TEX-based Production at the AMS*

Ralph Youngen†

American Mathematical Society,
P.O. Box 6248,
Providence, RI 02940, USA

June 1992

1 Background

The American Mathematical Society (AMS) is a both major publisher of mathematical research, and a professional organization whose members are mathematicians engaging in research at academic institutions and other research centers in the U.S. and around the world. A primary function of the Society is to provide channels of communication whereby these mathematicians can communicate the results of their research to each other, and to the broader scientific community. Foremost among these channels of communication is an extensive publications program which is based on the TEX typesetting system.

The AMS headquarters are in Providence, Rhode Island, and a sister office affiliated with the University of Michican is located in Ann Arbor, Michigan. The two sites employ a total of more than 200 people.

Computing hardware at the AMS consists of a VAX/VMS 6320 cluster, a 2-processor Solbourne 5E/902-64 which functions as a database server, a 4-processor Sun 690 MP which is the central server for the Ann Arbor office, a 2-processor Solbourne 5/500 which functions as a TEX server for the VMS cluster, approximately 12 other Unix and VMS workstations which serve as development machines, and approximately 20 PC-compatible and 20 Macintosh computers. High resolution output devices include two Autologic APS-μ5 phototypesetters, and an Agfa/Compugraphic 9600 PostScript imagesetter. Proofing devices include two high-speed QMS 300dpi laser printers, and approximately 20 other 300dpi laser printers. A 300dpi flatbed scanner is used for input of graphic material for the creation of line art and halftones.

The Society currently publishes 20 journals, ranging from tri-weekly's to annuals, as well as nearly 100 books per year, of which 95% are in 20 regular book series. Of the journals, six contain results of primary research, eight are translations from Russian, Japanese, and other languages, and the largest, *Mathematical Reviews (MR),* contains reviews and extensive indexes of current mathematical literature that together occupied more than 10,000 pages in 1991. AMS books and journals combine for about 70,000 pages per year, with 62,000 typeset pages and 8,000 author-prepared pages. In addition to its own publications, the AMS also does some portion or all of the composition, editorial, printing, binding, and distribution for nine other journals published by eight scientific societies. Furthermore, the AMS acts as a service bureau for typesetting jobs composed using the TEX typesetting language. Aside from a conventional desktop publishing segment for the production of a few newsletters, brochures, and posters, and any publications produced from author-supplied copy, all other publications are typeset using TEX, with the final camera copy generated on a phototypesetter at the AMS headquarters.

2 History and Importance of TEX at AMS

The AMS has used electronic publishing methods to produce its publications since 1971. Prior to the early 1970s all mathematical typesetting was done either by using typewriters or hot lead-type machines (Monotype for example). The AMS experimented with various electronic publishing systems through the early to mid 1970s, including paper-tape systems, until settling for a time with a system developed by Science Typographers Inc. (STI). The STI system was able to handle jobs containing mathematical material, but was not useful for multi-column administrative publications such as the AMS membership list or catalog.

In 1978 the AMS learned of a new typesetting system called TEX when Prof. Donald E. Knuth of Stanford University presented the Gibbs lecture at the AMS annual meeting in January 1978. Every year, a well-known mathematician is invited to present the Gibbs lecture on a topic of his or her choosing. Knuth chose to speak about mathematics and computer science in the service of technical typesetting. In the audience were several influential members and officers of the AMS, and they undertook the investigation into TEX and encouraged its adoption as a production tool for the Society's publishing program. Knuth furthered this

*Presented at the 9^e NTG meeting, June 4, 1992, Amsterdam.
†Technical Support Manager, American Mathematical Society.

adoption process by assigning the trademark rights to TeX to the AMS.

The promise of the TeX typesetting system was to provide a new typesetting system intended for the creation of beautiful books—and especially for books that contain a lot of mathematics. Over the past 14 years TeX has truly evolved into a powerful typesetting program, generally regarded as the best system in th world for handling complicated scientific text and displayed mathematical formulas, as well as producing copy near in quality and appearance to that produced by the finest traditional compositors.

