

GB/T 7714-2015 BibT_EX style

Zeping Lee*

2020/03/04 v2.0

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 author-year 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

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”。用户应在导言区调用宏包 gbt7714，并且使用 \bibliographystyle 命令选择参考文献表的样式，比如：

```
\bibliographystyle{gbt7714-numerical} % 顺序编码制
```

或者

```
\bibliographystyle{gbt7714-author-year} % 著者-出版年制
```

*zepinglee AT gmail.com

注意，版本 v2.0 更改了设置参考文献表样式的方法，要求直接使用 `\bibliographystyle`，不再使用宏包的参数，而且更改了 `bst` 的文件名。

顺序编码制的引用标注默认使用角标式，如“张三^[2] 提出”。如果要使用正文模式，如“文献 [3] 中说明”，可以使用 `\citestyle` 命令进行切换：

```
\citestyle{numbers}
```

同一处引用多篇文献时，应当将各篇文献的 `key` 一同写在 `\cite` 命令中。如遇连续编号，默认会自动转为起讫序号并用短横线连接（见 `natbib` 的 `compress` 选项）。如果要对引用的编号进行自动排序，需要在调用 `gbt7714` 时加 `sort&compress` 参数：

```
\usepackage[sort&compress]{gbt7714}
```

这些参数会传给 `natbib` 处理。

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]。

使用时需要注意以下几点：

- `.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。

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）。

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

translator* 译者

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

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 Bib_T_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 Bib_T_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},
  ...
}
```

注意名字之间需要额外的空格，比如“张三,李四”要排在“张三丰”前面。

6 自定义样式

Bib_T_EX 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
uppercase.name	#1	将著者姓名转为大写
max.num.authors	#3	输出著者的最多数
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
show.medium.type	#1	显示载体类型标识
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

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-caspervector^[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 Bib_T_EX 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] MITTELBAACH 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

兼容过时的接口

```
1 <*package>
2 \newif\ifgbt@legacy@interface
3 \newif\ifgbt@mmxv
4 \newif\ifgbt@numerical
5 \newif\ifgbt@super
6 \newcommand\gbt@obsolete@option[1]{%
7   \PackageWarning{gbt7714}{The option "#1" is obsolete}%
8 }
9 \DeclareOption{authoryear}{}
10
11 \DeclareOption{2015}{%
12   \gbt@obsolete@option{2015}%
13   \gbt@legacy@interfacetrue
14   \gbt@mmxvtrue
15 }
16 \DeclareOption{2005}{%
17   \gbt@obsolete@option{2005}%
18   \gbt@legacy@interfacetrue
19   \gbt@mmxvfalse
20 }
21 \DeclareOption{super}{%
22   \gbt@obsolete@option{super}%
23   \gbt@legacy@interfacetrue
24   \gbt@numericaltrue
25   \gbt@supertrue
26 }
27 \DeclareOption{numbers}{%
28   \gbt@obsolete@option{numbers}%
29   \gbt@legacy@interfacetrue
30   \gbt@numericaltrue
31   \gbt@superfalse
32 }
33 \DeclareOption{authoryear}{%
34   \gbt@obsolete@option{authoryear}%
35   \gbt@legacy@interfacetrue
36   \gbt@numericalfalse
37 }

将选项传递给 natbib
38 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{natbib}}
```

```
39 \ProcessOptions\relax
```

调用宏包，注意只需要 `compress` 不需要 `sort`。

```
40 \RequirePackage[compress]{natbib}
```

```
41 \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`。

```
42 \newcommand\bibstyle@super{\bibpunct{[ ]}{,}{s}{,}{\textsuperscript{,}}}
```

```
43 \newcommand\bibstyle@numbers{\bibpunct{[ ]}{,}{n}{,}{,}{,}}
```

```
44 \newcommand\bibstyle@authoryear{\bibpunct{({})}{;}{a}{,}{,}{,}}
```

```
45 \newcommand\bibstyle@inline{\bibstyle@numbers}
```

在使用 `\bibliographystyle` 时自动切换引用文献的标注的样式。

```
46 \@namedef{bibstyle@gbt7714-numerical}{\bibstyle@super}
```

```
47 \@namedef{bibstyle@gbt7714-author-year}{\bibstyle@authoryear}
```

```
48 \@namedef{bibstyle@gbt7714-2005-numerical}{\bibstyle@super}
```

```
49 \@namedef{bibstyle@gbt7714-2005-author-year}{\bibstyle@authoryear}
```

`\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：

```
50 \def\NAT@citenum[#1][#2]#3{%
```

```
51 \NAT@reset@parser
```

```
52 \NAT@sort@cites{#3}%
```

```
53 \NAT@reset@citea
```

```
54 \@cite{\def\NAT@num{-1}\let\NAT@last@yr\relax\let\NAT@nm\@empty
```

```
55 \@for\@citeb:=\NAT@cite@list\do
```

```
56 {\@safe@activestrue
```

```
57 \edef\@citeb{\expandafter\@firstofone\@citeb\@empty}%
```

```
58 \@safe@activesfalse
```

```
59 \@ifundefined{b@\@citeb\@extra@b@citeb}{%
```

```
60 {\reset@font\bfseries?}
```

```
61 \NAT@citeundefined\PackageWarning{natbib}%
```

```
62 {Citation '\@citeb' on page \thepage \space undefined}}%
```

```
63 {\let\NAT@last@num\NAT@num\let\NAT@last@nm\NAT@nm
```

```
64 \NAT@parse{\@citeb}%
```

```
65 \ifNAT@longnames\@ifundefined{bv@\@citeb\@extra@b@citeb}{%
```

```
66 \let\NAT@name=\NAT@all@names
```

```
67 \global\@namedef{bv@\@citeb\@extra@b@citeb}{}}{}}%
```

```
68 \fi
```

```
69 \ifNAT@full\let\NAT@nm\NAT@all@names\else
```

```
70 \let\NAT@nm\NAT@name\fi
```



```

71 \ifNAT@swa
72 \@ifnum{\NAT@ctype>\@ne}{%
73 \@citea
74 \NAT@hyper@{\@ifnum{\NAT@ctype=\tw@}{\NAT@test{\NAT@ctype}}{\NAT@alias}}%
75 }{%
76 \@ifnum{\NAT@cmprs>\z@}{%
77 \NAT@ifcat@num\NAT@num
78 {\let\NAT@nm=\NAT@num}%
79 {\def\NAT@nm{-2}}%
80 \NAT@ifcat@num\NAT@last@num
81 {\@tempcnta=\NAT@last@num\relax}%
82 {\@tempcnta\m@ne}%
83 \@ifnum{\NAT@nm=\@tempcnta}{%
84 \@ifnum{\NAT@merge>\@ne}{\NAT@last@yr@mbbox}%
85 }{%
86 \advance\@tempcnta by\@ne
87 \@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。

```

88 % \ifx\NAT@last@yr\relax
89 % \def@NAT@last@yr{\@citea}%
90 % \else
91 % \def@NAT@last@yr{--\NAT@penalty}%
92 % \fi
93 \def@NAT@last@yr{-\NAT@penalty}%
94 }{%
95 \NAT@last@yr@mbbox
96 }%
97 }%
98 }{%
99 \@tempwattrue
100 \@ifnum{\NAT@merge>\@ne}{\@ifnum{\NAT@last@num=\NAT@num\relax}{\@tempwafalse}{}}{%
101 \if@tempswa\NAT@citea@mbbox\fi
102 }%
103 }%
104 \NAT@def@citea
105 \else
106 \ifcase\NAT@ctype
107 \ifx\NAT@last@nm\NAT@nm \NAT@yrsep\NAT@penalty\NAT@space\else
108 \@citea \NAT@test{\@ne}\NAT@spacechar\NAT@mbbox{\NAT@super@kern\NAT@@open}%
109 \fi
110 \if*#1*\else#1\NAT@spacechar\fi

```

```

111      \NAT@mbox{\NAT@hyper@{\citenumfont{\NAT@num}}}%
112      \NAT@def@citea@box
113      \or
114      \NAT@hyper@citea@space{\NAT@test{\NAT@ctype}}%
115      \or
116      \NAT@hyper@citea@space{\NAT@test{\NAT@ctype}}%
117      \or
118      \NAT@hyper@citea@space\NAT@alias
119      \fi
120    \fi
121  }%
122 }%
123 \ifnum{\NAT@cmprs>\z@}{\NAT@last@yr}{}%
124 \ifNAT@swa\else

```

将页码放在括号外边，并且置于上标。

```

125   % \ifnum{\NAT@ctype=\z@}{%
126   %   \if*#2*\else\NAT@cmt#2\fi
127   % }{%
128   \NAT@mbox{\NAT@close}%
129   \ifnum{\NAT@ctype=\z@}{%
130     \if*#2*\else\textsuperscript{#2}\fi
131   }{%
132     \fi
133   }{#1}{#2}%
134 }%

```

Numerical 模式的 \citep 的页码:

```

135 \renewcommand\NAT@citesuper[3]{\ifNAT@swa
136   \if*#2*\else#2\NAT@spacechar\fi
137 \unskip\kern\p@\textsuperscript{\NAT@open#1\NAT@close\if*#3*\else#3\fi}%
138   \else #1\fi\endgroup}

```

Author-year 模式的 \citete 的页码:

```

139 \def\NAT@citex%
140   [#1][#2]#3{%
141   \NAT@reset@parser
142   \NAT@sort@cites{#3}%
143   \NAT@reset@citea
144   \@cite{\let\NAT@nm\@empty\let\NAT@year\@empty
145     \@for\@citeb:=\NAT@cite@list\do
146     {\@safe@activestrue
147       \edef\@citeb{\expandafter\@firstofone\@citeb\@empty}%
148       \@safe@activesfalse

```

```

149 \ifundefined{b@\@citeb\@extra@b@citeb}{\@citea%
150 {\reset@font\bfseries ?}\NAT@citeundefined
151 \PackageWarning{natbib}%
152 {Citation '\@citeb' on page \thepage \space undefined}\def\NAT@date{}}%
153 {\let\NAT@last@nm=\NAT@nm\let\NAT@last@yr=\NAT@year
154 \NAT@parse{\@citeb}%
155 \ifNAT@longnames\ifundefined{bv@\@citeb\@extra@b@citeb}{%
156 \let\NAT@name=\NAT@all@names
157 \global\@namedef{bv@\@citeb\@extra@b@citeb}{}}}%
158 \fi
159 \ifNAT@full\let\NAT@nm\NAT@all@names\else
160 \let\NAT@nm\NAT@name\fi
161 \ifNAT@swa\ifcase\NAT@ctype
162 \if\relax\NAT@date\relax
163 \@citea\NAT@hyper@{\NAT@nmfmt{\NAT@nm}\NAT@date}%
164 \else
165 \ifx\NAT@last@nm\NAT@nm\NAT@yrsep
166 \ifx\NAT@last@yr\NAT@year
167 \def\NAT@temp{?}%
168 \ifx\NAT@temp\NAT@exlab\PackageWarningNoLine{natbib}%
169 {Multiple citation on page \thepage: same authors and
170 year\MessageBreak without distinguishing extra
171 letter,\MessageBreak appears as question mark}\fi
172 \NAT@hyper@{\NAT@exlab}%
173 \else\unskip\NAT@spacechar
174 \NAT@hyper@{\NAT@date}%
175 \fi
176 \else
177 \@citea\NAT@hyper@{%
178 \NAT@nmfmt{\NAT@nm}%
179 \hyper@natlinkbreak{%
180 \NAT@aysep\NAT@spacechar}{\@citeb\@extra@b@citeb
181 }%
182 \NAT@date
183 }%
184 \fi
185 \fi
186 \or\@citea\NAT@hyper@{\NAT@nmfmt{\NAT@nm}}%
187 \or\@citea\NAT@hyper@{\NAT@date}%
188 \or\@citea\NAT@hyper@{\NAT@alias}%
189 \fi \NAT@def@citea
190 \else

```

```

191 \ifcase\NAT@ctype
192 \if\relax\NAT@date\relax
193 \@citea\NAT@hyper@{\NAT@nmfmt{\NAT@nm}}}%
194 \else
195 \ifx\NAT@last@nm\NAT@nm\NAT@yrsep
196 \ifx\NAT@last@yr\NAT@year
197 \def\NAT@temp{{?}}}%
198 \ifx\NAT@temp\NAT@exlab\PackageWarningNoLine{natbib}%
199 {Multiple citation on page \thepage: same authors and
200 year\MessageBreak without distinguishing extra
201 letter,\MessageBreak appears as question mark}\fi
202 \NAT@hyper@{\NAT@exlab}%
203 \else
204 \unskip\NAT@spacechar
205 \NAT@hyper@{\NAT@date}%
206 \fi
207 \else
208 \@citea\NAT@hyper@{%
209 \NAT@nmfmt{\NAT@nm}}%
210 \hyper@natlinkbreak{\NAT@spacechar\NAT@@open\if*#1*\else#1\NAT@spacechar\fi}%
211 {\@citeb\@extra@b@citeb}%
212 \NAT@date
213 }%
214 \fi
215 \fi
216 \or\@citea\NAT@hyper@{\NAT@nmfmt{\NAT@nm}}}%
217 \or\@citea\NAT@hyper@{\NAT@date}%
218 \or\@citea\NAT@hyper@{\NAT@alias}%
219 \fi
220 \if\relax\NAT@date\relax
221 \NAT@def@citea
222 \else
223 \NAT@def@citea@close
224 \fi
225 \fi
226 }}\ifNAT@swa\else
227 % \if*#2*\else\NAT@cmt#2\fi
228 \if\relax\NAT@date\relax\else\NAT@@close\fi
229 \if*#2*\else\textsuperscript{#2}\fi
230 \fi}{#1}{#2}}

```

将页码放在括号外边，并且置于上标。

Author-year 模式的 \citep 的页码:

```

231 \renewcommand\NAT@cite%
232   [3]{\ifNAT@swa\NAT@@open\if*#2*\else#2\NAT@spacechar\fi
233        #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

```
234 \renewcommand\@biblabel[1]{[#1]\hfill}
```

`\url` 使用 `xurl` 宏包的方法，增加 URL 可断行的位置。

```

235 \g@addto@macro\UrlBreaks{%
236   \do0\do1\do2\do3\do4\do5\do6\do7\do8\do9%
237   \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
238   \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
239   \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
240   \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
241 }
242 \Urlmuskip=0mu plus 0.1mu

```

兼容 v2.0 前过时的接口:

```

243 \newif\ifgbt@bib@style@written
244 \@ifpackageloaded{chapterbib}{\}%
245   \def\bibliography#1{%
246     \ifgbt@bib@style@written\else
247       \bibliographystyle{gbt7714-numerical}%
248     \fi
249     \if@filesw
250       \immediate\write\@auxout{\string\bibdata{\zap@space#1 \@empty}}%
251     \fi
252     \@input@{\jobname.bbl}}
253 \def\bibliographystyle#1{%
254   \gbt@bib@style@writtentruer
255   \ifx\@begindocumenthook\undefined\else
256     \expandafter\AtBeginDocument
257   \fi
258   {\if@filesw
259     \immediate\write\@auxout{\string\bibstyle{#1}}%
260   \fi}%
261 }%
262 }
263 \ifgbt@legacy@interface
264   \ifgbt@numerical
265     \ifgbt@super\else
266       \citestyle{numbers}

```

```

267   \fi
268   \bibliographystyle{gbt7714-numerical}
269 \else
270   \bibliographystyle{gbt7714-author-year}
271 \fi
272 \fi
273 \end{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。

```

274 (*authoryear | numerical)
275 INTEGERS {
276   uppercase.name
277   max.num.authors
278   period.between.author.year
279   sentence.case.title
280   link.title
281   show.mark
282   show.medium.type
283   slash.for.extraction
284   in.booktitle
285   italic.journal
286   bold.journal.volume
287   show.missing.address.publisher
288   show.url
289   show.doi
290   show.note
291 (*authoryear)
292   lang.zh.order
293   lang.ja.order
294   lang.en.order
295   lang.ru.order
296   lang.other.order
297 \authoryear}
298 }
299

```

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。

```

300 FUNCTION {load.config}
301 {

```

英文姓名转为全大写：

```

302 (*!nouppercase&!thu)
303 #1 'uppercase.name :=

```

```

304 </!nuppercase&!thu>
305 < *nuppercase | thu>
306 #0 'uppercase.name :=
307 </nuppercase | thu>

```

最多显示的作者数量:

```

308 #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:

```

309 < *authoryear>
310 < *!period&!2005&!ustc>
311 #0 'period.between.author.year :=
312 </!period&!2005&!ustc>
313 < *period | 2005 | ustc>
314 #1 'period.between.author.year :=
315 </period | 2005 | ustc>
316 </authoryear>

```

英文标题转为 sentence case (句首字母大写, 其余小写):

```

317 #1 'sentence.case.title :=
318 < *nosentencecase>
319 #0 'sentence.case.title :=
320 </nosentencecase>

```

在标题添加超链接:

```

321 #0 'link.title :=
322 < *linktitle>
323 #1 'link.title :=
324 </linktitle>

```

著录文献类型标识 (比如“[M/OL]”):

```

325 #1 'show.mark :=
326 < *nomark>
327 #0 'show.mark :=
328 </nomark>

```

是否显示载体类型标识 (比如“/OL”):

```

329 #1 'show.medium.type :=
330 < *no.medium.type>
331 #0 'show.medium.type :=
332 </no.medium.type>

```

使用“//”表示析出文献

```

333 #1 'slash.for.extraction :=
334 < *noslash>
335 #0 'slash.for.extraction :=
336 </noslash>

```

使用“In:”表示析出文献

```

337 #0 'in.booktitle :=

```

期刊名使用斜体:

```

338 #0 'italic.journal :=
339 < *italicjournal>

```

```

340 #1 'italic.journal :=
341 </italicjournal>

```

期刊的卷使用粗体:

```

342 #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.”:

```

343 <!*noslsn&!thu&!ustc>
344 #1 'show.missing.address.publisher :=
345 </!noslsn&!thu&!ustc>
346 <!*noslsn | thu | ustc>
347 #0 'show.missing.address.publisher :=
348 </noslsn | thu | ustc>

```

是否著录 URL:

```

349 #1 'show.url :=
350 <*nourl>
351 #0 'show.url :=
352 </nourl>

```

是否著录 DOI:

```

353 <!*nodoi&!2005>
354 #1 'show.doi :=
355 </!nodoi&!2005>
356 <!*nodoi | 2005>
357 #0 'show.doi :=
358 </nodoi | 2005>

```

在每一条文献最后输出注释 (note) 的内容:

```

359 #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:

```

360 <*authoryear>
361 #1 'lang.zh.order :=
362 #2 'lang.ja.order :=
363 #3 'lang.en.order :=
364 #4 'lang.ru.order :=
365 #5 'lang.other.order :=
366 </authoryear>
367 }
368

```

B.2 The ENTRY declaration

Like Scribe’s (according to pages 231-2 of the April ’84 edition), but no full-author 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.

```

369 ENTRY
370 { address

```



```

371     author
372     booktitle
373     date
374     doi
375     edition
376     editor
377     howpublished
378     institution
379     journal
380     key
381     language
382     mark
383     medium
384     note
385     number
386     organization
387     pages
388     publisher
389     school
390     series
391     title
392     translator
393     url
394     urldate
395     volume
396     year
397 }
398 { 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.

