

GB/T 7714-2015 BibT_EX style

Zeping Lee*

2020/03/14 v2.0.1

摘要

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

1 简介

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

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

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

2 使用方法

按照国标的规定，参考文献的标注体系分为“顺序编码制”和“著者-出版年制”。用户应在导言区调用宏包 gbt7714，并且使用 \bibliographystyle 命令选择参考文献表的样式，比如：

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

或者

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

*zepinglee AT gmail.com

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

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

```
\citestyle{numbers}
```

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

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

这些参数会传给 `natbib` 处理。

若需要标出引文的页码，可以标在 `\cite` 的可选参数中，如 `\cite[42]{knuth84}`。更多的引用标注方法可以参考 `natbib` 宏包的使用说明^[3]。

使用时需要注意以下几点：

- `.bib` 数据库应使用 UTF-8 编码。
- 使用著者-出版年制参考文献表时，中文的文献必须在 `key` 域填写作者姓名的拼音，才能按照拼音排序，详见第 5 节。

3 文献类型

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

4 著录项目

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

author 主要责任者

title 题名

mark* 文献类型标识

medium* 载体类型标识

translator* 译者

表 1: 全部文献类型

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

editor 编辑

organization 组织（用于会议）

booktitle 图书题名

series 系列

journal 期刊题名

edition 版本

address 出版地

publisher 出版者

school 学校（用于 phdthesis）

institution 机构（用于 techreport）

year 出版年

volume 卷

number 期（或者专利号）

pages 引文页码

date* 更新或修改日期

urldate* 引用日期

url 获取和访问路径
doi 数字对象唯一标识符
language* 语言
key 拼音（用于排序）

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

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

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

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

5 文献列表的排序

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

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

注意名字之间需要额外的空格，比如“张三,李四”要排在“张三丰”前面。

6 自定义样式

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

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

表 2: 参考文献表样式的配置参数

参数值	默认值	功能
uppercase.name	#1	将著者姓名转为大写
max.num.authors	#3	输出著者的最多数
period.between.author.year	#0	著者和年份之间使用句点连接
sentence.case.title	#1	将西文的题名转为 sentence case
link.title	#0	在题名上添加 url 的超链接
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”
space.before.pages	#0	页码前有空白
only.start.page	#0	只显示起始页码
show.url	#1	显示 url
show.doi	#1	显示 doi
show.note	#0	显示 note 域的信息

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

7 相关工作

TeX 社区也有其他关于 GB/T 7714 系列参考文献标准的工作。2005 年吴凯^[7]发布了基于 GB/T 7714-2005 的 BibTeX 样式，支持顺序编码制和著者出版年制两种风格。李志奇^[8]发布了严格遵循 GB/T 7714-2005 的 BibLaTeX 的样式。胡海星^[9]提供了另一个 BibTeX 实现，还给每行 bst 代码写了 java 语言注释。沈周^[10]基于 biblatex-casperiector^[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] 中国标准化委员会. 信息与文献 参考文献著录规则: GB/T 7714-2015[S]. 北京: 中国标准出版社, 2015.
- [2] PATASHNIK O. BibTeXing[M/OL]. 1988. <http://mirrors.ctan.org/biblio/bibtex/base/btxdoc.pdf>.

- [3] DALY P W. Natural sciences citations and references[M/OL]. 1999. <http://mirrors.ctan.org/macros/latex/contrib/natbib/natbib.pdf>.
- [4] PATASHNIK O. Designing BibTeX styles[M/OL]. 1988. <http://mirrors.ctan.org/biblio/bibtex/base/btxhak.pdf>.
- [5] MARKEY N. Tame the beast[M/OL]. 2003. http://mirrors.ctan.org/info/bibtex/tamethebeast/ttb_en.pdf.
- [6] MITTELBAACH F, GOOSSENS M, BRAAMS J, et al. The L^AT_EX companion [M]. 2nd ed. Reading, MA, USA: Addison-Wesley, 2004.
- [7] 吴凯. 发布 GBT7714-2005.bst version1 Beta 版[EB/OL]. 2006. <http://bbs.ctex.org/forum.php?mod=viewthread&tid=33591> (not accessible).
- [8] 李志奇. 基于 biblatex 的符合 GBT7714-2005 的中文文献生成工具[EB/OL]. 2013. <http://bbs.ctex.org/forum.php?mod=viewthread&tid=74474> (not accessible).
- [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@cites@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@mbox}%
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@mbox
95 }%
96 }%
97 }{%
98 \@tempwattrue
99 \@ifnum{\NAT@merge>\@ne}{\@ifnum{\NAT@last@num=\NAT@num\relax}{\@tempwafalse}{}}{%
100 \if@tempswa\NAT@citea@mbox\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@mbox{\NAT@super{kern\NAT@@open}%
108 \fi
109 \if*#1*\else#1\NAT@spacechar\fi

```

```

110      \NAT@mbox{\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@mbox{\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
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}}

```

将页码放在括号外边，并且置于上标。

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@writtentruer
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   space.before.pages
290   only.start.page
291   show.url
292   show.doi
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）：

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

在标题添加超链接：

```
325 #0 'link.title :=
326 <*linktitle>
327 #1 'link.title :=
328 </linktitle>
```

期刊是否含标题：

```
329 <!*title-in-journal&!npr>
330 #1 'title.in.journal :=
331 </!*title-in-journal&!npr>
332 <*title-in-journal | npr>
333 #0 'title.in.journal :=
334 </title-in-journal | npr>
```

著录文献类型标识（比如“[M/OL]”）：

```
335 #1 'show.mark :=
336 <*nomark>
337 #0 'show.mark :=
338 </nomark>
```

是否显示载体类型标识（比如“/OL”）：

```
339 #1 'show.medium.type :=
340 <*no.medium.type>
341 #0 'show.medium.type :=
342 </no.medium.type>
```

使用“//”表示析出文献

```

343 #1 'slash.for.extraction :=
344 (*noslash)
345 #0 'slash.for.extraction :=
346 (/noslash)

```

使用“In:”表示析出文献

```

347 #0 'in.booktitle :=

```

期刊名使用缩写:

```

348 (*!abbreviate-journal&!npr)
349 #0 'abbreviate.journal :=
350 (/!abbreviate-journal&!npr)
351 (*abbreviate-journal | npr)
352 #1 'abbreviate.journal :=
353 (/abbreviate-journal | npr)

```

期刊名使用斜体:

