



HAL
open science

mlBibTeX Architecture

Jean-Michel Hufflen

► **To cite this version:**

| Jean-Michel Hufflen. mlBibTeX Architecture. ArsTeXnica, 2006, pp.54–59. hal-00644462

HAL Id: hal-00644462

<https://hal.science/hal-00644462>

Submitted on 24 Nov 2011

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

MIBIB_TE_X's Architecture

Jean-Michel HUFFLEN

Abstract

This paper shows the architecture of MIBIB_TE_X, our reimplementation of BIB_TE_X focusing on multilingual features. Making precise the organisation and modules of this architecture allows us to show how MIBIB_TE_X works and focus on the differences between MIBIB_TE_X and BIB_TE_X from a conceptual point of view. We also explain why this implementation using the Scheme programming language allows users to scrutinise the result of intermediate steps.

Keywords L_AT_EX, BIB_TE_X, MIBIB_TE_X, bibliography files, bibliography styles, bst, nbst, XML, XSLT.

Sommario

Questo paper illustra l'architettura di MIBIB_TE_X, una reimplementazione di BIB_TE_X più adatta per lavorare in un contesto multilingua. Rendendo più precisa l'organizzazione ed i moduli è possibile mostrare come MIBIB_TE_X lavora evidenziando le principali differenze esistenti tra MIBIB_TE_X e BIB_TE_X da un punto di vista concettuale. Verrà inoltre spiegato perché quest'implementazione fatta usando il linguaggio di programmazione Scheme permette di verificare i risultati degli step intermedi.

Parole chiave L_AT_EX, BIB_TE_X, MIBIB_TE_X, bibliografia, stile bibliografico, bst, nbst, XML, XSLT.

0 Introduction

Often the L_AT_EX [15] word processor and BIB_TE_X [17] bibliography processor are used in conjunction. Such an approach allows the 'References' section of a printed document to be given nice layout. Homogeneous typesetting for items being same type is provided by means of bibliography styles ruling the layout of articles, books, etc. As mentioned in [16, § 13.1], BIB_TE_X has been stable for a long time. However, modern features, such as multilingual capabilities, have not been incorporated and are possible only by means of workarounds, most often by inserting L_AT_EX commands inside field values. As a consequence, using BIB_TE_X to generate bibliographies for another output format than L_AT_EX may be difficult.

There are several projects aiming to develop BIB_TE_X's successors [16, § 13.1.2]. Among them, MIBIB_TE_X—for 'MultiLingual BIB_TE_X'—focuses

on multilingual features. It uses XML¹ as a central formalism. Even if there is a compatibility mode for bibliography styles originating from BIB_TE_X [10], style designers are encouraged to develop new styles using a new language [4, App. A & B], nbst², close to XSLT³ [21], the language of transformations used for XML texts.

The broad outlines of MIBIB_TE_X have already been shown at the 2004 QJIT⁴ conference and given in [9]. This present paper focuses on MIBIB_TE_X's architecture, in the sense that we explain how MIBIB_TE_X's modules are organised and how they interact. First, we recall how BIB_TE_X proceeds. So we can point out the differences between BIB_TE_X and MIBIB_TE_X from a conceptual point of view. Reading this article only requires basic knowledge about L_AT_EX, BIB_TE_X and XML. About its implementation, MIBIB_TE_X is written using the Scheme programming language [8]. We will not go thoroughly into technical points in Scheme⁵, but mention how intermediate results are built.

1 Situation

1.1 BIB_TE_X's behaviour

As a recent reference, [16, § 12.1.3] explains how L_AT_EX and BIB_TE_X can cooperate. *Citation keys*—e.g., 'eco1980' in '\cite{eco1980}'—are stored by L_AT_EX into an auxiliary (.aux file) also used by BIB_TE_X. Now let us describe BIB_TE_X's behaviour roughly. First, it takes an .aux file⁶ and searches it for citations. This .aux file also contains the information about a bibliography style. Then this bibliography style file is run. If we look at some standard bibliography files, some variables, functions and macros are defined, bibliographic databases are searched for citation keys, and the accurate functions of the style are applied. Figure 1 describes the organisation of a bibliography style file as it is sketched in [16, § 13.6.2]. To sum up: once the citation keys and the bibliography style are known, the latter is run. The result is a .bbl file containing a thebibliography environment [16,

