Managing Order Relations in MlBibTeX*

Jean-Michel HUFFLEN
LIFC (EA CNRS 4157)
University of Franche-Comté
16, route de Gray
25030 BESANÇON CEDEX
FRANCE
hufflen@lifc.univ-fcomte.fr
http://lifc.univ-fcomte.fr/~hufflen

Abstract

Lexicographical order relations used within dictionaries are language-dependent. First, we describe the problems induced within automatic generation of multilingual bibliographies. Second, we explain how these problems are handled within MlBibTeX. To add or update an order relation for a particular natural language, we have to program in Scheme, but we show that MlBibTeX's environment eases this task as far as possible.

Keywords Lexicographical order relations, dictionaries, bibliographies, collation algorithm, Unicode, MlBibTeX, Scheme.

Streszczenie

Porządek leksykograficzny w słownikach jest zależny od języka. Najpierw omówimy problemy powstające przy automatycznym generowaniu bibliografii wielojęzycznych. Następnie wyjaśnimy, jak są one traktowane w MlBibTeX-u. Dodanie lub zaktualizowanie zasad sortowania dla konkretnego języka naturalnego umożliwia program napisany w języku Scheme. Pokażemy, jak bardzo otoczenie MlBibTeX-owe ułatwia to zadanie.

Słowa kluczowe Zasady sortowania leksykograficznego, słowniki, bibliografie, algorytmy sortowania leksykograficznego, Unikod, MlBibTeX, Scheme.

0 Introduction

Looking for a word within a dictionary or for a name in a phone book is common task. We get used to the lexicographic order for a long time. More precisely, we get used to *our own* lexicographic order, because it belongs to our cultural background. It depends on languages.

This problem is particularly acute when we deal with managing multilingual bibliographies, as in our program MlBibTeX — for '**M**ulti**L**ingual BibTeX'. Let us recall that this program aims to be a 'better' BibTeX [15], the bibliography processor usually associated with the LaTeX word processor [12]. When it builds a 'References' section for a LaTeX document, BibTeX uses a bibliography style ruling the layout of this section. Some bibliography styles are *unsorted*, that is, the order of bibliographical items within the bibliography is the order of first citations of these items throughout the document. However, most of BibTeX's styles sort these items w.r.t. the alphabetical order of authors' names. But the bst language of bibliography styles [14] only provides a `SORT` function [13, Table 13.7] suitable for the English language, the commands for accents and other diacritical signs being ignored by this sort operation.

The purpose of this article is to show how this problem is solved in MlBibTeX's first public version. In practice, this version only deals with European languages using the Latin alphabet. Besides, the MlBibTeX program is written using the Scheme programming language [10]. Some implementations provide partial support for Unicode [22], a proposal for a future standardised library has been established [18, §§ 1.1 & 1.2], but we cannot say that the present version of Scheme is Unicode-compliant.[1]

* Title in Polish: *Zarządzanie zasadami sortowania leksykograficznego w MlBibTeX-u*.

[1] At the time of writing the revised version of this article, the proposal for Scheme's next standard [19, 18] has been

So some parts of our present implementation of order relations are temporary, but we think that this implementation could be easily updated for future Unicode-compliant versions.

In a first section, we show how diverse lexicographic orders used throughout European countries are. This allows readers to estimate this diversity and to realise the complexity of this task. We also explain why this problem is made more difficult when we consider it within the framework of bibliographies. Then we show how order relations operate in MlBibTeX and how they are built. We also give some details about the common and different points betwen `xindy` [13, § 11.3] and MlBibTeX, these two programs using multilingual order relations.

Reading this article does not require advanced knowledge of Scheme;[2] in fact, we think that a non-programmer should be able to specify a new order relation. We give more technical details in an annex, for users that would like to do more experiment themselves. In particular, we explain how to deal with languages using the Latin 2 encoding, even if our implementation is based on Latin 1.

