ICTA'07, April 12-14, Hammamet, Tunisia

Accessibility to Mathematical Documents

Afef KACEM
Research Unit of Technologies of Information and Communication UTIC
Ecole Supérieure des Sciences et Techniques de Tunis,
5, Av. Taha Hussein, B.P. 56, Bab Mnara 1008, Tunis, TUNISIA
Afef.kacem@esstt.rnu.tn

Abstract

This paper explains how to make scientific information more accessible to wide audience. It describes a practical system for scientific documents including mathematical formulas, named 'EXTRAFOR'. The system recognizes scanned images of clearly printed mathematical documents and outputs the recognition results in XML format which can be converted into various formats: LATEX, MATHML, HTML and Brailles Codes. With EXTRAFOR, we are not only able to extract mathematical formulas automatically from image of 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 to communicate scientific information to people with a visual impairment.

1. Introduction

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. Beside that, advanced scientific documents have historically been inaccessible to the blind. The symbolic system normally used to express mathematical and scientific ideas is more complex than letter after letter, word after word, line after line of regular literature. This leads to difficulty when mathematics is translated into Braille or when mathematics is spoken. The primary goal of this work is to transcribe printed mathematical documents into digital data with accessibility. The idea is to start from digitally scanned images of documents containing formulas, to extract, recognize and convert them into various formats to be able to re-use them in other applications and contexts. This paper describes the outline of EXTRAFOR [1], a prototype designed and implemented for this purpose (see Figure 1).

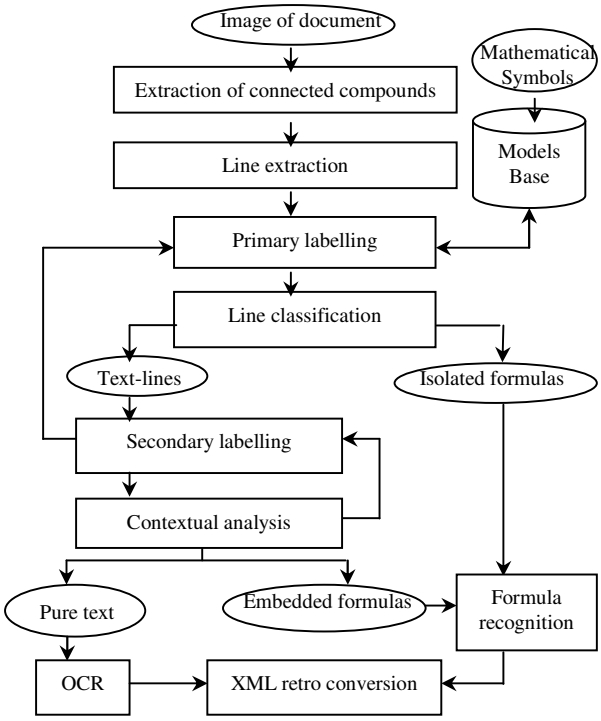


Figure 1: Overview of EXTRAFOR system

2. EXTRAFOR to extract and recognize mathematical formulas

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 XML retro-conversion of a mathematical document where formulas are encoded in MathML [2].

2.1. Extraction of mathematical formulas

Because of the failure of optical character recognition (OCR) systems to recognize mathematical documents, EXTRAFOR extracts formulas automatically from images of printed documents in order to mask them out in the OCR process, while on the other hand being able to analyse and recognize their content. The purpose 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.

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. It first extracts a set of connected compounds. Then using some attributes like ratio, area and density, it assigns 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, on having increased high and on being centered in the page. 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. Some used contextual rules are given here:

- R_1: If a subscript or superscript is found, then it is grouped with its closest neighbour.
- R_2: Each connected compound enclosed inside a radical symbol is considered as part of the formula.
- R_3: If a horizontal fraction bar is found, then the numerators and denominators are joined to it.
- R_4: If a summation, a product or an integral symbol is found, then its lower and upper bounds in addition of the components of its sub expression are joined to it.
- R_5: Each connected compounded enclosed inside a pair of vertical great delimiters should form a formula.
- R_6: when a mathematical symbol is found inside a pair of small delimiters should be part of the formula.
- R_7: If a binary operator (-, =, ≤, ≥, ≅, …) is found, then its left and right operands are joined to it.
- R_8: Two horizontal adjacent formulas constitute one formula
- R_9: Two formulas, separated by a reduced number of connected compounds compose one formula.

