EXTRAFOR for Math on the Web

Afef KACEM
UTIC, Tunis College of Sciences and Techniques, TUNISIA
Afef.kacem@esstt.rnu.tn

ABSTRACT

It is becoming increasingly important to find ways of communicating mathematics which facilitate automatic processing, searching and indexing, and reuse in other mathematical applications and contexts. With this advance in communication technology, there is an opportunity to expand our ability to represent, encode, and ultimately to communicate our mathematical insights and understanding with each other. At the same time, the web lacks an effective means for embedding mathematics in Web pages. Authors have been forced to use bitmapped images, which are slow to load, complex to produce, and look bad when printed. In this paper, we propose to use a mathematics extraction and recognition system, labelled EXTRAFOR, to provide an easy to add, use, learn and communicate Math on the Web. With EXTRAFOR, we are not only able to extract mathematical formulas automatically from digitally scanned image of a previously typeset documents but also to read, parse and re-use them in other applications and contexts. This can be done for a variety of purposes. An example of small-scale use is a reading machine for the visually impaired. Large scale use arises in the scanning and interpretation of a large collection of mathematical documents, for the creation of a database. Here, we investigate the use of such system for putting and communicating mathematics over the Web.

Keywords: Extraction, recognition, mathematics, formula, symbol, MathML, XML.

1. Introduction

Since its inception, the Web has demonstrated itself to be a very effective method of making information available to widely separated groups of individuals. However, even though the Web was initially conceived and implemented by scientists for scientists, the capability to include mathematical formulas in HTML is very limited. At present, most mathematics on the Web consists of text with GIF images of scientific notation, which are difficult to read and to author.

At the same time, many old documents in science and engineering disciplines contain mathematical formulas. But this material is not available in electronic form. Other newer sources are publications, articles, filled of useful information which is often difficult to get sources. Actually, the only way to use this mathematical information is to re-type formulas on keyboard to be able to add it in computer algebra system or in any application using mathematical input. The input of mathematical formulas into computers is often more difficult than the input of plain text, because mathematical formulas typically consist of special symbols and Greek letters in addition to English letters and digits.

Our aim is to start from digitally scanned images of documents containing formulas, to extract and recognize them to be able to re-use them in other applications and contexts. This paper is also aimed at authors of Web contents who wish including mathematical formulas, as well as persons who want to read these contents. The education community is a large and important group that must be able to put scientific curriculum materials on the Web. But, they often have limited resources of time and equipment, and are severely hampered by the difficulty of authoring technical Web documents. Students and teachers need to be able to create mathematical content quickly and easily, using intuitive, easy-to-learn, low-cost tools. This paper presents EXTRAFOR as a tool that eases the use of mathematical and scientific content on the web.

2. Problems in mathematics publishing and communicating

How hard can it be to communicate about math on the Internet? The truth is, it's a fairly difficult task. The World Wide Web Consortium (W3C) has been developing a Mathematical Mark-up Language (MathML) since 1998 [1]. In the meantime, there is no standard method for representing math notation on the Web, and there are many, many choices for doing so, each with its own advantages and drawbacks. While there are fewer methods for communicating about math via email, it's not much easier to work with. The W3C recognized that lack of support for scientific communication was a serious problem. This paragraph is an attempt to clarify the issues around effectively displaying and communicating mathematics on the Internet. The most obvious problems with HTML for mathematics are of two kinds: display and encoding of formulas. Let's consider this formula:

$$\frac{d}{dx}f(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

It was laid out with a math typesetting program, converted to a transparent-background GIF, and placed on a web page with an image tag. Anyone who views your formula on the web will see it with the font, type size, and spacing exactly as you've specified. Even if it looks fine on your own browser, someone may be viewing the page with a font/type size that clashes with what you've chosen for your GIF. A second point to observe that small images embedded in a line of text (like this: $\frac{\sqrt{2}}{2}$) can throw off the line-spacing of a paragraph. In addition, image-based formulas are generally harder to see, read and comprehend than the surrounding text in the browser window. Moreover, these problems become worse when the document is printed.

