MlBibTeX: Beyond LaTeX

Jean-Michel Hufflen
LIFC (FRE CNRS 2661)
University of Franche-Comté
16, route de Gray
25030 Besançon Cedex
France
hufflen@lifc.univ-fcomte.fr
http://lifc.univ-fcomte.fr/~hufflen

Abstract

This article sums up our experience with MlBibTeX, our multilingual implementation of BibTeX, and points out some possible improvements for better cooperation between LaTeX and MlBibTeX. Also, MlBibTeX may be used to generate bibliographies written according to other formalisms, especially formalisms related to XML, and we give some ways to ease that.
Keywords: Bibliographies, multilingual features, BibTeX, MlBibTeX, bst, nbst, XML, XSLT, XSL-FO, DocBook.

1 Introduction

MlBibTeX (for '**M**ulti**L**ingual BibTeX') is a reimplementation of BibTeX [21], the bibliography processor associated with LaTeX [19]. The project began in October 2000, and has resulted in two experimental versions [9, 11] and the present version (1.3), that will be available publicly by the time this article appears. As we explained in [15], a prototype using the Scheme programming language is working whilst we are developing a more robust program written in C. The prototype has allowed us to get some experience with real-sized bibliographies: this is the purpose of the first part of this article, after a short review of the *modus operandi* of MlBibTeX.

MlBibTeX's present version no longer uses the bst language of BibTeX for bibliography styles [20]. Such .bst files were used in MlBibTeX's first version, but since this old-fashioned language, based on simple stack manipulations, is not modular, we quickly realised that this choice would have led us to styles that were too complicated [12]. Thus, Version 1.3 uses the nbst (for '**N**ew **B**ibliography **ST**yles') language, described in [13] and similar to XSLT,[1] the language of transformations designed for XML texts [32]. More precisely, MlBibTeX 1.3 uses XML[2] as a central formalism in the sense that parsing files containing *bibliographical entries* (.bib files) results in a DOM[3] tree. Bibliography styles written using nbst are XML texts, too.

Of course, nbst can be used to generate bibliographies for documents other than those processed with LaTeX.[4] In particular, nbst eases the generation of bibliographies for documents written using XML-like syntax. Nevertheless, dealing with .bib files raises some problems: we go into them thoroughly in Section 4.

Reading this article requires only a basic knowledge of LaTeX, BibTeX and XML. Some examples given in the next section will use the commands provided by the multilingual babel package of LaTeX 2_ε [2]. Other examples given in Section 4 will use the Scheme programming language, but if need be, referring to an introductory book such as [28] is sufficient to understand them.

2 Architecture of MlBibTeX

2.1 How MlBibTeX Works

As a simple example of using MlBibTeX with LaTeX, let us consider the `silke1988` bibliographical entry given in Figure 1. As we explain in [15], the sequence '`[...] !` ⟨*idf*⟩' is one of the multilingual features

[1] **EX**tensible **S**tylesheet **L**anguage **T**ransformations.

[2] **EX**tensible **M**arkup **L**anguage. A good introduction to this formalism issued by the W3C (**W**orld **W**ide **W**eb **C**onsortium) is [24].

[3] **D**ocument **O**bject **M**odel. This is a W3C recommendation for a standard tree-based programming approach [24, p. 306–308], very often used to implement XML trees.

[4] This is also the case with the bst language of BibTeX, but in practice, it seems that this feature has not been used, except for documents written in SCRIBE [25], a predecessor of LaTeX.

```
@BOOK{silke1988,
        AUTHOR = {James~R. Silke},
        TITLE = {Prisoner of the Horned Helmet},
        PUBLISHER = {Grafton Books},
        YEAR = 1988,
        NUMBER = 1,
        SERIES = {Frank Frazetta's Death Dealer},
        NOTE = {[Pas de traduction fran\c{c}aise
                 connue] ! french
                [Keine deutsche Übersetzung]
                 ! german},
        LANGUAGE = english}
```

Figure 1: Example of a bibliographical entry in MlBibTeX.

provided by MlBibTeX, defining a string to be included when the language of a corresponding reference, appearing within a bibliography, is *`idf`*. So if this entry is cited throughout a document written in French and the 'References' section is also written in French, it will appear as:

> [1] James R. Silke : *Prisoner of the Horned Helmet.* N° 1 *in Frank Frazetta's Death Dealer.* Grafton Books, 1988. Pas de traduction française connue.

