

Arabic Mathematical e-Documents

Mustapha Eddahibi
m.eddahibi@ucam.ac.ma

Azzeddine Lazrek
lazrek@ucam.ac.ma

Khalid Sami
k.sami@ucam.ac.ma
Department of Computer Sciences,
Faculty of Sciences,
University Cadi Ayyad
P.O. Box 2390,
Marrakech, Morocco
<http://www.ucam.ac.ma/fssm/RyDArab>

© Springer-Verlag - LNCS, Volume 3130, pp. 158-168, 2004

<http://www.springer.de/comp/lncs/index.html>

<http://www.springeronline.com/sgw/cda/frontpage/0.11855.5-147-22-33856338-0.00.html>

Abstract

What problems do e-documents with mathematical expressions in an Arabic presentation present? In addition to the known difficulties of handling mathematical expressions based on Latin script on the web, Arabic mathematical expressions flow from *right to left* and use *specific symbols* with a *dynamic cursivity*. How might we extend the capabilities of tools such as MathML in order to structure Arabic mathematical e-documents? Those are the questions this paper will deal with. It gives a brief description of some steps toward an extension of MathML to mathematics in Arabic exposition. In order to evaluate it, this extension has been implemented in Mozilla.

KEYWORDS: Mathematical expressions, Arabic mathematical presentation, Multilingual documents, e-documents, Unicode, MathML, Mozilla.

1 Overview

It is well known that HTML authoring capabilities are limited. For instance, mathematics is difficult to search and web formatting is poor. For years, most mathematics on the web consisted of texts with scientific notation rendered as images. Image-based equations are generally harder to see, read and comprehend than the surrounding text in the browser window. Moreover, the large size of this kind of e-document can represent a serious problem. These problems become worse when the document is printed. For instance, the resolution of the equations will be around 72 dots per inch, while the surrounding text will typically be 300 or more dots per inch. In addition to the display problems, there are encoding difficulties. Mathematical objects can neither be searched nor exchanged between software systems nor cut and pasted for use in different contexts nor verified as being mathematically correct. As mathematical e-documents may have to

be converted to and from other mathematical formats, they need encoding with respect to both the mathematical notation and mathematical meaning.

The mathematical markup language MathML [14] offers good solutions to the previous problems. MathML is an XML application for describing mathematical notation and capturing both its structure, for high-quality visual display, and content, for more semantic applications like scientific software. XML stands for eXtensible Markup Language. It is designed as a simplified version of the meta-language SGML used, for example, to define the grammar and syntax of HTML. One of the goals of XML is to be suitable for use on the web by separating the presentation from the content. At the same time, XML grammar and syntax rules carefully enforce document structure to facilitate automatic processing and maintenance of large document collections.

MathML enables mathematics to be served, received, and processed on the web, just as HTML

has enabled this functionality for text. MathML elements can be included in XHTML documents with namespaces and links can be associated to any mathematical expression through XLink. Of course, there are complementary tools. For instance, the project OpenMath [12] also aims at encoding the semantics of mathematics without being in competition with MathML.

Now, what about some of the MathML internationalization aspects—say, for instance, its ability to structure and produce e-documents based on non Latin alphabets, such as mathematical documents in Arabic?

2 Arabic Mathematical Presentation

Arabic script is cursive. Small curves and ligatures join adjacent letters in a word. The shapes of most of the letters are context dependent; that is, they change according to their position in the word. Certain letters have up to four different shapes.

Although some mathematical documents using Arabic-based writing display mathematics in Latin characters, in general, not only the text is encoded with the Arabic script but mathematical objects and expressions are also encoded with special symbols flowing from right to left according to the Arabic writing. Moreover, some of these symbols are extensible.

Mathematical expressions are for the most part handwritten and introduced as images. A highly-evolved system of calligraphic rules governs Arabic handwriting. Though Arabic mathematical documents written by hand are sometimes of fair quality, the mere presentation of scientific documents is no longer enough, since there is a need for searchability, using them in software and so on.