Beginning around 1983, the AMS began using TeX for a small portion of its technical book and journal publishing. The AMS found TeX to be very well suited for a high volume of technical typesetting. Thus, the percentage of typesetting done in TeX increased over the next few years, until, by mid 1987, TeX was used for 100% of typeset book and journal production.

There are several factors which have contributed to TeX's huge success. Certainly one of these is the fact that unlike most proprietary composition systems, TeX can be used on almost any known computer hardware from personal microcomputers to Cray supercomputers. The fact that Knuth placed the source code for TeX into the public domain meant that any skilled systems programmer could easily tailor the TeX program to run on any computer. Also unlike many proprietary systems, the cost of most TeX implementations is reasonably low to minimal.

Another significant contribution to TeX's success is that it provided mathematicians with a tool for expressing mathematical notation in a simple text file on a computer. This allowed the mathematician to use the computer both as a means of typesetting mathematics, and for communicating mathematics to other colleagues' computers via electronic mail or other electronic transfers. It is not surprising that TeX is now pervasive throughout academia.

TeX's coding mechanism, however, sometimes appears overly complex and gives the impression to some that it would be difficult to learn. This is true if one wishes to become a devoted TeXnician and learn all of the intricacies of TeX. But TeX's powerful macro capability, which allows a single command to be an abbreviation for a complex set of commands, enables TeX programmers to develop macro packages that greatly simplify the *use* of TeX. The AMS developed such a macro package, namely AMS-TeX, in the early 1980s. AMS-TeX greatly simplifies the author's use of TeX to the point that simple documents can be typeset by example. In the late 1980s the AMS developed the AMS-LaTeX macro package as a result of users requesting better mathematics typesetting for the popular macro package LaTeX.

The AMS became interested in TeX both for the promise of very high quality output, and for the hope that authors would one day be able to submit their files electronically to the AMS for publication. The fact that TeX is in the public domain and therefore inexpensive to obtain, as well as the fact that TeX runs on almost any computer you can name, means that TeX is widely available to a large number of people—especially mathematicians at research centers who are potential authors of technical articles or books for publication. A mathematician at a university with a PC can be writing his or her document in TeX and obtain output from a dot matrix printer or laser printer on site. When the document is finished, the TeX source file can be sent electronically to the AMS, or any other publisher. Thus, the time it traditionally takes to typemark and keyboard the document from paper manuscript is saved, and the author gains in the assurance of knowing that when proof copy is received back from the AMS, the author is reading the same file that he or she has already carefully proofread. For example, the author doesn't have to painstakingly check all of the mathematical equations for correctness because this has already been done before submission to the AMS.

3 Electronic Submissions

The goal of receiving TeX-encoded electronic manuscripts from authors has now been realized. The AMS began its electronic submission program in 1988. The number of electronic manuscripts prepared in AMS-TeX and AMS-LaTeX has steadily increased over the past few years. In 1991 the AMS received nearly 500 electronic submissions, accounting for 16% of total submissions for that year, and this figure has risen to 22% to date for 1992 submissions. Over 75% of these electronic submissions were received via electronic mail. The rest were received on either PC or Macintosh disks sent via postal mail, with approximately twice as many PC disks as Macintosh disks being received.

The objective of the electronic submission program is to receive documents that will merge smoothly into the production stream at the AMS. For the AMS program, the key factors that have lead to realizing this goal are specific guidelines describing the preparation of an electronic manuscript, adequate author support via telephone and e-mail, and a limitation on the types of documents that will be accepted (only AMS-TeX and AMS-LaTeX documents are accepted electronically).

It became evident that early communication with authors, preferably before writing starts, was very important. Two documents entitled *Guidelines for Preparing Electronic Manuscripts* were written in both AMS-TeX and AMS-LaTeX versions. These documents provide the author with both general and specific instructions for the creation of the electronic manuscript, such as:

- a short description of how an electronic manuscript is processed;

- a checklist of information that must accompany a manuscript order to avoid delays;
- instructions for using the logical tags, which ones are required, and examples;
- references to more detailed manuals for the basic TEX package (AMS-TEX or LATEX);
- instructions for submitting the file on diskette, by e-mail or by FTP;
- where to go for help.

