Names in BibTeX and MlBibTeX

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

Abstract

Within the bibliographical entries managed by BibTeX, the bibliography processor usually associated with LaTeX, person and organisation names are specified with a rough syntax, whose details are not very well known. Likewise, the features related to formatting names within bibliography styles are often viewed as obscure. We explain these points in detail, giving some cases difficult or impossible to handle with BibTeX. Then we show how these problems can be solved within MlBibTeX, our reimplementation of BibTeX focusing on multilingual features and using an extension of XSLT as the language for bibliography styles.
Keywords BibTeX, MlBibTeX, Bibliographies, bibliography styles, specifying and formatting person names.

1 Introduction

Specifying meta-information related to persons is a difficult problem within databases from a general point of view. It is well known that the parts constituting the name of a person are insufficient to characterise only one person. However, we do not go thoroughly into this point and only focus on names we can find within bibliographical databases. Managing this information is crucial: it may be used for searching bibliographical databases, and many bibliography styles sort references w.r.t. alphabetical order of authors or editors.

When users typeset documents with the LaTeX word processor [15], searching bibliographical databases for citations and assembling references into a 'Bibliography' section, put at the end of a printed document, is usually done with the BibTeX bibliography processor [20]. Formats for names are defined as part of bibliography styles: first names may be abbreviated or put *in extenso*, they can be written before or after the last name, ... BibTeX uses a rough syntax for specifying authors' and editors' names within bibliographical entries. This syntax is suitable for simple cases, and powerful since it includes many interesting and useful features, sometimes not well known due to their complexity. In addition, some of these features are documented only partially, as far as we know, although many details are given in the second edition of the *LaTeX Companion* [16, § 13.2.2]. Likewise, the primitive `format.name$` function, which formats names within BibTeX's bibliography styles [19] is powerful but uses quite complicated patterns, documented only partially. In fact, only a small fraction of its expressive power is used in practice.

So we begin by describing BibTeX's syntax for names, with its advantages and limitations. We also show how names are formatted, as precisely as possible. Then we explain how these points have been improved within MlBibTeX (for '**M**ulti**L**ingual BibTeX) [7], our reimplementation focusing on multilingual aspects. Last, we discuss aspects of internationalization.

Reading this article requires good knowledge of BibTeX as an end-user, and a bit of experience with bibliography styles, that is, a good knowledge of the notions described in [16, § 13.6]. We will recapitulate more technical points, however.

2 How names are processed by BibTeX

2.1 Names' components

Inside `AUTHOR` and `EDITOR` fields, BibTeX allows the specification of successive names, separated by the '`and`' keyword, as shown in Figure 1. A large list of names that is not typed *in extenso* is ended with '`and others`':

```
{Karl-Heinz Scheer and Clark Darlton and
 others}
```

```
@BOOK{feist-wurts1991,
        AUTHOR = {Raymond E. Feist and
                  Janny Wurts},
        TITLE = {Servant of the Empire},
        PUBLISHER = {Grafton Books},
        ADDRESS = {London},
        YEAR = 1991}
```

Figure 1: Example of a BibTeX bibliographical entry.

A name consists of four components: *First* (for a first name), *von* (for a particle), *Last* (for a last name), and *Junior* (for a suffix), and recognizes them in the following possible syntaxes [20, § 4]:

(i) *First von Last*

(ii) *von Last*, *First*

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

As suggested by the word capitalisation used within this terminology — originating from BibTeX — the words belonging to the *von* field are supposed to begin with a lowercase character, whereas the words belonging to the *First* and *Last* fields are supposed to begin with an uppercase character, e.g.:

$$\underbrace{\texttt{Catherine Crook}}_{First}\ \underbrace{\texttt{de}}_{von}\ \underbrace{\texttt{Camp}}_{Last}$$

The rule common to these three syntaxes: if there is only one word, it is taken as the *Last* part, even if this word does not begin with an uppercase letter, e.g.:

$$\underbrace{\texttt{\{-\}ky}}_{Last}$$

If we consider the (i) syntax, two other rules are used when a name is split into its components:

- the *von* part takes as many words as possible, provided that its first and last words begin with a lowercase letter, e.g.:[1]

  $$\underbrace{\texttt{Jean}}_{First}\ \underbrace{\texttt{de la Fontaine du}}_{von}\ \underbrace{\texttt{Bois Joli}}_{Last}$$

  let us notice that the *First* part can be empty, whereas the *Last* part cannot:

  $$\underbrace{\texttt{jean de la fontaine du bois}}_{von}\ \underbrace{\texttt{joli}}_{Last}$$

- if all the words begin with an uppercase letter, the last word is the *Last* component, and the *First* part groups the other words, e.g.:

  $$\underbrace{\texttt{Kim Stanley}}_{First}\ \underbrace{\texttt{Robinson}}_{Last}$$

If we consider the (ii) or (iii) syntaxes, the *von* part takes as many words as possible, provided that only its last word begins with a lowercase letter, e.g.:

$$\underbrace{\texttt{De la Fontaine du}}_{von}\ \underbrace{\texttt{Bois Joli}}_{Last},\ \underbrace{\texttt{Jean}}_{First}$$

2.2 Using braces

If the first letter of a word is surrounded by braces,[2] it is supposed to be uppercase, unless it follows a TeX command and is lowercase. Here are some examples:

- $\underbrace{\texttt{Alfred Elton \{van\}}}_{First}\ \underbrace{\texttt{Vogt}}_{Last}$
- $\underbrace{\texttt{Alfred Elton}}_{First}\ \underbrace{\texttt{\{\textbackslash relax van\}}}_{van}\ \underbrace{\texttt{Vogt}}_{Last}$
  (`\relax` is a dummy command [14, Ch. 24]),
- $\underbrace{\texttt{Alfred Elton \{\textbackslash relax Van\}}}_{First}\ \underbrace{\texttt{Vogt}}_{Last}$

Remark 1 *Non-letter characters are ignored when BibTeX determines whether the first letter of a word is lower- or uppercase. That is why it considers '`{-}ky`' (see above) to begin with a lowercase letter, ignoring the hyphen sign between braces.*

As mentioned in [16, § 13.2.2], enclosing some characters between braces serves four purposes:

- treating accented letters — accented by means of TeX commands — as single letters:

  ```
  Christian Vil{\'{a}}
  ```

- treating multiple words as one, especially when a component (*Last*) consists of a single word by default:

  $$\underbrace{\texttt{Michael}}_{First}\ \underbrace{\texttt{\{Marshall Smith\}}}_{Last}$$

  this feature being commonly used for specifying an organisation name, which consists of only a *Last* part:

  ```
  EDITOR =
  {{Science Fiction and Fantasy
    Writers of America, Inc.}}
  ```

- treating '`and`' as an ordinary word[3] (see the example above);
- delimiting substrings which should *not* change case when the `change.case$` function [16, Table 13.8] is applied.[4]

[1] Only the following name is imaginary (although it is derived from a French poet's name). All the others — some being pseudonyms — name real persons, even if some look strange. That is, the problems raised by BibTeX in the examples we give may arise in real situations.

[2] In the following, we do not consider the braces that might delimit the value of a BibTeX field. 'Braces' should be understood as 'additional braces' in such a case.

[3] '`and`' is viewed as a keyword only at the topmost level, not surrounded by additional braces.

[4] ... although this use is rare within `AUTHOR` or `EDITOR` fields. See more details in Figure 3.

```
              "Charles" text.length$  pushes  #7              % '#' begins a number.
            "{Ch}arles" text.length$  ......  #7
      "{\relax Ch}arles" text.length$  ......  #6
     "{ \relax Ch}arles" text.length$  ......  #15             % Space after '{'.
         "Charles" #1 #1 substring$    ......  "C"
       "{Ch}arles" #1 #1 substring$    ......  "{"             % Unbalanced brace!
"{\relax Ch}arles" #1 #2 substring$    ......  "{\"
 "B{\'a}rt{\'o}k" #-2 #3 substring$    ......  "'o}"
           "Charles" #1 text.prefix$   ......  "C"
         "{Ch}arles" #1 text.prefix$   ......  "{C}"
        "}{Ch}arles" #1 text.prefix$   ......  "}{C}"
  "{\relax Ch}arles" #1 text.prefix$   ......  "{\relax Ch}"
 "{ \relax Ch}arles" #8 text.prefix$   ......  "{ \relax }"
 "{ \relax Ch}arles" #3 text.prefix$   ......  "{ \r}"         % Truncate command name!
```

Figure 2: Examples of functions dealing with strings in bst.

2.3 Words vs tokens

In the above subsections, 'word' has been used in the common sense, that is, 'an independent unit of the vocabulary of a language' [18], so that adjacent words are syntactically separated by space characters or punctuation signs. In fact, if we wish to characterise this notion in the sense of successive tokens handled by BibTeX inside a string representing a name, the separators are whitespace characters[5] — space, tabulation, line feed, form feed, and carriage return — unbreakable space characters, specified by '~' as in TeX, *and the hyphen sign* '-'. Such a choice allows BibTeX to process all the components of a first name more easily when it is abbreviated, even if these tokens are separated by hyphen signs, as in French:

```
Jean-Pierre Andrevon
```

However this design choice causes strange behaviour to happen in some particular cases:

$$\underbrace{\texttt{Jean-Claude Smit-}}_{First}\ \underbrace{\texttt{le}}_{von}\ \underbrace{\texttt{-B\{\'e\}n\{\'e\}dicte}}_{Last}$$

whereas this name has only *First* and *Last* parts, as suggested by the single space character. Of course, this name should be specified by:

$$\underbrace{\texttt{Jean-Claude}}_{First}\ \underbrace{\texttt{\{Smit-le-B\{\'e\}n\{\'e\}dicte\}}}_{Last}$$

More precisely, BibTeX retains only the first separator between two tokens, omitting any additional separators from the output:

$$\texttt{Edgar}_{\sqcup\sqcup}\texttt{Rice} \Longrightarrow \texttt{Edgar}_{\sqcup}\texttt{Rice}$$
$$\texttt{Edgar}_{\sqcup}\texttt{\~{}Rice} \Longrightarrow \texttt{Edgar}_{\sqcup}\texttt{Rice}$$
$$\texttt{Edgar\~{}}_{\sqcup}\texttt{Rice} \Longrightarrow \texttt{Edgar\~{}Rice}$$
$$\texttt{Karl-}_{\sqcup}\texttt{Heinz} \Longrightarrow \texttt{Karl-Heinz}$$

[5] This terminology originates from the Scheme programming language [11, § 6.3.4]: such characters are recognised by the `char-whitespace?` function of this language.