```

399 { label extra.label sort.label short.list entry.mark entry.url }
400

```

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

```

```

401 INTEGERS { output.state before.all mid.sentence after.sentence after.block after.slash }
402
403 INTEGERS { lang.zh lang.ja lang.en lang.ru lang.other }
404
405 INTEGERS { charptr len }
406
407 FUNCTION {init.state.consts}
408 { #0 'before.all :=
409   #1 'mid.sentence :=
410   #2 'after.sentence :=
411   #3 'after.block :=
412   #4 'after.slash :=
413   #3 'lang.zh :=
414   #4 'lang.ja :=
415   #1 'lang.en :=
416   #2 'lang.ru :=
417   #0 'lang.other :=
418 }
419

```

下面是一些常量的定义

```

420 FUNCTION {bbl.anonymous}
421 { entry.lang lang.zh =
422   { " 佚名" }
423   { "Anon" }
424   if$
425 }
426
427 FUNCTION {bbl.space}
428 { entry.lang lang.zh =
429   { "\ " }
430   { " " }
431   if$
432 }
433
434 FUNCTION {bbl.et.al}
435 { entry.lang lang.zh =
436   { " 等" }
437   { entry.lang lang.ja =
438     { " 他" }
439     { entry.lang lang.ru =

```

```

440         { "идр" }
441         { "et~al." }
442     if$
443     }
444     if$
445     }
446     if$
447 }
448
449 FUNCTION {citation.et.al}
450 { bbl.et.al }
451
452 FUNCTION {bbl.colon} { ": " }
453
454 < *2015 >
455 FUNCTION {bbl.wide.space} { "\quad " }
456 < /2015 >
457 < *2005 >
458 FUNCTION {bbl.wide.space} { "\ " }
459 < /2005 >
460
461 < *!thu >
462 FUNCTION {bbl.slash} { "//\allowbreak " }
463 < /!thu >
464 < *thu >
465 FUNCTION {bbl.slash} { " //" }
466 < /thu >
467
468 FUNCTION {bbl.sine.loco}
469 { entry.lang lang.zh =
470   { "[出版地不详]" }
471   { "[S.l.]" }
472   if$
473 }
474
475 FUNCTION {bbl.sine.nomine}
476 { entry.lang lang.zh =
477   { "[出版者不详]" }
478   { "[s.n.]" }
479   if$
480 }
481
482 FUNCTION {bbl.sine.loco.sine.nomine}
483 { entry.lang lang.zh =
484   { "[出版地不详: 出版者不详]" }
485   { "[S.l.: s.n.]" }
486   if$
487 }
488

```

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

```

489 FUNCTION {not}

```

```

490 { { #0 }
491   { #1 }
492   if$
493 }
494
495 FUNCTION {and}
496 {   'skip$
497   { pop$ #0 }
498   if$
499 }
500
501 FUNCTION {or}
502 {   { pop$ #1 }
503     'skip$
504   if$
505 }
506

```

the variables s and t are temporary string holders

```

507 STRINGS { s t }
508
509 FUNCTION {output.nonnull}
510 { 's :=
511   output.state mid.sentence =
512   { ", " * write$ }
513   { output.state after.block =
514     { add.period$ write$
515       newline$
516       "\newblock " write$
517     }
518     { output.state before.all =
519       'write$
520       { output.state after.slash =
521         { bbl.slash * write$
522           newline$
523         }
524         { add.period$ " " * write$ }
525       if$
526     }
527     if$
528   }
529   if$
530   mid.sentence 'output.state :=
531 }
532 if$
533 s
534 }
535
536 FUNCTION {output}
537 { duplicate$ empty$
538   'pop$
539   'output.nonnull
540   if$
541 }
542

```

```

543 FUNCTION {output.after}
544 { 't :=
545   duplicate$ empty$
546   'pop$
547   { 's :=
548     output.state mid.sentence =
549     { t * write$ }
550     { output.state after.block =
551       { add.period$ write$
552         newline$
553         "\newblock " write$
554       }
555       { output.state before.all =
556         'write$
557         { output.state after.slash =
558           { bbl.slash * write$ }
559           { add.period$ " " * write$ }
560         if$
561       }
562       if$
563     }
564     if$
565     mid.sentence 'output.state :=
566   }
567   if$
568   s
569 }
570 if$
571 }
572
573 FUNCTION {output.check}
574 { 't :=
575   duplicate$ empty$
576   { pop$ "empty " t * " in " * cite$ * warning$ }
577   'output.nonnull
578   if$
579 }
580

```

This function finishes all entries.

```

581 FUNCTION {fin.entry}
582 { add.period$
583   write$
584   newline$
585 }
586
587 FUNCTION {new.block}
588 { output.state before.all =
589   'skip$
590   { output.state after.slash =
591     'skip$
592     { after.block 'output.state := }
593   if$
594 }
595 if$

```

```

596 }
597
598 FUNCTION {new.sentence}
599 { output.state after.block =
600   'skip$
601   { output.state before.all =
602     'skip$
603     { output.state after.slash =
604       'skip$
605       { after.sentence 'output.state := }
606       if$
607     }
608     if$
609   }
610   if$
611 }
612
613 FUNCTION {new.slash}
614 { output.state before.all =
615   'skip$
616   { slash.for.extraction
617     { after.slash 'output.state := }
618     { after.block 'output.state := }
619     if$
620   }
621   if$
622 }
623

```

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.

```

624 FUNCTION {new.block.checka}
625 { empty$
626   'skip$
627   'new.block
628   if$
629 }
630
631 FUNCTION {new.block.checkb}
632 { empty$
633   swap$ empty$
634   and
635   'skip$
636   'new.block
637   if$
638 }
639

```

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

```

640 FUNCTION {new.sentence.checka}
641 { empty$
642   'skip$
643   'new.sentence

```



```

644  if$
645 }
646
647 FUNCTION {new.sentence.checkb}
648 { empty$
649   swap$ empty$
650   and
651   'skip$
652   'new.sentence
653   if$
654 }
655

```

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

```

656 FUNCTION {field.or.null}
657 { duplicate$ empty$
658   { pop$ "" }
659   'skip$
660   if$
661 }

```

```

662
663 FUNCTION {italicize}
664 { duplicate$ empty$
665   { pop$ "" }
666   { "\textit{" swap$ * "}" * }
667   if$
668 }
669

```

B.4.1 Detect Language

```

670 INTEGERS { byte second.byte }
671
672 INTEGERS { char.lang tmp.lang }
673
674 STRINGS { tmp.str }
675
676 FUNCTION {get.str.lang}
677 { 'tmp.str :=
678   lang.other 'tmp.lang :=
679   #1 'charptr :=
680   tmp.str text.length$ #1 + 'len :=
681   { charptr len < }
682   { tmp.str charptr #1 substring$ chr.to.int$ 'byte :=
683     byte #128 <
684     { charptr #1 + 'charptr :=
685       byte #64 > byte #91 < and byte #96 > byte #123 < and or
686       { lang.en 'char.lang := }
687       { lang.other 'char.lang := }
688       if$
689     }
690     { tmp.str charptr #1 + #1 substring$ chr.to.int$ 'second.byte :=
691       byte #224 <

```

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

```

692   { charptr #2 + 'charptr :=
693     byte #207 > byte #212 < and
694     byte #212 = second.byte #176 < and or
695     { lang.ru 'char.lang := }
696     { lang.other 'char.lang := }
697     if$
698   }
699   { byte #240 <

```

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

```

700   { charptr #3 + 'charptr :=
701     byte #227 > byte #234 < and
702     { lang.zh 'char.lang := }

```

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

```

703   { byte #227 =
704     { second.byte #143 >
705       { lang.zh 'char.lang := }

```

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

```
706             { second.byte #128 > second.byte #132 < and
707                 { lang.ja 'char.lang := }
708                 { lang.other 'char.lang := }
709             if$
710         }
711     if$
712 }
```

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

```
713         { byte #239 =
714             second.byte #163 > second.byte #172 < and and
715             { lang.zh 'char.lang := }
716             { lang.other 'char.lang := }
717         if$
718     }
719     if$
720 }
721 if$
722 }
```

