

GB/T 7714-2015 Bib_TE_X style

Zeping Lee*

2020/06/08 v2.0.2

摘要

The gbt7714 package provides a Bib_TE_X 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 Bib_TE_X^[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_TE_X 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_TE_X 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_TE_X 对自定义样式的支持比较有限，所以用户只能通过修改 `bst` 文件来修改文献列表的格式。本宏包提供了一些接口供用户更方便地修改。

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

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	输出著者的最多数目
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
title.in.journal	#1	期刊是否显示标题
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”
only.start.page	#0	只显示起始页码
show.url	#1	显示 url
show.doi	#1	显示 doi
show.preprint	#0	显示预印本
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-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/tb_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. CTeX 论坛 (已关闭) .
- [8] 李志奇. 基于 biblatex 的符合 GBT7714-2005 的中文文献生成工具[EB/OL]. 2013. CTeX 论坛 (已关闭) .
- [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{2015}{%
10  \gbt@obsolete@option{2015}%
11  \gbt@legacy@interfacetrue
12  \gbt@mmxvtrue
13 }
14 \DeclareOption{2005}{%
15  \gbt@obsolete@option{2005}%
16  \gbt@legacy@interfacetrue
17  \gbt@mmxvfalse
18 }
19 \DeclareOption{super}{%
20  \gbt@obsolete@option{super}%
21  \gbt@legacy@interfacetrue
22  \gbt@numericaltrue
23  \gbt@supertrue
24 }
25 \DeclareOption{numbers}{%
26  \gbt@obsolete@option{numbers}%
27  \gbt@legacy@interfacetrue
28  \gbt@numericaltrue
29  \gbt@superfalse
30 }
31 \DeclareOption{authoryear}{%
32  \gbt@obsolete@option{authoryear}%
33  \gbt@legacy@interfacetrue
34  \gbt@numericalfalse
35 }

将选项传递给 natbib
36 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{natbib}}
37 \ProcessOptions\relax
```

调用宏包，注意只需要 `compress` 不需要 `sort`。

```
38 \RequirePackage[compress]{natbib}
39 \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`。

```
40 \renewcommand\newblock{\space}
41 \newcommand\bibstyle@super{\bibpunct{[]}{,}{s}{,}{\textsuperscript{,}}{}}
42 \newcommand\bibstyle@numbers{\bibpunct{[]}{,}{n}{,}{,}}
43 \newcommand\bibstyle@authoryear{\bibpunct{({})}{;}{a}{,}{,}}
44 \newcommand\bibstyle@inline{\bibstyle@numbers}
```

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

```
45 \@namedef{bibstyle@gbt7714-numerical}{\bibstyle@super}
46 \@namedef{bibstyle@gbt7714-author-year}{\bibstyle@authoryear}
47 \@namedef{bibstyle@gbt7714-2005-numerical}{\bibstyle@super}
48 \@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：

```
49 \def\NAT@citexnum[#1][#2]#3{%
50   \NAT@reset@parser
51   \NAT@sort@cites{#3}%
52   \NAT@reset@citea
53   \cite{\def\NAT@num{-1}\let\NAT@last@yr\relax\let\NAT@nm\empty
54     \@for\@citeb:=\NAT@cite@list\do
55     {\@safe@activestrue
56       \edef\@citeb{\expandafter\@firstofone\@citeb\@empty}}%
57     \@safe@activesfalse
58     \ifundefined{b@\@citeb\@extra@b@citeb}{%
59       {\reset@font\bfseries?}
60       \NAT@citeundefined\PackageWarning{natbib}%
61       {Citation `@\@citeb' on page \thepage \space undefined}}%
62     {\let\NAT@last@num\NAT@num\let\NAT@last@nm\NAT@nm
63       \NAT@parse{\@citeb}}%
64     \ifNAT@longnames\ifundefined{bv@\@citeb\@extra@b@citeb}{%
65       \let\NAT@name=\NAT@all@names
66       \global\@namedef{bv@\@citeb\@extra@b@citeb}{}}}%
67     \fi
68     \ifNAT@full\let\NAT@nm\NAT@all@names\else
69       \let\NAT@nm\NAT@name\fi
```



```

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

```

87 % \ifx\NAT@last@yr\relax
88 % \def@NAT@last@yr{\@citea}%
89 % \else
90 % \def@NAT@last@yr{--\NAT@penalty}%
91 % \fi
92 \def@NAT@last@yr{-\NAT@penalty}%
93 }{%
94 \NAT@last@yr@mbbox
95 }%
96 }%
97 }{%
98 \@tempwattrue
99 \@ifnum{\NAT@merge>\@ne}{\@ifnum{\NAT@last@num=\NAT@num\relax}{\@tempswafalse}}{}}%
100 \if@tempswa\NAT@citea@mbbox\fi
101 }%
102 }%
103 \NAT@def@citea
104 \else
105 \ifcase\NAT@ctype
106 \ifx\NAT@last@nm\NAT@nm \NAT@yrsep\NAT@penalty\NAT@space\else
107 \@citea \NAT@test{\@ne}\NAT@spacechar\NAT@mbbox{\NAT@super@kern\NAT@open}%
108 \fi
109 \if*#1*\else#1\NAT@spacechar\fi

```

```

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

```

将页码放在括号外边，并且置于上标。

```

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

```

Numerical 模式的 \citep 的页码：

```

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

```

Author-year 模式的 \citet 的页码：

```

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

```

```

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

```

```

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

```

Author-year 模式的 \citep 的页码:

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

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

\url 使用 xurl 宏包的方法, 增加 URL 可断行的位置。

```
234 \g@addto@macro\UrlBreaks{%
235   \do0\do1\do2\do3\do4\do5\do6\do7\do8\do9%
236   \doA\doB\doC\doD\doE\doF\doG\doH\doI\doJ\doK\doL\doM
237   \doN\doO\doP\doQ\doR\doS\doT\doU\doV\doW\doX\doY\doZ
238   \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
239   \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
240 }
241 \Urlmuskip=0mu plus 0.1mu
```

兼容 v2.0 前过时的接口:

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

```

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

```

273 (*authoryear | numerical)
274 INTEGERS {
275   uppercase.name
276   max.num.authors
277   period.between.author.year
278   sentence.case.title
279   link.title
280   title.in.journal
281   show.mark
282   show.medium.type
283   slash.for.extraction
284   in.booktitle
285   abbreviate.journal
286   italic.journal
287   bold.journal.volume
288   show.missing.address.publisher
289   only.start.page
290   show.url
291   show.doi
292   show.preprint
293   show.note
294   show.english.translation
295 (*authoryear)
296   lang.zh.order
297   lang.ja.order
298   lang.en.order
299   lang.ru.order
300   lang.other.order
301 \end{authoryear}
302 }
303

```

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。

```

304 FUNCTION {load.config}
305 {

```

英文姓名转为全大写:

```
306 <!*nouppercase&!thu>
307 #1 'uppercase.name :=
308 </!nouppercase&!thu>
309 <!*nouppercase | thu>
310 #0 'uppercase.name :=
311 </nouppercase | thu>
```

最多显示的作者数量:

```
312 #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:

```
313 <!*authoryear>
314 <!*!period&!2005&!ustc>
315 #0 'period.between.author.year :=
316 </!period&!2005&!ustc>
317 <!*period | 2005 | ustc>
318 #1 'period.between.author.year :=
319 </period | 2005 | ustc>
320 </authoryear>
```

英文标题转为 sentence case (句首字母大写, 其余小写): <!*!nosentencecase>

```
321 </!nosentencecase>
322 #1 'sentence.case.title :=
323 <!*nosentencecase>
324 #0 'sentence.case.title :=
325 </nosentencecase>
```

在标题添加超链接:

```
326 <!*!linktitle>
327 #0 'link.title :=
328 </!linktitle>
329 <!*linktitle>
330 #1 'link.title :=
331 </linktitle>
```

期刊是否含标题:

```
332 <!*!title-in-journal&!npr>
333 #1 'title.in.journal :=
334 </!title-in-journal&!npr>
335 <!*title-in-journal | npr>
336 #0 'title.in.journal :=
337 </title-in-journal | npr>
```

著录文献类型标识 (比如“[M/OL]“):

```
338 <!*!nomark>
339 #1 'show.mark :=
340 </!nomark>
341 <!*nomark>
342 #0 'show.mark :=
343 </nomark>
```

是否显示载体类型标识 (比如“/OL“):

```
344 <!*!no.medium.type>
```

```

345 #1 'show.medium.type :=
346 </!no.medium.type>
347 <!*no.medium.type>
348 #0 'show.medium.type :=
349 </no.medium.type>

```

使用“//”表示析出文献

```

350 <!*noslash>
351 #1 'slash.for.extraction :=
352 </!noslash>
353 <!*noslash>
354 #0 'slash.for.extraction :=
355 </noslash>

```

使用“In:”表示析出文献

```

356 #0 'in.booktitle :=

```

期刊名使用缩写:

```

357 <!*abbreviate-journal&!npr>
358 #0 'abbreviate.journal :=
359 </!abbreviate-journal&!npr>
360 <!*abbreviate-journal | npr>
361 #1 'abbreviate.journal :=
362 </abbreviate-journal | npr>