1. eXtensible Markup Language. Readers interested in an introductory book to this formalism can refer to [18].
2. New Bibliography STyles.
3. eXtensible Stylesheet Language Transformations.
4. Gruppo Utilizzatori Italiani di T_EX (Italian T_EX user group).
5. The current reference manual of this language is [13].
6. Let us recall that when BIB_TE_X is invoked by means of a command line such as 'bibtex (job-name)', (job-name) refers to an .aux file.

```

ENTRY {...}{...}{...}      % Declaration of variables and functions.
FUNCTION {...}{...}
...
FUNCTION {article}{...}
...
MACRO {oct}{"October"}
...
READ                        % Search bibliography data base .bib files for entries.
...                          % Some preprocessing, including the sort of selected entries.
SORT
...
EXECUTE {begin.bib}        % Preamble and \begin{thebibliography}.
EXECUTE {...}              % Some initialisations.
ITERATE {call.type$}      % Loop over entries producing outputs.
EXECUTE {end.bib}         % Write \end{thebibliography} command.

```

FIGURE 1: Bibliography style file used by BIBT_EX: framework and basic flow.

§ 12.1.2], each reference being prefixed by means of the `\bibitem` command.

1.2 Going further

At a first glance, a successor of BIBT_EX might conserve the same *modus operandi*. That is not always possible. BIBT_EX never reads the `.tex` file because the information it needs can be found in `.aux` files only. The same behaviour is unsuitable for MIBIBT_EX—as a program managing multilingual bibliographies—when it generates a ‘References’ section for a L^AT_EX source text. Let us consider the following title, which uses a syntactical extension provided by MIBIBT_EX:

```
TITLE = {[Danse macabre] : french}
```

This book of Stephen King is written in English—of course—but its title uses French words, what is specified by the ‘[...] : french’ notation, where ‘french’ is a *language identifier* [4]. Let us assume that we are building a ‘References’ section for a document in English. In order for L^AT_EX to use the right hyphenation patterns if need be, the `.bbl` file generated should include such a title as:

```
\foreignlanguage{french}{Danse macabre}
```

provided that the `babel` package is loaded with the `french` option when L^AT_EX processes this document [16, Ch. 9]. First, there are several ways to write in French with L^AT_EX with the `babel` package or with other *ad hoc* packages. Second, let us recall that all the languages used throughout a document must be specified as options when the `babel` package is loaded. This package is static and the languages it uses cannot be extended dynamically. As we explain in [7], there are several solutions to this problem, but we think that the best consists of analysing the preamble of the `.tex` file, in order to get information about the multilingual capability

chosen by a user. Let us go back to our title: if the `french` option has not been selected, MIBIBT_EX will just produce ‘Danse macabre’ and warn the user that some words may be incorrectly hyphenated.

Another drawback of bibliography styles of BIBT_EX is that they are monolithic: one file for one style. There is a tool, `custom-bib` [16, § 13.5.2], that allows a bibliography style to be build automatically from users’ wishes. But there is no modular approach in the sense that a style designer could develop a new style from existing style fragments. In MIBIBT_EX, a style is assembled dynamically, according to what is needed. For example, let us consider that a `plain` style⁷ is to be used for a document written in English and including some fragments in Italian. The files that can be used are:

`plain.nbst` general definitions for this style.

`plain-{english,italian}.nbst` definitions suitable for the English and Italian languages and belonging to this style.

