

GB/T 7714-2015 BibT_EX style

Zeping Lee*

2019/11/20 v1.1.2

摘要

The gbt7714 package provides a BibT_EX 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 L^AT_EX 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》^[1]（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 BibT_EX^[2] 实现，具有以下特性：

- 兼容 natbib 宏包^[3]
- 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/CTeX-org/gbt7714-bibtex-style>。

2 使用方法

super
numbers
authoryear 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)，其中顺序编码制根据引用标注样式不同分为角标数字式 (super) 和与正文平排的数字式 (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^[3]。

\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 BibT_EX 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 BibT_EX 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 BibT_EX 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 BibT_EX 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},  
  ...
```

6 自定义样式

BibTeX 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。

表 2: 参考文献表样式的配置参数

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

若用户需要定制更多内容，可以学习 `bst` 文件的语法并修改^[4-6]，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^[7] 发布了基于 GB/T 7714-2005 的 BibTeX 样式，支持顺序编码制和著者出版年制两种风格。李志奇^[8] 发布了严格遵循 GB/T 7714-2005 的 BibLaTeX 的样式。胡海星^[9] 提供了另一个 BibTeX 实现，还给每行 `bst` 代码写了 java 语言注释。沈周^[10] 基于 `biblatex-casperi`^[11] 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^[12]，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. BibTeXing[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 BibTeX 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/ttb_en.pdf.
- [6] MITTELBACH F, GOOSSENS M, BRAAMS J, et al. The L^AT_EX 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>.

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}{\@gbt@mmxvtrue}
6 \DeclareOption{2005}{\@gbt@mmxvfalse}
7 \DeclareOption{super}{\@gbt@numericaltrue\@gbt@supertrue}
8 \DeclareOption{numbers}{\@gbt@numericaltrue\@gbt@superfalse}
9 \DeclareOption{authoryear}{\@gbt@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{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`。

```
18 \newcommand\bibstyle@super{\bibpunct{[ ]}{,}{s}{,}{\textsuperscript{,}}{,}}
19 \newcommand\bibstyle@numbers{\bibpunct{[ ]}{,}{n}{,}{,}{,}}
20 \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`。

```
21 \newcommand\gbtbibstyle[1]{%
22   \@ifundefined{gbt@bib@#1}{%
23     \PackageError{gbt7714}{Invalid argument #1}{}%
24   }{%
25     \@nameuse{gbt@bib@#1}
26   }%
27 }
28 \newcommand\gbt@bib@numerical{%
29   \if@gbt@mmxv
30     \bibliographystyle{gbt7714-unsrt}%
31   \else
32     \bibliographystyle{gbt7714-2005-unsrt}%
33   \fi
34 }
35 \newcommand\gbt@bib@authoryear{%
36   \if@gbt@mmxv
37     \bibliographystyle{gbt7714-plain}%
38   \else
39     \bibliographystyle{gbt7714-2005-plain}%
40   \fi
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，将页码写在上标位置。为了减少依赖的宏包，这里直接重定义命令不使用 \patchcmd。

Numerical 模式的 \citet 的页码：

```
54 \def\NAT@citexnum[#1][#2]#3{%
55   \NAT@reset@parser
56   \NAT@sort@cites{#3}%
57   \NAT@reset@citea
58   \@cite{\def\NAT@num{-1}\let\NAT@last@yr\relax\let\NAT@nm\@empty
59     \@for\@citeb:=\NAT@cite@list\do
60       {\@safe@activestrue
61         \edef\@citeb{\expandafter\@firstofone\@citeb\@empty}%
62         \@safe@activesfalse
63         \@ifundefined{b@\@citeb\@extra@b@citeb}{%
64           {\reset@font\bfseries?}
65           \NAT@citeundefined\PackageWarning{natbib}%
66             {Citation ``\@citeb' on page \thepage \space undefined}}%
67         {\let\NAT@last@num\NAT@num\let\NAT@last@nm\NAT@nm
68           \NAT@parse{\@citeb}%
69           \ifNAT@longnames\@ifundefined{bv@\@citeb\@extra@b@citeb}{%
70             \let\NAT@name=\NAT@all@names
71             \global\@namedef{bv@\@citeb\@extra@b@citeb}{}}{}%
72           \fi
73           \ifNAT@full\let\NAT@nm\NAT@all@names\else
74             \let\NAT@nm\NAT@name\fi
75           \ifNAT@swa
76             \@ifnum{\NAT@ctype>\@ne}{%
77               \@citea
78               \NAT@hyper@{\@ifnum{\NAT@ctype=\tw@}{\NAT@test{\NAT@ctype}}{\NAT@alias}}%
79             }{%
80               \@ifnum{\NAT@cmprs>\z@}{%
81                 \NAT@ifcat@num\NAT@num
82                 {\let\NAT@nm=\NAT@num}%
83                 {\def\NAT@nm{-2}}%
84                 \NAT@ifcat@num\NAT@last@num
85                 {\@tempcnta=\NAT@last@num\relax}%
```

```

86      {\@tempcnta\m@ne}%
87      \@ifnum{\NAT@nm=\@tempcnta}{%
88      \@ifnum{\NAT@merge>\@ne}{\NAT@last@yr@mbox}%
89      }{%
90      \advance\@tempcnta by\@ne
91      \@ifnum{\NAT@nm=\@tempcnta}{%

```

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。

```

92      % \ifx\NAT@last@yr\relax
93      %   \def@NAT@last@yr{\@citea}%
94      % \else
95      %   \def@NAT@last@yr{--\NAT@penalty}%
96      % \fi
97      \def@NAT@last@yr{-\NAT@penalty}%
98      }{%
99      \NAT@last@yr@mbox
100     }%
101     }%
102     }{%
103     \@tempwattrue
104     \@ifnum{\NAT@merge>\@ne}{\@ifnum{\NAT@last@num=\NAT@num\relax}{\@tempwafalse}{}}{%
105     \if@tempswa\NAT@citea@mbox\fi
106     }%
107     }%
108     \NAT@def@citea
109     \else
110     \ifcase\NAT@ctype
111     \ifx\NAT@last@nm\NAT@nm \NAT@yrsep\NAT@penalty\NAT@space\else
112     \@citea \NAT@test{\@ne}\NAT@spacechar\NAT@mbox{\NAT@super@kern\NAT@@@open}%
113     \fi
114     \if*#1*\else#1\NAT@spacechar\fi
115     \NAT@mbox{\NAT@hyper@{\citenumfont{\NAT@num}}}%
116     \NAT@def@citea@box
117     \or
118     \NAT@hyper@citea@space{\NAT@test{\NAT@ctype}}%
119     \or
120     \NAT@hyper@citea@space{\NAT@test{\NAT@ctype}}%
121     \or
122     \NAT@hyper@citea@space\NAT@alias
123     \fi
124     \fi
125     }%
126     }%
127     \@ifnum{\NAT@cmprs>\z@}{\NAT@last@yr}{}%
128     \ifNAT@swa\else

```

将页码放在括号外边，并且置于上标。

```

129     % \@ifnum{\NAT@ctype=\z@}{%
130     %   \if*#2*\else\NAT@cmt#2\fi

```



```

131      % }{}%
132      \NAT@mbx{\NAT@close}%
133      \@ifnum{\NAT@ctype=\z@}{%
134          \if*#2*\else\textsuperscript{#2}\fi
135      }{}%
136      \fi
137      }{#1}{#2}%
138 }%

```

Numerical 模式的 \citep 的页码:

```

139 \renewcommand\NAT@citesuper[3]{\ifNAT@swa
140     \if*#2*\else#2\NAT@spacechar\fi
141     \unskip\kern\p@\textsuperscript{\NAT@open#1\NAT@close\if*#3*\else#3\fi}%
142     \else #1\fi\endgroup}

```

Author-year 模式的 \citel 的页码:

```

143 \def\NAT@citex%
144     [#1][#2]#3{%
145     \NAT@reset@parser
146     \NAT@sort@cites{#3}%
147     \NAT@reset@citea
148     \@cite{\let\NAT@nm\@empty\let\NAT@year\@empty
149         \@for\@citeb:=\NAT@cite@list\do
150             {\@safe@activestrue
151                 \edef\@citeb{\expandafter\@firstofone\@citeb\@empty}%
152                 \@safe@activesfalse
153                 \@ifundefined{b@\@citeb\@extra@b@citeb}{\@citea%
154                     {\reset@font\bfseries ?}\NAT@citeundefined
155                     \PackageWarning{natbib}%
156                     {Citation '\@citeb' on page \thepage \space undefined}}{\def\NAT@date{}}%
157                 {\let\NAT@lastnm=\NAT@nm\let\NAT@lastyr=\NAT@year
158                     \NAT@parse{\@citeb}%
159                     \ifNAT@longnames\@ifundefined{bv@\@citeb\@extra@b@citeb}{%
160                         \let\NAT@name=\NAT@all@names
161                         \global\@namedef{bv@\@citeb\@extra@b@citeb}{}}{}%
162                     \fi
163                     \ifNAT@full\let\NAT@nm\NAT@all@names\else
164                         \let\NAT@nm\NAT@name\fi
165                     \ifNAT@swa\ifcase\NAT@ctype
166                         \if\relax\NAT@date\relax
167                             \@citea\NAT@hyper@{\NAT@nmfmt{\NAT@nm}\NAT@date}%
168                         \else
169                             \ifx\NAT@lastnm\NAT@nm\NAT@yrsep
170                                 \ifx\NAT@lastyr\NAT@year
171                                     \def\NAT@temp{{?}}%
172                                     \ifx\NAT@temp\NAT@exlab\PackageWarningNoLine{natbib}%
173                                         {Multiple citation on page \thepage: same authors and
174                                         year\MessageBreak without distinguishing extra
175                                         letter,\MessageBreak appears as question mark}\fi
176                                     \NAT@hyper@{\NAT@exlab}%
177                                 \else\unskip\NAT@spacechar

```