1 European languages and lexicographic orders

Figure 1 gives an idea about the diversity of order relations used throughout some European countries. In this figure, '$a < b$' denotes that the words beginning with a are less than the words beginning with b, whereas '$a \sim b$' expresses that the letters a and b are interleaved, except that a takes precedence over b if two words differ only by these two letters.

Roughly speaking, there are two family of languages, if we consider the associated lexicographic orders. In some languages, accented letters are fully viewed as 'real' letters, distinct from unaccented ones: examples are given by Slavonic languages. In other languages, accented letters are processed as if there is no accent. The precedence of a unaccented letter over an accented one is not managed the same way: it follows a left-to-right order in Irish, Italian, and Portuguese, a right-to-left order in French. The Estonian language 'mixes' the two approaches: some accented letters — '`õ`', '`ä`' — are alphabeticised, some — '`š`', '`ž`' — are interleaved. Last, some letter groups may be viewed as a single letter and alphabeticised as another letter. For example, the Hungarian words beginning with '`cs`' are alphabeticised separately from the words beginning with '`c`'. In fact, the '`c-`' entry, in a Hungarian dictionary, contains words beginning with '`c`' and not with '`cs`'. The '`c-`' entry is followed by the '`cs-`' entry, before the '`d-`' entry.

Anyway, it clearly appears that there cannot be a universal order, encompassing all lexicographic orders. Besides, these orders aim to classify words of a dictionary, that is, common words belonging to a language, even if some dictionaries may include some proper names. When bibliographies are generated, order relations are used to sort bibliographical items, most often w.r.t. authors' names. These names may be 'foreign' proper names if we consider the language used for the bibliography. So names can include characters outside of this language's alphabet. As a consequence, an order relation for sorting a bibliography should be able to deal with any letter, since any letter may appear in foreign names. A good choice is to associate such a foreign letter with a letter belonging to the 'basic' Latin alphabet, so this foreign letter is interleaved with this basic letter, which takes precedence over the foreign letter if two words differ only by these two letters. If we consider the English language, this means that accented letters are interleaved with unaccented letters, but unaccented letters take precedence. So proceed most of implementation of order relations.

Unicode provides a default algorithm to sort all its characters. This algorithm is based on a sort key table, DUCET[3] [23]. It is also based on a decomposition property for composite characters. For example, the '`ô`' letter, whose name and code point — given using hexadecimal numbers — are:

LATIN SMALL LETTER O WITH CIRCUMFLEX, U+00F4

can be decomposed into these 'simpler' characters:

LATIN SMALL LETTER O, U+006F
COMBINING CIRCUMFLEX ACCENT, U+0302

The sort algorithm requires several passes. To describe it roughly, an information about *weight*, given by *sort keys*, is associated with each string. Then this information is re-arranged according to sort levels, w.r.t. letters, w.r.t. accents, etc. Finally, a binary comparison between bytes is done, level by level, until the two strings can be distinguished. This algorithm can be refined for a particular language, by using a specialised sort key table, possibly including sort keys for accented letters and digraphs viewed as single letters.

This *modus operandi* would be difficult to put into action within MlBibTeX. First, we do not have

submitted for ratification. See `http://www.r6rs.org` for more details.

[2] Readers can refer to [20] for an introductory book about Scheme.

[3] **D**efault Unicode **C**ollation **E**lement **T**able.

- The Czech alphabet is: a < b < c < č < d < … < h < ch < i < … < r < ř < s < š < t < … z < ž.
- In Danish, accented letters are grouped at the end of the alphabet: a < … < z < æ < ø < å.
- The Estonian language does not use the same order for unaccented letters than usual Latin order; in addition, accented letters are either inserted into the alphabet or alphabeticised like the corresponding unaccented letter: a < … < s ∼ š < z ∼ ž < t < … < w < õ < ä < ö < ü < x < y.
- Here are the accented letters in the French language: à ∼ â, ç, è ∼ é ∼ ê ∼ ë, î ∼ ï, ô, ù ∼ û ∼ ü, ÿ.

