

# GB/T 7714-2015 BibTeX style

Zeping Lee\*

2018/08/05 v1.0.9

## 摘要

The `gbt7714` package provides a BibTeX implementation for the China's bibliography style standard GB/T 7714-2015. It consists of two `bst` files for numerical and `authoryear` styles as well as a `LaTeX` package which provides the citation style defined in the standard. It is compatible with `natbib` and supports language detection (Chinese and English) for each bibliography entry.

## 1 简介

GB/T 7714-2015 《信息与文献 参考文献著录规则》<sup>[1]</sup>（以下简称“国标”）是中国的参考文献推荐标准。本宏包是国标的 BibTeX<sup>[2]</sup> 实现，具有以下特性：

- 兼容 `natbib` 宏包<sup>[3]</sup>
- 支持顺序编码制和著者-出版年制两种风格
- 自动识别语言并进行相应处理
- 提供了简单的接口供用户修改样式

本宏包的主页：<https://github.com/zepinglee/gbt7714-bibtex-style>。

## 2 使用方法

`super` 按照国标的规定，参考文献的标注体系分为“顺序编码制”和“著者-出版年制”(authoryear)，  
`numbers` 其中顺序编码制根据引用标注样式的不同分为角标数字式 (`super`) 和与正文平排的数字式  
`authoryear` (`numbers`)。

用户应在导言区调用宏包 `gbt7714`，并在参数中选择参考文献的标注样式。默认的参数是 `super`，额外的参数会传递给 `natbib` 宏包，比如：

```
\usepackage[authoryear]{gbt7714}
```

然后不再需要调用 `\bibliographystyle` 命令设置参考文献列表风格。

使用时需要注意以下几点：

- 不再需要调用 `\bibliographystyle` 命令选择参考文献表的格式。
- `bib` 数据库应使用 UTF-8 编码。
- 使用著者-出版年制参考文献表时，中文的文献必须在 `key` 域填写作者姓名的拼音，才能按照拼音排序，详见第 5 节。

- `\cite` 在正文中引用文献时应使用 `\cite` 命令。同一处引用多篇文献时，应将各篇文献的 `key` 一同写在 `\cite` 命令中，如 `\cite{knuth84,lamport94,mittelbach04}`。如遇连续编号，可以自动转为起讫序号并用短横线连接。它可以自动排序并用处理连续编号。若需要标出引文的页码，可以标在 `\cite` 的可选参数中，如 `\cite[42]{knuth84}`。更多的引用标注方法可以参考 `natbib` 宏包的使用说明<sup>[3]</sup>。
- `\bibliography` 参考文献表可以在文中使用 `\bibliography` 命令调用。注意文献列表的样式已经在模板中根据选项设置，用户不再需要使用 `\bibliographystyle` 命令。

### 3 文献类型

国标中规定了 16 种参考文献类型，表 1 列举了 `bib` 数据库中对应的文献类型。这些尽可能兼容 `BibTeX` 的标准类型，但是新增了若干文献类型（带 \* 号）。

表 1: 全部文献类型

文献类型	标识代码	Entry Type
普通图书	M	book
图书的析出文献	M	incollection
会议录	C	proceedings
会议录的析出文献	C	inproceedings 或 conference
汇编	G	collection*
报纸	N	newspaper*
期刊的析出文献	J	article
学位论文	D	mastersthesis 或 phdthesis
报告	R	techreport
标准	S	standard*
专利	P	patent*
数据库	DB	database*
计算机程序	CP	software*
电子公告	EB	online*
档案	A	archive*
舆图	CM	map*
数据集	DS	dataset*
其他	Z	misc

### 4 著录项目

由于国标中规定的著录项目多于 `BibTeX` 的标准域，必须新增一些著录项目（带 \* 号），这些新增的类型在设计时参考了 `BibLaTeX`，如 `date` 和 `urldate`。本宏包支持的全部域如下：

- author** 主要责任者
- title** 题名
- mark\*** 文献类型标识
- medium\*** 载体类型标识

---

\* zepinglee AT gmail.com

**translator\*** 译者  
**editor** 编辑  
**organization** 组织（用于会议）  
**booktitle** 图书题名  
**series** 系列  
**journal** 期刊题名  
**edition** 版本  
**address** 出版地  
**publisher** 出版者  
**school** 学校（用于 phdthesis）  
**institution** 机构（用于 techreport）  
**year** 出版年  
**volume** 卷  
**number** 期（或者专利号）  
**pages** 引文页码  
**date\*** 更新或修改日期  
**urldate\*** 引用日期  
**url** 获取和访问路径  
**doi** 数字对象唯一标识符  
**language\*** 语言  
**key** 拼音（用于排序）

不支持的 BibTeX 标准著录项目有 `annotate`, `chapter`, `crossref`, `month`, `type`。

本宏包默认情况下可以自动识别文献语言，并自动处理文献类型和载体类型标识，但是在少数情况下需要用户手动指定，如：

```
@misc{citekey,  
  language = {japanese},  
  mark      = {Z},  
  medium    = {DK},  
  ...
```

可选的语言有 `english`, `chinese`, `japanese`, `russian`。

## 5 文献列表的排序

国标规定参考文献表采用著者-出版年制组织时，各篇文献首先按文种集中，然后按著者字顺和出版年排列；中文文献可以按著者汉语拼音字顺排列，也可以按著者的笔画笔顺排列。然而由于 BibTeX 功能的局限性，无法自动获取著者姓名的拼音或笔画笔顺，所以必须在 `bib` 数据库中的 `key` 域手动录入著者姓名的拼音，如：

```
@book{capital,  
  author = {马克思 and 恩格斯},  
  key    = {ma3 ke4 si1 en1 ge2 si1},  
  ...
```

表 2: 参考文献表样式的配置参数

参数值	默认值	功能
uppercase.name	#1	将著者姓名转为大写
max.num.authors	#3	输出著者的最多数量
period.between.author.year	#0	著者和年份之间使用句点连接
sentence.case.title	#1	将西文的题名转为 sentence case
link.title	#0	在题名上添加 url 的超链接
show.mark	#1	显示文献类型标识
italic.journal	#0	西文期刊名使用斜体
show.missing.address.publisher	#1	出版项缺失时显示“出版者不详”
show.url	#1	显示 url
show.doi	#1	显示 doi
show.note	#0	显示 note 域的信息

## 6 自定义样式

Bib<sub>T</sub><sub>E</sub>X 对自定义样式的支持比较有限，所以用户只能通过修改 bst 文件来修改文献列表的格式。本宏包提供了一些接口供用户更方便地修改。

在 bst 文件开始处的 load.config 函数中，有一组配置参数用来控制样式，表 2 列出了每一项的默认值和功能。若变量被设为 #1 则表示该项被启用，设为 #0 则不启用。默认的值是严格遵循国标的配置。

若用户需要定制更多内容，可以学习 bst 文件的语法并修改<sup>[4-6]</sup>，或者联系作者。

## 7 相关工作

TeX 社区也有其他关于 GB/T 7714 系列参考文献标准的工作。2005 年吴凯<sup>[7]</sup>发布了基于 GB/T 7714-2005 的 Bib<sub>T</sub><sub>E</sub>X 样式，支持顺序编码制和著者出版年制两种风格。李志奇<sup>[8]</sup>发布了严格遵循 GB/T 7714-2005 的 BibLaTeX 的样式。胡海星<sup>[9]</sup>提供了另一个 Bib<sub>T</sub><sub>E</sub>X 实现，还给每行 bst 代码写了 java 语言注释。沈周<sup>[10]</sup>基于 biblatex-caservector<sup>[11]</sup>进行修改，以符合国标的格式。胡振震发布了符合 GB/T 7714-2015 标准的 BibLaTeX 参考文献样式<sup>[12]</sup>，并进行了比较完善的持续维护。

## 参考文献

- [1] 中国国家标准委员会. 信息与文献 参考文献著录规则: GB/T 7714-2015[S]. 北京: 中国标准出版社, 2015.
- [2] PATASHNIK O. Bib<sub>T</sub><sub>E</sub>Xing[M/OL]. 1988. <http://mirrors.ctan.org/biblio/bibtex/base/btxdoc.pdf>.
- [3] DALY P W. Natural sciences citations and references[M/OL]. 1999. <http://mirrors.ctan.org/macros/latex/contrib/natbib/natbib.pdf>.
- [4] PATASHNIK O. Designing Bib<sub>T</sub><sub>E</sub>X styles[M/OL]. 1988. <http://mirrors.ctan.org/biblio/bibtex/base/btxhak.pdf>.

- [5] MARKEY N. Tame the beast[M/OL]. 2003. [http://mirrors.ctan.org/info/bibtex/tamethebeast/tt\\_b\\_en.pdf](http://mirrors.ctan.org/info/bibtex/tamethebeast/tt_b_en.pdf).
- [6] MITTELBACH F, GOOSSENS M, BRAAMS J, et al. The L<sup>A</sup>T<sub>E</sub>X companion[M]. 2nd ed. Reading, MA, USA: Addison-Wesley, 2004.
- [7] 吴凯. 发布 GBT7714-2005.bst version1 Beta 版[EB/OL]. 2006. <http://bbs.ctex.org/forum.php?mod=viewthread&tid=33591>.
- [8] 李志奇. 基于 biblatex 的符合 GBT7714-2005 的中文文献生成工具[EB/OL]. 2013. <http://bbs.ctex.org/forum.php?mod=viewthread&tid=74474>.
- [9] 胡海星. A GB/T 7714-2005 national standard compliant BibTeX style[EB/OL]. 2013. <https://github.com/Haixing-Hu/GBT7714-2005-BibTeX-Style>.
- [10] 沈周. 基于 caspervector 改写的符合 GB/T 7714-2005 标准的参考文献格式[EB/OL]. 2016. <https://github.com/szsdk/biblatex-gbt77142005>.
- [11] VECTOR C T. biblatex 参考文献和引用样式: caspervector[M/OL]. 2012. <http://mirrors.ctan.org/macros/latex/contrib/biblatex-contrib/biblatex-caspervector/doc/caspervector.pdf>.
- [12] 胡振震. 符合 GB/T 7714-2015 标准的 biblatex 参考文献样式[M/OL]. 2016. <http://mirrors.ctan.org/macros/latex/contrib/biblatex-contrib/biblatex-gb7714-2015/biblatex-gb7714-2015.pdf>.