The Guidelines do not try to show an author how to use TEX itself (or AMS-TEX or LATEX), but only what is specific to AMS publications and built into the author's macro packages. Although electronic submissions were being received as early as 1988, no formal guidelines existed until 1990. Since the publication of these Guidelines the quality of electronic submissions has improved greatly, as measured by the number of them that require special handling and the amount of attention required from editorial and composition staff.

In addition to these guidelines, generic documentstyles for both AMS-TEX and AMS-LATEX are also available to the author. Both these generic documentstyles and the ones use in production at the AMS are based on generic (SGML-like) markup. In these documentstyles, the elements of a document are tagged for logical structure, not for typographic appearance. In this way an author need not know exactly what publication his or her article will appear in while writing the paper, because logical tagging provides the flexibility of moving a paper from one publication to another by simply changing the documentstyle being used.

The documentstyles provided to authors define the appearance of a "generic journal" format, whose appearance is based loosely on the *Journal of the American Mathematical Society.* Again it is important to emphasize that a paper typeset with this generic documentstyle will resemble an article from the AMS journals, but it most certainly will not be identical in appearance. The purpose of the generic documentstyle is to provide the author with a logical framework into which the document should be encoded. Both AMS-TEX and AMS-LATEX contain definitions for mathematical elements such as displayed and aligned equations, matrices, case statements, etc. Macros for theorems, proofs, section headings, lists, and so forth are provided by the generic documentstyle, and the author is instructed by the Guidelines that these macros **must** be used. A heading in the form `\centerline{\bf Introduction}` is unacceptable because it defines the heading's typographic appearance but not its logical structure. The proper form would be `\heading{Introduction}`.

In fact any typographic-specific TEX commands such as specific font calls, vertical or horizontal space, and page and line breaks are specifically prohibited. The reason for this is twofold: AMS publications are typeset using Times Roman fonts and the differences in character widths between Times and Computer Modern will mean different line and page breaks than what the author sees, and the editorial staff may request changes to the text itself that would affect line and page breaks.

Occasionally an author who is also a TEX expert is troubled by the restrictive posture the AMS has taken toward electronic submissions. These authors generally wish to base their submissions on a favorite macro package they have personally developed or acquired elsewhere. While a few simple definitions to act as substitutes for repeated phrases or other constructs are allowed (even encouraged), submissions which are dependent on any additional packages other than those supplied by the AMS frequently will not merge into the AMS production system without attention from a skilled TEX programmer. This does not result in the anticipated savings of time and money for such submissions.

4 AMS Production Cycle

4.1 Typeset Books and Journals

The Society's TEX-based production cycle is really quite traditional. Manuscripts are refereed and accepted for publication by an editorial board for the book or journal. Members of these editorial boards are mathematicians from academic or research sites, and are not employees of the AMS. Whether or not an author used TEX to prepare their submission has no bearing in the acceptance process.

When an accepted manuscript is forwarded by the editorial board to the Providence office, it is logged into a tracking system which will monitor the paper's progress through various correction runs, until the paper is finally typeset as part of a specific publication. Upon receipt of the paper manuscript the existence of an electronic version is determined. If an electronic version is unavailable, the paper manuscript is delivered to the Editorial Department at the AMS, where the paper is typemarked and copyedited.

Typemarking refers to the process of labeling the logical components of the paper; all headings and constructs such as theorems, definitions, proofs, etc., are clearly marked by the editorial staff. At this point the paper is forwarded to the Composition Department where it is keyed in AMS-TEX. The composition staff consists of about 15 keyboarders, between a third and a half of whom work from home and use terminals connected to the AMS computers via modems and phone lines. The use of terminals and modems rather than PCs was deemed preferable for several reasons, such as centralization of data and ease of communication via electronic mail. Keyboarders working from home have access to the technical staff with electronic mail, and documentation and other information can easily be distributed to the at-home staff electronically.