  When two words differ by an acccent, the unaccented letter takes precedence, but w.r.t. a right-to-left order:[a] cote < c**ô**te < cot**é** < c**ô**t**é**.

  The French language also use two ligatures: '`æ`' (resp. '`œ`'), alphabeticised like 'ae' (resp. 'oe').
- There are three accented letters in German — '`ä`', '`ö`', '`ü`' — and three lexicographic orders:
  - DIN[b]-1: a ∼ ä, o ∼ ö, u ∼ ü;
  - DIN-2: ae ∼ ä, oe ∼ ö, ue ∼ ue;
  - Austrian: a < ä < … < o < ö < … < u < ü < v < … < z.
- The Hungarian alphabet is:

  a ∼ á < b < c < cs < d < dz < dzs < e ∼ é < f < g < gy < h < i ∼ í < j < k < l < ly < m < n < ny < o ∼ ó < ö ∼ ő < p < … < s < sz < t < ty < u ∼ ú < ü ∼ ű < v < … < z < zs

  Some double digraphs may be restored before sorting:

  `ccs` → `cs`+`cs`, `ddz` → `dz`+`dz`, `ggy` → `gy`+`gy`, `lly` → `ly`+`ly`, `nny` → `ny`+`ny`, `ssz` → `sz`+`sz`, `tty` → `ty`+`ty`

  The same for the double trigraph: `ddzs` → `dzs`+`dzs`.
- The Polish alphabet is:

  a < ą < b < c < ć < d < e < ę < … < l < ł < m < n < ń < o < ó < p < … < s < ś < t < … < z < ż
- The Romanian alphabet is: a < ă < â < b < … < i < î < j < … s < ş < t < ţ < u < … < z.
- The Slovak alphabet is:

  a < á < ä < b < c < č < d < ď < dz < dž < e < é < f < g < h < ch < i < í < j < k < l < Ĺ < l′ < m < n < ň < o < ó < ô < p < q < r < ŕ < s < š < t < ť < u < ú < … < y < ý < z < ž
- The Spanish alphabet was a < b < c < ch < d < … < l < ll < m < n < ñ < o < … < z until 1994. Now the digraphs '`ch`' and '`ll`' are no longer viewed as single letters in modern dictionaries, and the words using '`ñ`' are interleaved with words using '`n`'.
- In Swedish, accented letters are grouped at the end of the alphabet: a < … < z < å < ä < ö.

[a] Using a left-to-right order for this step is common mistake even for French people. But the accurate order is right-to-left, as specified in [7].

[b] ***D**eutsche **I**nstitut für **N**ormung* (German Institute of normalisation).

Figure 1: Some order relations used in European languages.

complete support for Unicode:[4] for example, we cannot directly deal with caracters such as the 'combining circumflex accent', not included in the Latin-1 encoding. But we keep the idea about decomposition, replacing the combining characters by ASCII[5] characters. For example, the 'combining circumflex accent' will be replaced by the '`^`' character. To sum up, our order relations are based on a 3-step algorithm:

- replace composite characters ('foreign' letters or composite characters not viewed as single letters) when extracting successive letter groups and compare the two results,
- refine the sort about accent information when accented letters are interleaved with others,
- test the case.

2 Generating order relations

Let us recall that MlBibTeX can apply BibTeX's bibliography styles using a compatibility mode [6], but in order to take advantage of MlBibTeX's multilingual features as far as possible, it is better to use the nbst[6] language [4], close to XSLT[7] [24], the language of transformations used for XML[8] documents. Let us recall that parsing a bibliography data base (.bib) results in the representation of an XML tree in

[4] See the annex.

[5] **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange.

[6] **N**ew **B**ibliography **ST**yles.

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

[8] e**X**tensible **M**arkup **L**anguage.