## 版本历史

v1.0 (2018/01/01)	v1.0.6 (2018/05/10)
General: Initial release. . . . . 1	<b>thebibliography</b> : 文献列表的数字标签左对齐 8
v1.0.1 (2018/03/09)	<b>bst</b> : 不再处理中文标题的英文单词的大小写 . . . 22
General: 著者出版年制的文献引用不再排序 . . . 6	v1.0.7 (2018/05/12)
v1.0.2 (2018/03/16)	<b>bst</b> : 修正了检测 Unicode 语言 . . . . . 17
<b>bst</b> : 正确识别姓名中的“others” . . . . . 20	v1.0.8 (2018/06/23)
v1.0.3 (2018/03/29)	<b>bst</b> : 使用“~”连接英文姓名 . . . . . 20
<b>\cite</b> : 顺序编码制连续两个文献引用之间使用	支持 <b>howpublished</b> 中的 <b>url</b> . . . . . 32
连接号 . . . . . 8	新增接口供用户自定义样式 . . . . . 8
v1.0.4 (2018/04/12)	<b>\url</b> : 使用 <b>xurl</b> 的方法改进 URL 断行 . . . . . 8
<b>\cite</b> : 页码的连接号由 <b>en dash</b> 改为 <b>hyphen</b> . . . 8	v1.0.9 (2018/08/05)
v1.0.5 (2018/04/18)	<b>bst</b> : 不再转换题名 <b>volume</b> 的大小写 . . . . . 22
<b>bst</b> : 允许著录多个 DOI . . . . . 32	修正不显示 <b>url</b> 的选项 . . . . . 38
	增加选项在题名添加超链接 . . . . . 22

## A 宏包的代码实现

下面声明和处理宏包的选项，有 `authoryear` 和 `numbers`。

```
1 \langle *package\rangle
2 \newif\if@gbt@mmxv
3 \newif\if@gbt@numerical
4 \newif\if@gbt@super
5 \DeclareOption{2015}{\@gt@mmxvtrue}
6 \DeclareOption{2005}{\@gt@mmxvfalse}
7 \DeclareOption{super}{\@gt@numericaltrue\@gt@supertrue}
8 \DeclareOption{numbers}{\@gt@numericaltrue\@gt@superfalse}
9 \DeclareOption{authoryear}{\@gt@numericalfalse}
10 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{natbib}}
11 \ExecuteOptions{2015,super}
12 \ProcessOptions\relax
```

只在顺序编码时使用 `sort&compress`。

```
13 \if@gbt@numerical
14   \PassOptionsToPackage{sort&compress}{natbib}
15 \fi
16 \RequirePackage{natbib}
17 \RequirePackage{etoolbox}
18 \RequirePackage{url}
```

`\citestyle` 定义接口切换引用文献的标注法，可用 `\citestyle` 调用 `numerical` 或 `authoryear`，参见 `natbib`。

```
19 \newcommand\bibstyle@super{\bibpunct{[ ]{ }{, }{s}{, }{\textsuperscript{,}}{}}
20 \newcommand\bibstyle@numbers{\bibpunct{[ ]{ }{, }{n}{, }{, }{}}
21 \newcommand\bibstyle@authoryear{\bibpunct{( ){ }{; }{a}{, }{, }{}}
```

`\gbtbibstyle` 定义接口切换参考文献表的风格，可选 `authoryear` 和 `numerical`，这个仅用于 `chapterbib`。

```
22 \newcommand\gbtbibstyle[1]{%
23   \ifstrequal{#1}{numerical}{%
24     \if@gbt@mmxv
25       \bibliographystyle{gbt7714-unsrt}%
26     \else
27       \bibliographystyle{gbt7714-2005-unsrt}%
28     \fi
29   }{%
30     \ifstrequal{#1}{authoryear}{%
31       \if@gbt@mmxv
32         \bibliographystyle{gbt7714-plain}%
33       \else
34         \bibliographystyle{gbt7714-2005-plain}%
35       \fi
36     }{%
37       \PackageError{gbt7714}{Unknown argument #1.}%
38       {It should be `numerical' or `authoryear'.}%
39     }%
40   }%
41 }
```

处理宏包选项。

```
42 \if@gbt@numerical
43   \if@gbt@super
44     \citestyle{super}%
45     \gbtbibstyle{numerical}%
46   \else
47     \citestyle{numbers}
48     \gbtbibstyle{numerical}%
49   \fi
50 \else
51   \citestyle{authoryear}
52   \gbtbibstyle{authoryear}%
53 \fi
```

`\cite` 下面修改 `natbib` 的引用格式，主要是将页码写在上标位置。Numerical 模式的 `\citet` 的页码：

```
54 \newcommand\gbt@patchfailure[1]{%
55   \ClassError{ustcthis}{Failed to patch command \protect#1.\MessageBreak
56     Please contact the template author.%
57   }{}%
58 }
59 \patchcmd{\NAT@citexnum}{%
60   \@ifnum{\NAT@ctype=\z@}{%
61     \if*#2*\else\NAT@cmt#2\fi
62   }{}%
63   \NAT@mbox{\NAT@close}%
64 }{}%
65   \NAT@mbox{\NAT@close}%
66   \@ifnum{\NAT@ctype=\z@}{%
67     \if*#2*\else\textsuperscript{#2}\fi
68   }{}%
69 }{}{\gbt@patchfailure{\NAT@citexnum}}
```

Numerical 模式的 `\citet` 的页码：

```
70 \renewcommand\NAT@citesuper[3]{\ifNAT@swa
71   \if*#2*\else#2\NAT@spacechar\fi
72   \unskip\kern\p@\textsuperscript{\NAT@open#1\NAT@close\if*#3*\else#3\fi}%
73   \else #1\fi\endgroup}
```

Author-year 模式的 `\citet` 的页码：

```
74 \patchcmd{\NAT@citex}{%
75   \if*#2*\else\NAT@cmt#2\fi
76   \if\relax\NAT@date\relax\else\NAT@close\fi
77 }{}%
78   \if\relax\NAT@date\relax\else\NAT@close\fi
79   \if*#2*\else\textsuperscript{#2}\fi
80 }{}{\gbt@patchfailure{\NAT@citex}}
```

Author-year 模式的 `\citet` 的页码：

```
81 \renewcommand\NAT@cite%
82   [3]{\ifNAT@swa\NAT@open\if*#2*\else#2\NAT@spacechar\fi
83     #1\NAT@close\if*#3*\else\textsuperscript{#3}\fi\else#1\fi\endgroup}
```

在顺序编码制下，`natbib` 只有在三个以上连续文献引用才会使用连接号，这里修改为允许两个引用使用连接号。

```

84 \patchcmd{\NAT@citexnum}{%
85 \ifx\NAT@last@yr\relax
86 \def@NAT@last@yr{\@citea}%
87 \else
88 \def@NAT@last@yr{--\NAT@penalty}%
89 \fi
90 }{%
91 \def@NAT@last@yr{-\NAT@penalty}%
92 }{\gbt@patchfailure{\NAT@citexnum}}

```

`thebibliography` 参考文献列表的标签左对齐

```

93 \renewcommand\@biblabel[1]{[#1]\hfill}

```

`\url` 使用 `xurl` 宏包的方法，增加 URL 可断行的位置。

```

94 \def\UrlBreaks{%
95 \do\/%
96 \do\a\do\b\do\c\do\d\do\e\do\f\do\g\do\h\do\i\do\j\do\k\do\l%
97 \do\m\do\n\do\o\do\p\do\q\do\r\do\s\do\t\do\u\do\v\do\w\do\x\do\y\do\z%
98 \do\A\do\B\do\C\do\D\do\E\do\F\do\G\do\H\do\I\do\J\do\K\do\L%
99 \do\M\do\N\do\O\do\P\do\Q\do\R\do\S\do\T\do\U\do\V\do\W\do\X\do\Y\do\Z%
100 \do\0\do\1\do\2\do\3\do\4\do\5\do\6\do\7\do\8\do\9\do=\do/\do.\do:%
101 \do*\do-\do\~\do\' \do\" \do\~}
102 \Urlmuskip=0mu plus 0.1mu
103 \</package>

```

## B BibTeX 样式的代码实现

### B.1 自定义选项

`bst` 这里定义了一些变量用于定制样式，可以在下面的 `load.config` 函数中选择是否启用。

```

104 (*authoryear | numerical)
105 INTEGERS {
106 uppercase.name
107 max.num.authors
108 period.between.author.year
109 sentence.case.title
110 link.title
111 show.mark
112 italic.journal
113 show.missing.address.publisher
114 show.url
115 show.doi
116 show.note
117 }
118

```

下面每个变量若被设为 `#1` 则启用该项，若被设为 `#0` 则不启用。默认的值是严格遵循国标的配置。

```

119 FUNCTION {load.config}
120 {

```

英文姓名转为全大写:

```
121 #1 'uppercase.name :=
```

最多显示的作者数量:

```
122 #3 'max.num.authors :=
```

采用著者-出版年制时, 作者姓名与年份之间使用句点连接:

```
123 #0 'period.between.author.year :=
```

英文标题转为 sentence case (句首字母大写, 其余小写):

```
124 #1 'sentence.case.title :=
```

在标题添加超链接:

```
125 #0 'link.title :=
```

著录文献类型标识 (比如“[M/OL]“):

```
126 #1 'show.mark :=
```

期刊名使用斜体:

```
127 #0 'italic.journal :=
```

无出版地或出版者时, 著录“出版地不详”, “出版者不详”, “S.l.”或“s.n.”:

```
128 #1 'show.missing.address.publisher :=
```

是否著录 URL:

```
129 #1 'show.url :=
```

是否著录 DOI:

```
130 <*2015>
131 #1 'show.doi :=
132 </2015>
133 <*2005>
134 #0 'show.doi :=
135 </2005>
```

在每一条文献最后输出注释 (note) 的内容:

```
136 #0 'show.note :=
137 }
138
```

## B.2 The ENTRY declaration

Like Scribe's (according to pages 231-2 of the April '84 edition), but no fullauthor or editors fields because BibTeX does name handling. The annotate field is commented out here because this family doesn't include an annotated bibliography style. And in addition to the fields listed here, BibTeX has a built-in crossref field, explained later.

```
139 ENTRY
140 { address
141   author
142   booktitle
143   date
144   doi
145   edition
146   editor
147   howpublished
148   institution
149   journal
150   key
151   language
152   mark
153   medium
```

```

154  note
155  number
156  organization
157  pages
158  publisher
159  school
160  series
161  title
162  translator
163  url
164  urldate
165  volume
166  year
167  }
168  { entry.lang entry.is.electronic }

```

These string entry variables are used to form the citation label. In a storage pinch, `sort.label` can be easily computed on the fly.

```

169  { label extra.label sort.label short.list entry.mark entry.url }
170

```

### B.3 Entry functions

Each entry function starts by calling `output.bibitem`, to write the `\bibitem` and its arguments to the `.BBL` file. Then the various fields are formatted and printed by `output` or `output.check`. Those functions handle the writing of separators (commas, periods, `\newblock`'s), taking care not to do so when they are passed a null string. Finally, `fin.entry` is called to add the final period and finish the entry.

A bibliographic reference is formatted into a number of 'blocks': in the open format, a block begins on a new line and subsequent lines of the block are indented. A block may contain more than one sentence (well, not a grammatical sentence, but something to be ended with a sentence ending period). The entry functions should call `new.block` whenever a block other than the first is about to be started. They should call `new.sentence` whenever a new sentence is to be started. The output functions will ensure that if two `new.sentence`'s occur without any non-null string being output between them then there won't be two periods output. Similarly for two successive `new.block`'s.

The output routines don't write their argument immediately. Instead, by convention, that argument is saved on the stack to be output next time (when we'll know what separator needs to come after it). Meanwhile, the output routine has to pop the pending output off the stack, append any needed separator, and write it.

To tell which separator is needed, we maintain an `output.state`. It will be one of these values: `before.all` just after the `\bibitem` `mid.sentence` in the middle of a sentence: `comma` needed if more sentence is output `after.sentence` just after a sentence: `period` needed `after.block` just after a block (and sentence): `period` and `\newblock` needed. Note: These styles don't use `after.sentence`

VAR: `output.state` : INTEGER – state variable for output

The `output.nonnull` function saves its argument (assumed to be nonnull) on the stack, and writes the old saved value followed by any needed separator. The ordering of the tests is decreasing frequency of occurrence.

