems the notation for the formula should be such that a particular type of manipulation on a particular system is possible, given a 'background' of concepts and assumptions that enables the system to interpret the input as a mathematical statement.

The coding of a formula that is adequate for document formatting, for example the TEX notation `f^{(2)} (x)`, is very unlikely to contain much of the information required for a manipulation system to make use of it. However, for a limited field of discourse it is feasible to use the same coding for both types of system [16].

Some examples: the square of $\sin x$ is typographically represented as $\sin^2 x$, but a system like Mathematica or Maple would probably prefer something like $(\sin x)^2$ as input. Typesetting the inverse of $\sin x$ as $\sin^{-1} x$, however, could be confusing: does it mean $1/(\sin x)$ or $\arcsin x$?

An MMS would probably require the second derivative of a function f with respect to its argument x to be coded as $(D, x)((D, x)\, f(x)))$, but on paper this would be represented as $f''(x)$, or $f^{(2)}(x)$, or

$$\frac{\mathrm{d}^2 f(x)}{\mathrm{d}x^2}.$$

On the output side of a MMS there are other problems since some of the coding necessary for typographically acceptable output cannot be automatically derived by the system from the coding used by the MMS.

The Euromath view [17] is that a common interface should be designed together with the manufacturer of a MMS. Perhaps an MMS-type dtd will be required.

5 Related problems

Another problem is, of course, that mathematics is by its nature extensible, so there will always be new types of manipulations to be done. Notations are changed or new notations are invented almost every day, figuratively speaking. Normally these new subjects will use existing typographic representations, but the computer algebra system will not know what formatting to use! Occasionally a new typographic convention will be needed. And although there is agreement on the notation for most mathematical concepts, authors of books on mathematics tend to introduce alternative notations, for instance when they feel this is necessary for didactic reasons. Mathematical notation is not standardized, and it is open—anyone can use it, and add to it, in any way they wish.

If we consider a given dtd at any time, we have to ask ourselves: can an author add elements when the need for this arises? Theoretically the answer is 'Yes, he can' [23, p. 71], although it is not straightforward to include the new elements in the content models of existing elements.

Are such modifications by the author desirable? A dtd which is locally modified by an author will quickly give rise to the situation described in the introduction to this paper, and this should therefore probably be discouraged. Others, however, have also noticed a need for private elements, as described in EPSIG News 3, #4: one of the challenging aspects of using SGML being encountered by the TEI is that the guidelines need to be extensible by researchers. They need to be able to extend the dtd's in some disciplined way [24].

This problem, however, may not be a serious one. The collection of style elements is almost a closed set, since the number of fonts, symbols and ways to combine them is limited. In fact, most notation is not syntactically new, since the limited number of constructs works well as a notation. The multitude of notations is obtained by combinations of fonts, symbols and positions (left or right subscript, left or right superscript, atop, below, ...), and by giving one notation more than one meaning. This again seems to support our view that only a P-type dtd can be constructed for *all* of mathematics.

An SGML dtd, of whatever type, also doesn't solve the problems of new atomic or composite symbols, which occur frequently in mathematics. As with new elements, an author can add entities for these new symbols. There is no method to add the name of a new symbol, whether atomic or composite, to an existing set of entity definitions for symbols, other than to contact the owner of the set and wait for an update.

Although there is now a standard method to describe that symbol's glyph (shape) [25], it is not practical for

an author to include it. A compromise solution seems to be to extend an existing set, such as the one from ISO [26], as much as possible, and try to standardize its use.

6 Conclusions

We have argued as follows:

- That a logical dtd in the sense of describing the structure of the mathematical meaning is as impossible for maths as it is for natural language, and also it is useless for formatting since the same mathematical structure can be visually represented in many different ways. The correct one for any given occurrence of that structure cannot be determined automatically, but must be specified by the author.
- That what needs to be encoded for formatting purposes, is information that enables a particular set of detailed rules for maths typesetting to be applied. This could be described as a 'generic-visual encoding' or 'encoding the logic of the visual structure'. To establish exactly what these codes should be will require an expert analysis (probably involving expertise from mathematicians, particularly editors, and from typographers aware of the traditions of mathematical typesetting).
- That this is different to what needs to be encoded for use in mathematical manipulation software. Since neither of these encodings can be deduced automatically from the other, a useful database will need to store both. Perhaps a separate dtd will be required to enable this communication.