Scheme [11], this `nbst` language includes an element for sorting selected subtrees of an XML document [4, App. A], this element being analogous to XSLT's [24, § 10]. For example, the following two elements can be used to sort bibliographical items by the first author's last name, and then the items left unsorted by this first step are sorted by the first author's first name:[9]

```
<nbst:sort select="author/name[1]/last"
           language="german"/>
<nbst:sort select="author/name[1]/first"
           language="german"/>
```

Due to the `language` attribute's value, this sort operation will use the lexicographic order for the German language. Such an order relation is to be specified in Scheme, as a 2-argument function taking two strings s_0 and s_1 and returning a 'true' value (`#t`) if s_0 is strictly less than s_1, a 'false' value (`#f`) otherwise.

The best way to define such a function is to derive it from a generator of order relations, as shown in Figure 2. This `<mk-order-relation` generator has four arguments.

- A list whose elements are *separator* characters, viewed less than any letter. Usually, this list contains only the space character, in which case, the `<space-only` variable can be used. This is not universal: for example, space characters are ignored when words are sorted in Hungarian (cf. the definition of the `<hungarian?` variable in Figure 2).
- An alphabet, given w.r.t. the increasing order, as a list of strings. If the 'classical' alphabet is used — unaccented letters of the Latin alphabet, sorted according to the usual order — just put the 'false' value (cf. the definition of the `<english?` variable).
- An association list for additional sequences of characters, each sequence being followed by a replacement and a weight.
- A function related to the sense of the second step: when the first is finished and the second is about to start, weights appear in reverse order, so put `reverse!`[10] (resp. `identity` — the identity function) to put the second step into action according to a left-to-right (resp. right-to-left) order. Cf. the use of these two values for `<french?` and `<english?`.

It should be noticed that only lowercase letters have to be specified, the equivalent relations among uppercase letters will be deduced.

Let us come back to associations for additional sequence characters, there are default associations, comparable to the information given by the decomposition property in Unicode. For example:

$$\texttt{é} \mapsto \texttt{e} + |\texttt{'}|$$

where "$|\texttt{'}|$" denotes the default weight of the "`'`" character. MlBibTeX knows such decomposition information for each accented letter of `Latin 1`. These default associations can be overrided by alphabet-specific associations given to the function building orders. Weights are managed as follows.

- By default, the weight of each component of an alphabet — appearing within the second argument of `<mk-order-relation` — is 1.
- If we consider only one substitution, that is, a word $\mathcal{W}_0$ where a sequence $\mathcal{S}_0$ is to be replaced by a sequence $\mathcal{S}_1$ with a weight w_1, this substitution resulting in a word $\mathcal{W}_1$. The $\mathcal{W}_0$ word will be alphabeticised at first if $w_1 < 1$, put after otherwise.

Here are some examples.

- In French, the only accent put on the '`o`' letter is circumflex. When '`ô`' is replaced by '`o`' for the first step, we must ensure that '`ô`' will be ranked after '`o`' if two words differ only by these two letters at the same position. We must also ensure that the other accented letters based on '`o`' — in 'foreign' words will be put after. So the weight of the replacement of '`ô`' by '`o`' is 2, as it can be seen in Figure 2 (cf. the definition of `<french?`). The defaults weights for accents are higher, so this accented letter is ranked before the other accented letters based on the '`o`' letter and possibly used in languages other than French.
- Similarly, the two accents allowed on the '`a`' letter are grave and circumflex, the correct order being `a` < `à` < `â`. So the replacement of '`à`' (resp. '`â`') by '`a`' for the first step is 2-weight (resp. 3-weight).

Given a language, if a character belongs neither to separators, nor to the alphabet, it is ignored, unless it is an accented letter included in default associations.

Given an alphabet's specification — the second argument of the `<mk-order-relation` function —

[9] This illustrative example would be too restrictive for an 'actual' bibliographystyle: there may be several authors, and some authors may be denoted by an organisation name, in which case the element's name is not `name`.