Here and in the bibliography of this article, we use a 'plain' style, that is, references are labelled with numbers. More precisely, the source processed by LaTeX, included into the .bbl file generated by MlBibTeX, is:

```
\begin{thebibliography}{...}
...
\bibitem{silke1988}
\begin{otherlanguage*}{english}
James~R. \textsc{Silke}: \emph{Prisoner of the
Horned Helmet}.
\foreignlanguage{french}{\bblno~1 \bblof}
\emph{Frank Frazetta's Death Dealer}. Grafton
Books, 1988. \foreignlanguage{french}{Pas de
traduction fran\c{c}aise connue}.
\end{otherlanguage*}
...
\end{thebibliography}
```

Let us examine this source text. We can notice the use of additional LaTeX commands to put some keywords ('`\bblin`' for '*in*', '`\bblno`' for 'N°', that is, 'number' in French). In [14], we explain how to put them into action within LaTeX and how MlBibTeX uses them. This source also shows how English words, originating from an entry in English (see the value of the `LANGUAGE` field in Figure 1), are processed. If the document uses the babel package, and if the french option of this package is selected, we use the `\foreignlanguage` command of this package [2], as shown above. Users do not have to select its english option; if it is not active, the source text generated by MlBibTeX looks like:

```
\bibitem{silke1988}James~R. \textsc{Silke}:
\emph{Prisoner of the Horned Helmet}. \bblno~1
\bblof\ \emph{Frank Frazetta's Death Dealer}.
Grafton Books, 1988. Pas de traduction
fran\c{c}aise connue.
```

but the English words belonging to this reference will be taken as French by LaTeX and thus may be processed or hyphenated incorrectly.

```
<book id="silke1988" language="english">
  <author>
    <name>
      <personname>
        <first>James&#160;R.</first>
        <last>Silke</last>
      </personname>
    </name>
  </author>
  <title>Prisoner of the Horned Helmet</title>
  <publisher>Grafton Books</publisher>
  <year>1988</year>
  <number>1</number>
  <series>
    Frank Frazetta's Death Dealer
  </series>
  <note>
    <group language="french">
      Pas de traduction française connue
    </group>
    <group language="german">
      Keine deutsche Übersetzung
    </group>
  </note>
</book>
```

Figure 2: The XML tree corresponding to the entry of Figure 1.

2.2 The Modules of MlBibTeX

As mentioned in the introduction, parsing a .bib file results in a DOM tree. In fact, .bib files are processed as if they were XML trees, but without whitespace nodes.[5] Following this approach, the entry `silke1988` given in Figure 1 is viewed as the tree of Figure 2, except that the whitespace nodes that an XML parser would produce are excluded.

[5] These are text nodes whose contents are only whitespace characters, originating from what has been typed between two tags [27, p. 25–26]. For example, if the XML text of Figure 2 is parsed, there is a whitespace node, containing a newline and four space characters between the opening tags `<author>` and `<name>`. XML parsers are bound by the 'all text counts' constraint included in the XML specification [33, § 2.10], and cannot ignore such whitespace characters.

We can see that some LaTeX commands and special characters are converted according to the conventions of XML.

- The commands used for accents and special letters are replaced by the letter itself. This poses no problem since DOM trees are encoded in Unicode [29]. As an example, the '`\c{c}`' sequence in the value of the `NOTE` field in Figure 1 is replaced by 'ç' in Figure 2. (By the way, let us remark that MlBibTeX can handle the 8-bit latin1 encoding:[6] notice the 'Ü' character inside this value.)
- Likewise, the commands:
  - '`\␣`' for a simple space character,
  - '`\\`' for an end-of-line character,

  and the sequences of characters:
  - '`~`', for an unbreakable space character,
  - '`--`', and '`---`' for dash characters,

  are replaced by the corresponding Unicode values for these characters:[7]