```

178         \NAT@hyper@{\NAT@date}%
179     \fi
180 \else
181     \@citea\NAT@hyper@{%
182         \NAT@nmfmt{\NAT@nm}%
183         \hyper@natlinkbreak{%
184             \NAT@aysep\NAT@spacechar}{\@citeb\@extra@b@citeb
185         }%
186         \NAT@date
187     }%
188 \fi
189 \fi
190 \or\@citea\NAT@hyper@{\NAT@nmfmt{\NAT@nm}}%
191 \or\@citea\NAT@hyper@{\NAT@date}%
192 \or\@citea\NAT@hyper@{\NAT@alias}%
193 \fi \NAT@def@citea
194 \else
195     \ifcase\NAT@ctype
196     \if\relax\NAT@date\relax
197         \@citea\NAT@hyper@{\NAT@nmfmt{\NAT@nm}}%
198     \else
199         \ifx\NAT@last@nm\NAT@nm\NAT@yrsep
200             \ifx\NAT@last@yr\NAT@year
201                 \def\NAT@temp{{?}}%
202                 \ifx\NAT@temp\NAT@exlab\PackageWarningNoLine{natbib}%
203                     {Multiple citation on page \thepage: same authors and
204                     year\MessageBreak without distinguishing extra
205                     letter,\MessageBreak appears as question mark}\fi
206                 \NAT@hyper@{\NAT@exlab}%
207             \else
208                 \unskip\NAT@spacechar
209                 \NAT@hyper@{\NAT@date}%
210             \fi
211         \else
212             \@citea\NAT@hyper@{%
213                 \NAT@nmfmt{\NAT@nm}%
214                 \hyper@natlinkbreak{\NAT@spacechar\NAT@@open\if*#1*\else#1\NAT@spacechar\fi}%
215                 {\@citeb\@extra@b@citeb}%
216                 \NAT@date
217             }%
218         \fi
219     \fi
220 \or\@citea\NAT@hyper@{\NAT@nmfmt{\NAT@nm}}%
221 \or\@citea\NAT@hyper@{\NAT@date}%
222 \or\@citea\NAT@hyper@{\NAT@alias}%
223 \fi
224 \if\relax\NAT@date\relax
225     \NAT@def@citea
226 \else

```

```

227 \NAT@def@citea@close
228 \fi
229 \fi
230 }}\ifNAT@swa\else

```

将页码放在括号外边，并且置于上标。

```

231 % \if*#2*\else\NAT@cmt#2\fi
232 \if\relax\NAT@date\relax\else\NAT@@close\fi
233 \if*#2*\else\textsuperscript{#2}\fi
234 \fi}{#1}{#2}}

```

Author-year 模式的 \citep 的页码：

```

235 \renewcommand\NAT@cite%
236 [3]{\ifNAT@swa\NAT@@open\if*#2*\else#2\NAT@spacechar\fi
237 #1\NAT@@close\if*#3*\else\textsuperscript{#3}\fi\else#1\fi\endgroup}

```

thebibliography 参考文献列表的标签左对齐

```

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

```

\url 使用 xurl 宏包的方法，增加 URL 可断行的位置。

```

239 \g@addto@macro\UrlBreaks{%
240 \do0\do1\do2\do3\do4\do5\do6\do7\do8\do9%
241 \do\A\do\B\do\C\do\D\do\E\do\F\do\G\do\H\do\I\do\J\do\K\do\L\do\M
242 \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
243 \do\a\do\b\do\c\do\d\do\e\do\f\do\g\do\h\do\i\do\j\do\k\do\l\do\m
244 \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
245 }
246 \Urlmuskip=0mu plus 0.1mu
247 \</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。

```

248 (*authoryear | numerical)
249 INTEGERS {
250 uppercase.name
251 max.num.authors
252 period.between.author.year
253 sentence.case.title
254 link.title
255 show.mark
256 slash.for.extraction
257 in.booktitle
258 italic.journal
259 bold.journal.volume
260 show.missing.address.publisher
261 show.url
262 show.doi
263 show.note
264 (*authoryear)
265 lang.zh.order
266 lang.ja.order
267 lang.en.order

```

```

268 lang.ru.order
269 lang.other.order
270 </authoryear>
271 }
272

```

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。

```

273 FUNCTION {load.config}
274 {

```

英文姓名转为全大写：

```

275 <!*nouppercase&!thu>
276 #1 'uppercase.name :=
277 </!*nouppercase&!thu>
278 <!*nouppercase | thu>
279 #0 'uppercase.name :=
280 </!*nouppercase | thu>

```

最多显示的作者数量：

```

281 #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：

```

282 <!*authoryear>
283 <!*period&!2005&!ustc>
284 #0 'period.between.author.year :=
285 </!*period&!2005&!ustc>
286 <!*period | 2005 | ustc>
287 #1 'period.between.author.year :=
288 </period | 2005 | ustc>
289 </authoryear>

```

英文标题转为 sentence case（句首字母大写，其余小写）：

```

290 #1 'sentence.case.title :=
291 <!*nosentencecase>
292 #0 'sentence.case.title :=
293 </!*nosentencecase>

```

在标题添加超链接：

```

294 #0 'link.title :=
295 <!*linktitle>
296 #1 'link.title :=
297 </linktitle>

```

著录文献类型标识（比如“[M/OL]”）：

```

298 #1 'show.mark :=
299 <!*nomark>
300 #0 'show.mark :=
301 </nomark>

```

使用“/”表示析出文献

```

302 #1 'slash.for.extraction :=
303 <!*noslash>
304 #0 'slash.for.extraction :=
305 </noslash>

```

使用“In:”表示析出文献

```

306 #0 'in.booktitle :=

```

期刊名使用斜体：

```

307 #0 'italic.journal :=
308 <!*italicjournal>
309 #1 'italic.journal :=
310 </italicjournal>

```

期刊的卷使用粗体:

```
311 #0 'bold.journal.volume :=
```

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.”:

```
312 <!*noslsn&!thu&!ustc>
313 #1 'show.missing.address.publisher :=
314 </!noslsn&!thu&!ustc>
315 <!*noslsn | thu | ustc>
316 #0 'show.missing.address.publisher :=
317 </noslsn | thu | ustc>
```

是否著录 URL:

```
318 #1 'show.url :=
319 <*nourl>
320 #0 'show.url :=
321 </nourl>
```

是否著录 DOI:

```
322 <!*nodoi&!2005>
323 #1 'show.doi :=
324 </!nodoi&!2005>
325 <!*nodoi | 2005>
326 #0 'show.doi :=
327 </nodoi | 2005>
```

在每一条文献最后输出注释 (note) 的内容:

```
328 #0 'show.note :=
```

参考文献表按照“著者-出版年”组织时, 各个文种的顺序:

```
329 <*authoryear>
330 #1 'lang.zh.order :=
331 #2 'lang.ja.order :=
332 #3 'lang.en.order :=
333 #4 'lang.ru.order :=
334 #5 'lang.other.order :=
335 </authoryear>
336 }
337
```

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.

