Tralics, a LaTeX to XML Translator

José Grimm
INRIA
2002, Route des Lucioles, BP 93
06902 Sophia Antipolis CEDEX
Jose.Grimm@sophia.inria.fr
http://www-sop.inria.fr/miaou/Jose.Grimm

Abstract

In this paper we describe Tralics, a LaTeX to XML translator. A previous version of the software (written in Perl) was used to obtain the pdf version of Inria's "Rapport d'Activité" for year 2001. The current version of the software (written in C++) is used for both the HTML and pdf version for the year 2002. The XML generated by Tralics conforms to a local DTD, similar to the TEI; it was converted to pdf using pdfTeX and the `xmltex` package, and the HTML was obtained via an XSLT processor.

We explain here the philosophy of the software, its usage, limitations, and customization.

Résumé

Dans cet article nous décrivons le logiciel Tralics, un traducteur de LaTeX vers XML. Une version antérieure de ce logiciel, écrite en Perl, a été utilisée pour générer la version pdf du Rapport d'activité de l'Inria en 2001. La version actuelle du logiciel, écrite en C++, a été utilisée pour obtenir à la fois le HTML et le pdf de la version 2002 : nous avons utilisé une DTD locale, similaire à la TEI, et pdfTeX plus `xmltex` pour obtenir le pdf.

Nous expliquons ici la philosophie de Tralics, son usage, ses limitations, et comment paramétrer le logiciel.

Introduction

If you run Tralics on a document such as this article, you will get as output an XML document that starts and ends like this:

```
<?xml version='1.0' encoding='iso-8859-1'?>
<!DOCTYPE eurotex SYSTEM 'tugboat.dtd'>
<!-- translated from latex by tralics 1.8a-->
<eurotex language='english'>
<titlepage>
<ti>Tralics, a <LaTeX/> to XML translator</ti>
<NetA>Jose.Grimm@sophia.inria.fr</NetA>
<U>http://www-sop.inria.fr/miaou/Jose.Grimm</U>
<resume><p>
In this paper we describe Tralics,
...
<byear>1999</byear>
</citation>
</biblio></eurotex>
```

All that is needed for this example to work is to put, in the configuration file read by Tralics, the following lines:

```
BeginType  eurotex
  DocType = eurotex tugboat.dtd
BeginTitlePage
  \maketitle <titlepage> "" ""
  \title <ti> "No title"
  \netaddress <NetA>  "No address"
  \personalURL <U> "No url given"
  \resume E<resume> "Pas de résumé"
  \abstract E<abstract> "no abstract"
  \author + <author> <auth> "No authors"
  \address p<address>  "no address"
End
End
```

Once you have an XML version of the document, you can use any tool you wish to process it (we used the Gnome library, `http://xmlsoft.org/`), but something has to be done. There are typically two kinds of applications. The first is that everybody publishing an Inria Technical Report has to give, together with the typeset PostScript document, the start of the LaTeX source, which is processed by an ad hoc tool that adds a new item to the publication database. One could use Tralics to convert the partial document, and an XSLT style sheet to extract from the `<titlepage>` element all relevant information: author, title, abstract, keywords, etc. In such a situation, the presence of an unknown element like `<LaTeX/>` or an unexpected math formula causes no problem; the element can be replaced by its name, and a math formula removed (the current process behaves this way).

A second type of application is the following: for some reasons, the LaTeX document needs to exist in both PostScript and HTML form, and XML is used as an intermediate, common, format. In such a case there should be little difference between the PostScript as seen by the author (direct route) and the final version (the XML route). As a result, the translation of `\LaTeX` must match the LaTeX logo at best, and it is not possible to reject a math formula under the pretext that it does not fit in the MathML standard. As indicated on the figure on page 104 of [1], there are lots of possibilities, none of which is very simple. In practice, even more tools are needed if the objective is a complete web site.

This paper has two parts: we first explain the main application of Tralics (why a new program, how it is used, what is done with the resulting XML, etc.); after that, we shall explain what Tralics does in the same order as the *LaTeX Companion* [3].

Why another LaTeX to X translator?

Inria's annual activity report has a long history, as long as the history of Inria itself. In fact, each research team has to report, each year, on its activity; this set of documents, which we call the RA in what follows, is sometimes called the scientific annexes to the annual report (which is a bilingual document on glazed paper with lots of figures and photographs, produced by a specialized company).

Since 1987, the RA has been typeset by LaTeX, using a specific document style (or document class), see [7]. Since then, the number of pages has evolved from one thousand to three thousand, and the document is no longer printed on paper (a CD-ROM version has been available since 1999). On the other hand, an HTML version of the document has been available since 1994; it was produced by `latex2html` (using SGML as an intermediate language for the first year).

In the years 1997–98, some discussions showed that it would be nice to redesign the RA, using XML as the intermediate language. We contacted some companies, but got no good answer: nobody was able to translate the LaTeX input into an XML output, no guarantee was given about the quality of a printed version of the XML, and mathematical support was very poor. These three points will be discussed later on.

These discussions were not completely negative: a new concept emerged, the "raweb". Essentially, it is formed of "modules" that can be read independently (lots of nice properties of these modules are still to be implemented; for instance, depending on the user profile, these modules could be presented in different orders). Each module is equivalent to a web page, and can be put in a "classeur" (French equivalent of a loose-leaf file, it's like a caddy but elements are ordered; there is no official English name for it). The objects in the caddy can be re-ordered, and the caddy can be transformed into a single HTML document (in 1999, the caddy contained the LaTeX version of the modules, hence was able to produce a PostScript document; we hope that next year it will contain the XML version of the modules, but running pdfTeX on the web server is not an option).