The at-home keyboarders code the papers in AMS-TEX and run the paper through a syntax check when finished. The paper must successfully pass the syntax checker before it can be submitted for proof copy. These keyboarders do not have graphic terminals nor printers at their disposal, so they are never able to actually view the output of their AMS-TEX coding. They are encouraged to key the paper as quickly as possible, and are paid by the keystroke.

Every night proof copy is generated for all papers keyed that day, and these are delivered back to the editorial staff along with the original manuscript. The proof copy is read against the orginal manuscript, and corrections are marked on the proof copy. The paper is then given back to the same keyboarder who originally keyed the paper for corrections. Payment at this stage is a small, fixed amount per page. When these corrections have been made a set of proof copy is sent to the author for approval.

In the case of an electronic manuscript, after it is logged into the tracking database the file is given to a keyboarder who changes the documentstyle being used from a generic style to the appropriate publication style. A utility is run from within the keyboarder's editor which searches the file for certain unwanted typesetting codes such as those dealing with page and line breaks, vertical and horizontal space, magnification, `\hsize` and `\vsize`, and control characters. This utility produces a report at the top of the data file listing what it located, and the keyboarder searches the file for these items and alters them as needed. The keyboarder must also check any local definitions made by the author to make sure they don't conflict with AMS definitions.

At this point proof copy is generated and sent to the editorial staff, where it is copyedited only to the extent necessary to mark for change anything that does not conform to AMS style. After these corrections are made a proof copy is sent to the author for approval.

An electronic submission can be rejected if the author has not used AMS-TEX or AMS-LATEX, or if they did use one of these packages but did not follow the suggestions in the Guidelines. The overriding consideration in making this decision is whether it would be quicker to alter the author's file to make it compatible with the production system, or to re-key the author's paper. A short paper (5–10 pages) submitted in plain TEX without a great amount of mathematics will invariably be quicker to re-key.

At the point in which a paper is sent to the author, the production cycle becomes identical for papers originating either as electronic submissions or as paper manuscripts. After the author has approved the proof copy and perhaps requested some alterations, corrections are made and proof is again returned to the editorial staff. At this stage the editorial staff will combine individual papers to comprise an issue of a book or journal, and from this point onward these papers will be processed as a unit.

The issue is now paged, and specific formatting instructions such as line and page breaks are inserted. No line and page breaks are allowed in any correction runs prior to this stage, since the author may request alterations which would affect paging. These specific formatting instructions are held to a minimum, however, and the camera copy is sometimes altered by hand in preference to additional TEX runs.

4.2 Author-Prepared Books

The AMS also chooses to publish certain books directly from camera copy supplied by the author as a cost saving device, and to speed publication. The popularity of TEX among academic and other research centers has allowed for the production of these low-cost books with little sacrifice in quality.

Aside from the generic documentstyles described above, the Society also distributes AMS-TEX and AMS-LATEX documentstyles for a few specific book series which publish author-prepared copy. Authors are encouraged to use these documentstyles in the preparation of their work, and submission of both the TEX source file and the DVI file is requested. The DVI file is simply spooled to a typesetter, and the TEX source file is archived for possible future use. If the author has followed the suggestions in the Guidelines and used the package correctly, the final result of books published in this fashion will have the appearance of a comparable book produced via the traditional route, with the exception that the author-prepared work will be published in Computer Modern fonts while all other AMS publications use Times Roman.

5 Future Directions

5.1 Production System Enhancements

Several major enhancements to our current production are either under way or being planned for the next few years. The TEX macros which comprise the current system have been altered many times by many different people over the past decade. A study last year revealed that somewhere around 50% of the code in the current system was either obsolete or never executed. A rewrite of the system is currently underway.

One major difficulty that the rewrite will address is the ability to easily produce a set of typographic specifications for each publication. The Editorial Department has a set of written specifications for each publication, but it is a difficult task to ensure that the TEX macros which produce the publication exactly match the written specifications. Work is being done to have the TEX macros actually read a file containing the typographic specifications for each publication.