```

期刊名使用斜体:

```

363 <!*italicjournal>
364 #0 'italic.journal :=
365 </!italicjournal>
366 <!*italicjournal>
367 #1 'italic.journal :=
368 </italicjournal>

```

期刊的卷使用粗体:

```

369 #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.”:

```

370 <!*noslsn&!thu&!ustc&!npr>
371 #1 'show.missing.address.publisher :=
372 </!noslsn&!thu&!ustc&!npr>
373 <!*noslsn | thu | ustc | npr>
374 #0 'show.missing.address.publisher :=
375 </noslsn | thu | ustc | npr>

```

页码是否只含起始页:

```

376 <!*only-start-page&!npr>
377 #0 'only.start.page :=
378 </!only-start-page&!npr>
379 <!*only-start-page | npr>
380 #1 'only.start.page :=
381 </only-start-page | npr>

```

是否著录 URL:

```

382 <!*nourl>
383 #1 'show.url :=

```



```

384 </!nourl>
385 <!*nourl>
386 #0 'show.url :=
387 </nourl>

```

是否著录 DOI:

```

388 <!*!nodoi&!2005>
389 #1 'show.doi :=
390 </!nodoi&!2005>
391 <!*nodoi | 2005>
392 #0 'show.doi :=
393 </nodoi | 2005>

```

是否著录 e-print:

```

394 <!*preprint&!npr>
395 #0 'show.preprint :=
396 </!preprint&!npr>
397 <!*preprint | npr>
398 #1 'show.preprint :=
399 </preprint | npr>

```

在每一条文献最后输出注释 (note) 的内容:

```

400 #0 'show.note :=

```

中文文献是否显示英文翻译

```

401 <!*show-english-translation&!npr>
402 #0 'show.english.translation :=
403 </!show-english-translation&!npr>
404 <!*show-english-translation | npr>
405 #1 'show.english.translation :=
406 </show-english-translation | npr>

```

参考文献表按照“著者-出版年”组织时, 各个文种的顺序:

```

407 <!*authoryear>
408 #1 'lang.zh.order :=
409 #2 'lang.ja.order :=
410 #3 'lang.en.order :=
411 #4 'lang.ru.order :=
412 #5 'lang.other.order :=
413 </authoryear>
414 }
415

```

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 `annote` 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.

```

416 ENTRY

```

```

417 { address
418   archivePrefix
419   author
420   booktitle
421   date
422   doi
423   edition
424   editor
425   eprint
426   howpublished
427   institution
428   journal
429   key
430   language
431   mark
432   medium
433   note
434   number
435   organization
436   pages
437   publisher
438   school
439   series
440   title
441   translation
442   translator
443   url
444   urldate
445   volume
446   year
447 }
448 { 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.

```

449 { label extra.label sort.label short.list entry.mark entry.url }
450

```

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 argu-

ment 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

```

```

451 INTEGERS { output.state before.all mid.sentence after.sentence after.block after.slash }
452
453 INTEGERS { lang.zh lang.ja lang.en lang.ru lang.other }
454
455 INTEGERS { charptr len }
456
457 FUNCTION {init.state.consts}
458 { #0 'before.all :=
459   #1 'mid.sentence :=
460   #2 'after.sentence :=
461   #3 'after.block :=
462   #4 'after.slash :=
463   #3 'lang.zh :=
464   #4 'lang.ja :=
465   #1 'lang.en :=
466   #2 'lang.ru :=
467   #0 'lang.other :=
468 }
469

```

下面是一些常量的定义

```

470 FUNCTION {bbl.anonymous}
471 { entry.lang lang.zh =
472   { "佚名" }
473   { "Anon" }
474   if$
475 }
476
477 FUNCTION {bbl.space}
478 { entry.lang lang.zh =
479   { "\ " }
480   { " " }
481   if$
482 }
483
484 FUNCTION {bbl.et.al}
485 { entry.lang lang.zh =
486   { "等" }

```

```

487 { entry.lang lang.ja =
488   { "他" }
489   { entry.lang lang.ru =
490     { "идр" }
491     { "et~al." }
492     if$
493   }
494   if$
495 }
496 if$
497 }
498
499 FUNCTION {citation.et.al}
500 { bbl.et.al }
501
502 FUNCTION {bbl.colon} { ": " }
503
504 <*/2015>
505 FUNCTION {bbl.wide.space} { "\quad " }
506 </2015>
507 <*/2005>
508 FUNCTION {bbl.wide.space} { "\ " }
509 </2005>
510
511 <*/!thu>
512 FUNCTION {bbl.slash} { "//\allowbreak " }
513 </!thu>
514 <*/thu>
515 FUNCTION {bbl.slash} { " //" }
516 </thu>
517
518 FUNCTION {bbl.sine.loco}
519 { entry.lang lang.zh =
520   { "[出版地不详]" }
521   { "[S.l.]" }
522   if$
523 }
524
525 FUNCTION {bbl.sine.nomine}
526 { entry.lang lang.zh =
527   { "[出版者不详]" }
528   { "[s.n.]" }
529   if$
530 }
531
532 FUNCTION {bbl.sine.loco.sine.nomine}
533 { entry.lang lang.zh =
534   { "[出版地不详: 出版者不详]" }
535   { "[S.l.: s.n.]" }
536   if$
537 }
538

```

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

```
539 FUNCTION {not}
540 { { #0 }
541   { #1 }
542   if$
543 }
544
545 FUNCTION {and}
546 { 'skip$
547   { pop$ #0 }
548   if$
549 }
550
551 FUNCTION {or}
552 { { pop$ #1 }
553   'skip$
554   if$
555 }
556
```

the variables s and t are temporary string holders

```
557 STRINGS { s t }
558
559 FUNCTION {output.nonnull}
560 { 's :=
561   output.state mid.sentence =
562     { ", " * write$ }
563     { output.state after.block =
564       { add.period$ write$
565         newline$
566         "\newblock " write$
567       }
568       { output.state before.all =
569         'write$
570         { output.state after.slash =
571           { bbl.slash * write$
572             newline$
573           }
574           { add.period$ " " * write$ }
575         if$
576       }
577       if$
578     }
579     if$
580     mid.sentence 'output.state :=
581   }
582   if$
583   s
584 }
585
586 FUNCTION {output}
587 { duplicate$ empty$
588   'pop$
589   'output.nonnull
590   if$
```

```

591 }
592
593 FUNCTION {output.after}
594 { 't :=
595   duplicate$ empty$
596   'pop$
597   { 's :=
598     output.state mid.sentence =
599     { t * write$ }
600     { output.state after.block =
601       { add.period$ write$
602         newline$
603         "\newblock " write$
604       }
605       { output.state before.all =
606         'write$
607         { output.state after.slash =
608           { bbl.slash * write$ }
609           { add.period$ " " * write$ }
610         }
611         }
612       }
613     }
614     if$
615     mid.sentence 'output.state :=
616   }
617   if$
618   s
619 }
620 if$
621 }
622
623 FUNCTION {output.check}
624 { 't :=
625   duplicate$ empty$
626   { pop$ "empty " t * " in " * cite$ * warning$ }
627   'output.nonnull
628   if$
629 }
630

```

This function finishes all entries.

```

631 FUNCTION {fin.entry}
632 { add.period$
633   write$
634   show.english.translation entry.lang lang.zh = and
635   { ""}
636   write$
637   }
638   'skip$
639   if$
640   newline$
641 }
642
643 FUNCTION {new.block}

```



```

644 { output.state before.all =
645     'skip$
646     { output.state after.slash =
647         'skip$
648         { after.block 'output.state := }
649     if$
650     }
651 if$
652 }
653
654 FUNCTION {new.sentence}
655 { output.state after.block =
656     'skip$
657     { output.state before.all =
658         'skip$
659         { output.state after.slash =
660             'skip$
661             { after.sentence 'output.state := }
662         if$
663         }
664     if$
665     }
666 if$
667 }
668
669 FUNCTION {new.slash}
670 { output.state before.all =
671     'skip$
672     { slash.for.extraction
673         { after.slash 'output.state := }
674         { after.block 'output.state := }
675     if$
676     }
677 if$
678 }
679

```

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.