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.

```
723         { charptr #4 + 'charptr :=
724             byte #240 = second.byte #159 > and
725             { lang.zh 'char.lang := }
726             { lang.other 'char.lang := }
727         if$
728     }
729     if$
730 }
731 if$
732 }
733 if$
734 char.lang tmp.lang >
735     { char.lang 'tmp.lang := }
736     'skip$
737 if$
738 }
739 while$
740 tmp.lang
741 }
742
743 FUNCTION {check.entry.lang}
744 { author field.or.null
745   title field.or.null *
746   get.str.lang
747 }
748
749 FUNCTION {set.entry.lang}
750 { language empty$
751   { check.entry.lang }
752   { language "english" = language "american" = or language "british" = or
```

```

753     { lang.en }
754     { language "chinese" =
755         { lang.zh }
756         { language "japanese" =
757             { lang.ja }
758             { language "russian" =
759                 { lang.ru }
760                 { check.entry.lang }
761             if$
762         }
763     if$
764 }
765 if$
766 }
767 if$
768 }
769 if$
770 'entry.lang :=
771 }
772
773 FUNCTION {set.entry.numbered}
774 { type$ "patent" =
775   type$ "standard" = or
776   type$ "techreport" = or
777   { #1 'entry.numbered := }
778   { #0 'entry.numbered := }
779   if$
780 }
781

```

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: nameresult: 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 nameresult := nameresult * ", " * t
            else if numnames > 2

```

```

        then nameresult := nameresult * ","
      fi
      if t = "others"
        then nameresult := nameresult * " et~al."
        else nameresult := nameresult * " and " * t
      fi
    fi
  else nameresult := t
  fi
  nameptr := nameptr + 1
  namesleft := namesleft - 1
od
return nameresult
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.

```

782 INTEGERS { nameptr namesleft numnames name.lang }
783
784 FUNCTION {format.names}
785 { 's :=
786   #1 'nameptr :=
787   s num.names$ 'numnames :=
788   numnames 'namesleft :=
789   { namesleft #0 > }
790   { s nameptr "{vv~}{ll}{, jj}{, ff}" format.name$ 't :=
791     nameptr max.num.authors >
792     { bbl.et.al

```

```

793         #1 'namesleft :=
794     }
795     { t "others" =
796         { bbl.et.al }
797         { t get.str.lang 'name.lang :=
798             name.lang lang.en =
799             { t #1 "{vv~}{ll}{~f{~}}" format.name$
800                 uppercase.name
801                 { "u" change.case$ }
802                 'skip$
803                 if$
804                 t #1 "{, jj}" format.name$ *
805             }
806             { t #1 "{ll}{ff}" format.name$ }
807         if$
808     }
809     if$
810 }
811 if$
812 nameptr #1 >
813 { ", " swap$ * * }
814 'skip$
815 if$
816 nameptr #1 + 'nameptr :=
817 namesleft #1 - 'namesleft :=
818 }
819 while$
820 }
821
822 FUNCTION {format.key}
823 { empty$
824     { key field.or.null }
825     { "" }
826     if$
827 }
828
829 FUNCTION {format.authors}
830 { author empty$ not
831     { author format.names }
832     { "empty author in " cite$ * warning$
833     (*authoryear)
834         bbl.anonymous
835     (/authoryear)
836     (*numerical)
837         ""
838     (/numerical)
839     }
840     if$
841 }
842
843 FUNCTION {format.editors}
844 { editor empty$
845     { "" }
846     { editor format.names }
847     if$

```

```

848 }
849
850 FUNCTION {format.translators}
851 { translator empty$
852   { "" }
853   { translator format.names
854     entry.lang lang.zh =
855     { translator num.names$ #3 >
856       { " 译" * }
857       { ", 译" * }
858       if$
859     }
860     'skip$
861   if$
862 }
863 if$
864 }
865
866 FUNCTION {format.full.names}
867 { 's :=
868   #1 'nameptr :=
869   s num.names$ 'numnames :=
870   numnames 'namesleft :=
871   { namesleft #0 > }
872   { s nameptr "{vv~}{ll}{, jj}{, ff}" format.name$ 't :=
873     t get.str.lang 'name.lang :=
874     name.lang lang.en =
875     { t #1 "{vv~}{ll}" format.name$ 't := }
876     { t #1 "{ll}{ff}" format.name$ 't := }
877   if$
878   nameptr #1 >
879   {
880     namesleft #1 >
881     { ", " * t * }
882     {
883       numnames #2 >
884       { ", " * }
885       'skip$
886       if$
887       t "others" =
888       { " et~al." * }
889       { " and " * t * }
890       if$
891     }
892     if$
893   }
894   't
895   if$
896   nameptr #1 + 'nameptr :=
897   namesleft #1 - 'namesleft :=
898 }
899 while$
900 }
901
902 FUNCTION {author.editor.full}

```

```

903 { author empty$
904   { editor empty$
905     { "" }
906     { editor format.full.names }
907   if$
908   }
909   { author format.full.names }
910 if$
911 }
912
913 FUNCTION {author.full}
914 { author empty$
915   { "" }
916   { author format.full.names }
917 if$
918 }
919
920 FUNCTION {editor.full}
921 { editor empty$
922   { "" }
923   { editor format.full.names }
924 if$
925 }
926
927 FUNCTION {make.full.names}
928 { type$ "book" =
929   type$ "inbook" =
930   or
931   'author.editor.full
932   { type$ "collection" =
933     type$ "proceedings" =
934     or
935     'editor.full
936     'author.full
937   if$
938   }
939 if$
940 }
941
942 FUNCTION {output.bibitem}
943 { newline$
944   "\bibitem[" write$
945   label ")" *
946   make.full.names duplicates$ short.list =
947   { pop$ }
948   { * }
949 if$
950 's :=
951 s text.length$ 'charptr :=
952 { charptr #0 > s charptr #1 substring$ "[" = not and }
953 { charptr #1 - 'charptr := }
954 while$
955 charptr #0 >
956 { "{" s * "}" * }
957 { s }

```



```

958 if$
959 "]"{" * write$
960 cite$ write$
961 "}" write$
962 newline$
963 ""
964 before.all 'output.state :=
965 }
966

```

B.4.3 Format title

The `format.title` function is used for non-book-like titles. For most styles we convert to lowercase (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.

```

967 FUNCTION {change.sentence.case}
968 { entry.lang lang.en =
969   { "t" change.case$ }
970   'skip$
971   if$
972 }
973
974 FUNCTION {add.link}
975 { url empty$ not
976   { "\href{" url * "{" * swap$ * "}" * }
977   { doi empty$ not
978     { "\href{http://dx.doi.org/" doi * "{" * swap$ * "}" * }
979     'skip$
980     if$
981   }
982   if$
983 }
984
985 FUNCTION {format.title}
986 { title empty$
987   { "" }
988   { title
989     sentence.case.title
990     'change.sentence.case
991     'skip$
992     if$
993     entry.numbered number empty$ not and
994     { bbl.colon * number * }
995     'skip$
996     if$
997     link.title
998     'add.link
999     'skip$
1000    if$
1001   }
1002   if$

```

```
1003 }
1004
```

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
```

```
1005 FUNCTION {tie.or.space.connect}
1006 { duplicate$ text.length$ #3 <
1007   { "~" }
1008   { " " }
1009   if$
1010   swap$ * *
1011 }
1012
```

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
```

```
1013 FUNCTION {either.or.check}
1014 { empty$
1015   'pop$
1016   { "can't use both " swap$ * " fields in " * cite$ * warning$ }
1017   if$
1018 }
1019
```

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.

```

1020 FUNCTION {is.digit}
1021 { duplicate$ empty$
1022   { pop$ #0 }
1023   { chr.to.int$
1024     duplicate$ "0" chr.to.int$ <
1025     { pop$ #0 }
1026     { "9" chr.to.int$ >
1027       { #0 }
1028       { #1 }
1029     if$
1030   }
1031   if$
1032 }
1033 if$
1034 }
1035
1036 FUNCTION {is.number}
1037 { 's :=
1038   s empty$
1039   { #0 }
1040   { s text.length$ 'charptr :=
1041     { charptr #0 >
1042       s charptr #1 substring$ is.digit
1043       and
1044     }
1045     { charptr #1 - 'charptr := }
1046     while$
1047     charptr not
1048   }
1049   if$
1050 }
1051
1052 FUNCTION {format.volume}
1053 { volume empty$ not
1054   { volume is.number
1055     { entry.lang lang.zh =
1056       { " 第 " volume * " 卷" * }
1057       { "volume" volume tie.or.space.connect }
1058     if$
1059   }
1060   { volume }
1061   if$
1062 }
1063 { "" }
1064 if$
1065 }
1066

```

```

1067 FUNCTION {format.number}
1068 { number empty$ not
1069   { number is.number
1070     { entry.lang lang.zh =
1071       { " 第 " number * " 册" * }
1072       { "number" number tie.or.space.connect }
1073       if$
1074     }
1075     { number }
1076     if$
1077   }
1078   { "" }
1079   if$
1080 }
1081
1082 FUNCTION {format.volume.number}
1083 { volume empty$ not
1084   { format.volume }
1085   { format.number }
1086   if$
1087 }
1088
1089 FUNCTION {format.title.vol.num}
1090 { title
1091   sentence.case.title
1092   'change.sentence.case
1093   'skip$
1094   if$
1095   entry.numbered
1096   { number empty$ not
1097     { bbl.colon * number * }
1098     'skip$
1099     if$
1100   }
1101   { format.volume.number 's :=
1102     s empty$ not
1103     { bbl.colon * s * }
1104     'skip$
1105     if$
1106   }
1107   if$
1108 }
1109
1110 FUNCTION {format.series.vol.num.title}
1111 { format.volume.number 's :=
1112   series empty$ not
1113   { series
1114     sentence.case.title
1115     'change.sentence.case
1116     'skip$
1117     if$
1118     entry.numbered
1119     { bbl.wide.space * }
1120     { bbl.colon *
1121       s empty$ not