```

354 #0 'italic.journal :=
355 (*italicjournal)
356 #1 'italic.journal :=
357 (/italicjournal)

```

期刊的卷使用粗体:

```

358 #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.”:

```

359 (*!noslsn&!thu&!ustc)
360 #1 'show.missing.address.publisher :=
361 (/!noslsn&!thu&!ustc)
362 (*noslsn | thu | ustc)
363 #0 'show.missing.address.publisher :=
364 (/noslsn | thu | ustc)

```

页码是否只含起始页:

```

365 (*!space-begore-pages&!npr)
366 #0 'space.before.pages :=
367 (/!space-begore-pages&!npr)
368 (*space-begore-pages | npr)
369 #1 'space.before.pages :=
370 (/space-begore-pages | npr)

```

页码前是否有空白:

```

371 (*!only-start-page&!npr)
372 #0 'only.start.page :=
373 (/!only-start-page&!npr)
374 (*only-start-page | npr)
375 #1 'only.start.page :=
376 (/only-start-page | npr)

```

是否著录 URL:

```

377 #1 'show.url :=
378 (*nourl)
379 #0 'show.url :=
380 (/nourl)

```


是否著录 DOI:

```
381 <!*!nodoi&!2005>
382 #1 'show.doi :=
383 </!nodoi&!2005>
384 <!*nodoi | 2005>
385 #0 'show.doi :=
386 </nodoi | 2005>
```

在每一条文献最后输出注释 (note) 的内容:

```
387 #0 'show.note :=
```

中文文献是否显示英文翻译

```
388 <!*show-english-translation&!npr>
389 #0 'show.english.translation :=
390 </!show-english-translation&!npr>
391 <!*show-english-translation | npr>
392 #1 'show.english.translation :=
393 </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:

```
394 <!*authoryear>
395 #1 'lang.zh.order :=
396 #2 'lang.ja.order :=
397 #3 'lang.en.order :=
398 #4 'lang.ru.order :=
399 #5 'lang.other.order :=
400 </authoryear>
401 }
402
```

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.

```
403 ENTRY
404 { address
405   author
406   booktitle
407   date
408   doi
409   edition
410   editor
411   howpublished
412   institution
413   journal
414   key
415   language
416   mark
```

```

417     medium
418     note
419     number
420     organization
421     pages
422     publisher
423     school
424     series
425     title
426     translator
427     translation
428     url
429     urldate
430     volume
431     year
432 }
433 { 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.

```

434 { label extra.label sort.label short.list entry.mark entry.url }
435

```

B.3 Entry functions

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

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

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

To tell which separator is needed, we maintain an `output.state`. It will be one of

these values: before.all just after the \bibitem mid.sentence in the middle of a sentence: comma needed if more sentence is output after.sentence just after a sentence: period needed after.block just after a block (and sentence): period and \newblock needed. Note: These styles don't use after.sentence

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

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

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

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

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

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

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

```
output.check(s,t) ==
```

```

BEGIN
  if empty$(s) then
    warning$("empty " * t * " in " * cite$)
  else output.nonnull(s)
  fi
END

```

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

```

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

```

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

```

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

```

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

```

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

```

```

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

```

END

```
436 INTEGERS { output.state before.all mid.sentence after.sentence after.block after.slash }
437
438 INTEGERS { lang.zh lang.ja lang.en lang.ru lang.other }
439
440 INTEGERS { charptr len }
441
442 FUNCTION {init.state.consts}
443 { #0 'before.all :=
444   #1 'mid.sentence :=
445   #2 'after.sentence :=
446   #3 'after.block :=
447   #4 'after.slash :=
448   #3 'lang.zh :=
449   #4 'lang.ja :=
450   #1 'lang.en :=
451   #2 'lang.ru :=
452   #0 'lang.other :=
453 }
454
```

下面是一些常量的定义

```
455 FUNCTION {bbl.anonymous}
456 { entry.lang lang.zh =
457   { "佚名" }
458   { "Anon" }
459   if$
460 }
461
462 FUNCTION {bbl.space}
463 { entry.lang lang.zh =
464   { "\ " }
465   { " " }
466   if$
467 }
468
469 FUNCTION {bbl.et.al}
470 { entry.lang lang.zh =
471   { "等" }
472   { entry.lang lang.ja =
473     { "他" }
474     { entry.lang lang.ru =
475       { "и др" }
476       { "et~al." }
477       if$
478     }
479     if$
480   }
481   if$
482 }
483
484 FUNCTION {citation.et.al}
485 { bbl.et.al }
486
```

```

487 FUNCTION {bbl.colon} { ":" " }
488
489 < *2015 >
490 FUNCTION {bbl.wide.space} { "\quad " }
491 < /2015 >
492 < *2005 >
493 FUNCTION {bbl.wide.space} { "\ " }
494 < /2005 >
495
496 < *!thu >
497 FUNCTION {bbl.slash} { "//\allowbreak " }
498 < /!thu >
499 < *thu >
500 FUNCTION {bbl.slash} { " //" " }
501 < /thu >
502
503 FUNCTION {bbl.sine.loco}
504 { entry.lang lang.zh =
505   { "[出版地不详]" }
506   { "[S.l.]" }
507   if$
508 }
509
510 FUNCTION {bbl.sine.nomine}
511 { entry.lang lang.zh =
512   { "[出版者不详]" }
513   { "[s.n.]" }
514   if$
515 }
516
517 FUNCTION {bbl.sine.loco.sine.nomine}
518 { entry.lang lang.zh =
519   { "[出版地不详: 出版者不详]" }
520   { "[S.l.: s.n.]" }
521   if$
522 }
523

```

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

```

524 FUNCTION {not}
525 {   { #0 }
526     { #1 }
527   if$
528 }
529
530 FUNCTION {and}
531 {   'skip$
532     { pop$ #0 }
533   if$
534 }
535
536 FUNCTION {or}