Moreover, the notion of an RA-conforming document was formalized (and this served as the basis of the raweb DTD), and a syntax checker was written in Perl. This checker has some intrinsic limitations: since it was unaware of macro expansion and commands like `\csname`, it was very easy to fool it.

Our new translator

In fall 2001, we decided to fabricate a translator and to test it; the tool was based on `ltx2x` (a translator by Peter Wilson, available on CTAN) and Ω (a TeX extension that can produce MathML as a byproduct, [5]), together with a Perl script that was used as a syntax checker, a module splitter, and a pre- and post-processor. A second version of the tool, written entirely in Perl, was used for the pdf version of the RA2001.

The description of the software can be found in [4], together with lots of examples that could not be translated. For instance this one:

```
\catcode `$=\active %$emacs
\def$#1~{\catcode`$=3 zut}
$\left[1=2\right)$ is a formula~!
```

In May 2002, we decided to completely rewrite the software, and to call it `tralics`. The previous example is understood correctly by Tralics, meaning that Tralics groks catcodes, active characters, delimited arguments, etc. More complicated is

```
{\catcode`\_\active\global\let_X}
\begingroup\lccode `\~=`\_
\catcode`\~\active
\lowercase\relax{~\def~{u}\endgroup ~a}~
\MakeUppercase{abAB\ae\i}
```

This shows that `\lowercase` is fully implemented, including funny details, like the optional `\relax` before the open brace. One objective was to fully implement the TeX macro expansion mechanism, including conditionals (and to translate most LaTeX constructs). The following example provokes an error:

```
\font\foo=cmr10 scaled 1023
\setbox0=\hbox to 10pt{\foo p\hss}
\ifdim\dp0>0pt 1\else 2\fi
```

Here, the first line defines a command `\foo` that selects some font; since we do not want to parse font metric files, all these fonts are equivalent and `\foo` is a no-op. The second line constructs a horizontal box and puts its content into `\box0`; we shall explain later what Tralics does with the box. One thing is clear: the objective of Tralics is not to split a paragraph into lines, so there is no packaging, no overfull hboxes, no underfull vboxes. In fact, the box specifications and the `\hss` command are ignored, the resulting box contains only the letter p. If you ask for the depth of the box, Tralics signals an error (and returns 0).

Note that `\ifvmode` and `\ifinner` are part of TeX's macro expansion mechanism, but these commands are not implemented, mostly because there is little relationship between the modes of TeX and the modes of Tralics. All you need to know is that the following lines work as expected:

```
\ensuremath{\Omega}
\leavevmode
\mbox{\par}
```

The last line is silly: inside a box, `\par` cannot start a new paragraph, and the resulting box is empty. For simplicity, the `\par` token is allowed anywhere (as if every command were prefixed with `\long`). Why `\mbox` is declared `\long` in LaTeX is beyond me.

The target language

The main job of a translator like Tralics is to read and analyze the input document, manage it (and simplify, complete, re-order, etc.) and produce a new document. The big question here is: what elements should be chosen? In the case of `latex2html`, the choice is limited to HTML version 2.0, 3.0, 3.2, and 4.0. In the case of Tralics, there is no predefined set of elements. In the opening example, we used `<U>` for the `\personalURL` command, purely to avoid long lines; usually, more suggestive names should be used.

The default behavior of Tralics is defined by its main application, namely the raweb, with the following constraints: the resulting XML has to be converted to a set of HTML pages, one page for each module; it has to be converted to PostScript or pdf, one document per research team, without loss in typographic quality; and finally, mathematics should not be forgotten. Two style sheets were used, one to produce HTML, and another for pdf. Formatting a document is not trivial, because some objects have to be evaluated more than once, sometimes depending on the context (this explains why some LaTeX commands are fragile). In our case, the formatting was done by TeX or pdfTeX, which was able to parse XML files thanks to the `xmltex` package, see [1]. There is still work to do, in particular concerning tables: the implementation of tables in `fotex.sty` is rather disturbing; 400 lines of TeX code were needed (for instance to patch the behavior of `<mfenced>` and `<mover>`, and to get the cover page right).

For the DTD, we first considered the Docbook DTD, [10], and discarded it, because it was too complicated. We then considered the TEI DTD, and simplified it drastically. Lots of interesting material can be found on that web site (`http://www.tei-c.org/`), including PassiveTeX [8], a companion tool to `xmltex`, and a model for the above-mentioned style sheets.

On the other hand, we added some features, in order to implement figures, tables, math, etc. For obvious reasons, we use the presentation elements of MathML [2]. The design of the bibliography is not yet done (more on this subject will follow).

There are nearly one hundred elements in this DTD, 30 elements being specific to the raweb, 20 for the bibliography; other elements are borrowed from TEI. Via the configuration file, you can change the default names (except those defined by MathML, and those specific to the raweb); using commands like `\xmlelement`, you can generate additional elements. If you do so, you can convince Tralics to use your own DTD.

The Tralics software, the documentation, and the raweb DTD can be found at:

`http://www-sop.inria.fr/apics/tralics/`

Application to the raweb

The main application of Tralics can be found on Inria's web site:

`http://www.inria.fr/rapportsactivite`

For year 2002, 125 teams have written their activity report (in LaTeX, with one exception) and the result is over 3000 pages of pdf (A4 format, 10 point font size), resulting in 3890 HTML pages, that occupy 598.386 MB of a CD-ROM. Translating these files took 100 seconds (30 seconds if Tralics is compiled with the `-O2` switch, this shows that the C++ optimizer does a good job). The most time consuming operation was the conversion from XML to pdf (the `xmltex` parser is very slow), and production of images (conversion from PostScript to pdf via the `epstopdf` Perl script; and to png, using tools borrowed from `latex2html`).