A prototype of this file is being developed. Every logical component of a book or journal article is listed in this file, along with such attributes as the font to use, the amount of space (maximum, minimum, and optimal) before and after the element, penalties for page breaks before and after, other attributes such as linespacing, wordspacing, case, justification, etc., are listed in the file in a format that is both human readable and can be parsed by TeX.

Another problem area being worked on is the ability to change a documentstyle for a given publication without affecting papers for that publication which are already in the production stream. Changes to publications are sometimes requested to begin with a specific issues (such as the first issue of a new year), but when papers are accepted for many journals, they are not accepted for a specific issue but instead go into a pool for that journal. Therefore, it is often the case that papers for several different issues of a journal are already in process at various points along the production stream at any given time.

The idea here is to embed a date into the data file which TeX will read. In addition, whenever changes are made to any macro files in the production system, a copy of the code is first copied to the bottom of the macro file before any changes are made. The unaltered code is marked with the date the revisions are being made to the file, and then changes to this code are made in the main section of the file. When a data file is processed, TeX reads the date in the data file and then reads the macro file(s). After the main section of the macro files are read, TeX checks the dates on the altered code at the bottom of the macro file, and stops reading when the date in the macro file is less than the date in the data file.

For example, supposed `\heading` is to be changed from boldface to italic type, and suppose there are two data files, one with a date of 1-MAY-1992 and the other with 1-JUN-1992. Suppose also that the change to `\heading` was made on 15-MAY-1992. The desired result is that the data file keyed on 1-MAY-1992 should retain boldface headings, while the 1-JUN-1992 should pick up the new italic headings. The diagram below shows how the above scheme accomplishes this.

macro file: `\def\heading#1{\noindent\bf#1}`
(additional code)

`\dateofchange{15-MAY-1992}`
`\def\heading#1{\noindent\it#1}`

data file #1: `\filedate{1-MAY-1992}`
ITALIC Headings
data file #2: `\filedate{1-JUN-1992}`
BOLD Headings

5.2 Changing Fonts: CM, Autologic Times Roman, PostScript Times Roman

The STI system used during the late 1970s and into the early 1980s typeset the AMS publications using Times Roman fonts. For a period of several years during the mid 1980s some AMS journals were published which contained articles typeset with both STI and TeX in the same issue. The enthusiastic move toward a TeX-based production system was slightly diminished by the lack of TeX-compatible Times Roman fonts at that time. Many people involved in the publications process felt that Times Roman had become an industry standard for technical publications for its readability and appearance, and they were somewhat reluctant to publish using TeX's Computer Modern fonts.

In 1987 the first Autologic APS-μ5 was purchased, and work was immediately begun to create a set of TeX-compatible Autologic Times Roman fonts. Accessing the Autologic Times Roman, Italic, and Bold fonts were fairly straightforward, but the creation of an Autologic Times Math Italic font for use with TeX was a major effort. The details of this work were presented at tenth annual TeX User's Group meeting at Stanford University in August of 1989. A published version of this presentation, entitled *Migration from Computer Modern Fonts to Times Fonts* by R. E. Youngen, W. B. Woolf, and D. C. Latterner can be found in TUGboat, volume 10, number 4.

At the time when the APS was purchased, high-resolution PostScript typesetters were available in the marketplace, but they were all much too slow for the volume of production at the AMS. However, since that time the hardware has improved dramatically, and PostScript has assumed its place as an industry standard. The Society is now preparing to move book and journal production to PostScript fonts over the next few years.

Switching production to PostScript Times fonts is contingent on a PostScript Times Math Italic font, equivalent to the Autologic version in current use. The *MathTime*™ fonts recently released by Michael Spivak contain such a font, and plans call for this to be evaluated along with the possibility of the AMS creating its own such font.

The move to PostScript is driven not only by its status as an industry standard, but also for the promise of such things as auto-inclusion of PostScript graphics into TeX documents and the ability to output completed pages directly to film negatives or plate materials. Authors' graphics are currently re-rendered on a Macintosh using Adobe Illustrator, which is a PostScript-based application. These graphics are currently being output to the Society's PostScript imagesetter and stripped onto the camera copy of the TeX document by hand.