  `&#32;   &#10;   &#160;   &#8211;   &#8212;`

  An example is given by the value of the `AUTHOR` field, see Figures 1 & 2.
- Some characters escaped in LaTeX (for example, '`$`', '`%`', '`&`') lose the escape character:

  $$\texttt{\textbackslash\%} \Longrightarrow \texttt{\%}$$

  The escape is restored if MlBibTeX generates a .bbl file to be processed by LaTeX. Other characters are replaced by a reference to a character entity:[8]

  $$\texttt{\textbackslash\&} \Longrightarrow \texttt{\&amp;} \quad \texttt{<} \Longrightarrow \texttt{\&lt;} \quad \texttt{>} \Longrightarrow \texttt{\&gt;}$$
- Balanced delimiters for quotations ("`` ` ``" and "`'`" or '`` `` ``' and '`''`') are replaced by an `emph` element:[9]

  ```
  `Tooth and Claw' ⟹
    <emph emf='no' quotedf='yes'>
      Tooth and Claw
    </emph>
  ```

  If '`'`' or '`"`' characters are unbalanced, they are replaced by references to character entities used in XML documents:

  $$\texttt{'} \Longrightarrow \texttt{\&apos;} \qquad \texttt{"} \Longrightarrow \texttt{\&quot;}$$

[6] See [7, Table C.4] for more details.

[7] That was not the case in earlier versions; for instance, [12, Figure 3] improperly includes a tilde character in a text node. This bug was fixed at the end of 2003.

[8] See [24, p. 48] for more details.

[9] '`emph`' is of course for 'emphasise': all the attributes (for example, '`quotedf`' for 'quoted-flag', used for specifying a quotation) default to `no`, except `emf`, which defaults to `yes`. The complete specification is given in [10].

Such an XML tree, resulting from our parser, may be validated using a DTD;[10] more precisely, by a revised version of the DTD sketched in [10].

Some examples of using nbst for bibliography styles are given in [12, 13, 14]. We give another example in Figure 3. We can see that this language is close to XSLT and it uses *path expressions* as in the XPath language [31]. Also, the example shows how multilingual features (for example, the sequence '`[...] ! ...`') are processed: we use some external functions in order to determine which LaTeX command can be used to switch to another language. These external functions are written using the language of MlBibTeX's implementation: Scheme for the prototype, C for the final program.

3 MlBibTeX with LaTeX

When BibTeX generates a .bbl file, it does not use the source file processed by LaTeX, but only the auxiliary (.aux) file, in which the definition of all the labels provided by the commands `\label` and `\bibitem` is stored. This file also contains the name of the bibliography style to be used and the paths of bibliography data bases to be searched, so BibTeX need not look at any other file.

This is not true for MlBibTeX. It still uses the .aux file as far as possible, but it also has to determine which multilingual packages are used: first of all babel, but also some packages devoted to particular languages: french [6], german [23], ... So we have to do a *partial parsing* of the .tex file for that. For better co-operation between LaTeX and MlBibTeX, this could be improved, in that information about multilingual packages used, and languages available, could be put in the .aux file. In fact, the external functions of our new bibliography styles are only used to manage information extracted from a .tex file. Expressing such operations using nbst would be tedious.

Another improvement regarding the natural languages known by LaTeX would be a connection between:

a) the language codes used in XML, specified by means of a two-letter language abbreviation, optionally followed by a two-letter country code [1] (for example, '`de`' for '*deutsch*' ('German'), '`en-UK`', '`en-US`, etc.)'; and
b) the *resources* usable to write texts in these languages.

For example, a default framework could be the use of the babel package, and '`de`' would get access to

[10] **D**ocument **T**ype **D**efinition. A DTD defines a document markup model [24, Ch.5].

```
<nbst:template match="group">
  <nbst:if test="@language=$language">
    <!--  The $language variable is set to the current language.      -->
    <nbst:value-of select="call(language_open_change,@language)"/>
    <!--  If the babel package is used and a known option has been selected, this external function
          writes the \foreignlanguage command...
      -->
    <nbst:apply-templates use-language="@language"/>
    <nbst:value-of select="call(language_close_change,@language)"/>
    <!--  ... and this external function puts a closing brace.      -->
  <nbst:if>
</nbst:template>
```

Figure 3: Example of calling an external function.

the german option of this package, although it could be redefined to use the *ad hoc* package name german. In the future, such a framework would allow us to homogenise all the notations for natural languages to those of XML. In addition, let us notice that ConTEXt[11] [8], already uses these two-letter codes in its `\selectlanguage` command.