From the experiments, we found that the obtained results have demonstrated the applicability of our system since the most of mathematical formulas are extracted from documents printed with high quality (see Figure 2). 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.

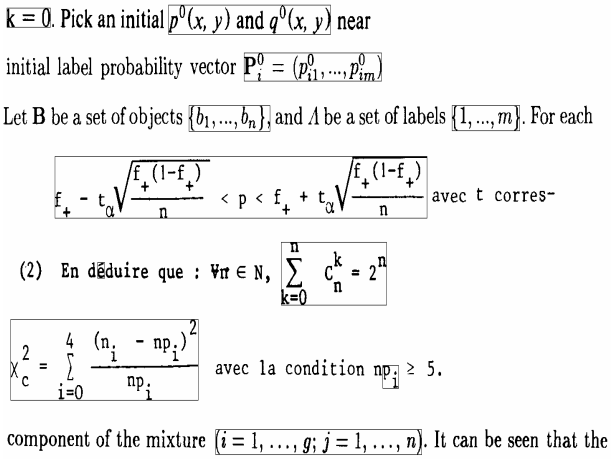


Figure 2: Some obtained results of formula extraction

2.2. Recognition of mathematical formulas

The recognition of mathematical formulas involves 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 the symbols to recover the information content of formula. Character recognition has been an active research area for more than three decades. Structural analysis of two dimensional patterns also has a long history. However, few papers have addressed specific problems related to mathematical formula recognition [3]. In fact, in a

mathematical formula, characters and symbols can be arranged as a complex two-dimensional structure, possibly of different characters and symbol sizes. They differ greatly from text since a line of text is one-dimensional and discrete: characters are placed one after another on the same line when symbols in formulas may be under, above, on the right and far, included in another, etc, with continuous distances. This makes its recognition process more complicated even when all the individual characters and symbols can be recognised correctly.

2.2.1. Recognition of mathematical symbols

The recognition of mathematical symbols 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. Here are some conventions we have used:

- 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 '*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.

EXTRAFOR uses a commercial OCR engine for alpha-numeric characters in usual text area. For mathematical formula area, EXTRAFOR makes recourse to the labelling results.

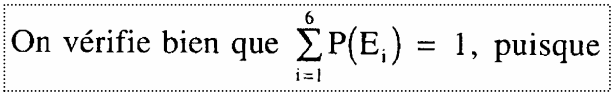
2.2.2. Recognition of mathematical formula structure

As we know, two dimensional mathematics notations is an excellent domain for the application of syntactic pattern recognition techniques. EXTRAFOR is based on material on [4] and includes complete grammars for the recognition of commonly used arithmetic notation. The syntax rules provide instructions for the partitioning of a symbol set into subsets, and assign a syntactic goal to each of these subsets. EXTRAFOR starts the parsing by locating of the main operator in the formula and attempts to partition it into sub formulas which are similarly analyzed. The importance of one operator 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.

As spatial relationships are especially critical for the recognition of formulas using implicit operators like subscripts, superscripts and implied multiplication, geometric criteria are used here to check if the symbols of one formula have possible links between them. It is about to subdivide plan into seven regions around the considered symbol: A (*above*), D (*down*), L (*on the left*), R (*on the right*), S (*superscript*), s (*subscript*) and I (*inside)*. We give, in table 1, some production rules, used for the recognition of the commonly used arithmetic formulas.

2.3. Retro-conversion of mathematical document

EXTRAFOR outputs the recognition results in XML format. The figure 3 presents the results of the XML retro conversion of a line of a mathematical document where the embedded formula is encoded in MathML.




```
<?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>
```

Figure 3: XML and MathML conversion

Note that once the document is converted, 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 document. The figure 4 shows a sample of the web page of a mathematical document, segmented then recognized by EXTRAFOR, as displayed in internet explorer with Mathplayer which is a high-performance MathML display engine. Note that visually impaired users of Internet Explorer use screen reader software packages that speak the words on the page. Many of the most popular windows screen readers, such as Windows-eyes, HAL, read & Write, and JAWS, will work with Mathplayer to speak the math in the page along with the words.