Lots of people were involved in the process: first, the raweb team, which contributed to the design of the raweb (the web site, the paper version, the idea of using XML, the DTD), the authors of the texts, the people who collected the texts and who corrected some typos, and Marie-Pierre Durollet who deserves special thanks: she wrote some shell scripts, Perl scripts, cgi scripts, part of the style sheets, etc., in fact, all that is needed to make the web site function. She was also the first real user of Tralics, in that she translated all the files, on her Linux box (other people tried Tralics on SunOS, MacOS, Windows).

One non-trivial point is the question of mathematics: there are some browsers (Amaya, Mozilla, etc.), that understand MathML, or claim to do so, together with plugins and other tools supposed to visualize MathML, but we decided nevertheless to convert all the math. In fact, Tralics comes with a Perl script (a bit over one thousand lines), that reads an XML file, and converts math and images.

Image post-processing Consider a simple, classical, example:

```
\begin{figure}
  \begin{center}
    \includegraphics{miaou_transf.ps}
  \end{center}
  \caption{Modèle de transducteur.}
  \label{trans}
\end{figure}
```

It is translated into

```
<figure file='miaou_transf' id='uid30'>
<head>Modèle de transducteur.</head>
</figure>
```

As can be seen, the argument of the `\label` command was normalized from 'trans' to 'uid30' and became an attribute of the figure element, the centering environment was ignored, the underscore treated as a normal character, the extension of the file name was removed, and the file became an attribute of the figure element (see below an example where more than one image appears in a figure environment). The post-processing script has the job of making sure that the image exists in PostScript, pdf and png formats. It also modifies the XML:

```
<figure aux='image_1.png'
  file='miaou_transf' id='uid30'>
<head>Modèle de transducteur.</head>
</figure>
```

and the HTML result will be:

```
<div align="center">
  <a name="uid30"></a>
  <table>
    <caption align="bottom">
      <strong>Figure 1. </strong>
      Mod&#xe8;le de transducteur.
    </caption>
    <tr><td><img alt="miaou_transf"
           src="image_1.png"></td></tr>
  </table>
</div>
```

A few remarks are needed: the initial LaTeX environment is a floating one, so that the result is outside a `<p>` element, and using `<div>` for grouping is justified. The figure and its caption are centered. Note that the image is a cell of a table, because this is the easiest way to associate a caption to the table.

The è character was transformed into `&#xe8;` by an ad hoc program, because, for some unknown reasons, the style sheet refused to use latin1 encoding, forcing UTF-8 instead, an encoding not understood by the program that indexes Inria's web site.

The figure was the first one in the file, so that the style sheet numbered it as figure one (in the next example, there is no `\caption`, hence the figure has an empty `<caption>`, but nevertheless a unique number). The HTML translation of `\ref{trans}` is something like `<a...>1</a>`. The `<a>` element has a href attribute whose value depends on the name of the label, and the current web page, which was renamed from 'resonn', the name given by the author, to 'module7', which has no meaning (searching for 'module50' on Inria's web site reveals that three teams have written at least 50 modules). The important point however is the content of the `<a>` element: the value 1 is not computed by Tralics, but by the style sheet.

Another example is the following.

```
\begin{figure}[htbp]
\begin{center}
\begin{tabular}{ccc}
   \includegraphics[width=3.5cm]{imgl}&
   \includegraphics[width=3.5cm]{imgc}&
   \includegraphics[width=3.5cm]{imgr}\\
cap l&cap c&cap r
\end{tabular}
\end{center}
\end{figure}
```

The same effect could be achieved with the subfigure environment, but we wanted to give an example of a table. Note that, if more than one image is to be put in a figure, we suggest using a table, since there are still unresolved problems regarding spacing. The translation of the previous example is the following

```
<figure rend='array' id='uid9'><p>
  <table rend='inline'> <row>
     <cell><figure file='imgl'/></cell>
     <cell><figure file='imgc'/></cell>
     <cell><figure file='imgr'/></cell>
    </row><row>
      <cell>cap l</cell>
      <cell>cap c</cell>
      <cell>cap r</cell>
  </row></table></p>
</figure>
```

In order to reduce the size, we did omit the halign = center attributes for the six `<cell>` elements, and the width=3.5cm, rend=inline attributes of the three non-toplevel `<figure>` elements. The HTML translation is a table in a table, the inner table has no caption, the outer table has an empty caption. One of these two tables could be removed.

Math post-processing Note that a table or a figure is a float (hence is numbered) if and only if its rend attribute is not inline; on the other hand, math formulas never float, but are numbered only if they are non-inline, and have a label. In what follows, we shall distinguish between the math (defined by MathML) and the formula (something we borrowed from the TEI DTD). It is the formula that has a label (hence a number), and this explains why we have some troubles translating environments like `align` and commands like `\tag`. Nevertheless, the translation of the simple formula $K \neq \Gamma$ is

```
<formula type='inline'><math><mrow>
<mi>K</mi><mo>&ne;</mo><mi>&Gamma;</mi>
</mrow></math></formula>
```

For completeness, we show here the XSL-FO output that is used to convert the formula into pdf (complete code for the example above, skeleton for a display-math formula, numbered and unnumbered).

```
<m:math overflow="scroll"><m:mrow>
  <m:mi>K</m:mi>
  <m:mo>&#x2260;</m:mo>
  <m:mi>&#x393;</m:mi>
</m:mrow></m:math>

<fotex:displaymath>
  display-math stuff
</fotex:displaymath>

<fo:inline id="uid11"><fotex:equation>
  numbered math stuff
</fotex:equation></fo:inline>
```