[10] Some Schemers could observe that this function does not belong to pure functional style, because it is potentially destructive [17]. But it is more efficient than the `reverse` function and the weight list is not shared with other lists.

```
(define <english (<mk-order-relation <space-only #f '() reverse!))
(define <austrian?
  (<mk-order-relation
   <space-only
   '("a" "ä" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "ö" "p" "q" "r" "s" "t" "u"
     "ü" "v" "w" "x" "y" "z")
   '() reverse!))
(define <czech?
  (<mk-order-relation
   <space-only
   '("a" "b" "c" "\\v{c}" "d" "e" "f" "g" "h" "ch" "i" "j" "k" "l" "m" "n" "o" "p" "q" "r" "\\v{r}"
     "s" "\\v{s}" "t" "u" "v" "w" "x" "y" "z" "\\v{z}")
   '() reverse!))
(define <danish?
  (<mk-order-relation
   <space-only
   '("a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q" "r" "s" "t" "u" "v" "w"
     "x" "y" "z" "æ" "ø" "å")
   '(("aa" ("å" . 2))) ; In Danish, 'aa' is equivalent to 'å'.
   reverse!))
(define <estonian?
  (<mk-order-relation
   <space-only
   '("a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q" "r" "s" "z" "t" "u" "v"
     "w" "õ" "ä" "ö" "ü" "x" "y")
   '(("\\v{s}" ("s" . 2)) ("\\v{z}" ("z" . 2))) reverse!))
(define <french?
  (<mk-order-relation <space-only #f
                      '(("à" ("a" . 2)) ("â" ("a" . 3)) ("ê" ("e" . 2)) ("é" ("e" . 3))
                        ("ê" ("e" . 4)) ("ë" ("e" . 5)) ("î" ("i" . 2)) ("ï" ("i" . 3))
                        ("ö" ("o" . 2)) ("ù" ("u" . 2)) ("ü" ("u" . 3)) ("ÿ" ("y" . 2)))
                      identity))
(define <german-din-1?
  (<mk-order-relation <space-only #f '(("ä" ("a" . 2)) ("ö" ("o" . 2)) ("ü" ("u" . 2))) reverse!))
(define <german-din-2?
  (<mk-order-relation
   <space-only #f '(("ä" ("a" . 2) ("e" . 2)) ("ö" ("o" . 2) ("e" . 2)) ("ü" ("u" . 2) ("e" . 2)))
   reverse!))
(define <hungarian?
  (<mk-order-relation
   '()   ; In Hungarian, a space character is irrelevant when words are sorted.
   '("a" "b" "c" "cs" "d" "dz" "dzs" "e" "f" "g" "gy" "h" "i" "j" "k" "l" "ly" "m" "n" "ny" "o" "ö"
     "p" "q" "r" "s" "sz" "t" "ty" "u" "ü" "v" "w" "x" "y" "z" "zs")
   `(("á" ("a" . 2)) ("é" ("e" . 2)) ("ccs" ("cs" . 2) ("cs" . 2))
     ("ddz" ("dz" . 2) ("dz" . 2)) ("ddzs" ("dzs" . 2) ("dzs" . 2)) ("ggy" ("gy" . 2) ("gy" . 2))
     ("í" ("i" . 2)) ("lly" ("ly" . 2) ("ly" . 2)) ("nny" ("ny" . 2) ("ny" . 2)) ("ó" ("o" . 2))
     ("\\H{o}" ("ö" . 2)) ("ssz" ("sz" . 2) ("sz" . 2)) ("tty" ("ty" . 2) ("ty" . 2))
     ("ú" ("u" . 2)) ("\\H{u}" ("ü" . 2))))
   reverse!))
(define <polish?
  (<mk-order-relation
   <space-only
   '("a" "{\\aob}" "b" "c" "\\'{c}" "d" "e" "{\\eob}" "f" "g" "h" "i" "j" "k" "l" "{\\l}" "m" "n"
     "\\'{n}" "o" "ó" "p" "q" "r" "s" "\\'{s}" "t" "u" "v" "w" "x" "y" "z" "\\.{z}")
   '() reverse!))
```

Figure 2: Building order relations for some European languages.