Some display problems associated with including math notation in HTML documents as images could be addressed by improving browser image handling. However, even if image handling were improved, the problem of making the information contained in mathematical expressions available to other applications would remain. Consider trying to search this page for part of the formula above, for example, the "**lim**". In a similar vein, consider trying to cut and paste a formula into another application; even more demanding is to cut and paste a sub-formula. Using image based methods; neither of these common needs can be adequately addressed. Another problem with encoding mathematics as images is that it requires more bandwidth.

Many people are working on ways to make mathematical notation easier to display on the web. None of these methods is without fault; most are in beta version, and most rely on the user (or web page author) having particular software and/or technical expertise. By using markup-based encoding, more of the rendering process is moved to the client machine. Markup describing a formula is typically smaller and more compressible than an image of the formula. Therefore, in planning for the future, it is not sufficient to merely upgrade image-based methods. To fully integrate mathematical material into Web documents, a markup-based encoding of mathematical notation and content is required.

The goal of this study is to bring the power of mathematical formula to as many people as possible, and as quickly and easily as possible. The proposed medium is a Math Mark-up language called MathML designed for this purpose. But a language alone is not enough: with this design comes a deployed implementation that immediately makes formulas a reality for the scientific and mathematical Internet community. With EXTRAFOR, it's possible to reuse or insert mathematical formulas directly into a document and have them correctly displayed.

3. Automatic segmentation and recognition of mathematical documents

EXTRAFOR (automatic EXTRAction of mathematical FORmulas) was firstly conceived to extract mathematical formulas, outside and inside text-lines, automatically from images of printed documents without optical character recognition [2-4]. Then, it was extended to analyse, recognize and encode formulas using MATHML language. An overview of EXTRAFOR system is shown in Fig. 1.

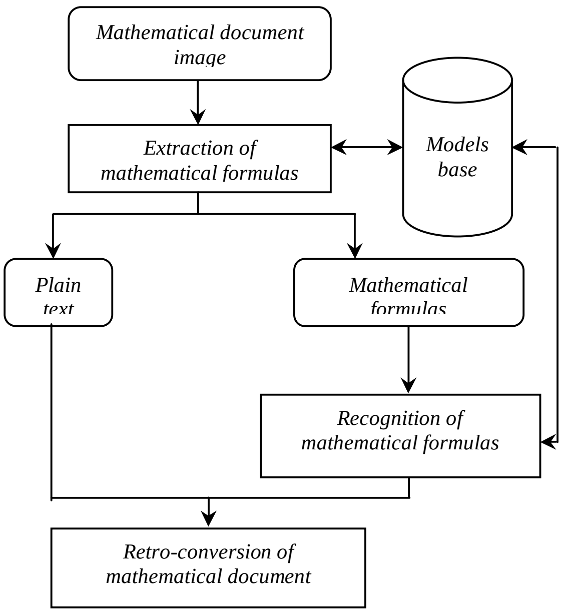

Mathematical document image
Extraction of mathematical formulas
Models base
Plain text
Mathematical formulas
Recognition of mathematical formulas
Retro-conversion of mathematical document

Fig. 1: EXTRAFOR system overview

Mathematical formulas are typically embedded in text documents, either as offset formulas, or embedded directly into a line text. Thus, the first step in mathematics recognition is to identify where formulas are located on the page. We will bravely explain how EXTRAFOR works to find the mathematical formulas in the document. Then, we will mainly focus on the formula recognition steps especially on the structural analysis aspect which is particularly difficult for mathematics, due to the subtle use of space in this notation. We finally give an illustrative example of the retro-conversion XML of a mathematical document where formulas are encoded using MathML.

3.1. Extraction of mathematicalformulas

EXTRAFOR extracts mathematical formulas automatically from images of printed documents in order to mask them out in the optical character recognition (OCR) process, while on the other hand being able to analyse and recognize their content. The main goal is to have an OCR free system for the separation of text versus mathematics; hence it is mostly based on reasoning on bounding boxes of formulas components.