The RyDArab [8] system makes it possible to compose Arabic mathematical expressions of high typographical quality. RyDArab complements \TeX for typesetting Arabic mathematical documents. RyDArab uses the Computer Modern fonts and those of Ω [4] or Arab \TeX [7]. The output is DVI, PS, PDF or HTML with mathematical expressions as images. The RyDArab [2] system does not replace or modify the functionality of the \TeX engine, so it does not restrict in any way the set of macros used for authoring. Automatic translation from and to Latin-based expressions is provided beginning with the latest RyDArab version. Will this be enough to structure and typeset e-documents with mathematics even when they are based on an alternative script? Starting from this material with \TeX and Ω , will MathML be able to handle Arabic mathematics?

3 MathML and Arabic Mathematics

Of course, semantically speaking, an Arabic mathematical expression is the same as a Latin-based one. Thus, only display problems need be taken into account. In any way, encoding semantics are beyond the scope of this paper.

In order to know if there really is a need to construct a new tool or only to improve an already available one, what are the possibilities offered by the known MathML renderers? As much of the work is built around \TeX , an *open source community effort*, it is hard to be precise about the current status of all \TeX /MathML related projects. Most of these projects belong to one of three basic categories:

- Conversions from \TeX to MathML. Of particular note here, are Ω [5, 6] and $\TeX4ht$ [13], a highly specialized editor/DVI driver. Both of these systems are capable of writing presentation MathML from \TeX documents. There are other converters such as $\LaTeX2HTML$ and \tralics [1].
- Conversions from MathML to \TeX . The conversion from MathML to \TeX can be done for instance, through reading MathML into Mathematica or other similar tools and then saving the result back out as \TeX , or using Scientific WorkPlace for suitable \LaTeX sources. The Con \TeX t system is another example.
- Direct typesetting of MathML using \TeX .

Currently, MathML is supported by many applications. This fact shows not only that it is the format of choice for publishing equations on the web but also that it is a universal interchange format for mathematics. More than twenty implementations are listed on the MathML official website, showing that all categories of mathematical software can handle MathML. Actually,

- most mathematical software, such as Scientific WorkPlace, Maple, MathCad and Mathematica, can export and import MathML;
- all common browsers can display MathML equations either natively or through the use of plug-ins;
- editors such as MathType, Amaya, \TeX macs, and WebEQ support MathML.

Once non-free or non-open-source tools are omitted, two web browsers remain: the well-known Mozilla system [11] and Amaya. The W3C's Amaya editor/browser allows authors to include mathematical expressions in web pages, following the MathML specification. Mathematical expressions are handled as structured components, in the same way

and in the same environment as HTML elements. All editing commands provided by Amaya for handling text are also available for mathematics, and there are some additional controls to enter and edit mathematical constructs. Amaya shows how other W3C specifications can be used in conjunction with MathML.

In the end, we chose to adapt Mozilla to the needs of the situation, mainly because of its popularity and widespread adoption as well as the existence of an Arabic version. The layout of mathematical expressions in Latin writing, and consequently that of the mathematical documents in Mozilla is more elegant and of good typographical quality compared to other systems.

For this implementation, we used the Mozilla 1.5 C++ source under Linux. Until now, there was no Mozilla version with support for bidirectionality or cursivity in a mathematical environment. In math mode, only left to right arrangement is supported. Thus, the first step is to find out how to get bidirectionality and cursivity inside a MathML passage.

In fact, adding the property of bidirectionality to MathML elements is a delicate task. It requires a careful study of the various possible conflicts. The bidirectionality algorithm for mathematical expressions is probably different from that originally in use for text.

Now, let us have a look at what would happen if the bidirectionality algorithm for HTML were used for MathML elements.

The MathML expression

```
<mn>1</mn>
<mo>+</mo>
<mi>ب</mi>
<mo>-</mo>
<mn>2</mn>
```

will be rendered as $1 + 2 - ب$ instead of the expected equation: $1 + ب - 2$.