```

```

1122         { s * bbl.wide.space * }
1123         'skip$
1124     if$
1125     }
1126     if$
1127     title *
1128     sentence.case.title
1129     'change.sentence.case
1130     'skip$
1131     if$
1132     entry.numbered number empty$ not and
1133     { bbl.colon * number * }
1134     'skip$
1135     if$
1136     }
1137     { format.title.vol.num }
1138     if$
1139     link.title
1140     'add.link
1141     'skip$
1142     if$
1143 }
1144
1145 FUNCTION {format.booktitle.vol.num}
1146 { booktitle
1147   entry.numbered
1148   'skip$
1149   { format.volume.number 's :=
1150     s empty$ not
1151     { bbl.colon * s * }
1152     'skip$
1153     if$
1154   }
1155   if$
1156 }
1157
1158 FUNCTION {format.series.vol.num.booktitle}
1159 { format.volume.number 's :=
1160   series empty$ not
1161   { series bbl.colon *
1162     entry.numbered not s empty$ not and
1163     { s * bbl.wide.space * }
1164     'skip$
1165     if$
1166     booktitle *
1167   }
1168   { format.booktitle.vol.num }
1169   if$
1170   in.booktitle
1171   { duplicate$ empty$ not entry.lang lang.en = and
1172     { "In: " swap$ * }
1173     'skip$
1174     if$
1175   }
1176   'skip$

```

```

1177 if$
1178 }
1179
1180 FUNCTION {format.journal}
1181 { journal
1182   italic.journal entry.lang lang.en = and
1183   'italicize
1184   'skip$
1185   if$
1186 }
1187

```

B.4.4 Format entry type mark

```

1188 FUNCTION {set.entry.mark}
1189 { entry.mark empty$ not
1190   'pop$
1191   { mark empty$ not
1192     { pop$ mark 'entry.mark := }
1193     { 'entry.mark := }
1194     if$
1195   }
1196   if$
1197 }
1198
1199 FUNCTION {format.mark}
1200 { show.mark
1201   < *thu>
1202   type$ "phdthesis" = type$ "mastersthesis" = or type$ "patent" = or
1203   medium empty$ not or entry.is.electronic or
1204   and
1205   < /thu>
1206   { entry.mark
1207     show.medium.type
1208     { medium empty$ not
1209       { "/" * medium * }
1210       { entry.is.electronic
1211         { "/OL" * }
1212         'skip$
1213       }
1214     }
1215     if$
1216   }
1217   'skip$
1218   if$
1219   'entry.mark :=
1220   < *!thu>
1221   "\allowbreak[" entry.mark * "]" *
1222   < /!thu>
1223   < *thu>
1224   " [" entry.mark * "]" *
1225   < /thu>
1226   }
1227   { "" }
1228   if$

```

```
1229 }
1230
```

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.

```
1231 FUNCTION {num.to.ordinal}
1232 { duplicate$ text.length$ 'charptr :=
1233   duplicate$ charptr #1 substring$ 's :=
1234   s "1" =
1235     { "st" * }
1236   { s "2" =
1237     { "nd" * }
1238     { s "3" =
1239       { "rd" * }
1240       { "th" * }
1241       if$
1242     }
1243     if$
1244   }
1245   if$
1246 }
1247
1248 FUNCTION {format.edition}
1249 { edition empty$
1250   { "" }
1251   { edition is.number
1252     { entry.lang lang.zh =
1253       { edition " 版" * }
1254       { edition num.to.ordinal " ed." * }
1255       if$
1256     }
1257     { entry.lang lang.en =
1258       { edition change.sentence.case 's :=
1259         s "Revised" = s "Revised edition" = or
1260         { "Rev. ed." }
1261         { s " ed." *}
1262         if$
1263       }
1264       { edition }
1265       if$
1266     }
1267     if$
1268   }
1269   if$
1270 }
1271
```

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。

```

1272 FUNCTION {format.publisher}
1273 { publisher empty$ not
1274   { publisher }
1275   { school empty$ not
1276     { school }
1277     { organization empty$ not
1278       { organization }
1279       { institution empty$ not
1280         { institution }
1281         { "" }
1282       if$
1283     }
1284   if$
1285 }
1286 if$
1287 }
1288 if$
1289 }
1290
1291 FUNCTION {format.address.publisher}
1292 { address empty$ not
1293   { address
1294     format.publisher empty$ not
1295     { bbl.colon * format.publisher * }
1296     { entry.is.electronic not show.missing.address.publisher and
1297       { bbl.colon * bbl.sine.nomine * }
1298       'skip$
1299     if$
1300   }
1301   if$
1302 }
1303 { entry.is.electronic not show.missing.address.publisher and
1304   { format.publisher empty$ not
1305     { bbl.sine.loco bbl.colon * format.publisher * }
1306     { bbl.sine.loco.sine.nomine }
1307   if$
1308 }
1309 { format.publisher empty$ not
1310   { format.publisher }
1311   { "" }
1312 if$
1313 }
1314 if$
1315 }
1316 if$
1317 }
1318

```

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。

```

1319 FUNCTION {extract.before.dash}
1320 { duplicate$ empty$
1321   { pop$ "" }
1322   { 's :=
1323     #1 'charptr :=
1324     s text.length$ #1 + 'len :=
1325     { charptr len <
1326       s charptr #1 substring$ "-" = not
1327       and
1328     }
1329     { charptr #1 + 'charptr := }
1330     while$
1331     s #1 charptr #1 - substring$
1332   }
1333   if$
1334 }
1335
1336 FUNCTION {extract.after.dash}
1337 { duplicate$ empty$
1338   { pop$ "" }
1339   { 's :=
1340     #1 'charptr :=
1341     s text.length$ #1 + 'len :=
1342     { charptr len <
1343       s charptr #1 substring$ "-" = not
1344       and
1345     }
1346     { charptr #1 + 'charptr := }
1347     while$
1348     { charptr len <
1349       s charptr #1 substring$ "-" =
1350       and
1351     }
1352     { charptr #1 + 'charptr := }
1353     while$
1354     s charptr global.max$ substring$
1355   }
1356   if$
1357 }
1358
1359 FUNCTION {contains.dash}
1360 { duplicate$ empty$
1361   { pop$ #0 }
1362   { 's :=
1363     { s empty$ not
1364       s #1 #1 substring$ "-" = not
1365       and
1366     }
1367     { s #2 global.max$ substring$ 's := }
1368     while$
1369     s empty$ not
1370   }

```

```

1371 if$
1372 }
1373

```

著者-出版年制必须提取出年份

```

1374 FUNCTION {format.year}
1375 { year empty$ not
1376   { year extract.before.dash }
1377   { date empty$ not
1378     { date extract.before.dash }
1379     { "empty year in " cite$ * warning$
1380       urldate empty$ not
1381       { "[" urldate extract.before.dash * "]" * }
1382       { "" }
1383       if$
1384     }
1385     if$
1386   }
1387   if$
1388   extra.label *
1389 }
1390

```

专利和报纸都是使用日期而不是年

```

1391 FUNCTION {format.date}
1392 { type$ "patent" = type$ "newspaper" = or
1393   date empty$ not and
1394   { date }
1395   { year }
1396   if$
1397 }
1398

```

更新、修改日期只用于电子资源 **elctronic**

```

1399 FUNCTION {format.editdate}
1400 { date empty$ not
1401   { "\allowbreak(" date * ")" * }
1402   { "" }
1403   if$
1404 }
1405

```

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 **BibT_EX** 标准的域，但是实际中很常见。

```

1406 FUNCTION {format.urldate}
1407 { urldate empty$ not entry.is.electronic and
1408   { "\allowbreak[" urldate * "]" * }
1409   { "" }
1410   if$
1411 }
1412

```

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`。

```
1413 FUNCTION {hyphenate}
1414 { 't :=
1415   ""
1416   { t empty$ not }
1417   { t #1 #1 substring$ "-" =
1418     { "-" *
1419       { t #1 #1 substring$ "-" = }
1420       { t #2 global.max$ substring$ 't := }
1421       while$
1422     }
1423     { t #1 #1 substring$ *
```

```

1424         t #2 global.max$ substring$ 't :=
1425     }
1426     if$
1427 }
1428 while$
1429 }
1430

```

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

```

1431 FUNCTION {format.pages}
1432 { pages empty$
1433   { "" }
1434   { pages hyphenate }
1435   if$
1436 }
1437

```

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。

```

1438 FUNCTION {format.journal.volume}
1439 { volume empty$ not
1440   { bold.journal.volume
1441     { "\textbf{" volume * "}" * }
1442     { volume }
1443     if$
1444   }
1445   { "" }
1446   if$
1447 }
1448
1449 FUNCTION {format.journal.number}
1450 { number empty$ not
1451   { "\penalty0 (" number * ")" * }
1452   { "" }
1453   if$
1454 }
1455
1456 FUNCTION {format.journal.pages}
1457 { pages empty$
1458   { "" }
1459   { ":\penalty0 " pages hyphenate * }
1460   if$
1461 }
1462

```

连续出版物的年卷期有起止范围，需要特殊处理

```

1463 FUNCTION {format.periodical.year.volume.number}
1464 { year empty$ not
1465   { year extract.before.dash }
1466   { "empty year in periodical " cite$ * warning$ }