MlBibTeX notices the possible presence of multi-character sequences (e.g., digraphs or trigraphs). If need be, it builds a lexical analyser able to return the longest sequence of characters belonging to this alphabet,[11] an example of use being given in Figure 3. Let us mention that these analysers extract as few sequences of characters as possible. For example, if we have to compare a word beginning with 'a' and a word beginning with 'b' in English, only the first letters — "a" and "b" — are extracted because that is sufficient to conclude.

From a point of view related to implementation, the encoding of the sequences of an alphabet w.r.t. an increasing order is implemented by means of *hash tables*,[12] what ensures efficiency. Let us not forget that these order relations are used to sort bibliographical items, and sorting requires many calls to the function modelling an order relation.

3 MlBibTeX vs xindy

xindy [9] and MlBibTeX do not aim to perfom the same task, since xindy is an index processor. However, both have common points: they reimplement 'old' programs belonging to TeX's galaxy — makeindex [13, § 11.2] and BibTeX — with particular focus on multilingual features, they are both written using a Lisp[13] dialect: Common Lisp [21] for xindy, Scheme for MlBibTeX. Of course, the successive steps used for putting an order relation into action — needed to arrange the successive entries of an index — also exist in xindy. But the specification of an order relation is different because it is done step by step. There are forms:

```
define-alphabet     define-letter-group
merge-rule          sort-rule
```

to specify an alphabet, a letter group, and the replacement of a pattern. If a sort procedure is quite close to the standard way used in English, it is probably easier to use xindy's forms, because only small changes have to be expressed. In MlBibTeX, we chose to develop less functions, but more powerful. This allows a global view of a new order relation and makes easier some coherence tests among the information about this relation.

[11] Such lexical analysers are implemented by means of *tries*. In MlBibTeX, this structure is also used to manage the information related to language identifiers, as explained in [5].

[12] A hash table has a set of entries, and can efficiently map an object to another object. This structure is described in [1] from a general point of view, our implementation of hash tables in MlBibTeX is inspired by [8].

[13] **LIS**t **P**rocessor.

4 Conclusion

The availability of these language-dependent order relations within a unique program has been planned through the use of the `language` attribute, as specified in the W3C[14] recommendation about XSLT [24, § 10]. However, these relations have been implemented only partially in most of XSLT processors. Of course, our implementation also provides this service partially, because we are limited to European languages. But we think that the orders we define are correct w.r.t. these languages and they are actually running. Our implementation is clearly influenced by the Unicode collation algorithm, it is a first step towards general algorithms for lexicographic orders, it is a first version subject to changes when we explore other languages or get criticisms from end-users. In many domains, improvement has existed because first versions existed. We think that will be also the case for our functions.

5 Acknowledgements

Many thanks to Jerzy B. Ludwichowski, who has written the Polish translation of the abstract. I also thank Gyöngyi Bujdosó, Hans Hagen, Karel Horák, Dag Langmyhr, who helped me fix some errors.

A How to use MlBibTeX's functions

A.1 Getting started

To use the functions dealing with multilingual ordering, change your current directory into the `src` subdirectory of MlBibTeX's main directory, launch a Scheme interpreter, and proceed as follows:

```
(load "common.scm") ; Loading general
                    ; definitions.
(load "orders.scm") ; Loading all the
    ; definitions related to orders. This causes
    ; the other files needed to be loaded, too.
```

Then you can use the functions described in Figure 2. Use a Scheme interpreter R5RS-compliant [10] and able to deal with the Latin 1 encoding: bigloo [16], MIT Scheme[3], PLT Scheme [2] do that.[15]

Now we describe the conventions used within strings resulting from parsing a .bib file. These conventions are supposed to be followed by the arguments of the functions modelling order relations, so you have to know them. You can directly type accented letters belonging to the Latin 1 encoding:

```
"Frank Böhmert"
```

[14] **W**orld **W**ide **W**eb **C**onsortium.

