MlBibTeX's Version 1.3

Jean-Michel Hufflen

Abstract

We present the features of the new version of MlBibTeX, a new multilingual implementation of BibTeX, the bibliography program associated with (La)TeX. The main point of this new version is the use of a new language for designing bibliography styles. This language is close to XSLT and we give its manual as an annex.

Keywords bibliographies, multilingual features, BibTeX, bst, nbst, XML, XSLT, MlBibTeX.

1 Introduction

It is well known that a bibliography program should be associated with a text processor. If such a program is used for documents such as history articles, technical documentation, or research work, where many references may be cited, the role of a bibliography program is to search a database containing **bibliographical entries** for the citations throughout the document, sort them and arrange the information associated with each selected entry. In short, it has to build the 'References' section of the document, containing bibliographical **references**, which can be processed by the text processor at next run.

A bibliography program may look for *keys* surrounded by special markers within a source text, as does Tib [1]. Or it may use information included in auxiliary (.aux) files, as does BibTeX [16], most commonly used with (La)TeX [14]. Here is an example of a bibliographical entry using BibTeX's syntax:

```
@BOOK{howard1967b,
        AUTHOR = {Robert~Ervin Howard},
        TITLE = {Conan the Conqueror},
        PUBLISHER = {Ace Books},
        ADDRESS = {New York, New York},
        NOTE = {Edited by L. Sprague de Camp},
        YEAR = 1967}
```

If the entry howard1967b is cited within a document, this information is put into an auxiliary file when LaTeX runs, so BibTeX can generate a .bbl file containing the corresponding reference. When LaTeX runs again, this reference will look like:

> [1] Robert Ervin Howard. *Conan the Conqueror.* Ace Books, New York, New York, 1967. Edited by L. Sprague de Camp.

according to the bibliography **style** chosen. Here and in the 'References' section of this article, we use a 'plain' style, that is, references are labelled with numbers, authors' last names are written using small capitals, and first names are not abbreviated. Other choices are possible: see [6, §13.2] for a survey of available bibliography styles.

Due to its conception, BibTeX has some limitations: its syntax is rough, bibliographic styles are written using an old-fashioned language [15], and multilingual bibliographies are supported only through workarounds. We personally missed this last point very much, thus we have put into action a new implementation of BibTeX, named MlBibTeX (for '**M**ulti**L**ingual BibTeX'), with many multilingual features. The first version (1.1) was described in [8]. But as we explained in [12], the new version described here (1.3) takes advantage of XML[1] and uses a new language, nbst, for '**n**ew **b**ibliography **st**yles', close to XSLT[2] [21].

This article aims to give a survey of all the new features introduced by MlBibTeX's present version. It is not a complete reference manual, but gives a good overview of the program. First, we describe the new syntactical features provided by MlBibTeX.[3] Then, we give some words about the implementation, showing the connection with XML and discussing two approaches for multiligual bibliographies. Then we explain how the information about languages is managed within our bibliography styles and show that nbst allows creators of bibliography styles to put them both into action. Last, a manual of elements and functions of the nbst language is given as an annex.

2 New syntactic features

Historically, we first added syntax for multilingual features [8]. Then we realised that some fields' values could be structured better with some new syntax. Here are the results of our choices.

2.1 Syntax for names

When BibTeX processes the value of an AUTHOR or EDITOR field, it divides a family name into four fields: *First* (for a first name), *von* (for a particle), *Last* (for a last name), and *Junior* and recognizes these components according to the following possible syntaxes [16, §4]:

(i) *First von Last*
(ii) *von Last*, *First*
(iii) *von Last*, *Junior*, *First*

As suggested by the cases used within this terminology, the words belonging to the *von* field are supposed to use only lowercase characters, whereas the

[1] E**X**tensible **M**arkup **L**anguage.

[2] E**X**tensible **S**tylesheet **L**anguage **T**ransformations.

[3] Let us note that 'old' .bib files are parsed successfully by MlBibTeX and give outputs comparable to BibTeX's, unless square brackets are used in field values.

```
@BOOK{howard1969,
        AUTHOR = {Robert Ervin Howard, abbr => R. with
                  first => Lyon Sprague, von => de, last => Camp, abbr => L. Sprague with
                  Lin Carter}
        TITLE = {Conan of {Cimmeria}},
        PUBLISHER = {Ace Books},
        ADDRESS = {New York, New York},
        NOTE = {[Titre de la traduction fran\c{c}aise : ‘‘Conan le Cimm\'{e}rien’’] ! french
                [Titel der deutschen \"{U}bersetzung: ‘‘Conan von Cimmerien’’] ! german}
        YEAR = 1969,
        LANGUAGE = english}
```

Figure 1: A multilingual entry using MlBibTeX's syntax.

words belonging to other fields are supposed to be capitalised. These rules are too restrictive: some particles may be capitalised, while some words belonging to a last name may be written using lowercase characters. Using additional braces solves some problems, but not all. In addition, BibTeX abbreviates a first name by retaining only the first letter of each word belonging to the *First* field, such letters being followed with a period character. That is sometimes incorrect: ‘Jon L White’ should be abbreviated to ‘J. L White’ or ‘J. White’, not to ‘J. L. White’. First and middle American names are handled differently from one name to another. ‘Robert Ervin Howard’ is usually written down as ‘Robert E. Howard’, which becomes ‘R. Howard’ when the first name is abbreviated. In contrast, ‘Henry Rider Haggard’ is usually written down as ‘H. Rider Haggard’, and becomes ‘H. R. Haggard’ in styles where first names are abbreviated. In addition, several letters may be retained when abbreviating a non-English first name:

- in French, ‘Charles Duits’ is abbreviated to ‘Ch. Duits’, because the ‘ch’ group stands for one digraph ([ʃ]);
- likewise, ’Christian’ is abbreviated to ‘Chr.’ in German.