By typesetting completed pages, substantial cost savings may be realized if typesetting directly on plate materials proves practical. The current cycle is to produce camera copy from a typesetter which is taken to the print shop and photographed to produce a negative. The negative is then used to make a metal plate that can be placed on the press. Typesetting directly to plate material would save in both labor and materials costs for the intermediate steps.

5.3 Electronic Dissemination/Publication

Dissemination of information by electronic means is not a new concept to the AMS. Since 1982, *Mathematical Reviews (MR)* has been distributed electronically as well as on paper, with the data being installed in the public databases of vendors such as Dialog and ESA. Prior to 1985, MR was typeset using the STI system, and the data had to be converted into a human-readable form for database installation. Effective with the January 1985 issue, TEX became the sole input language. Since then, a set of macros has been developed to permit database users to download items from the databases to their local computer, process it with TEX, and print out the result, which is for all practical purposes identical to the published review. In 1989 MR was also released on CD-ROM which is accessible using search software on PC-compatible and Macintosh computers. Full reviews from 1980 to the present are now available on CD-ROM, along with bibliographic information from 1940–1979.

Beginning this year, the *Bulletin of the American Mathematical Society* is also being distributed electronically via the Society's e-MATH node (e-MATH.ams.com) on the Internet. The *Bulletin* is a free journal as a privilege of membership and AMS-TEX files for this journal are installed on e-MATH for public access. Files for this journal are prepared in the same way as for any other journal, and a paper copy is still produced. Experience with this service will help determine the direction of other similar experiments.

5.4 Standard Generalized Markup Language (SGML)

The AMS has already concluded, independent of the SGML initiative, that generic markup is the only sensible way to approach electronic storage and processing of manuscripts and bibliographic data. Though strict SGML syntax is not used in the production of AMS publications, the underlying principles of SGML are in practice.

The Ann Arbor office is developing an SGML-based software tool which will perform several important document-handling functions, such as:

- accept an AMS-LATEX file from an author, parse it into an SGML document, and store it in a document database which recognizes and manages the structural parts of the document;
- accept an SGML file and produce an AMS-LATEX file which can be typeset through TEX;
- allow several people (cooperating authors, editors, etc.) to work together (from remote sites) on the creation of a document, maintaining *version control*, which will allow authors, editors, etc., to track the modifications each has made to the file;
- allow referees, editors, reviewers, and readers to make comments about the file, which will be stored as *annotations* attached to specific points or regions in the document, with identification of the commentator and a record of the time of annotation.

The expectation is that this tool will someday allow for on-line editing in the production of the AMS publications, replacing the cumbersome process of creating paper galleys for each correction run. Instead of corrections being marked on paper with colored pencils by an editor, then keyboarded by a typist, and finally re-proofread by an editor, the editor will be able to make corrections directly in the file and see the results instantly on the screen.

Perhaps even more importantly, this tool will define an environment for true collaborative development of electronic and paper publications by authors situated at remote sites throughout the world. *Version control* and *annotation* are the key capabilities required for a tool appropriate to collaborative research and writing in mathematics.

6 Conclusion

It is impossible to predict to what extent electronic publishing will replace traditional publishing on paper. While the advantages of electronically stored and produced documents are many and conclusive, the affinity for paper publications may be difficult to overcome. As computing devices become smaller and more portable and are linked by radio networks, the difference between sitting beside the fireplace with a book you pulled off the shelf and a book reader which electronically retrieved your favorite text becomes less obvious. TEX and SGML, or some offshoots thereof, are well positioned to play an important role in this future.

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.