[15] In fact, these three Scheme interpreters include partial support of Unicode, as mentioned in the introduction.

```
(define mk-hungarian-word-sectioner   ; Building a generator of sectioning functions for Hungarian words.
  (<mk-otoken-generator '()                          ; The first three arguments of the <mk-order-relation
                        '("a" "b" "c" "cs" ...)      ; function in the definition of the <hungarian? variable:
                        '(("á" ("a" . 2)) ...)))     ; cf. Figure 2.
(define g                                            ; Definition of a zero-argument function that will
  (mk-hungarian-word-sectioner "sz\\H{o}l\\H{o}"))   ; section the word 'szőlő' ('grape').
(g) ⟹ ("sz" . 1)    ; The successive equivalent letters, digraphs, etc. of this word are returned in turn, with
(g) ⟹ ("ö" . 2)     ; the corresponding weight.
(g) ⟹ ("l" . 1)
(g) ⟹ ("ö" . 2)
(g) ⟹ #f    ; The word is finished, so all the calls of the g function will return the 'false' value, from now on.
```

Figure 3: How to section Hungarian words.

In Scheme, the '`"`' character being the delimiter of constant strings, it must be escaped by a '\' character if it belongs to a string:

```
"\"Perry Rhodan\" Series"
```

If you are interested in strings using other encodings (in particular, the Latin 2 encoding, used in Eastern Europe), you cannot specify them directly; you must use the LaTeX command producing accents and other diacritical signs not included in Latin 1. For example, 'Henryk Mikołaj Górecki' should be typed as follows:

```
"Henryk Miko{\\l}aj Górecki"
```

because '`ó`' belongs to Latin 1, not '`ł`'. Remember that the '\' escape character must be itself escaped within a string. If such an accent command has no argument — e.g., the '`\l`' command — write it between braces, as suggested by the previous example. If it has an argument, write this argument between braces, as in `"Rezs\\H{o} Kókai"` for 'Rezső Kókai'.

Now you can type some expressions and evaluate them:

```
(<english? "coté" "côte") ⟹ #t   ; True.
(<french? "coté" "côte")  ⟹ #f   ; False.
...
```

Of course, you can define new order relations according to the *modus operandi* we explain in § 2 and try to model some 'exotic' order relations.[16]

A.2 Testing decomposition

To see how words are sectioned into successive letters, digraphs, etc. according to a particular alphabet, then use the `<mk-otoken-generator` function to build a generator of functions sectioning words for a particular language. This `<mk-otoken-generator` function is automatically called when we apply the `<mk-order-relation` function, and its three arguments are the second, third and fourth arguments of the `<mk-order-relation` function. As an example, Figure 3 shows how to build such a generator for Hungarian words and use it.

A.3 Going further

If you want to use MlBibTeX for producing bibliographies — in which case you have to load more files by means of evaluating the expression:

```
(load "mlbibtex.scm")
```

— and would like to change the association of a language with an order relation, use the Scheme function `c-language->order-relation` as follows:

```
(c-language->order-relation
 "german"
 <german-din-2?) ⟹ #t
```

This causes the order relation `<-german-din-2?` to be the order relation used for the German language. If another order relation was previously associated with this language,[17] it is replaced by this new value, that is, the `<-german-din-2?` function. If no order relation was known for this language,[18] the association is created. The `c-language->order-relation` function returns `#t` if the association succeeds, `#f` otherwise (for example, if it is called with a string whose value is an unknown language).