```

```

537 { { pop$ #1 }
538     'skip$
539 if$
540 }
541

```

the variables s and t are temporary string holders

```

542 STRINGS { s t }
543
544 FUNCTION {output.nonnull}
545 { 's :=
546   output.state mid.sentence =
547   { ", " * write$ }
548   { output.state after.block =
549     { add.period$ write$
550       newline$
551       "\newblock " write$
552     }
553     { output.state before.all =
554       'write$
555       { output.state after.slash =
556         { bbl.slash * write$
557           newline$
558         }
559         { add.period$ " " * write$ }
560       }
561     }
562   if$
563 }
564 if$
565 mid.sentence 'output.state :=
566 }
567 if$
568 s
569 }
570
571 FUNCTION {output}
572 { duplicate$ empty$
573   'pop$
574   'output.nonnull
575   if$
576 }
577
578 FUNCTION {output.after}
579 { 't :=
580   duplicate$ empty$
581   'pop$
582   { 's :=
583     output.state mid.sentence =
584     { t * write$ }
585     { output.state after.block =
586       { add.period$ write$
587         newline$
588         "\newblock " write$
589       }

```

```

590         { output.state before.all =
591           'write$
592           { output.state after.slash =
593             { bbl.slash * write$ }
594             { add.period$ " " * write$ }
595             if$
596           }
597           if$
598         }
599         if$
600         mid.sentence 'output.state :=
601       }
602       if$
603       s
604     }
605   if$
606 }
607
608 FUNCTION {output.check}
609 { 't :=
610   duplicate$ empty$
611   { pop$ "empty " t * " in " * cite$ * warning$ }
612   'output.nonnull
613   if$
614 }
615

```

This function finishes all entries.

```

616 FUNCTION {fin.entry}
617 { add.period$
618   write$
619   show.english.translation entry.lang lang.zh = and
620   { ""
621     write$
622   }
623   'skip$
624   if$
625   newline$
626 }
627
628 FUNCTION {new.block}
629 { output.state before.all =
630   'skip$
631   { output.state after.slash =
632     'skip$
633     { after.block 'output.state := }
634     if$
635   }
636   if$
637 }
638
639 FUNCTION {new.sentence}
640 { output.state after.block =
641   'skip$
642   { output.state before.all =

```



```

643         'skip$
644         { output.state after.slash =
645           'skip$
646           { after.sentence 'output.state := }
647         if$
648       }
649     if$
650   }
651 if$
652 }
653
654 FUNCTION {new.slash}
655 { output.state before.all =
656   'skip$
657   { slash.for.extraction
658     { after.slash 'output.state := }
659     { after.block 'output.state := }
660   if$
661   }
662 if$
663 }
664

```

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.

```

665 FUNCTION {new.block.checka}
666 { empty$
667   'skip$
668   'new.block
669 if$
670 }
671
672 FUNCTION {new.block.checkb}
673 { empty$
674   swap$ empty$
675   and
676   'skip$
677   'new.block
678 if$
679 }
680

```

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

```

681 FUNCTION {new.sentence.checka}
682 { empty$
683   'skip$
684   'new.sentence
685 if$
686 }
687
688 FUNCTION {new.sentence.checkb}
689 { empty$
690   swap$ empty$

```

```

691 and
692     'skip$
693     'new.sentence
694 if$
695 }
696

```

B.4 Formatting chunks

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

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

```

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

```

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

```

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

```

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

```

697 FUNCTION {field.or.null}
698 { duplicate$ empty$
699   { pop$ "" }
700   'skip$
701   if$
702 }
703
704 FUNCTION {italicize}
705 { duplicate$ empty$
706   { pop$ "" }
707   { "\textit{" swap$ * "}" * }
708   if$

```

```
709 }
710
```

B.4.1 Detect Language

```
711 INTEGERS { byte second.byte }
712
713 INTEGERS { char.lang tmp.lang }
714
715 STRINGS { tmp.str }
716
717 FUNCTION {get.str.lang}
718 { 'tmp.str :=
719   lang.other 'tmp.lang :=
720   #1 'charptr :=
721   tmp.str text.length$ #1 + 'len :=
722   { charptr len < }
723   { tmp.str charptr #1 substring$ chr.to.int$ 'byte :=
724     byte #128 <
725     { charptr #1 + 'charptr :=
726       byte #64 > byte #91 < and byte #96 > byte #123 < and or
727       { lang.en 'char.lang := }
728       { lang.other 'char.lang := }
729       if$
730     }
731     { tmp.str charptr #1 + #1 substring$ chr.to.int$ 'second.byte :=
732       byte #224 <
```

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

```
733       { charptr #2 + 'charptr :=
734         byte #207 > byte #212 < and
735         byte #212 = second.byte #176 < and or
736         { lang.ru 'char.lang := }
737         { lang.other 'char.lang := }
738       if$
739     }
740     { byte #240 <
```

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

```
741       { charptr #3 + 'charptr :=
742         byte #227 > byte #234 < and
743         { lang.zh 'char.lang := }
```

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

BF.

```
744       { byte #227 =
745         { second.byte #143 >
746         { lang.zh 'char.lang := }
```

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

```
747       { second.byte #128 > second.byte #132 < and
748         { lang.ja 'char.lang := }
749         { lang.other 'char.lang := }
750       if$
751     }
```

```

752             if$
753         }

```

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

```

754         { byte #239 =
755             second.byte #163 > second.byte #172 < and and
756             { lang.zh 'char.lang := }
757             { lang.other 'char.lang := }
758         if$
759     }
760     if$
761 }
762 if$
763 }

```

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.

```

764         { charptr #4 + 'charptr :=
765         byte #240 = second.byte #159 > and
766         { lang.zh 'char.lang := }
767         { lang.other 'char.lang := }
768     if$
769 }
770 if$
771 }
772 if$
773 }
774 if$
775 char.lang tmp.lang >
776 { char.lang 'tmp.lang := }
777 'skip$
778 if$
779 }
780 while$
781 tmp.lang
782 }
783
784 FUNCTION {check.entry.lang}
785 { author field.or.null
786   title field.or.null *
787   get.str.lang
788 }
789
790 FUNCTION {set.entry.lang}
791 { language empty$
792   { check.entry.lang }
793   { language "english" = language "american" = or language "british" = or
794     { lang.en }
795     { language "chinese" =
796       { lang.zh }
797       { language "japanese" =
798         { lang.ja }
799         { language "russian" =