Since XML supports Unicode, we might expect that the introduction of Arabic text into MathML encoding would go without any problem. In other words, the Arabic text would be rendered from right to left, and letters would be connected just as they should be in their cursive writing. Will the use of the element `<mtext>` (similar to the use of the \TeX command `\hbox`) be enough to get a satisfactory rendering of Arabic?

The following Arabic text is a sample of what is obtained with `<mtext>` in Mozilla:

```
<mtext>
نص رياضي
</mtext>
```

رياضي

The following Arabic abbreviation of the cosine function is an example of what we get if we introduce it with `<mi>`:

```
<mi>جتا</mi>
```

جتا

In order to allow the arrangement of sub-expressions from right to left in a given mother expression, a new element denoted `<rl>` is introduced.¹

```
<mrow>
<rl>
<mi>ب</mi>
<mo>+</mo>
<mi>س</mi>
</rl>
</mrow>
```

ب + س

The use of the element `<rl>` also allows solving the previous problem of introducing Arabic text in a mathematical expression.

```
<mtext>
<rl>نص رياضي</rl>
</mtext>
<mi><rl>جتا</rl></mi>
```

نص رياضي

جتا

The element `<rl>` can be used to transform some mathematical objects, such as left/right or open/close parentheses, into their mirror image.

```
<rl>
<mo><rl>[</rl></mo>
<mi>ب</mi>
<mo>,</mo>
<mn>3</mn>
<mo><rl>]</rl></mo>
</rl>
```

(3 , ب]

We can remark here that the symbol “,” has not changed to its mirror image “;”. The result is the same even when the comma is governed by `<rl>` (i.e., `<rl>`, `</rl>`). This symbol is not yet mentioned in the Bidi Mirroring list in the Unicode Character Database.

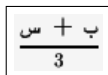
Particular arrangement of the arguments is made necessary by the MathML renderer for any presentation elements requiring more than one argument. On the other hand, elements of vertical arrangement such as `<mfrac>` do not need special handling.

¹ The name `rl` reminds us of the initials of right-to-left. Furthermore, because of the expected heavy use of this element, its name should be as short as possible.

```

<mfrac>
  <rl>
    <mi>ب</mi>
    <mo>+</mo>
    <mi>س</mi>
  </rl>
  <mn>3</mn>
</mfrac>

```



Although the addition of `<rl>` helps to get Arabic text rendered as expected and to solve arrangement of sub-expressions within an expression, for certain elements, it doesn't work.

Using the element `<rl>` to get a superscript element `<msup>` or a subscript element `<msub>`, in the suitable right to left positions, generates a syntax error because `<msup>` requires two arguments, whereas there is only one argument, as can be seen in the following example:

```

<msup>
  <rl>
    <mi>س</mi>
    <mi>ب</mi>
  </rl>
</msup>

```

invalid-markup

In this case, we introduce a new markup element `<amsup>`.² It changes the direction of rendering expressions while keeping the size of superscripts as it is with `<msup>`.

```

<amsup>
  <mi>س</mi>
  <mi>ب</mi>
</amsup>

```



The same principle is applied to other elements like `<msub>`. The notation of the arrangement in the Arabic combination analysis is different from its Latin equivalent.

```

<amarrange>
  <mi>ل</mi>
  <mn>5</mn>
  <mn>2</mn>
</amarrange>

```



The next step is related to the shape of some symbols. In Arabic mathematical presentation, certain symbols, such as the square root symbol or the sum, in some Arabic areas, are built through a symmetric reflection of the corresponding Latin ones. These symbols require first the introduction of a new font family such as the one offered in the Arabic

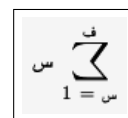
² The following new elements defined in this system are prefixed with the initial of Arabic "a".