```

680 FUNCTION {new.block.checka}
681 { empty$
682     'skip$
683     'new.block
684 if$
685 }
686
687 FUNCTION {new.block.checkb}
688 { empty$
689     swap$ empty$
690     and
691     'skip$
692     'new.block
693 if$

```

```
694 }
695
```

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

```
696 FUNCTION {new.sentence.checka}
697 { empty$
698   'skip$
699   'new.sentence
700   if$
701 }
702
703 FUNCTION {new.sentence.checkb}
704 { empty$
705   swap$ empty$
706   and
707   'skip$
708   'new.sentence
709   if$
710 }
711
```

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 emphasized, 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

```
712 FUNCTION {field.or.null}
713 { duplicate$ empty$
714   { pop$ "" }
715   'skip$
716   if$
717 }
718
719 FUNCTION {italicize}
720 { duplicate$ empty$
721   { pop$ "" }
722   { "\textit{" swap$ * ""} * }
723   if$
724 }
725
```

B.4.1 Detect Language

```
726 INTEGERS { byte second.byte }
727
728 INTEGERS { char.lang tmp.lang }
729
730 STRINGS { tmp.str }
731
732 FUNCTION {get.str.lang}
733 { 'tmp.str :=
734   lang.other 'tmp.lang :=
735   #1 'charptr :=
736   tmp.str text.length$ #1 + 'len :=
737   { charptr len < }
738   { tmp.str charptr #1 substring$ chr.to.int$ 'byte :=
739     byte #128 <
740     { charptr #1 + 'charptr :=
741       byte #64 > byte #91 < and byte #96 > byte #123 < and or
742       { lang.en 'char.lang := }
743       { lang.other 'char.lang := }
744       if$
745     }
746     { tmp.str charptr #1 + #1 substring$ chr.to.int$ 'second.byte :=
747       byte #224 <
```

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

```
748     { charptr #2 + 'charptr :=
749       byte #207 > byte #212 < and
750       byte #212 = second.byte #176 < and or
751       { lang.ru 'char.lang := }
752       { lang.other 'char.lang := }
753       if$
754     }
755     { byte #240 <
```

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

```
756     { charptr #3 + 'charptr :=
```

```

757         byte #227 > byte #234 < and
758         { lang.zh 'char.lang := }

```

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

```

759         { byte #227 =
760         { second.byte #143 >
761         { lang.zh 'char.lang := }

```

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

```

762         { second.byte #128 > second.byte #132 < and
763         { lang.ja 'char.lang := }
764         { lang.other 'char.lang := }
765         if$
766         }
767         if$
768         }

```

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

```

769         { byte #239 =
770         second.byte #163 > second.byte #172 < and and
771         { lang.zh 'char.lang := }
772         { lang.other 'char.lang := }
773         if$
774         }
775         if$
776         }
777         if$
778         }

```

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.

```

779         { charptr #4 + 'charptr :=
780         byte #240 = second.byte #159 > and
781         { lang.zh 'char.lang := }
782         { lang.other 'char.lang := }
783         if$
784         }
785         if$
786         }
787         if$
788         }
789         if$
790         char.lang tmp.lang >
791         { char.lang 'tmp.lang := }
792         'skip$
793         if$
794         }
795     while$
796     tmp.lang
797 }
798
799 FUNCTION {check.entry.lang}

```

```

800 { author field.or.null
801   title field.or.null *
802   get.str.lang
803 }
804
805 FUNCTION {set.entry.lang}
806 { language empty$
807   { check.entry.lang }
808   { language "english" = language "american" = or language "british" = or
809     { lang.en }
810     { language "chinese" =
811       { lang.zh }
812       { language "japanese" =
813         { lang.ja }
814         { language "russian" =
815           { lang.ru }
816           { check.entry.lang }
817         if$
818       }
819     }
820   }
821   if$
822 }
823 if$
824 }
825 if$
826 'entry.lang :=
827 }
828
829 FUNCTION {set.entry.numbered}
830 { type$ "patent" =
831   type$ "standard" = or
832   type$ "techreport" = or
833   { #1 'entry.numbered := }
834   { #0 'entry.numbered := }
835   if$
836 }
837

```

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.

<pre> VAR: nameptr, namesleft, numnames: INTEGER pseudoVAR: namesresult: STRING (it's what's accumulated on the stack) format.names(s) == BEGIN nameptr := 1 numnames := num.names\$(s) namesleft := numnames </pre>
--

```

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.

```

838 INTEGERS { nameptr namesleft numnames name.lang }
839

```

```

840 FUNCTION {format.names}
841 { 's :=
842   #1 'nameptr :=
843   s num.names$ 'numnames :=
844   ""
845   numnames 'namesleft :=
846   { namesleft #0 > }
847   { s nameptr "{vv~}{ll}{, jj}{, ff}" format.name$ 't :=
848     nameptr #1 >
849     { ", " * }
850     'skip$
851     if$
852     nameptr max.num.authors >
853     { bbl.et.al *
854       #1 'namesleft :=
855       }
856     { t "others" =
857       { bbl.et.al * }
858       { t get.str.lang 'name.lang :=
859         name.lang lang.en =
860         { t #1 "{vv~}{ll}{~f{~}}" format.name$
861           uppercase.name
862           { "u" change.case$ }
863           'skip$
864           if$
865           t #1 "{, jj}" format.name$ *
866           }
867           { t #1 "{ll}{ff}" format.name$ }
868           if$
869           *
870           }
871           if$
872           }
873           if$
874           nameptr #1 + 'nameptr :=
875           namesleft #1 - 'namesleft :=
876           }
877   while$
878 }
879
880 FUNCTION {format.key}
881 { empty$
882   { key field.or.null }
883   { "" }
884   if$
885 }
886
887 FUNCTION {format.authors}
888 { author empty$ not
889   { author format.names }
890   { "empty author in " cite$ * warning$
891     (*authoryear)
892     bbl.anonymous
893     (/authoryear)
894     (*numerical)

```

```

895     ""
896 </numerical>
897   }
898   if$
899 }
900
901 FUNCTION {format.editors}
902 { editor empty$
903   { "" }
904   { editor format.names }
905   if$
906 }
907
908 FUNCTION {format.translators}
909 { translator empty$
910   { "" }
911   { translator format.names
912     entry.lang lang.zh =
913     { translator num.names$ #3 >
914       { " 译" * }
915       { ", 译" * }
916       if$
917     }
918     'skip$
919     if$
920   }
921   if$
922 }
923
924 FUNCTION {format.full.names}
925 { 's :=
926   #1 'nameptr :=
927   s num.names$ 'numnames :=
928   numnames 'namesleft :=
929   { namesleft #0 > }
930   { s nameptr "{vv~}{ll}{, jj}{, ff}" format.name$ 't :=
931     t get.str.lang 'name.lang :=
932     name.lang lang.en =
933     { t #1 "{vv~}{ll}" format.name$ 't := }
934     { t #1 "{ll}{ff}" format.name$ 't := }
935     if$
936     nameptr #1 >
937     {
938       namesleft #1 >
939       { ", " * t * }
940       {
941         numnames #2 >
942         { ", " * }
943         'skip$
944         if$
945         t "others" =
946         { " et~al." * }
947         { " and " * t * }
948         if$
949       }

```



```

950         if$
951     }
952     't
953     if$
954     nameptr #1 + 'nameptr :=
955     namesleft #1 - 'namesleft :=
956     }
957 while$
958 }
959
960 FUNCTION {author.editor.full}
961 { author empty$
962   { editor empty$
963     { "" }
964     { editor format.full.names }
965     if$
966   }
967   { author format.full.names }
968   if$
969 }
970
971 FUNCTION {author.full}
972 { author empty$
973   { "" }
974   { author format.full.names }
975   if$
976 }
977
978 FUNCTION {editor.full}
979 { editor empty$
980   { "" }
981   { editor format.full.names }
982   if$
983 }
984
985 FUNCTION {make.full.names}
986 { type$ "book" =
987   type$ "inbook" =
988   or
989   'author.editor.full
990   { type$ "collection" =
991     type$ "proceedings" =
992     or
993     'editor.full
994     'author.full
995     if$
996   }
997   if$
998 }
999
1000 FUNCTION {output.bibitem}
1001 { newline$
1002   "\bibitem[" writes$
1003   label ")" *
1004   make.full.names duplicate$ short.list =

```

```

1005     { pop$ }
1006     { * }
1007   if$
1008   's :=
1009   s text.length$ 'charptr :=
1010     { charptr #0 > s charptr #1 substring$ "[" = not and }
1011     { charptr #1 - 'charptr := }
1012   while$
1013   charptr #0 >
1014     { "{" s * "}" * }
1015     { s }
1016   if$
1017   "]" * write$
1018   cite$ write$
1019   "]" write$
1020   newline$
1021   ""
1022   before.all 'output.state :=
1023 }
1024

```

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.

```

1025 FUNCTION {change.sentence.case}
1026 { entry.lang lang.en =
1027   { "t" change.case$ }
1028   'skip$
1029   if$
1030 }
1031
1032 FUNCTION {add.link}
1033 { url empty$ not
1034   { "\href{" url * "}" * swap$ * "}" * }
1035   { doi empty$ not
1036     { "\href{http://dx.doi.org/" doi * "}" * swap$ * "}" * }
1037     'skip$
1038     if$
1039   }
1040   if$
1041 }
1042
1043 FUNCTION {format.title}
1044 { title empty$
1045   { "" }
1046   { title
1047     sentence.case.title
1048     'change.sentence.case
1049     'skip$

```

```

1050     if$
1051     entry.numbered number empty$ not and
1052         { bbl.colon * number * }
1053         'skip$
1054     if$
1055     link.title
1056         'add.link
1057         'skip$
1058     if$
1059     }
1060 if$
1061 }
1062

```

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

```

```

1063 FUNCTION {tie.or.space.connect}
1064 { duplicate$ text.length$ #3 <
1065   { "~" }
1066   { " " }
1067   if$
1068   swap$ * *
1069 }
1070

```

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