Possible solutions are

- A dtd based on a hybrid of visual structure and logical structure
- Two dtd's, one for visual structure and one for logical structure, that are linked in some fashion
- Two concurrent dtd's, one for visual structure and one for logical structure.

The simplest solution is probably to have a basic visual structure which is described as an SGML entity, supplemented with a (redundant) logical structure, described by a second SGML entity. This solution avoids any special SGML features and gives the user all flexibility for mixing and matching as required.

We believe that similar reasoning can be applied to tables and chemical formulas, where the problem of separation form from content is just as complex, or even more.

A Existing mathematical notations

A.1 Comparison of existing dtd's

In making comparisons between existing dtd's we shall refer often to what is probably the best-known system for coding mathematical notation in documents. This is the version of TEX coding used in LATEX [27] (which differs little from Knuth's Plain TEX notation described in [20]), now a de facto standard in many areas. It is a mixture of visual and logical tagging, with a bias towards the visual which probably results from reasoning similar to that in this paper.

The following document type definitions for mathematical formulas were investigated for this paper: AAP [28], ISO [29] and Euromath [5].

We will try to give a few general characteristics of each of them:

AAP This dtd shows a hybrid of visual and logical tagging. It is quite similar to the mathematical notation of TEX [20].

Integrals, sums and similar constructions have subelements tagged explicitly as lower limit, upper limit and integrand (summand, ...).

The same goes for fractions, roots, and limit-like constructions.

All rectangular schemes of mathematical expressions, e.g. matrices and determinants, are tagged as 'array' in this dtd. The delimiters are not part of the construction, although matrices are usually indicated by $(\cdot)$ or as $[\cdot]$, and determinants as $|\cdot|$ Alignment of rows, columns and cells is indicated by attributes, even though they have nothing to do with function, but are in fact processing information. This idea also appears in the array notation of LATEX [27].

A subscripts or superscript is indicated as such, and not as power, index, ... Greek letters, italics, emphasized letters are all specifically marked up, which could cause ambiguity as regards the semantics of a given symbol.

It contains tags of a non-presentational nature, for example for vectors, dyads and tensors. It is possible, however, to use font or style commands to obtain the same result. Therefore, these tags, and similar ones, appear to be superfluous.

For advanced calculus, this dtd is as complete as is required by many papers. To capture the semantics of calculus, some modifications are required.

Euromath This dtd evolved from a earlier dtd in the Grif [30] system and provides a hybrid of visual and logical markup. One of the design principles was the possibility of conversion to and from mathematical notation in LATEX [27]. This could be dangerous, since the design of a dtd is a modelling activity, and the data being modelled is mathematics, not a text formatter. An immediate consequence is that the semantics that can be associated to a formula is that which is present in LATEX (which excludes tensors). It does, however, show the importance of LATEX for scientific publishing. Another consequence is that the Euromath dtd bears a strong similarity to the AAP dtd.

Alignment of rows, columns and cells is not indicated at all. However, every expression or sub-expression carries an attribute that expresses the 'style' of the expression, a concept borrowed from TEX [20]. This style specifies, among others, the placement of limits of integrals and sums, and the position and size of superscripts and subscripts. The reason for adding this attribute is unclear, since it can be derived from the context.[6]

In the Euromath dtd, a formula is considered as a sequence of parts or *constructions* that are arranged horizontally one after the other, and aligned along their reference axes. This reflects the fact that most formulas can be read aloud, e.g. over the phone, and in that sense are almost one-dimensional. Modes of mathematical expression that are clearly two-dimensional, such as commutative diagrams – see [31, p. 125] or [15, p. 312], are not covered by this dtd. These are tacitly assumed to be part of a higher level dtd.