```

```

800             { lang.ru }
801             { check.entry.lang }
802         if$
803     }
804     if$
805 }
806     if$
807 }
808     if$
809 }
810 if$
811 'entry.lang :=
812 }
813
814 FUNCTION {set.entry.numbered}
815 { type$ "patent" =
816   type$ "standard" = or
817   type$ "techreport" = or
818   { #1 'entry.numbered := }
819   { #0 'entry.numbered := }
820 if$
821 }
822

```

B.4.2 Format names

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

```

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

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

```

```

        fi
    else nameresult := t
    fi
    nameptr := nameptr + 1
    namesleft := namesleft - 1
od
return nameresult
END

```

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

```

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

```

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

```

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

```

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

```

823 INTEGERS { nameptr namesleft numnames name.lang }
824
825 FUNCTION {format.names}
826 { 's :=
827   #1 'nameptr :=
828   s num.names$ 'numnames :=
829   numnames 'namesleft :=
830   { namesleft #0 > }
831   { s nameptr "{vv~}{ll}{, jj}{, ff}" format.name$ 't :=
832     nameptr max.num.authors >
833     { bbl.et.al
834       #1 'namesleft :=
835     }
836     { t "others" =
837       { bbl.et.al }
838       { t get.str.lang 'name.lang :=
839         name.lang lang.en =

```

```

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

```

```

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

```



```

950     { author format.full.names }
951   if$
952 }
953
954 FUNCTION {author.full}
955 { author empty$
956   { "" }
957   { author format.full.names }
958   if$
959 }
960
961 FUNCTION {editor.full}
962 { editor empty$
963   { "" }
964   { editor format.full.names }
965   if$
966 }
967
968 FUNCTION {make.full.names}
969 { type$ "book" =
970   type$ "inbook" =
971   or
972   'author.editor.full
973   { type$ "collection" =
974     type$ "proceedings" =
975     or
976     'editor.full
977     'author.full
978     if$
979   }
980   if$
981 }
982
983 FUNCTION {output.bibitem}
984 { newline$
985   "\bibitem[" write$
986   label "]" *
987   make.full.names duplicate$ short.list =
988   { pop$ }
989   { * }
990   if$
991   's :=
992   s text.length$ 'charptr :=
993   { charptr #0 > s charptr #1 substring$ "[" = not and }
994   { charptr #1 - 'charptr := }
995   while$
996   charptr #0 >
997   { "{" s * "}" * }
998   { s }
999   if$
1000  "]" * write$
1001  cite$ write$
1002  "]" write$
1003  newline$
1004  ""

```

```

1005 before.all 'output.state :=
1006 }
1007

```

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 capitilized; for some styles, however, we leave it as it is in the database.

```

1008 FUNCTION {change.sentence.case}
1009 { entry.lang lang.en =
1010   { "t" change.case$ }
1011   'skip$
1012   if$
1013 }
1014
1015 FUNCTION {add.link}
1016 { url empty$ not
1017   { "\href{" url * "}" * swap$ * "}" * }
1018   { doi empty$ not
1019     { "\href{http://dx.doi.org/" doi * "}" * swap$ * "}" * }
1020     'skip$
1021     if$
1022   }
1023   if$
1024 }
1025
1026 FUNCTION {format.title}
1027 { title empty$
1028   { "" }
1029   { title
1030     sentence.case.title
1031     'change.sentence.case
1032     'skip$
1033     if$
1034     entry.numbered number empty$ not and
1035     { bbl.colon * number * }
1036     'skip$
1037     if$
1038     link.title
1039     'add.link
1040     'skip$
1041     if$
1042   }
1043   if$
1044 }
1045

```

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
```

```
1046 FUNCTION {tie.or.space.connect}
1047 { duplicate$ text.length$ #3 <
1048   { "~" }
1049   { " " }
1050   if$
1051   swap$ * *
1052 }
1053
```

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
```

```
1054 FUNCTION {either.or.check}
1055 { empty$
1056   'pop$
1057   { "can't use both " swap$ * " fields in " * cite$ * warning$ }
1058   if$
1059 }
1060
```

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.

```

1061 FUNCTION {is.digit}
1062 { duplicate$ empty$
1063   { pop$ #0 }
1064   { chr.to.int$
1065     duplicate$ "0" chr.to.int$ <
1066     { pop$ #0 }
1067     { "9" chr.to.int$ >
1068       { #0 }
1069       { #1 }
1070     if$
1071   }
1072   if$
1073 }
1074 if$
1075 }
1076
1077 FUNCTION {is.number}
1078 { 's :=
1079   s empty$
1080   { #0 }
1081   { s text.length$ 'charptr :=
1082     { charptr #0 >
1083       s charptr #1 substring$ is.digit
1084       and
1085     }
1086     { charptr #1 - 'charptr := }
1087     while$
1088     charptr not
1089   }
1090   if$
1091 }
1092
1093 FUNCTION {format.volume}
1094 { volume empty$ not
1095   { volume is.number
1096     { entry.lang lang.zh =
1097       { " 第 " volume * " 卷" * }
1098       { "volume" volume tie.or.space.connect }
1099     if$
1100   }
1101   { volume }
1102   if$
1103 }
1104 { "" }
1105 if$
1106 }
1107
1108 FUNCTION {format.number}
1109 { number empty$ not
1110   { number is.number
1111     { entry.lang lang.zh =
1112       { " 第 " number * " 册" * }

```

```

1113         { "number" number tie.or.space.connect }
1114     if$
1115 }
1116 { number }
1117 if$
1118 }
1119 { "" }
1120 if$
1121 }
1122
1123 FUNCTION {format.volume.number}
1124 { volume empty$ not
1125   { format.volume }
1126   { format.number }
1127   if$
1128 }
1129
1130 FUNCTION {format.title.vol.num}
1131 { title
1132   sentence.case.title
1133   'change.sentence.case
1134   'skip$
1135   if$
1136   entry.numbered
1137   { number empty$ not
1138     { bbl.colon * number * }
1139     'skip$
1140     if$
1141   }
1142   { format.volume.number 's :=
1143     s empty$ not
1144     { bbl.colon * s * }
1145     'skip$
1146     if$
1147   }
1148   if$
1149 }
1150
1151 FUNCTION {format.series.vol.num.title}
1152 { format.volume.number 's :=
1153   series empty$ not
1154   { series
1155     sentence.case.title
1156     'change.sentence.case
1157     'skip$
1158     if$
1159     entry.numbered
1160     { bbl.wide.space * }
1161     { bbl.colon *
1162       s empty$ not
1163       { s * bbl.wide.space * }
1164       'skip$
1165       if$
1166     }
1167     if$