```

```

1071 FUNCTION {either.or.check}
1072 { empty$
1073   'pop$
1074   { "can't use both " swap$ * " fields in " * cite$ * warning$ }
1075   if$
1076 }
1077

```

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.

```

1078 FUNCTION {is.digit}
1079 { duplicate$ empty$
1080   { pop$ #0 }
1081   { chr.to.int$
1082     duplicate$ "0" chr.to.int$ <
1083     { pop$ #0 }
1084     { "9" chr.to.int$ >
1085       { #0 }
1086       { #1 }
1087     if$
1088   }
1089   if$
1090 }
1091 if$
1092 }
1093
1094 FUNCTION {is.number}
1095 { 's :=
1096   s empty$
1097   { #0 }
1098   { s text.length$ 'charptr :=
1099     { charptr #0 >
1100       s charptr #1 substring$ is.digit
1101       and
1102     }
1103     { charptr #1 - 'charptr := }
1104     while$
1105     charptr not
1106   }
1107   if$
1108 }
1109
1110 FUNCTION {format.volume}
1111 { volume empty$ not
1112   { volume is.number
1113     { entry.lang lang.zh =
1114       { " 第 " volume * " 卷" * }

```

```

1115         { "volume" volume tie.or.space.connect }
1116         if$
1117         }
1118         { volume }
1119     if$
1120     }
1121     { "" }
1122 if$
1123 }
1124
1125 FUNCTION {format.number}
1126 { number empty$ not
1127   { number is.number
1128     { entry.lang lang.zh =
1129       { "第 " number * "册" * }
1130       { "number" number tie.or.space.connect }
1131     if$
1132     }
1133     { number }
1134   if$
1135   }
1136   { "" }
1137 if$
1138 }
1139
1140 FUNCTION {format.volume.number}
1141 { volume empty$ not
1142   { format.volume }
1143   { format.number }
1144 if$
1145 }
1146
1147 FUNCTION {format.title.vol.num}
1148 { title
1149   sentence.case.title
1150   'change.sentence.case
1151   'skip$
1152 if$
1153   entry.numbered
1154   { number empty$ not
1155     { bbl.colon * number * }
1156     'skip$
1157   if$
1158   }
1159   { format.volume.number 's :=
1160     s empty$ not
1161     { bbl.colon * s * }
1162     'skip$
1163   if$
1164   }
1165 if$
1166 }
1167
1168 FUNCTION {format.series.vol.num.title}
1169 { format.volume.number 's :=

```

```

1170 series empty$ not
1171   { series
1172     sentence.case.title
1173     'change.sentence.case
1174     'skip$
1175     if$
1176     entry.numbered
1177     { bbl.wide.space * }
1178     { bbl.colon *
1179       s empty$ not
1180       { s * bbl.wide.space * }
1181       'skip$
1182     if$
1183     }
1184     if$
1185     title *
1186     sentence.case.title
1187     'change.sentence.case
1188     'skip$
1189     if$
1190     entry.numbered number empty$ not and
1191     { bbl.colon * number * }
1192     'skip$
1193     if$
1194   }
1195   { format.title.vol.num }
1196 if$
1197 link.title
1198   'add.link
1199   'skip$
1200 if$
1201 }
1202
1203 FUNCTION {format.booktitle.vol.num}
1204 { booktitle
1205   entry.numbered
1206   'skip$
1207   { format.volume.number 's :=
1208     s empty$ not
1209     { bbl.colon * s * }
1210     'skip$
1211     if$
1212   }
1213 if$
1214 }
1215
1216 FUNCTION {format.series.vol.num.booktitle}
1217 { format.volume.number 's :=
1218   series empty$ not
1219   { series bbl.colon *
1220     entry.numbered not s empty$ not and
1221     { s * bbl.wide.space * }
1222     'skip$
1223     if$
1224     booktitle *

```

```

1225     }
1226     { format.booktitle.vol.num }
1227   if$
1228   in.booktitle
1229     { duplicate$ empty$ not entry.lang lang.en = and
1230       { "In: " swap$ * }
1231       'skip$
1232     if$
1233   }
1234   'skip$
1235   if$
1236 }
1237
1238 FUNCTION {remove.period}
1239 { 't :=
1240   "" 's :=
1241   { t empty$ not }
1242   { t #1 #1 substring$ 'tmp.str :=
1243     tmp.str "." = not
1244     { s tmp.str * 's := }
1245     'skip$
1246     if$
1247     t #2 global.max$ substring$ 't :=
1248   }
1249   while$
1250   s
1251 }
1252
1253 FUNCTION {abbreviate}
1254 { remove.period
1255   't :=
1256   t "l" change.case$ 's :=
1257   ""
1258   s "physical review letters" =
1259     { "Phys Rev Lett" }
1260     'skip$
1261   if$
1262   (*npr)
1263   s "china physics c" =
1264     { "Chin Phys C" }
1265     'skip$
1266   if$
1267   s "chinese physics letters" =
1268     { "Chin Phys Lett" }
1269     'skip$
1270   if$
1271   s "nuclear instruments and methods in physics research section a" =
1272     { "Nucl Instr and Meth A" }
1273     'skip$
1274   if$
1275   s "nuclear instruments and methods in physics research section a: accelerators, spectrometers,"
1276     { "Nucl Instr and Meth A" }
1277     'skip$
1278   if$
1279   s "nuclear instruments and methods in physics research section b" =

```

```

1280   { "Nucl Instr and Meth B" }
1281   'skip$
1282   if$
1283   s "nuclear instruments and methods in physics research section b: beam interactions with materi
1284   { "Nucl Instr and Meth B" }
1285   'skip$
1286   if$
1287   s "physical review c" =
1288   { "Phys Rev C" }
1289   'skip$
1290   if$
1291   s "physical review d" =
1292   { "Phys Rev D" }
1293   'skip$
1294   if$
1295   s "physical review e" =
1296   { "Phys Rev E" }
1297   'skip$
1298   if$
1299   s "physics letters b" =
1300   { "Phys Lett B" }
1301   'skip$
1302   if$
1303    $\langle$ /npr $\rangle$ 
1304   's :=
1305   s empty$
1306   { t }
1307   { pop$ s }
1308   if$
1309 }
1310
1311 FUNCTION {format.journal}
1312 { journal empty$ not
1313   { journal
1314     abbreviate.journal
1315     'abbreviate
1316     'skip$
1317     if$
1318     italic.journal entry.lang lang.en = and
1319     'italicize
1320     'skip$
1321     if$
1322   }
1323   { "" }
1324   if$
1325 }
1326

```

B.4.4 Format entry type mark

```

1327 FUNCTION {set.entry.mark}
1328 { entry.mark empty$ not
1329   'pop$
1330   { mark empty$ not
1331     { pop$ mark 'entry.mark := }

```



```

1332         { 'entry.mark := }
1333     if$
1334 }
1335 if$
1336 }
1337
1338 FUNCTION {format.mark}
1339 { show.mark
1340 < *thu)
1341 type$ "phdthesis" = type$ "mastersthesis" = or type$ "patent" = or
1342 medium empty$ not or entry.is.electronic or
1343 and
1344 </thu)
1345     { entry.mark
1346       show.medium.type
1347         { medium empty$ not
1348           { "/" * medium * }
1349           { entry.is.electronic
1350             { "/OL" * }
1351             'skip$
1352             if$
1353           }
1354         if$
1355       }
1356       'skip$
1357     if$
1358     'entry.mark :=
1359 < *!thu)
1360     "\allowbreak[" entry.mark * "]" *
1361 </!thu)
1362 < *thu)
1363     "[" entry.mark * "]" *
1364 </thu)
1365   }
1366   { "" }
1367 if$
1368 }
1369

```

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.

```

1370 FUNCTION {num.to.ordinal}
1371 { duplicate$ text.length$ 'charptr :=
1372   duplicate$ charptr #1 substring$ 's :=
1373   s "1" =
1374     { "st" * }
1375     { s "2" =
1376       { "nd" * }
1377       { s "3" =
1378         { "rd" * }
1379         { "th" * }

```

```

1380         if$
1381     }
1382     if$
1383 }
1384 if$
1385 }
1386
1387 FUNCTION {format.edition}
1388 { edition empty$
1389   { "" }
1390   { edition is.number
1391     { entry.lang lang.zh =
1392       { edition " 版" * }
1393       { edition num.to.ordinal " ed." * }
1394       if$
1395     }
1396     { entry.lang lang.en =
1397       { edition change.sentence.case 's :=
1398         s "Revised" = s "Revised edition" = or
1399         { "Rev. ed." }
1400         { s " ed." *}
1401         if$
1402       }
1403       { edition }
1404       if$
1405     }
1406     if$
1407   }
1408 if$
1409 }
1410

```

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。

```

1411 FUNCTION {format.publisher}
1412 { publisher empty$ not
1413   { publisher }
1414   { school empty$ not
1415     { school }
1416     { organization empty$ not
1417       { organization }
1418       { institution empty$ not
1419         { institution }
1420         { "" }
1421         if$
1422       }
1423       if$
1424     }
1425     if$
1426   }
1427 if$
1428 }
1429