It also omits any separator before the first token.

Remark 2 *That is why we put the hyphen sign between braces in '`{-}ky`' (see above). If the braces are removed, the hyphen disappears.*

Now let us assume that there is no *von* part.

- If the *Last* part follows the '~' sign, the '~' is omitted:

$$\underbrace{\texttt{Kenneth}}_{First}\texttt{\~{}}\underbrace{\texttt{Robeson}}_{Last}$$

- If the *Last* part follows the '-' character, the word before belongs to this part, too:

$$\underbrace{\texttt{Louis-Albert}}_{Last}$$

2.4 Applying a bibliography style

In general, we can see that strings are not handled homogeneously in bst, the language used to write BibTeX's bibliography styles [19]. Let us consider the three following bst functions:

$$\mathcal{S}\ \texttt{text.length\$}$$
$$\mathcal{S}\ \mathcal{I}\ \texttt{text.prefix\$}$$
$$\mathcal{S}\ \mathcal{I}_1\ \mathcal{I}_2\ \texttt{substring\$}$$

where $\mathcal{S}$ is a string and $\mathcal{I}, \mathcal{I}_1, \mathcal{I}_2$ are integers. The bst language is based on handling a stack, so the arguments are passed to a function by putting their values *before* the function name.[6] These three functions respectively push (return) the length of $\mathcal{S}$; the first $\mathcal{I}$ characters of $\mathcal{S}$; and $\mathcal{I}_2$ characters of $\mathcal{S}$, starting at position $\mathcal{I}_1$ if $\mathcal{I}_1 > 0$, or ending at this position if $\mathcal{I}_1 < 0$, in which case positions are counted backward from the end of the string.

Braces do not count when the `text.length$` function is called, but they do for the `substring$` function, as shown by the examples of Figure 2. We

[6] MlBibTeX's compatibility mode allows users to see how this stack works [9].

```
"frank"               ⟹ "FRANK"
"{frank}"             ⟹ "{frank}"
"{\relax frank}"      ⟹ "{\relax FRANK}"
"{\relax {frank}}"    ⟹ "{\relax {FRANK}}"
"{\relax {{frank}}}"  ⟹ "{\relax {{FRANK}}}"
"{\a frank \b f}"     ⟹ "{\a FRANK \b F}"
"{ \relax frank}"     ⟹ "{ \RELAX FRANK}"
"{{\relax frank}}"    ⟹ "{{\relax frank}}"
```

Figure 3: ... `"u" case.change$` results.

also see that the `substring$` function may push a string where braces are unbalanced. Braces do not count for the `text.prefix$` function, either, and we can observe strange behaviour:

- if an unbalanced left or right brace is encountered, it is put into the string pushed,
- right braces closing unbalanced left braces are added at the end of the string pushed.

The `substring$` function counts each character, in the sense that any typed character is relevant, including enclosed braces and characters surrounded by braces. Other functions — `text.length$` and `text.prefix$` — consider that if a left brace is immediately followed by a '\' character, the complete group between the left enclosing brace and the corresponding right one is viewed as one single character. Such a group is called **special character** within BibTeX's terminology [16, pp. 768–769]. This distinction between special characters and other groups surrounded by braces explains some results shown in Figure 2. Anyway, let us notice that this distinction is recognised by the `change.case$` function:

$$\mathcal{S}\ \mathcal{S}_0\ \texttt{change.case\$}$$

— where $\mathcal{S}$ is a string — converts $\mathcal{S}$ to lowercase (resp. uppercase) if $\mathcal{S}_0$ is the string `"l"` (resp. `"u"`), and pushes the result. If $\mathcal{S}_0$ is `"t"`, $\mathcal{S}$ is converted to lowercase except for the first character or the first group if $\mathcal{S}$ begins with a left brace. As shown by the examples given in Figure 3, the non-command parts of a special character are processed like ordinary characters — even if some subparts are surrounded by braces — whereas all the other groups surrounded by braces are left unchanged by the `change.case$` function. That recalls what is written in [16, p. 768] about protecting some uppercase letters ... provided that the first character after a left brace is not a '\' character.

Let us now go back to the operations returning subparts of a string. We see that this data structure is handled with difficulty in the bst language, except for simple strings. If we consider values of the `AUTHOR` and `EDITOR` fields, the usual way to deal

```
"Frank Frazetta" ⟹ "Frazetta, +Frank:"
"{-}ky"          ⟹ "{-}ky, :"
```

Figure 4: ... `"{ll}, {+ff}:" format.name$` results.

with them is the `format.name$` function, used as follows:

$$\mathcal{S}_1\ \mathcal{I}\ \mathcal{S}_2\ \texttt{format.name\$}$$

where $\mathcal{S}_1, \mathcal{S}_2$ are strings and $\mathcal{I}$ is a natural number. This function formats the $\mathcal{I}$th name of $\mathcal{S}_1$ according to the pattern given by $\mathcal{S}_2$, and pushes the result. The pattern is written as follows:

- if characters are not enclosed by braces, they are unconditionally inserted into the result;
- at the first level:
  - if braces enclose other characters than letters, they also are unconditionally inserted into the result;
  - if braces enclose letters other than '`f`', '`j`', '`l`', '`v`' (for '*First*', '*Junior*', '*Last*', '*von*'), it is an error;[7]
  - a single letter ('`f`', etc.) inserts an abbreviation of the corresponding part (see below), while a doubled letter ('`ff`') inserts the complete part.