```

```

1168     title *
1169     sentence.case.title
1170     'change.sentence.case
1171     'skip$
1172     if$
1173     entry.numbered number empty$ not and
1174     { bbl.colon * number * }
1175     'skip$
1176     if$
1177     }
1178     { format.title.vol.num }
1179     if$
1180     link.title
1181     'add.link
1182     'skip$
1183     if$
1184 }
1185
1186 FUNCTION {format.booktitle.vol.num}
1187 { booktitle
1188   entry.numbered
1189   'skip$
1190   { format.volume.number 's :=
1191     s empty$ not
1192     { bbl.colon * s * }
1193     'skip$
1194     if$
1195   }
1196   if$
1197 }
1198
1199 FUNCTION {format.series.vol.num.booktitle}
1200 { format.volume.number 's :=
1201   series empty$ not
1202   { series bbl.colon *
1203     entry.numbered not s empty$ not and
1204     { s * bbl.wide.space * }
1205     'skip$
1206     if$
1207     booktitle *
1208   }
1209   { format.booktitle.vol.num }
1210   if$
1211   in.booktitle
1212   { duplicate$ empty$ not entry.lang lang.en = and
1213     { "In: " swap$ * }
1214     'skip$
1215     if$
1216   }
1217   'skip$
1218   if$
1219 }
1220
1221 FUNCTION {remove.period}
1222 { 't :=

```

```

1223 "" 's :=
1224 { t empty$ not }
1225 { t #1 #1 substring$ 'tmp.str :=
1226   tmp.str "." = not
1227   { s tmp.str * 's := }
1228   'skip$
1229   if$
1230   t #2 global.max$ substring$ 't :=
1231 }
1232 while$
1233 s
1234 }
1235
1236 FUNCTION {abbreviate}
1237 { remove.period
1238   't :=
1239   t "l" change.case$ 's :=
1240   ""
1241   s "physical review letters" =
1242   { "Phys Rev Lett" }
1243   'skip$
1244   if$
1245   (*npr)
1246   s "china physics c" =
1247   { "Chin Phys C" }
1248   'skip$
1249   if$
1250   s "chinese physics letters" =
1251   { "Chin Phys Lett" }
1252   'skip$
1253   if$
1254   s "nuclear instruments and methods in physics research section a" =
1255   { "Nucl Instr and Meth A" }
1256   'skip$
1257   if$
1258   s "nuclear instruments and methods in physics research section a: accelerators, spectrometers,
1259   { "Nucl Instr and Meth A" }
1260   'skip$
1261   if$
1262   s "nuclear instruments and methods in physics research section b" =
1263   { "Nucl Instr and Meth B" }
1264   'skip$
1265   if$
1266   s "nuclear instruments and methods in physics research section b: beam interactions with materi
1267   { "Nucl Instr and Meth B" }
1268   'skip$
1269   if$
1270   s "physical review c" =
1271   { "Phys Rev C" }
1272   'skip$
1273   if$
1274   s "physical review d" =
1275   { "Phys Rev D" }
1276   'skip$
1277   if$

```

```

1278 s "physical review e" =
1279 { "Phys Rev E" }
1280 'skip$
1281 if$
1282 s "physics letters b" =
1283 { "Phys Lett B" }
1284 'skip$
1285 if$
1286 </npr>
1287 's :=
1288 s empty$
1289 { t }
1290 { pop$ s }
1291 if$
1292 }
1293
1294 FUNCTION {format.journal}
1295 { journal empty$ not
1296   { journal
1297     abbreviate.journal
1298     'abbreviate
1299     'skip$
1300     if$
1301     italic.journal entry.lang lang.en = and
1302     'italicize
1303     'skip$
1304     if$
1305   }
1306   { "" }
1307   if$
1308 }
1309

```

B.4.4 Format entry type mark

```

1310 FUNCTION {set.entry.mark}
1311 { entry.mark empty$ not
1312   'pop$
1313   { mark empty$ not
1314     { pop$ mark 'entry.mark := }
1315     { 'entry.mark := }
1316     if$
1317   }
1318   if$
1319 }
1320
1321 FUNCTION {format.mark}
1322 { show.mark
1323 < *thu>
1324 type$ "phdthesis" = type$ "mastersthesis" = or type$ "patent" = or
1325 medium empty$ not or entry.is.electronic or
1326 and
1327 </thu>
1328 { entry.mark
1329   show.medium.type

```



```

1330         { medium empty$ not
1331           { "/" * medium * }
1332           { entry.is.electronic
1333             { "/OL" * }
1334             'skip$
1335             if$
1336           }
1337         if$
1338       }
1339       'skip$
1340       if$
1341       'entry.mark :=
1342 < *!thu>
1343       "\allowbreak[" entry.mark * "]" *
1344 < /!thu>
1345 < *thu>
1346       " [" entry.mark * "]" *
1347 < /thu>
1348     }
1349     { "" }
1350   if$
1351 }
1352

```

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.

```

1353 FUNCTION {num.to.ordinal}
1354 { duplicate$ text.length$ 'charptr :=
1355   duplicate$ charptr #1 substring$ 's :=
1356   s "1" =
1357     { "st" * }
1358   { s "2" =
1359     { "nd" * }
1360     { s "3" =
1361       { "rd" * }
1362       { "th" * }
1363     if$
1364   }
1365   if$
1366 }
1367 if$
1368 }
1369
1370 FUNCTION {format.edition}
1371 { edition empty$
1372   { "" }
1373   { edition is.number
1374     { entry.lang lang.zh =
1375       { edition " 版" * }
1376       { edition num.to.ordinal " ed." * }
1377     if$

```

```

1378     }
1379     { entry.lang lang.en =
1380         { edition change.sentence.case 's :=
1381             s "Revised" = s "Revised edition" = or
1382             { "Rev. ed." }
1383             { s " ed." *}
1384             if$
1385         }
1386         { edition }
1387     if$
1388     }
1389 if$
1390 }
1391 if$
1392 }
1393