Computer Modern fonts. This family corresponds to the Computer Modern fonts with a mirror effect on some glyphs. In the same way that the Computer Modern fonts are used in the Latin mathematical environment with `<math>` the new element `<amath>` will allow the use of the Arabic Computer Modern fonts in the Arabic mathematical environment. The element `<amath>` would not be necessary if the Arabic mathematical symbols were already added in the Unicode tables. In fact, we use the same entity name and code for some symbols and their mirror images used in the Arabic presentation. For example, the Unicode name N-ARY SUMMATION coded by U+02211 is associated simultaneously to the Latin \sum symbol and to its Arabic equivalent mirror \sum . Thus, to specify which glyph, and consequently which font, is called, the introduction of a new element `<amath>` is necessary. This element would not be necessary if the symbols were denoted with two different entity names and consequently two different codes.

```

<amath>
  <rl>
    <mstyle displaystyle="true">
      <munderover>
        <mo>&sum;</mo>
        <mrow>
          <rl>
            <mi>س</mi>
            <mo>=</mo>
            <mn>1</mn>
          </rl>
        </mrow>
        <mi>ف</mi>
      </munderover>
    </mstyle>
    <mi>س</mi>
  </rl>
</amath>

```



In order to distinguish alphabetical symbols, in different shapes, from letters used in Arabic texts, and to avoid the heterogeneity resulting from the use of several fonts, there is a need for a complete Arabic mathematical font. That's exactly what we are trying to do in another project discussed elsewhere in this volume [10]. While waiting for their adoption by Unicode, the symbols in use in this font will be located in the Private Use Area E000-F8FF in the Basic Multilingual Plane.

```

<mi>&#xE004;</mi>

```



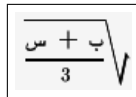
The use of the Arabic Computer Modern fonts is not enough for composed symbols. For example,

³ introduced as `∑` or `∑`.

the square root symbol is composed of the square root glyph supplemented by an over bar. This over bar is added by the renderer, which, thanks to a calculation of the width of the base, gives the length of this over bar.

In this case, neither the inversion of the glyph nor the use of the right-to-left element `<rl>` changes the direction of the visual rendering of the square root. For this reason we have introduced a new element (`<amsqrt>`), which uses the square root glyph from the Arabic Computer Modern font that shows the over bar to its left.

```
<amath>
<amsqrt>
<mfrac>
<rl>
<mi>ب</mi>
<mo>+</mo>
<mi>س</mi>
</rl>
<mn>3</mn>
</mfrac>
</amsqrt>
</amath>
```



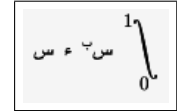
The root element `<amroot>` requires a treatment similar to that of the square root combined with the positioning of the index on the right of the baseline.

```
<amath>
<amroot>
<mi>س</mi>
<mn>3</mn>
</amroot>
</amath>
```



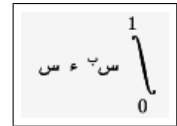
For the elements `<munderover>`, `<munder>`, `<mover>`, and `<mssubsup>`, italic correction needs to be done. In fact, mathematical symbols like the integral are slanted and the indices and exponents are shifted in the direction of the symbol's slant. This fact appears clearly in the following example representing two integrals, while using `<amsubsup>` in the first:

```
<amath>
<rl>
<mstyle displaystyle="true">
<amsubsup>
<mo>&int;</mo>
<mn>0</mn>
<mn>1</mn>
</amsubsup>
</mstyle>
<amsup>
<mi>س</mi>
<mi>ب</mi>
</amsup>
<mi>ء</mi>
<mi>س</mi>
</rl>
</amath>
```

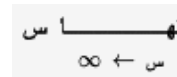


or `<amunderover>` in the second:

```
<amath>
<rl>
<mstyle displaystyle="true">
<amunderover>
<mo>&int;</mo>
<mn>0</mn>
<mn>1</mn>
</amunderover>
</mstyle>
<amsup>
<mi>س</mi>
<mi>ب</mi>
</amsup>
<mi>ء</mi>
<mi>س</mi>
</rl>
</amath>
```

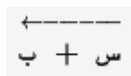


For the limit of an expression, manual lengthening of the limit symbol is performed. Of course, dynamic lengthening via automatic calculation of the width of the text under the limit sign would be better.



A lengthening of the straight line is not in conformity with the rules of the Arabic typography. A curvilinear lengthening is required, which can be obtained by using CurExt [9], which makes it possible to stretch Arabic letters according to calligraphic rules.

