MlBibTeX in Scheme

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

We present the main functions of MlBibTeX's implementation using Scheme. In particular, that allows us to see how the modules are organised and how to run the different parts of MlBibTeX step by step. Let us recall that MlBibTeX deals with several data formats (syntaxes w.r.t. TeX, bibliography files, XML) and we show how such coexistence is managed.
Keywords MlBibTeX, Scheme, managing data formats.

Streszczenie

Pokazujemy główne funkcje implementacji MlBibTeX-a używającej Scheme. Przede wszystkim umożliwia nam to podgląd jak są zorganizowane moduły i jak uruchamiać różne częśi MlBibTeX-a krok po kroku. Przypomnijmy, że MlBibTeX obsługuje kilka formatów danych (składnie w odniesieniu do: TeX-a, plików bibliograficznych, XML-a) i pokazujemy jak to osiągnąć.
Słowa kluczowe MlBibTeX, Scheme, zarządzanie formatami danych.

Introduction

MlBibTeX — for '**M**ulti**L**ingual BibTeX' — is a reimplementation of BibTeX [19], the bibliography processor associated with the LaTeX word processor [15]. It extends BibTeX about multilingual features. As explained in [6], MlBibTeX's present version is based on XML[1] in the sense that parsing bibliography (.bib) files result in an XML tree. MlBibTeX can use bibliography styles written using the bst language of BibTeX [18] in compatibility mode [5]. However it is preferable for MlBibTeX bibliography styles to take as much advantage of nbst — for '**N**ew **B**ibliography **ST**yles' — as possible; we show how to proceed in [10]. This 'new' language is an extension of XSLT[2] [24] with a kind of inheritance on language expressions [6, 7].

In [8], we explained why we started up a new implementation using the Scheme programming language, after a first try using the C programming language [13]. We describe how to install and use this implementation hereafter. In the second section ('MlBibTeX in Scheme'), readers are assumed to be quite familiar with the Scheme language and can refer to [3, 22] for more details.

[1] e**X**tensible **M**arkup **L**anguage. Readers interested in an introduction it can refer to [21].

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

Disclaimer

The information given hereafter about MlBibTeX's *installation* is subject to change, because we are still currently working on that. At the time of writing, MlBibTeX's *Web* page:

`http://lifc.univ-fcomte.fr/~hufflen/texts/mlbibtex/mlbibtex/`

is still under construction, but more details about the installation procedure and its possible improvement will be reported over there.

Installation

Requirements As we explained in [9], to use the present version of MlBibTeX (1.3), you need a working and recent version of (LA)TeX, of course, including a recent version of the babel package [17, Ch. 9]. Some *ad hoc* packages are also in interface with our program: french [4], german [20], polski [2, § F7]. MlBibTeX's specific requirements are:

```
(define bibliographystyle-pv
  (let ((the-bibliographystyle #f))
    (lambda (msg)
      (cond ((eq? msg 'see) (lambda ()
                              (cond (the-bibliographystyle)
                                    ;; msg-manager is the function managing the display of
                                    ;; messages. It returns #f.
                                    (else (msg-manager 'no-bibliographystyle)))))
            ((eq? msg 'set)
             (lambda (stylename)
               (cond (the-bibliographystyle (msg-manager 'bibliographystyle-already-set))
                     (else (set! the-bibliographystyle stylename)
                           stylename)))))))))
```

Figure 1: A protected variable in MlBibTeX.

- an R5RS-compliant Scheme interpreter[3] or compiler;
- to install the SXML[4] library [14], available at the *Web* address

  `http://pair.com/lisovsky/xml/ssax`

  (`ssax-sxml` is the better choice).

We have tested MlBibTeX:

- with MIT Scheme and bigloo as Scheme interpreters;
- on Linux SuSE and Red Hat.

Distribution It consists of five directories:

doc contains the documentation (still under construction);

latex groups the files containing the definition of additional LaTeX commands, in order for this word processor to be able to process the files generated by MlBibTeX; examples of such files are given in [9]; notice that when MlBibTeX's installation is finished, you have to add this directory to the specification of the `TEXINPUTS` environment variable, in order for LaTeX to be able to find these files;

nbst its subdirectories contain the predefined bibliography styles; most of current styles of BibTeX have been translated [10]; the organisation of the different files for a bibliography style is explained in [9, 10];

obj the place where object files are placed when the source files are compiled;

[3] ... and not R4RS-compliant, that is, based on [1]. MlBibTeX uses some new features of the last revision: hygienic macros, functions returning multiple values and the `dynamic-wind` function.