In this case, the Perl script that handles images and math formulas classifies the formula as a simple one, formed of three tokens in a row, a Latin letter, a symbol and a Greek letter. It replaces the formula by the following XML code.

```
<span class='math'><hi rend='it'>K</hi>
  <img src='img_other_ne.png' alt='$\ne$'
   width='14' height='26' align='middle'/>
 <img src='img_Gamma.png' alt='$\Gamma$'
   width='12' height='13' align='bottom'/>
</span>
```

Images for Greek letters and other symbols were precomputed (by `latex2html`). The dimensions given here are nearly twice the size of a ten-point $\neq$ and $\Gamma$, except that the depth of the $\neq$ should be 4, and there is no way to indicate a depth in HTML. That's the reason why we indicate a total height plus depth of 26, together with an 'align=middle' attribute. The resulting HTML is ugly (too much blank space with the line and the following one), and uses a deprecated feature, but we do not know how to do better. The resulting HTML is

```
<SPAN class="math">
<SPAN class="it">K</SPAN>
<IMG width="14" height="26" align="middle"
border="0" alt="$\ne$"
src="../images/img_other_ne.png">
<IMG width="12" height="13" align="bottom"
border="0" alt="$\Gamma$"
src="../images/img_Gamma.png"></SPAN>
```

Let's give another example: $L^2 \rightarrow L^\infty$ is translated into MathML as

```
<math><mrow>
  <msup><mi>L</mi><mn>2</mn> </msup>
  <mo>&rightarrow;</mo>
  <msup><mi>L</mi><mi>&infin;</mi></msup>
</mrow></math>
```

The result here is a single image:

```
<span class='math'>
 <img align = 'bottom'
  width ='71' height ='13'
  src='math_image_1.png' border='0'
 alt='Im1 ${L^2\rightarrow L^\infin }$'/>
</span>
```

A formula of the form a^b or a_b is considered simple (and translates to `<sup>` or `<sub>`) in the case where b is an HTML character. Here, one exponent is infinity, hence an image is needed. Note how the alt field is constructed: we try to reconstruct the TeX formula from the MathML element. The first token, the `Im1`, indicates the image number, and is only useful for debugging. Constructing the image is not so trivial. First, a file is created that contains lines of the form

```
<formula id="1"><math><mrow>
  <msup><mi>L</mi> <mn>2</mn></msup>
  <mo>&#8594;</mo>
  <msup><mi>L</mi><mi>&#8734;</mi></msup>
</mrow></math></formula>
```

In fact, the file contains all formulas that need to be converted, with a unique id identifying the image. The xmllint processor is used to replace entity names by their Unicode values. The resulting file is processed by LaTeX, in the same fashion as the main document, in order to get a DVI file. The interpretation of the `<formula>` element (which is absent from the XSL-FO file) uses code borrowed from `latex2html`, which has two side effects. First, the log file contains a line like

```
l2hSize :1:8.14003pt::0.0pt::48.73616pt.
```

and second, each page of the DVI file contains a single math formula in its upper left corner. For each such page a PostScript file is generated by `dvips` and then converted to png via the `pstoimg` utilities, which use the size information shown above. After a magnification factor of 40%, this gives a resulting image of 13 by 71 pixels, and this information is pushed back in the XML file.

The structure of a LaTeX document

Tralics assumes that the document to be translated conforms to LaTeX standards: that there is a line with `\documentclass`, followed by some lines containing `\usepackage` commands, followed by a `document` environment. This is a very special environment, because its content is at brace level zero (as in standard LaTeX), and the tokens read by `\AtBeginDocument` and `\AtEndDocument`
are inserted at the right place, for instance

```
\documentclass{article}
\usepackage{calc}
\AtBeginDocument{start}
\AtEndDocument{end}
\newlength{\foo}
\AtEndDocument{\AtEndDocument{ realend}}
\begin{document}
\setlength{\foo}{1cm+3pt}
\the\foo
\end{document}
This line is not translated
```

would produce essentially the following

```
<p>start
31.45274pt
end realend</p>
```

There is a special hack here: a special `\endinput` token is inserted, whose effect is to stop translation of the current file (after the tokens remembered by the document hook), but the job of the translator continues, because it is at this moment that the bibliography is translated: Tralics has the list of all `\cite` commands, so it can do the equivalent of BibTeX. The resulting bibliography is inserted where the `\bibliography` command is located. It is an error if a `\cite` command is seen after the end of the document (i.e. comes from a BibTeX file).

Parameterization The first job of Tralics is to read the source file and find a `\documentclass` command, in order to apply document specific rules found in the `.tralics_rc` file (the configuration file, it can be in the current or home directory). In the case where the configuration file contains

```
BeginType article
  on package loaded calc CALC = "true"
  on package loaded foo/bar FOO = "true"
End
```

then, whenever the document class is `article` (trailing digits are ignored in the name) the two lines are executed. It follows that the root element of the resulting XML document will have the attribute CALC set to true, in case the `calc` package is loaded, and FOO is set to true in case the `foo` package is loaded with the bar option. By default, the attribute language is set (to french or english) when Tralics is able to determine the main language, either because (as in this document), english is an option to `\documentclass`, or because the `babel` package is loaded with recognized parameters. Finally, the meaning of `\setlength` depends on whether the `calc` package is loaded or not.