Examples are given in Figure 4.

Let us say that '`f`' or '`ff`', '`j`' or '`jj`', ... are called **subpatterns**.[8] Each part is processed as follows:

- if this part is absent within the person name, the complete specification surrounded by braces is ignored;
- if this part is not empty:
  - all the characters before the subpattern are inserted before the corresponding part;
  - if the subpattern is immediately followed by a group surrounded by braces, this last group — which may be empty — replaces the separator between two adjacent tokens of the corresponding part;
  - the other characters following the subpattern are inserted after the corresponding part.

If '`~`' characters are put at the end of characters following a subpattern or a separator replacement:

- one '`~`' causes a space character ('␣') to be inserted after the name's part,

[7] If an error occurs, such a bst function pushes a dummy value: an empty string in this case.

[8] This terminology is used within the source files of MlBibTeX's compatibility mode.

```
last => Le Clerc De La Herverie
"{ll}"          ⟹ Le~Clerc De La~Herverie
"{ll/}"         ⟹ Le~Clerc De La~Herverie/
"{ll/,}"        ⟹ Le~Clerc De La~Herverie/,
"{ll{/},}"      ⟹ Le/Clerc/De/La/Herverie,
"{ll{},}"       ⟹ LeClercDeLaHerverie,
"{ll~}"         ⟹ Le~Clerc␣De␣La~Herverie␣
"{ll~~}"        ⟹ Le~Clerc De La~Herverie~
"{ll{~}~}"      ⟹ Le~Clerc~De~La~Herverie␣
"{ll{~}~~}"     ⟹ Le~Clerc~De~La~Herverie~
"{ll{/},~}"     ⟹ Le/Clerc/De/La/Herverie,␣
"{ll{/}~,~}"    ⟹ Le/Clerc/De/La/Herverie~,␣
"{ll{/}~~,~~}" ⟹ Le/Clerc/De/La/Herverie~~,~

last => Zeb Chillicothe Mantey
"{ll}"          ⟹ Zeb~Chillicothe~Mantey
```

```
first => Jean-Michel-Georges-Albert
"{f}"           ⟹ J.-M.-G.-A
"{f/}"          ⟹ J.-M.-G.-A/
"{f/,}"         ⟹ J.-M.-G.-A/,
"{f{/},}"       ⟹ J/M/G/A,
"{f{},}"        ⟹ JMGA,
"{f~}"          ⟹ J.-M.-G.-A
"{f~~}"         ⟹ J.-M.-G.-A~
"{f{~}~}"       ⟹ J~M~G~A␣
"{f{~}~~}"      ⟹ J~M~G~A~
"{f{/},~}"      ⟹ J/M/G/A,␣
"{f{/}~,~}"     ⟹ J/M/G/A~,␣
"{f{/}~~,~~}" ⟹ J/M/G/A~~,~

last => Cousin De Grainville
"{ll}"          ⟹ Cousin De~Grainville
```

Figure 5: Examples of using patterns with the `format.name$` function of BibTeX.

- if there are several '~' characters, the first is dropped, while the others are inserted after the name's part.

At other places, tilde characters are inserted like ordinary characters.[9]

Let us assume that there are several tokens for a name's part. By default — without separator redefinition — a '~' character is inserted:

- always between the next-to-last and last tokens,
- between the first and second tokens: if there are three or more tokens, and the first token is one or two letters long, a special character or period belonging to an abbreviated word is counted as one letter.

All these cases are summarised in the examples given in Figure 5.

A token is abbreviated by retaining only its first letter. So, other characters inserted before this first letter may be dropped. When BibTeX abbreviates a name part, it recognises special characters, as shown in Figure 6. But it does not insert braces for other groups surrounded by braces, as the `text.prefix$` function would do: compare the examples given in Figures 2 & 6 for the string `{Ch}arles`. Last, let us mention that by default — without separator redefinition — this first letter is followed by a period character and the separator — a space, '~' or '-' character — put after this token. This mark after the first letter is not appended after the abbreviation of the last token. Some of these rules might seem strange if we think of them only in relation to abbreviating first names, but let us recall that some styles use initials of the *von* and *Last* parts as labels of bibliographical references. For example, the alpha bibliography style uses the `{v{}}{l{}}` pattern to generate these labels.

[9] Let us notice that, in contrast, any occurrence of the space character is processed w.r.t. a 'standard way'.

2.5 Criticism

The functions provided by BibTeX work fine in most practical cases, in the sense that most BibTeX end-users accept the results of the standard bibliography styles. However, the cases not 'naturally' included in this framework are difficult to handle. A simple example is given by abbreviations. Let us consider the two following names:

Edgar Rice Burroughs
Jon L White

The `{f.}` subpattern would cause a period to be appended after 'R' in 'E. R. Burroughs', but incorrectly abbreviates the second example to 'J. L. White'.[10] The `{f}` subpattern correctly puts 'J. L White' in the second case, but then a period is missing after Edgar Rice Burroughs' middle name. In addition, it is difficult to specify a particular abbreviation for a middle name only. For example, Nicholas deBelleville Katzenbach's middle name is correctly abbreviated to 'deB.'.