```
338 ENTRY
339 { address
340   author
341   booktitle
342   date
343   doi
344   edition
345   editor
346   howpublished
347   institution
348   journal
349   key
350   language
351   mark
352   medium
353   note
```

```

354     number
355     organization
356     pages
357     publisher
358     school
359     series
360     title
361     translator
362     url
363     urldate
364     volume
365     year
366 }
367 { entry.lang entry.is.electronic entry.numbered }

```

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

```

368 { label extra.label sort.label short.list entry.mark entry.url }
369

```

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
```

```
370 INTEGERS { output.state before.all mid.sentence after.sentence after.block after.slash }
371
372 INTEGERS { lang.zh lang.ja lang.en lang.ru lang.other }
373
374 INTEGERS { charptr len }
375
376 FUNCTION {init.state.consts}
377 { #0 'before.all :=
378   #1 'mid.sentence :=
379   #2 'after.sentence :=
380   #3 'after.block :=
381   #4 'after.slash :=
382   #3 'lang.zh :=
383   #4 'lang.ja :=
384   #1 'lang.en :=
385   #2 'lang.ru :=
386   #0 'lang.other :=
387 }
388
```

下面是一些常量的定义

```
389 FUNCTION {bbl.anonymous}
390 { entry.lang lang.zh =
391   { "佚名" }
392   { "Anon" }
393   if$
394 }
395
396 FUNCTION {bbl.space}
397 { entry.lang lang.zh =
398   { "\ " }
399   { " " }
400   if$
401 }
402
```



```

403 FUNCTION {bbl.et.al}
404 { entry.lang lang.zh =
405   { " 等" }
406   { entry.lang lang.ja =
407     { " 他" }
408     { entry.lang lang.ru =
409       { "идр" }
410       { "et~al." }
411       if$
412     }
413   if$
414 }
415 if$
416 }
417
418 FUNCTION {citation.et.al}
419 { bbl.et.al }
420
421 FUNCTION {bbl.colon} { ": " }
422
423 < *2015 >
424 FUNCTION {bbl.wide.space} { "\quad " }
425 < /2015 >
426 < *2005 >
427 FUNCTION {bbl.wide.space} { "\ " }
428 < /2005 >
429
430 < !thu >
431 FUNCTION {bbl.slash} { "//\allowbreak " }
432 < /!thu >
433 < *thu >
434 FUNCTION {bbl.slash} { " //" }
435 < /thu >
436
437 FUNCTION {bbl.sine.loco}
438 { entry.lang lang.zh =
439   { "[出版地不详]" }
440   { "[S.l.]" }
441   if$
442 }
443
444 FUNCTION {bbl.sine.nomine}
445 { entry.lang lang.zh =
446   { "[出版者不详]" }
447   { "[s.n.]" }
448   if$
449 }
450
451 FUNCTION {bbl.sine.loco.sine.nomine}
452 { entry.lang lang.zh =
453   { "[出版地不详: 出版者不详]" }
454   { "[S.l.: s.n.]" }
455   if$
456 }
457

```

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

```

458 FUNCTION {not}
459 { { #0 }
460   { #1 }
461   if$

```

```

462 }
463
464 FUNCTION {and}
465 {   'skip$
466     { pop$ #0 }
467   if$
468 }
469
470 FUNCTION {or}
471 {   { pop$ #1 }
472     'skip$
473   if$
474 }
475

```

the variables s and t are temporary string holders

```

476 STRINGS { s t }
477
478 FUNCTION {output.nonnull}
479 { 's :=
480   output.state mid.sentence =
481     { ", " * write$ }
482     { output.state after.block =
483         { add.period$ write$
484           newline$
485           "\newblock " write$
486         }
487         { output.state before.all =
488             'write$
489             { output.state after.slash =
490                 { bbl.slash * write$
491                   newline$
492                 }
493                 { add.period$ " " * write$ }
494             if$
495           }
496         if$
497       }
498     if$
499     mid.sentence 'output.state :=
500   }
501   if$
502   s
503 }
504
505 FUNCTION {output}
506 { duplicate$ empty$
507   'pop$
508   'output.nonnull
509   if$
510 }
511
512 FUNCTION {output.after}
513 { 't :=
514   duplicate$ empty$
515   'pop$
516   { 's :=
517     output.state mid.sentence =
518       { t * write$ }
519       { output.state after.block =
520           { add.period$ write$
521             newline$
522             "\newblock " write$
523           }

```

```

524         { output.state before.all =
525           'write$
526         { output.state after.slash =
527           { bbl.slash * write$ }
528           { add.period$ " " * write$ }
529         if$
530       }
531     if$
532   }
533   if$
534   mid.sentence 'output.state :=
535 }
536 if$
537 s
538 }
539 if$
540 }
541
542 FUNCTION {output.check}
543 { 't :=
544   duplicates$ empty$
545   { pop$ "empty " t * " in " * cite$ * warning$ }
546   'output.nonnull
547   if$
548 }
549

```

This function finishes all entries.

```

550 FUNCTION {fin.entry}
551 { add.period$
552   write$
553   newline$
554 }
555
556 FUNCTION {new.block}
557 { output.state before.all =
558   'skip$
559   { output.state after.slash =
560     'skip$
561     { after.block 'output.state := }
562   if$
563 }
564 if$
565 }
566
567 FUNCTION {new.sentence}
568 { output.state after.block =
569   'skip$
570   { output.state before.all =
571     'skip$
572     { output.state after.slash =
573       'skip$
574       { after.sentence 'output.state := }
575     if$
576   }
577   if$
578 }
579 if$
580 }
581
582 FUNCTION {new.slash}
583 { output.state before.all =
584   'skip$
585   { slash.for.extraction

```

```

586         { after.slash 'output.state := }
587         { after.block 'output.state := }
588     if$
589 }
590 if$
591 }
592

```

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.

```

593 FUNCTION {new.block.checka}
594 { empty$
595   'skip$
596   'new.block
597   if$
598 }
599
600 FUNCTION {new.block.checkb}
601 { empty$
602   swap$ empty$
603   and
604   'skip$
605   'new.block
606   if$
607 }
608

```

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

```

609 FUNCTION {new.sentence.checka}
610 { empty$
611   'skip$
612   'new.sentence
613   if$
614 }
615
616 FUNCTION {new.sentence.checkb}
617 { empty$
618   swap$ empty$
619   and
620   'skip$
621   'new.sentence
622   if$
623 }
624

```

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.

<pre> field.or.null(s) == BEGIN </pre>
--

```

        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

```

625 FUNCTION {field.or.null}
626 { duplicate$ empty$
627   { pop$ "" }
628   'skip$
629   if$
630 }
631
632 FUNCTION {italicize}
633 { duplicate$ empty$
634   { pop$ "" }
635   { "\textit{" swap$ * "}" * }
636   if$
637 }
638

```

B.4.1 Detect Language

```

639 INTEGERS { byte second.byte }
640
641 INTEGERS { char.lang tmp.lang }
642
643 STRINGS { tmp.str }
644
645 FUNCTION {get.str.lang}
646 { 'tmp.str :=
647   lang.other 'tmp.lang :=
648   #1 'charptr :=
649   tmp.str text.length$ #1 + 'len :=
650   { charptr len < }
651   { tmp.str charptr #1 substring$ chr.to.int$ 'byte :=
652     byte #128 <
653     { charptr #1 + 'charptr :=
654       byte #64 > byte #91 < and byte #96 > byte #123 < and or
655       { lang.en 'char.lang := }
656       { lang.other 'char.lang := }
657       if$
658     }
659     { tmp.str charptr #1 + #1 substring$ chr.to.int$ 'second.byte :=
660       byte #224 <

```

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

```

661     { charptr #2 + 'charptr :=
662       byte #207 > byte #212 < and
663       byte #212 = second.byte #176 < and or
664       { lang.ru 'char.lang := }
665       { lang.other 'char.lang := }
666       if$

```

```

667     }
668     { byte #240 <

```

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

```

669         { charptr #3 + 'charptr :=
670           byte #227 > byte #234 < and
671           { lang.zh 'char.lang := }

```

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

```

672         { byte #227 =
673           { second.byte #143 >
674             { lang.zh 'char.lang := }

```

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

```

675           { second.byte #128 > second.byte #132 < and
676             { lang.ja 'char.lang := }
677             { lang.other 'char.lang := }
678           if$
679         }
680       if$
681     }

```

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

```

682     { byte #239 =
683       second.byte #163 > second.byte #172 < and and
684       { lang.zh 'char.lang := }
685       { lang.other 'char.lang := }
686     if$
687   }
688   if$
689 }
690 if$
691 }

```

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.

```

692     { charptr #4 + 'charptr :=
693       byte #240 = second.byte #159 > and
694       { lang.zh 'char.lang := }
695       { lang.other 'char.lang := }
696     if$
697   }
698   if$
699 }
700 if$
701 }
702 if$
703   char.lang tmp.lang >
704   { char.lang 'tmp.lang := }
705   'skip$
706   if$
707 }
708 while$
709   tmp.lang
710 }
711
712 FUNCTION {check.entry.lang}
713 { author field.or.null
714   title field.or.null *
715   get.str.lang
716 }
717
718 FUNCTION {set.entry.lang}
719 { language empty$
720   { check.entry.lang }

```

```

721 { language "english" = language "american" = or language "british" = or
722   { lang.en }
723   { language "chinese" =
724     { lang.zh }
725     { language "japanese" =
726       { lang.ja }
727       { language "russian" =
728         { lang.ru }
729         { check.entry.lang }
730       if$
731     }
732   if$
733 }
734 if$
735 }
736 if$
737 }
738 if$
739 'entry.lang :=
740 }
741
742 FUNCTION {set.entry.numbered}
743 { type$ "patent" =
744   type$ "standard" = or
745   type$ "techreport" = or
746   { #1 'entry.numbered := }
747   { #0 'entry.numbered := }
748   if$
749 }
750

```

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.