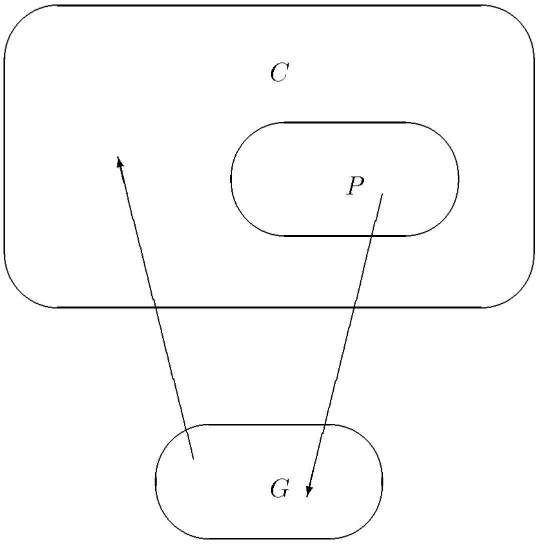
‡CERN, 1211-CH Geneva 23, Switzerland.

§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 → a </sub> f(x)`
- `<plex><operator>lim</operator><from>x ↓ a</from> <of>f(x)</of></plex>`
- `<mfn name=lim><sub pos=mid>x → 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 x² 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

<table>
<tr><th>ISO</th><th>AAP equivalent</th><th>Difference</th></tr>
<tr><td colspan="3">Horizontal and vertical alignment</td></tr>
<tr><td><code>mark id=</code></td><td><code>hmk id=</code></td><td>identical</td></tr>
<tr><td><code>markref refid=</code></td><td><code>hmkr rid=</code></td><td></td></tr>
<tr><td></td><td><code>vmk id=</code></td><td>vertical alignment</td></tr>
<tr><td></td><td><code>vmkr rid=</code></td><td>is absent in ISO</td></tr>
<tr><td colspan="3">Division points in a formula</td></tr>
<tr><td><code>break type=</code></td><td><code>tu</code></td><td>Both elements are empty. <code>type=</code> indicates optionality. <code>tu</code> is excluded from various constructs.</td></tr>
<tr><td colspan="3">Superscripts and subscripts</td></tr>
<tr><td><code>sub</code></td><td><code>inf</code></td><td></td></tr>
<tr><td><code>sub pos=(pre|mid|post)</code></td><td><code>inf loc=(pre|post)</code></td><td>mid value (ISO) missing in AAP</td></tr>
<tr><td><code>sup</code></td><td><code>sup</code></td><td></td></tr>
<tr><td><code>sup pos=(pre|mid|post)</code></td><td><code>sup pos=(pre|post)</code></td><td>mid value (ISO) missing in AAP</td></tr>
<tr><td colspan="3">Boxes</td></tr>
<tr><td><code>box</code></td><td><code>box style=</code></td><td>style atribute (AAP) to to change rule type missing in ISO</td></tr>
<tr><td colspan="3">Over embellishments</td></tr>
<tr><td><code>ov</code></td><td><code>a</code></td><td></td></tr>
<tr><td><code>ov pos=</code></td><td><code>a valign=</code></td><td></td></tr>
<tr><td><code>ov type=</code></td><td><code>ac</code></td><td><code>ac</code> element (AAP) allows simultaneous embellishments</td></tr>
<tr><td><code>ov style=</code></td><td><code>ac</code></td><td><code>style</code> attribute (ISO) can be achieved with <code>ac</code></td></tr>
<tr><td colspan="3">Tensors</td></tr>
<tr><td><code>tensor</code></td><td></td><td>the tensor element(ISO)</td></tr>
<tr><td><code>tensor suffix=</code></td><td></td><td>is absent in AAP.</td></tr>
<tr><td><code>tensor posf=</code></td><td></td><td>can be visually achieved with <code>sup</code>, <code>inf</code> and <code>zw</code> (zero width character, absent in ISO)</td></tr>
<tr><td colspan="3">Functions</td></tr>
<tr><td><code>mfn name=</code></td><td><code>rf</code></td><td>name attribute (ISO) has a finite list of values.</td></tr>
<tr><td><code>fname</code></td><td></td><td><code>fname</code> (ISO) indicates</td></tr>
<tr><td><code>of</code></td><td></td><td>arbitrary roman function; of the argument; <code>rf</code> (AAP) only marks up the function name</td></tr>
<tr><td colspan="3">Roman and italic fonts</td></tr>
<tr><td><code>roman</code></td><td><code>rm</code></td><td>equivalent</td></tr>
<tr><td><code>italic</code></td><td><code>it</code></td><td></td></tr>
<tr><td colspan="3">Vectors</td></tr>
<tr><td><code>vec</code></td><td><code>v</code></td><td>equivalent</td></tr>
<tr><td colspan="3">Fractions</td></tr>
<tr><td><code>frac</code></td><td><code>fr</code></td><td>equivalent</td></tr>
</table>