Since Version 1.2 [10], MlBibTeX allows an explicit syntax for these fields and the abbreviation of a first name, if it is different from the ‘standard way’:

```
first => ..., von => ..., last => ...,
junior => ..., abbr => ...
```

The order of the keywords is irrelevant and some may be absent, provided that the last name is specified. For example:

```
first => Henry Rider, last => Haggard
```

where the *von* field is empty, and the abbreviation of the first name is standard, that is, ‘H. R.’ For a more complex example, see the specification of ‘Lyon Sprague de Camp’ in Figure 1. You can mix the ‘old’ and ‘new’ syntaxes, in which case a name is parsed like (i) if no comma occurs, like (ii) (resp. (iii)) if the number of commas not followed with a keyword is one (resp. two) and the keywords give additional information.[4] This is useful when we have to give a specific abbreviation for a first name: see the specification of ‘Robert Ervin Howard’ in Figure 1. In fact, this syntax is close to that for passing values inside a subprogram call in Ada [18, §6.4] and other languages.

When a name is not for a person but for an organisation, it is well known to BibTeX users that such an expression should be surrounded by additional braces:

```
EDITOR = {{\TUGboard 2003}}
```

so BibTeX considers it as a one-component name, this component being a *Last* part. However, this syntax poses a problem when a TeX command is used within such a name. In the given example, ‘`\TUGboard`’ is viewed as an accent command: when the bibliography is sorted, the corresponding entry is alphabeticised as ‘2003’. MlBibTeX's new syntax allows the specification of both an organisation name and a key for sorting:

```
EDITOR = {org => \TUGboard 2003,
          sortingkey => TUG Board 2003}
```

As in BibTeX, co-authors are connected by the ‘`and`’ keyword within .bib files. After one author or several successive co-authors, MlBibTeX allows the addition of *collaborators*, introduced by the ‘`with`’ keyword. Figure 1 gives an example in MlBibTeX.[5]

[4] Nevertheless, defining any part of a name twice causes an error.

[5] Besides, the entry given in this figure allows us to emphasise the difference between co-authors and collaborators. In fact, L. Sprague de Camp and L. Carter sorted and arranged R. Howard's manuscripts after his death. So they are more ‘collaborators’ than co-authors. The entry `howard1967b`, given in the introduction, might be rewritten using this syntax, instead of using a `NOTE` field.

```
<book id="howard1969" language="english">
  <author>
    <name><personname><first abbr="R.">Robert Ervin</first><last>Howard</last></personname></name>
    <with/>
    <name>
      <personname>
        <first abbr="L. Sprague">Lyon Sprague</first><von>de</von><last>Camp</last>
      </personname>
    </name>
    <with/>
    <name><personname><first>Lin</first><last>Carter</last></personname></name>
  <title>
    Conan of <asitis>Cimmeria</asitis>
    <!--  asitis is for a group of words that should not be case-converted.  -->
  </title>
  <publisher>Ace Books</publisher>
  <year>1969</year>
  <address>New York, New York</address>
  <note>
    <group language="french">
      Titre de la traduction française : <emph emf="yes" quotedbf="yes">Conan le Cimmérien</emph>
    </group>
    <group language="german">
      Titel der deutschen Übersetzung: <emph emf="no" quotedbf="yes">Conan von Cimmerien</emph>
    </group>
  </note>
</inproceedings>
```

Figure 2: The entry given in Figure 1 viewed as an XML tree.

As in BibTeX, the ‘`others`’ keyword can be used when additional names are left unspecified: ‘`and others`’ and ‘`with others`’ are allowed. In the bibliography of this article, reference [7] shows how such an entry using collaborators is formatted.

2.2 Syntax for multilingual features

In MlBibTeX's terminology, a **language identifier** is a non-ambiguous prefix of:

- an option of the babel package [2],
- or a multilingual package name such as french [5], german [17] or polski [4].[6]

The language of an entry is given by the `LANGUAGE` field, whose value is a language identifier (see Figure 1). This field defaults to ‘`english`’.

Here we only show the syntax we use for multilingual features included in .bib files; a more complete description can be found in [8], and more examples in [12]. In the following, ‘s’, ‘s_1’, . . . , ‘s_n’ are strings; n is a positive natural number; and ‘l’, ‘l_1’, . . . , ‘l_n’ are language identifiers.

A **language change** is denoted by ‘$\texttt{[}s\texttt{] : }l$’. It is used for foreign words and in particular, it allows a text processor to hyphenate them correctly.

A **language switch without default language** is expressed by the following syntax:

$$\texttt{[}s_1\texttt{] ! }l_1\ \ldots\ \texttt{[}s_n\texttt{] ! }l_n \tag{1}$$

If there exists i $(0 \leq i \leq n)$ such that the reference's language is equal to l_i, then Expression (1) yields s_i; otherwise, this expression is replaced by an empty string. In other words, this syntax is used for additional information that must be typeset in a particular language. For example, if we process the entry `howard1969` in French (resp. German), we can add the title of the French (resp. German) translation, as shown in the `NOTE` field in Figure 1.

A **language switch with default language** is expressed by the following syntax:

$$\texttt{[}s_1\texttt{] * }l_1\ \ldots\ \texttt{[}s_n\texttt{] * }l_n \tag{2}$$

This syntax is used for information that *must* be included, possibly in another language. If there exists i $(0 \leq i \leq n)$ such that the reference's language is equal to l_i, then Expression (2) yields s_i; otherwise, this expression is replaced by the string associated

[6] This choice of a non-ambiguous prefix allows a language identifier to get access to several ways to process a language. For example, a language identifier set to `french` works with the frenchb option of the babel package as well as the french package.

with the language's entry if such a string exists, or by the string associated with the English language if not. For example, we could allow the publisher's address of the howard1969 entry to use a Russian transliteration for a reference to this entry in Russian. Of course, this address is to be put in English otherwise. To do that, the ADDRESS field should be given such a value:

```
ADDRESS =
  {[New-York]
   [⟨Russian transliteration⟩] * russian}
```

Notice that '[...]', not followed with '*', '!' or ':' means '[...] * *l*', where '*l*' is the language's entry.