```

751 INTEGERS { nameptr namesleft numnames name.lang }
752
753 FUNCTION {format.names}
754 { 's :=
755   #1 'nameptr :=
756   s num.names$ 'numnames :=
757   numnames 'namesleft :=
758   { namesleft #0 > }
759   { s nameptr "{vv~}{ll}{, jj}{, ff}" format.name$ 't :=
760     nameptr max.num.authors >
761     { bbl.et.al
762       #1 'namesleft :=
763     }
764     { t "others" =
765       { bbl.et.al }
766       { t get.str.lang 'name.lang :=
767         name.lang lang.en =
768         { t #1 "{vv~}{ll}{~f{~}}" format.name$
769           uppercase.name
770           { "u" change.case$ }
771           'skip$
772           if$
773           t #1 "{, jj}" format.name$ *
774         }
775         { t #1 "{ll}{ff}" format.name$ }
776       if$
777     }
778     if$
779   }
780   if$
781   nameptr #1 >
782   { ", " swap$ * * }

```



```

783         'skip$
784     if$
785         nameptr #1 + 'nameptr :=
786         namesleft #1 - 'namesleft :=
787     }
788 while$
789 }
790
791 FUNCTION {format.key}
792 { empty$
793   { key field.or.null }
794   { "" }
795   if$
796 }
797
798 FUNCTION {format.authors}
799 { author empty$ not
800   { author format.names }
801   { "empty author in " cite$ * warning$
802 (*authoryear)
803   bbl.anonymous
804 (/authoryear)
805 (*numerical)
806   ""
807 (/numerical)
808   }
809   if$
810 }
811
812 FUNCTION {format.editors}
813 { editor empty$
814   { "" }
815   { editor format.names }
816   if$
817 }
818
819 FUNCTION {format.translators}
820 { translator empty$
821   { "" }
822   { translator format.names
823     entry.lang lang.zh =
824     { translator num.names$ #3 >
825       { " 译" * }
826       { ", 译" * }
827       if$
828     }
829     'skip$
830     if$
831   }
832   if$
833 }
834
835 FUNCTION {format.full.names}
836 { 's :=
837   #1 'nameptr :=
838   s num.names$ 'numnames :=
839   numnames 'namesleft :=
840   { namesleft #0 > }
841   { s nameptr "{vv~}{ll}{, jj}{, ff}" format.name$ 't :=
842     t get.str.lang 'name.lang :=
843     name.lang lang.en =
844     { t #1 "{vv~}{ll}" format.name$ 't := }
845     { t #1 "{ll}{ff}" format.name$ 't := }
846     if$

```

```

847     nameptr #1 >
848     {
849         namesleft #1 >
850         { ", " * t * }
851         {
852             numnames #2 >
853             { ", " * }
854             'skip$
855             if$
856             t "others" =
857             { " et~al." * }
858             { " and " * t * }
859             if$
860         }
861         if$
862     }
863     't
864     if$
865     nameptr #1 + 'nameptr :=
866     namesleft #1 - 'namesleft :=
867 }
868 while$
869 }
870
871 FUNCTION {author.editor.full}
872 { author empty$
873   { editor empty$
874     { "" }
875     { editor format.full.names }
876     if$
877   }
878   { author format.full.names }
879   if$
880 }
881
882 FUNCTION {author.full}
883 { author empty$
884   { "" }
885   { author format.full.names }
886   if$
887 }
888
889 FUNCTION {editor.full}
890 { editor empty$
891   { "" }
892   { editor format.full.names }
893   if$
894 }
895
896 FUNCTION {make.full.names}
897 { type$ "book" =
898   type$ "inbook" =
899   or
900   'author.editor.full
901   { type$ "collection" =
902     type$ "proceedings" =
903     or
904     'editor.full
905     'author.full
906     if$
907   }
908   if$
909 }
910

```

```

911 FUNCTION {output.bibitem}
912 { newline$
913   "\bibitem[" write$
914     label "]" *
915     make.full.names duplicate$ short.list =
916       { pop$ }
917       { * }
918   if$
919   's :=
920   s text.length$ 'charptr :=
921     { charptr #0 > s charptr #1 substring$ "[" = not and }
922     { charptr #1 - 'charptr := }
923   while$
924   charptr #0 >
925     { "{" s * "}" * }
926     { s }
927   if$
928   "]" * write$
929   cite$ write$
930   "]" write$
931   newline$
932   ""
933   before.all 'output.state :=
934 }
935

```

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.

```

936 FUNCTION {change.sentence.case}
937 { entry.lang lang.en =
938   { "t" change.case$ }
939   'skip$
940   if$
941 }
942
943 FUNCTION {add.link}
944 { url empty$ not
945   { "\href{" url * "}" * swap$ * "}" * }
946   { doi empty$ not
947     { "\href{http://dx.doi.org/" doi * "}" * swap$ * "}" * }
948     'skip$
949     if$
950   }
951   if$
952 }
953
954 FUNCTION {format.title}
955 { title empty$
956   { "" }
957   { title
958     sentence.case.title
959     'change.sentence.case
960     'skip$
961     if$
962     entry.numbered number empty$ not and
963     { bbl.colon * number * }
964     'skip$
965     if$

```

```

966      link.title
967      'add.link
968      'skip$
969      if$
970    }
971  if$
972 }
973

```

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

```

```

974 FUNCTION {tie.or.space.connect}
975 { duplicate$ text.length$ #3 <
976   { "~" }
977   { " " }
978   if$
979   swap$ * *
980 }
981

```

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

```

```

982 FUNCTION {either.or.check}
983 { empty$
984   'pop$
985   { "can't use both " swap$ * " fields in " * cite$ * warning$ }
986   if$
987 }
988

```

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.

```

989 FUNCTION {is.digit}

```

```

990 { duplicate$ empty$
991   { pop$ #0 }
992   { chr.to.int$
993     duplicate$ "0" chr.to.int$ <
994     { pop$ #0 }
995     { "9" chr.to.int$ >
996       { #0 }
997       { #1 }
998     if$
999   }
1000   if$
1001 }
1002 if$
1003 }
1004
1005 FUNCTION {is.number}
1006 { 's :=
1007   s empty$
1008   { #0 }
1009   { s text.length$ 'charptr :=
1010     { charptr #0 >
1011       s charptr #1 substring$ is.digit
1012       and
1013     }
1014     { charptr #1 - 'charptr := }
1015     while$
1016     charptr not
1017   }
1018   if$
1019 }
1020
1021 FUNCTION {format.volume}
1022 { volume empty$ not
1023   { volume is.number
1024     { entry.lang lang.zh =
1025       { " 第 " volume * " 卷" * }
1026       { "volume" volume tie.or.space.connect }
1027     if$
1028   }
1029   { volume }
1030   if$
1031 }
1032 { "" }
1033 if$
1034 }
1035
1036 FUNCTION {format.number}
1037 { number empty$ not
1038   { number is.number
1039     { entry.lang lang.zh =
1040       { " 第 " number * " 册" * }
1041       { "number" number tie.or.space.connect }
1042     if$
1043   }
1044   { number }
1045   if$
1046 }
1047 { "" }
1048 if$
1049 }
1050
1051 FUNCTION {format.volume.number}
1052 { volume empty$ not
1053   { format.volume }

```

```

1054     { format.number }
1055   if$
1056 }
1057
1058 FUNCTION {format.title.vol.num}
1059 { title
1060   sentence.case.title
1061   'change.sentence.case
1062   'skip$
1063   if$
1064   entry.numbered
1065     { number empty$ not
1066       { bbl.colon * number * }
1067       'skip$
1068       if$
1069     }
1070     { format.volume.number 's :=
1071       s empty$ not
1072       { bbl.colon * s * }
1073       'skip$
1074       if$
1075     }
1076   if$
1077 }
1078
1079 FUNCTION {format.series.vol.num.title}
1080 { format.volume.number 's :=
1081   series empty$ not
1082   { series
1083     sentence.case.title
1084     'change.sentence.case
1085     'skip$
1086     if$
1087     entry.numbered
1088     { bbl.wide.space * }
1089     { bbl.colon *
1090       s empty$ not
1091       { s * bbl.wide.space * }
1092       'skip$
1093     }
1094     if$
1095     if$
1096     title *
1097     sentence.case.title
1098     'change.sentence.case
1099     'skip$
1100     if$
1101     entry.numbered number empty$ not and
1102     { bbl.colon * number * }
1103     'skip$
1104     if$
1105   }
1106   { format.title.vol.num }
1107   if$
1108   link.title
1109   'add.link
1110   'skip$
1111   if$
1112 }
1113
1114 FUNCTION {format.booktitle.vol.num}
1115 { booktitle
1116   entry.numbered
1117   'skip$

```