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.

By some converter tools, the XML file could be also converted into various formats of mathematical documents; LATEX, MathML, HTML, and especially for users with visual disability, HR-TEX (Human Readable TEX), KAMS (Karlsruhe ASCII Mathematics Script) and Braille Codes, in UBC (Unified Braille Codes) for English texts. The HR-TEX file is a text file in which mathematical expressions are written in simplified LATEX notations for convenience to be read by users with visual disability, omitting various commands and symbols unnecessary to understand the meaning. KAMS is a notation of mathematics developed in the Study Centre for Blind and Partially Sighted Students at the University of Karlsruhe [5].

4. Conclusion and future works

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 community. With EXTRAFOR, it's possible to extract mathematical formulas, outside and inside text-lines, automatically from images of printed documents without optical character recognition, reuse or insert mathematical formulas directly into a document and have them correctly displayed.

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] Afef KACEM, “segmentation automatique pour la retro-conversion de documents mathématiques”,Thèse de doctorat, Ecole nationale des sciences de l’informatique, Tunisia, 2001.

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

[3] Kam-Fai Chan and Dit-Yan Yeung, « An Efficient Syntactic Approach to Structural Analysis of On-line Handwritten Mathematical Expressions », Rapport Technique HKUST-CS98-10, The Hong Kong University of Science & Technology Department of Computer Science, August 1998.

[4] 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.

[5] M. Suzuki & al, “Integrated OCR software for mathematical documents and its output with accessibility”

[6] W. A. Barry & al, “Accesibility to scientific information by the blind: Dotsplus and ASTER could

make it easy", Department of Physics, Oregon state university.

Production rules	**Spatial relations hip**	**MathML Encoding**
R_1 : E → 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 → 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 → 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 → 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 → 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 → 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 → E E	E_1=L(2) and E_2=R(1)	E_0.code=***<mrow>***E_1.code***<mo>&invisibletimes;</mo>***E_2.code***</mrow>***
R_8 : E → + T	T=R(1)	E.code=***<mrow> <mo>+</mo>*** T.code ***</mrow>***
R_9 : E → - T	T=R(1)	E.code= ***<mrow> <mo> - </mo>*** T.code ***</mrow>***
R_{10} : E → T		E.code=T.code
R_{11} : E → $\int_L^L E\,d$ 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>∫</mo>***L_1.code L_2.code***</munsubsup>*** E_1.code ***<mo>DifferentialD ;</mo>*** V.code **else** E_0.code =***<mo>∫</mo>*** E_1.code ***<mo>DifferentialD ;</mo>*** V.code
R_{12} : E → $\sum_S^L E$	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 → $\prod_S^L E$	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>∏</mo>***S.code L.code ***</munderover>***E_1.code **else** E_0.code = ***<munder><mo>∏</mo>*** S.code ***</munder>***E_1.code
R_{14} : E → $\frac{E}{E}$	E_1=A(2) and E_2=D(2)	E_0.code = ***<mfrac>***E_1.code E_2.code ***</mfrac>***
R_{15} : E → $\sqrt[R]{E}$	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 → | E |	E_1=(R(1) and L(3))	E_0.code=***<mrow><mfenced open="|" close="|">***E_1.code ***</mfenced></mrow>***
R_{17} : E → (E)	E_1=(R(1) and L(3))	E_0.code= ***<mrow><mfenced>*** E_1.code ***</mfenced></mrow>***
R_{18} : E → 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 → ***det*** E	E_1=R(1)	E_0.code=***<mrow> <mo>det</mo>*** E_1.code***</mrow>***
R_{20} : E → ***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 → V		T.code = V.code
R_{22} :T → *unsigned_float*		T.code = ***<mn>Unisgned_float.val</mn>***
R_{23} :T→ *unsigned_integer*		T.code = ***<mn>Unsigned_integer.val</mn>***
R_{24} : L → E		L.code = E.code
R_{25} : L → +∞		L.code = ***<mrow><mo>+</mo><mo>∞</mo></mrow>***