The following mathematical expression is an example of the use of `<mover>`, with automatic lengthening of the over arrow.



In fact, the use of the element `<r1>` doesn't represent a very practical solution as the encoding becomes heavier. The addition of this element must be transparent for the user; the same for all other new elements since they affect only the presentation and not the semantics of expression. An alternative solution consists of either building a new algorithm of bidirectionality for mathematics, or of adding attributes that will make it possible to choose the mathematical notation of the expression. We intend to use a new attribute `nota` for the root element `<math>`. It would indicate whether Arabic or Latin is used inside the mathematical expression. As the layout of a mathematical expression follows a precise logic, the direction of writing would be handled automatically without requiring the use of direction attributes for each child of the element `<math>`.

The FIGUE [3] system is an engine for the interactive rendering of structured objects. It allows the rendering of an Arabic text from right to left including some Latin mathematical expressions flowing from left to right thanks to a proposed bidirectional extension of MathML.

4 Conclusion

Our goal was to identify the difficulties and limitations that might obstruct the use of MathML for writing mathematics in Arabic. The main adaptation we made to MathML for Arabic mathematics was the addition of the element `<r1>` that allows:

- writing mathematical expressions from right-to-left;
- the use of specific symbols thanks to the modification of other elements;
- and handling the cursivity of writing.

Now, Arabic mathematical e-documents can be structured and published on the web using this extended version of Mozilla. Such documents can thus benefit from all the advantages of using MathML. Our project for the development of communication and publication tools for scientific and technical e-documents in Arabic is still at its beginning. We hope that the proposals contained in this paper will help to find suitable recommendations for Arabic mathematics in Unicode and MathML.

References

- [1] <http://www-sop.inria.fr/miaou/tralics/>.
- [2] Mustapha Eddahibi and Azzeddine Lazrek, *Arabic scientific document composition*, International Conference on Information Technology and Natural Sciences (ICITNS 2003, Amman, Jordan), 2003.
- [3] Hanane Naciri et Laurence Rideau, *Affichage et diffusion sur Internet d'expressions en langue arabe de preuves mathématiques*, CARI 2002 (Cameroun), 2002.
- [4] Yannis Haralambous and John Plaice, Multilingual Typesetting with Ω , a Case Study: Arabic, *Proceedings of the International Symposium on Multilingual Information Processing (Tsukuba)*, 1997, pp. 137–154.
- [5] Yannis Haralambous and John Plaice, Produire du MathML et autres *ML à partir d' Ω : Ω se généralise, *Cahiers GUTenberg*, vol. 33-34, 1999, pp. 173–182.
- [6] Yannis Haralambous and John Plaice, *X_{La}TeX, a DTD/Schema Which is Very Close to L_ATeX*, EuroTeX 2003: 14th European TeX Conference (ENST Bretagne, France), 2003 (to appear in *TUGboat*).
- [7] Klaus Lagally, *ArabTeX — Typesetting Arabic with Vowels and Ligatures*, EuroTeX'92 (Prague), 1992.
- [8] Azzeddine Lazrek, A package for typesetting arabic mathematical formulas, *Die TeXnische Komödie*, DANTE e.V., vol. 13. (2/2001), 2001, pp. 54–66.
- [9] Azzeddine Lazrek, *CurExt, Typesetting variable-sized curved symbols*, EuroTeX 2003 preprints: 14th European TeX Conference (Brest, France), 2003, pp. 47–71 (to appear in *TUGboat*).
- [10] Mostafa Banouni, Mohamed Elyaakoubi and Azzeddine Lazrek, *Dynamic Arabic mathematical fonts*, International Conference on TeX, XML and Digital Typography (TUG 2004, Xanthi, Greece), 2004.
- [11] Mozilla, <http://www.mozilla.org>.
- [12] OpenMath, <http://www.openmath.org/>.
- [13] TeX4ht, <http://www.cis.ohio-state.edu/~gurari/TeX4ht/mn.html>.
- [14] Presentation MathML and Content MathML, <http://www.w3.org/TR/MathML2>.