由于专著中的析出文献需要用到很特殊的“//”，所以我又加了一个 `after.slash`。其他需要在特定符号后面输出，所以写了一个 `output.after`。

```

output.nonnull(s) ==
BEGIN

```

```

s := argument on stack
if output.state = mid.sentence then
  write$(pop() * ",")
  -- "pop" isn't a function: just use stack top
else
  if output.state = after.block then
    write$(add.period$(pop()))
    newline$
    write$("\newblock_")
  else
    if output.state = before.all then
      write$(pop())
    else      -- output.state should be after.sentence
      write$(add.period$(pop()) * ",")
    fi
  fi
  output.state := mid.sentence
fi
push s on stack
END

```

The output function calls output.nonnull if its argument is non-empty; its argument may be a missing field (thus, not necessarily a string)

```

output(s) ==
BEGIN
  if not empty$(s) then output.nonnull(s)
  fi
END

```

The output.check function is the same as the output function except that, if necessary, output.check warns the user that the t field shouldn't be empty (this is because it probably won't be a good reference without the field; the entry functions try to make the formatting look reasonable even when such fields are empty).

```

output.check(s,t) ==
BEGIN
  if empty$(s) then
    warning$("empty_" * t * "_in_" * cite$)
  else output.nonnull(s)
  fi
END

```

The output.bibitem function writes the \bibitem for the current entry (the label should already have been set up), and sets up the separator state for the output functions. And, it leaves a string on the stack as per the output convention.

```

output.bibitem ==
BEGIN
  newline$
  write$("\bibitem{")      % for alphabetic labels,
  write$(label)          % these three lines
  write$("{}")           % are used
  write$("\bibitem{")    % this line for numeric labels
  write$(cite$)
  write$("{}")
  push "" on stack
  output.state := before.all
END

```

The `fin.entry` function finishes off an entry by adding a period to the string remaining on the stack. If the state is still `before.all` then nothing was produced for this entry, so the result will look bad, but the user deserves it. (We don't omit the whole entry because the entry was cited, and a `bibitem` is needed to define the citation label.)

```
fin.entry ==
BEGIN
  write$(add.period$(pop()))
  newline$
END
```

The `new.block` function prepares for a new block to be output, and `new.sentence` prepares for a new sentence.

```
new.block ==
BEGIN
  if output.state <> before.all then
    output.state := after.block
  fi
END
```

```
new.sentence ==
BEGIN
  if output.state <> after.block then
    if output.state <> before.all then
      output.state := after.sentence
    fi
  fi
END
```

```
171 INTEGERS { output.state before.all mid.sentence after.sentence after.block after.slash }
172
173 INTEGERS { lang.zh lang.ja lang.en lang.ru lang.other }
174
175 INTEGERS { charptr len }
176
177 FUNCTION {init.state.consts}
178 { #0 'before.all :=
179   #1 'mid.sentence :=
180   #2 'after.sentence :=
181   #3 'after.block :=
182   #4 'after.slash :=
183   #3 'lang.zh :=
184   #4 'lang.ja :=
185   #1 'lang.en :=
186   #2 'lang.ru :=
187   #0 'lang.other :=
188 }
189
```

下面是一些常量的定义

```
190 FUNCTION {bbl.anonymous}
191 { lang.zh entry.lang =
192   { "佚名" }
193   { "Anon" }
194   if$
195 }
196
197 FUNCTION {bbl.space} { "\ " }
198
199 FUNCTION {bbl.et.al}
200 { lang.zh entry.lang =
```

```

201 { " 等" }
202 { lang.ja entry.lang =
203   { " 他" }
204   { lang.ru entry.lang =
205     { "идр" }
206     { "et~al." }
207     if$
208   }
209   if$
210 }
211 if$
212 }
213
214 FUNCTION {bbl.colon} { ": " }
215
216 ⟨*2015⟩
217 FUNCTION {bbl.wide.space} { "\quad " }
218 ⟨/2015⟩
219 ⟨*2005⟩
220 FUNCTION {bbl.wide.space} { "\ " }
221 ⟨/2005⟩
222
223 FUNCTION {bbl.slash} { "//\allowbreak{" }
224
225 FUNCTION {bbl.sine.loco}
226 { lang.zh entry.lang =
227   { "[出版地不详]" }
228   { "[S.l.]" }
229   if$
230 }
231
232 FUNCTION {bbl.sine.nomine}
233 { lang.zh entry.lang =
234   { "[出版者不详]" }
235   { "[s.n.]" }
236   if$
237 }
238
239 FUNCTION {bbl.sine.loco.sine.nomine}
240 { lang.zh entry.lang =
241   { "[出版地不详: 出版者不详]" }
242   { "[S.l.: s.n.]" }
243   if$
244 }
245

```

These three functions pop one or two (integer) arguments from the stack and push a single one, either 0 or 1. The 'skip\$ in the 'and' and 'or' functions are used because the corresponding if\$ would be idempotent

```

246 FUNCTION {not}
247 { { #0 }
248   { #1 }
249   if$
250 }
251
252 FUNCTION {and}
253 { 'skip$
254   { pop$ #0 }
255   if$
256 }
257
258 FUNCTION {or}
259 { { pop$ #1 }

```

```

260     'skip$
261   if$
262 }
263

```

the variables s and t are temporary string holders

```

264 STRINGS { s t }
265
266 FUNCTION {output.nonnull}
267 { 's :=
268   output.state mid.sentence =
269   { ", " * write$ }
270   { output.state after.block =
271     { add.period$ write$
272       newline$
273       "\newblock " write$
274     }
275     { output.state before.all =
276       'write$
277       { output.state after.slash =
278         { bbl.slash * write$ }
279         { add.period$ " " * write$ }
280       }
281     }
282     if$
283   }
284   if$
285   mid.sentence 'output.state :=
286 }
287 if$
288 s
289 }
290
291 FUNCTION {output}
292 { duplicate$ empty$
293   'pop$
294   'output.nonnull
295   if$
296 }
297
298 FUNCTION {output.after}
299 { 't :=
300   duplicate$ empty$
301   'pop$
302   { 's :=
303     output.state mid.sentence =
304     { t * write$ }
305     { output.state after.block =
306       { add.period$ write$
307         newline$
308         "\newblock " write$
309       }
310       { output.state before.all =
311         'write$
312         { output.state after.slash =
313           { bbl.slash * write$ }
314           { add.period$ " " * write$ }
315         }
316       }
317       if$
318     }
319     if$
320     mid.sentence 'output.state :=
321   }

```

```

322     if$
323     s
324 }
325 if$
326 }
327
328 FUNCTION {output.check}
329 { 't :=
330   duplicate$ empty$
331   { pop$ "empty " t * " in " * cite$ * warning$ }
332   'output.nonnull
333   if$
334 }
335

```

This function finishes all entries.

```

336 FUNCTION {fin.entry}
337 { add.period$
338   write$
339   newline$
340 }
341
342 FUNCTION {new.block}
343 { output.state before.all =
344   'skip$
345   { output.state after.slash =
346     'skip$
347     { after.block 'output.state := }
348     if$
349   }
350   if$
351 }
352
353 FUNCTION {new.sentence}
354 { output.state after.block =
355   'skip$
356   { output.state before.all =
357     'skip$
358     { output.state after.slash =
359       'skip$
360       { after.sentence 'output.state := }
361       if$
362     }
363     if$
364   }
365   if$
366 }
367
368 FUNCTION {new.slash}
369 { output.state before.all =
370   'skip$
371   { after.slash 'output.state := }
372   if$
373 }
374

```

Sometimes we begin a new block only if the block will be big enough. The `new.block.checka` function issues a `new.block` if its argument is nonempty; `new.block.checkb` does the same if either of its TWO arguments is nonempty.

```

375 FUNCTION {new.block.checka}
376 { empty$
377   'skip$
378   'new.block

```

```

379  if$
380 }
381
382 FUNCTION {new.block.checkb}
383 { empty$
384  swap$ empty$
385  and
386   'skip$
387   'new.block
388  if$
389 }
390

```

The `new.sentence.check` functions are analogous.

```

391 FUNCTION {new.sentence.checka}
392 { empty$
393   'skip$
394   'new.sentence
395  if$
396 }
397
398 FUNCTION {new.sentence.checkb}
399 { empty$
400  swap$ empty$
401  and
402   'skip$
403   'new.sentence
404  if$
405 }
406

```

## B.4 Formatting chunks

Here are some functions for formatting chunks of an entry. By convention they either produce a string that can be followed by a comma or period (using `add.period$`, so it is OK to end in a period), or they produce the null string.

A useful utility is the `field.or.null` function, which checks if the argument is the result of pushing a ‘missing’ field (one for which no assignment was made when the current entry was read in from the database) or the result of pushing a string having no non-white-space characters. It returns the null string if so, otherwise it returns the field string. Its main (but not only) purpose is to guarantee that what’s left on the stack is a string rather than a missing field.

```

field.or.null(s) ==
BEGIN
  if empty$(s) then return ""
  else return s
END

```

Another helper function is `emphasize`, which returns the argument emphasised, if that is non-empty, otherwise it returns the null string. Italic corrections aren’t used, so this function should be used when punctuation will follow the result.

```

emphasize(s) ==
BEGIN
  if empty$(s) then return ""
  else return "{\em_} * s * {"

```

The ‘`pop$`’ in this function gets rid of the duplicate ‘empty’ value and the ‘`skip$`’ returns the duplicate field value

```

407 FUNCTION {field.or.null}
408 { duplicate$ empty$
409   { pop$ "" }
410   'skip$
411   if$
412 }
413
414 FUNCTION {italicize}
415 { duplicate$ empty$
416   { pop$ "" }
417   { "\textit{" swap$ * "" * }
418   if$
419 }
420

```

#### B.4.1 Detect Language

```

421 INTEGERS { byte second.byte }
422
423 INTEGERS { char.lang tmp.lang }
424
425 STRINGS { tmp.str }
426
427 FUNCTION {get.str.lang}
428 { 'tmp.str :=
429   lang.other 'tmp.lang :=
430   #1 'charptr :=
431   tmp.str text.length$ #1 + 'len :=
432   { charptr len < }
433   { tmp.str charptr #1 substring$ chr.to.int$ 'byte :=
434     byte #128 <
435     { charptr #1 + 'charptr :=
436       byte #64 > byte #91 < and byte #96 > byte #123 < and or
437       { lang.en 'char.lang := }
438       { lang.other 'char.lang := }
439       if$
440     }
441     { tmp.str charptr #1 + #1 substring$ chr.to.int$ 'second.byte :=
442       byte #224 <

```

俄文西里尔字母: U+0400 到 U+052F, 对应 UTF-8 从 D0 80 到 D4 AF.

```

443     { charptr #2 + 'charptr :=
444       byte #207 > byte #212 < and
445       byte #212 = second.byte #176 < and or
446       { lang.ru 'char.lang := }
447       { lang.other 'char.lang := }
448       if$
449     }
450     { byte #240 <

```

CJK Unified Ideographs: U+4E00–U+9FFF; UTF-8: E4 B8 80–E9 BF BF.

```

451     { charptr #3 + 'charptr :=
452       byte #227 > byte #234 < and
453       { lang.zh 'char.lang := }

```

CJK Unified Ideographs Extension A: U+3400–U+4DBF; UTF-8: E3 90 80–E4 B6 BF.

```

454     { byte #227 =
455       { second.byte #143 >
456       { lang.zh 'char.lang := }

```

日语假名: U+3040–U+30FF, UTF-8: E3 81 80–E3 83 BF.

```

457     { second.byte #128 > second.byte #132 < and
458       { lang.ja 'char.lang := }
459       { lang.other 'char.lang := }
460     if$

```

```

461         }
462         if$
463     }

```

CJK Compatibility Ideographs: U+F900–U+FAFF, UTF-8: EF A4 80–EF AB BF.

```

464         { byte #239 =
465           second.byte #163 > second.byte #172 < and and
466           { lang.zh 'char.lang := }
467           { lang.other 'char.lang := }
468         if$
469       }
470     if$
471   }
472   if$
473 }

```

CJK Unified Ideographs Extension B–F: U+20000–U+2EBEF, UTF-8: F0 A0 80 80–F0 AE AF AF.