R_{26} : L → **-∞**		L.code = ***<mrow><mo>-</mo><mo>∞</mo></mrow>***
R_{27} : L → **∞**		L.code = ***<mo>∞</mo>***
R_{28} : L → **ε**		
R_{29} : S → V = E	V=L(2) and E=R(2)	S.code = ***<mrow>***V.code***<mo>=</mo>***E.code***</mrow>***
R_{30} : S → V		S.code = V.code
R_{31} : R → V		R.code=V.code
R_{32}: R→***unsigned_integer***		R.code=***<mn> unsigned_integer.val</mn>***
R_{33} : R →**ε**		
R_{34} : X →E		X.code=E.code
R_{35} : X →**ε**		
R_{36} : D →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 → E		D.code=E.code
R_{38} : V → ***letter*** $_D$	D=s(1)	V.code=***<msub><mi> letter.val</mi>***D.code***</msub>***
R_{39} : V → ***letter***		V.code=***<mi> letter.val</mi>***

Table 1: Rule production and MathML encoding

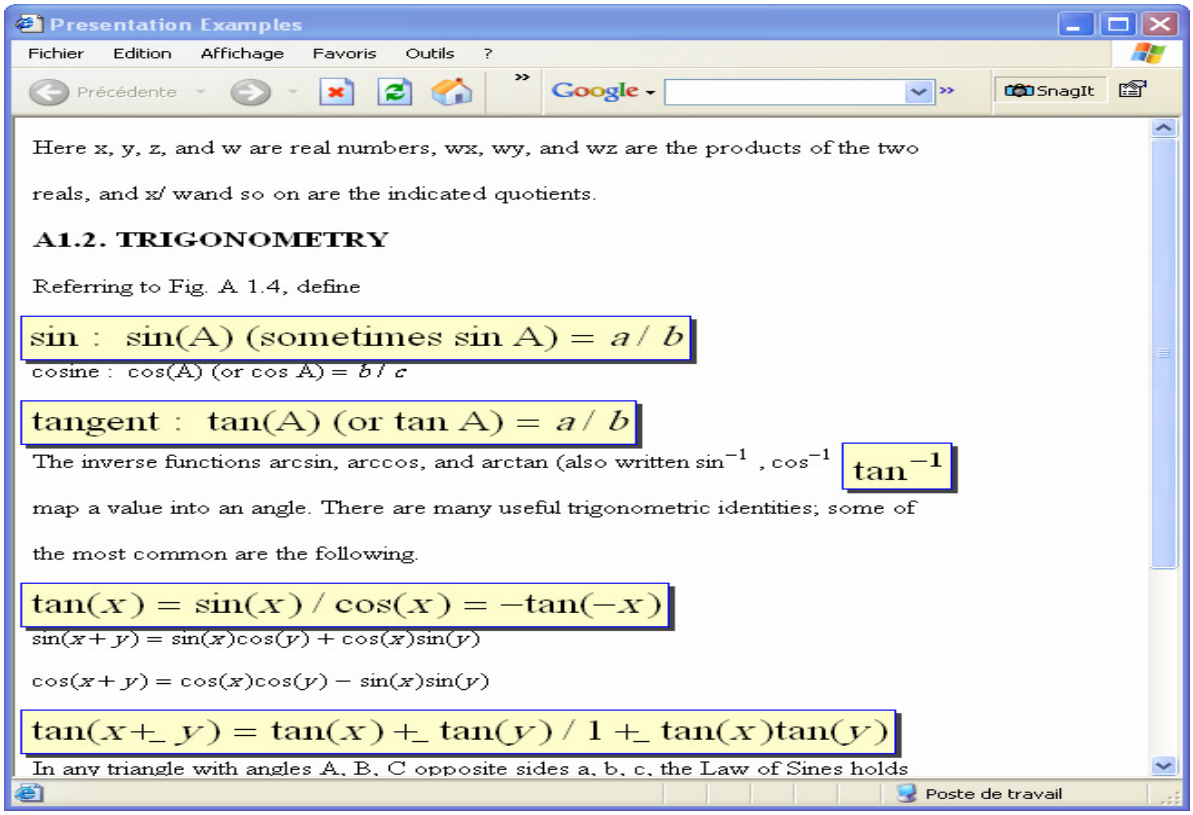


Figure 4: XML retro-conversion of a mathematical document