[4] Scheme implementation of XML.

src contains the source files written in Scheme.

Some additional files are given:

configure.in configure Makefile.in

Now they are configured to use MIT Scheme as a compiler. In such a case, the installation is 'classical' for an UNIX-like system:

-
  ```
  ./configure --prefix=...\
    --with-sxml-library=...
  ```
  `prefix` (resp. `with-sxml-library`) being set to the directory where MlBibTeX's distribution (resp. SXML library) has been put, both default to the /usr/local directory,
- `make`
  compiles the files of the src directory and builds an executable file mlbibtex, launching the main function,
- `make install`
  installs the mlbibtex file in a public directory.

You can use this executable file as follows:

`mlbibtex` *`job-name`*

where '*`job-name`*' is the name — with or without suffix — of an auxiliairy (.aux) file. You can force the use of the language of a document by:

`mlbibtex` *`job-name`* `--language=...`

but we do not recommend this feature: multilingual functions are not used, so some parts of the resulting text can be processed incorrectly by LaTeX. More generally, how languages are managed within MlBibTeX will be described in [11].

Using source files in interpreted mode This way should work with any Scheme interpreter, it should also work on the Windows operating system.

- Edit the file src/config.scm and put the right values for the variables:

```
@INPROCEEDINGS{zemianski2002,
        AUTHOR = {first => Andrzej,
                  last => Zemia\'{n}ski},
        TITLE = {Waniliowe plantacje
                 Wroc{\l}awia},
        BOOKTITLE = {Zajdel 2002},
        EDITOR = {},
        PAGES = {99--164},
        PUBLISHER = {Fabryka Sl\'{o}w},
        ADDRESS = {Lublin},
        NOTE = {[Not yet translated in
                 English] ! english},
        YEAR = 2002,
        LANGUAGE = polish}
```

Figure 2: Example of MlBibTEX's entry.

pl-mlbibtex the absolute address where the distribution of MlBibTEX is located,

pl-sxml-library the absolute address where SXML library is located.

- Launch a Scheme interpreter and load[5] the file src/pilot.scm.
- Now you can use the functions described in the next section.

MlBibTEX in Scheme

Protected variables As far as possible, we want to avoid direct side effects, that is, using the special form `set!` at the top level. We take advantage of *lexical closures* and *unlimited extent* in Scheme, and use **protected variables**, close to objects within an object-oriented approach. An example of such a variable is given in Figure 1: we send messages to the `bibliographystyle-pv` variable to see and set the bibliography style used. We can see that this style can be set only once, the side effect being enclosed in the value of `bibliographystyle-pv`.

Here are information that is managed this way:

- the bibliography style,
- the name of the '`log`' file for a job,
- the list of BibTEX keys cited throughout the document whose we are building the 'References' section: `bibtexkey-list-pv`,
- the list of bibliography styles to be searched: `bibfile-list-pv`.

So, if you consider the MlBibTEX entry given in Figure 2 and would like to add it to the list of keys cited, unless it has already been included, just type:

[5] That is, use the Scheme function `load`.

```
<mlbiblio>
  <inproceedings id="zemianski2002"
                 language="polish">
    <author>
      <name>
        <personname>
          <first>Andrzej</first>
          <last>Zemiański</last>
        </personname>
      </name>
    </author>
    >title>
      Waniliowe plantacje
      <asitis>Wrocławia</asitis>
    </title>
    <booktitle>Zajdel 2002</booktitle>
    <publisher>Fabryka Słów</publisher>
    <year>2002</year>
    <pages>
      <firstpage>99</firstpage>
      <lastpage>164</lastpage>
    </pages>
    <note>
      <group language="english">
        Not yet translated in English
      </group>
    </note>
  </inproceedings>
  ...
</mlbiblio>
```

Figure 3: The entry of Figure 4, using XML-like syntax.

```
((bibtexkey-list-pv 'adjoin)
 "zemianski2002")
```

and this expression returns the updated list of keys cited. Similarly, evaluate:

```
((bibtexkey-list-pv 'remove)
 "zemianski2002")
```

if you would like this key to be removed from the list.