```

```

1430 FUNCTION {format.address.publisher}
1431 { address empty$ not
1432   { address
1433     format.publisher empty$ not
1434     { bbl.colon * format.publisher * }
1435     { entry.is.electronic not show.missing.address.publisher and
1436       { bbl.colon * bbl.sine.nomine * }
1437       'skip$
1438     if$
1439   }
1440   if$
1441 }
1442 { entry.is.electronic not show.missing.address.publisher and
1443   { format.publisher empty$ not
1444     { bbl.sine.loco bbl.colon * format.publisher * }
1445     { bbl.sine.loco.sine.nomine }
1446     if$
1447   }
1448   { format.publisher empty$ not
1449     { format.publisher }
1450     { "" }
1451     if$
1452   }
1453   if$
1454 }
1455 if$
1456 }
1457

```

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`。

```

1458 FUNCTION {extract.before.dash}
1459 { duplicate$ empty$
1460   { pop$ "" }
1461   { 's :=
1462     #1 'charptr :=
1463     s text.length$ #1 + 'len :=
1464     { charptr len <
1465       s charptr #1 substring$ "-" = not
1466       and
1467     }
1468     { charptr #1 + 'charptr := }
1469     while$
1470     s #1 charptr #1 - substring$
1471   }
1472   if$
1473 }

```

```

1474
1475 FUNCTION {extract.after.dash}
1476 { duplicate$ empty$
1477   { pop$ "" }
1478   { 's :=
1479     #1 'charptr :=
1480     s text.length$ #1 + 'len :=
1481     { charptr len <
1482       s charptr #1 substring$ "-" = not
1483       and
1484     }
1485     { charptr #1 + 'charptr := }
1486     while$
1487     { charptr len <
1488       s charptr #1 substring$ "-" =
1489       and
1490     }
1491     { charptr #1 + 'charptr := }
1492     while$
1493     s charptr global.max$ substring$
1494   }
1495   if$
1496 }
1497
1498 FUNCTION {contains.dash}
1499 { duplicate$ empty$
1500   { pop$ #0 }
1501   { 's :=
1502     { s empty$ not
1503       s #1 #1 substring$ "-" = not
1504       and
1505     }
1506     { s #2 global.max$ substring$ 's := }
1507     while$
1508     s empty$ not
1509   }
1510   if$
1511 }
1512
    著者-出版年制必须提取出年份
1513 FUNCTION {format.year}
1514 { year empty$ not
1515   { year extract.before.dash }
1516   { date empty$ not
1517     { date extract.before.dash }
1518     { "empty year in " cite$ * warning$
1519       urldate empty$ not
1520       { "[" urldate extract.before.dash * "]" * }
1521       { "" }
1522     }
1523     if$
1524   }
1525   if$
1526 }

```

```

1527 extra.label *
1528 }
1529

```

专利和报纸都是使用日期而不是年

```

1530 FUNCTION {format.date}
1531 { type$ "patent" = type$ "newspaper" = or
1532   date empty$ not and
1533   { date }
1534   { year }
1535   if$
1536 }
1537

```

更新、修改日期只用于电子资源 `elctronic`

```

1538 FUNCTION {format.editeddate}
1539 { date empty$ not
1540   { "\allowbreak(" date * ")" * }
1541   { "" }
1542   if$
1543 }
1544

```

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。

```

1545 FUNCTION {format.urldate}
1546 { urldate empty$ not entry.is.electronic and
1547   { "\allowbreak[" urldate * "]" * }
1548   { "" }
1549   if$
1550 }
1551

```

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

<pre> pseudoVAR: pageresult: STRING (it's what's accumulated on the stack) n.dashify(s) == BEGIN t := s pageresult := "" while (not empty\$(t)) do </pre>

```

        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。

```

1552 FUNCTION {hyphenate}
1553 { 't :=
1554   ""
1555   { t empty$ not }
1556   { t #1 #1 substring$ "-" =
1557     { "-" *
1558       { t #1 #1 substring$ "-" = }
1559       { t #2 global.max$ substring$ 't := }
1560       while$
1561     }
1562     { t #1 #1 substring$ *
1563       t #2 global.max$ substring$ 't :=
1564     }
1565     if$
1566   }
1567   while$
1568 }
1569

```

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

```

1570 FUNCTION {format.pages}
1571 { pages empty$
1572   { "" }
1573   { pages hyphenate }
1574   if$
1575 }
1576
1577 FUNCTION {format.extracted.pages}
1578 { pages empty$
1579   { "" }
1580   { pages

```

```

1581     only.start.page
1582     'extract.before.dash
1583     'hyphenate
1584     if$
1585   }
1586   if$
1587 }
1588

```

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。

```

1589 FUNCTION {format.journal.volume}
1590 { volume empty$ not
1591   { bold.journal.volume
1592     { "\textbf{" volume * "}" * }
1593     { volume }
1594     if$
1595   }
1596   { "" }
1597   if$
1598 }
1599
1600 FUNCTION {format.journal.number}
1601 { number empty$ not
1602   { "\penalty0 (" number * ")" * }
1603   { "" }
1604   if$
1605 }
1606
1607 FUNCTION {format.journal.pages}
1608 { pages empty$
1609   { "" }
1610   { ": "
1611     format.extracted.pages *
1612   }
1613   if$
1614 }
1615

```

连续出版物的年卷期有起止范围，需要特殊处理

```

1616 FUNCTION {format.periodical.year.volume.number}
1617 { year empty$ not
1618   { year extract.before.dash }
1619   { "empty year in periodical " cite$ * warning$ }
1620   if$
1621   volume empty$ not
1622   { ", " * volume extract.before.dash * }
1623   'skip$
1624   if$
1625   number empty$ not
1626   { "\penalty0 (" * number extract.before.dash * ")" * }

```

```

1627     'skip$
1628   if$
1629   year contains.dash
1630     { "--" *
1631     year extract.after.dash empty$
1632     volume extract.after.dash empty$ and
1633     number extract.after.dash empty$ and not
1634     { year extract.after.dash empty$ not
1635     { year extract.after.dash * }
1636     { year extract.before.dash * }
1637     if$
1638     volume empty$ not
1639     { ", " * volume extract.after.dash * }
1640     'skip$
1641     if$
1642     number empty$ not
1643     { "\penalty0 (" * number extract.after.dash * ")" * }
1644     'skip$
1645     if$
1646     }
1647     'skip$
1648   if$
1649   }
1650   'skip$
1651 if$
1652 }
1653

```

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。

```

1654 FUNCTION {check.url}
1655 { url empty$ not
1656   { "\url{" url * "}" * 'entry.url :=
1657     #1 'entry.is.electronic :=
1658   }
1659   { howpublished empty$ not
1660     { howpublished #1 #5 substring$ "\url{" =
1661       { howpublished 'entry.url :=
1662         #1 'entry.is.electronic :=
1663       }
1664       'skip$
1665     if$
1666   }
1667   { note empty$ not
1668     { note #1 #5 substring$ "\url{" =
1669       { note 'entry.url :=
1670         #1 'entry.is.electronic :=
1671       }
1672       'skip$
1673     if$
1674   }
1675   'skip$
1676 if$

```



```

1677     }
1678     if$
1679   }
1680   if$
1681 }
1682
1683 FUNCTION {format.url}
1684 { entry.url
1685 }
1686
1687 FUNCTION {output.url}
1688 { entry.url empty$ not
1689   { new.block
1690     entry.url output
1691   }
1692   'skip$
1693   if$
1694 }
1695

```

需要检测 DOI 是否已经包含在 URL 中。

```

1696 FUNCTION {check.doi}
1697 { doi empty$ not
1698   { #1 'entry.is.electronic := }
1699   'skip$
1700   if$
1701 }
1702
1703 FUNCTION {is.in.url}
1704 { 's :=
1705   s empty$
1706   { #1 }
1707   { entry.url empty$
1708     { #0 }
1709     { s text.length$ 'len :=
1710       entry.url text.length$ 'charptr :=
1711       { entry.url charptr len substring$ s = not
1712         charptr #0 >
1713         and
1714       }
1715       { charptr #1 - 'charptr := }
1716       while$
1717       charptr
1718     }
1719     if$
1720   }
1721   if$
1722 }
1723
1724 FUNCTION {format.doi}
1725 { ""
1726   doi empty$ not
1727   { "" 's :=
1728     doi 't :=
1729     #0 'numnames :=

```

```

1730     { t empty$ not}
1731     { t #1 #1 substring$ 'tmp.str :=
1732     tmp.str "," = tmp.str " " = or t #2 #1 substring$ empty$ or
1733     { t #2 #1 substring$ empty$
1734     { s tmp.str * 's := }
1735     'skip$
1736     if$
1737     s empty$ s is.in.url or
1738     'skip$
1739     { numnames #1 + 'numnames :=
1740     numnames #1 >
1741     { ", " * }
1742     { "DOI: " * }
1743     if$
1744     "\doi{" s * "}" * *
1745     }
1746     if$
1747     "" 's :=
1748     }
1749     { s tmp.str * 's := }
1750     if$
1751     t #2 global.max$ substring$ 't :=
1752     }
1753     while$
1754     }
1755     'skip$
1756 if$
1757 }
1758
1759 FUNCTION {output.doi}
1760 { doi empty$ not show.doi and
1761   show.english.translation entry.lang lang.zh = and not and
1762   { new.block
1763     format.doi output
1764   }
1765   'skip$
1766 if$
1767 }
1768
1769 FUNCTION {check.electronic}
1770 { "" 'entry.url :=
1771   #0 'entry.is.electronic :=
1772   'check.doi
1773   'skip$
1774 if$
1775   'check.url
1776   'skip$
1777 if$
1778 medium empty$ not
1779   { medium "MT" = medium "DK" = or medium "CD" = or medium "OL" = or
1780     { #1 'entry.is.electronic := }
1781     'skip$
1782   if$
1783   }
1784   'skip$

```