As mathematical formulas are represented with various objects: alphabetic characters, numerals, math operators such as +, *, -, ∑, ∏, ∫, (, [, and so on, the extraction of those objects is the first step to locate mathematical formulas. EXTRAFOR must tentatively identify many of the connected components as particular characters. Characters that are known to be mathematical are used as tokens for formula extraction. EXTRAFOR first extracts a set of connected compounds using some attributes like ratio, area and density. It assigns then a label to each of them according the role played in formula composition. This primary labelling serves to identify some mathematical symbols by the means of models created at a learning step using fuzzy logic. This labelling allows a global segmentation of the document which aims to discard isolated formulas from plain text. Text lines are labeled as isolated formulas based both on internal properties and on having increased high and similar white space on their left and right. The remaining text-lines consist of a mixture of pure text and text with embedded formulas. For embedded formulas, a local segmentation of text-lines is required. It is done by location of their most significant symbols then extension to adjoining symbols using contextual rules until delimitation of whole formulas spaces.

Though a satisfactory rate of extraction is obtained, more research is still required to be able to attain human-like performance.

Further work is required to extend this method to low quality documents with broken or touching characters. In fact, for low-resolution, noisy, or poorly scanned images, this processing may be not so efficient. Old papers may also do not scan well even at high resolution.

3.2. Recognition of mathematical formulas

Recognition of mathematical formulae has been a widely studied problem. In a mathematical formula, characters and symbols can be spatially arranged as a complex two-dimensional structure, possibly of different character and symbol sizes. This makes the recognition process more complicated even when all the individual characters and symbols can be recognized correctly. There are many contributions concerning this topic – several methods have been designed or adopted for the formulae recognition, taxonomy can be found in [5]. Most of the known methods follow two main stages: symbol recognition and symbol-arrangement analysis. The former converts the input image into a set of symbols. The latter analyzes the spatial arrangement of this set of symbols to recover the information content of the given mathematical formula. Character recognition has been an active research area for more than three decades. Optical character recognition systems for mathematical documents which contain not only ordinary texts but also mathematical expressions have been investigated [6]. Structural analysis of two dimensional patterns also has a long history.

3.2.1. Recognition of mathematical symbols

Symbol recognition is difficult because there is a large character set (roman letters, Greek letters, operators, symbols…) with a variety of typefaces (normal, bold, italic), and a range of font sizes (subscripts, superscripts, limit expressions…). Certain symbols have an enormous range of possible scales (brackets, parentheses, $\int$, Π, Σ…). To be able to recognize mathematical symbols, EXTRAFOR must first group them properly into units. This can be done by using some conventions in writing mathematical formulas as heuristics. Some of these conventions are as follows:

- Some pieces multi-part characters or symbols should be joined together by vertical grouping to form symbols like '=' and attaching the dot over the "i" or the "j" to their body.
- Some letters together form a unit, like trigonometric function names such as *tan*, *sin* and *cos*. Before considering a group of letters as a concatenation of variables, we have to first check whether they are in fact some predefined function names.
- Symbols other than letters and digits should be considered as separate unit.

Because of failure of OCR systems when treating with mathematical document, EXTRAFOR supply an automatic symbol-recognizer. It works in conjunction with an OCR system by applying this latter only on characters and reusing the results obtained from the labelling step.

3.2.2. Structural analysis of mathematical formulas

Formulas contain significant structural information which is expressed as 2D spatial relations. Thus, mathematical formulae recognition is an appropriate application area for testing 2D grammars-based approach. EXTRAFOR is based on material on [7] and includes complete grammars for the recognition of commonly used arithmetic notation. It uses a top-down parsing scheme which starts with the ultimate syntactic goal and the entire set of input and attempts to partition the problem into sub goals until either every sub goal is reached, or else all possibilities have failed. A syntax rule provides instructions for the partitioning of a symbol set into subsets, and assigns a syntactic goal to each of these subsets. EXTRAFOR starts the parsing by locating of the most important operator in the formula and attempts to partition it into sub formulas which are similarly analyzed by looking

for a starting point. The choice of the starting point is function of its precedence and dominance in the formula. When consecutive operators exist in a formula, we apply operator precedence rules. However, when those operators are not lined up, we have to use the concept of operator dominance. For example, in «$a+\frac{b}{c}$ », the meaning is « a+(b/c) » due to the fact that the operator '+' dominates '/' (where '/' lies in the range of '+'). However, in « $\frac{b+c}{d}$ », the meaning becomes « (a+b)/c » since '/' dominates '+' (where '+' lies in the range of '/') in this case.
Table 1 gives the input characters and symbols needed for mathematical formula syntaxes and the syntactic category which each of them is given by the pre-processing.