Prefixes for modules A drawback of Scheme is the absence of modules[6] or *packages*, w.r.t. the termonology of Lisp[7]. That is why we are especially careful to add a prefix to our functions' names. Non-prefixed names are:

[6] Some interpreters provide them, but they have not been included in standard Scheme.

[7] **LIS**t **P**rocessing. Lisp dialects — including Scheme — are the successors of the language designed by John McCarthy [16].

```
(*top* (mlbiblio (inproceedings (@ (id "zemianski2002") (language "polish"))
                                (author (name (personname (first "Andrzej")
                                                          (last "Zemia\'{n}ski"))))
                                (title "Waniliowe plantacje Wroc{\l}awia")
                                (booktitle "Zajdel 2002") (publisher "Fabryka Słów")
                                (year "2002") (address "Lublin")
                                (pages (firstpage "99") (lastpage "164"))
                                (note (group (@ (language "english"))
                                             "Not yet translated in English")))
                 ...))
```

Figure 4: What MlBibTeX's parser results in.

- the name of local variables and functions,
- the names of some functions and macros of general interest, that is, usable outside MlBibTeX (they are grouped in the file src/common.scm),
- the names of protected variables (but they end with '-pv', as shown by the abovementioned examples.

For example, all the functions of our parser of .bib files (resp. TeX files) begin with 's-' (resp. 't-').

Using MlBibTeX Most often, MlBibTeX's main function can be used by:

`(mlbibtex` *job-name*`)`

more generally by:

`(mlbibtex` *job-name* `.` *alist*`)`

when 'a-list' is an associatlion list whose keys are interface keywords for MlBibTeX, for example:

`(mlbibtex` *job-name* `'(language . "polish"))`
— compare this expression to the second example given in Subsection 'Distribution' — this convention is close to the keywords used in COMMON LISP [23, § 5.2.2] or [12, § 8.3.1.4].

Parsers MlBibTeX uses the SSAX[8] parser, included in SXML [14]. It can be used by:

```
(define an-sxml-tree
  (call-with-input-file input-file
    (lambda (input-p)
      (SSAX:XML->SXML input-p '()))))
```

and, as described in [14], we can asj for a linear list grouping all the parts addressed by an XPath expression:

```
((sxpath an-XPath-expression)
 an-xsml-tree)
```

There are two other parsers.

[8] Scheme implementation of SAX (Simple API for XML, cf. [21, pp. 289–291]).

- the parser of .bib files, resulting in SXML trees:

  `(s-parse-bib-file-list` *bib-file-list*`)`

  uses the value enclosed by the protected variable `bibtexkey-list-pv` to match the right entries. For one .bib file, the function to call is:

  `(s-parse-bib-file` *bib-file*`)`

- the parsers of files written w.r.t. TeX's syntax, they are used to parse .aux files:

  `(t-parse-aux-file` *aux-filename*`)`

  and to parse the preamble of a source file, in order to know which multilingual packages are used:

  `(t-parse-tex-preamble` *tex-filename*`)`

These both parsers are derived from a common basis sketched in Figure 5.

If `bibtexkey-list-pv` contains an empty list of keys, the complete list of entries is returned. If you consider the entry given in Figure 2, the result of our parser is displayed in Figure 4. To display it using an XML-like syntax, do:

`((xml-file 'from-sxml-tree)` *an-sxml-tree*`)`

Acknowledgements

Many thanks to Paweł D. Mogielnicki, who has written the Polish translation of the abstract.