From a general point of view, some workarounds exist, but may be viewed as *hacks*. Often they consist in inserting LaTeX commands into fields' values. For example:

- a command deferring the right case [16, p. 767]:

```
Maria {\MakeUppercase{d}e La} Cruz
      ──────────────────────
            von
```

[10] Anyway, [2, § 14.4] recommends the systematic use of a final period. But such a sign is supposed to replace some letters omitted.

```
Charles           ⟹ C
{Ch}arles         ⟹ C
{\relax Ch}arles  ⟹ {\relax Ch}
{-}ky             ⟹ k
```

Figure 6: Abbreviating first names in bst.

so BibTeX interprets '`{de...}`' as the beginning of the *von* part, because the '`d`' letter is lowercase, even though LaTeX will typeset 'De' when the command is processed;

- inserting a dummy accent command [13, § 251]:

  ```
  {\relax Ph}ilippe Djian
  ```

  in order for this French first name to be abbreviated to 'Ph.' because French digraphs — 'ch' is a digraph for '[ʃ]' — should not be reduced.

Other 'tricks' are more subtle. For example, standard bibliography styles put a space character between the *von* and *Last* part. That is suitable for most cases, e.g., 'Lyon Sprague de Camp', but not when the particle ends with an elision, e.g., 'Guy d'Antin'. Testing whether or not the *von* part ends with an apostrophe character ("'") is tedious, because the bst language does not provide a natural way to store part of a name in a variable. In addition, let us not forget that there may be several names inside an `AUTHOR` or `EDITOR` field, which complicates the search of the accurate indices. One solution is:

```
Guy d'\unskip Antin
```

— see [14, Ch. 24] about the `\unskip` command — but this is not fully satisfactory: it works only if we are sure that the `text.prefix$` function is not applied to the *von* part for a prefix length greater than 2. Likewise, the `change.case$` function should not be used (*cf. supra*). Protecting this `\unskip` command by braces:

```
Guy d'{\unskip} Antin
```

would solve these problems but annihilate the effect of this command. The insertion of an `\aftergroup` command [14, Ex. 24.7], deferring some tokens until the end of a group is processed:

```
Guy d'{\aftergroup\unskip} Antin
```

is useless because the offending space character *follows* the brace closing the group with `\aftergroup`. And this space, not surrounded by braces, is needed when BibTeX separates the *von* and *Last* parts. The best solution — cf. [14, Ch. 24] — is:

```
Guy d'{\aftergroup\ignorespaces} Antin
```

in the sense that it works in most cases, provided that the abbreviated first name is not to be followed by a delimiter, as in '[Guy d']Antin'.

```
<author>
  <name>
    <personname>
      <first abbrev="L. Sprague">
        Lyon Sprague
      </first>
      <von>de</von>
      <last>Camp</last>
    </personname>
  </name>
  <and/>
  <name>
    <personname>
      <first>David</first>
      <last>Drake</last>
    </personname>
  </name>
</author>
```

Figure 7: The internal representation of names in MlBibTeX.

As mentioned above, the `format.name$` function allows good control of separators between the tokens belonging to a same part. However, some limitations exist. For example, let us consider:

```
Ursula Kroeber {Le~Guin}
```

By default — that is, using the `{ff}` pattern — an unbreakable space character will be inserted between the 'actual' first name ('Ursula') and the middle name ('Kroeber'). If we would like to allow a linebreak after the first name because the bibliography will be typeset on a small text width, we can do that by the '`{ff{ }}`' pattern. But such a redefinition is impossible for the two tokens of the last name ('Le Guin'), surrounded by braces. In such a case, the name may be specified by:

```
Le Guin, Ursula Kroeber
```

but braces are needed for an organisation name:

```
AUTHOR = {{Hidalgo~Trading~Company}}
```

and redefining the separator between words becomes impossible by means of the `format.name$` function since BibTeX considers there to be only one token. We can program this operation with the functions `substring$` and `*` — concatenation of two strings [16, Table 13.8] — but it is very tedious.

From our point of view, the use of additional braces belongs to BibTeX's 'philosophy', but is complicated in the sense that some functions process braced groups opening a TeX command differently from other groups surrounded by braces. Often this design choice is good — for example, it allows the `change.case$` function to change the case of accented letters typed by means of TeX commands —

```
<nbst:template match="von">
  <nbst:variable name="the-part" select="."/>
  <nbst:value-of select="$the-part"/>
  <nbst:if test='substring($the-part,string-length($the-part),1) != "&quot;"'>
    <!--  '&quot;' is a predefined entity for an apostrophe character [21, p. 48].  -->
    <nbst:text> </nbst:text>
  </nbst:if>
</nbst:template>
```

Figure 8: Putting a particle's name down.

but other operations are difficult to perform. The insertion of (LA)TeX commands seems to us to be just a workaround. It works since BibTeX is used in conjunction with LaTeX. That was true when BibTeX came out,[11] but is not always the case nowadays. BibTeX may be used to generate bibliographies displayed on Web pages written in HTML,[12] by means of a converter like BibTeX2HTML [3]. Another example, closer to LaTeX, is Hans Hagen's format ConTeXt [4]. Using LaTeX commands within values associated with BibTeX fields can cause trouble when these programs run (some examples concerning ConTeXt and solutions are given in [10]).