```

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。

```

1394 FUNCTION {format.publisher}
1395 { publisher empty$ not
1396   { publisher }
1397   { school empty$ not
1398     { school }
1399     { organization empty$ not
1400       { organization }
1401       { institution empty$ not
1402         { institution }
1403         { "" }
1404       if$
1405     }
1406   if$
1407   }
1408 if$
1409 }
1410 if$
1411 }
1412
1413 FUNCTION {format.address.publisher}
1414 { address empty$ not
1415   { address
1416     format.publisher empty$ not
1417     { bbl.colon * format.publisher * }
1418     { entry.is.electronic not show.missing.address.publisher and
1419       { bbl.colon * bbl.sine.nomine * }
1420       'skip$
1421     if$
1422   }
1423   if$
1424 }
1425 { entry.is.electronic not show.missing.address.publisher and
1426   { format.publisher empty$ not
1427     { bbl.sine.loco bbl.colon * format.publisher * }

```

```

1428         { bbl.sine.loco.sine.nomine }
1429     if$
1430 }
1431 { format.publisher empty$ not
1432     { format.publisher }
1433     { "" }
1434 if$
1435 }
1436 if$
1437 }
1438 if$
1439 }
1440

```

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 `paptent` 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`。

```

1441 FUNCTION {extract.before.dash}
1442 { duplicate$ empty$
1443   { pop$ "" }
1444   { 's :=
1445     #1 'charptr :=
1446     s text.length$ #1 + 'len :=
1447     { charptr len <
1448       s charptr #1 substring$ "-" = not
1449       and
1450     }
1451     { charptr #1 + 'charptr := }
1452     while$
1453     s #1 charptr #1 - substring$
1454   }
1455 if$
1456 }
1457
1458 FUNCTION {extract.after.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     { charptr len <
1471       s charptr #1 substring$ "-" =

```

```

1472         and
1473     }
1474     { charptr #1 + 'charptr := }
1475     while$
1476     s charptr global.max$ substring$
1477 }
1478 if$
1479 }
1480
1481 FUNCTION {contains.dash}
1482 { duplicate$ empty$
1483   { pop$ #0 }
1484   { 's :=
1485     { s empty$ not
1486       s #1 #1 substring$ "-" = not
1487       and
1488     }
1489     { s #2 global.max$ substring$ 's := }
1490     while$
1491     s empty$ not
1492   }
1493   if$
1494 }
1495

```

著者-出版年制必须提取出年份

```

1496 FUNCTION {format.year}
1497 { year empty$ not
1498   { year extract.before.dash }
1499   { date empty$ not
1500     { date extract.before.dash }
1501     { "empty year in " cite$ * warning$
1502       urldate empty$ not
1503       { "[" urldate extract.before.dash * "]" * }
1504       { "" }
1505     }
1506     if$
1507   }
1508   if$
1509   extra.label *
1510 }
1511 }
1512

```

专利和报纸都是使用日期而不是年

```

1513 FUNCTION {format.date}
1514 { type$ "patent" = type$ "newspaper" = or
1515   date empty$ not and
1516   { date }
1517   { year }
1518   if$
1519 }
1520

```

更新、修改日期只用于电子资源 elctronic

```

1521 FUNCTION {format.editdate}
1522 { date empty$ not
1523   { "\allowbreak(" date * ")" * }
1524   { "" }
1525   if$
1526 }
1527

```

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。

```

1528 FUNCTION {format.urldate}
1529 { urldate empty$ not entry.is.electronic and
1530   { "\allowbreak[" urldate * "]" * }
1531   { "" }
1532   if$
1533 }
1534

```

B.4.8 Format pages

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

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

```

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

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

```

```

        t := t with the first character removed
    fi
od
return pageresult
END

```

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

```

1535 FUNCTION {hyphenate}
1536 { 't :=
1537   ""
1538   { t empty$ not }
1539   { t #1 #1 substring$ "-" =
1540     { "-" *
1541       { t #1 #1 substring$ "-" = }
1542       { t #2 global.max$ substring$ 't := }
1543       while$
1544     }
1545     { t #1 #1 substring$ *
1546       t #2 global.max$ substring$ 't :=
1547     }
1548     if$
1549   }
1550   while$
1551 }
1552

```

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

```

1553 FUNCTION {format.pages}
1554 { pages empty$
1555   { "" }
1556   { pages
1557     only.start.page
1558     'extract.before.dash
1559     'hyphenate
1560     if$
1561   }
1562   if$
1563 }
1564

```

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。

```

1565 FUNCTION {format.journal.volume}
1566 { volume empty$ not
1567   { bold.journal.volume
1568     { "\textbf{" volume * "}" * }
1569     { volume }
1570     if$
1571   }

```

```

1572     { "" }
1573   if$
1574 }
1575
1576 FUNCTION {format.journal.number}
1577 { number empty$ not
1578   { "\penalty0 (" number * ")" * }
1579   { "" }
1580   if$
1581 }
1582
1583 FUNCTION {format.journal.pages}
1584 { pages empty$
1585   { "" }
1586   { space.before.pages
1587     { ": " }
1588     { ":\penalty0 " }
1589     if$
1590     format.pages *
1591   }
1592   if$
1593 }
1594

```

连续出版物的年卷期有起止范围，需要特殊处理

```

1595 FUNCTION {format.periodical.year.volume.number}
1596 { year empty$ not
1597   { year extract.before.dash }
1598   { "empty year in periodical " cite$ * warning$ }
1599   if$
1600   volume empty$ not
1601   { ", " * volume extract.before.dash * }
1602   'skip$
1603   if$
1604   number empty$ not
1605   { "\penalty0 (" * number extract.before.dash * ")" * }
1606   'skip$
1607   if$
1608   year contains.dash
1609   { "--" *
1610     year extract.after.dash empty$
1611     volume extract.after.dash empty$ and
1612     number extract.after.dash empty$ and not
1613     { year extract.after.dash empty$ not
1614       { year extract.after.dash * }
1615       { year extract.before.dash * }
1616       if$
1617       volume empty$ not
1618       { ", " * volume extract.after.dash * }
1619       'skip$
1620       if$
1621       number empty$ not
1622       { "\penalty0 (" * number extract.after.dash * ")" * }
1623       'skip$
1624       if$

```

```

1625     }
1626     'skip$
1627     if$
1628   }
1629   'skip$
1630   if$
1631 }
1632