References

[1] 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 KOHBECKER, JR., Donald OXLEY, Kent M. PITMAN, Guillermo Juan ROZAS, Guy Lewis STEELE, JR., Gerald Jay SUSSMAN and Mitchell WAND:

```
(define (parsers-make-launching filename launcher)
  ;; launcher is the function that rules the analysis of the input file. Its arguments are the function
  ;; going forward through the file and the function managing errors.
  (call-with-current-continuation (lambda (parser-exit-c)
                                     (parsers-filename-rp-loop filename launcher
                                                               parser-exit-c))))

(define (parsers-filename-rp-loop filename launcher parser-exit-c)
  (let ((input-p '*dummy-value*))
    (dynamic-wind
      ;; Even if the launcher function encounters errors, the input port is closed.
      (lambda () (set! input-p (open-input-file filename)))
      (lambda () (launcher (make-r-thunk input-p) parser-exit-c))
      (lambda () (close-output-port input-p)))))

(define (make-r-thunk input-p)
  ;; The result is a thunk (0-argument function) that moves forward through the input file.
  (lambda () (read-char input-p)))

(define (make-x-function parser-exit-c)
  ;; The result is a function that displays an error message, and stops reading the input file.
  (lambda (msg-idf)
    (msg-manager msg-idf)
    (parser-exit-c #f)))
```

Figure 5: Basic functions to build MlBibTEX's parsers.

"Revised Report[4] on the Algorithmic Language Scheme". ACM *Lisp Pointers*, Vol. 4, no. 3. July 1991.

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

[3] R. Kent Dybvig: *The Scheme Programming Language*. ANSI *Scheme*. 2nd edition. Prentice-Hall. 1996.

[4] 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.

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

[6] Jean-Michel Hufflen: "European Bibliography Styles and MlBibTEX". TUG*boat*, Vol. 24, no. 3, p. 489–490. EuroTEX 2003, Brest, France. June 2003.

[7] Jean-Michel Hufflen: "MlBibTEX's Version 1.3". TUG*boat*, Vol. 24, no. 2, p. 249–262. July 2003.

[8] Jean-Michel Hufflen: "A Tour around MlBibTEX and Its Implementation(s)". *Biuletyn* GUST, Vol. 20, p. 21–28. In *BachoTEX 2004 conference*. April 2004.

[9] Jean-Michel Hufflen: "Making MlBibTEX Fit for a Particular Language. Example of the Polish Language". *Biuletyn* GUST, Vol. 21, p. 14–26. 2004.

[10] Jean-Michel Hufflen: "Bibliography Styles Easier with MlBibTEX". In: *EuroTEX 2005 conference, program and preprints*, p. 106–119. Pont-à Mousson, France. March 2005.

[11] Jean-Michel Hufflen: *Managing Languages within MlBibTEX*. Will be presented at PracTEX conference, Chapel Hill, North Carolina. June 2005.

[12] International Standard ISO/IEC 10179:1996(E): DSSSL. 1996.

[13] Brian W. Kernighan and Denis M. Ritchie: *The C Programming Language*. 2nd edition. Prentice Hall. 1988.

[14] Oleg Kiselyov and Kirill Lisovsky: "XML, XPath, XSLT Implementations as SXML, SXPath, and SXSLT". In: *International Lisp Conference 2002*. San Francisco, California. October 2002.

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

[16] John MCCARTHY: "Recursive Functions of Symbolic Expressions and Their Computation by Machine, Part I". *Communications of the ACM*, Vol. 3, no. 4, p. 184–195. April 1960.

[17] Frank MITTELBACH, Michel GOOSSENS, Joannes BRAAMS, David CARLISLE, Chris A. ROWLEY, Christine DETIG and Joachim SCHROD: *The LATEX Companion.* 2nd edition. Addison-Wesley Publishing Company, Reading, Massachusetts. August 2004.

[18] Oren PATASHNIK: *Designing BIBTEX Styles.* February 1988. Part of BIBTEX's distribution.

[19] Oren PATASHNIK: *BIBTEXing.* February 1988. Part of BIBTEX's distribution.

[20] 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.

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

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

[23] Guy Lewis STEELE, JR., Scott E. FAHLMAN, Richard P. GABRIEL, David A. MOON, Daniel L. WEINREB, Daniel Gureasko BOBROW, Linda G. DEMICHIEL, Sonya E. KEENE, Gregor KICZALES, Crispin PERDUE, Kent M. PITMAN, Richard WATERS and Jon L WHITE: COMMON LISP. *The Language. Second Edition.* Digital Press. 1990.

[24] W3C: XSL *Transformations (*XSLT*). Version 1.0.* W3C Recommendation. Edited by James Clark. November 1999. `http://www.w3.org/TR/1999/REC-xslt-19991116`.