```

1118 { format.volume.number 's :=
1119     s empty$ not
1120     { bbl.colon * s * }
1121     'skip$
1122     if$
1123 }
1124 if$
1125 }
1126
1127 FUNCTION {format.series.vol.num.booktitle}
1128 { format.volume.number 's :=
1129     series empty$ not
1130     { series bbl.colon *
1131         entry.numbered not s empty$ not and
1132         { s * bbl.wide.space * }
1133         'skip$
1134         if$
1135         booktitle *
1136     }
1137     { format.booktitle.vol.num }
1138     if$
1139     in.booktitle
1140     { duplicates$ empty$ not entry.lang lang.en = and
1141         { "In: " swap$ * }
1142         'skip$
1143         if$
1144     }
1145     'skip$
1146     if$
1147 }
1148
1149 FUNCTION {format.journal}
1150 { journal
1151     italic.journal entry.lang lang.en = and
1152     'italicize
1153     'skip$
1154     if$
1155 }
1156

```

B.4.4 Format entry type mark

```

1157 FUNCTION {set.entry.mark}
1158 { entry.mark empty$ not
1159     'pop$
1160     { mark empty$ not
1161         { pop$ mark 'entry.mark := }
1162         { 'entry.mark := }
1163         if$
1164     }
1165     if$
1166 }
1167
1168 FUNCTION {format.mark}
1169 { show.mark
1170     < *thu>
1171     type$ "phdthesis" = type$ "mastersthesis" = or type$ "patent" = or
1172     medium empty$ not or entry.is.electronic or
1173     and
1174     < /thu>
1175     { medium empty$ not
1176         { entry.mark "/" * medium * 'entry.mark := }
1177         { entry.is.electronic
1178             { entry.mark "/OL" * 'entry.mark := }

```

```

1179         'skip$
1180     if$
1181 }
1182 if$
1183 <!*thu>
1184     "\allowbreak[" entry.mark * "]" *
1185 </!thu>
1186 <*thu>
1187     " [" entry.mark * "]" *
1188 </thu>
1189 }
1190 { "" }
1191 if$
1192 }
1193

```

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.

```

1194 FUNCTION {num.to.ordinal}
1195 { duplicate$ text.length$ 'charptr :=
1196   duplicate$ charptr #1 substring$ 's :=
1197   s "1" =
1198     { "st" * }
1199     { s "2" =
1200       { "nd" * }
1201       { s "3" =
1202         { "rd" * }
1203         { "th" * }
1204         if$
1205       }
1206     }
1207   if$
1208 }
1209 }
1210
1211 FUNCTION {format.edition}
1212 { edition empty$
1213   { "" }
1214   { edition is.number
1215     { entry.lang lang.zh =
1216       { edition " 版" * }
1217       { edition num.to.ordinal " ed." * }
1218       if$
1219     }
1220     { entry.lang lang.en =
1221       { edition change.sentence.case 's :=
1222         s "Revised" = s "Revised edition" = or
1223         { "Rev. ed." }
1224         { s " ed." *}
1225         if$
1226       }
1227       { edition }
1228       if$
1229     }
1230   }
1231 }
1232 if$
1233 }
1234

```


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。

```
1235 FUNCTION {format.publisher}
1236 { publisher empty$ not
1237   { publisher }
1238   { school empty$ not
1239     { school }
1240     { organization empty$ not
1241       { organization }
1242       { institution empty$ not
1243         { institution }
1244         { "" }
1245       }
1246     }
1247   }
1248   if$
1249 }
1250 }
1251 if$
1252 }
1253
1254 FUNCTION {format.address.publisher}
1255 { address empty$ not
1256   { address
1257     format.publisher empty$ not
1258     { bbl.colon * format.publisher * }
1259     { entry.is.electronic not show.missing.address.publisher and
1260       { bbl.colon * bbl.sine.nomine * }
1261       'skip$
1262     }
1263   }
1264   if$
1265 }
1266 { entry.is.electronic not show.missing.address.publisher and
1267   { format.publisher empty$ not
1268     { bbl.sine.loco bbl.colon * format.publisher * }
1269     { bbl.sine.loco.sine.nomine }
1270   }
1271   if$
1272   { format.publisher empty$ not
1273     { format.publisher }
1274     { "" }
1275   }
1276   if$
1277 }
1278 }
1279 if$
1280 }
1281
```

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 patent 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`。

```
1282 FUNCTION {extract.before.dash}
1283 { duplicate$ empty$
1284   { pop$ "" }
1285   { 's :=
```

```

1286     #1 'charptr :=
1287     s text.length$ #1 + 'len :=
1288     { charptr len <
1289       s charptr #1 substring$ "-" = not
1290       and
1291     }
1292     { charptr #1 + 'charptr := }
1293     while$
1294     s #1 charptr #1 - substring$
1295   }
1296   if$
1297 }
1298
1299 FUNCTION {extract.after.dash}
1300 { duplicate$ empty$
1301   { pop$ "" }
1302   { 's :=
1303     #1 'charptr :=
1304     s text.length$ #1 + 'len :=
1305     { charptr len <
1306       s charptr #1 substring$ "-" = not
1307       and
1308     }
1309     { charptr #1 + 'charptr := }
1310     while$
1311     { charptr len <
1312       s charptr #1 substring$ "-" =
1313       and
1314     }
1315     { charptr #1 + 'charptr := }
1316     while$
1317     s charptr global.max$ substring$
1318   }
1319   if$
1320 }
1321
1322 FUNCTION {contains.dash}
1323 { duplicate$ empty$
1324   { pop$ #0 }
1325   { 's :=
1326     { s empty$ not
1327       s #1 #1 substring$ "-" = not
1328       and
1329     }
1330     { s #2 global.max$ substring$ 's := }
1331     while$
1332     s empty$ not
1333   }
1334   if$
1335 }
1336

```

著者-出版年制必须提取出年份

```

1337 FUNCTION {format.year}
1338 { year empty$ not
1339   { year extract.before.dash }
1340   { date empty$ not
1341     { date extract.before.dash }
1342     { "empty year in " cite$ * warning$
1343       urldate empty$ not
1344       { "[" urldate extract.before.dash * "]" * }
1345       { "" }
1346     } if$
1347   }

```

```

1348     if$
1349   }
1350   if$
1351   extra.label *
1352 }
1353

```

专利和报纸都是使用日期而不是年

```

1354 FUNCTION {format.date}
1355 { type$ "patent" = type$ "newspaper" = or
1356   date empty$ not and
1357   { date }
1358   { year }
1359   if$
1360 }
1361

```

更新、修改日期只用于电子资源 `elctronic`

```

1362 FUNCTION {format.editdate}
1363 { date empty$ not
1364   { "\allowbreak(" date * ")" * }
1365   { "" }
1366   if$
1367 }
1368

```

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。

```

1369 FUNCTION {format.urldate}
1370 { urldate empty$ not entry.is.electronic and
1371   { "\allowbreak[" urldate * "]" * }
1372   { "" }
1373   if$
1374 }
1375

```

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**。

```

1376 FUNCTION {hyphenate}
1377 { 't :=
1378   ""
1379   { t empty$ not }
1380   { t #1 #1 substring$ "-" =
1381     { "-" *
1382       { t #1 #1 substring$ "-" = }
1383       { t #2 global.max$ substring$ 't := }
1384       while$
1385     }
1386     { t #1 #1 substring$ *
1387       t #2 global.max$ substring$ 't :=
1388     }
1389     if$
1390   }
1391   while$
1392 }
1393

```

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

```

1394 FUNCTION {format.pages}
1395 { pages empty$
1396   { "" }
1397   { pages hyphenate }
1398   if$
1399 }
1400

```

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。

```

1401 FUNCTION {format.journal.volume}
1402 { volume empty$ not
1403   { bold.journal.volume
1404     { "\textbf{" volume * "}" * }
1405     { volume }
1406     if$
1407   }
1408   { "" }
1409   if$
1410 }
1411
1412 FUNCTION {format.journal.number}
1413 { number empty$ not
1414   { "\penalty0 (" number * ")" * }
1415   { "" }
1416   if$

```

```

1417 }
1418
1419 FUNCTION {format.journal.pages}
1420 { pages empty$
1421   { "" }
1422   { "\penalty0 " pages hyphenate * }
1423   if$
1424 }
1425

```

连续出版物的年卷期有起止范围，需要特殊处理

```

1426 FUNCTION {format.periodical.year.volume.number}
1427 { year empty$ not
1428   { year extract.before.dash }
1429   { "empty year in periodical " cite$ * warnings$ }
1430   if$
1431   volume empty$ not
1432   { ", " * volume extract.before.dash * }
1433   'skip$
1434   if$
1435   number empty$ not
1436   { "\penalty0 (" * number extract.before.dash * ")" * }
1437   'skip$
1438   if$
1439   year contains.dash
1440   { "--" *
1441     year extract.after.dash empty$
1442     volume extract.after.dash empty$ and
1443     number extract.after.dash empty$ and not
1444     { year extract.after.dash empty$ not
1445       { year extract.after.dash * }
1446       { year extract.before.dash * }
1447       if$
1448       volume empty$ not
1449       { ", " * volume extract.after.dash * }
1450       'skip$
1451       if$
1452       number empty$ not
1453       { "\penalty0 (" * number extract.after.dash * ")" * }
1454       'skip$
1455       if$
1456     }
1457     'skip$
1458   }
1459   if$
1460   'skip$
1461   if$
1462 }
1463

```

B.4.9 Format url and doi

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