ISO This dtd is designed for logical tagging, covering mathematical formulas from advanced calculus.

There is redundancy in its notation. For example, elements exist for tensors and their indices, but also for superscripts and subscripts. A good WYSIWYG formula editor should help authors to unambiguously markup their mathematics and to preserve its semantics.

There are tags of a non-presentational nature, for example for vectors and tensors. However, the vector tag can only be used for the object as a whole, and it is not clear how an individual component should be coded. Furthermore, this dtd contains tags for superscripts and subscripts, thus allowing – but not forcing! – a user of this dtd to write 'the square of x' as `x<sup>2</sup>` instead of `<power>2<of>x</power>`.

The ISO dtd has a large overlap with the AAP dtd, as is shown in appendix B. Both dtds can be used as an intermediate language for conversion between word processors. To capture the semantics of a an arbitrary calculus formula, more is required.

B Comparison between ISO TR 9573 and AAP math dtd's

B.1 Formula and formula reference

Note: ISO assume spaces are ignored by the text formatter and that positioning is done by according to the rules of mathematical typesetting. AAP have a NOTATION attribute on their formula that would allow blanks etc. to be ignored, and various characters recognized as operators.

ISO	AAP equivalent	Difference
Inline formula		
`f`	`f`	geo.form NOTATION (AAP)
Display formula		
`df`	`fd`	geo.form NOTATION (AAP)
`df id=`	`la`	
`df align=(left|right|center)`	`la pre post`	1. `align=center` (ISO)
		2. `la` element (AAP) allows multiple numbers
`df num=`	`contents of la element`	
Display formula group		
`dfg`		AAP has no
`dfg id=`		display formula groups
`dfg align=(left|right|center)`		
`dfg num=`		
Formula reference		
`dfref refid=`	`lar`	`lar` element has content; `dfref` is empty
`dfref page=`		the page attribute (ISO) adds the page number

[6] TEX does this too, but the user of TEX can override the program's automatic choice.

B.2 Formula content

ISO	AAP equivalent	Difference
Horizontal and vertical alignment		
`mark id=`	`hmk id=`	identical
`markref refid=`	`hmkr rid=`	
	`vmk id=`	vertical alignment
	`vmkr rid=`	is absent in ISO
Division points in a formula		
`break type=`	`tu`	Both elements are empty. `type=` indicates optionality. `tu` is excluded from various constructs.
Superscripts and subscripts		
`sub`	`inf`	
`sub pos=(pre|mid|post)`	`inf loc=(pre|post)`	mid value (ISO) missing in AAP
`sup`	`sup`	
`sup pos=(pre|mid|post)`	`sup pos=(pre|post)`	mid value (ISO) missing in AAP
Boxes		
`box`	`box style=`	style atribute (AAP) to to change rule type missing in ISO
Over embellishments		
`ov`	`a`	
`ov pos=`	`a valign=`	
`ov type=`	`ac`	`ac` element (AAP) allows simultaneous embellishments
`ov style=`	`ac`	`style` attribute (ISO) can be achieved with `ac`
Tensors		
`tensor`		the tensor element(ISO)
`tensor suffix=`		is absent in AAP.
`tensor posf=`		can be visually achieved with `sup`, `inf` and `zw` (zero width character, absent in ISO)
Functions		
`mfn name=`	`rf`	name attribute (ISO) has a finite list of values.
`fname`		`fname` (ISO) indicates
`of`		arbitrary roman function; of the argument; `rf` (AAP) only marks up the function name
Roman and italic fonts		
`roman`	`rm`	equivalent
`italic`	`it`	
Vectors		
`vec`	`v`	equivalent
Fractions		
`frac`	`fr`	equivalent