2.3 Syntax for page numbers

In a PAGES field, MlBibTeX recognizes:

- a single page (one token): {2003};
- the first and last pages (three tokens):

  {2000--2003} or {2000-2003}
- the first page and an unspecified number of following ones (two tokens): {2003+};[7]
- some enumerated pages (five tokens in the example below): {2000,2003,2005}.

The tokens may or may not be separated by whitespace[8] characters. In all the other cases, the value associated with this field is kept *verbatim* and appears as-is for any predefined bibliography style.

3 Implementation issues

MlBibTeX's first version [8] was written using C, for the sake of efficiency and portability. When we started implementation of the present version, we realised that we needed calls to *external functions* within our bibliography styles.[9] So we realised that it was preferable for our program to be written in a higher-level programming language. This way, the interface between bibliography styles and external functions would be designed better, so developers of new styles could write extensions in the source language more easily. We decided to develop a prototype in Scheme, with the features related to XML put into action by SXML[10] [13], an implementation of XML trees by means of Scheme expressions. Our nbst language, for bibliography styles, includes a call function (see Appendix B), that gives access to Scheme functions of MlBibTeX's library.

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

 <!-- Reference-dependent approach: -->
 <nbst:param name="language" select="'*self*'"/>

 <!-- Root element grouping entries: -->
 <nbst:template match="mlbiblio">
   ...
 </nbst:template>

 ...
</nbst:bst>
```

Figure 3: Layout of a bibliography style file using nbst.

Parsing an MlBibTeX entry results in a representation of an XML tree in SXML; for example, the entry of Figure 1 is equivalent to the XML tree given in Figure 2, that is, if the SSAX[11] parser of SXML is applied to this XML tree, it yields the same result. Our XML trees modelling entries are conformant with a revised version of the DTD[12] sketched in [9]. They are rooted by the mlbiblio element, as suggested by the first template given in Figure 3.

In addition, SXML relies on functions extending the basic encoding of characters used in Scheme. These functions should allow Scheme programs to handle Unicode, but they are platform-dependent: some interpreters provide them, possibly partially, some do not. In practice, MlBibTeX can handle 8-bit latin1 encoding;[13] further development will be needed to adapt MlBibTeX to the whole of Unicode,[14] but the framework to do that is already present.

4 Multilingual approaches

As mentioned in [8], multilingual bibliographies can be organised with respect to two approaches, both of which can be put into action by MlBibTeX:

reference-dependent each reference of the document's bibliography is expressed using its own language: for example, the month name of a reference to a book written in English (resp. French, German, ...) is given in English (resp. French, German, ...);

[7] Such a specification is typeset as 'pp. 2003 ff.' in English-speaking bibliographies [3, §15.191].

[8] The whitespace characters are space, tab, newline, carriage return, and form feed.

[9] These external calls are used to manage information not included in .aux files. So it has to be directly extracted from .tex files.

[10] **S**cheme implementation of XML.

[11] Scheme implementation of SAX ('**S**imple API for XML').

[12] **D**ocument **T**ype **D**efinition (document markup model).

[13] [7, Table C.4] has more details about encodings.

[14] If you would like to use characters from non-Latin alphabets (e.g., Cyrillic characters), now put the LaTeX commands to produce them, rather than these characters themselves. A temporary situation, we hope.

```
<nbst:template match="author">
  <nbst:apply-templates/>
  <nbst:text>: </nbst:text>
</nbst:template>

<nbst:template match="name">
  <nbst:apply-templates/>
</nbst:template>
```

Figure 4: Formatting names in nbst.

document-dependent all references are expressed using the document's language, as far as possible.

5 The nbst Language

Most elements of nbst behave like their namesakes in XSLT. Figure 3 gives the general layout of a bibliography style and a representative example is given in Figures 4 & 5. The path expressions used in these figures are related to the tree given in Figure 2. Let us notice that some elements and attributes of are recognised by the nbst processor, but do not have any effect presently — they have been planned for future use of MlBibTeX, especially for generating XML documents[15] — this information is given in Appendix A. We assume that readers are quite familiar with XPath [20] and XSLT [21] — there exist some good introductory books about them, for example, [19] — so in this section we only explain how the language information is managed by the nbst processor.

Given a fragment of an entry viewed as a node (an XML subtree), its **current language** is the value of the `language` attribute if it exists, the value of the current language of its parent otherwise. The current language for an entry is the entry's language (see Section 2.2).

When templates are to be instantiated, the rule added to those inherited from XSLT is that a template with the `language` attribute has higher priority than the same template without it.[16] This rule overrides all the others. In particular, it applies if a template is invoked by name,[17] as well being applied if the current node matches the pattern of its `match` attribute.

When we begin to apply a bibliography style, the `language` attribute is associated with the document's language[18] (resp. the '`*self*`' value) according to the document-dependent (resp. reference-dependent) approach. When a template is to be invoked by name by means of such a statement:

```
<nbst:call-template name="..."/>
```

then we look for the current language. If this value is different from '`*self*`', we look for the named template with the `language` attribute set to this value if it exists. If not, the default named template, that is, without the `language` attribute, is invoked. The `use-language` attribute allows the redefinition of the current language; for example:

```
<nbst:call-template
    name="..." use-language="portuguese"/>
```

invokes a named template with the `language` attribute set to '`portuguese`' if such a template exists, its namesake without this attribute if not. The same rules applies for the `nbst:apply-templates` element:

```
<nbst:apply-templates
  select="S" use-language="finnish"/>
```

tries to find, for each node selected by the expression S, a template with the `language` attribute set to the right value (here, `finnish`) before instantiating the template without the `language` attribute. The same rule holds for templates with a `mode` attribute: given a set of templates with the same value associated with the `mode` attribute, we apply first the template with the right value for the `language` attribute, second the template without this attribute. As in XSLT [21, § 5.7], an `nbst:apply-templates` element with a `mode` attribute can only apply templates with the same value for this mode.

Using the '`*self*`' value is of little interest with an `nbst:call-template` element since the current node does not change when a template is invoked by its name. So the statement:

```
<nbst:call-template name="..."
                     use-language="*self*"/>
```

is equivalent to:

```
<nbst:call-template name="..."/>
```

unless the language of the template instantiated is not the current node's language. The statement:

```
<nbst:apply-templates
  select="S" use-language="*self*"/>
```

dispatches all the selected nodes w.r.t. their associated languages. It is equivalent to:

[15] In particular, we plan to investigate the generation of 'References' sections for DocBook documents [22].

[16] In fact, there are two levels of priority: the first is ruled by the `language` attribute, the second defined by XSLT, including the `priority` attribute.

[17] As a consequence, there can be several templates with the same name — which is an error in XSLT [21, §6] — provided that the values possibly associated with the different `language` attributes are pairwise-different.

[18] MlBibTeX tries to determine it as far as possible. Most often, it is the last option given to the babel package.