```

```

1467 if$
1468 volume empty$ not
1469 { ", " * volume extract.before.dash * }
1470 'skip$
1471 if$
1472 number empty$ not
1473 { "\penalty0 (" * number extract.before.dash * ")" * }
1474 'skip$
1475 if$
1476 year contains.dash
1477 { "--" *
1478   year extract.after.dash empty$
1479   volume extract.after.dash empty$ and
1480   number extract.after.dash empty$ and not
1481   { year extract.after.dash empty$ not
1482     { year extract.after.dash * }
1483     { year extract.before.dash * }
1484     if$
1485     volume empty$ not
1486     { ", " * volume extract.after.dash * }
1487     'skip$
1488     if$
1489     number empty$ not
1490     { "\penalty0 (" * number extract.after.dash * ")" * }
1491     'skip$
1492     if$
1493   }
1494   'skip$
1495   if$
1496 }
1497 'skip$
1498 if$
1499 }
1500

```

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。

```

1501 FUNCTION {check.url}
1502 { url empty$ not
1503   { "\url{" url * "}" * 'entry.url :=
1504     #1 'entry.is.electronic :=
1505   }
1506   { howpublished empty$ not
1507     { howpublished #1 #5 substring$ "\url{" =
1508       { howpublished 'entry.url :=
1509         #1 'entry.is.electronic :=
1510       }
1511       'skip$
1512     if$
1513   }
1514   { note empty$ not
1515     { note #1 #5 substring$ "\url{" =
1516       { note 'entry.url :=

```

```

1517             #1 'entry.is.electronic :=
1518             }
1519             'skip$
1520         if$
1521     }
1522     'skip$
1523     if$
1524 }
1525 if$
1526 }
1527 if$
1528 }
1529
1530 FUNCTION {format.url}
1531 { entry.url empty$ not
1532   { new.block entry.url }
1533   { "" }
1534   if$
1535 }
1536

```

需要检测 DOI 是否已经包含在 URL 中。

```

1537 FUNCTION {check.doi}
1538 { doi empty$ not
1539   { #1 'entry.is.electronic := }
1540   'skip$
1541   if$
1542 }
1543
1544 FUNCTION {is.in.url}
1545 { 's :=
1546   s empty$
1547   { #1 }
1548   { entry.url empty$
1549     { #0 }
1550     { s text.length$ 'len :=
1551       entry.url text.length$ 'charptr :=
1552       { entry.url charptr len substring$ s = not
1553         charptr #0 >
1554         and
1555       }
1556       { charptr #1 - 'charptr := }
1557       while$
1558       charptr
1559     }
1560     if$
1561   }
1562   if$
1563 }
1564
1565 FUNCTION {format.doi}
1566 { ""
1567   doi empty$ not show.doi and
1568   { "" 's :=
1569     doi 't :=

```

```

1570     #0 'numnames :=
1571     { t empty$ not}
1572     { t #1 #1 substring$ 'tmp.str :=
1573     tmp.str "," = tmp.str " " = or t #2 #1 substring$ empty$ or
1574     { t #2 #1 substring$ empty$
1575     { s tmp.str * 's := }
1576     'skip$
1577     if$
1578     s empty$ s is.in.url or
1579     'skip$
1580     { numnames #1 + 'numnames :=
1581     numnames #1 >
1582     { ", " * }
1583     { "DOI: " * }
1584     if$
1585     "\doi{" s * "}" * *
1586     }
1587     if$
1588     "" 's :=
1589     }
1590     { s tmp.str * 's := }
1591     if$
1592     t #2 global.max$ substring$ 't :=
1593     }
1594     while$
1595     's :=
1596     s empty$ not
1597     { new.block s }
1598     { "" }
1599     if$
1600     }
1601     'skip$
1602 if$
1603 }
1604
1605 FUNCTION {check.electronic}
1606 { "" 'entry.url :=
1607   #0 'entry.is.electronic :=
1608   'check.doi
1609   'skip$
1610   if$
1611   'check.url
1612   'skip$
1613   if$
1614   medium empty$ not
1615   { medium "MT" = medium "DK" = or medium "CD" = or medium "OL" = or
1616     { #1 'entry.is.electronic := }
1617     'skip$
1618     if$
1619     }
1620     'skip$
1621     if$
1622 }
1623
1624 FUNCTION {format.note}

```

```

1625 { note empty$ not show.note and
1626     { note }
1627     { "" }
1628   if$
1629 }
1630

```

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

```

1631 FUNCTION {empty.misc.check}
1632 { author empty$ title empty$
1633   year empty$
1634   and and
1635   key empty$ not and
1636   { "all relevant fields are empty in " cite$ * warning$ }
1637   'skip$
1638   if$
1639 }
1640

```

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

```

1641 FUNCTION {monograph}
1642 { output.bibitem
1643   author empty$ not
1644     { format.authors }
1645     { editor empty$ not
1646       { format.editors }
1647       { "empty author and editor in " cite$ * warning$
1648 (*authoryear)
1649   bbl.anonymous
1650 (/authoryear)
1651 (*numerical)
1652   ""
1653 (/numerical)
1654   }
1655   if$
1656   }
1657   if$
1658   output
1659 (*authoryear)
1660   period.between.author.year

```



```

1661     'new.sentence
1662     'skip$
1663   if$
1664     format.year "year" output.check
1665 </authoryear>
1666   new.block
1667   format.series.vol.num.title "title" output.check
1668   "M" set.entry.mark
1669   format.mark "" output.after
1670   new.block
1671   format.translators output
1672   new.sentence
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.5.2 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, chapter, pages, address, edition, month, note

```

1688 FUNCTION {incollection}
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   "M" set.entry.mark
1702   format.mark "" output.after
1703   new.block

```

```

1704 format.translators output
1705 new.slash
1706 format.editors output
1707 new.block
1708 format.series.vol.num.booktitle "booktitle" output.check
1709 new.block
1710 format.edition output
1711 new.block
1712 format.address.publisher output
1713 (*numerical)
1714 format.year "year" output.check
1715 (/numerical)
1716 format.pages bbl.colon output.after
1717 format.urldate "" output.after
1718 format.url output
1719 format.doi output
1720 new.block
1721 format.note output
1722 fin.entry
1723 }
1724

```

B.5.3 连续出版物

```

1725 FUNCTION {periodical}
1726 { output.bibitem
1727   format.authors output
1728   author format.key output
1729 (*authoryear)
1730   period.between.author.year
1731   'new.sentence
1732   'skip$
1733   if$
1734   format.year "year" output.check
1735 (/authoryear)
1736 new.block
1737 format.title "title" output.check
1738 "J" set.entry.mark
1739 format.mark "" output.after
1740 new.block
1741 format.periodical.year.volume.number output
1742 new.block
1743 format.address.publisher output
1744 (*numerical)
1745 format.date "year" output.check
1746 (/numerical)
1747 format.urldate "" output.after
1748 format.url output
1749 format.doi output
1750 new.block
1751 format.note output
1752 fin.entry
1753 }
1754

```

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.

```
1755 FUNCTION {article}
1756 { output.bibitem
1757   format.authors output
1758   author format.key output
1759 } (*authoryear)
1760   period.between.author.year
1761   'new.sentence
1762   'skip$
1763   if$
1764   format.year "year" output.check
1765 } (/authoryear)
1766   new.block
1767   format.title "title" output.check
1768   "J" set.entry.mark
1769   format.mark "" output.after
1770   new.block
1771   format.journal "journal" output.check
1772 } (*numerical)
1773   format.date "year" output.check
1774 } (/numerical)
1775   format.journal.volume output
1776   format.journal.number "" output.after
1777   format.journal.pages "" output.after
1778   format.urldate "" output.after
1779   format.url output
1780   format.doi output
1781   new.block
1782   format.note output
1783   fin.entry
1784 }
1785
```

B.5.5 专利文献

number 域也可以用来表示专利号。

```
1786 FUNCTION {patent}
1787 { output.bibitem
1788   format.authors output
1789   author format.key output
1790 } (*authoryear)
1791   period.between.author.year
1792   'new.sentence
1793   'skip$
```

```

1794  if$
1795  format.year "year" output.check
1796 </authoryear>
1797  new.block
1798  format.title "title" output.check
1799  "P" set.entry.mark
1800  format.mark "" output.after
1801  new.block
1802  format.date "year" output.check
1803  format.urldate "" output.after
1804  format.url output
1805  format.doi output
1806  new.block
1807  format.note output
1808  fin.entry
1809 }
1810

```

B.5.6 电子资源

```

1811 FUNCTION {electronic}
1812 { #1 #1 check.electronic
1813   #1 'entry.is.electronic :=
1814   output.bibitem
1815   format.authors output
1816   author format.key output
1817 (*authoryear)
1818   period.between.author.year
1819   'new.sentence
1820   'skip$
1821   if$
1822   format.year "year" output.check
1823 </authoryear>
1824   new.block
1825   format.series.vol.num.title "title" output.check
1826   "EB" set.entry.mark
1827   format.mark "" output.after
1828   new.block
1829   format.address.publisher output
1830 (*numerical)
1831   date empty$
1832   { format.date output }
1833   'skip$
1834   if$
1835 </numerical>
1836   format.pages bbl.colon output.after
1837   format.editdate "" output.after
1838   format.urldate "" output.after
1839   format.url output
1840   format.doi output
1841   new.block
1842   format.note output
1843   fin.entry
1844 }
1845

```

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。