`numer`	`nu`	
`over`	`de`	
`frac` `align=(left|right|center)`	`fr align=(r|l|c)`	equivalent
		AAP has shape attribute, `fr shape=(case|built|sol)`, to indicate shape of fraction; missing in ISO
Derivatives		
`diff`	`inc`	The `inc` (AAP) element
`diffof`		is for increments. The type
`by`		is given in the content,
`diff type=partial`		but it is not marked up
General plex (limits)		
`plex`	`lim`	equivalent
`operator`	`op`	
`from`	`ll`	
`to`	`up`	
`of`	`opd`	
`sum`	`sum`	equivalent
`integral`	`in`	
`product`	`pr`	
Piles		
`pile`	`stk`	equivalent
`above1`	`lyr`	the role of `above1` (ISO) is not clear
`above`		
`pile` `align=(left|right|center)`	`align=(l|r|c)`	equivalent
Matrices		
`matrix`	`ar`	ISO marks up rows
`col`	`arc`	inside columns via
`above`	`arr`	above. AAP has elements
		for both.
	`ar cs`	AAP allows various
	`ar rs`	column and row
		separators.
	`ar ca`	AAP allows overall
		column alignment
`col` `align=(left|right|center)`	`arc` `align=(l|r|c|d|e)`	d and e for alignment on exponents and decimal points
Square root and square		
`sqrt`	`rad, rdx 2`	no separate elements
`square`	`sup 2`	for square roots and squares in AAP
Root and power		
`root`	`rad`	The content of `root`
`degree`	`radix`	is the content of `rad`
`of`		in AAP;
		`of` element not required.

`power`	`sup`	No general power
`degree`		element in AAP
`of`		
Open and close brackets, fences and posts		
`fence`	`fen`	equivalent
`fence style=`	`fen style=`	
`fence open=`	`fen lp=`	
`fence type=`	`fen post=`	
`fence close=`	`rp`	
	`rp style=`	
	`rp post=`	
`middle`	`cp`	
`middle style=`	`cp style=`	
	`cp post=`	

B.2.1 Short references

Note: short reference maps are defined for fences in both applications. In addition, AAP has maps for starting rows and columns in arrays, and for certain accented characters.

B.3 Facilities in AAP not available in ISO

B.3.1 Phrases in formulas

The `phr` element changes to roman font inside a formula. The ISO application uses the roman element for this.

B.3.2 Type style tags

By introducing elements that represent type styles (e.g. Greek, bold, sans serif), some redundancy is built into the AAP dtd. They are: bold (`b`), bold German (`bge`), bold Greek (`bg`), bold italic (`bi`), bold italic sans serif (`bsf`), bold script (`bsc`), Greek (`g`), italic (`it`), italic sans serif (`isf`), monospace (`ty`), openface (`op` – note that this element already exists for operator), roman (`rm`), sans serif (`ssf`), script (`sc`), and small capitals (`scp`). It is our opinion that one should differentiate between alphabets and presentation properties:

- Alphabets are: Latin, Greek, Hebrew and Cyrillic (sufficient for math)
  Properties are: roman, bold, italic, slant, sans-serif, script, fraktur, openface,. . .
- All, or most, properties can be applied to the Latin alphabet, as transformations.

Cyrillic is not a transformation of Latin, wheras bold Latin is! Some combinations of properties are also allowed, but there is, for example, no sans-serif fraktur or fraktur Cyrillic, so some combinations are meaningless.

B.3.3 Some maths objects

AAP has tags for dyadics (`dy`), and fields (`fi`).

B.3.4 Horizontal and vertical spacing

AAP has tags for horizontal space (`hsp`) and vertical space (`vsp`), for use in cases where space needs to be explicitly indicated by the author.

B.3.5 Atom change tags

Mathematical formulae are composed of atoms. There are 7 types, adapted from TEX [20]: Ord (ordinary), Op (operators), Bin (binary operation, Rel (relation), Open, Close, and Punct (punctuation). The `ach` (atom change) tag changes a character's atom type.

B.4 Future developments

There is a large overlap between the ISO and AAP (and hence Euromath) dtd's. It should be possible to make a single dtd that contains 'the best of both'. Indeed, a working group is now studying this problem under the auspices of the AAP, and a dtd designed along the lines sketched in this article is under development.