```
<nbst:template match="personname">
  <nbst:if test="first"><nbst:value-of select="first"/><nbst:text> </nbst:text></nbst:if>
  <nbst:if test="von"><nbst:value-of select="von"/><nbst:text> </nbst:text></nbst:if>
  <nbst:text>\textsc{</nbst:text><nbst:value-of select="last"/><nbst:text>}</nbst:text>
  <nbst:if test="junior">, Junior</nbst:if>
</nbst:template>

<nbst:template match="and">
  <nbst:choose>
    <nbst:when test="following-sibling::and or following-sibling::and-others">
      <nbst:text>, </nbst:text>
    </nbst:when>
    <nbst:otherwise>
      <nbst:text> </nbst:text><nbst:value-of select="$bbl.and"/><nbst:text> </nbst:text>
    </nbst:otherwise>
  </nbst:choose>
</nbst:template>

<nbst:template match="and-others">
  <nbst:text> </nbst:text><nbst:value-of select="$bbl.etal"/>
</nbst:template>
```

Figure 5: Formatting names with the nbst language (*continued*).

```
<nbst:for-each select="S">
  <nbst:apply-templates select="."
                        use-language="L"/>
</nbst:for-each>
```

where L is the current language of the current node. This expression is used for the mlbiblio element to build references in the reference-dependent approach.

As an example, the template given in Figure 6 is instantiated for this name:

```
AUTHOR = {[Zoltán Kodály] : hungarian}
```

6 Conclusion

Roughly speaking, we can consider that getting a bibliographical reference from an entry is a particular case of transformation — the same information, arranged differently. Thus, an XSLT-like language should be suitable for the task. In addition, our management of the information related to particular languages should ease the making of mutilingual bibliographies. At the time of writing, our program is in beta test and we have successfully rewritten a representative range of bibliography styles of BibTeX. So we think we are ready for public use and larger experiment.

7 Acknowledgements

Special thanks to Hans Hagen and Volker R. W. Schaa, who agreed to give the show associated with a preliminary version of this paper at the TUG 2003 conference. Thanks to Karl Berry and Barbara Beeton who proofread this revised and updated version.

References

[1] James C. Alexander: *Tib: A TeX Bibliographic Preprocessor*. Version 2.2, see CTAN: biblios/tib/tibdoc.tex. 1989.

[2] Johannes 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] *The Chicago Manual of Style*. The University of Chicago Press. The 14th edition of a manual of style revised and expanded. 1993.

[4] Antoni Diller: *LaTeX wiersz po wierszu*. Wydawnictwo Helio, Gliwice. Polish translation of *LaTeX Line by Line* with an additional annex by Jan Jelowicki. 2001.

[5] 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.

[6] Michel Goossens, Frank Mittelbach and Alexander Samarin: *The LaTeX Companion*. Addison-Wesley Publishing Company, Reading, Massachusetts. 1994.

[7] Michel Goossens and Sebastian Rahtz, with Eitan M. Gurari, Ross Moore and Robert S. Sutor: *The LaTeX Web Companion*. Addison-Wesley Longman, Inc., Reading, Massachusetts. May 1999.