```

1785 if$
1786 }
1787
1788 FUNCTION {format.eprint}
1789 { ""
1790   archivePrefix empty$ not
1791   { archivePrefix * ": " *
1792     "\eprint{https://" *
1793     archivePrefix "l" change.case$ * ".org/abs/" * eprint * "}" *
1794     eprint * "}" *
1795   }
1796   { eprint }
1797 if$
1798 }
1799
1800 FUNCTION {output.eprint}
1801 { show.preprint eprint empty$ not and
1802   { new.block
1803     format.eprint output
1804   }
1805   'skip$
1806 if$
1807 }
1808
1809 FUNCTION {format.note}
1810 { note empty$ not show.note and
1811   { note }
1812   { "" }
1813 if$
1814 }
1815
1816 FUNCTION {output.translation}
1817 { show.english.translation entry.lang lang.zh = and
1818   { translation empty$ not
1819     { translation }
1820     { "[English translation missing!]" }
1821   if$
1822   " (in Chinese)" * output
1823   write$
1824   format.doi duplicate$ empty$ not
1825     { newline$
1826       write$
1827     }
1828   'pop$
1829   if$
1830   " \\" write$
1831   newline$
1832   "(" write$
1833   ""
1834   before.all 'output.state :=
1835   }
1836   'skip$
1837 if$
1838 }
1839

```

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

```
1840 FUNCTION {empty.misc.check}
1841 { author empty$ title empty$
1842   year empty$
1843   and and
1844   key empty$ not and
1845   { "all relevant fields are empty in " cite$ * warning$ }
1846   'skip$
1847   if$
1848 }
1849
```

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

```
1850 FUNCTION {monograph}
1851 { output.bibitem
1852   output.translation
1853   author empty$ not
1854     { format.authors }
1855     { editor empty$ not
1856       { format.editors }
1857       { "empty author and editor in " cite$ * warning$
1858 (*authoryear)
1859       bbl.anonymous
1860 </authoryear)
1861 (*numerical)
1862 ""
1863 </numerical)
1864   }
1865   if$
1866   }
1867   if$
1868   output
1869 (*authoryear)
1870   period.between.author.year
1871   'new.sentence
1872   'skip$
1873   if$
1874   format.year "year" output.check
1875 </authoryear)
```

```

1876 new.block
1877 format.series.vol.num.title "title" output.check
1878 "M" set.entry.mark
1879 format.mark "" output.after
1880 new.block
1881 format.translators output
1882 new.sentence
1883 format.edition output
1884 new.block
1885 format.address.publisher output
1886 (*numerical)
1887 format.year "year" output.check
1888 (/numerical)
1889 format.pages bbl.colon output.after
1890 format.urldate "" output.after
1891 output.url
1892 output.doi
1893 new.block
1894 format.note output
1895 fin.entry
1896 }
1897

```

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

```

1898 FUNCTION {incollection}
1899 { output.bibitem
1900   output.translation
1901   format.authors output
1902   author format.key output
1903   (*authoryear)
1904   period.between.author.year
1905     'new.sentence
1906     'skip$
1907   if$
1908   format.year "year" output.check
1909 } (/authoryear)
1910 new.block
1911 format.title "title" output.check
1912 "M" set.entry.mark
1913 format.mark "" output.after
1914 new.block
1915 format.translators output
1916 new.slash
1917 format.editors output
1918 new.block

```

```

1919 format.series.vol.num.booktitle "booktitle" output.check
1920 new.block
1921 format.edition output
1922 new.block
1923 format.address.publisher output
1924 (*numerical)
1925 format.year "year" output.check
1926 (/numerical)
1927 format.extracted.pages bbl.colon output.after
1928 format.urldate "" output.after
1929 output.url
1930 output.doi
1931 new.block
1932 format.note output
1933 fin.entry
1934 }
1935

```

B.5.3 连续出版物

```

1936 FUNCTION {periodical}
1937 { output.bibitem
1938   output.translation
1939   format.authors output
1940   author format.key output
1941   (*authoryear)
1942   period.between.author.year
1943     'new.sentence
1944     'skip$
1945   if$
1946   format.year "year" output.check
1947 (/authoryear)
1948 new.block
1949 format.title "title" output.check
1950 "J" set.entry.mark
1951 format.mark "" output.after
1952 new.block
1953 format.periodical.year.volume.number output
1954 new.block
1955 format.address.publisher output
1956 (*numerical)
1957 format.date "year" output.check
1958 (/numerical)
1959 format.urldate "" output.after
1960 output.url
1961 output.doi
1962 new.block
1963 format.note output
1964 fin.entry
1965 }
1966

```

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.

```
1967 FUNCTION {article}
1968 { output.bibitem
1969   output.translation
1970   format.authors output
1971   author format.key output
1972 <{*authoryear}
1973   period.between.author.year
1974     'new.sentence
1975     'skip$
1976   if$
1977   format.year "year" output.check
1978 </authoryear>
1979   new.block
1980   title.in.journal
1981     { format.title "title" output.check
1982       "J" set.entry.mark
1983       format.mark "" output.after
1984       new.block
1985     }
1986     'skip$
1987   if$
1988   format.journal "journal" output.check
1989 <{*numerical}
1990   format.date "year" output.check
1991 </numerical>
1992   format.journal.volume output
1993   format.journal.number "" output.after
1994   format.journal.pages "" output.after
1995   format.urldate "" output.after
1996   output.url
1997   output.doi
1998   new.block
1999   format.note output
2000   fin.entry
2001 }
2002
```

B.5.5 专利文献

number 域也可以用来表示专利号。

```
2003 FUNCTION {patent}
2004 { output.bibitem
2005   output.translation
```

```

2006 format.authors output
2007 author format.key output
2008 (*authoryear)
2009 period.between.author.year
2010     'new.sentence
2011     'skip$
2012 if$
2013 format.year "year" output.check
2014 </authoryear>
2015 new.block
2016 format.title "title" output.check
2017 "P" set.entry.mark
2018 format.mark "" output.after
2019 new.block
2020 format.date "year" output.check
2021 format.urldate "" output.after
2022 output.url
2023 output.doi
2024 new.block
2025 format.note output
2026 fin.entry
2027 }
2028

```

B.5.6 电子资源

```

2029 FUNCTION {electronic}
2030 { #1 #1 check.electronic
2031   #1 'entry.is.electronic :=
2032   output.bibitem
2033   output.translation
2034   format.authors output
2035   author format.key output
2036 (*authoryear)
2037   period.between.author.year
2038     'new.sentence
2039     'skip$
2040   if$
2041   format.year "year" output.check
2042 </authoryear>
2043   new.block
2044   format.series.vol.num.title "title" output.check
2045   "EB" set.entry.mark
2046   format.mark "" output.after
2047   new.block
2048   format.address.publisher output
2049 (*numerical)
2050   date empty$
2051   { format.date output }
2052   'skip$
2053   if$
2054 </numerical>
2055   format.pages bbl.colon output.after
2056   format.editdate "" output.after
2057   format.urldate "" output.after

```



```

2058 output.url
2059 output.doi
2060 new.block
2061 format.note output
2062 fin.entry
2063 }
2064

```

B.5.7 预印本

```

2065 FUNCTION {preprint}
2066 { output.bibitem
2067   output.translation
2068   author empty$ not
2069     { format.authors }
2070     { editor empty$ not
2071       { format.editors }
2072       { "empty author and editor in " cite$ * warning$
2073 (*authoryear)
2074       bbl.anonymous
2075 </authoryear>
2076 (*numerical)
2077       ""
2078 </numerical>
2079     }
2080     if$
2081   }
2082   if$
2083   output
2084 (*authoryear)
2085   period.between.author.year
2086   'new.sentence
2087   'skip$
2088   if$
2089   format.year "year" output.check
2090 </authoryear>
2091   new.block
2092   title.in.journal
2093     { format.series.vol.num.title "title" output.check
2094       "Z" set.entry.mark
2095       format.mark "" output.after
2096       new.block
2097     }
2098     'skip$
2099   if$
2100   format.translators output
2101   new.sentence
2102   format.edition output
2103   new.block
2104   output.eprint
2105 (*numerical)
2106   format.year "year" output.check
2107 </numerical>
2108   format.pages bbl.colon output.after
2109   format.urldate "" output.after
2110   output.url

```

```

2111 new.block
2112 format.note output
2113 fin.entry
2114 }
2115

```

B.5.8 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。