And last but not least, auxiliary files should include information about the encoding used in the source text. As can be seen in the examples of Section 2.1, accented letters are replaced by the commands used to produce them in LaTeX, even though LaTeX can of course handle 8-bit encodings (provided that the inputenc package is loaded with the right option). This is to avoid encoding problems. In addition, such information would ease the processing of languages written using non-Latin alphabets.

4 Towards the XML World

Since a .bib file can be processed as an XML tree by a bibliography style written in nbst, MlBibTEX opens a window on XML's world. A converter from .bib files to a file written using HTML,[12] the language of Web pages, becomes easy to write. So does a tool to write a bibliography as an XSL-FO[13] document [34]. More precisely, we give in Figure 4 an example of using the root element of nbst. Possible values for the `method` of the `nbst:output` element are:

[11] TEX, defined by Donald E. Knuth [18], provides a general framework to format texts. To be fit for use, the definitions of this framework need to be organised in a *format*. Two such formats are plain TEX and LaTeX, and another is ConTEXt, created by Hans Hagen.

[12] **H**yper**T**ext **M**arkup **L**anguage.

[13] E**X**tensible **S**tylesheet **L**anguage—**F**ormatting **O**bjects: this language aims to describe high-quality print outputs. Such documents can be processed by the shell command `xmltex` (resp. the shell command `pdfxmltex`) from PassiveTEX [22, p. 180] to get .dvi files (resp. .pdf files).

```
<nbst:bst version="1.3" id="plain" xmlns:nbst=
 "http://lifc.univ-fcomte.fr/~hufflen/mlbibtex"
 >
  <nbst:output method="LaTeX"/>
  ...
</nbst:bst>
```

Figure 4: Root element for a bibliography style written using nbst.

```
LaTeX   xml   html   text
```

Nevertheless, this approach has an important limitation in practice. Since BibTEX has traditionally been used to generate files suitable for LaTeX, users often put LaTeX commands inside values of BibTEX fields.[14] For example:

```
ORGANIZATION = {\textsc{tug}}
```

In such a case, we would have to write a mini-LaTeX program (or perhaps a new output mode for LaTeX) that would transform such a value into a string suitable for an XML parser.

The problem is more complicated when commands are defined by end-users. For instance:

```
ORGANIZATION = {\logo{tug}}
```

works with BibTEX—or MlBibTEX when we use it for generating LaTeX output—even though `\logo` has an arbitrary definition; for example,

```
\newcommand{\logo}[1]{\textsc{#1}}
```

according to LaTeX's conventions, or:

```
\def\logo#1{\textsc{#1}}
```

if a style close to plain TEX is used. Likewise, such commands can be known when an output file from MlBibTEX is processed by ConTEXt.

[14] The author personally confesses to using many `\foreignlanguage` commands within the values of BibTEX fields, before deciding to develop MlBibTEX.

```
<bibliography>

  <title>References</title>

  <biblioentry>
    <abbrev>silke1989</abbrev>
    <authorgroup>
      <author>
        <firstname>James R.</firstname>
        <surname>Silke</surname>
      </author>
    </authorgroup>
    <copyright><year>1989</year></copyright>
    <isbn>0-586-07018-4</isbn>
    <publisher>
      <publishername>
        Grafton Books
      </publishername>
    </publisher>
    <title>Lords of Destruction</title>
    <seriesinfo>
      <title>
        <othercredit>
          <firstname>Frank</firstname>
          <surname>Frazetta</surname>
        </othercredit>'s Death Dealer
      </title>
      <volumenum>2</volumenum>
    </seriesinfo>
  </biblioentry>

</bibliography>
```

Figure 5: The bibliographical reference from Figure 1 expressed in DocBook. Note the *ad hoc* tag `<othercredit>`.

Moreover, let us consider the bibliographical reference given in Figure 5, according to the conventions of DocBook, a system for writing structured documents [36] (we use the conventions of the XML version of DocBook, described in [26]). We can see that some information is more precise than that provided in Figure 1. But there are still complexities: the person name given in the value of the `SERIES` field is surrounded by an *ad hoc* element in the DocBook version.

If we want to take advantage of the expressive power of DocBook, we can:

- directly process an XML file for bibliographical entries. In this case, our DTD should be extended; that is possible, but we still need a solution to process the huge number of existing .bib files;
- introduce some new syntax inside .bib files, that might be complicated and thus perhaps unused in practice,
- introduce new LaTeX commands, to process like the `\logo` example mentioned above.