Sectioning commands The XML model of Tralics is based on the notion of paragraph (`\par` command, `<p>` element). This element can contain inline stuff (text, images, tables, math formulas); it is at the same level as non-inline stuff (images, tables, math, notes, bibliographic entries), and can be contained in a sectioning command (for instance the paragraph is in a subsection in a section in a chapter in a part). These elements have in general a number (computed by an external program rather than defined by the user, never computed by Tralics), and can be referenced. The translation of a sectioning command is a `<div`*i*`>` element, where *i* is an integer between 0 and 6. Example:

```
\section{x}A\label{a}
\subsection{y}
B\label{b}C\label{c}D
\paragraph{z}
\ref{a}\ref{b}
\subsection{t}
```

gives

```
<div0 id='uid1'><head>x</head>
 <p>A</p>
 <div1 id='uid2'><head>y</head>
  <p>BCD</p>
  <div3 id='uid3'><head>z</head>
    <p><ref target='uid1'/>
       <ref target='uid2'/></p>
  </div3>
 </div1>
 <div1 id='uid4'><head>t</head>...
```

By default, `\section` is the top-level division, but chapters are allowed in a report, and parts in a book. The '...' is not part of the output, it just indicates the current position in the XML tree. Note that Tralics is in outer vertical mode, said otherwise, the occurrence of a character will imply the creation of a `<p>` element; in LaTeX, the current mode would depend on the document class. This is one reason why `\ifvmode` is not implemented. The example shows a logical error: there is a `<div3>` element in a `<div1>` element. It would be numbered 1.1.0.1 in LaTeX, and 1.1.1 (without the zeros) using an XSLT processor.

Cross references Tralics implements `\label` and `\ref` but not `\pageref`. The basic idea is to put a mark in the XML document, and use references to this mark. The mark is an attribute of type ID, and there are some limitations in XML that do not exist in LaTeX (for instance no element type may have more than one ID attribute specified, and an ID cannot start with a digit). In order to remove these difficulties, Tralics uses its own list of IDs (`uid1`, `uid2`, etc.). Associated with the mark is a value, produced by `\p@foo\thefoo`. This mechanism is not implemented in Tralics: the value associated to a label is not in the XML document but must be computed by the application (the style sheet for instance). In order to distinguish between Figure 4 or Table 5, the label must be associated with a figure or a table (a footnote, an item in a list, a formula, a division, that's all). Hence the following changes with LaTeX: inside a figure or table environment, there must be at most one label, and it can be before or after the caption. A math formula can have a label only if it is a display math equation, and it is numbered only if it has a label. See example above. There is no need to call Tralics twice or more: a single pass is enough.

References to external documents are understood. In the following example, we switch to French in order to show the behavior of special characters (like colon and underscore) with or without the `\url` command:

```
\language=1
\href{http://foo\_ba}{http://foo_bar}
\href{\url{http://foo_ba}}{http://foo_bar}
```

The translation is

```
<xref url='http://foo_bar'>
  http&nbsp;://foo_ba</xref>
<xref url='http://foo_bar'>http://foo_ba
</xref>
```

Basic formatting tools

The following example is from page 218 of *The TeXbook* [6], with an addition to verify that `\trialdivision` is really called 132 times.

```
\tracingall
\countdef\td 4 \td=0
\newif\ifprime \newif\ifunknown
\newcount\n \newcount\p \newcount\d
\newcount\a
\def\primes#1{2,~3% assume that #1 >= 3
  \n=#1 \advance\n by-2 % n more to go
  \p=5 % odd primes starting with p
  \loop\ifnum\n>0 \printifprime
      \advance\p by2 \repeat}
\def\printp{, %
  \ifnum\n=1 and~\fi %
  \number\p \advance\n by -1 }
\def\printifprime{\testprimality
  \ifprime\printp\fi}
\def\testprimality{{\d=3 \global\primetrue
  \loop\trialdivision
    \ifunknown\advance\d by2 \repeat}}
\def\trialdivision{\a=\p
  \global\advance\td by 1
  \divide\a by\d
  \ifnum\a>\d \unknowntrue\else\unknownfalse\fi
  \multiply\a by\d
  \ifnum\a=\p \global\primefalse
    \unknownfalse\fi}

The first thirty prime numbers are \primes{30}.
trialdivision macro was expanded \the\td\ times
```

This example works in LaTeX and in Tralics. All TeX primitives that start with 'tracing' are implemented, although most of them refer to behavior that is not implemented in Tralics. If you run Tralics on the previous example, the log file will contain:

```
[706] \countdef\td 4 \td=0
{\countdef}
+scanint for \countdef->4
{\td}
+scanint for \td->0
[709] \newcount\a
{\countdef \a=1550}
```

You can see the input source line whenever it is read, with its line number, and the commands that are evaluated. On page 269 of *The TeXbook*, you can see the definition of an integer. This is so complicated that Tralics prints the value whenever scanned. The last line shows that the `\a` command will access the internal tables at position 1550.

```
\iterate->\ifnum \n >0 \printifprime
   \advance  \p by2 \relax \expandafter
   \iterate \fi
+\ifnum
+scanint for \ifnum->28
+scanint for \ifnum->0
+iftest true
```

Here we can see that the `\loop` macro is implemented as in LaTeX and not as in plain TeX. The `\iterate` token is a private one, the `\loop` macro does not kill your command (but the loop inside the loop will modify it, and this explains why extra braces are needed).

```
{begin group character}
+stack: level + 2
```

The first line indicates that Tralics has seen an open brace (technically, a character of catcode one), and that it creates a new frame on the save stack (the outer level is numbered 1, as in TeX).

```
{end group character}
{Text:, 5}
+stack: restoring \ifunknown
+stack: restoring integer value 1550 0
+stack: restoring \iterate
+stack: restoring integer value 1549 0
+stack: level - 2
```

And this is done when Tralics sees the closing brace. It restores two commands and the two registers `\a` and `\d` (the association between `\d` and the number 1549 can be found in the log file). A side effect of seeing the closing brace is to flush the XML buffer (what follows the 'Text:' on the second line), for the case where the current font might change.