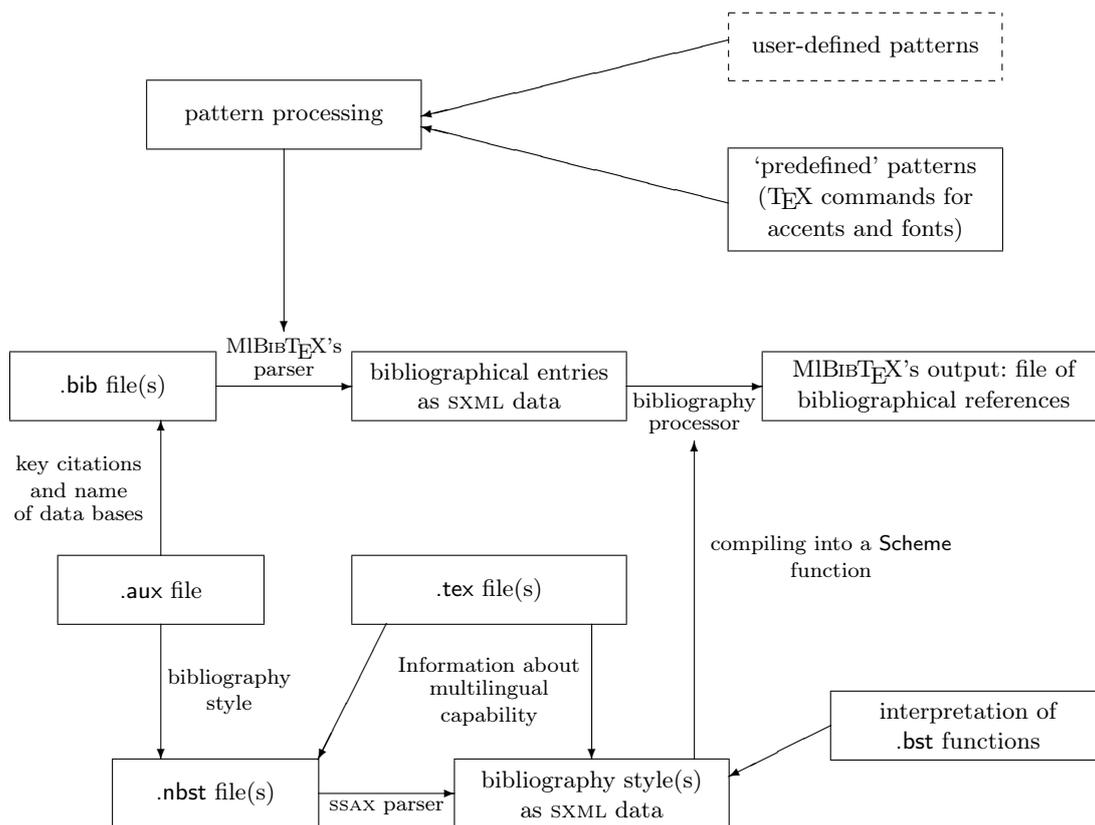
`{-english,-italian}.nbst` general definitions for the English and Italian languages.

So we need this information about available languages before applying a style.

There is another difference, which can be noticed by an end-user, about abbreviations usable in a `.bib` file. Some may be defined using the `@STRING{...}` command of BIBT_EX [16, § 13.2.3], some may be defined as macros within the bibliography style [16, Table 13.7], for example, month names:

```
MONTH = oct
```

7. That is, items are labelled by numbers.



' $A \leftarrow B$ ' means that A uses B . More precisely, functions or data put in A use functions of B or data from B . A dashed box means that the corresponding module is planned, but not fully implemented yet.

FIGURE 2: Data flow in MIBIB_{TEX}.

(cf. Figure 1). When BIB_{TEX} reads a .bib file, all the abbreviations must be expanded into strings. That is, all must be known at this time. On the contrary, in the styles of MIBIB_{TEX}, you can perform different operations whether or not a symbol is defined:

```
<nbst:choose>
  <nbst:when test="is-defined('roma')">
    ...
    <!-- The is-defined function does not exist
         in XSLT, but exists in nbst.
    -->
  </nbst:when>
  ...
</nbst:choose>
```

We think that all these differences between MIBIB_{TEX} and BIB_{TEX} are improvements but they lead to a redesign of the data flow in MIBIB_{TEX}.

2 Organisation of MIBIB_{TEX}

The organisation of MIBIB_{TEX} is pictured at Figure 2. Like BIB_{TEX}, MIBIB_{TEX} reads an .aux file and gets a list of citation keys. It also gets infor-

mation about some bibliography data base files and a bibliography style.

Then it parses .bib files, searching them for the citation keys. This step results in an SXML⁸ tree, that is, a representation of an XML tree using Scheme [14]. Let us consider that the entry given in Figure 3 is to be put within a bibliography. Conceptually, it can be viewed as the XML tree given in Figure 4, represented in SXML by the Scheme lists of Figure 5. Bibliography styles use XPath⁹ expressions related to XML trees resulting from parsing a .bib file, that is, XML trees comparable with the example given in Figure 4.