```
<nbst:template match="personname" language="hungarian">   <!--  Here, the family name comes first.  -->
  <nbst:text>\textsc{</nbst:text>
  <nbst:if test="von"><nbst:value-of select="von"/><nbst:text> </nbst:text></nbst:if>
  <nbst:value-of select="last"/><nbst:text>}</nbst:text>
  <nbst:if test="first"><nbst:text> </nbst:text><nbst:value-of select="first"></nbst:if>
  <nbst:if test="junior">, Junior</nbst:if>
</nbst:template>
```

Figure 6: Formatting Hungarian names with the nbst language.

[8] Jean-Michel Hufflen: “MlBibTeX: a New Implementation of BibTeX”. In: *EuroTeX 2001* (pp. 74–94). Kerkrade, The Netherlands. September 2001.

[9] Jean-Michel Hufflen: “Multilingual Features for Bibliography Programs: from XML to MlBibTeX”. In: *EuroTeX 2002* (pp. 46–59). Bachotek, Poland. April 2002.

[10] Jean-Michel Hufflen: “Towards MlBibTeX’s Versions 1.2 & 1.3”. MaTeX Conference. Budapest, Hungary. November 2002.

[11] Jean-Michel Hufflen: “Mixing Two Bibliography Style Languages”. In: *LDTA 2003*, Vol. 82.3 of *ENTCS*. Elsevier, Warsaw, Poland. April 2003.

[12] Jean-Michel Hufflen: “European Bibliography Styles and MlBibTeX”. *TUGboat*, Vol. 24, no. 3 (in process). EuroTeX 2003, Brest, France. June 2003.

[13] Oleg Kiselyov: “A Better XML Parser through Functional Programming”. In: *4th International Symposium on Practical Aspects of Declarative Languages*, Vol. 2257 of *LNCS*. Springer-Verlag. 2002.

[14] Leslie Lamport: *LaTeX: A Document Preparation System. User’s Guide and Reference Manual*. Addison-Wesley Publishing Company, Reading, Massachusetts. 1994.

[15] Oren Patashnik: “Designing BibTeX styles”. February 1988. Part of BibTeX distributions.

[16] Oren Patashnik: “BibTeXing”. February 1988. Part of BibTeX distributions.

[17] Bernd Raichle: *Die Makropakete „german“ und „ngerman“ für LaTeX 2ε, LaTeX 2.09, Plain-TeX and andere darauf Basierende Formate*. Version 2.5. Juli 1998. Im Software LaTeX.

[18] S. Tucker Taft and Robert A. Duff, eds.: *Ada 95 Reference Manual. Language and Standard Libraries*. No. 1246 in LNCS. Springer-Verlag. International Standard ISO/IEC 8652:1995(E). 1995.

[19] Doug Tidwell: *XSLT*. O’Reilly & Associates, Inc. August 2001.

[20] 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.

[21] 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.

[22] Norman Walsh and Leonard Muellner: *DocBook: The Definitive Guide*. O’Reilly & Associates, Inc. October 1999.

Appendix A Elements of nbst

Hereafter, we describe each element of nbst. For each of them, we give its *syntax*: the attributes associated with it, and its content. For each attribute, we underline its name if it is required, and give the type of its possible values. When these values are enumerated, the default value is underlined.

The syntax is defined using regular expressions: the ‘|’ sign means an alternative, ‘?’ is used for an optional element, ‘*’ (resp. ‘+’) means zero (resp. one) or more occurrences of an element.

Here are the type identifiers used throughout this section:

CDATA for ‘**C**haracter **DATA**’, that is, literal data characters without ‘<, ‘>’, ‘&’;[19]

char literal character;

expr analogous to an XPath expression;

id *unique* identifier for a resource;

lg-expr expression that results in either a non-ambiguous prefix of available languages or the ‘*self*’ keyword;

name simple identifier;[20]

[19] As in XML, use the entities ‘&lt;’, ‘&gt;’, ‘&amp;’ for these characters.

[20] ‘*name*’ is used instead of ‘qualified name’ within XSLT since Version 1.3 does not allow namespaces, except for nbst.

nmtoken whitespace-free sequence of characters;

number *constant* number;

pattern expression allowed within the `match` attribute of the `nbst:template` element;

template any (possibly empty) sequence of nbst elements, except for top-level ones;

top-level-elt element allowed at the top level;

uri-ref now a simple identifier.[21]

Plurals denote non-empty sequences whose elements are separated by whitespace characters: for example, '*names*' is for a non-empty sequence of objects each of type '*name*'.

`<nbst:accumulate>`

Synt.:
```
<nbst:accumulate>
  template
</nbst:accumulate>
```

Pushes the result of *template* onto the stack used when we process a bst function (see [11] for more details). Several `nbst:accumulate` elements can be given sequentially, but they cannot be nested.

`<nbst:apply-templates>`

Synt.:
```
<nbst:apply-templates
    select=expr mode=name
    use-language=lg-expr>
  (nbst:with-param |
   nbst:sort)*
</nbst:apply-templates>
```

Processes the node set selected by the value of the `select` attribute, or all the children of the current node by default. The selected node set is processed in document order, unless a sorting specification is present. About the attributes `mode` and `use-language`, see Section 5.

`<nbst:attribute>`

Synt.:
```
<nbst:attribute name=name>
  template
</nbst:attribute>
```

Recognised, but does not have any effect, like `nbst:attribute-set` and `nbst:element`. See Section 5.

`<nbst:attribute-set>`

Synt.:
```
<nbst:attribute-set
    name=name
    use-attribute-sets=names>
  nbst:attribute*
</nbst:attribute-set>
```

See `nbst:attribute`.

`<nbst:bst>`

Synt.:
```
<nbst:bst id=id version=number>
  top-level-elt*
</nbst:bst>
```

Root element of a bibliography style. The only version number presently recognised is 1.3.

`<nbst:call-template>`

Synt.:
```
<nbst:call-template
    name=name
    use-language=lg-expr>
  nbst:with-param*
</nbst:call-template>
```

Invokes a template by name by means of the required `name` attribute. See Section 5 about the `use-language` attribute.

`<nbst:choose>`

Synt.:
```
<nbst:choose>
  nbst:when+ nbst:otherwise?
</nbst:choose>
```

Each of the `nbst:when` elements is tested in turn, until reaching an element whose test is *true*, in which case the content is instantiated. If no such element exists, then the content of the `nbst:otherwise` element is instantiated if it exists, otherwise nothing is created.

`<nbst:comment>`

Synt.:
```
<nbst:comment>
  template
</nbst:comment>
```

Puts the result of *template* as a comment. In practice, now used to write lines beginning with '%' in LaTeX mode.

`<nbst:copy>`

Synt.:
```
<nbst:copy
    use-attribute-sets=names>
  template
</nbst:copy>
```

Copies the current node at the first level onto the result. The `use-attribut-sets` attribute does not have any effect presently.

`<nbst:copy-of>`

Synt.:
```
<nbst:copy-of select=expr/>
```

Copies the whole of the node set selected by the required `select` attribute.

`<nbst:decimal-format>`

Synt.:
```
<nbst:decimal-format
    name=name
    decimal-separator=char
    grouping-separator=char
    infinity=cdata
    minus-sign=char NaN=cdata
```

[21] True ***U****niform* ***R****esource* ***I****dentifiers*, in the sense of XML, will be allowed in a future version.