CJK Compatibility Ideographs Supplement: U+2F800–U+2FA1F, UTF-8: F0 AF A0 80–F0 AF A8 9F.

```

474     { charptr #4 + 'charptr :=
475       byte #240 = second.byte #159 > and
476       { lang.zh 'char.lang := }
477       { lang.other 'char.lang := }
478     if$
479   }
480   if$
481 }
482   if$
483 }
484   if$
485   char.lang tmp.lang >
486     { char.lang 'tmp.lang := }
487     'skip$
488   if$
489 }
490 while$
491 tmp.lang
492 }
493
494 FUNCTION {check.entry.lang}
495 { author field.or.null
496   title field.or.null *
497   get.str.lang
498 }
499
500 FUNCTION {set.entry.lang}
501 { language empty$
502   { check.entry.lang }
503   { language "english" = language "american" = or language "british" = or
504     { lang.en }
505     { language "chinese" =
506       { lang.zh }
507       { language "japanese" =
508         { lang.ja }
509         { language "russian" =
510           { lang.ru }
511           { check.entry.lang }
512         if$
513       }
514     if$
515   }
516   if$
517 }
518   if$
519 }

```

```

520 if$
521 'entry.lang :=
522 }
523

```

#### B.4.2 Format names

The `format.names` function formats the argument (which should be in BibTeX name format) into "First Von Last, Junior", separated by commas and with an "and" before the last (but ending with "et al." if the last of multiple authors is "others"). This function's argument should always contain at least one name.

```

VAR: nameptr, namesleft, numnames: INTEGER
pseudoVAR: namerresult: STRING      (it's what's accumulated on the stack)

format.names(s) ==
BEGIN
  nameptr := 1
  numnames := num.names$(s)
  namesleft := numnames
  while namesleft > 0
    do
      % for full names:
      t := format.name$(s, nameptr, "{ff~}{vv~}{ll}{,jj}")
      % for abbreviated first names:
      t := format.name$(s, nameptr, "{f.~}{vv~}{ll}{,jj}")
      if nameptr > 1 then
        if namesleft > 1 then namerresult := namerresult * ", " * t
        else if numnames > 2
          then namerresult := namerresult * ","
          fi
          if t = "others"
            then namerresult := namerresult * "_et~al."
            else namerresult := namerresult * "_and_" * t
          fi
        fi
      else namerresult := t
      fi
      nameptr := nameptr + 1
      namesleft := namesleft - 1
    od
  return namerresult
END

```

The `format.authors` function returns the result of `format.names(author)` if the author is present, or else it returns the null string

```

format.authors ==
BEGIN
  if empty$(author) then return ""
  else return format.names(author)
  fi
END

```

`Format.editors` is like `format.authors`, but it uses the editor field, and appends ", editor" or ", editors"

```

format.editors ==
BEGIN
  if empty$(editor) then return ""
  else
    if num.names$(editor) > 1 then
      return format.names(editor) * ",_editors"
    else

```

```

        return format.names(editor) * ",_editor"
    fi
fi
END

```

Other formatting functions are similar, so no "comment version" will be given for them.

```

524 INTEGERS { nameptr namesleft numnames name.lang }
525
526 FUNCTION {format.names}
527 { 's :=
528   #1 'nameptr :=
529   s num.names$ 'numnames :=
530   numnames 'namesleft :=
531   { namesleft #0 > }
532   { s nameptr "{vv~}{ll}{, jj}{, ff}" format.name$ 't :=
533     nameptr max.num.authors >
534     { bbl.et.al
535       #1 'namesleft :=
536     }
537     { t "others" =
538       { bbl.et.al }
539       { t get.str.lang 'name.lang :=
540         name.lang lang.en =
541         { t #1 "{vv~}{ll}{~f{~}}" format.name$
542           uppercase.name
543           { "u" change.case$ }
544           'skip$
545           if$
546           t #1 "{, jj}" format.name$ *
547         }
548         { t #1 "{ll}{ff}" format.name$ }
549         if$
550       }
551     }
552   }
553   if$
554   nameptr #1 >
555   { ", " swap$ * * }
556   'skip$
557   if$
558   nameptr #1 + 'nameptr :=
559   namesleft #1 - 'namesleft :=
560 }
561 while$
562 }
563
564 FUNCTION {format.key}
565 { empty$
566   { key field.or.null }
567   { "" }
568   if$
569 }
570
571 FUNCTION {format.authors}
572 { author empty$
573   (*authoryear)
574   { bbl.anonymous }
575   (/authoryear)
576   (*numerical)
577   { "" }
578   (/numerical)
579   { author format.names }
580   if$
581 }

```

```

582
583 FUNCTION {format.editors}
584 { editor empty$
585   { "" }
586   { editor format.names }
587   if$
588 }
589
590 FUNCTION {format.translators}
591 { translator empty$
592   { "" }
593   { translator format.names
594     lang.zh entry.lang =
595     { translator num.names$ #3 >
596       { " 译" * }
597       { ", 译" * }
598       if$
599     }
600     'skip$
601     if$
602   }
603   if$
604 }
605
606 FUNCTION {format.full.names}
607 { 's :=
608   #1 'nameptr :=
609   s num.names$ 'numnames :=
610   numnames 'namesleft :=
611   { namesleft #0 > }
612   { s nameptr "{vv~}{ll}{, jj}{, ff}" format.name$ 't :=
613     t get.str.lang 'name.lang :=
614     name.lang lang.en =
615     { t #1 "{vv~}{ll}" format.name$ 't := }
616     { t #1 "{ll}{ff}" format.name$ 't := }
617     if$
618     nameptr #1 >
619     {
620       namesleft #1 >
621       { ", " * t * }
622       {
623         numnames #2 >
624         { ", " * }
625         'skip$
626         if$
627         t "others" =
628         { " et~al." * }
629         { " and " * t * }
630         if$
631       }
632       if$
633     }
634     't
635     if$
636     nameptr #1 + 'nameptr :=
637     namesleft #1 - 'namesleft :=
638   }
639   while$
640 }
641
642 FUNCTION {author.editor.full}
643 { author empty$
644   { editor empty$
645     { "" }

```

```

646     { editor format.full.names }
647     if$
648   }
649   { author format.full.names }
650   if$
651 }
652
653 FUNCTION {author.full}
654 { author empty$
655   { "" }
656   { author format.full.names }
657   if$
658 }
659
660 FUNCTION {editor.full}
661 { editor empty$
662   { "" }
663   { editor format.full.names }
664   if$
665 }
666
667 FUNCTION {make.full.names}
668 { type$ "book" =
669   type$ "inbook" =
670   or
671   'author.editor.full
672   { type$ "collection" =
673     type$ "proceedings" =
674     or
675     'editor.full
676     'author.full
677     if$
678   }
679   if$
680 }
681
682 FUNCTION {output.bibitem}
683 { newline$
684   "\bibitem[" write$
685   label write$
686   ")" make.full.names duplicate$ short.list =
687   { pop$ }
688   { * }
689   if$
690   "{" * write$
691   cite$ write$
692   "}" write$
693   newline$
694   ""
695   before.all 'output.state :=
696 }
697

```

### B.4.3 Format title

The `format.title` function is used for non-book-like titles. For most styles we convert to lower-case (except for the very first letter, and except for the first one after a colon (followed by whitespace)), and hope the user has brace-surrounded words that need to stay capitalized; for some styles, however, we leave it as it is in the database.

```

698 FUNCTION {change.sentence.case}
699 { entry.lang lang.en =
700   { "t" change.case$ }

```

```

701   'skip$
702   if$
703 }
704
705 FUNCTION {add.link}
706 { url empty$ not
707   { "\href{" url * "}{ * swap$ * "}" * }
708   { doi empty$ not
709     { "\href{http://dx.doi.org/" doi * "}{ * swap$ * "}" * }
710     'skip$
711     if$
712   }
713   if$
714 }
715
716 FUNCTION {format.title}
717 { title empty$
718   { "" }
719   { title
720     sentence.case.title
721     'change.sentence.case
722     'skip$
723     if$
724     link.title
725     'add.link
726     'skip$
727     if$
728   }
729   if$
730 }
731

```

For several functions we'll need to connect two strings with a tie (~) if the second one isn't very long (fewer than 3 characters). The tie.or.space.connect function does that. It concatenates the two strings on top of the stack, along with either a tie or space between them, and puts this concatenation back onto the stack:

```

tie.or.space.connect(str1,str2) ==
BEGIN
  if text.length$(str2) < 3
  then return the concatenation of str1, "~", and str2
  else return the concatenation of str1, "_", and str2
END

```

```

732 FUNCTION {tie.or.space.connect}
733 { duplicate$ text.length$ #3 <
734   { "~" }
735   { " " }
736   if$
737   swap$ * *
738 }
739

```

The either.or.check function complains if both fields or an either-or pair are nonempty.

```

either.or.check(t,s) ==
BEGIN
  if empty$(s) then
    warning$(can't use both "_*t_*" fields in "_*cite$)
fi
END

```

```

740 FUNCTION {either.or.check}
741 { empty$

```

```

742     'pop$
743     { "can't use both " swap$ * " fields in " * cite$ * warning$ }
744   if$
745 }
746

```

The `format.bvolume` function is for formatting the volume and perhaps series name of a multivolume work. If both a volume and a series field are there, we assume the series field is the title of the whole multivolume work (the title field should be the title of the thing being referred to), and we add an "of <series>". This function is called in mid-sentence.