3 How names are processed by MlBibTeX

3.1 Implementation issues

We suggest that the components of a name should be directly accessible by means of different placeholders within the functions of a bibliography style. As explained in [7], parsing bibliographical entries from a .bib file results in an XML[13] tree in MlBibTeX.[14] The organisation of our elements is a revision and extension of the DTD[15] given in [6], influenced by BibTeX. Concerning names, fields related to authors and editors are split into subtrees, as shown in the example of Figure 7. In fact, this figure is a 'pretty-printing' of such a tree: in reality, the contents of text nodes — e.g., `first`, `von`, `last` — are 'space-normalised', that is, stripped of leading and trailing whitespace characters, multiple consecutive occurrences of whitespace characters being replaced by a single space character. Likewise, most of the blank nodes[16] pictured in Figure 7 do not exist in the 'actual' representation. Besides, this representation uses Latin 1 encoding,[17] and some special characters of LaTeX are processed; for example, the '~' character is replaced by an unbreakable space character (numbered 160 in Latin 1). In addition, some LaTeX commands are expanded; for example, accent commands applied to suitable letters are replaced by the corresponding accented letters included in Latin 1 [9].

Bibliography styles are written using the nbst[18] language, close to XSLT,[19] a language of transformations used for XML texts. This nbst language is described in [7].

An example written using nbst is given in Figure 8. If the *von* part of a name exists, this template is invoked, the contents of this *von* part is written down. This part is followed by a space character, unless its last character is an apostrophe. This template allows the two examples given above — 'Lyon Sprague de Camp' and 'Guy d'Antin' — to be displayed nicely. Of course, this is an *ad hoc* solution, but it shows that we get the full expressive power of a programming language. In addition, we can call functions written in Scheme — the implementation language of MlBibTeX — for difficult cases [7, 8].

3.2 Syntactic issues

MlBibTeX can process any .bib file designed for 'old' BibTeX, except that square brackets are syntactic delimiters used for multilingual features [7]. So, most of the 'tricks' used within 'old' .bib files should work. In addition, MlBibTeX allows more explicit syntax for the components of a person name and the abbreviation of a first name, when needed:

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

[11] Initially, BibTeX was designed to work with Scribe [22].

[12] **H**yper**T**ext **M**arkup **L**anguage. See [17] for an introduction.

[13] E**X**tensible **M**arkup **L**anguage. See [21] for an introduction.

[14] More precisely, an XML tree represented in Scheme using the conventions of SXML (Scheme implementation of XML) [12].

[15] **D**ocument **T**ype **D**efinition. A DTD defines a document markup model [21, Ch. 5].

[16] Anonymous text nodes whose contents are only whitespace characters; two examples can be found around the `and` tag in Figure 7. In XML, all characters are preserved [21, p. 38].

[17] Future versions of MlBibTeX should be able to deal with Unicode characters [24].

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

[19] E**X**tensible **S**tylesheet **L**anguage **T**ransformations. See [21, Ch. 6] for ashort introduction.

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

```
first => Kim Stanley, last => Robinson
```

where the *von* field is empty, and the abbreviation of the first name is standard, that is, '`K.~S.`' Let us notice that in MlBibTeX, the period character ending an abbreviation belongs to it by default. In addition, this new syntax can be used with 'old' bibliography styles, written in bst, as we show in Appendix A.

You can mix the 'old' and 'new' syntaxes, in which case a name is parsed like (i) if no comma occurs before a keyword, like (ii) (resp. (iii)) if the number of commas not followed with a keyword is one (resp. two) and the keywords give additional information. Let us give some examples:

```
Robinson, first => Kim Stanley
```

is allowed, because '`Robinson`' is parsed as the *Last* part, so '`Kim Stanley`' is allowed to be the *First* part. But:

```
Kim Stanley, last => Robinson
```

WRONG!

is an incorrect specification, because '`Stanley`' is supposed to be the *Last* part, so this part cannot be redefined to '`Robinson`'.

In practice, mixing the old and new syntaxes is useful when we have just to give a specific abbreviation for a first name:

```
Lyon Sprague de Camp, abbr => L. Sprague
```

Roughly speaking, this syntax is close to Ada's for passing values inside a subprogram call [23, § 6.4].

Other keywords can be used for both an organisation name and a key for sorting:

```
EDITOR = {org => \TUG 2006,
          sortingkey => TUG2006}
```

As in BibTeX, co-authors are connected with the '`and`' keyword. After the different co-authors, MlBibTeX allows the addition of *collaborators*, introduced by the '`with`' keyword:[20]

```
Clive Cussler with Paul Kemprecos
```

The format for several co-authors and collaborators:

```
... and ... and ... with ... with ...
```

In the present article's bibliography, reference [16] gives an example of how such an entry using several co-authors and collaborators is formatted. As in BibTeX, the '`others`' keyword can be used when additional names are left unspecified: '`and others`' (resp. '`with others`') for additional unspecified co-authors (resp. collaborators).

[20] ... at the topmost level only. See Footnote 3, p. 244.