<table>
<tr><td>numer</td><td>nu</td><td></td></tr>
<tr><td>over</td><td>de</td><td></td></tr>
<tr><td>frac align=(left|right|center)</td><td>fr align=(r|l|c)</td><td>equivalent</td></tr>
<tr><td></td><td></td><td>AAP has shape attribute, fr shape=(case|built|sol), to indicate shape of fraction; missing in ISO</td></tr>
<tr><td colspan="3">Derivatives</td></tr>
<tr><td>diff</td><td>inc</td><td>The inc (AAP) element</td></tr>
<tr><td>diffof</td><td></td><td>is for increments. The type</td></tr>
<tr><td>by</td><td></td><td>is given in the content,</td></tr>
<tr><td>diff type=partial</td><td></td><td>but it is not marked up</td></tr>
<tr><td colspan="3">General plex (limits)</td></tr>
<tr><td>plex</td><td>lim</td><td>equivalent</td></tr>
<tr><td>operator</td><td>op</td><td></td></tr>
<tr><td>from</td><td>ll</td><td></td></tr>
<tr><td>to</td><td>up</td><td></td></tr>
<tr><td>of</td><td>opd</td><td></td></tr>
<tr><td>sum</td><td>sum</td><td>equivalent</td></tr>
<tr><td>integral</td><td>in</td><td></td></tr>
<tr><td>product</td><td>pr</td><td></td></tr>
<tr><td colspan="3">Piles</td></tr>
<tr><td>pile</td><td>stk</td><td>equivalent</td></tr>
<tr><td>above1</td><td>lyr</td><td>the role of above1 (ISO) is not clear</td></tr>
<tr><td>above</td><td></td><td></td></tr>
<tr><td>pile align=(left|right|center)</td><td>align=(l|r|c)</td><td>equivalent</td></tr>
<tr><td colspan="3">Matrices</td></tr>
<tr><td>matrix</td><td>ar</td><td>ISO marks up rows</td></tr>
<tr><td>col</td><td>arc</td><td>inside columns via</td></tr>
<tr><td>above</td><td>arr</td><td>above. AAP has elements</td></tr>
<tr><td></td><td></td><td>for both.</td></tr>
<tr><td></td><td>ar cs</td><td>AAP allows various</td></tr>
<tr><td></td><td>ar rs</td><td>column and row</td></tr>
<tr><td></td><td></td><td>separators.</td></tr>
<tr><td></td><td>ar ca</td><td>AAP allows overall</td></tr>
<tr><td></td><td></td><td>column alignment</td></tr>
<tr><td>col align=(left|right|center)</td><td>arc align=(l|r|c|d|e)</td><td>d and e for alignment on exponents and decimal points</td></tr>
<tr><td colspan="3">Square root and square</td></tr>
<tr><td>sqrt</td><td>rad, rdx 2</td><td>no separate elements</td></tr>
<tr><td>square</td><td>sup 2</td><td>for square roots and squares in AAP</td></tr>
<tr><td colspan="3">Root and power</td></tr>
<tr><td>root</td><td>rad</td><td>The content of root</td></tr>
<tr><td>degree</td><td>radix</td><td>is the content of rad</td></tr>
<tr><td>of</td><td></td><td>in AAP;</td></tr>
<tr><td></td><td></td><td>of element not required.</td></tr>
</table>

`power` `degree` `of`	`sup`	No general power element in AAP
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.