We have experimentally gone quite far in the third direction, which also allows to us to deal with the LaTeX commands already in .bib files. In Figure 6, we give some examples of such processing, as implemented in the prototype.[15]

As can be seen, we have defined a new function in Scheme, named `define-pattern`, with two formal arguments. The first is a string viewed as a *pattern*, following the conventions of TeX for defining commands, that is, the arguments of a command are denoted by '`#1`', '`#2`', ... (cf. [18, Ch. 20]). The second argument may also be a string, in which case it specifies a replacement. The arguments of the corresponding command are processed recursively. In case of conflict among patterns, the longest is chosen. So, the pattern `"\\logo{#1}"`[16] takes precedence over the pattern `"{#1}"`.

If the second argument of the `define-pattern` function is not a string, it must be a zero-argument function that results in a string. In this case, all the operations must be specified explicitly, using the following functions we wrote:

`pattern-matches?` returns a *true* value if its first argument matches the second, a *false* value otherwise;

`pattern-process` recursively processes its only argument, after replacing sub-patterns by corresponding values;[17]

`pattern-replace` replaces the sub-patterns of its argument by corresponding values; these value are not processed, just replaced *verbatim*.

Whether given directly as the second argument to `define-pattern` or resulting from applying a zero-argument function, the string must be well-formed w.r.t. XML's conventions, that is, tags must be balanced, attributes must be well-formed, etc. In other words, such a string must be acceptable to an XML parser: in our case, the parser is SSAX[18] [17].

The examples given in Figure 6 allow us to see that we can deal with simple commands, like:

`\logo{...}` $\Longrightarrow$ `<emph ...>...</emph>`

[15] This feature has not yet been implemented in the final version.

[16] Let us recall that in Scheme, the backslash character ('\') is used to escape special characters in string constants. To include it within a string, it must itself be escaped.

[17] In fact, using a string *s* as a second argument of `define-pattern` yields the evaluation of the expression `(lambda () (pattern-process` *s*`))`.

[18] **S**cheme implementation of SAX. 'SAX' is for '**S**imple API (**A**pplication **P**rogramming **I**nterface) for XML': this name denotes a kind of parser, see [24, p. 290–292].

```
(define-pattern "{#1}"
  ;; The asitis element is used for words that should never be uncapitalised, that is, proper names. In BibTEX,
  ;; we specify such behaviour by surrounding words by additional braces.
  "<asitis>#1</asitis>")

(define-pattern "\\logo{#1}" "<emph emf='no' scf='yes'>#1</emph>")

(define-pattern "\\foreignlanguage{#1}{#2}"
  "<foreigngroup language='#1'>#2</foreigngroup>")

(define-pattern "\\iflanguage{#1}{#2}{#3}"
  (lambda ()   ; Zero-argument function.
    (string-append ; Concatenation of strings.
                   "<nonemptyinformation>"
                   "<group language='" (pattern-replace "#1") "'>" (pattern-process "#2")
                   "</group>"
                   (let loop ((pattern (pattern-replace "#3")))
                     ;; This form—named let (cf. [28, Exercise 14.8])—defines an internal function loop and
                     ;; launches its first call:
                     (if (pattern-matches? "\\iflanguage{#4}{#5}{#6}" pattern)
                         (string-append "<group language='" (pattern-replace "#4") "'>"
                                        (pattern-process "#5")
                                        "</group>"
                                        ;; The internal function loop is recursively called with a new value:
                                        (loop (pattern-replace "#6")))
                         (string-append "<group>" (pattern-process pattern) "</group>")))
                   "</nonemptyinformation>")))
```

Figure 6: Patterns for some LaTeX commands in Scheme.

as well as more complicated cases, like a cascade of `\iflanguage` commands [2]:

```
\iflanguage{...}{...}{%
 \iflanguage{...}{...}{  ...  }}
```

which becomes:

```
<nonemptyinformation>
  <group language='...'>...</group>
  <group language='...'>...</group>
  ...
</nonemptyinformation>
```

The `nonemptyinformation` element is used for information that must be output, possibly in a default language if no translation into the current language is available.