```

2116 FUNCTION {misc}
2117 { journal empty$ not
2118   'article
2119   { booktitle empty$ not
2120     'incollection
2121     { publisher empty$ not
2122       'monograph
2123       { eprint empty$ not show.preprint and
2124         'preprint
2125         { entry.is.electronic
2126           'electronic
2127           { "Z" set.entry.mark
2128             monograph
2129           }
2130         }
2131       }
2132     }
2133   }
2134   if$
2135 }
2136 if$
2137 }
2138 if$
2139 empty.misc.check
2140 }
2141
2142 FUNCTION {archive}
2143 { "A" set.entry.mark
2144   misc
2145 }
2146

```

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

```

2147 FUNCTION {book} { monograph }
2148

```

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

Required: title

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

```
2149 FUNCTION {booklet} { book }
2150
2151 FUNCTION {collection}
2152 { "G" set.entry.mark
2153   monograph
2154 }
2155
2156 FUNCTION {database}
2157 { "DB" set.entry.mark
2158   electronic
2159 }
2160
2161 FUNCTION {dataset}
2162 { "DS" set.entry.mark
2163   electronic
2164 }
2165
```

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。

```
2166 FUNCTION {inbook} { book }
2167
```

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

```
2168 FUNCTION {inproceedings}
2169 { "C" set.entry.mark
2170   incollection
2171 }
2172
```

The conference function is included for Scribe compatibility.

```
2173 FUNCTION {conference} { inproceedings }
2174
2175 FUNCTION {map}
2176 { "CM" set.entry.mark
2177   misc
2178 }
2179
```

A manual is technical documentation.

Required: title

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

```
2180 FUNCTION {manual} { monograph }
```

```
2181
```

A mastersthesis is a Master's thesis.

Required: author, title, school, year

Optional: type, address, month, note

```
2182 FUNCTION {mastersthesis}
```

```
2183 <!*thu>
```

```
2184 { "D" set.entry.mark
```

```
2185 </!thu>
```

```
2186 <*thu>
```

```
2187 { lang.zh entry.lang =
```

```
2188   { " 硕士学位论文" }
```

```
2189   { "D" }
```

```
2190   if$
```

```
2191   set.entry.mark
```

```
2192 </thu>
```

```
2193   monograph
```

```
2194 }
```

```
2195
```

```
2196 FUNCTION {newspaper}
```

```
2197 { "N" set.entry.mark
```

```
2198   article
```

```
2199 }
```

```
2200
```

```
2201 FUNCTION {online}
```

```
2202 { "EB" set.entry.mark
```

```
2203   electronic
```

```
2204 }
```

```
2205
```

A phdthesis is like a mastersthesis.

Required: author, title, school, year

Optional: type, address, month, note

```
2206 <!*thu>
```

```
2207 FUNCTION {phdthesis} { mastersthesis }
```

```
2208 </!thu>
```

```
2209 <*thu>
```

```
2210 FUNCTION {phdthesis}
```

```
2211 { lang.zh entry.lang =
```

```
2212   { " 博士学位论文" }
```

```
2213   { "D" }
```

```
2214   if$
```

```
2215   set.entry.mark
```

```
2216   monograph
```

```
2217 }
```

```
2218 </thu>
```

```
2219
```

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

```
2220 FUNCTION {proceedings}
2221 { "C" set.entry.mark
2222   monograph
2223 }
2224
2225 FUNCTION {software}
2226 { "CP" set.entry.mark
2227   electronic
2228 }
2229
2230 FUNCTION {standard}
2231 { "S" set.entry.mark
2232   misc
2233 }
2234
```

A techreport is a technical report.

Required: author, title, institution, year

Optional: type, number, address, month, note

```
2235 FUNCTION {techreport}
2236 { "R" set.entry.mark
2237   misc
2238 }
2239
```

An unpublished is something that hasn't been published.

Required: author, title, note

Optional: month, year

```
2240 FUNCTION {unpublished} { misc }
2241
```

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

```
2242 FUNCTION {default.type} { misc }
2243
```

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

2244 MACRO {jan} {"January"}
 2245
 2246 MACRO {feb} {"February"}
 2247
 2248 MACRO {mar} {"March"}
 2249
 2250 MACRO {apr} {"April"}
 2251
 2252 MACRO {may} {"May"}
 2253
 2254 MACRO {jun} {"June"}
 2255
 2256 MACRO {jul} {"July"}
 2257
 2258 MACRO {aug} {"August"}
 2259
 2260 MACRO {sep} {"September"}
 2261
 2262 MACRO {oct} {"October"}
 2263
 2264 MACRO {nov} {"November"}
 2265
 2266 MACRO {dec} {"December"}
 2267

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

2268 MACRO {acmcs} {"ACM Computing Surveys"}
 2269
 2270 MACRO {acta} {"Acta Informatica"}
 2271
 2272 MACRO {cacm} {"Communications of the ACM"}
 2273
 2274 MACRO {ibmjrd} {"IBM Journal of Research and Development"}
 2275
 2276 MACRO {ibmsj} {"IBM Systems Journal"}
 2277
 2278 MACRO {ieeese} {"IEEE Transactions on Software Engineering"}
 2279
 2280 MACRO {ieeetc} {"IEEE Transactions on Computers"}
 2281
 2282 MACRO {ieeetcad}
 2283 {"IEEE Transactions on Computer-Aided Design of Integrated Circuits"}
 2284
 2285 MACRO {ipl} {"Information Processing Letters"}
 2286
 2287 MACRO {jacm} {"Journal of the ACM"}
 2288
 2289 MACRO {jcss} {"Journal of Computer and System Sciences"}
 2290

```

2291 MACRO {scp} {"Science of Computer Programming"}
2292
2293 MACRO {sicomp} {"SIAM Journal on Computing"}
2294
2295 MACRO {tocs} {"ACM Transactions on Computer Systems"}
2296
2297 MACRO {tods} {"ACM Transactions on Database Systems"}
2298
2299 MACRO {tog} {"ACM Transactions on Graphics"}
2300
2301 MACRO {toms} {"ACM Transactions on Mathematical Software"}
2302
2303 MACRO {toois} {"ACM Transactions on Office Information Systems"}
2304
2305 MACRO {toplas} {"ACM Transactions on Programming Languages and Systems"}
2306
2307 MACRO {tcs} {"Theoretical Computer Science"}
2308

```

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`.

```

2309 FUNCTION {sortify}
2310 { purify$
2311 "l" change.case$
2312 }
2313

```

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

```

2314 FUNCTION {chop.word}
2315 { 's :=
2316 'len :=
2317 s #1 len substring$ =
2318 { s len #1 + global.max$ substring$ }
2319 's
2320 if$
2321 }
2322

```

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$(s)
  if numnames > 1 then
    if numnames > 4 then
      namesleft := 3
    else
      namesleft := numnames
    nameptr := 1
    namerresult := ""
    while namesleft > 0
      do
        if (name_ptr = numnames) and
          format.name$(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$` in desperation when this happens. The resulting label has the year part, but not the name part, `purify$ed` (`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
```

```
2323 FUNCTION {format.lab.names}
2324 { 's :=
2325 s #1 "{vv~}{ll}{, jj}{, ff}" format.name$ 't :=
2326 t get.str.lang 'name.lang :=
2327 name.lang lang.en =
2328   { t #1 "{vv~}{ll}" format.name$}
2329   { t #1 "{ll}{ff}" format.name$}
2330 if$
2331 s num.names$ #1 >
2332   { bbl.space * citation.et.al * }
2333   'skip$
2334 if$
2335 }
2336
2337 FUNCTION {author.key.label}
2338 { author empty$
2339   { key empty$
2340     { cite$ #1 #3 substring$ }
2341     'key
2342   if$
2343   }
2344 { author format.lab.names }
```

```

2345 if$
2346 }
2347
2348 FUNCTION {author.editor.key.label}
2349 { author empty$
2350   { editor empty$
2351     { key empty$
2352       { cite$ #1 #3 substring$ }
2353       'key
2354       if$
2355     }
2356     { editor format.lab.names }
2357     if$
2358   }
2359   { author format.lab.names }
2360   if$
2361 }
2362
2363 FUNCTION {author.key.organization.label}
2364 { author empty$
2365   { key empty$
2366     { organization empty$
2367       { cite$ #1 #3 substring$ }
2368       { "The " #4 organization chop.word #3 text.prefix$ }
2369       if$
2370     }
2371     'key
2372     if$
2373   }
2374   { author format.lab.names }
2375   if$
2376 }
2377
2378 FUNCTION {editor.key.organization.label}
2379 { editor empty$
2380   { key empty$
2381     { organization empty$
2382       { cite$ #1 #3 substring$ }
2383       { "The " #4 organization chop.word #3 text.prefix$ }
2384       if$
2385     }
2386     'key
2387     if$
2388   }
2389   { editor format.lab.names }
2390   if$
2391 }
2392
2393 FUNCTION {calc.short.authors}
2394 { type$ "book" =
2395   type$ "inbook" =
2396   or
2397   'author.editor.key.label
2398   { type$ "collection" =
2399     type$ "proceedings" =