Some symbols are not expanded because they have not been defined by a @STRING command in a .bib file—e.g., the roma symbol in Figure 3, see also Appendix A—and are retained for future processing. In other words, that is a kind of 'benefit of the doubt'. The commands generating accented letters and other signs belonging to the Latin 1 encoding are expanded: for exam-

8. Scheme implementation of XML.

9. XPath is the language used to address parts of an XML document [20].

```
@BOOK{eco1980,
  AUTHOR = {Umberto Eco},
  TITLE = {Il nome della rosa},
  PUBLISHER = {Bompiani},
  ADDRESS = roma,
  YEAR = 1980,
  LANGUAGE = italian}
```

Notice the `LANGUAGE` field and language identifier used to specify this entry for a book written in Italian.

FIGURE 3: MIBIB_TE_X's bibliographical entry.

ple, `\'e` is expanded to the accented letter 'é' of this encoding¹⁰. As a consequence, the accented and special letters included in the values of MIBIB_TE_X's fields are viewed as single letters when entries are sorted. Besides, this sort operation can be performed w.r.t. a language's lexicographical order. The `nbst` language provides a `nbst:sort` element [4, App. A] that is analogous to the `xsl:sort` element of XSLT [21, § 10]: if the data to be sorted are strings, a `language` attribute allows sorting w.r.t. a particular language (English, Italian, Spanish, ...)

Likewise, some predefined commands of L_AT_EX for fonts (`\emph`, `\textbf`, ...) are recognised and expanded to XML elements. This is done by means of patterns, expressed in Scheme. Now, only some predefined patterns are processed, e.g.:

```
(define-pattern "\'e" "é")
(define-pattern "\'e}" "é")
```

we plan to extend this capability to user-defined patterns, so users will be able to mark up their `.bib` files, and tell MIBIB_TE_X how it can understand this markup [5].

During the next step, MIBIB_TE_X deduces the source `.tex` file's name from the `.aux` file's¹¹. Then it parses the preamble of this file and try to know:

- which languages are used throughout this document and how;
- which encoding may be used: pure ASCII¹² or Latin 1. So, users can type accented letters directly inside field values of `.bib` files processed by MIBIB_TE_X, even if they do not use the `inputenc` package of L_AT_EX, for dealing with encodings other than ASCII [16, § 7.5.2]. In this last case, the commands for generating accents will be used, for example, 'é' will be transformed into `\'e`.

10. Now, most Scheme interpreters can only deal with the Latin 1 encoding. Future versions should be Unicode-compliant [19] and able to deal with more encodings, such as Latin 2, UTF-8, ...

11. You can specify this `.tex` file's name by means of an option of MIBIB_TE_X, if need be.

12. American Standard Code for Information Interchange.

```
<book id="eco1980" language="italian">
  <author>
    <name>
      <personname>
        <first>Umberto</first>
        <last>Eco</last>
      </personname>
    </name>
  </author>
  <title>Il nome della rosa</title>
  <publisher>Bompiani</publisher>
  <year>1980</year>
  <address><symbol name="roma"/></address>
</book>
```

FIGURE 4: Entry as an XML tree (Fig. 3 cont'd).

This step results in:

- a list of possible file names¹³ whose union is the multilingual bibliography style;
- updating a set of Scheme functions controlling MIBIB_TE_X's output.

Then the files of the bibliography style are parsed by means of SSAX¹⁴ [14]. This parser returns the SXML representation of an XML tree. So, the result is comparable to the expression given in Figure 5, but concerns the elements of a bibliography style. Finally, the fragments of a bibliography style are assembled and compiled into a Scheme function¹⁵ [12]. Then this function is applied to the list of bibliographical entries selected.

Last, bibliography styles may use functions written using the `bst` language of B_IB_TE_X. That allows 'old' bibliography styles to be replaced progressively and explains the arrow, in Figure 2, from the box concerning `bst` functions to the box concerning bibliography styles expressed using the new language `nbst`.