The `format.number.series` function is for formatting the series name and perhaps number of a work in a series. This function is similar to `format.bvolume`, although for this one the series must exist (and the volume must not exist). If the number field is empty we output either the series field unchanged if it exists or else the null string. If both the number and series fields are there we assume the series field gives the name of the whole series (the title field should be the title of the work being one referred to), and we add an "in <series>". We capitalize Number when this function is used at the beginning of a block.

```

747 FUNCTION {is.digit}
748 { duplicate$ empty$
749   { pop$ #0 }
750   { chr.to.int$
751     duplicate$ "0" chr.to.int$ <
752     { pop$ #0 }
753     { "9" chr.to.int$ >
754       { #0 }
755       { #1 }
756     if$
757   }
758   if$
759 }
760 if$
761 }
762
763 FUNCTION {is.number}
764 { 's :=
765   s empty$
766   { #0 }
767   { s text.length$ 'charptr :=
768     { charptr #0 >
769       s charptr #1 substring$ is.digit
770       and
771     }
772     { charptr #1 - 'charptr := }
773     while$
774     charptr not
775   }
776   if$
777 }
778
779 FUNCTION {format.volume}
780 { volume empty$
781   { "" }
782   { lang.zh entry.lang =
783     { " 第 " volume * " 卷" * }
784     { "volume" volume tie.or.space.connect }
785   if$
786 }
787 if$
788 }

```

```

789
790 FUNCTION {format.number}
791 { number empty$
792   { "" }
793   { lang.zh entry.lang =
794     { "第" number * "册" * }
795     { "number" number tie.or.space.connect }
796     if$
797   }
798   if$
799 }
800
801 FUNCTION {format.volume.number}
802 { volume empty$ not
803   { format.volume }
804   { format.number }
805   if$
806 }
807
808 FUNCTION {format.series.vol.num.title}
809 { format.volume.number 's :=
810   series empty$ not
811   { series
812     sentence.case.title
813     'change.sentence.case
814     'skip$
815     if$
816     bbl.colon *
817     s empty$ not
818     { s * bbl.wide.space * }
819     'skip$
820     if$
821     title
822     sentence.case.title
823     'change.sentence.case
824     'skip$
825     if$
826     *
827   }
828   { title
829     sentence.case.title
830     'change.sentence.case
831     'skip$
832     if$
833     s empty$ not
834     { bbl.colon * s * }
835     'skip$
836     if$
837   }
838   if$
839   link.title
840   'add.link
841   'skip$
842   if$
843 }
844
845 FUNCTION {format.series.vol.num.booktitle}
846 { format.volume.number 's :=
847   series empty$ not
848   { series bbl.colon *
849     s empty$ not
850     { s * bbl.wide.space * }
851     'skip$
852     if$

```

```

853     booktitle *
854   }
855   { booktitle
856     s empty$ not
857     { bbl.colon * s * }
858     'skip$
859     if$
860   }
861 if$
862 }
863
864 FUNCTION {format.journal}
865 { journal
866   italic.journal
867   'italicize
868   'skip$
869   if$
870 }
871

```

#### B.4.4 Format entry type mark

```

872 FUNCTION {set.entry.mark}
873 { entry.mark empty$ not
874   'pop$
875   { mark empty$ not
876     { pop$ mark 'entry.mark := }
877     { 'entry.mark := }
878     if$
879   }
880 if$
881 }
882
883 FUNCTION {format.mark}
884 { show.mark
885   { medium empty$ not
886     { entry.mark "/" * medium * 'entry.mark := }
887     { entry.is.electronic
888       { entry.mark "/OL" * 'entry.mark := }
889       'skip$
890       if$
891     }
892     if$
893     "\allowbreak[" entry.mark * "]" *
894   }
895   { "" }
896 if$
897 }
898

```

#### B.4.5 Format edition

The `format.edition` function appends " edition" to the edition, if present. We lowercase the edition (it should be something like "Third"), because this doesn't start a sentence.

```

899 FUNCTION {num.to.ordinal}
900 { duplicate$ text.length$ 'charptr :=
901   duplicate$ charptr #1 substring$ 's :=
902   s "1" =
903     { "st" * }
904   { s "2" =
905     { "nd" * }
906     { s "3" =
907       { "rd" * }
908       { "th" * }

```

```

909     if$
910     }
911     if$
912     }
913     if$
914 }
915
916 FUNCTION {format.edition}
917 { edition empty$
918   { "" }
919   { edition is.number
920     { lang.zh entry.lang =
921       { edition " 版" * }
922       { edition num.to.ordinal " ed." * }
923       if$
924     }
925     { entry.lang lang.en =
926       { edition change.sentence.case 's :=
927         s "Revised" = s "Revised edition" = or
928         { "Rev. ed." }
929         { s " ed." *}
930       if$
931     }
932     { edition }
933     if$
934   }
935   if$
936 }
937 if$
938 }
939

```

#### B.4.6 Format publishing items

出版地址和出版社会有“[S.l.: s.n.]”的情况，所以必须一起处理。

```

940 FUNCTION {format.publisher}
941 { publisher empty$ not
942   { publisher }
943   { school empty$ not
944     { school }
945     { organization empty$ not
946       { organization }
947       { institution empty$ not
948         { institution }
949         { "" }
950       if$
951     }
952     if$
953   }
954   if$
955 }
956 if$
957 }
958
959 FUNCTION {format.address.publisher}
960 { address empty$ not
961   { address
962     format.publisher empty$ not
963     { bbl.colon * format.publisher * }
964     { entry.is.electronic not show.missing.address.publisher and
965       { bbl.colon * bbl.sine.nomine * }
966       'skip$
967     if$

```

```

968     }
969     if$
970 }
971 { entry.is.electronic not show.missing.address.publisher and
972     { format.publisher empty$ not
973         { bbl.sine.loco bbl.colon * format.publisher * }
974         { bbl.sine.loco.sine.nomine }
975     if$
976 }
977 { format.publisher empty$ not
978     { format.publisher }
979     { "" }
980 if$
981 }
982 if$
983 }
984 if$
985 }
986

```

#### B.4.7 Format date

The `format.date` function is for the month and year, but we give a warning if there's an empty year but the month is there, and we return the empty string if they're both empty.

`Newspaper` 和 `papent` 要显示完整的日期，同时不再显示修改日期。但是在 `author-year` 模式下，需要单独设置 `format.year`。

```

987 FUNCTION {extract.before.dash}
988 { duplicate$ empty$
989   { pop$ "" }
990   { 's :=
991     #1 'charptr :=
992     s text.length$ #1 + 'len :=
993     { charptr len <
994       s charptr #1 substring$ "-" = not
995       and
996     }
997     { charptr #1 + 'charptr := }
998     while$
999     s #1 charptr #1 - substring$
1000   }
1001 if$
1002 }
1003
1004 FUNCTION {extract.after.dash}
1005 { duplicate$ empty$
1006   { pop$ "" }
1007   { 's :=
1008     #1 'charptr :=
1009     s text.length$ #1 + 'len :=
1010     { charptr len <
1011       s charptr #1 substring$ "-" = not
1012       and
1013     }
1014     { charptr #1 + 'charptr := }
1015     while$
1016     { charptr len <
1017       s charptr #1 substring$ "-" =
1018       and
1019     }
1020     { charptr #1 + 'charptr := }
1021     while$
1022     s charptr global.max$ substring$

```

```

1023   }
1024   if$
1025 }
1026
1027 FUNCTION {contains.dash}
1028 { duplicate$ empty$
1029   { pop$ #0 }
1030   { 's :=
1031     { s empty$ not
1032       s #1 #1 substring$ "-" = not
1033       and
1034     }
1035     { s #2 global.max$ substring$ 's := }
1036   while$
1037   s empty$ not
1038 }
1039 if$
1040 }
1041

```

著者-出版年制必须提取出年份

```

1042 FUNCTION {format.year}
1043 { year empty$ not
1044   { year extract.before.dash }
1045   { date empty$ not
1046     { date extract.before.dash }
1047     { "empty year in " cite$ * warning$
1048       ""
1049     }
1050   }
1051   if$
1052 }
1053 extra.label *
1054 }
1055

```

专利和报纸都是使用日期而不是年

```

1056 FUNCTION {format.date}
1057 { type$ "patent" = type$ "newspaper" = or
1058   date empty$ not and
1059   { date }
1060   { year }
1061 }
1062 }
1063

```

更新、修改日期只用于电子资源 electronic

```

1064 FUNCTION {format.editdate}
1065 { date empty$ not
1066   { "\allowbreak(" date * ")" * }
1067   { "" }
1068 }
1069 }
1070

```

国标中的“引用日期”都是与 URL 同时出现的，所以其实为 `urldate`，这个虽然不是 BibTeX 标准的域，但是实际中很常见。

```

1071 FUNCTION {format.urldate}
1072 { urldate empty$ not entry.is.electronic and
1073   { "\allowbreak[" urldate * "]" * }
1074   { "" }
1075 }
1076 }
1077

```

## B.4.8 Format pages

By default, BibTeX sets the global integer variable `global.max$` to the BibTeX constant `glob_str_size`, the maximum length of a global string variable. Analogously, BibTeX sets the global integer variable `entry.max$` to `ent_str_size`, the maximum length of an entry string variable. The style designer may change these if necessary (but this is unlikely)

The `n.dashify` function makes each single ``-'` in a string a double ``--'` if it's not already

```
pseudoVAR: pageresult: STRING          (it's what's accumulated on the stack)

n.dashify(s) ==
BEGIN
  t := s
  pageresult := ""
  while (not empty$(t))
  do
    if (first character of t = "-")
    then
      if (next character isn't)
      then
        pageresult := pageresult * "--"
        t := t with the "-" removed
      else
        while (first character of t = "-")
        do
          pageresult := pageresult * "--"
          t := t with the "-" removed
        od
      fi
    else
      pageresult := pageresult * the first character
      t := t with the first character removed
    fi
  od
  return pageresult
END
```

国标里页码范围的连接号使用 `hyphen`，需要将 `dash` 转为 `hyphen`。

```
1078 FUNCTION {hyphenate}
1079 { 't :=
1080   ""
1081   { t empty$ not }
1082   { t #1 #1 substring$ "-" =
1083     { "-" *
1084       { t #1 #1 substring$ "-" = }
1085       { t #2 global.max$ substring$ 't := }
1086       while$
1087     }
1088     { t #1 #1 substring$ *
1089       t #2 global.max$ substring$ 't :=
1090     }
1091   if$
1092 }
1093 while$
1094 }
1095
```

This function doesn't begin a sentence so "pages" isn't capitalized. Other functions that use this should keep that in mind.

```
1096 FUNCTION {format.pages}
1097 { pages empty$
```

```

1098 { "" }
1099 { pages hyphenate }
1100 if$
1101 }
1102

```

The `format.vol.num.pages` function is for the volume, number, and page range of a journal article. We use the format: `vol(number):pages`, with some variations for empty fields. This doesn't begin a sentence.

报纸在卷号缺失时，期号与前面的日期直接相连，所以必须拆开输出。

```

1103 FUNCTION {format.journal.number}
1104 { number empty$ not
1105   { "\penalty0 (" number * ")" * }
1106   { "" }
1107   if$
1108 }
1109
1110 FUNCTION {format.journal.pages}
1111 { pages empty$
1112   { "" }
1113   { ":\penalty0 " pages hyphenate * }
1114   if$
1115 }
1116

```

连续出版物的年卷期有起止范围，需要特殊处理

```

1117 FUNCTION {format.periodical.year.volume.number}
1118 { year empty$ not
1119   { year extract.before.dash }
1120   { "empty year in periodical " cite$ * warning$ }
1121   if$
1122   volume empty$ not
1123     { ", " * volume extract.before.dash * }
1124     'skip$
1125   if$
1126   number empty$ not
1127     { "\penalty0 (" * number extract.before.dash * ")" * }
1128     'skip$
1129   if$
1130   year contains.dash
1131     { "--" *
1132       year extract.after.dash empty$
1133       volume extract.after.dash empty$ and
1134       number extract.after.dash empty$ and not
1135       { year extract.after.dash empty$ not
1136         { year extract.after.dash * }
1137         { year extract.before.dash * }
1138         if$
1139         volume empty$ not
1140           { ", " * volume extract.after.dash * }
1141           'skip$
1142         if$
1143         number empty$ not
1144           { "\penalty0 (" * number extract.after.dash * ")" * }
1145           'skip$
1146         if$
1147       }
1148     'skip$
1149   if$
1150 }
1151 'skip$
1152 if$
1153 }

```

### B.4.9 Format url and doi

传统的 BibTeX 习惯使用 howpublished 著录 url，这里提供支持。

```

1155 FUNCTION {check.url}
1156 { url empty$ not
1157   { "\url{" url * "}" * 'entry.url :=
1158     #1 'entry.is.electronic :=
1159   }
1160   { howpublished empty$ not
1161     { howpublished #1 #5 substring$ "\url{" =
1162       { howpublished 'entry.url :=
1163         #1 'entry.is.electronic :=
1164       }
1165       'skip$
1166     if$
1167   }
1168   { note empty$ not
1169     { note #1 #5 substring$ "\url{" =
1170       { note 'entry.url :=
1171         #1 'entry.is.electronic :=
1172       }
1173       'skip$
1174     if$
1175   }
1176   'skip$
1177 if$
1178 }
1179 if$
1180 }
1181 if$
1182 }
1183
1184 FUNCTION {format.url}
1185 { entry.url empty$ not
1186   { new.block entry.url }
1187   { "" }
1188 if$
1189 }
1190