```
1846 FUNCTION {misc}
1847 { journal empty$ not
1848   'article
1849   { booktitle empty$ not
1850     'incollection
1851     { publisher empty$ not
1852       'monograph
1853       { entry.is.electronic
1854         'electronic
1855         { "Z" set.entry.mark
1856           monograph
1857         }
1858       } if$
1859     }
1860   } if$
1861 }
1862 if$
1863 }
1864 if$
1865 empty.misc.check
1866 }
1867
1868 FUNCTION {archive}
1869 { "A" set.entry.mark
1870   misc
1871 }
1872
```

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

```
1873 FUNCTION {book} { monograph }
1874
```

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

Required: title

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

```
1875 FUNCTION {booklet} { book }
1876
1877 FUNCTION {collection}
1878 { "G" set.entry.mark
1879   monograph
1880 }
1881
1882 FUNCTION {database}
```

```

1883 { "DB" set.entry.mark
1884   electronic
1885 }
1886
1887 FUNCTION {dataset}
1888 { "DS" set.entry.mark
1889   electronic
1890 }
1891

```

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。

```

1892 FUNCTION {inbook} { book }
1893

```

An inproceedings is an article in a conference proceedings, and it may CROSS-REF 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

```

1894 FUNCTION {inproceedings}
1895 { "C" set.entry.mark
1896   incollection
1897 }
1898

```

The conference function is included for Scribe compatibility.

```

1899 FUNCTION {conference} { inproceedings }
1900
1901 FUNCTION {map}
1902 { "CM" set.entry.mark
1903   misc
1904 }
1905

```

A manual is technical documentation.

Required: title

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

```

1906 FUNCTION {manual} { monograph }
1907

```

A mastersthesis is a Master's thesis.

Required: author, title, school, year

Optional: type, address, month, note

```
1908 FUNCTION {mastersthesis}
1909 <*/!thu>
1910 { "D" set.entry.mark
1911 </!thu>
1912 <*/thu>
1913 { lang.zh entry.lang =
1914   { " 硕士学位论文" }
1915   { "D" }
1916   if$
1917   set.entry.mark
1918 </thu>
1919 monograph
1920 }
1921
1922 FUNCTION {newspaper}
1923 { "N" set.entry.mark
1924   article
1925 }
1926
1927 FUNCTION {online}
1928 { "EB" set.entry.mark
1929   electronic
1930 }
1931
```

A phdthesis is like a mastersthesis.

Required: author, title, school, year

Optional: type, address, month, note

```
1932 <*/!thu>
1933 FUNCTION {phdthesis} { mastersthesis }
1934 </!thu>
1935 <*/thu>
1936 FUNCTION {phdthesis}
1937 { lang.zh entry.lang =
1938   { " 博士学位论文" }
1939   { "D" }
1940   if$
1941   set.entry.mark
1942   monograph
1943 }
1944 </thu>
1945
```

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

```
1946 FUNCTION {proceedings}
1947 { "C" set.entry.mark
1948   monograph
1949 }
1950
1951 FUNCTION {software}
1952 { "CP" set.entry.mark
1953   electronic
1954 }
1955
1956 FUNCTION {standard}
1957 { "S" set.entry.mark
1958   misc
1959 }
1960
```

A techreport is a technical report.

Required: author, title, institution, year

Optional: type, number, address, month, note

```
1961 FUNCTION {techreport}
1962 { "R" set.entry.mark
1963   misc
1964 }
1965
```

An unpublished is something that hasn't been published.

Required: author, title, note

Optional: month, year

```
1966 FUNCTION {unpublished}
1967 { "Z" set.entry.mark
1968   misc
1969 }
1970
```

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

```
1971 FUNCTION {default.type} { misc }
1972
```

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

```
1973 MACRO {jan} {"January"}
1974
1975 MACRO {feb} {"February"}
1976
1977 MACRO {mar} {"March"}
```


1978
 1979 MACRO {apr} {"April"}
 1980
 1981 MACRO {may} {"May"}
 1982
 1983 MACRO {jun} {"June"}
 1984
 1985 MACRO {jul} {"July"}
 1986
 1987 MACRO {aug} {"August"}
 1988
 1989 MACRO {sep} {"September"}
 1990
 1991 MACRO {oct} {"October"}
 1992
 1993 MACRO {nov} {"November"}
 1994
 1995 MACRO {dec} {"December"}
 1996

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

1997 MACRO {acmcs} {"ACM Computing Surveys"}
 1998
 1999 MACRO {acta} {"Acta Informatica"}
 2000
 2001 MACRO {cacm} {"Communications of the ACM"}
 2002
 2003 MACRO {ibmjrd} {"IBM Journal of Research and Development"}
 2004
 2005 MACRO {ibmsj} {"IBM Systems Journal"}
 2006
 2007 MACRO {ieeese} {"IEEE Transactions on Software Engineering"}
 2008
 2009 MACRO {ieeetc} {"IEEE Transactions on Computers"}
 2010
 2011 MACRO {ieeetcad} {"IEEE Transactions on Computer-Aided Design of Integrated Circuits"}
 2012 {"IEEE Transactions on Computer-Aided Design of Integrated Circuits"}
 2013
 2014 MACRO {ipl} {"Information Processing Letters"}
 2015
 2016 MACRO {jacm} {"Journal of the ACM"}
 2017
 2018 MACRO {jcss} {"Journal of Computer and System Sciences"}
 2019
 2020 MACRO {scp} {"Science of Computer Programming"}
 2021
 2022 MACRO {sicomp} {"SIAM Journal on Computing"}
 2023
 2024 MACRO {tocs} {"ACM Transactions on Computer Systems"}

```

2025
2026 MACRO {tods} {"ACM Transactions on Database Systems"}
2027
2028 MACRO {tog} {"ACM Transactions on Graphics"}
2029
2030 MACRO {toms} {"ACM Transactions on Mathematical Software"}
2031
2032 MACRO {toois} {"ACM Transactions on Office Information Systems"}
2033
2034 MACRO {toplas} {"ACM Transactions on Programming Languages and Systems"}
2035
2036 MACRO {tcs} {"Theoretical Computer Science"}
2037

```

B.7 Format labels

The `sortify` function converts to lower case after `purify$ing`; 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`.

```

2038 FUNCTION {sortify}
2039 {  purify$
2040   "l" change.case$
2041 }
2042

```

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

```

2043 FUNCTION {chop.word}
2044 {  's :=
2045   'len :=
2046   s #1 len substring$ =
2047     { s len #1 + global.max$ substring$ }
2048     's
2049   if$
2050 }
2051

```

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$(s)
if numnames > 1 then
  if numnames > 4 then
    namesleft := 3
  else
    namesleft := numnames
  nameptr := 1
  nameresult := ""
  while namesleft > 0
    do
      if (name_ptr = numnames) and
        format.name$(s, nameptr, "{ff }{vv }{ll}{ jj}") = "others"
      then nameresult := nameresult * "{\etalchar{+}}"
        et.al.char.used := true
      else nameresult := nameresult *
        format.name$(s, nameptr, "{v}{l}")
        nameptr := nameptr + 1
        namesleft := namesleft - 1
    od
  if numnames > 4 then
    nameresult := nameresult * "{\etalchar{+}}"
    et.al.char.used := true
  else
    t := format.name$(s, 1, "{v}{l}")
    if text.length$(t) < 2 then % there's just one name-token
      nameresult := text.prefix$(format.name$(s,1,"{ll}"),3)
    else
      nameresult := t
    fi
  fi
  return nameresult
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, purified (purifying 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
```

```
2052 FUNCTION {format.lab.names}
2053 { 's :=
2054   s #1 "{vv~}{ll}{, jj}{, ff}" format.name$ 't :=
2055   t get.str.lang 'name.lang :=
2056   name.lang lang.en =
2057     { t #1 "{vv~}{ll}" format.name$ }
2058     { t #1 "{ll}{ff}" format.name$ }
2059   if$
2060   s num.names$ #1 >
2061     { bbl.space * citation.et.al * }
2062     'skip$
2063   if$
2064 }
2065
2066 FUNCTION {author.key.label}
2067 { author empty$
2068   { key empty$
2069     { cite$ #1 #3 substring$ }
2070     'key
2071     if$
2072   }
2073   { author format.lab.names }
2074   if$
2075 }
2076
2077 FUNCTION {author.editor.key.label}
2078 { author empty$
```

```

2079 { editor empty$
2080   { key empty$
2081     { cite$ #1 #3 substring$ }
2082     'key
2083     if$
2084   }
2085   { editor format.lab.names }
2086   if$
2087 }
2088 { author format.lab.names }
2089 if$
2090 }
2091
2092 FUNCTION {author.key.organization.label}
2093 { author empty$
2094   { key empty$
2095     { organization empty$
2096       { cite$ #1 #3 substring$ }
2097       { "The " #4 organization chop.word #3 text.prefix$ }
2098       if$
2099     }
2100     'key
2101     if$
2102   }
2103   { author format.lab.names }
2104   if$
2105 }
2106
2107 FUNCTION {editor.key.organization.label}
2108 { editor empty$
2109   { key empty$
2110     { organization empty$
2111       { cite$ #1 #3 substring$ }
2112       { "The " #4 organization chop.word #3 text.prefix$ }
2113       if$
2114     }
2115     'key
2116     if$
2117   }
2118   { editor format.lab.names }
2119   if$
2120 }
2121
2122 FUNCTION {calc.short.authors}
2123 { type$ "book" =
2124   type$ "inbook" =
2125   or
2126   'author.editor.key.label
2127   { type$ "collection" =
2128     type$ "proceedings" =
2129     or
2130     { editor empty$ not
2131       'editor.key.organization.label
2132       'author.key.organization.label
2133       if$