What we do by means of our `define-pattern` function is like the additional procedures in Perl[19] that the converter `LaTeX2HTML` [4] can use to translate additional commands.

5 Conclusion

Managing several formalisms can be tedious. This fact was one of main elements in XML's design: giving a central formalism, able to be used for representing trees, and allowing many tools using different formalisms to communicate.

[19] **P**ractical **E**xtraction and **R**eport **L**anguage.

BibTEX deals with three formalisms: .aux files, .bib files and .bst files. As Jonathan Fine notes in [5], the applications devoted to a particular formalism cannot be shared with other applications. MlBibTEX attempts to use XML as far as possible, although there is still much to do. For example, defining a syntax for the entries for which we are looking, when using MlBibTEX to generate XSL-FO or DocBook documents. (For our tests, this list of entry names is simply given on the command line).

The next step will probably be a more intensive use of XML, that is, the direct writing of bibliographical entries using XML conventions. For this, we need something more powerful than DTDs, with a richer type structure, namely, *schemas*.[20] In addition, we should be able to easily add new fields to bibliographical entries: the example given using DocBook shows that additional information must be able to be supplied to take advantage of the expressive power of this system. But such additions are

[20] Schemas have more expressive power than DTDs, because they allow users to define types precisely, which in turn makes for a better validation of an XML text. In addition, this approach is more homogeneous since schemas are XML texts, whereas DTDs are not.

There are currently four ways to specify schemas: Relax NG [3], Schematron [16], Examplotron [30], XML Schema [35]. At present, it seems to us that XML Schema is the most suitable for describing bibliographical entries.

difficult to model with DTDs.[21] We are presently going thoroughly into replacing our DTD by a schema; when this work reaches maturity, bibliographical entries using XML syntax could be directly validated using schemas.

6 Acknowledgements

Many thanks to Karl Berry for his patience while waiting for this article. In addition he proofread a first version and gave me many constructive criticisms. Thanks also to Barbara Beeton.

References

[1] Harald Tveit ALVESTRAND: *Request for Comments: 1766. Tags for the Identification of Languages.* UNINETT, Network Working Group. March 1995. `http://www.cis.ohio-state.edu/cgi-bin/rfc/rfc1766.html`.

[2] Joannes BRAAMS: *Babel, a Multilingual Package for Use with LATEX's Standard Document Classes.* Version 3.7. May 2002. `CTAN:macros/latex/required/babel/babel.dvi`.

[3] James CLARK *et al.*: *Relax NG.* `http://www.oasis-open.org/committees/relax-ng/`. 2002.

[4] Nicos DRAKOS: *The LATEX2HTML Translator.* March 1999. Computer Based Learning Unit, University of Leeds.

[5] Jonathan FINE: "TEX as a Callable Function". In: *EuroTEX 2002*, (pp. 26–30). Bachotek, Poland. April 2002.

[6] Bernard GAULLE : *Notice d'utilisation du style french multilingue pour LATEX.* Version pro V5.01. Janvier 2001. `CTAN:loria/language/french/pro/french/ALIRE.pdf`.

[7] Michel GOOSSENS, Sebastian RAHTZ and Frank MITTELBACH: *The LATEX Graphics Companion. Illustrating Documents with TEX and PostScript.* Addison-Wesley Publishing Company, Reading, Massachusetts. March 1997.

[8] Hans HAGEN: *ConTEXt, the Manual.* November 2001. `http://www.pragma-ade.com`.

[9] Jean-Michel HUFFLEN: "MlBibTEX: A New Implementation of BibTEX". In: *EuroTEX 2001*, (pp. 74–94). Kerkrade, The Netherlands. September 2001.

[10] Jean-Michel HUFFLEN: "Multilingual Features for Bibliography Programs: From XML to MlBibTEX". In: *EuroTEX 2002*, (pp. 46–59). Bachotek, Poland. April 2002.

[11] Jean-Michel HUFFLEN: *Towards MlBIBTEX's Versions 1.2 & 1.3.* MaTEX Conference. Budapest, Hungary. November 2002.

[12] Jean-Michel HUFFLEN: "European Bibliography Styles and MlBibTEX". EuroTEX 2003, Brest, France. June 2003. (To appear in *TUGboat.*)

[13] Jean-Michel HUFFLEN: *MlBIBTEX's Version 1.3.* TUG 2003, Outrigger Waikoloa Beach Resort, Hawaii. July 2003.