```

需要检测 DOI 是否已经包含在 URL 中。

```

1191 FUNCTION {check.doi}
1192 { doi empty$ not
1193   { #1 'entry.is.electronic := }
1194   'skip$
1195 if$
1196 }
1197
1198 FUNCTION {is.in.url}
1199 { 's :=
1200   s empty$
1201   { #1 }
1202   { entry.url empty$
1203     { #0 }
1204     { s text.length$ 'len :=
1205       entry.url text.length$ 'charptr :=
1206         { entry.url charptr len substring$ s = not
1207           charptr #0 >
1208           and
1209         }
1210       { charptr #1 - 'charptr := }

```

```

1211     while$
1212     charptr
1213     }
1214     if$
1215     }
1216     if$
1217 }
1218
1219 FUNCTION {format.doi}
1220 { ""
1221   doi empty$ not show.doi and
1222   { "" 's :=
1223     doi 't :=
1224     #0 'numnames :=
1225     { t empty$ not}
1226     { t #1 #1 substring$ 'tmp.str :=
1227       tmp.str "," = tmp.str " " = or t #2 #1 substring$ empty$ or
1228       { t #2 #1 substring$ empty$
1229         { s tmp.str * 's := }
1230         'skip$
1231         if$
1232         s empty$ s is.in.url or
1233         'skip$
1234         { numnames #1 + 'numnames :=
1235           numnames #1 >
1236           { ", " * }
1237           { "DOI: " * }
1238           if$
1239           "\doi{" s * "}" * *
1240         }
1241         if$
1242         "" 's :=
1243       }
1244       { s tmp.str * 's := }
1245       if$
1246       t #2 global.max$ substring$ 't :=
1247     }
1248     while$
1249     's :=
1250     s empty$ not
1251     { new.block s }
1252     { "" }
1253     if$
1254     }
1255     'skip$
1256     if$
1257 }
1258
1259 FUNCTION {check.electronic}
1260 { "" 'entry.url :=
1261   #0 'entry.is.electronic :=
1262   'check.doi
1263   'skip$
1264   if$
1265   'check.url
1266   'skip$
1267   if$
1268   medium empty$ not
1269   { medium "MT" = medium "DK" = or medium "CD" = or medium "OL" = or
1270     { #1 'entry.is.electronic := }
1271     'skip$
1272     if$
1273   }
1274   'skip$

```

```

1275 if$
1276 }
1277
1278 FUNCTION {format.note}
1279 { note empty$ not show.note and
1280   { note }
1281   { "" }
1282   if$
1283 }
1284

```

The function `empty.misc.check` complains if all six fields are empty, and if there's been no sorting or alphabetic-label complaint.

```

1285 FUNCTION {empty.misc.check}
1286 { author empty$ title empty$
1287   year empty$
1288   and and
1289   key empty$ not and
1290   { "all relevant fields are empty in " cite$ * warnings$ }
1291   'skip$
1292   if$
1293 }
1294

```

## B.5 Functions for all entry types

Now we define the type functions for all entry types that may appear in the .BIB file—e.g., functions like ‘article’ and ‘book’. These are the routines that actually generate the .BBL-file output for the entry. These must all precede the READ command. In addition, the style designer should have a function ‘default.type’ for unknown types. Note: The fields (within each list) are listed in order of appearance, except as described for an ‘inbook’ or a ‘proceedings’.

### B.5.1 专著

```

1295 FUNCTION {monograph}
1296 { output.bibitem
1297   author empty$ not
1298     { format.authors }
1299     { editor empty$ not
1300       { format.editors }
1301     }
1302     { bbl.anonymous }
1303   /authoryear)
1304 { *numerical)
1305   { "" }
1306 /numerical)
1307   if$
1308   }
1309   if$
1310   output
1311 { *authoryear)
1312   period.between.author.year
1313   'new.sentence
1314   'skip$
1315   if$
1316   format.year "year" output.check
1317 /authoryear)
1318   new.block
1319   format.series.vol.num.title "title" output.check
1320   "M" set.entry.mark
1321   format.mark "" output.after

```

```

1322 new.block
1323 format.translators output
1324 new.sentence
1325 format.edition output
1326 new.block
1327 format.address.publisher output
1328 (*numerical)
1329 format.year "year" output.check
1330 (/numerical)
1331 format.pages bbl.colon output.after
1332 format.urldate "" output.after
1333 format.url output
1334 format.doi output
1335 new.block
1336 format.note output
1337 fin.entry
1338 }
1339

```

## B.5.2 专著中的析出文献

An incollection is like inbook, but where there is a separate title for the referenced thing (and perhaps an editor for the whole). An incollection may CROSSREF a book.

Required: author, title, booktitle, publisher, year

Optional: editor, volume or number, series, type, chapter, pages, address, edition, month, note

```

1340 FUNCTION {incollection}
1341 { output.bibitem
1342 format.authors "author" output.check
1343 author format.key output
1344 (*authoryear)
1345 period.between.author.year
1346 'new.sentence
1347 'skip$
1348 if$
1349 format.year "year" output.check
1350 (/authoryear)
1351 new.block
1352 format.title "title" output.check
1353 "M" set.entry.mark
1354 format.mark "" output.after
1355 new.block
1356 format.translators output
1357 new.slash
1358 format.editors output
1359 new.block
1360 format.series.vol.num.booktitle "booktitle" output.check
1361 new.block
1362 format.edition output
1363 new.block
1364 format.address.publisher output
1365 (*numerical)
1366 format.year "year" output.check
1367 (/numerical)
1368 format.pages bbl.colon output.after
1369 format.urldate "" output.after
1370 format.url output
1371 format.doi output
1372 new.block
1373 format.note output
1374 fin.entry
1375 }
1376

```

### B.5.3 连续出版物

```
1377 FUNCTION {periodical}
1378 { output.bibitem
1379   format.authors "author" output.check
1380   author format.key output
1381   (*authoryear)
1382   period.between.author.year
1383     'new.sentence
1384     'skip$
1385   if$
1386   format.year "year" output.check
1387 }/authoryear)
1388 new.block
1389 format.title "title" output.check
1390 "J" set.entry.mark
1391 format.mark "" output.after
1392 new.block
1393 format.periodical.year.volume.number output
1394 new.block
1395 format.address.publisher output
1396 (*numerical)
1397   format.date "year" output.check
1398 }/numerical)
1399 format.urldate "" output.after
1400 format.url output
1401 format.doi output
1402 new.block
1403 format.note output
1404 fin.entry
1405 }
1406
```

### B.5.4 连续出版物中的析出文献

The article function is for an article in a journal. An article may CROSSREF another article.

Required fields: author, title, journal, year

Optional fields: volume, number, pages, month, note

The other entry functions are all quite similar, so no "comment version" will be given for them.

```
1407 FUNCTION {article}
1408 { output.bibitem
1409   format.authors "author" output.check
1410   author format.key output
1411   (*authoryear)
1412   period.between.author.year
1413     'new.sentence
1414     'skip$
1415   if$
1416   format.year "year" output.check
1417 }/authoryear)
1418 new.block
1419 format.title "title" output.check
1420 "J" set.entry.mark
1421 format.mark "" output.after
1422 new.block
1423 format.journal "journal" output.check
1424 (*numerical)
1425   format.date "year" output.check
1426 }/numerical)
1427   volume output
1428   format.journal.number "" output.after
1429   format.journal.pages "" output.after
1430   format.urldate "" output.after
```

```

1431 format.url output
1432 format.doi output
1433 new.block
1434 format.note output
1435 fin.entry
1436 }
1437

```

### B.5.5 专利文献

number 域也可以用来表示专利号。

```

1438 FUNCTION {patent}
1439 { output.bibitem
1440   format.authors output
1441   author format.key output
1442   (*authoryear)
1443   period.between.author.year
1444   'new.sentence
1445   'skip$
1446   if$
1447   format.year "year" output.check
1448   (/authoryear)
1449   new.block
1450   format.title
1451   number empty$ not
1452     { bbl.colon * number * }
1453     'skip$
1454   if$
1455   "title" output.check
1456   "P" set.entry.mark
1457   format.mark "" output.after
1458   new.block
1459   format.date "year" output.check
1460   format.urldate "" output.after
1461   format.url output
1462   format.doi output
1463   new.block
1464   format.note output
1465   fin.entry
1466 }
1467

```

### B.5.6 电子资源

```

1468 FUNCTION {electronic}
1469 { #1 #1 check.electronic
1470   #1 'entry.is.electronic :=
1471   output.bibitem
1472   format.authors output
1473   author format.key output
1474   (*authoryear)
1475   period.between.author.year
1476   'new.sentence
1477   'skip$
1478   if$
1479   format.year "year" output.check
1480   (/authoryear)
1481   new.block
1482   format.series.vol.num.title "title" output.check
1483   "EB" set.entry.mark
1484   format.mark "" output.after
1485   new.block
1486   format.address.publisher output

```

```

1487 <*numerical>
1488   date empty$
1489   { format.date output }
1490   'skip$
1491   if$
1492 </numerical>
1493   format.pages bbl.colon output.after
1494   format.editdate "" output.after
1495   format.urldate "" output.after
1496   format.url output
1497   format.doi output
1498   new.block
1499   format.note output
1500   fin.entry
1501 }
1502

```

### B.5.7 其他文献类型

A misc is something that doesn't fit elsewhere.

Required: at least one of the 'optional' fields

Optional: author, title, howpublished, month, year, note

Misc 用来自动判断类型。

```

1503 FUNCTION {misc}
1504 { journal empty$ not
1505   'article
1506   { booktitle empty$ not
1507     'incollection
1508     { publisher empty$ not
1509       'monograph
1510       { entry.is.electronic
1511         'electronic
1512         { "Z" set.entry.mark
1513           monograph
1514         }
1515       }
1516     }
1517   }
1518   if$
1519 }
1520 }
1521 if$
1522 empty.misc.check
1523 }
1524
1525 FUNCTION {archive}
1526 { "A" set.entry.mark
1527   misc
1528 }
1529

```

The book function is for a whole book. A book may CROSSREF another book.

Required fields: author or editor, title, publisher, year

Optional fields: volume or number, series, address, edition, month, note

```

1530 FUNCTION {book} { monograph }
1531

```

A booklet is a bound thing without a publisher or sponsoring institution.

Required: title

Optional: author, howpublished, address, month, year, note

```

1532 FUNCTION {booklet} { book }

```

```

1533
1534 FUNCTION {collection}
1535 { "G" set.entry.mark
1536  monograph
1537 }
1538
1539 FUNCTION {database}
1540 { "DB" set.entry.mark
1541  electronic
1542 }
1543
1544 FUNCTION {dataset}
1545 { "DS" set.entry.mark
1546  electronic
1547 }
1548

```

An inbook is a piece of a book: either a chapter and/or a page range. It may CROSSREF a book. If there's no volume field, the type field will come before number and series.