```

1464 FUNCTION {check.url}
1465 { url empty$ not
1466   { "\url{" url * "}" * 'entry.url :=
1467     #1 'entry.is.electronic :=
1468   }
1469   { howpublished empty$ not
1470     { howpublished #1 #5 substring$ "\url{" =
1471       { howpublished 'entry.url :=
1472         #1 'entry.is.electronic :=
1473     }

```

```

1474         'skip$
1475     if$
1476 }
1477 { note empty$ not
1478     { note #1 #5 substring$ "\url{" =
1479         { note 'entry.url :=
1480             #1 'entry.is.electronic :=
1481         }
1482         'skip$
1483     if$
1484     }
1485     'skip$
1486 if$
1487 }
1488 if$
1489 }
1490 if$
1491 }
1492
1493 FUNCTION {format.url}
1494 { entry.url empty$ not
1495     { new.block entry.url }
1496     { "" }
1497 if$
1498 }
1499

```

需要检测 DOI 是否已经包含在 URL 中。

```

1500 FUNCTION {check.doi}
1501 { doi empty$ not
1502     { #1 'entry.is.electronic := }
1503     'skip$
1504 if$
1505 }
1506
1507 FUNCTION {is.in.url}
1508 { 's :=
1509     s empty$
1510     { #1 }
1511     { entry.url empty$
1512         { #0 }
1513         { s text.length$ 'len :=
1514             entry.url text.length$ 'charptr :=
1515             { entry.url charptr len substring$ s = not
1516                 charptr #0 >
1517                 and
1518             }
1519             { charptr #1 - 'charptr := }
1520             while$
1521             charptr
1522         }
1523     if$
1524     }
1525 if$
1526 }
1527
1528 FUNCTION {format.doi}
1529 { ""
1530     doi empty$ not show.doi and
1531     { "" 's :=
1532         doi 't :=
1533         #0 'numnames :=
1534         { t empty$ not}
1535         { t #1 #1 substring$ 'tmp.str :=

```

```

1536         tmp.str "," = tmp.str " " = or t #2 #1 substring$ empty$ or
1537         { t #2 #1 substring$ empty$
1538           { s tmp.str * 's := }
1539           'skip$
1540         if$
1541         s empty$ s is.in.url or
1542         'skip$
1543         { numnames #1 + 'numnames :=
1544           numnames #1 >
1545           { ", " * }
1546           { "DOI: " * }
1547           if$
1548           "\doi{" s * "}" * *
1549         }
1550         if$
1551         "" 's :=
1552       }
1553       { s tmp.str * 's := }
1554     if$
1555     t #2 global.max$ substring$ 't :=
1556   }
1557   while$
1558   's :=
1559   s empty$ not
1560   { new.block s }
1561   { "" }
1562   if$
1563 }
1564 'skip$
1565 if$
1566 }
1567
1568 FUNCTION {check.electronic}
1569 { "" 'entry.url :=
1570   #0 'entry.is.electronic :=
1571   'check.doi
1572   'skip$
1573   if$
1574   'check.url
1575   'skip$
1576   if$
1577   medium empty$ not
1578   { medium "MT" = medium "DK" = or medium "CD" = or medium "OL" = or
1579     { #1 'entry.is.electronic := }
1580     'skip$
1581     if$
1582   }
1583   'skip$
1584   if$
1585 }
1586
1587 FUNCTION {format.note}
1588 { note empty$ not show.note and
1589   { note }
1590   { "" }
1591   if$
1592 }
1593

```

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

```

1594 FUNCTION {empty.misc.check}
1595 { author empty$ title empty$
1596   year empty$

```

```

1597 and and
1598 key empty$ not and
1599 { "all relevant fields are empty in " cite$ * warnings$ }
1600 'skip$
1601 if$
1602 }
1603

```

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

```

1604 FUNCTION {monograph}
1605 { output.bibitem
1606   author empty$ not
1607     { format.authors }
1608     { editor empty$ not
1609       { format.editors }
1610       { "empty author and editor in " cite$ * warnings$
1611 (*authoryear)
1612       bbl.anonymous
1613 (/authoryear)
1614 (*numerical)
1615       ""
1616 (/numerical)
1617     }
1618     if$
1619   }
1620   if$
1621   output
1622 (*authoryear)
1623   period.between.author.year
1624   'new.sentence
1625   'skip$
1626   if$
1627   format.year "year" output.check
1628 (/authoryear)
1629   new.block
1630   format.series.vol.num.title "title" output.check
1631   "M" set.entry.mark
1632   format.mark "" output.after
1633   new.block
1634   format.translators output
1635   new.sentence
1636   format.edition output
1637   new.block
1638   format.address.publisher output
1639 (*numerical)
1640   format.year "year" output.check
1641 (/numerical)
1642   format.pages bbl.colon output.after
1643   format.urldate "" output.after
1644   format.url output
1645   format.doi output
1646   new.block

```



```

1647 format.note output
1648 fin.entry
1649 }
1650

```

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

```

1651 FUNCTION {incollection}
1652 { output.bibitem
1653   format.authors output
1654   author format.key output
1655   (*authoryear)
1656   period.between.author.year
1657   'new.sentence
1658   'skip$
1659   if$
1660   format.year "year" output.check
1661   (/authoryear)
1662   new.block
1663   format.title "title" output.check
1664   "M" set.entry.mark
1665   format.mark "" output.after
1666   new.block
1667   format.translators output
1668   new.slash
1669   format.editors output
1670   new.block
1671   format.series.vol.num.booktitle "booktitle" output.check
1672   new.block
1673   format.edition output
1674   new.block
1675   format.address.publisher output
1676   (*numerical)
1677   format.year "year" output.check
1678   (/numerical)
1679   format.pages bbl.colon output.after
1680   format.urldate "" output.after
1681   format.url output
1682   format.doi output
1683   new.block
1684   format.note output
1685   fin.entry
1686 }
1687

```

B.5.3 连续出版物

```

1688 FUNCTION {periodical}
1689 { output.bibitem
1690   format.authors output
1691   author format.key output
1692   (*authoryear)
1693   period.between.author.year
1694   'new.sentence
1695   'skip$
1696   if$
1697   format.year "year" output.check
1698   (/authoryear)
1699   new.block

```

```

1700 format.title "title" output.check
1701 "J" set.entry.mark
1702 format.mark "" output.after
1703 new.block
1704 format.periodical.year.volume.number output
1705 new.block
1706 format.address.publisher output
1707 <*numerical>
1708 format.date "year" output.check
1709 </numerical>
1710 format.urldate "" output.after
1711 format.url output
1712 format.doi output
1713 new.block
1714 format.note output
1715 fin.entry
1716 }
1717

```

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.

```

1718 FUNCTION {article}
1719 { output.bibitem
1720 format.authors output
1721 author format.key output
1722 <*authoryear>
1723 period.between.author.year
1724 'new.sentence
1725 'skip$
1726 if$
1727 format.year "year" output.check
1728 </authoryear>
1729 new.block
1730 format.title "title" output.check
1731 "J" set.entry.mark
1732 format.mark "" output.after
1733 new.block
1734 format.journal "journal" output.check
1735 <*numerical>
1736 format.date "year" output.check
1737 </numerical>
1738 format.journal.volume output
1739 format.journal.number "" output.after
1740 format.journal.pages "" output.after
1741 format.urldate "" output.after
1742 format.url output
1743 format.doi output
1744 new.block
1745 format.note output
1746 fin.entry
1747 }
1748

```

B.5.5 专利文献

number 域也可以用来表示专利号。

```

1749 FUNCTION {patent}
1750 { output.bibitem

```

```

1751 format.authors output
1752 author format.key output
1753 {*authoryear}
1754 period.between.author.year
1755 'new.sentence
1756 'skip$
1757 if$
1758 format.year "year" output.check
1759 /authoryear}
1760 new.block
1761 format.title "title" output.check
1762 "P" set.entry.mark
1763 format.mark "" output.after
1764 new.block
1765 format.date "year" output.check
1766 format.urldate "" output.after
1767 format.url output
1768 format.doi output
1769 new.block
1770 format.note output
1771 fin.entry
1772 }
1773

```

B.5.6 电子资源