```

```

2134     }
2135     'author.key.label
2136     if$
2137   }
2138   if$
2139   'short.list :=
2140 }
2141
2142 FUNCTION {calc.label}
2143 { calc.short.authors
2144   short.list
2145   "("
2146   *
2147   format.year duplicate$ empty$
2148   short.list key field.or.null = or
2149   { pop$ "" }
2150   'skip$
2151   if$
2152   *
2153   'label :=
2154 }
2155

```

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`.

```

2156 (*authoryear)
2157 FUNCTION {sort.language.label}
2158 { entry.lang lang.zh =
2159   { lang.zh.order }
2160   { entry.lang lang.ja =

```

```

2161         { lang.ja.order }
2162     { entry.lang lang.en =
2163         { lang.en.order }
2164         { entry.lang lang.ru =
2165             { lang.ru.order }
2166             { lang.other.order }
2167         if$
2168     }
2169     if$
2170 }
2171     if$
2172 }
2173 if$
2174 int.to.chr$
2175 }
2176
2177 FUNCTION {sort.format.names}
2178 { 's :=
2179   #1 'nameptr :=
2180   ""
2181   s num.names$ 'numnames :=
2182   numnames 'namesleft :=
2183   { namesleft #0 > }
2184   {
2185     s nameptr "{vv{ } }{ll{ }}{ ff{ }}{ jj{ }}" format.name$ 't :=
2186     nameptr #1 >
2187     {
2188       " " *
2189       namesleft #1 = t "others" = and
2190       { "zzzzz" * }
2191       { numnames #2 > nameptr #2 = and
2192         { "zz" * year field.or.null * " " * }
2193         'skip$
2194       if$
2195       t sortify *
2196     }
2197     if$
2198   }
2199   { t sortify * }
2200   if$
2201   nameptr #1 + 'nameptr :=
2202   namesleft #1 - 'namesleft :=
2203 }
2204 while$
2205 }
2206

```

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

```

2207 FUNCTION {sort.format.title}
2208 { 't :=
2209   "A " #2
2210   "An " #3

```

```

2211     "The " #4 t chop.word
2212     chop.word
2213     chop.word
2214     sortify
2215     #1 global.max$ substring$
2216 }
2217

```

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.

```

2218 FUNCTION {anonymous.sort}
2219 { entry.lang lang.zh =
2220   { "yi4 ming2" }
2221   { "anon" }
2222   if$
2223 }
2224
2225 FUNCTION {warn.empty.key}
2226 { entry.lang lang.zh =
2227   { "empty key in " cite$ * warning$ }
2228   'skip$
2229   if$
2230 }
2231
2232 FUNCTION {author.sort}
2233 { key empty$
2234   { warn.empty.key
2235     author empty$
2236     { anonymous.sort }
2237     { author sort.format.names }
2238     if$
2239   }
2240   { key sortify }
2241   if$
2242 }
2243
2244 FUNCTION {author.editor.sort}
2245 { key empty$
2246   { warn.empty.key
2247     author empty$
2248     { editor empty$
2249       { anonymous.sort }
2250       { editor sort.format.names }
2251       if$
2252     }
2253     { author sort.format.names }
2254     if$
2255   }
2256   { key sortify }
2257   if$
2258 }

```



```

2259
2260 FUNCTION {author.organization.sort}
2261 { key empty$
2262   { warn.empty.key
2263     author empty$
2264       { organization empty$
2265         { anonymous.sort }
2266         { "The " #4 organization chop.word sortify }
2267         if$
2268       }
2269       { author sort.format.names }
2270       if$
2271     }
2272   { key sortify }
2273   if$
2274 }
2275
2276 FUNCTION {editor.organization.sort}
2277 { key empty$
2278   { warn.empty.key
2279     editor empty$
2280       { organization empty$
2281         { anonymous.sort }
2282         { "The " #4 organization chop.word sortify }
2283         if$
2284       }
2285       { editor sort.format.names }
2286       if$
2287     }
2288     { key sortify }
2289     if$
2290 }
2291
2292 </authoryear>

```

顺序编码制的排序要简单得多

```

2293 < *numerical>
2294 INTEGERS { seq.num }
2295
2296 FUNCTION {init.seq}
2297 { #0 'seq.num :=}
2298
2299 FUNCTION {int.to.fix}
2300 { "000000000" swap$ int.to.str$ *
2301   #-1 #10 substring$
2302 }
2303
2304 </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!

```

2305 FUNCTION {presort}
2306 { set.entry.lang

```

```

2307 set.entry.numbered
2308 show.url show.doi check.electronic
2309 calc.label
2310 label sortify
2311 " "
2312 *
2313 {*authoryear}
2314 sort.language.label
2315 type$ "book" =
2316 type$ "inbook" =
2317 or
2318   'author.editor.sort
2319   { type$ "collection" =
2320     type$ "proceedings" =
2321     or
2322       'editor.organization.sort
2323       'author.sort
2324     if$
2325   }
2326 if$
2327 *
2328 " "
2329 *
2330 year field.or.null sortify
2331 *
2332 " "
2333 *
2334 cite$
2335 *
2336 #1 entry.max$ substring$
2337 {/authoryear}
2338 {*numerical}
2339 seq.num #1 + 'seq.num :=
2340 seq.num int.to.fix
2341 {/numerical}
2342 'sort.label :=
2343 sort.label *
2344 #1 entry.max$ substring$
2345 'sort.key$ :=
2346 }
2347

```

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.

<pre> VAR: longest.label, last.sort.label, next.extra: string longest.label.width, last.extra.num: integer initialize.longest.label == BEGIN </pre>
--

```

        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

```

```

2348 STRINGS { longest.label last.label next.extra }
2349
2350 INTEGERS { longest.label.width last.extra.num number.label }
2351
2352 FUNCTION {initialize.longest.label}
2353 { "" 'longest.label :=
2354   #0 int.to.chr$ 'last.label :=
2355   "" 'next.extra :=
2356   #0 'longest.label.width :=
2357   #0 'last.extra.num :=
2358   #0 'number.label :=
2359 }
2360
2361 FUNCTION {forward.pass}
2362 { last.label label =
2363   { last.extra.num #1 + 'last.extra.num :=
2364     last.extra.num int.to.chr$ 'extra.label :=
2365   }
2366   { "a" chr.to.int$ 'last.extra.num :=
2367     "" 'extra.label :=
2368     label 'last.label :=
2369   }
2370 if$

```

```

2371 number.label #1 + 'number.label :=
2372 }
2373
2374 FUNCTION {reverse.pass}
2375 { next.extra "b" =
2376   { "a" 'extra.label := }
2377   'skip$
2378   if$
2379   extra.label 'next.extra :=
2380   extra.label
2381   duplicate$ empty$
2382   'skip$
2383   { "{\natexlab{" swap$ * "}" * }
2384   if$
2385   'extra.label :=
2386   label extra.label * 'label :=
2387 }
2388
2389 FUNCTION {bib.sort.order}
2390 { sort.label 'sort.key$ :=
2391 }
2392

```

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.

```

2393 FUNCTION {begin.bib}
2394 { preamble$ empty$
2395   'skip$
2396   { preamble$ write$ newline$ }
2397   if$
2398   "\begin{thebibliography}{ " number.label int.to.str$ * "}" *
2399   write$ newline$
2400   "\providecommand{\natexlab}[1]{#1}"
2401   write$ newline$
2402   "\providecommand{\url}[1]{#1}"
2403   write$ newline$
2404   "\expandafter\ifx\csname urlstyle\endcsname\relax\else"
2405   write$ newline$
2406   " \urlstyle{same}\fi"
2407   write$ newline$
2408   show.doi
2409   { "\expandafter\ifx\csname href\endcsname\relax"
2410     write$ newline$
2411     " \def\doi#1{#1}\else"
2412     write$ newline$
2413     " \def\doi#1{\href{https://doi.org/#1}{#1}}\fi"
2414     write$ newline$

```

```

2415     }
2416     'skip$
2417   if$
2418 }
2419

```

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

```

2420 FUNCTION {end.bib}
2421 { newline$
2422   "\end{thebibliography}" write$ newline$
2423 }
2424

```

B.10 Main execution

Now we read in the .BIB entries.

```

2425 READ
2426
2427 EXECUTE {init.state.consts}
2428
2429 EXECUTE {load.config}
2430
2431 < *numerical>
2432 EXECUTE {init.seq}
2433
2434 < /numerical>
2435 ITERATE {presort}
2436

```

And now we can sort

```

2437 SORT
2438
2439 EXECUTE {initialize.longest.label}
2440
2441 ITERATE {forward.pass}
2442
2443 REVERSE {reverse.pass}
2444
2445 ITERATE {bib.sort.order}
2446
2447 SORT
2448
2449 EXECUTE {begin.bib}
2450

```

Now we produce the output for all the entries

```

2451 ITERATE {call.type$}
2452
2453 EXECUTE {end.bib}
2454 < /authoryear | numerical>

```