Required: author or editor, title, chapter and/or pages, publisher, year

Optional: volume or number, series, type, address, edition, month, note

inbook 类是不含 booktitle 域的，所以不应该适用于“专著中的析出文献”，而应该是专著，即 book 类。

```

1549 FUNCTION {inbook} { book }
1550

```

An inproceedings is an article in a conference proceedings, and it may CROSSREF a proceedings. If there's no address field, the month (& year) will appear just before note.

Required: author, title, booktitle, year

Optional: editor, volume or number, series, pages, address, month, organization, publisher, note

```

1551 FUNCTION {inproceedings}
1552 { "C" set.entry.mark
1553  incollection
1554 }
1555

```

The conference function is included for Scribe compatibility.

```

1556 FUNCTION {conference} { inproceedings }
1557
1558 FUNCTION {map}
1559 { "CM" set.entry.mark
1560  misc
1561 }
1562

```

A manual is technical documentation.

Required: title

Optional: author, organization, address, edition, month, year, note

```

1563 FUNCTION {manual} { monograph }
1564

```

A mastersthesis is a Master's thesis.

Required: author, title, school, year

Optional: type, address, month, note

```

1565 FUNCTION {mastersthesis}
1566 { "D" set.entry.mark
1567  monograph
1568 }

```

1569  
 1570 FUNCTION {newspaper}  
 1571 { "N" set.entry.mark  
 1572 article  
 1573 }  
 1574  
 1575 FUNCTION {online}  
 1576 { "EB" set.entry.mark  
 1577 electronic  
 1578 }  
 1579

A phdthesis is like a mastersthesis.

Required: author, title, school, year

Optional: type, address, month, note

1580 FUNCTION {phdthesis} { mastersthesis }  
 1581

A proceedings is a conference proceedings. If there is an organization but no editor field, the organization will appear as the first optional field (we try to make the first block nonempty); if there's no address field, the month (& year) will appear just before note.

Required: title, year

Optional: editor, volume or number, series, address, month, organization, publisher, note

1582 FUNCTION {proceedings}  
 1583 { "C" set.entry.mark  
 1584 monograph  
 1585 }  
 1586  
 1587 FUNCTION {software}  
 1588 { "CP" set.entry.mark  
 1589 electronic  
 1590 }  
 1591  
 1592 FUNCTION {standard}  
 1593 { "S" set.entry.mark  
 1594 misc  
 1595 }  
 1596

A techreport is a technical report.

Required: author, title, institution, year

Optional: type, number, address, month, note

1597 FUNCTION {techreport}  
 1598 { "R" set.entry.mark  
 1599 misc  
 1600 }  
 1601

An unpublished is something that hasn't been published.

Required: author, title, note

Optional: month, year

1602 FUNCTION {unpublished}  
 1603 { "Z" set.entry.mark  
 1604 misc  
 1605 }  
 1606

We use entry type 'misc' for an unknown type; BibTeX gives a warning.

1607 FUNCTION {default.type} { misc }  
 1608

## B.6 Common macros

Here are macros for common things that may vary from style to style. Users are encouraged to use these macros.

Months are either written out in full or abbreviated

```
1609 MACRO {jan} {"January"}
1610
1611 MACRO {feb} {"February"}
1612
1613 MACRO {mar} {"March"}
1614
1615 MACRO {apr} {"April"}
1616
1617 MACRO {may} {"May"}
1618
1619 MACRO {jun} {"June"}
1620
1621 MACRO {jul} {"July"}
1622
1623 MACRO {aug} {"August"}
1624
1625 MACRO {sep} {"September"}
1626
1627 MACRO {oct} {"October"}
1628
1629 MACRO {nov} {"November"}
1630
1631 MACRO {dec} {"December"}
1632
```

Journals are either written out in full or abbreviated; the abbreviations are like those found in ACM publications.

To get a completely different set of abbreviations, it may be best to make a separate .bib file with nothing but those abbreviations; users could then include that file name as the first argument to the `\bibliography` command

```
1633 MACRO {acmcs} {"ACM Computing Surveys"}
1634
1635 MACRO {acta} {"Acta Informatica"}
1636
1637 MACRO {cacm} {"Communications of the ACM"}
1638
1639 MACRO {ibmjrd} {"IBM Journal of Research and Development"}
1640
1641 MACRO {ibmsj} {"IBM Systems Journal"}
1642
1643 MACRO {ieeese} {"IEEE Transactions on Software Engineering"}
1644
1645 MACRO {ieeetc} {"IEEE Transactions on Computers"}
1646
1647 MACRO {ieeetcad}
1648 {"IEEE Transactions on Computer-Aided Design of Integrated Circuits"}
1649
1650 MACRO {ipl} {"Information Processing Letters"}
1651
1652 MACRO {jacm} {"Journal of the ACM"}
1653
1654 MACRO {jcss} {"Journal of Computer and System Sciences"}
1655
1656 MACRO {scp} {"Science of Computer Programming"}
1657
1658 MACRO {sicomp} {"SIAM Journal on Computing"}
```

```

1659
1660 MACRO {tocs} {"ACM Transactions on Computer Systems"}
1661
1662 MACRO {tods} {"ACM Transactions on Database Systems"}
1663
1664 MACRO {tog} {"ACM Transactions on Graphics"}
1665
1666 MACRO {toms} {"ACM Transactions on Mathematical Software"}
1667
1668 MACRO {toois} {"ACM Transactions on Office Information Systems"}
1669
1670 MACRO {toplas} {"ACM Transactions on Programming Languages and Systems"}
1671
1672 MACRO {tcs} {"Theoretical Computer Science"}
1673

```

## B.7 Format labels

The `sortify` function converts to lower case after `purify`; it's used in sorting and in computing alphabetic labels after sorting

The `chop.word(w,len,s)` function returns either `s` or, if the first `len` letters of `s` equals `w` (this comparison is done in the third line of the function's definition), it returns that part of `s` after `w`.

```

1674 FUNCTION {sortify}
1675 { purify$
1676   "l" change.case$
1677 }
1678

```

We need the `chop.word` stuff for the dubious `unsorted-list-with-labels` case.

```

1679 FUNCTION {chop.word}
1680 { 's :=
1681   'len :=
1682   s #1 len substring$ =
1683     { s len #1 + global.max$ substring$ }
1684   's
1685   if$
1686 }
1687

```

The `format.lab.names` function makes a short label by using the initials of the von and Last parts of the names (but if there are more than four names, (i.e., people) it truncates after three and adds a superscripted "+"; it also adds such a "+" if the last of multiple authors is "others"). If there is only one name, and its von and Last parts combined have just a single name-token ("Knuth" has a single token, "Brinch Hansen" has two), we take the first three letters of the last name. The boolean `et.al.char.used` tells whether we've used a superscripted "+", so that we know whether to include a LaTeX macro for it.

```

format.lab.names(s) ==
BEGIN
  numnames := num.names$(s)
  if numnames > 1 then
    if numnames > 4 then
      namesleft := 3
    else
      namesleft := numnames
  nameptr := 1
  namesresult := ""
  while namesleft > 0
  do
    if (name_ptr = numnames) and
      format.name$(s, nameptr, "{ff_}{vv_}{ll}{_jj}") = "others"

```

```

        then namerresult := namerresult * "{\etalchar{+}}"
            et.al.char.used := true
        else namerresult := namerresult *
            format.name$(s, nameptr, "{v{}}{l{}}")
        nameptr := nameptr + 1
        namesleft := namesleft - 1
    od
    if numnames > 4 then
        namerresult := namerresult * "{\etalchar{+}}"
        et.al.char.used := true
    else
        t := format.name$(s, 1, "{v{}}{l{}}")
        if text.length$(t) < 2 then % there's just one name-token
            namerresult := text.prefix$(format.name$(s,1,"{ll}"),3)
        else
            namerresult := t
        fi
    fi
    return namerresult
END

```

Exactly what fields we look at in constructing the primary part of the label depends on the entry type; this selectivity (as opposed to, say, always looking at author, then editor, then key) helps ensure that "ignored" fields, as described in the LaTeX book, really are ignored. Note that MISC is part of the deepest 'else' clause in the nested part of calc.label; thus, any unrecognized entry type in the database is handled correctly.

There is one auxiliary function for each of the four different sequences of fields we use. The first of these functions looks at the author field, and then, if necessary, the key field. The other three functions, which might look at two fields and the key field, are similar, except that the key field takes precedence over the organization field (for labels—not for sorting).

The calc.label function calculates the preliminary label of an entry, which is formed by taking three letters of information from the author or editor or key or organization field (depending on the entry type and on what's empty, but ignoring a leading "The " in the organization), and appending the last two characters (digits) of the year. It is an error if the appropriate fields among author, editor, organization, and key are missing, and we use the first three letters of the cite\$ in desperation when this happens. The resulting label has the year part, but not the name part, purify\$(purify\$(year)) (purify\$(year) allows some sorting shenanigans by the user).

This function also calculates the version of the label to be used in sorting.

The final label may need a trailing 'a', 'b', etc., to distinguish it from otherwise identical labels, but we can't calculate those "extra.label"s until after sorting.

```

calc.label ==
BEGIN
    if type$ = "book" or "inbook" then
        author.editor.key.label
    else if type$ = "proceedings" then
        editor.key.organization.label
    else if type$ = "manual" then
        author.key.organization.label
    else
        author.key.label
    fi fi fi
    label := label * substring$(purify$(field.or.null(year)), -1, 2)
        % assuming we will also sort, we calculate a sort.label
    sort.label := sortify(label), but use the last four, not two, digits

```

END

```
1688 FUNCTION {format.lab.names}
1689 { 's :=
1690 s #1 "{vv~}{ll}{, jj}{, ff}" format.name$ 't :=
1691 t get.str.lang 'name.lang :=
1692 name.lang lang.en =
1693   { t #1 "{vv~}{ll}" format.name$}
1694   { t #1 "{ll}{ff}" format.name$}
1695 if$
1696 s num.names$ #1 >
1697   { bbl.space * bbl.et.al * }
1698   'skip$
1699 if$
1700 }
1701
1702 FUNCTION {author.key.label}
1703 { author empty$
1704   { key empty$
1705     { cite$ #1 #3 substring$ }
1706     'key
1707     if$
1708   }
1709   { author format.lab.names }
1710 if$
1711 }
1712
1713 FUNCTION {author.editor.key.label}
1714 { author empty$
1715   { editor empty$
1716     { key empty$
1717       { cite$ #1 #3 substring$ }
1718       'key
1719       if$
1720     }
1721     { editor format.lab.names }
1722   if$
1723   }
1724   { author format.lab.names }
1725 if$
1726 }
1727
1728 FUNCTION {author.key.organization.label}
1729 { author empty$
1730   { key empty$
1731     { organization empty$
1732       { cite$ #1 #3 substring$ }
1733       { "The " #4 organization chop.word #3 text.prefix$ }
1734     if$
1735   }
1736   'key
1737   if$
1738 }
1739 { author format.lab.names }
1740 if$
1741 }
1742
1743 FUNCTION {editor.key.organization.label}
1744 { editor empty$
1745   { key empty$
1746     { organization empty$
1747       { cite$ #1 #3 substring$ }
1748       { "The " #4 organization chop.word #3 text.prefix$ }
1749     if$
```

```

1750     }
1751     'key
1752     if$
1753   }
1754   { editor format.lab.names }
1755   if$
1756 }
1757
1758 FUNCTION {calc.short.authors}
1759 { type$ "book" =
1760   type$ "inbook" =
1761   or
1762   'author.editor.key.label
1763   { type$ "collection" =
1764     type$ "proceedings" =
1765     or
1766     { editor empty$ not
1767       'editor.key.organization.label
1768       'author.key.organization.label
1769       if$
1770     }
1771     'author.key.label
1772   }
1773   if$
1774   'short.list :=
1775 }
1776 }
1777
1778 FUNCTION {calc.label}
1779 { calc.short.authors
1780   short.list
1781   "("
1782   *
1783   format.year duplicate$ empty$
1784   short.list key field.or.null = or
1785   { pop$ "" }
1786   'skip$
1787   if$
1788   *
1789   'label :=
1790 }
1791

```

## B.8 Sorting

When sorting, we compute the sortkey by executing "presort" on each entry. The presort key contains a number of "sortify"ed strings, concatenated with multiple blanks between them. This makes things like "brinch per" come before "brinch hansen per".