Character or mathematical symbol	**Syntactic category**
a, b, …, A, B,…, Z, α, β, …, ζ,∞	Letter
0, 1, …, 9	Digit
23, 1089, 0, 56,….	Unsigned_integer
12.34, 8.709,…	Unsigned_float
+, -, $\pm$, *, ., /, =, …$\sum$, $\prod$, $\int$, $\sqrt{\ }$,—, (,), [,], {, }, |, $\left(\vphantom{\int}\ ,\ \right)$, $\left[\vphantom{\int}\ ,\ \right]$, $\left\{\vphantom{\int}\ ,\ \right\}$, $\left\langle\vphantom{\int}\ ,\ \right\rangle$, $\left\lceil\vphantom{\int}\ ,\ \right\rceil$, $\left\lfloor\vphantom{\int}\ ,\ \right\rfloor$,…	Syntactic category represented by the symbol itself
sin, cos, tan	Trigonometric_function
Det	Determinant

Table 1: Input objects and their syntactic categories

Although mathematics is a relatively standardized notation, considerable variation is permitted in relative symbol placement. In light of this variability, it is not clear how to define and identify meaningful spatial relationships among symbols in a formula. Spatial relationships are especially critical for the recognition of implicit mathematical operators (subscripts, superscripts, implied multiplication…)[7]. Geometric criteria are used here to check if the symbols of a formula have possible links between them. It is about to subdivide plan into seven regions around the symbol: A (above), D(down), L (left), R(right), S (superscript), s(subscript) and I (included). We give, in table 2, some production rules, used for formula recognition and MathML encoding.

Production rules	**Spatial relations hip**	**MathML Encoding**
R_1 : E **à** E + E	E_1=L(2) and E_2=R(2)	E_0.code=<mrow> E_1.code <mo>+</mo> E_2.code </mrow>
R_2 : E **à** E – E	E_1=L(2) and E_2=R(2)	E_0.code=<mrow> E_1.code <mo> - </mo> E_2.code </mrow>
R_3 : E **à** E ± E	E_1=L(2) and E_2=R(2)	E_0.code=<mrow>E_1.code<mo>&plusminus;</mo>E_2.code</mrow >
R_4 : E **à** E * E	E_1=L(2) and E_2=R(2)	E_0.code=<mrow>E_1.code<mo*</mo>E_2.code</mrow>
R_5 : E **à** E / E	E_1=L(2) and E_2=R(2)	E_0.code=<mrow>E_1.code<mo>/</mo>E_2.code</mrow>
R_6 : E **à** E . E	E_1=L(2) and E_2=R(2)	E_0.code=<mrow>E_1.code<mo>.</mo>E_2.code</mrow>
R_7 : E **à** E E	E_1=L(2) and E_2=R(1)	E_0.code=<mrow>E_1.code<mo>&invisibletimes;</mo>E_2.code</m row>
R_8 : E **à** + T	T=R(1)	E.code=<mrow> <mo>+</mo> T.code </mrow>
R_9 : E **à** - T	T=R(1)	E.code= <mrow> <mo> - </mo> T.code </mrow>
R_{10} : E **à** T		E.code=T.code
R_{11} : E **à** $\int_L^L E\, \boldsymbol{d}\, V$	L_1= (D(1) or s(1)), L_2= (A(1) or S(1)), E_1= R(1) and V=R(5)	**if** (L_1≠ε et L_2≠ε) **then** E_0.code = <munsubsup> <mo>&int;</mo>L_1.code L_2.code</munsubsup> E_1.code <mo>DifferentialD ;</mo> V.code **else** E_0.code =<mo>&int;</mo> E_1.code <mo>DifferentialD ;</mo> V.code
R_{12} : E **à** $\sum_S^L E$	S=(D(1) or s(1)), L=((A(1) or S(1)) and E_1=R(1)	**if** (L≠ε) **then** E_0.code = <munderover><mo>&sum ;</mo> S.code L.code </munderover>E_1.code **else** E_0.code = <munder><mo>&sum ;</mo> S.code </munder>E_1.code