```

```

2400     or
2401     { editor empty$ not
2402       'editor.key.organization.label
2403       'author.key.organization.label
2404     if$
2405     }
2406     'author.key.label
2407   if$
2408 }
2409 if$
2410 'short.list :=
2411 }
2412
2413 FUNCTION {calc.label}
2414 { calc.short.authors
2415   short.list
2416   "("
2417   *
2418   format.year duplicate$ empty$
2419   short.list key field.or.null = or
2420     { pop$ "" }
2421     'skip$
2422   if$
2423   *
2424   'label :=
2425 }
2426

```

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`.

```

2427 (*authoryear)
2428 FUNCTION {sort.language.label}
2429 { entry.lang lang.zh =
2430   { lang.zh.order }
2431   { entry.lang lang.ja =
2432     { lang.ja.order }
2433     { entry.lang lang.en =
2434       { lang.en.order }
2435       { entry.lang lang.ru =
2436         { lang.ru.order }
2437         { lang.other.order }
2438         if$
2439       }
2440     }
2441   }
2442   if$
2443 }
2444 if$
2445 int.to.chr$
2446 }
2447
2448 FUNCTION {sort.format.names}
2449 { 's :=
2450   #1 'nameptr :=
2451   ""
2452   s num.names$ 'numnames :=
2453   numnames 'namesleft :=
2454   { namesleft #0 > }
2455   {
2456     s nameptr "{vv{ } }{ll{ }}{ ff{ }}{ jj{ }}" format.name$ 't :=
2457     nameptr #1 >
2458     {
2459       " " *
2460       namesleft #1 = t "others" = and
2461       { "zzzzz" * }
2462       { numnames #2 > nameptr #2 = and
2463         { "zz" * year field.or.null * " " * }
2464         'skip$
2465         if$
2466         t sortify *
2467       }
2468       if$
2469     }
2470     { t sortify * }
2471     if$
2472     nameptr #1 + 'nameptr :=
2473     namesleft #1 - 'namesleft :=
2474   }
2475   while$
2476 }
2477

```

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
2478 FUNCTION {sort.format.title}
2479 { 't :=
2480   "A " #2
2481   "An " #3
2482   "The " #4 t chop.word
2483   chop.word
2484   chop.word
2485   sortify
2486   #1 global.max$ substring$
2487 }
2488

```

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.

```

2489 FUNCTION {anonymous.sort}
2490 { entry.lang lang.zh =
2491   { "yi4 ming2" }
2492   { "anon" }
2493   if$
2494 }
2495
2496 FUNCTION {warn.empty.key}
2497 { entry.lang lang.zh =
2498   { "empty key in " cite$ * warning$ }
2499   'skip$
2500   if$
2501 }
2502
2503 FUNCTION {author.sort}
2504 { key empty$
2505   { warn.empty.key
2506     author empty$
2507     { anonymous.sort }
2508     { author sort.format.names }
2509     if$
2510   }
2511   { key sortify }
2512   if$
2513 }
2514
2515 FUNCTION {author.editor.sort}
2516 { key empty$
2517   { warn.empty.key
2518     author empty$
2519     { editor empty$
2520       { anonymous.sort }
2521       { editor sort.format.names }
2522       if$
2523     }
2524     { author sort.format.names }

```

```

2525     if$
2526   }
2527   { key sortify }
2528   if$
2529 }
2530
2531 FUNCTION {author.organization.sort}
2532 { key empty$
2533   { warn.empty.key
2534     author empty$
2535       { organization empty$
2536         { anonymous.sort }
2537         { "The " #4 organization chop.word sortify }
2538         if$
2539       }
2540       { author sort.format.names }
2541       if$
2542     }
2543     { key sortify }
2544     if$
2545   }
2546
2547 FUNCTION {editor.organization.sort}
2548 { key empty$
2549   { warn.empty.key
2550     editor empty$
2551       { organization empty$
2552         { anonymous.sort }
2553         { "The " #4 organization chop.word sortify }
2554         if$
2555       }
2556       { editor sort.format.names }
2557       if$
2558     }
2559     { key sortify }
2560     if$
2561   }
2562
2563 </authoryear>

```

顺序编码制的排序要简单得多

```

2564 (*numerical)
2565 INTEGERS { seq.num }
2566
2567 FUNCTION {init.seq}
2568 { #0 'seq.num :=}
2569
2570 FUNCTION {int.to.fix}
2571 { "000000000" swap$ int.to.str$ *
2572   #-1 #10 substring$
2573 }
2574
2575 </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!

```
2576 FUNCTION {presort}
2577 { set.entry.lang
2578   set.entry.numbered
2579   show.url show.doi check.electronic
2580   calc.label
2581   label sortify
2582   " "
2583   *
2584   ⟨*authoryear⟩
2585   sort.language.label
2586   type$ "book" =
2587   type$ "inbook" =
2588   or
2589     'author.editor.sort
2590     { type$ "collection" =
2591       type$ "proceedings" =
2592       or
2593         'editor.organization.sort
2594         'author.sort
2595       if$
2596     }
2597   if$
2598   *
2599   " "
2600   *
2601   year field.or.null sortify
2602   *
2603   " "
2604   *
2605   cite$
2606   *
2607   #1 entry.max$ substring$
2608   ⟨/authoryear⟩
2609   ⟨*numerical⟩
2610   seq.num #1 + 'seq.num :=
2611   seq.num int.to.fix
2612   ⟨/numerical⟩
2613   'sort.label :=
2614   sort.label *
2615   #1 entry.max$ substring$
2616   'sort.key$ :=
2617 }
2618
```

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

```

```

2619 STRINGS { longest.label last.label next.extra }
2620
2621 INTEGERS { longest.label.width last.extra.num number.label }
2622
2623 FUNCTION {initialize.longest.label}
2624 { "" 'longest.label :=
2625 #0 int.to.chr$ 'last.label :=
2626 "" 'next.extra :=
2627 #0 'longest.label.width :=
2628 #0 'last.extra.num :=
2629 #0 'number.label :=
2630 }
2631
2632 FUNCTION {forward.pass}
2633 { last.label label =
2634   { last.extra.num #1 + 'last.extra.num :=
2635     last.extra.num int.to.chr$ 'extra.label :=
2636   }
2637   { "a" chr.to.int$ 'last.extra.num :=

```



```

2638     "" 'extra.label :=
2639     label 'last.label :=
2640   }
2641   if$
2642   number.label #1 + 'number.label :=
2643 }
2644
2645 FUNCTION {reverse.pass}
2646 { next.extra "b" =
2647   { "a" 'extra.label := }
2648   'skip$
2649   if$
2650   extra.label 'next.extra :=
2651   extra.label
2652   duplicate$ empty$
2653     'skip$
2654     { "{\natexlab{" swap$ * "}" * }
2655     if$
2656     'extra.label :=
2657     label extra.label * 'label :=
2658 }
2659
2660 FUNCTION {bib.sort.order}
2661 { sort.label 'sort.key$ :=
2662 }
2663

```

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.

```

2664 FUNCTION {begin.bib}
2665 { preamble$ empty$
2666   'skip$
2667   { preamble$ write$ newline$ }
2668   if$
2669   "\begin{thebibliography}{ number.label int.to.str$ * "}" *
2670   write$ newline$
2671   "\providecommand{\natexlab}[1]{#1}"
2672   write$ newline$
2673   "\providecommand{\url}[1]{#1}"
2674   write$ newline$
2675   "\expandafter\ifx\csname urlstyle\endcsname\relax\else"
2676   write$ newline$
2677   " \urlstyle{same}\fi"
2678   write$ newline$
2679   "\expandafter\ifx\csname href\endcsname\relax"
2680   write$ newline$
2681   " \DeclareUrlCommand\doi{\urlstyle{rm}}"

```

```

2682 write$ newline$
2683 " \def\eprint#1#2{#2}"
2684     write$ newline$
2685 "\else"
2686 write$ newline$
2687 " \def\doi#1{\href{https://doi.org/#1}{\nolinkurl{#1}}}"
2688 write$ newline$
2689 " \let\eprint\href"
2690     write$ newline$
2691 "\fi"
2692     write$ newline$
2693 }
2694

```

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

```

2695 FUNCTION {end.bib}
2696 { newline$
2697   "\end{thebibliography}" write$ newline$
2698 }
2699

```

B.10 Main execution

Now we read in the .BIB entries.

```

2700 READ
2701
2702 EXECUTE {init.state.consts}
2703
2704 EXECUTE {load.config}
2705
2706 < *numerical >
2707 EXECUTE {init.seq}
2708
2709 < /numerical >
2710 ITERATE {presort}
2711

```

And now we can sort

```

2712 SORT
2713
2714 EXECUTE {initialize.longest.label}
2715
2716 ITERATE {forward.pass}
2717
2718 REVERSE {reverse.pass}
2719
2720 ITERATE {bib.sort.order}
2721
2722 SORT
2723
2724 EXECUTE {begin.bib}
2725

```

Now we produce the output for all the entries

```
2726 ITERATE {call.type$}  
2727  
2728 EXECUTE {end.bib}  
2729 </authorityyear | numerical>
```