3 Conclusion

When we started to develop MIBIB_TE_X's first version [2], we only aimed to provide a 'better B_IB_TE_X'. The only changes from end-users' point of view were additional syntactic features. This approach was not satisfactory enough and we decided to design a new language for bibliography styles [3], taking advantage of XML features as far as possible. This design choice and the implementation of multilingual features led to an

13. 'Possible files' because some files specific to a particular language may be inexistent, in which case they are ignored.

14. Scheme implementation of SAX (Simple API for XML).

15. Displaying a function's body is difficult and often not provided by most Scheme interpreters, that is why we do not give more details about this step. But we plan to develop tools for browsing bibliography styles.

```
(*top* ...
(book (@ (id "eco1980") (language "italian"))
(author
(name (personname (first "Umberto")
(last "Eco"))))
(title "Il nome della rosa")
(publisher "Bompiani") (year "1980")
(address (symbol (@ (name "roma"))))))))
```

FIGURE 5: Entry in SXML (Fig. 4 cont'd).

architecture quite far from BIB \TeX 's. It may appear as more complicated, but we think that the role of each module is precise and can be understood by end-users, provided that they have got first experience with the Scheme programming language. But this architecture seems to us to be stable and robust. In particular, we succeeded in using MIBIB \TeX for deriving 'References' sections for Con \TeX t [11], another format built out of \TeX ¹⁶ and very different from L \TeX . So, extending MIBIB \TeX to other output formats is possible.

A Abbreviations in MIBIB \TeX

This appendix aims to make precise how abbreviations—possibly multilingual—can be specified in MIBIB \TeX . As an example, we consider the `roma` symbol, that appears in the `eco1980` entry of Figure 3. Another example is given in [6].

You can use a `@STRING` command, like in 'old' BIB \TeX [16, § 13.2.3]:

```
@STRING{roma = {[Roma] * italian
[Rome] * english
[Rom] * german}}
```

the sequence of '`[...] * ...`' notations being a syntactical extension of MIBIB \TeX for information that *must* be put, possibly in another language if no translation is available [4]. In other words, processing such a sequence never yields an empty string. But this `@STRING` command is expanded *verbatim* when a `.bib` file is processed—as in 'old' BIB \TeX —so it must be put in every `.bib` file where this `roma` symbol is used.

A better solution, from our point of view, is to define this symbol like month names or predefined names of journals within the standard `bst` styles of BIB \TeX . If the `roma` symbol is not defined by a `@STRING` command, it is processed as we show in Figure 4 when a `.bib` file is parsed. It can be defined within a bibliography style—that is, within a `.nbs` file—as follows:

16. \TeX provides a powerful framework to format texts nicely, but to be fit for use, the definitions of this framework need to be organised in a *format*. The first formats were *Plain \TeX* and L \TeX , another format, more modern, is Con \TeX t [1].

```
<nbst:variable name="roma">
  <nonemptyinformation>
    <group language="italian">Roma</group>
    <group language="english">Rome</group>
    <group language="german">Rom</group>
  </nonemptyinformation>
</nbst:variable>
```

These two ways to define a symbol are close together, since a sequence of '`[...] * ...`' yields a `nonemptyinformation` element when `.bib` files are parsed. However, if an abbreviation has not been defined in 'old' BIB \TeX , a warning message is issued and the undefined symbol is just replaced by an empty string. As mentioned in § 1.2, the `nbst` language allows you to check that a symbol has been defined or not.

Acknowledgements

Many thanks to Maurizio W. Himmelmann for his patience when he was waiting for the first version of this article and for its Italian translation of the abstract. I also thank the referee of this article for the improvements suggested to me.