```

1774 FUNCTION {electronic}
1775 { #1 #1 check.electronic
1776 #1 'entry.is.electronic :=
1777 output.bibitem
1778 format.authors output
1779 author format.key output
1780 {*authoryear}
1781 period.between.author.year
1782 'new.sentence
1783 'skip$
1784 if$
1785 format.year "year" output.check
1786 /authoryear}
1787 new.block
1788 format.series.vol.num.title "title" output.check
1789 "EB" set.entry.mark
1790 format.mark "" output.after
1791 new.block
1792 format.address.publisher output
1793 {*numerical}
1794 date empty$
1795 { format.date output }
1796 'skip$
1797 if$
1798 /numerical}
1799 format.pages bbl.colon output.after
1800 format.editdate "" output.after
1801 format.urldate "" output.after
1802 format.url output
1803 format.doi output
1804 new.block
1805 format.note output
1806 fin.entry
1807 }
1808

```

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。

```
1809 FUNCTION {misc}
1810 { journal empty$ not
1811   'article
1812   { booktitle empty$ not
1813     'incollection
1814     { publisher empty$ not
1815       'monograph
1816       { entry.is.electronic
1817         'electronic
1818         { "Z" set.entry.mark
1819           monograph
1820         }
1821       }
1822     }
1823   }
1824   if$
1825 }
1826 }
1827 if$
1828 empty.misc.check
1829 }
1830
1831 FUNCTION {archive}
1832 { "A" set.entry.mark
1833   misc
1834 }
1835
```

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

```
1836 FUNCTION {book} { monograph }
1837
```

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

Required: title

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

```
1838 FUNCTION {booklet} { book }
1839
1840 FUNCTION {collection}
1841 { "G" set.entry.mark
1842   monograph
1843 }
1844
1845 FUNCTION {database}
1846 { "DB" set.entry.mark
1847   electronic
1848 }
1849
1850 FUNCTION {dataset}
1851 { "DS" set.entry.mark
1852   electronic
1853 }
1854
```

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。

```
1855 FUNCTION {inbook} { book }
1856
```

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

```
1857 FUNCTION {inproceedings}
1858 { "C" set.entry.mark
1859   incollection
1860 }
1861
```

The conference function is included for Scribe compatibility.

```
1862 FUNCTION {conference} { inproceedings }
1863
1864 FUNCTION {map}
1865 { "CM" set.entry.mark
1866   misc
1867 }
1868
```

A manual is technical documentation.

Required: title

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

```
1869 FUNCTION {manual} { monograph }
1870
```

A mastersthesis is a Master's thesis.

Required: author, title, school, year

Optional: type, address, month, note

```
1871 FUNCTION {mastersthesis}
1872 { *!thu }
1873 { "D" set.entry.mark
1874   /!thu }
1875 { *thu }
1876 { lang.zh entry.lang =
1877   { " 硕士学位论文" }
1878   { "D" }
1879   if$
1880   set.entry.mark
1881 /!thu }
1882 monograph
1883 }
1884
1885 FUNCTION {newspaper}
1886 { "N" set.entry.mark
1887   article
1888 }
1889
1890 FUNCTION {online}
1891 { "EB" set.entry.mark
1892   electronic
1893 }
1894
```

A phdthesis is like a mastersthesis.

Required: author, title, school, year

Optional: type, address, month, note

```
1895 <!*thu>
1896 FUNCTION {phdthesis} { mastersthesis }
1897 </!thu>
1898 <*thu>
1899 FUNCTION {phdthesis}
1900 { lang.zh entry.lang =
1901   { " 博士学位论文" }
1902   { "D" }
1903   if$
1904   set.entry.mark
1905   monograph
1906 }
1907 </thu>
1908
```

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

```
1909 FUNCTION {proceedings}
1910 { "C" set.entry.mark
1911   monograph
1912 }
1913
1914 FUNCTION {software}
1915 { "CP" set.entry.mark
1916   electronic
1917 }
1918
1919 FUNCTION {standard}
1920 { "S" set.entry.mark
1921   misc
1922 }
1923
```

A techreport is a technical report.

Required: author, title, institution, year

Optional: type, number, address, month, note

```
1924 FUNCTION {techreport}
1925 { "R" set.entry.mark
1926   misc
1927 }
1928
```

An unpublished is something that hasn't been published.

Required: author, title, note

Optional: month, year

```
1929 FUNCTION {unpublished}
1930 { "Z" set.entry.mark
1931   misc
1932 }
1933
```

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

```
1934 FUNCTION {default.type} { misc }
```

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

```

1936 MACRO {jan} {"January"}
1937
1938 MACRO {feb} {"February"}
1939
1940 MACRO {mar} {"March"}
1941
1942 MACRO {apr} {"April"}
1943
1944 MACRO {may} {"May"}
1945
1946 MACRO {jun} {"June"}
1947
1948 MACRO {jul} {"July"}
1949
1950 MACRO {aug} {"August"}
1951
1952 MACRO {sep} {"September"}
1953
1954 MACRO {oct} {"October"}
1955
1956 MACRO {nov} {"November"}
1957
1958 MACRO {dec} {"December"}
1959

```

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

```

1960 MACRO {acmcs} {"ACM Computing Surveys"}
1961
1962 MACRO {acta} {"Acta Informatica"}
1963
1964 MACRO {cacm} {"Communications of the ACM"}
1965
1966 MACRO {ibmjrd} {"IBM Journal of Research and Development"}
1967
1968 MACRO {ibmsj} {"IBM Systems Journal"}
1969
1970 MACRO {ieeese} {"IEEE Transactions on Software Engineering"}
1971
1972 MACRO {ieeetc} {"IEEE Transactions on Computers"}
1973
1974 MACRO {ieeetcad} {"IEEE Transactions on Computer-Aided Design of Integrated Circuits"}
1975 {"IEEE Transactions on Computer-Aided Design of Integrated Circuits"}
1976
1977 MACRO {ipl} {"Information Processing Letters"}
1978
1979 MACRO {jacm} {"Journal of the ACM"}
1980
1981 MACRO {jcss} {"Journal of Computer and System Sciences"}
1982

```

```

1983 MACRO {scp} {"Science of Computer Programming"}
1984
1985 MACRO {sicomp} {"SIAM Journal on Computing"}
1986
1987 MACRO {tocs} {"ACM Transactions on Computer Systems"}
1988
1989 MACRO {tods} {"ACM Transactions on Database Systems"}
1990
1991 MACRO {tog} {"ACM Transactions on Graphics"}
1992
1993 MACRO {toms} {"ACM Transactions on Mathematical Software"}
1994
1995 MACRO {toois} {"ACM Transactions on Office Information Systems"}
1996
1997 MACRO {toplas} {"ACM Transactions on Programming Languages and Systems"}
1998
1999 MACRO {tcs} {"Theoretical Computer Science"}
2000

```

B.7 Format labels

The `sortify` function converts to lower case after purifying; 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`.

```

2001 FUNCTION {sortify}
2002 { purify$
2003   "l" change.case$
2004 }
2005

```

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

```

2006 FUNCTION {chop.word}
2007 { 's :=
2008   'len :=
2009   s #1 len substring$ =
2010     { s len #1 + global.max$ substring$ }
2011     's
2012   if$
2013 }
2014

```

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\$ing the 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

```

```

2015 FUNCTION {format.lab.names}
2016 { 's :=
2017   s #1 "{vv~}{ll}{, jj}{, ff}" format.name$ 't :=
2018   t get.str.lang 'name.lang :=
2019   name.lang lang.en =
2020   { t #1 "{vv~}{ll}" format.name$ }
2021   { t #1 "{ll}{ff}" format.name$ }
2022   if$
2023   s num.names$ #1 >
2024   { bbl.space * citation.et.al * }
2025   'skip$
2026   if$
2027 }
2028
2029 FUNCTION {author.key.label}
2030 { author empty$
2031   { key empty$
2032     { cite$ #1 #3 substring$ }
2033     'key
2034     if$
2035   }
2036   { author format.lab.names }
2037   if$
2038 }
2039
2040 FUNCTION {author.editor.key.label}
2041 { author empty$
2042   { editor empty$
2043     { key empty$
2044       { cite$ #1 #3 substring$ }
2045       'key
2046       if$
2047     }
2048     { editor format.lab.names }
2049     if$
2050   }
2051   { author format.lab.names }
2052   if$
2053 }
2054
2055 FUNCTION {author.key.organization.label}
2056 { author empty$
2057   { key empty$
2058     { organization empty$
2059       { cite$ #1 #3 substring$ }
2060       { "The " #4 organization chop.word #3 text.prefix$ }
2061       if$
2062     }
2063     'key
2064     if$
2065   }
2066   { author format.lab.names }
2067   if$
2068 }
2069
2070 FUNCTION {editor.key.organization.label}
2071 { editor empty$
2072   { key empty$
2073     { organization empty$