```
        percent=char per-mille=char
        zero-digit=char digit=char
        pattern-separator=char/>
```

Declares a decimal format, which rules the interpretation of a format pattern used by the `format-number` function. If there is a `name` attribute, then this element declares a named decimal format; otherwise, it declares the default decimal format. Here are the other attributes:

- `decimal-separator` specifies the character used for the decimal sign, defaults to the period character ('`.`');
- `grouping-separator`: the character used as a grouping (e.g., thousands) separator, defaults to '`,`';
- `infinity`: the identifier used to represent infinity, defaults to '`Infinity`';
- `minus-sign`: the character used as the default minus sign, defaults to '`-`';
- `NaN`: the identifier used to represent a value that should be a number but is not, defaults to '`NaN`' ('**N**ot **a** **N**umber');
- `percent` and `per-mille`: the two characters used as percent and per-mille signs ('%' and '‰'); in LaTeX mode, default to the command producing them ('`\%`' and '`\textperthousand`'[22]);
- `zero-digit`: a character always replaced by a digit, defaults to '`0`';
- `digit`: a character used for a digit, left blank for a missing digit, defaults to '`#`';
- `pattern-separator`: the character used to separate sub-patterns for positive and negative patterns, defaults to '`;`'.

`<nbst:element>`

Synt.:

```
<nbst:element
    name=name
    use-attribute-sets=names>
  template
</nbst:element>
```

See `nbst:attribute`.

`<nbst:for-each>`

Synt.:

```
<nbst:for-each select=expr>
  nbst:sort* template
</nbst:for-each>
```

`template` is instantiated for each node selected by the required `select` expression, which must evaluate to a node set. The selected nodes are processed in document order, unless a sorting specification is present.

`<nbst:if>`

Synt.:

```
<nbst:if test=expr>
  template
</nbst:if>
```

If the evaluation of the `test` attribute results in *true*, then *`template`* is instantiated; otherwise, nothing is created.

`<nbst:include>`

Synt.:

```
<nbst:include href=uri-ref/>
```

Includes elements belonging to another nbst or bst file, identified by the `href` attribute. Allowed as a top-level element only.

`<nbst:key>`

Synt.:

```
<nbst:key name=name
          match=pattern
          use=expr/>
```

Recognised but does not have any effect.

`<nbst:message>`

Synt.:

```
<nbst:message
     terminate=("yes" | "no")>
   template
</nbst:template>
```

Displays the result of *`template`* as a message. If the `terminate` attribute has the value '`yes`', then the program terminates after displaying the message.

`<nbst:number>`

Synt.:

```
<nbst:number
     level=("single" |
            "multiple" | "any")
     count=pattern from=pattern
     value=expr format=cdata
     language=lg-expr
     letter-value=
     ("alphabetic" |
      "traditional")
     grouping-separator=char
     grouping-size=number/>
```

Puts a formatted number. The number may be specified by means of the `value` attribute, in which case the expression is evaluated and the `number` and `round` functions are applied to the resulting object. If no `value` attribute is specified, then the inserted number is based on the position of the current node, controlled by the following attributes:

- `level` specifies which levels of the source tree should be considered;
- `count` attribute is a pattern that specifies what nodes should be counted at those levels: if it is unspecified, it defaults to the

[22] Notice that this command can be used with the Cork encoding, that is, the `T1` option of the fontenc package.

pattern matching any node with the same node type as the current node;

- `from`: a pattern that specifies where counting starts.

The `format` attribute is split into alphanumeric and non-alphanumeric characters. The former are formats for numbers:

- '`1`' for `1`, `2`, ...
- '`i`' (resp. '`I`') for `i`, `ii`, ... (resp. `I`, `II`, ...)
- '`a`' (resp. '`A`') for `a`, `b`, ... (resp. `A`, `B`, ...), the `language` attribute being used to determine the alphabetical order.

The latter are copied *verbatim* onto the formatted string. Consult `nbst:decimal-format` about the `grouping-separator` attribute. The `grouping-size` attribute specifies the size of the grouping, defaulting to 3. If only one of these two attributes is specified, then it is ignored. The `letter-value` attribute does not have any effect.

`<nbst:otherwise>`

Synt.:
```
<nbst:otherwise>
  template
</nbst:otherwise>
```

See `nbst:choose`.

`<nbst:output>`

Synt.:
```
<nbst:output
    method=("LaTeX" | "xml" |
            "html" | "text")
    version=nmtoken
    encoding=cdata
    omit-xml-declaration=
    ("yes" | "no")
    standalone=("yes" | "no")
    doctype-public=cdata
    doctype-system=uri-ref
    cdata-section-elements=
    names
    indent=("yes" | "no")
    media-type=cdata/>
```

Only allowed as a top-level element. Allows bibliography style writers to specify how they wish the result to be output. Presently, the values allowed for the `method` attribute are:

- '`LaTeX`', for LaTeX output;
- '`xml`' (resp. '`html`'), for XML (resp. HTML) output; however, do not forget that, as with XSLT, the output for an HTML file must be written according to XHTML[23] conventions;
- '`text`', for verbatim text output.

Other attributes:

- `version` specifies the version of the output method,
- `encoding`: the character encoding to be used;
- `omit-xml-declaration`: whether or not the XML declaration should be output;
- the other attributes do not have any effect.

`<nbst:param>`

Synt.:
```
<nbst:param name=name
            select=expr>
  template
</nbst:param>
```

Used at the top level to define an external parameter or within a template rule to specify a local parameter. The `select` attribute gives a default value. When this attribute is absent, the default value is given by instantiating *`template`* if it is not empty. If this parameter is not given a default value, nbst pops the stack used when we process a bst function; if this stack is empty, the value given to the parameter is the empty string.

`<nbst:sort>`

Synt.:
```
<nbst:sort
    select=expr
    language=lg-expr
    data-type=
    ("text" | "number")
    order=("ascending" |
           "descending")
    case-order=("upper-first" |
                "lower-first")/>
```

Used as a child of an `nbst:apply-templates` or `nbst:for-each` element. The first occurrence specifies the primary sort key, the second occurrence the secondary sort key used for elements left unsorted, and so on. The key is given by the `select` attribute, which defaults to '`.`'. This expression is applied to each node of the current set, and the result is converted into a string or a number, w.r.t. the value of the `data-type` attribute. In addition:

- `order` can be ascending or descending;
- `language`: the sort keys' language;
- `data-type`: the sort keys' data type:
  - '`text`' means that they should be lexicographically sorted in the culturally correct way for the current language,
  - '`number`' specifies a numerical sort, in which case `language` is ignored;

[23] **EX**tensible **H**yper**T**ext **M**arkup **L**anguage.

- the possible values for `case-order` apply when `data-type` is '`text`', and specifies that upper-case letters should sort before lower-case letters or *vice-versa*. The default value is language-dependent.

`<nbst:template>`

Synt.:
```
<nbst:template
    match=pattern name=name
    language=lg-expr
    priority=number mode=name>
  nbst:param* template
</nbst:template>
```

Defines a template rule. The `match` attribute is a pattern that identifies the source node to which the rules apply. The `match` attribute is required unless a `name` attribute is given, but both attributes can be specified. It is an error for the value of the `match` attribute to contain a reference to a variable. When such a rule is applied, *`template`* is instantiated.

Templates can be invoked by name, in which case the `match` attribute has no effect; likewise with the `name` attribute if the template is invoked by an `nbst:apply-templates` element. The role of the attributes `language`, `mode` and `priority` is explained in Section 5.

`<nbst:text>`

Synt.:
```
<nbst:text
    disable-output-escaping=
    ("yes" | "no")>
  cdata
</nbst:text>
```

Copies its content *verbatim* onto the output. The `disable-output-escaping` attribute does not have any effect.

`<nbst:variable>`

Synt.:
```
<nbst:variable name=name
               select=expr>
  template
</nbst:variable>
```

Analogous to `nbst:param`, but the value associated with a variable cannot be redefined by an element such as `nbst:with-param`.

`<nbst:value-of>`

Synt.:
```
<nbst:value-of
    select=expr
    disable-output-escaping=
    ("yes" | "no")/>
```

The value of the required `select` attribute is evaluated and the resulting object is converted to a string. The `disable-output-escaping` attribute does not have any effect.

`<nbst:warning>`

Synt.:
```
<nbst:warning>
  template
</nbst:warning>
```

Equivalent to `nbst:message` with `terminate` set to '`no`'.

`<nbst:when>`

Synt.:
```
<nbst:when test=expr>
  template
</nbst:when>
```

See `nbst:choose`.

`<nbst:with-param>`

Synt.:
```
<nbst:with-param
    name=name select=expr>
  template
</nbst:with-param>
```

Passes values to parameters before instantiating templates. The required `name` attribute specifies the name of the parameter, its value is specified in the same way as for `nbst:param`. The current node and node list used for computing the value are the same as for the element within which it can occur (`nbst:apply-templates` or `nbst:call-template`).

Appendix B Functions associated with our paths

We begin this section by describing the types used within the functions associated with our paths. As in XPath, we allow some type conversions. So, for each type, we mention which other types can be converted into it.

`boolean` is for the truth values: *true* and *false*. A node set is viewed as *false* if it is empty, as *true* otherwise. Likewise a string. A number is viewed as *false* if it is equal to zero, *true* otherwise.

`node-set` A node set belonging to the tree of bibliographical entries. A string can be converted into a one-element node set if it is a well-formed XML text, otherwise the result is an empty node set. A boolean or numerical value can be converted into a text node.

`number` When applied to integers, functions using numbers return integer results as far as possible, real numbers otherwise. A string can be converted into a number, provided the characters it contains form a number, possibly surrounded by whitespace characters:

`"␣-273.15"` is a number,
`"-␣273.15"` is not.

If such a conversion fails, the result is `NaN`. If `NaN` is used instead of a number as an argument of a numeric function, the result is `NaN`.

`string` Boolean and numbers can be converted into strings. So can the values for numeric errors, `Infinity` and `NaN`. Node sets too, in which case an attribute node is converted into its associated value, whereas an element node is converted into the concatenated values of all the text nodes inside it.

Throughout this section, 'n', 'ns', 's' denote variables of type `number`, `node-set`, `string` respectively, whereas 'x' is for an expression of any type. If several variables of the same type are needed, we use indices. Some functions can be applied to any number of arguments, in which case the additional optional arguments are denoted by '`...`'. As in XPath, some arguments can be omitted, in which case the current node set is passed: we denote this behaviour by a question mark ('`?`'). For each function, we give the type of its result, a template of its use and a short description of its behaviour.

`!=`

Use: `boolean` x_1 `!=` x_2

Returns *true* if x_1 and x_2 are distinct objects, *false* otherwise.[24]

`*`, `+`, `-`

Use: `number` n_1 `*` n_2 (resp. n_1 `+` n_2, n_1 `-` n_2)

Returns $n_1 * n_2$ (resp. $n_1 + n_2$, $n_1 - n_2$).

`<`, `<=`

Use: `boolean` n_1 `<` n_2 (resp. n_1 `<=` n_2)

Returns *true* if $n_1 < n_2$ (resp. $n_1 \leq n_2$), *false* otherwise.[24]

`=`

Use: `boolean` x_1 `=` x_2

Returns *true* if:

- x_1 and x_2 are the same object,
- or have a common element if x_1 or x_2 is a node set;

returns *false* otherwise.[24]

`>`, `>=`

Use: `boolean` n_1 `>` n_2 (resp. n_1 `>=` n_2)

Returns *true* if $n_1 > n_2$ (resp. $n_1 \geq n_2$), *false* otherwise.[24]

`abbreviate`

Use: `string abbreviate(`s`)`

Assuming that s is a first name, returns its abbreviation. If an *ad hoc* abbreviation has been specified by means of the `abbr` keyword, returns it. Otherwise, s is abbreviated in a standard way, that is, the initials and the hyphen character are retained:

```
abbreviate("John Fitzgerald)"
                          yields "J. F."
abbreviate("Paul-Loup") ..... "P.-L."
```

`abs`

Use: `number abs(`n`)`

Returns the absolute value of n.

`and`

Use: `boolean` b_1 `and` b_2

Returns *true* if b_1 and b_2 are both *true*, *false* otherwise.

`boolean`

Use: `boolean boolean(`x`)`

Converts x to a boolean *true* or *false* value.

`call`

Use: `string call(`s_1`,`s_2`,...)`

Calls s_1, a function included in MlBibTeX's library, with the arguments s_2, ... The s_1 function must return a string which is the result of the `call` function. In practice, this function is used by the multilingual interface.

`ceiling`

Use: `number ceiling(`n`)`

Returns the smallest integer that is greater than or equal to n.

`concat`

Use: `string concat(`s_1`,`s_2`,...)`

Returns the concatenation of the values of the passed arguments.

`contains`

Use: `string contains(`s_1`,`s_2`)`

Returns *true* if s_1 contains s_2, *false* otherwise.

`count`

Use: `number count(`ns`)`

Returns the number of nodes in ns.

`current`

Use: `node-set current()`

Returns the current node as a node set.

`div`

Use: `number` n_1 `div` n_2

Divides n_1 by n_2. If n_2 is equal to zero, this operation results in `Infinity` — this value is not a string.

`false`

Use: `boolean false()`

Returns the *false* value.

`firstcapitalize`

Use: `string firstcapitalize(`s`)`

Converts s to all lowercase except for the first word, which is capitalised.

[24] Notice that `NaN != NaN` yields *true*, whereas `NaN` *op* `NaN` yields *false* if $op \in \{$`<`, `<=`, `=`, `>`, `>=`$\}$.

floor

Use: number floor(n)

Returns the largest integer that is less than or equal to n.

format-number

Use: number format-number(n,s_1,s_2?)

Formats n according to the specifications of s_1 (see nbst:decimal-format) and the name s_2.

generate-newly

Use: string generate-newly(s_1,s_2,ns?)

Returns a unique string associated with the first node of ns. If s_1 is not empty, it is used as result's prefix. If s_2 is not empty, it must be a format used for numbers (see the description of the format attribute of nbst:number) and is used to generate result's suffixes.

id

Use: node-set id(x)

Returns the element node with an ID-type equal to the value of x. This function is useful when we are looking for an entry.

is-boolean

Use: boolean is-boolean(x)

Returns *true* if x is a boolean value, *false* otherwise.

is-defined

Use: boolean is-defined(s)

Returns *true* if s is the name of a parameter or variable bound to a value, *false* otherwise.

is-node-set

Use: boolean is-node-set(x)

Returns *true* if x is a (possibly empty) node set, *false* otherwise.

is-number

Use: boolean is-number(x)

Returns *true* if x is a number, *false* otherwise.

is-string

Use: boolean is-string(x)

Returns *true* if x is a string, *false* otherwise.

key

Use: node-set key(s,x)

Not implemented presently, so always returns an empty node set.

last

Use: integer last()

Returns the number of nodes in the current node set.

local-name

Use: string local-name(ns?)

Returns the name of the first node of ns?.

lowercase

Use: string lowercase(s)

Converts s completely to lowercase.

mod

Use: number n_1 mod n_2

Returns the remainder after dividing n_1 by n_2. The result always has the sign of n_1. If n_2 is equal to zero, the result is NaN.

name

Use: string name(ns?)

Returns the name of the first node of ns.[25]

node-set

Use: node-set node-set(x)

Converts x to a node set.

normalize-space

Use: string normalize-space(s)

Returns the whitespace-normalised value of s, that is, s is stripped of leading and trailing whitespace characters, and multiple consecutive occurrences of whitespace characters are replaced by a single space.

not

Use: boolean not(b)

Returns *true* (resp. *false*) if b is *false* (resp. *true*).

number

Use: number number(x)

Converts x to a numerical value.

or

Use: boolean b_1 or b_2

Returns *true* if b_1 or b_2 is *true*, *false* otherwise.

position

Use: integer position()

Returns the ordinal position of the context node within the context node set. These positions are counted starting from one, as in XPath.

round

Use: number round(n)

Returns the integer nearest in value to n. If n has a decimal portion of exactly .5, rounds up.

starts-with

Use: boolean starts-with(s_1,s_2)

Returns *true* if s_1 begins with s_2, *false* otherwise.

string

Use: string string(x)

Converts x to a string.

[25] Presently, the name and local-name functions return the same result since Version 1.3 does not allow namespaces.

string-length

Use: number string-length(s)

Returns the number of characters in s.

substring

Use: string substring(s,n_1,n_2)

Returns the portion of s starting at character n_1, for a length of n_2 characters.

substring-after

Use: string substring-after(s_1,s_2)

Returns the portion of s_1 following s_2.

substring-before

Use: string substring-before(s_1,s_2)

Returns the portion of s_1 preceding s_2.

sum

Use: number sum(ns)

Returns the sum of all nodes in ns after converting each to a number.

translate

Use: string translate(s_1,s_2,s_3)

Replaces any individual characters appearing in both s_1 and s_2 with corresponding characters in s_3.

true

Use: boolean true()

Returns the *true* value.

uppercase

Use: string uppercase(s)

Converts s completely to uppercase.

Appendix C Comparison with XPath and XSLT

Here we sum up the differences between XPath and XSLT on the one hand, and nbst on the other. These languages are close to each other, so learning nbst is easy if you know XPath and XSLT.

C.1 nbst vs XSLT

The corresponding element of the xsl:stylesheet element in XSLT is nbst:bst in nbst. For the sake of compatibility with the bst language of BibTeX, we added the nbst:warning element, but it can be viewed as a particular case of nbst:message, close to xsl:message.

- XSLT elements without equivalent in nbst:

  xsl:apply-imports
  xsl:fallback
  xsl:import
  xsl:processing-instruction
  xsl:namespace-alias
  xsl:preserve-space
  xsl:strip-space

- nbst element without equivalent in XSLT:

  nbst:accumulate

C.2 XPath vs nbst paths

- XPath functions not included in nbst:

  document
  element-available
  function-available
  lang
  namespace-uri
  system-property
  unparsed-entity-uri

- Additional functions in nbst:

  abbreviate
  call
  firstcapitalize
  is-boolean
  is-defined
  is-node-set
  is-number
  is-string
  lowercase
  node-set[26]
  uppercase

- Close, but not identical functions:

  (XSLT) generate-id ~ generate-newly (nbst)

◇ 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

[26] This function is provided by some XSLT processors, but has not been included in the 'official' specification of XSLT [21]. It belongs to the additional functions of the EXSLT ('**Ex**tensions to XSLT') project (for more details, see the Web page http://www.exslt.org).