List structures Standard LaTeX lists are implemented, but they are not customizable. The only non trivial part is that the optional argument of the `\item` command should be evaluated in a group. As an example:

```
\begin{itemize}
 \item a
 \item [\it b] c
 \begin{enumerate}
   \item d
   \begin{description}
     \item e
   \end{description}
 \end{enumerate}
\end{itemize}
```

This will translate to

```
<list type='simple'>
 <item id='uid14'><p>a</p></item>
 <label><hi rend='it'>b</hi></label>
 <item id='uid15'>
  <p>c</p>
  <list type='ordered'>
   <item id='uid16'><p>d</p>
    <list type='description'>
     <item id='uid17'><p>e</p>
     </item>
    </list>
   </item>
  </list>
 </item>
</list>
```

Here, in order to make the XML output more readable, we added and removed some space and newline tokens. The general rule for Tralics is to output one space (or newline character), whenever TeX would output one space character. In particular, the TeX scanner converts two consecutive newline characters in a space character and a `\par` token. This space character will be output by Tralics as a newline character.

Footnotes The translation of

```
\footnote{a}
\footnote{Y \AddAttToLast{x}{y}b\par
  \AddAttToCurrent*{place}{here}c}
```

is

```
<note id='uid14' place='foot'>a</note>
<note place='here' id='uid15'>
 <p x='y'>Y b</p>
 <p>c</p>
</note>
```

Two remarks: each note has a uid, hence can be referenced, but there is nothing special about it (no counter, no mark, no restrictions).

On the other hand, the example shows how to add an attribute to an XML element, either the element created latest (here the `<p>` element that contains the text of the footnote), or the current element (the footnote, since the translation of `\par` does not create a new `<p>` element, it is the translation of the character c that forces a second `<p>` in the note). In the unlikely event that the element already has an attribute of the same name (for instance, a footnote has a default place attribute), the command is ignored — unless you use its starred form to overwrite.

New elements It is easy to use new elements, just say

```
\begin{xmlelement}{main-elt}
\begin{xmlelement}{sub-elt1}
text1
\end{xmlelement}
\begin{xmlelement}{sub-elt2}
text2
\end{xmlelement}
\AddAttToLast{sb2-att}{value1}
\AddAttToCurrent{foo-att}{att-value''}
\end{xmlelement}
```

and you will get

```
<main-elt foo-att='att-value&apos;&apos;'>
<sub-elt1>
text1
</sub-elt1>
<sub-elt2 sb2-att='value1'>
text2
</sub-elt2>
</main-elt>
```

You can also try

```
\hbox{a\it b}
\vbox{c\it d}
\newcommand\AGtest{AG}
\setbox0=\xbox{myelt}
  {\aftergroup\AGtest e\it f}
\copy0 \copy0
```

This will work in Tralics, since either an `\hbox` or a `\vbox` is just an unnamed `\xbox`. The braces serve for grouping, and as argument delimiters. The result is

```
a<hi rend='it'>b</hi>
c<hi rend='it'>d</hi>
AG
<myelt>e<hi rend='it'>f</hi></myelt>
```

```
<myelt>e<hi rend='it'>f</hi></myelt></p>
```

Note: when Tralics constructs an element, the equivalent of a box in TeX, it is in a special mode that does not match any of TeX's modes.

Verbatim material A document like this one uses lots of verbatim material. Tralics is familiar with standard verbatim environments, and some extensions. For instance, the translation of

```
\DefineShortVerb{\|}
you can say |\foo| or \verb*+foo bar+.
\begin{Verbatim}
verb line1
$<&\>
\end{Verbatim}
```

would be

```
you can say <hi rend='tt'>\foo</hi>
or <hi rend='tt'>foo&blank;bar</hi>.</p>
<p noindent='true'><hi rend='tt'>
   verb&nbsp;line1</hi></p>
<p noindent='true'><hi rend='tt'>
   $&lt;&amp;\&gt;</hi></p>
<p noindent='true'>
```

Note here that the `\end` commands check whether the environment is followed by an empty line. If it is not, a `\noindent` token is inserted.

The layout of the page

The essential idea of Tralics is that the result of the translation is independent of the layout of the page.

But if you give `0.8\textwidth` as the width of a figure, Tralics will replace this by 12cm, because something has to be done. Optional arguments of sectioning commands are ignored: in general one would use them for marking the document. We have implemented the TeX markup commands, but they do nothing; we think of extending the functionalities of the title-page mechanism of Tralics. In fact, if the title-page specifies something like (see opening example):

```
  \address p<address> "no address"
```

then there is a command `\address` that takes one argument, whose translation is put in an `<address>` element, and recorded. The 'p' marker says that paragraphs are allowed, an 'E' marker indicates that the command is an environment, a '+' that the command can be issued more than once, etc. As you can see, there is still work to be done.

Tabular material

There are essentially four points to be considered: tables in math mode (described later), standard LaTeX tabular (see earlier example), the tabbing environment (not implemented, because it implies implementing all typesetting algorithms), and the `\halign` primitive, which is much too complicated to implement.

One problem is how to translate a table specification such as `r||l`. Tralics understands that there are two columns, right and left justified, with a rule on the right and the left, but this is impossible to translate into HTML, that knows only `|r|l|` or `rl`. The `\hline` and `\cline` are implemented, but suffer from the same limitations.

Mastering floats

There is in general enough flexibility for adding or removing one line on the current page, so that, for instance, TeX is not faced with the dilemma of either putting a section title at the bottom of a page, or generating a horribly underfull page. The situation is quite different for tables or images, which can be much larger; this is the reason why they can float (i.e. items are put in the output in a different order). Handling of floats is non-trivial; in some cases, the best thing to do is to resize (the image, or even the text).

The philosophy of Tralics is the following: in the case where the XML is translated into HTML, no object has to float. On the other hand, since the author of the document is not the person that does the final typesetting, no fine tuning of floats can be achieved. As a result, Tralics just ignores optional arguments of float environments, float parameters, and things like `wrapfig` environments.