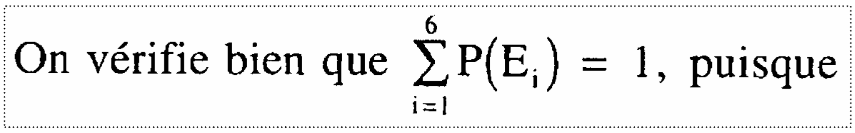
R_{13} : E **à** $\prod_{S}^{L} E$	S=(D(1) or s(1)), L=((A(1) or S(1)) and E_1=R(1)	**if** (L≠ε) **then** E_0.code = ***<munderover><mo>&prod;</mo>***S.code L.code ***</munderover>***E_1.code **else** E_0.code = ***<munder><mo>&prod;</mo>*** S.code ***</munder>***E_1.code
R_{14} : E **à** $\frac{E}{E}$	E_1=A(2) and E_2=D(2)	E_0.code = ***<mfrac>***E_1.code E_2.code ***</mfrac>***
R_{15} : E **à** $\sqrt[R]{E}$	E_1=I(1) and R=S(1)	**if** (R≠ε) **then** E_0.code= ***<mroot>***E_1.code R.code***</mroot>*** **sinon** E_0.code=***<sqrt>*** E_1.code ***</sqrt>***
R_{16} : E **à** | E |	E_1=(R(1) and L(3))	E_0.code=***<mrow><mfenced open=”|” close=”|”>***E_1.code ***</mfenced></mrow>***
R_{17} : E **à** (E)	E_1=(R(1) and L(3))	E_0.code= ***<mrow><mfenced>*** E_1.code ***</mfenced></mrow>***
R_{18} : E **à** E_{D}^{X}	D=s(1) and X=S(1,3)	**if** (D≠ε et X≠ε) **then** E_0.code = ***<msubsup>***E_1.code D.code X.code***</msubsup>*** **else if (**X=ε) **then** E_0.code =***<msub>***E_1.code D.code***</msub>*** **else** E_0.code =***<msup>*** E_1.code X.code***</msup>***
R_{19} : E **à** ***det*** E	E_1=R(1)	E_0.code=***<mrow> <mo>det</mo>*** E_1.code***</mrow>***
R_{20} : E **à** ***trig*** ^{X}E	X=S(1) and E_1=R(1)	**if**(X≠ε) **then** E_0.code= ***<msup> <&Applyfunction ;> <mo>trig.val</mo>*** X.code ***</msup>*** E_1.code **else** E_0.code=***<&Applyfunction ;> <mo>trig.val</mo>*** E_1.code
R_{21} :T **à** V		T.code = V.code
R_{22} :T **à** ***unsigned_float***		T.code = ***<mn>Unisgned_float.val</mn>***
R_{23} :T **à** ***unsigned _integer***		T.code = ***<mn>Unsigned_integer.val</mn>***
R_{24} : L **à** E		L.code = E.code
R_{25} : L **à** +¥		L.code = ***<mrow><mo>+</mo><mo>&infin;</mo></mrow>***
R_{26} : L **à** -¥		L.code = ***<mrow><mo>-</mo><mo>&infin;</mo></mrow>***
R_{27} : L **à** ¥		L.code = ***<mo>&infin;</mo>***
R_{28} : L **à** *e*		
R_{29} : S **à** V = E	V=L(2) and E=R(2)	S.code = ***<mrow>***V.code***<mo>=</mo>***E.code***</mrow>***
R_{30} : S **à** V		S.code = V.code
R_{31} : R **à** V		R.code=V.code
R_{32}: R **à** ***unsigned_integer***		R.code=***<mn> unsigned_integer.val</mn>***
R_{33} : R **à** *e*		
R_{34} : X **à** E		X.code=E.code
R_{35} : X **à** *e*		
R_{36} : D **à** D, E	D_1=L(2) and E=R(2)	D_0.code= ***<mrow>***D_1.code ***<mo>,</mo>*** E.code***</mrow>***
R_{37} : D **à** E		D.code=E.code
R_{38} : V **à** ***letter*** $_D$	D=s(1)	V.code=***<msub><mi> letter.val</mi>***D.code***</msub>***
R_{39} : V **à** ***letter***		V.code=***<mi> letter.val</mi>***

Table 2: Rule production and MathML encoding

We plan to deal with more complex formulas and confirm efficiency and performance of our method using a large database of mathematical formulas and documents.