```

B.4.9 Format url and doi

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

```

1633 FUNCTION {check.url}
1634 { url empty$ not
1635   { "\url{" url * "}" * 'entry.url :=
1636     #1 'entry.is.electronic :=
1637   }
1638   { howpublished empty$ not
1639     { howpublished #1 #5 substring$ "\url{" =
1640       { howpublished 'entry.url :=
1641         #1 'entry.is.electronic :=
1642       }
1643       'skip$
1644     }
1645     if$
1646   }
1647   { note empty$ not
1648     { note #1 #5 substring$ "\url{" =
1649       { note 'entry.url :=
1650         #1 'entry.is.electronic :=
1651       }
1652       'skip$
1653     }
1654     if$
1655   }
1656   }
1657   if$
1658 }
1659 if$
1660 }
1661
1662 FUNCTION {format.url}
1663 { entry.url
1664 }
1665
1666 FUNCTION {output.url}
1667 { entry.url empty$ not
1668   { new.block
1669     entry.url output
1670   }
1671   'skip$
1672   if$
1673 }
1674

```


需要检测 DOI 是否已经包含在 URL 中。

```
1675 FUNCTION {check.doi}
1676 { doi empty$ not
1677   { #1 'entry.is.electronic := }
1678   'skip$
1679   if$
1680 }
1681
1682 FUNCTION {is.in.url}
1683 { 's :=
1684   s empty$
1685   { #1 }
1686   { entry.url empty$
1687     { #0 }
1688     { s text.length$ 'len :=
1689       entry.url text.length$ 'charptr :=
1690       { entry.url charptr len substring$ s = not
1691         charptr #0 >
1692         and
1693       }
1694       { charptr #1 - 'charptr := }
1695       while$
1696       charptr
1697     }
1698     if$
1699   }
1700   if$
1701 }
1702
1703 FUNCTION {format.doi}
1704 { ""
1705   doi empty$ not
1706   { "" 's :=
1707     doi 't :=
1708     #0 'numnames :=
1709     { t empty$ not}
1710     { t #1 #1 substring$ 'tmp.str :=
1711       tmp.str "," = tmp.str " " = or t #2 #1 substring$ empty$ or
1712       { t #2 #1 substring$ empty$
1713         { s tmp.str * 's := }
1714         'skip$
1715         if$
1716         s empty$ s is.in.url or
1717         'skip$
1718         { numnames #1 + 'numnames :=
1719           numnames #1 >
1720           { ", " * }
1721           { "DOI: " * }
1722           if$
1723           "\doi{" s * "}" * *
1724         }
1725         if$
1726         "" 's :=
1727       }
1728       { s tmp.str * 's := }
```

```

1729         if$
1730         t #2 global.max$ substring$ 't :=
1731     }
1732     while$
1733 }
1734 'skip$
1735 if$
1736 }
1737
1738 FUNCTION {output.doi}
1739 { doi empty$ not show.doi and
1740   show.english.translation entry.lang lang.zh = and not and
1741   { new.block
1742     format.doi output
1743   }
1744   'skip$
1745   if$
1746 }
1747
1748 FUNCTION {check.electronic}
1749 { "" 'entry.url :=
1750   #0 'entry.is.electronic :=
1751   'check.doi
1752   'skip$
1753   if$
1754   'check.url
1755   'skip$
1756   if$
1757   medium empty$ not
1758   { medium "MT" = medium "DK" = or medium "CD" = or medium "OL" = or
1759     { #1 'entry.is.electronic := }
1760     'skip$
1761     if$
1762   }
1763   'skip$
1764   if$
1765 }
1766
1767 FUNCTION {format.note}
1768 { note empty$ not show.note and
1769   { note }
1770   { "" }
1771   if$
1772 }
1773
1774 FUNCTION {output.translation}
1775 { show.english.translation entry.lang lang.zh = and
1776   { translation empty$ not
1777     { translation }
1778     { "[English translation missing!]" }
1779     if$
1780     " (in Chinese)" * output
1781     write$
1782     format.doi duplicate$ empty$ not
1783     { newlines$

```

```

1784         write$
1785     }
1786     'pop$
1787     if$
1788     " \\" write$
1789     newline$
1790     "(" write$
1791     ""
1792     before.all 'output.state :=
1793 }
1794 'skip$
1795 if$
1796 }
1797

```

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

```

1798 FUNCTION {empty.misc.check}
1799 { author empty$ title empty$
1800   year empty$
1801   and and
1802   key empty$ not and
1803   { "all relevant fields are empty in " cite$ * warning$ }
1804   'skip$
1805   if$
1806 }
1807

```

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

```

1808 FUNCTION {monograph}
1809 { output.bibitem
1810   output.translation
1811   author empty$ not
1812     { format.authors }
1813     { editor empty$ not
1814       { format.editors }
1815       { "empty author and editor in " cite$ * warning$
1816 (*authoryear)
1817   bbl.anonymous
1818 (/authoryear)
1819 (*numerical)

```

```

1820      ""
1821 </numerical>
1822   }
1823   if$
1824   }
1825   if$
1826   output
1827 <(*authoryear)
1828   period.between.author.year
1829   'new.sentence
1830   'skip$
1831   if$
1832   format.year "year" output.check
1833 </authoryear>
1834   new.block
1835   format.series.vol.num.title "title" output.check
1836   "M" set.entry.mark
1837   format.mark "" output.after
1838   new.block
1839   format.translators output
1840   new.sentence
1841   format.edition output
1842   new.block
1843   format.address.publisher output
1844 <(*numerical)
1845   format.year "year" output.check
1846 </numerical>
1847   format.pages bbl.colon output.after
1848   format.urldate "" output.after
1849   output.url
1850   output.doi
1851   new.block
1852   format.note output
1853   fin.entry
1854 }
1855

```

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

```

1856 FUNCTION {incollection}
1857 { output.bibitem
1858   output.translation
1859   format.authors output
1860   author format.key output
1861 <(*authoryear)
1862   period.between.author.year