Multilingual switches with a default[21] are allowed for names, which is useful for names originating in languages using other alphabets:

```
AUTHOR =
  {[Сергей Сергеевич Прокофьев] * russian
   [Sergey Sergeyevich Prokofiev]}
```

4 Internationalization of names

MlBibTeX allows names to be displayed according the cultural background of a language. For example, accurate templates of the nbst language [8] allow names of Hungarian people:

```
AUTHOR = {[Béla Bartók] : magyar}
```

to be displayed like 'Bartók Béla' — that is, the last name at first, followed by the first name — whereas other names (English, French, ...) are displayed as usual, that is, the first name followed by the last name. However, the specification of any person name has to be dispatched into the four components inherited from BibTeX. This is not a problem related to parsing, because the new keywords allow us to specify each component separately. In practice, some cases are solved by extending the *First* part in order to include:

- a middle name for American people:

  Ursula K. Le Guin

- a *patronym* (father's first name) for a Russian name, here 'Sergey, Sergey's son':

  Сергей Сергеевич Прокофьев

Other cases may more difficult to handle. Here are some exotic examples given in [27]:

- an Assyrian name consisted of a personal name, the father's name, and the grandfather's name,
- in South India, a personal name may be preceded by the father's name, usually written as an initial, and possibly replaced or supplemented by the birthplace or mother's house name, e.g. 'Trivandrum R. S. Mani'.[22]

We plan to go thoroughly into this notion when the document model of the bibliographies handled by MlBibTeX is revised by using *schemas*. For small examples of using the XML Schema standard for describing an organisation for names, [25, pp. 91–107] can be consulted. DocBook, an XML system for writing structured documents [26], proposes a more open approach: some optional elements are

[21] This means that something is *always* produced, even if no information is available in the reference's language. In other words, this kind of switch never yields nothing. See [7] for more details.

[22] In such a case, periods are sometimes omitted.

```
<author>
  <honorific>Sir</honorific>
  <firstname>Arthur</firstname>
  <surname>Conan Doyle</surname>
</author>

<author>
  <firstname>Edgar</firstname>
  <surname>Burroughs</surname>
  <othername role="mi">Rice</othername>
</author>

<author lang="fr-BE">
  <firstname>J.-H.</firstname>
  <surname>Rosny</surname>
  <lineage>
    <!--  Junior, Senior, etc. [26, p. 308].  -->
    aîné
  </lineage>
</author>
```

Figure 9: Names specified using DocBook.

defined — e.g., `honorific` (see Figure 9) — including a hold-all element, `othername`, for information that does not fit in other categories. This element — which may be repeated — has a `role` attribute that classifies the different kinds of these 'other' names. However, official documents do not make precise the possible values for this attribute: so names' taxonomy is not really fixed in DocBook, except for classic cases, for example, '`mi`' for 'middle names', as given in [26, p. 149]. Figure 9 shows a DocBook example, including language information. This information is not only a single identifier, like an option of the babel package, but allows the specification of variants of a language. It consists of a two-letter language code using lowercase letters, optionally followed by a two-letter country code using uppercase letters [1]: '`fr`' is for the French language, '`BE`' for Belgium. Such elements are not limited to a bibliography: as another example, the specification of an author's article also uses them [26, p. 143].

Another formalism belonging to the XML family widely used for bibliographical metadata is the *Dublin Core*,[23] but here the contents of the elements representing people's names are just strings; such elements are not structured by means of subelements as in DocBook. For example, an entity primarily responsible for making the content of a resource may be a person name and is specified by the `dc:creator` element. BibTeX-like syntax is used in practice, as shown in Figure 10; [27] gives some guidelines.

[23] The Dublin Core metadata standard is a simple yet effective element set for describing a wide range of networked resources. See [5] for more details.

```
<dc:creator>Kay, Guy Gavriel</dc:creator>
<dc:creator xml:lang="fr">
  <!--  xml:lang is a predefined attribute in XML
        [21, p. 41], its values are codes described in
        [1].
    -->
  Pierre Pelot
</dc:creator>
```

Figure 10: Examples with the *Dublin Core*.

5 Further development and conclusion

As mentioned above, MlBibTeX includes a compatibility mode, so it can apply 'old' bibliography styles, written using bst.[24] To put this implementation of bst into action, of course, we studied the behaviour of this language's functions precisely, preparing very many tests. That is why we hope that, in particular, we know precisely how BibTeX deals with names. This task of reverse engineering showed us that it was designed in order for current format operations to be specified concisely. The price paid is complicated specification of general cases.

Due to the huge number of BibTeX database files written by end-users, a successor of BibTeX has to ensure compatibility with existing .bib files. So we have to pay attention whenever we introduce new syntactic sugar: could existing files be processed as previously? Another question is: could we add more and more syntactic sugar to a formalism without damaging it? Redefining a new one might be better ...

We think that the work on the new features of MlBibTeX should be done based on a specification in XML, and some improvements can be given syntactic sugar, usable within .bib files. For example, we plan to add a `space-after` attribute to the `von` element:

```
<von space-after="no">d&apos;</von>
```

and specify empty space after a particle by an additional comma in `AUTHOR` and `EDITOR` fields:

```
{first => Guy, von => d',, last => Antin}
```

or — when the *von* part is specified last — :

```
{first => Gilles, last => Argyre,
 von => d',}
```

On the contrary, future features related to person names whose structure is to be studied will be added only to the XML document model of bibliographies, when it is ready. We think that BibTeX has not been much used for such names, so there should be no need for additional syntactic sugar and

[24] See [9] for an overview of the functions of this mode.

```
first => J.-H., last => Rosny, junior => jeune and
first => Pierre Alexis, last => Ponson du Terrail and
first => Eric, von => Van, last => Lustbader
```

```
((*name*)
 "Rosny, jeune, J.-H. and Ponson du Terrail, Pierre Alexis and Van Lustbader, Eric" .
 #(#((14 . 19) (7 . 12) (0 . 5) #f)    ; The order is First, Junior, Last, von. The First part of the first
                                       ; element starts at position 14 and ends just before position 19. This
                                       ; element does not have a von part.
   #((43 . 56) #f (24 . 41) #f)
   #((76 . 80) #f (65 . 74) (61 . 64))))
```

Figure 11: How MlBibTeX's names are sent to original BibTeX's functions.