[14] Jean-Michel HUFFLEN: "Making MlBibTEX Fit for a Particular Language. Example of the Polish Language". *Biuletyn GUST.* Forthcoming. Presented at the BachoTEX 2003 conference. 2004.

[15] Jean-Michel HUFFLEN: "A Tour around MlBibTEX and Its Implementation(s)". *Biuletyn GUST*, Vol. 20, pp. 21–28. In *Proc. BachoTEX Conference.* April 2004.

[16] ISO/IEC 19757: *The Schematron. An XML Structure Validation Language Using Patterns in Trees.* `http://www.ascc.net/xml/resource/schematron/schematron.html`. June 2003.

[17] Oleg KISELYOV: "A Better XML Parser through Functional Programming". In: *4th International Symposium on Practical Aspects of Declarative Languages*, Vol. 2257 of *Lecture Notes in Computer Science.* Springer-Verlag. 2002.

[18] Donald Ervin KNUTH: *Computers & Typesetting. Vol. A: The TEXbook.* Addison-Wesley Publishing Company, Reading, Massachusetts. 1984.

[19] Leslie LAMPORT: *LATEX. A Document Preparation System. User's Guide and Reference Manual.* Addison-Wesley Publishing Company, Reading, Massachusetts. 1994.

[20] Oren PATASHNIK: *Designing BIBTEX styles.* February 1988. Part of the BibTEX distribution.

[21] Oren PATASHNIK: *BIBTEXing.* February 1988. Part of the BibTEX distribution.

[22] Dave PAWSON: *XSL-FO.* O'Reilly & Associates, Inc. August 2002.

[23] Bernd RAICHLE: *Die Makropakete „german" und „ngerman" für LATEX 2$_\varepsilon$, LATEX 2.09, Plain-TEX and andere darauf Basierende Formate.* Version 2.5. Juli 1998. Im Software LATEX.

[24] Erik T. RAY: *Learning XML.* O'Reilly & Associates, Inc. January 2001.

[21] Whereas that is easy with 'old' BibTEX, provided that you use a bibliography style able to deal with additional fields.

[25] Brian Keith REID: *SCRIBE Document Production System User Manual.* Technical Report, Unilogic, Ltd. 1984.

[26] Thomas SCHRAITLE: *DocBook-XML—Medienneutrales und plattformunabhändiges Publizieren.* SuSE Press. 2004.

[27] John E. SIMPSON: *XPath and XPointer.* O'Reilly & Associates, Inc. August 2002.

[28] George SPRINGER and Daniel P. FRIEDMAN: *Scheme and the Art of Programming.* The MIT Press, McGraw-Hill Book Company. 1989.

[29] *The Unicode Standard Version 3.0.* Addison-Wesley. February 2000.

[30] Eric VAN DER VLIST: *Examplotron.* `http://examplotron.org`. February 2003.

[31] W3C: *XML Path Language (XPath). Version 1.0.* W3C Recommendation. Edited by James Clark and Steve DeRose. November 1999. `http://www.w3.org/TR/1999/REC-xpath-19991116`.

[32] W3C: *XSL Transformations (XSLT). Version 1.0.* W3C Recommendation. Written by Sharon Adler, Anders Berglund, Jeff Caruso, Stephen Deach, Tony Graham, Paul Grosso, Eduardo Gutentag, Alex Milowski, Scott Parnell, Jeremy Richman and Steve Zilles. November 1999. `http://www.w3.org/TR/1999/REC-xslt-19991116`.

[33] W3C: *Extensible Markup Language (XML) 1.0 (Second Edition).* W3C Recommendation. Edited by Tim Bray, Jean Paoli, C. M. Sperberg-McQueen and Eve Maler. October 2000. `http://www.w3.org/TR/2000/REC-xml-20001006`.

[34] W3C: *Extensible Stylesheet Language (XSL). Version 1.0.* W3C Recommendation. Edited by James Clark. October 2001. `http://www.w3.org/TR/2001/REC-xsl-20011015/`.

[35] W3C: *XML Schema.* November 2003. `http://www.w3.org/XML/Schema`.

[36] Norman WALSH and Leonard MUELLNER: *DocBook: The Definitive Guide.* O'Reilly & Associates, Inc. October 1999.