```

```

2074         { cite$ #1 #3 substring$ }
2075         { "The " #4 organization chop.word #3 text.prefix$ }
2076     if$
2077     }
2078     'key
2079     if$
2080     }
2081     { editor format.lab.names }
2082 if$
2083 }
2084
2085 FUNCTION {calc.short.authors}
2086 { type$ "book" =
2087   type$ "inbook" =
2088   or
2089   'author.editor.key.label
2090   { type$ "collection" =
2091     type$ "proceedings" =
2092     or
2093     { editor empty$ not
2094       'editor.key.organization.label
2095       'author.key.organization.label
2096     if$
2097     }
2098     'author.key.label
2099   if$
2100   }
2101 if$
2102 'short.list :=
2103 }
2104
2105 FUNCTION {calc.label}
2106 { calc.short.authors
2107   short.list
2108   "("
2109   *
2110   format.year duplicate$ empty$
2111   short.list key field.or.null = or
2112   { pop$ "" }
2113   'skip$
2114 if$
2115 *
2116 'label :=
2117 }
2118

```

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`.

```

2119 (*authoryear)
2120 FUNCTION {sort.language.label}
2121 { entry.lang lang.zh =
2122   { lang.zh.order }
2123   { entry.lang lang.ja =
2124     { lang.ja.order }
2125     { entry.lang lang.en =
2126       { lang.en.order }
2127       { entry.lang lang.ru =
2128         { lang.ru.order }
2129         { lang.other.order }
2130       if$
2131     }
2132   if$
2133 }
2134 if$
2135 }
2136 if$
2137 int.to.chr$
2138 }
2139
2140 FUNCTION {sort.format.names}
2141 { 's :=
2142   #1 'nameptr :=
2143   ""
2144   s num.names$ 'numnames :=
2145   numnames 'namesleft :=
2146   { namesleft #0 > }
2147   {
2148     s nameptr "{vv{ } }{ll{ }}{ ff{ }}{ jj{ }}" format.name$ 't :=
2149     nameptr #1 >
2150     {
2151       " " *
2152       namesleft #1 = t "others" = and
2153       { "zzzzz" * }
2154       { numnames #2 > nameptr #2 = and
2155         { "zz" * year field.or.null * " " * }
2156         'skip$
2157       if$
2158       t sortify *
2159     }
2160     if$
2161   }
2162   { t sortify * }
2163   if$
2164   nameptr #1 + 'nameptr :=
2165   namesleft #1 - 'namesleft :=
2166 }
2167 while$
2168 }
2169

```

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`

```

2170 FUNCTION {sort.format.title}
2171 { 't :=
2172   "A " #2
2173   "An " #3
2174   "The " #4 t chop.word
2175   chop.word

```

```

2176 chop.word
2177 sortify
2178 #1 global.max$ substring$
2179 }
2180

```

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.

```

2181 FUNCTION {anonymous.sort}
2182 { entry.lang lang.zh =
2183   { "yi4 ming2" }
2184   { "anon" }
2185   if$
2186 }
2187
2188 FUNCTION {warn.empty.key}
2189 { entry.lang lang.zh =
2190   { "empty key in " cite$ * warning$ }
2191   'skip$
2192   if$
2193 }
2194
2195 FUNCTION {author.sort}
2196 { key empty$
2197   { warn.empty.key
2198     author empty$
2199     { anonymous.sort }
2200     { author sort.format.names }
2201     if$
2202   }
2203   { key sortify }
2204   if$
2205 }
2206
2207 FUNCTION {author.editor.sort}
2208 { key empty$
2209   { warn.empty.key
2210     author empty$
2211     { editor empty$
2212       { anonymous.sort }
2213       { editor sort.format.names }
2214       if$
2215     }
2216     { author sort.format.names }
2217     if$
2218   }
2219   { key sortify }
2220   if$
2221 }
2222
2223 FUNCTION {author.organization.sort}
2224 { key empty$
2225   { warn.empty.key
2226     author empty$
2227     { organization empty$
2228       { anonymous.sort }
2229       { "The " #4 organization chop.word sortify }
2230       if$
2231     }
2232     { author sort.format.names }
2233     if$
2234   }

```

```

2235     { key sortify }
2236   if$
2237 }
2238
2239 FUNCTION {editor.organization.sort}
2240 { key empty$
2241   { warn.empty.key
2242     editor empty$
2243       { organization empty$
2244         { anonymous.sort }
2245         { "The " #4 organization chop.word sortify }
2246       if$
2247     }
2248     { editor sort.format.names }
2249   if$
2250 }
2251 { key sortify }
2252 if$
2253 }
2254
2255 </authoryear>

```

顺序编码制的排序要简单得多

```

2256 (*numerical)
2257 INTEGERS { seq.num }
2258
2259 FUNCTION {init.seq}
2260 { #0 'seq.num :=}
2261
2262 FUNCTION {int.to.fix}
2263 { "000000000" swap$ int.to.str$ *
2264   #-1 #10 substring$
2265 }
2266
2267 </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!

```

2268 FUNCTION {presort}
2269 { set.entry.lang
2270   set.entry.numbered
2271   show.url show.doi check.electronic
2272   calc.label
2273   label sortify
2274   " "
2275   *
2276 (*authoryear)
2277   sort.language.label
2278   type$ "book" =
2279   type$ "inbook" =
2280   or
2281   'author.editor.sort
2282   { type$ "collection" =
2283     type$ "proceedings" =
2284     or
2285     'editor.organization.sort
2286     'author.sort
2287   if$
2288 }
2289 if$
2290 *
2291 " "

```

```

2292 *
2293 year field.or.null sortify
2294 *
2295 "    "
2296 *
2297 cite$
2298 *
2299 #1 entry.max$ substring$
2300 (/authoryear)
2301 (*numerical)
2302 seq.num #1 + 'seq.num :=
2303 seq.num int.to.fix
2304 (/numerical)
2305 'sort.label :=
2306 sort.label *
2307 #1 entry.max$ substring$
2308 'sort.key$ :=
2309 }
2310

```

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

```

```

2311 STRINGS { longest.label last.label next.extra }

```

```

2312
2313 INTEGERS { longest.label.width last.extra.num number.label }
2314
2315 FUNCTION {initialize.longest.label}
2316 { "" 'longest.label :=
2317   #0 int.to.chr$ 'last.label :=
2318   "" 'next.extra :=
2319   #0 'longest.label.width :=
2320   #0 'last.extra.num :=
2321   #0 'number.label :=
2322 }
2323
2324 FUNCTION {forward.pass}
2325 { last.label label =
2326   { last.extra.num #1 + 'last.extra.num :=
2327     last.extra.num int.to.chr$ 'extra.label :=
2328   }
2329   { "a" chr.to.int$ 'last.extra.num :=
2330     "" 'extra.label :=
2331     label 'last.label :=
2332   }
2333   if$
2334   number.label #1 + 'number.label :=
2335 }
2336
2337 FUNCTION {reverse.pass}
2338 { next.extra "b" =
2339   { "a" 'extra.label := }
2340   'skip$
2341   if$
2342   extra.label 'next.extra :=
2343   extra.label
2344   duplicate$ empty$
2345   'skip$
2346   { "{\natexlab{" swap$ * "}" * }
2347   if$
2348   'extra.label :=
2349   label extra.label * 'label :=
2350 }
2351
2352 FUNCTION {bib.sort.order}
2353 { sort.label 'sort.key$ :=
2354 }
2355

```

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.

```

2356 FUNCTION {begin.bib}
2357 { preamble$ empty$
2358   'skip$
2359   { preamble$ write$ newline$ }
2360   if$
2361   "\begin{thebibliography}{\" number.label int.to.str$ * \"} *
2362   write$ newline$
2363   "\providecommand{\natexlab}[1]{#1}"
2364   write$ newline$

```



```

2365 "\providecommand{\url}[1]{#1}"
2366 write$ newline$
2367 "\expandafter\ifx\csname urlstyle\endcsname\relax\relax\else"
2368 write$ newline$
2369 " \urlstyle{same}\fi"
2370 write$ newline$
2371 show.doi
2372 { "\providecommand{\href}[2]{\url{#2}}"
2373   write$ newline$
2374   "\providecommand{\doi}[1]{\href{https://doi.org/#1}{#1}}"
2375   write$ newline$
2376 }
2377 'skip$
2378 if$
2379 }
2380

```

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

```

2381 FUNCTION {end.bib}
2382 { newline$
2383   "\end{thebibliography}" write$ newline$
2384 }
2385

```

B.10 Main execution

Now we read in the .BIB entries.

```

2386 READ
2387
2388 EXECUTE {init.state.consts}
2389
2390 EXECUTE {load.config}
2391
2392 ⟨*numerical⟩
2393 EXECUTE {init.seq}
2394
2395 ⟨/numerical⟩
2396 ITERATE {presort}
2397

```

And now we can sort

```

2398 SORT
2399
2400 EXECUTE {initialize.longest.label}
2401
2402 ITERATE {forward.pass}
2403
2404 REVERSE {reverse.pass}
2405
2406 ITERATE {bib.sort.order}
2407
2408 SORT
2409
2410 EXECUTE {begin.bib}
2411

```

Now we produce the output for all the entries

```

2412 ITERATE {call.type$}
2413
2414 EXECUTE {end.bib}
2415 ⟨/authoryear | numerical⟩

```