The fields used here are: the sort.label for alphabetic labels (as set by calc.label), followed by the author names (or editor names or organization (with a leading "The " removed) or key field, depending on entry type and on what's empty), followed by year, followed by the first bit of the title (chopping off a leading "The ", "A ", or "An "). Names are formatted: Von Last First Junior. The names within a part will be separated by a single blank (such as "brinch hansen"), two will separate the name parts themselves (except the von and last), three will separate the names, four will separate the names from year (and from label, if alphabetic), and four will separate year from title.

The sort.format.names function takes an argument that should be in BibTeX name format, and returns a string containing " "-separated names in the format described above. The function is almost

the same as `format.names`.

```
1792 (*authoryear)
1793 FUNCTION {sort.language.label}
1794 { entry.lang lang.zh =
1795   { "a zh " }
1796   { entry.lang lang.ja =
1797     { "b ja " }
1798     { entry.lang lang.en =
1799       { "c en " }
1800       { entry.lang lang.ru =
1801         { "d ru " }
1802         { "e other " }
1803         if$
1804       }
1805     }
1806   }
1807   if$
1808 }
1809 if$
1810 }
1811
1812 FUNCTION {sort.format.names}
1813 { 's :=
1814   #1 'nameptr :=
1815   ""
1816   s num.names$ 'numnames :=
1817   numnames 'namesleft :=
1818   { namesleft #0 > }
1819   {
1820     s nameptr "{vv{ }}{ll{ }}{ ff{ }}{ jj{ }}" format.name$ 't :=
1821     nameptr #1 >
1822     {
1823       " " *
1824       namesleft #1 = t "others" = and
1825       { "zzzzz" * }
1826       { numnames #2 > nameptr #2 = and
1827         { "zz" * year field.or.null * " " * }
1828         'skip$
1829       }
1830       if$
1831       t sortify *
1832     }
1833     if$
1834     { t sortify * }
1835     if$
1836     nameptr #1 + 'nameptr :=
1837     namesleft #1 - 'namesleft :=
1838   }
1839   while$
1840 }
1841
```

The `sort.format.title` function returns the argument, but first any leading "A "'s, "An "'s, or "The "'s are removed. The `chop.word` function uses `s`, so we need another string variable, `t`

```
1842 FUNCTION {sort.format.title}
1843 { 't :=
1844   "A " #2
1845   "An " #3
1846   "The " #4 t chop.word
1847   chop.word
1848   chop.word
1849   sortify
1850   #1 global.max$ substring$
1851 }
```

1852

The auxiliary functions here, for the presort function, are analogous to the ones for calc.label; the same comments apply, except that the organization field takes precedence here over the key field. For sorting purposes, we still remove a leading "The " from the organization field.

```
1853 FUNCTION {anonymous.sort}
1854 { lang.zh entry.lang =
1855   { "yi4 ming2" }
1856   { "anon" }
1857   if$
1858 }
1859
1860 FUNCTION {author.sort}
1861 { key empty$
1862   { entry.lang lang.zh =
1863     { "empty key in " cite$ * warning$ }
1864     'skip$
1865     if$
1866     author empty$
1867     { anonymous.sort }
1868     { author sort.format.names }
1869     if$
1870   }
1871   { key sortify }
1872   if$
1873 }
1874
1875 FUNCTION {author.editor.sort}
1876 { key empty$
1877   { author empty$
1878     { editor empty$
1879       { anonymous.sort }
1880       { editor sort.format.names }
1881     }
1882     { author sort.format.names }
1883     if$
1884   }
1885   { key sortify }
1886   if$
1887 }
1888
1889
1890 FUNCTION {author.organization.sort}
1891 { key empty$
1892   { author empty$
1893     { organization empty$
1894       { anonymous.sort }
1895       { "The " #4 organization chop.word sortify }
1896     }
1897     { author sort.format.names }
1898     if$
1899   }
1900   { key sortify }
1901   if$
1902 }
1903
1904
1905 FUNCTION {editor.organization.sort}
1906 { key empty$
1907   { editor empty$
1908     { organization empty$
1909       { anonymous.sort }
1910       { "The " #4 organization chop.word sortify }

```

```

1911         if$
1912         }
1913         { editor sort.format.names }
1914     if$
1915     }
1916     { key sortify }
1917 if$
1918 }
1919
1920 </authoryear>

```

顺序编码制的排序要简单得多

```

1921 <*numerical>
1922 INTEGERS { seq.num }
1923
1924 FUNCTION {init.seq}
1925 { #0 'seq.num :=}
1926
1927 FUNCTION {int.to.fix}
1928 { "00000000" swap$ int.to.str$ *
1929  #-1 #10 substring$
1930 }
1931
1932 </numerical>

```

There is a limit, `entry.max$`, on the length of an entry string variable (which is what its `sort.key$` is), so we take at most that many characters of the constructed key, and hope there aren't many references that match to that many characters!

```

1933 FUNCTION {presort}
1934 { set.entry.lang
1935   show.url show.doi check.electronic
1936   calc.label
1937   label sortify
1938   " "
1939   *
1940 <*authoryear>
1941   sort.language.label
1942   type$ "book" =
1943   type$ "inbook" =
1944   or
1945     'author.editor.sort
1946     { type$ "collection" =
1947       type$ "proceedings" =
1948       or
1949         'editor.organization.sort
1950         'author.sort
1951     } if$
1952   }
1953 if$
1954 *
1955 " "
1956 *
1957 year field.or.null sortify
1958 *
1959 " "
1960 *
1961 cite$
1962 *
1963 #1 entry.max$ substring$
1964 </authoryear>
1965 <*numerical>
1966 seq.num #1 + 'seq.num :=
1967 seq.num int.to.fix

```

```

1968 (/numerical)
1969 'sort.label :=
1970 sort.label *
1971 #1 entry.max$ substring$
1972 'sort.key$ :=
1973 }
1974

```

Now comes the final computation for alphabetic labels, putting in the 'a's and 'b's and so forth if required. This involves two passes: a forward pass to put in the 'b's, 'c's and so on, and a backwards pass to put in the 'a's (we don't want to put in 'a's unless we know there are 'b's). We have to keep track of the longest (in width\$ terms) label, for use by the "thebibliography" environment.

```

VAR: longest.label, last.sort.label, next.extra: string
     longest.label.width, last.extra.num: integer

initialize.longest.label ==
BEGIN
  longest.label := ""
  last.sort.label := int.to.chr$(0)
  next.extra := ""
  longest.label.width := 0
  last.extra.num := 0
END

forward.pass ==
BEGIN
  if last.sort.label = sort.label then
    last.extra.num := last.extra.num + 1
    extra.label := int.to.chr$(last.extra.num)
  else
    last.extra.num := chr.to.int$("a")
    extra.label := ""
    last.sort.label := sort.label
  fi
END

reverse.pass ==
BEGIN
  if next.extra = "b" then
    extra.label := "a"
  fi
  label := label * extra.label
  if width$(label) > longest.label.width then
    longest.label := label
    longest.label.width := width$(label)
  fi
  next.extra := extra.label
END

```

```

1975 STRINGS { longest.label last.label next.extra }
1976
1977 INTEGERS { longest.label.width last.extra.num number.label }
1978
1979 FUNCTION {initialize.longest.label}
1980 { "" 'longest.label :=
1981 #0 int.to.chr$ 'last.label :=
1982 "" 'next.extra :=
1983 #0 'longest.label.width :=
1984 #0 'last.extra.num :=
1985 #0 'number.label :=
1986 }
1987

```

```

1988 FUNCTION {forward.pass}
1989 { last.label label =
1990   { last.extra.num #1 + 'last.extra.num :=
1991     last.extra.num int.to.chr$ 'extra.label :=
1992   }
1993   { "a" chr.to.int$ 'last.extra.num :=
1994     "" 'extra.label :=
1995     label 'last.label :=
1996   }
1997   if$
1998   number.label #1 + 'number.label :=
1999 }
2000
2001 FUNCTION {reverse.pass}
2002 { next.extra "b" =
2003   { "a" 'extra.label := }
2004   'skip$
2005   if$
2006   extra.label 'next.extra :=
2007   extra.label
2008   duplicate$ empty$
2009   'skip$
2010   { "{\natexlab{" swap$ * "}" * }
2011   if$
2012   'extra.label :=
2013   label extra.label * 'label :=
2014 }
2015
2016 FUNCTION {bib.sort.order}
2017 { sort.label 'sort.key$ :=
2018 }
2019

```

## B.9 Write bbl file

Now we're ready to start writing the .BBL file. We begin, if necessary, with a  $\LaTeX$  macro for unnamed names in an alphabetic label; next comes stuff from the 'preamble' command in the database files. Then we give an incantation containing the command `\begin{thebibliography}{...}` where the '...' is the longest label.

We also call `init.state.consts`, for use by the output routines.

```

2020 FUNCTION {begin.bib}
2021 { preamble$ empty$
2022   'skip$
2023   { preamble$ write$ newline$ }
2024   if$
2025   "\begin{thebibliography}{ number.label int.to.str$ * "}" *
2026   write$ newline$
2027   "\providecommand{\natexlab}[1]{#1}"
2028   write$ newline$
2029   show.url show.doi or
2030   { "\providecommand{\url}[1]{#1}"
2031     write$ newline$
2032     "\expandafter\ifx\curname urlstyle\endcurname\relax\relax\else"
2033     write$ newline$
2034     " \urlstyle{same}\fi"
2035     write$ newline$
2036   }
2037   'skip$
2038   if$
2039   show.doi
2040   { "\providecommand{\href}[2]{\url{#2}}"

```

```

2041     write$ newline$
2042     "\providecommand{\doi}[1]{\href{https://doi.org/#1}{#1}}"
2043     write$ newline$
2044   }
2045   'skip$
2046   if$
2047 }
2048

```

Finally, we finish up by writing the `\end{thebibliography}` command.

```

2049 FUNCTION {end.bib}
2050 { newline$
2051   "\end{thebibliography}" write$ newline$
2052 }
2053

```

## B.10 Main execution

Now we read in the .BIB entries.

```

2054 READ
2055
2056 EXECUTE {init.state.consts}
2057
2058 EXECUTE {load.config}
2059
2060 (*numerical)
2061 EXECUTE {init.seq}
2062
2063 (/numerical)
2064 ITERATE {presort}
2065

```

And now we can sort

```

2066 SORT
2067
2068 EXECUTE {initialize.longest.label}
2069
2070 ITERATE {forward.pass}
2071
2072 REVERSE {reverse.pass}
2073
2074 ITERATE {bib.sort.order}
2075
2076 SORT
2077
2078 EXECUTE {begin.bib}
2079

```

Now we produce the output for all the entries

```

2080 ITERATE {call.type$}
2081
2082 EXECUTE {end.bib}
2083 (/authoryear | numerical)

```