Font selection

The introductory example shows some features of Tralics. For instance, the default input encoding of the document is latin1, and this is the same as the output encoding: the translation of `é\oe` will be `é&oelig;`, you can also say `\'e\.E` if you want éĖ; all Unicode characters with code less than x180 are recognized. If you need other characters, you must use a construct like

```
Hàn Th\xmllatex{&\#x1ebf;}{\'{\^e}} Thành
```

The command `\xmllatex` takes two arguments, the first one is ignored by standard LaTeX, and the second one by Tralics. Note how the sharp sign is protected. See the Unicode manual [9] for the numeric value `x1ebf`. There is no way to construct a logo, like the TeX logo, with `\kern` or `\lower`, but instead, you could say

```
\newcommand\MyLogo{\xmlemptyelt{MyLogo}}
```

and define a `<MyLogo/>` element in your DTD. You could also use a parameterized version of the logo, like this

```
\newcommand\ParLogo[1]{\xmlelt{PLogo}{#1}}
\MyLogo
\ParLogo{2$\varepsilon$}
```

The resulting XML document will contain

```
<MyLogo/>
<PLogo>2<formula type='inline'><math>
<mi>&epsiv;</mi></math></formula></PLogo>
```

As explained above, Tralics knows of the `\font` command, but this command should not be used. It is much better to say something like

```
{\bfseries a\itshape b\small c\ttfamily d}
```

which translates into:

```
<hi rend='bold'>a</hi>
<hi rend='it'><hi rend='bold'>b</hi></hi>
<hi rend='small'><hi rend='it'>
  <hi rend='bold'>c</hi></hi></hi>
<hi rend='small'><hi rend='it'>
  <hi rend='tt'><hi rend='bold'>d</hi>
</hi></hi></hi>
```

As this example shows, a font is defined by a size, shape, series and family, nothing more (is it possible to convert to HTML a document written in Computer Modern Funny Roman using the U encoding?). Tralics understands all ten standard font sizes, but uses only three (a normal size, a larger one, and a smaller one); there are some bugs in the implementation. In an example like this

```
{\it a\par b}
```

a new paragraph is created when the b character is sensed, and a font element must be inserted in this paragraph, since the current font is not the default one. The commands `\it`, `\textit`, `\itshape` have the same meaning as in LaTeX. The `\em` command also (but Tralics does not know if the current font is slanted or not, and may generate the wrong result).

Higher mathematics

TeX has a reputation of producing very high quality mathematics, and people at the AMS have worked hard to make it easy to use. As a result, trying to translate the entire AMS functionality to MathML is nearly impossible (too many features have no equivalent). Even converting the MathML into pdf was a challenge (there are still unresolved problem, like tables in tables, missing characters, etc.), and the rendering by tools like Amaya is a bit strange.

The translation of $\int_0^\infty \mathcal{F}(x)^2\,\mathrm{d}x$ is unsatisfactory to us. In particular, Tralics fails to notice that the exponent applies to the '$\mathcal{F}(x)$', and we wonder whether an expert system should be used (consider the scope of the `<mrow>` element in the translation below).

```
<math>
 <mrow>
  <msubsup><mo>&int;</mo>
    <mn>0</mn> <mi>&infin;</mi>
  </msubsup>
  <mrow><mi>&Fscr;</mi><mo>(</mo>
   <mi>x</mi> </mrow>
  <msup><mo>)</mo><mn>2</mn> </msup>
  <mspace width='3pt'/><mi> d </mi>
  <mi>x</mi>
</mrow></math>
```

LaTeX in a multilingual environment

Inria's Rapport d'Activité is, by nature and law, a French document. The RA, or the raweb, being just its scientific annex, can be in English (with a French summary). It was decided that starting in year 2003, the whole thing (the source document, the web site, the logo, the these-pages-in-French-only pointers) will be in English. Thus, special attention was given to this problem in Tralics. For instance, whether the title of the bibliography should be "References" or "Références" does not depend on Tralics, but on the style sheet (or the DTD). On the other hand, one may imagine a document (like this one) that has an abstract in two languages, and that the first abstract should be in the main language. Using the title page mechanism of Tralics will give you a fixed order for the XML result, but as the introductory example shows, the main language is an attribute of the root element (and it is the `\maketitle` command that selects the current language as main language).

In fact, TeX provides a `\setlanguage` command, which is ignored by Tralics, and a `\language` command that selects a language: Tralics assumes that English has number zero, French has number one, and your favorite language has number two (in fact, only two languages are really supported). The BibTeX translator (see below) uses the value of the current language for the interpretation of strings like 'jan'. The following example shows the differences between French and English:

```
\language =1
<<guill'' ponctuation;
\verb+<<guill'' ponctuation;+
a\xspace b\xspace !
\language=0
<<guill'' ponctuation;
a\xspace b\xspace !
```

The translation is

```
«&nbsp;guill&nbsp;» ponctuation&nbsp;;
<hi rend='tt'>&lt;&lt;guill
'&ZeroWidthSpace;'&ZeroWidthSpace;&nbsp;
ponctuation;</hi>
a b&nbsp;!
«&nbsp;guill'' ponctuation;
a b!
```

The important points are the following: a space is added before some punctuation characters (colon, semi-colon, guillemets, etc.), even when the `xspace` package (which is language independent) thinks it useless. Everything that looks like opening or closing double quotes is converted to French guillemets, with the proper spacing. Note the translation of the `\verb` command: the two single quotes are followed by a special (invisible) character, the purpose of which is to avoid ligatures in the case where the XML is processed by TeX; the line breaks were manually added in order to avoid overfull lines.