3.3. Retro-conversion of the mathematical document

Fig. 2 shows the result obtained by EXTRAFOR to convert a line of a mathematical document in XML where Formulas are encoded in MathML. Note that whatever is the method used to create a Web page with a contents MathML, once this one exists, all advantages of a layer of powerful general communication appear. A diversity of software MathML can all use the same document to return it in a spoken or printed document, to send it to a system of algebra or to manage it as party of a vast collection of web document. For a high quality printed mathematical returning, the codification MathML will be often reconverted towards languages of standard composition, such as TEX, which is very appreciated for this job.


On vérifie bien que $\sum_{i=1}^{6} P(E_i) = 1$, puisque

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="mathml.xsl"?>
<HTML xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr-FR" xmlns:m = "http://www.w3.org/1998/Math/MathML">
<body>
<p>On vérifie bien que
<math xmlns="http://www.w3.org/1998/Math/MathML">
<math display='block'>
  <semantics>
    <mrow>
      <munderover>
        <mo>&sum;</mo>
        <mrow>
          <mi>i</mi><mo>=</mo><mn>1</mn>
        </mrow>
        <mn>6</mn>
      </munderover>
      <mrow>
            <mi>P</mi><mo stretchy='false'>(</mo><msub>
          <mi>E</mi>
          <mi>i</mi>
        </msub>
        <mo stretchy='false'>)</mo><mo>=</mo><mn>1</mn>
      </mrow>
     </mrow>
    <annotation encoding='MathType-MTEF'>
      </annotation>
  </semantics>
</math>
, puisque
</p>
</body>
</html>
```

Fig. 2: Example of XML converting of a part of mathematical document

4. Conclusion and future works

The image – based methods that are the predominant means of transmitting mathematic notation over the web are primitive and inadequate: document quality poor, authoring and reading are difficult, and mathematical information contained in images is not available for searching, indexing, or reuse in other applications and contexts. To put, use and communicate mathematical documents, math on the web must provide a standard way of publishing and sharing information that can be easily read, processed and generated using commonly available tools. EXTRAFOR is involved with math on the Web at the level of images of print books. In order to meet the diverse needs of the scientific community, EXTRAFOR has been designed with the ultimate goal in mind is to provides an easy to reuse formulas included in mathematical documents and encode mathematical material suitable for teaching and scientific communication.

In this paper, we have demonstrated how EXTRAFOR extract then parse mathematical formulas using a coordinate grammar and encode them in MathML which facilitate automatic processing, searching and indexing, and reuse of mathematical documents. The overall system has shown its efficiency on a number of practical mathematical formulas. But, further work is required to extend this method to more complex formulas and documents and confirm efficiency and performance of our method using a large database.

References

[1] TOPPING, P., « Les mathématiques sur le Web : MathML et MathType», Design Science, Inc. http://www.mathtype.com/, 21 janvier, 1999.

[2] Afef KACEM, "Mathematics Extraction from Images of Scientific Documents", in proceedings of International Conference on Document Analysis and Recognition", Bangalore-India, 20-23 Septembre 1998.

[3] Afef KACEM "A Proposal Syntax-Directed System for Mathematical Document Recognition", in I.J.D.A.R : International Journal on Document Analysis and Recognition, volume 4, Number 2, pp. 97-108, Décembre 2001.

[4] Afef KACEM, "segmentation automatique pour la retro-conversion de documents mathématiques", Ecole nationale des sciences de l'informatique, Tunisia, 2001.

[5] K.-F. Chan and D.-Y. Yeung. Mathematical expression recognition: a survey. In *IJDAR*, volume 3, pages 3–15, 2000.

[6] Masakazu Suzuki, Fumikazu Tamari, Ryoji Fukuda, Seiichi Uchida, (2003). Toshihiro Kanahori. INFTY: an integrated OCR system for mathematical documents. Proceedings of the 2003 ACM symposium on Document engineering table of contents Grenoble, 95/104.

[7] R. H. Anderson, "Two-Dimensional Mathematical Notation", In proceedings of Syntactic Pattern Recognition Applications, K.S. Fu, Ed. Springer Verlag, New York , pp. 147-177, 1977.