References

- [1] Hans HAGEN: *Con \TeX t, the Manual*. November 2001. <http://www.pragma-ade.com/general/manuals/cont-enc.pdf>.
- [2] Jean-Michel HUFFLEN: "MIBIB \TeX : a New Implementation of BIB \TeX ". In: Simon PEPPING, ed., *Euro \TeX 2001*, pp. 74–94. Kerkrade, The Netherlands. September 2001.
- [3] Jean-Michel HUFFLEN: "European Bibliography Styles and MIBIB \TeX ". *TUGboat*, Vol. 24, no. 3, pp. 489–498. Euro \TeX 2003, Brest, France. June 2003.
- [4] Jean-Michel HUFFLEN: "MIBIB \TeX 's Version 1.3". *TUGboat*, Vol. 24, no. 2, pp. 249–262. July 2003.
- [5] Jean-Michel HUFFLEN: "MIBIB \TeX : beyond L \TeX ". In: Apostolos SYROPOULOS, Karl BERRY, Yannis HARALAMBOUS, Baden HUGUES, Steven PETER and John PLAICE, eds., *International Conference on \TeX , XML, and Digital Typography*, Vol. 3130 of LNCS, pp. 203–215. Springer, Xanthi, Greece. August 2004.
- [6] Jean-Michel HUFFLEN: "Making MIBIB \TeX Fit for a Particular Language. Example of the Polish Language". *Biuletyn GUST*, Vol. 21, pp. 14–26. 2004.
- [7] Jean-Michel HUFFLEN: *Managing Languages within MIBIB \TeX* . To appear. June 2005.

- [8] Jean-Michel HUFFLEN: “Implementing a Bibliography Processor in Scheme”. In: J. Michael ASHLEY and Michel SPERBER, eds., *Proc. of the 6th Workshop on Scheme and Functional Programming*, Vol. 619 of *Indiana University Computer Science Department*, pp. 77–87. Tallinn. September 2005.
- [9] Jean-Michel HUFFLEN: “MiBIBT_EX: a Survey”. In: *Proc. GUIT Meeting*, pp. 171–179. Pisa, Italy. October 2005.
- [10] Jean-Michel HUFFLEN: “BiB_T_EX, MiBIBT_EX and Bibliography Styles”. *Biuletyn GUST*, Vol. 23, pp. 76–80. In *Bachot_EX 2006 conference*. April 2006.
- [11] Jean-Michel HUFFLEN: “MiBIBT_EX Meets ConT_EXt”. In: *EuroT_EX 2006 conference, preprints*, pp. 71–76. Debrecen, Hungary. July 2006.
- [12] Jean-Michel HUFFLEN: “Implementing a variant of XSLT in Scheme”. In: *Proc. IFL’06*, pp. 484–491. Budapest, Hungary. September 2006.
- [13] Richard KELSEY, William D. CLINGER, Jonathan A. REES, Harold ABELSON, Norman I. ADAMS IV, David H. BARTLEY, Gary BROOKS, R. Kent DYBVIG, Daniel P. FRIEDMAN, Robert HALSTEAD, Chris HANSON, Christopher T. HAYNES, Eugene Edmund KOHLBECKER, JR, Donald OXLEY, Kent M. PITMAN, Guillermo J. ROZAS, Guy Lewis STEELE, JR, Gerald Jay SUSSMAN and Mitchell WAND: “Revised⁵ Report on the Algorithmic Language Scheme”. *HOSC*, Vol. 11, no. 1, pp. 7–105. August 1998.
- [14] Oleg E. KISELYOV: *XML and Scheme*. September 2005. <http://okmij.org/ftp/Scheme/xml.html>.
- [15] Leslie LAMPORT: *L^AT_EX: A Document Preparation System. User’s Guide and Reference Manual*. Addison-Wesley Publishing Company, Reading, Massachusetts. 1994.
- [16] Frank MITTELBACH and Michel GOOSSENS, with Johannes BRAAMS, David CARLISLE, Chris A. ROWLEY, Christine DETIG and Joachim SCHROD: *The L^AT_EX Companion*. 2nd edition. Addison-Wesley Publishing Company, Reading, Massachusetts. August 2004.
- [17] Oren PATASHNIK: *BiB_T_EXing*. February 1988. Part of the BiB_T_EX distribution.
- [18] Erik T. RAY: *Learning XML*. O’Reilly & Associates, Inc. January 2001.
- [19] THE UNICODE CONSORTIUM: *The Unicode Standard Version 4.0*. Addison-Wesley. August 2003.
- [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. Edited by James Clark. November 1999. <http://www.w3.org/TR/1999/REC-xslt-19991116>.

▷ Jean-Michel HUFFLEN
LIFC (FRE CNRS 2661)
University of Franche-Comté
16, route de Gray
25030 BESANÇON CEDEX
FRANCE
hufflen@lifc.univ-fcomte.fr