References

[1] Charles Goldfarb. *The SGML Handbook*. Oxford University Press, Oxford, 1990.

[2] Standard for electronic manuscript preparation and markup version 2.0. Technical report, EPSIG, Dublin (Ohio), 1987. ANSI/NISO Z39.59-1988.

[3] Techniques for using SGML. ISO Technical Report 9573, 1988.

[4] American Chemical Society. ACS journal dtd. (unpublished draft version).

[5] Björn von Sydow. On the `math` type in Euromath. (preprint).

[6] N.A.F.M. Poppelier. SGML and TEX in scientific publishing. *TUGboat*, 12:105–109, 1991.

[7] E. van Herwijnen, N.A.F.M. Poppelier, and J.C. Sens. Using the electronic manuscript standard for document conversion. *EPSIG News*, 1:14, 1992.

[8] E. van Herwijnen. The use of text interchange standards for submitting physics articles to journals. *Comp. Phys. Comm.*, 57:244–250, 1989.

[9] E. van Herwijnen and J.C. Sens. Streamlining publishing procedures. *Europhysics News*, pages 171–174, November 1989.
[10] International standard ISO 8879: Standard generalized markup language (SGML). Technical report, ISO, Geneva, 1986.
[11] M. Abramovitz and I. Stegun. *Handbook of mathematical functions*. Dover, New York, 1972.
[12] I.S. Gradshteyn and I.M. Ryzhik. *Tables of integrals, series, and products*. Academic Press, New York, 1980.
[13] S.A. Mamrak, C.S. O'Connell, and J. Barnes. Technical documentation for the integrated chameleon architecture. Technical report, March 1992. (Part of the documentation of the ICA software).
[14] Neil M. Soiffer. *The design of a user interface for computer algebra systems*. PhD thesis, Computer Science Division (EECS), University of California, Berkeley, 1991. Report UCB/USD 91/626.
[15] M. Nakahara. *Geometry, Topology and Physics*. Adam Hilger, Bristol, 1990.
[16] Dennis S. Arnon and Sandra A. Mamrak. On the logical structure of mathematical notation. *TUGboat*, 12:479–484, 1991.
[17] Björn von Sydow. private communication to one of us (EvH).
[18] Stephen Wolfram. *Mathematica: a system for doing mathematics by computer*. Addison-Wesley, Reading, 1991.
[19] Bruce W. Char, Keith O. Geddes, Gaston H. Gonnet, and Stephen M. Watt. *Maple User's Guide*. WATCOM Publications Ltd., 1985.
[20] Donald E. Knuth. *The TEX book*. Addison-Wesley, Reading, 1984.
[21] Joseph F. Osanna. *UNIX Programmer's Manual (2b), chapter Nroff/troff*. Bell Laboratories, 1978.
[22] Brian W. Kernighan and Linda Cherry. *UNIX Programmer's Manual (2b), chapter Typesetting mathematics*. Bell Laboratories, 1978.
[23] E. van Herwijnen. *Practical SGML*. Kluwer Academic Publishers, Dordrecht, 1990.
[24] SGML'90 report. *EPSIG News*, 3(4):4, 1990.
[25] International standard ISO 9541: Font information interchange. Technical report, ISO, Geneva, 1991.
[26] Information processing – SGML support facilities – techniques for using SGML – part 13. Proposed Draft Technical Report ISO 9573, 1991.
[27] Leslie Lamport. *LATEX: a document preparation system*. Addison-Wesley, Reading, 1985.
[28] Markup of mathematical formulas. Technical report, EPSIG, Dublin (Ohio), 1989.
[29] Information processing – SGML support facilities – techniques for using SGML – part 11. Proposed Draft Technical Report 9573, 1991.
[30] Grif SGML editor 2.1, December 1991.
[31] Yvonne Choquet-Bruhat and Cécile DeWitt-Morette. *Analysis, manifolds and physics*. North-Holland, Amsterdam, revised edition, 1987.