MlBibTeX should be able to get such information by parsing XML files.

6 Acknowledgements

Many end-users told me that they had difficulty understanding names' specification within BibTeX. I was thinking of them when I was writing this article and I hope they will enjoy reading it as much as I enjoyed writing it. Many thanks to Karl Berry and Barbara Beeton, both of whom proofread a first version, suggested some improvements and additional examples.

A MlBibTeX's names within compatibility mode

As mentioned above, MlBibTeX allows users to run 'old' bibliography styles of BibTeX [19], by means of a *compatibility mode*, sketched in [9]. This compatibility mode, programmed in Scheme, actually uses a stack and allows users to learn the bst language easily, since they can run a bst program step by step. The data used within the result of parsing .bib files are serialised w.r.t. types used within the bst language's functions, that is, strings, integers, and literals [16, Table 13.8]. Each function belonging to the bst library is implemented by a Scheme function.

If you push a string and would like to give it to the Scheme function implementing `format.name$`, only the conventions of BibTeX will be put into action. If the value of an `AUTHOR` or `EDITOR` has been processed by MlBibTeX's parser and has to be passed to the compatibility mode, the transmitted value is a list consisting of a string $\mathcal{S}$, followed by a *vector*,[25] whose size is the number of the names of $\mathcal{S}$. Let us consider a name belonging to $\mathcal{S}$, for each part of it — *First*, *Junior*, *Last*, *von*, w.r.t. this order — a pair of indices shows where this part begins and ends. This part is replaced by the false value — `#f` — if it does not exist. Unless a name only consists of a *Last* part, the possible syntaxes for the string $\mathcal{S}$ are either (ii) or (iii) — cf. § 2.1.

An example is given in Figure 11: given three person names specified by means of keywords, we show the three elements of the corresponding structure passed to the compatibility mode. Index pairs are organised into vectors: one vector for the parts of each, these three vectors being elements of a vector of vectors. If such a list, beginning with the '`*name*`' marker, is popped by function other than `b-bst-format.name$` and `b-bst-num.names$`, the implementations of `format.name$` and `num.names$`, only the string is retained.

This design choice allows us to have BibTeX's behaviour by default if these two functions are given only a string. But if end-users take advantage of the keyword specification of name parts, we do not need workarounds within the strings passed to these two functions. We do not have to use additional braces or dummy LaTeX commands. Besides, our indices are coherent with the way used by Scheme for selecting substrings: they are characters beginning with a first inclusive index and ending with a second exclusive index. For example, let `s` be the long string given in Figure 11: the expression `(substring s 14 19)` evaluates to the string `"J.-H."`.

Last, the names of collaborators, introduced after an occurrence of the '`with`' keyword (cf. § 3.2), are ruled out. Finally, let us remark that these names would be processed improperly by BibTeX's bibliography styles.

[25] Vectors of Scheme are analogous to arrays of 'classical' programming languages.

References

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

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

[3] Jean-Christophe Filliâtre and Claude Marché: *The BibTeX2HTML Home Page*. June 2006. http://www.lri.fr/~filliatr/bibtex2html/.

[4] Hans Hagen: *ConTeXt, the Manual*. November 2001. http://www.pragma-ade.com/general/manuals/cont-enp.pdf.

[5] Diane Hillman: *Using Dublin Core*. November 2005. http://dublincore.org/documents/usageguide/.

[6] Jean-Michel Hufflen: "Multilingual Features for Bibliography Programs: From XML to MlBibTeX". In: *EuroTeX 2002*, pp. 46–59. Bachotek, Poland. April 2002.

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

[8] Jean-Michel Hufflen: "Bibliography Styles Easier with MlBibTeX". In: *Proc. EuroTeX 2005*, pp. 179–192. Pont-à Mousson, France. March 2005.

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

[10] Jean-Michel Hufflen: "MlBibTeX Meets ConTeXt". In: *EuroTeX 2006 conference, preprints*, pp. 71–76. Debrecen, Hungary. July 2006.

[11] 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.

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

[13] Marie-Paule Kluth : *FAQ LaTeX française pour débutants et confirmés*. Vuibert Informatique, Paris. Également disponible sur CTAN:help/LaTeX-FAQ-francaise/. Janvier 1999.

[14] Donald Ervin Knuth: *Computers & Typesetting. Vol. A: The TeXbook*. Addison-Wesley Publishing Company, Reading, Massachusetts. 1984.

[15] Leslie Lamport: *LaTeX: 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 LaTeX Companion*. 2nd edition. Addison-Wesley Publishing Company, Reading, Massachusetts. August 2004.

[17] Chuck Musciano and Bill Kennedy: *HTML & XHTML: The Definitive Guide*. 5th edition. O'Reilly & Associates, Inc. August 2002.

[18] *Oxford Advanced Learner's Dictionary of Current English*. Oxford University Press. 1989.

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

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

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

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

[23] 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.

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

[25] Eric van der Vlist: *XML Schema*. O'Reilly & Associates, Inc. June 2002.

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

[27] Andrew Waugh: *Representing People's Names in Dublin Core*. February 1998. http://dublincore.org/documents/name-representation/.