References

[1] Alfred V. Aho, Ravi Sethi and Jeffrey D. Ullman: *Compilers, Principles, Techniques and Tools*. Addison-Wesley Publishing Company. 1986.

[16] It can be noticed that all the names of the Scheme functions described above begin with '`<`'. A convention within the source files of MlBibTeX is that all the definitions stored in the same file are the same prefix. That allows a 'kind of modularity', even if Scheme's standard does not provide a way to emphasise modular decomposition. Of course, we recommend you to choose a not-yet-used prefix for your own definitions.

[17] In fact, when MlBibTeX is initialised, the order relation for the German language is the `<german-din-1?` function.

[18] ... in which case the default order relation is the `<english?` function.

[2] Matthew Flatt: *PLT MzScheme: Language Manual. Version 360*. August 2004. http://download.plt-scheme.org/doc/360/pdf/mzscheme.pdf.

[3] Chris Hanson, the MIT Scheme team *et al.*: *MIT Scheme Reference Manual*, 1st edition. March 2002. Massachusetts Institute of Technology.

[4] Jean-Michel Hufflen: "MlBibTeX's Version 1.3". *TUGboat*, Vol. 24, no. 2, pp. 249–262. July 2003.

[5] Jean-Michel Hufflen: *Managing Languages within MlBibTeX*. To appear. June 2005.

[6] Jean-Michel Hufflen: "BibTeX, MlBibTeX and Bibliography Styles". *Biuletyn GUST*, Vol. 23, pp. 76–80. In *BachoTeX 2006 conference*. April 2006.

[7] ISO-IEC CD 14651: *International String Ordering — Method for Comparing Character Strings and Description of a Default Tailorable Ordering*. May 1996.

[8] Panu Kalliokoski: *Basic Hash Tables*. September 2005. http://srfi.schemers.org/srfi-69/.

[9] Roger Kehr: *x̊indy Manual*. February 1998. http://www.xindy.org/doc/manual.html.

[10] 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: "Revised^5 Report on the Algorithmic Language Scheme". *HOSC*, Vol. 11, no. 1, pp. 7–105. August 1998.

[11] Oleg E. Kiselyov: *XML and Scheme*. September 2005. http://okmij.org/ftp/Scheme/xml.html.

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

[13] Frank Mittelbach, Michel Goossens, Johannes Braams, David Carlisle, Chris A. Rowley, Christine Detig and Joachim Schrod: *The LaTeX Companion*. 2nd edition. Addison-Wesley Publishing Company, Reading, Massachusetts. August 2004.

[14] Oren Patashnik: *Designing BibTeX Styles*. February 1988. Part of the BibTeX distribution.

[15] Oren Patashnik: *BibTeXing*. February 1988. Part of the BibTeX distribution.

[16] Manuel Serrano: *Bigloo. A Practical Scheme Compiler. User Manual for Version 2.9a*. December 2006.

[17] Olin Shivers: *List Library*. October 1999. http://srfi.schemers.org/srfi-1/.

[18] Michael Sperber, William Clinger, R. Kent Dybvig, Matthew Flatt, Anton van Straaten, Richard Kelsey and Jonathan Rees: *Revised$^{5.97}$ Report on the Algorithmic Language Scheme — Standard Libraries*. June 2007. hhttp://www.r6rs.org.

[19] Michael Sperber, William Clinger, R. Kent Dybvig, Matthew Flatt, Anton van Straaten, Richard Kelsey, Jonathan Rees, Robert Bruce Findler and Jacob Matthews: *Revised$^{5.97}$ Report on the Algorithmic Language Scheme*. June 2007. hhttp://www.r6rs.org.

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

[21] 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.

[22] The Unicode Consortium: *The Unicode Standard Version 4.0*. Addison-Wesley. August 2003.

[23] The Unicode Consortium, http://unicode.org/reports/tr10/: *Unicode Collation Algorithm*. Unicode Technical Standard #10. July 2006.

[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.