Portable graphics in LaTeX

We tried to implement some commands, for instance

```
\setlength{\unitlength}{.0075\textwidth}
\begin{picture}(90,50)
\put(40,25){\framebox(10,10){$H(s)$}}
\put(19,30){\vector(1,0){9}}
\put(60,15){\vector(-1,0){10}}
\put(17.75,0.5){\oval(1.5,1.5)[r]}
\end{picture}
```

The result is given below. However, for the RA, we copied the content of the picture environment in a file, called LaTeX on it, and replaced the picture by a reference to the PostScript file.

```
<picture width='90.0pt' height='50.0pt'>
<put xpos='40.0pt' ypos='25.0pt'>
 <box width='10.0pt' height='10.0pt'>
 <formula type='inline'><math>
  <mrow><mi>H</mi><mo>(</mo><mi>s</mi>
  <mo>)</mo></mrow></math>
 </formula></box></put>
<put xpos='19.0pt' ypos='30.0pt'>
 <vector xdir='1.0pt' ydir='0.0pt'
  width='9.0pt'/></put>
<put xpos='60.0pt' ypos='15.0pt'>
  <vector xdir='-1.0pt' ydir='0.0pt
   width='10.0pt'/></put>
<put xpos='17.75pt' ypos='0.5pt'>
   <oval xpos='1.5pt' ypos='1.5pt'
  specs='r'/></put>
</picture>
```

Using PostScript

Since our main application uses pdf in preference to PostScript, we will not speak about PostScript here (some people use `psfrags` in order to replace PostScript fonts in their figures by standard LaTeX ones; this is strange).

Index generation

Tralics offers no specific tool.

Bibliography generation

The design of the bibliography of the raweb is still a subject of research. We were faced with the following problems: first, the raweb uses three different BibTeX files, and the translation of one of these files depends on whether it was put on the web or printed on paper, and the items in the main bibliography file are grouped in different categories, depending on their type; a total of four bst files are used. One idea was to parse the bbl files (but, if you look at the sources of the `footbib` package, you can see how hard it is; by the way, bibliographic entries are no longer footnotes). The second idea was to modify the bst file in order to generate an environment per item, in order to simplify the parsing. The third idea was to abandon BibTeX; since there is still no XML standard for bibliographies, and no universal tools, we just wrote a BibTeX to LaTeX translator. For instance, the TeXbook entry looks like this

```
\citation{Knu84}{cite:texbook}{year}{book}
\bauteurs{\bpers{D. E.}{}{Knuth}{}}
\cititem{btitle}{The \TeX book}
\cititem{bpublisher}{Addison Wesley}
\cititem{byear}{1984}
\endcitation
```

This part of the translator reads the BibTeX file, expands the BibTeX macros (predefined, or used defined), removes useless stuff, sorts the entries, and returns the LaTeX equivalent (in fact, the stuff is written in a bbl file, but only for you to see what happens in case of error). The same entry processed by BibTeX:

```
\bibitem[\protect\citeauthoryear{Knuth}
{Knuth}{1984}]{texbook}
Knuth, D.~E.
\newblock {\em The \TeX book}.
\newblock Addison Wesley, 1984
\UseExtraLabel{}.
```

A comparison shows that BibTeX adds some tokens that would be very hard to remove: tilda, comma, period, together with some others (`\em`, `\newblock`) that can be context sensitive. On the other hand, the LaTeX to XML translator generates

```
<citation from='year' key='Knu84'
   id='cite:texbook' type='book'>
 <bauteurs>
   <bpers prenom='D. E.' nom='Knuth'/>
 </bauteurs>
 <btitle>The <TeX/>book</btitle>
 <bpublisher>Addison Wesley</bpublisher>
 <byear>1984</byear>
</citation>
```

so that it is up to the style sheet to do all the real work (using `\em` for the title if it's a book, use the plural of 'editor' if there is more than one, etc.). Each style has its own peculiarities: the `tugboat` style generates a special optional argument for `\bibitem`, but the raweb tools need the `from` and `type` attributes for sorting (problem here: citations are already sorted).

Conclusion

We have shown in this paper that a LaTeX to XML translator can be very useful, when you have the tools to manage the resulting XML. We hope that standardization will continue, so that it will become easier to write such tools, and adapt them. Putting high quality mathematics on the web is still a challenge, and we hope that others will appreciate our contribution.

References

[1] David Carlisle, Michel Goossens, and Sebastian Rahtz. De XML à PDF avec `xmltex` et PassiveTeX. In *Cahiers Gutenberg*, number 35-36, pages 79–114, 2000.

[2] David Carlisle, Patrick Ion, Robert Miner, and Nico Poppelier. Mathematical Markup Language (MathML) version 2.0. `http://www.w3.org/TR/MathML2/`, 2001.

[3] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The LaTeX Companion*. Addison Wesley, 1993.

[4] José Grimm. Outils pour la manipulation du rapport d'activité. Technical Report RT-0265, Inria, 2002. `http://www.inria.fr/rrrt/rt-0265.html`.

[5] Yannis Haralambous and John Plaice. Produire du MathML et autres ...ML à partir d'Ω : Ω se généralise. In *Cahiers Gutenberg*, number 33-34, pages 173–182, 1999.

[6] Donald E. Knuth. *The TeXbook*. Addison Wesley, 1984.

[7] Philippe Louarn. Une expérience d'utilisation de LaTeX : le Rapport d'activité de l'INRIA. *Cahiers Gutenberg*, (0):17–24, apr 1988.

[8] Sebastian Rahtz. Passive TeX. `http://www.tei-c.org.uk/Software/passivetex/`, 2003.

[9] The Unicode Consortium. *The Unicode Standard, version* 3.0. Addison Wesley, 2000.

[10] Norman Walsh and Leonard Muellner. *Docbook: The Definitive Guide*. O'Reilly & Associates, Inc, 1999.