```

```

1863     'new.sentence
1864     'skip$
1865     if$
1866     format.year "year" output.check
1867 </authoryear>
1868     new.block
1869     format.title "title" output.check
1870     "M" set.entry.mark
1871     format.mark "" output.after
1872     new.block
1873     format.translators output
1874     new.slash
1875     format.editors output
1876     new.block
1877     format.series.vol.num.booktitle "booktitle" output.check
1878     new.block
1879     format.edition output
1880     new.block
1881     format.address.publisher output
1882 <(*numerical>
1883     format.year "year" output.check
1884 </numerical>
1885     format.pages bbl.colon output.after
1886     format.urldate "" output.after
1887     output.url
1888     output.doi
1889     new.block
1890     format.note output
1891     fin.entry
1892 }
1893

```

B.5.3 连续出版物

```

1894 FUNCTION {periodical}
1895 { output.bibitem
1896   output.translation
1897   format.authors output
1898   author format.key output
1899 <(*authoryear>
1900   period.between.author.year
1901     'new.sentence
1902     'skip$
1903     if$
1904     format.year "year" output.check
1905 </authoryear>
1906     new.block
1907     format.title "title" output.check
1908     "J" set.entry.mark
1909     format.mark "" output.after
1910     new.block
1911     format.periodical.year.volume.number output
1912     new.block
1913     format.address.publisher output
1914 <(*numerical>

```

```

1915 format.date "year" output.check
1916 </numerical>
1917 format.urldate "" output.after
1918 output.url
1919 output.doi
1920 new.block
1921 format.note output
1922 fin.entry
1923 }
1924

```

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.

```

1925 FUNCTION {article}
1926 { output.bibitem
1927 output.translation
1928 format.authors output
1929 author format.key output
1930 <*authoryear>
1931 period.between.author.year
1932 'new.sentence
1933 'skip$
1934 if$
1935 format.year "year" output.check
1936 </authoryear>
1937 new.block
1938 title.in.journal
1939 { format.title "title" output.check
1940 "J" set.entry.mark
1941 format.mark "" output.after
1942 new.block
1943 }
1944 'skip$
1945 if$
1946 format.journal "journal" output.check
1947 <*numerical>
1948 format.date "year" output.check
1949 </numerical>
1950 format.journal.volume output
1951 format.journal.number "" output.after
1952 format.journal.pages "" output.after
1953 format.urldate "" output.after
1954 output.url
1955 output.doi
1956 new.block
1957 format.note output

```

```

1958   fin.entry
1959 }
1960

```

B.5.5 专利文献

number 域也可以用来表示专利号。

```

1961 FUNCTION {patent}
1962 { output.bibitem
1963   output.translation
1964   format.authors output
1965   author format.key output
1966   (*authoryear)
1967   period.between.author.year
1968     'new.sentence
1969     'skip$
1970   if$
1971   format.year "year" output.check
1972 } /authoryear)
1973 new.block
1974 format.title "title" output.check
1975 "P" set.entry.mark
1976 format.mark "" output.after
1977 new.block
1978 format.date "year" output.check
1979 format.urldate "" output.after
1980 output.url
1981 output.doi
1982 new.block
1983 format.note output
1984 fin.entry
1985 }
1986

```

B.5.6 电子资源

```

1987 FUNCTION {electronic}
1988 { #1 #1 check.electronic
1989   #1 'entry.is.electronic :=
1990   output.bibitem
1991   output.translation
1992   format.authors output
1993   author format.key output
1994   (*authoryear)
1995   period.between.author.year
1996     'new.sentence
1997     'skip$
1998   if$
1999   format.year "year" output.check
2000 } /authoryear)
2001 new.block
2002 format.series.vol.num.title "title" output.check
2003 "EB" set.entry.mark
2004 format.mark "" output.after

```

```

2005 new.block
2006 format.address.publisher output
2007 <{*numerical>
2008   date empty$
2009     { format.date output }
2010     'skip$
2011   if$
2012 </numerical>
2013 format.pages bbl.colon output.after
2014 format.editeddate "" output.after
2015 format.urldate "" output.after
2016 output.url
2017 output.doi
2018 new.block
2019 format.note output
2020 fin.entry
2021 }
2022

```

B.5.7 其他文献类型

A misc is something that doesn't fit elsewhere.

Required: at least one of the 'optional' fields

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

Misc 用来自动判断类型。

```

2023 FUNCTION {misc}
2024 { journal empty$ not
2025   'article
2026   { booktitle empty$ not
2027     'incollection
2028     { publisher empty$ not
2029       'monograph
2030       { entry.is.electronic
2031         'electronic
2032         { "Z" set.entry.mark
2033           monograph
2034         }
2035       }
2036     }
2037   }
2038   if$
2039 }
2040 if$
2041 }
2042 empty.misc.check
2043 }
2044
2045 FUNCTION {archive}
2046 { "A" set.entry.mark
2047   misc
2048 }
2049

```

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

2050 FUNCTION {book} { monograph }

2051

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

Required: title

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

2052 FUNCTION {booklet} { book }

2053

2054 FUNCTION {collection}

2055 { "G" set.entry.mark

2056 monograph

2057 }

2058

2059 FUNCTION {database}

2060 { "DB" set.entry.mark

2061 electronic

2062 }

2063

2064 FUNCTION {dataset}

2065 { "DS" set.entry.mark

2066 electronic

2067 }

2068

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。

2069 FUNCTION {inbook} { book }

2070

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

Required: author, title, booktitle, year

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

2071 FUNCTION {inproceedings}

2072 { "C" set.entry.mark

2073 incollection

2074 }

2075

The conference function is included for Scribe compatibility.

```
2076 FUNCTION {conference} { inproceedings }
2077
2078 FUNCTION {map}
2079 { "CM" set.entry.mark
2080   misc
2081 }
2082
```

A manual is technical documentation.

Required: title

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

```
2083 FUNCTION {manual} { monograph }
2084
```

A mastersthesis is a Master's thesis.

Required: author, title, school, year

Optional: type, address, month, note

```
2085 FUNCTION {mastersthesis}
2086 <!*thu>
2087 { "D" set.entry.mark
2088 </!thu>
2089 <!*thu>
2090 { lang.zh entry.lang =
2091   { " 硕士学位论文" }
2092   { "D" }
2093   if$
2094   set.entry.mark
2095 </thu>
2096   monograph
2097 }
2098
2099 FUNCTION {newspaper}
2100 { "N" set.entry.mark
2101   article
2102 }
2103
2104 FUNCTION {online}
2105 { "EB" set.entry.mark
2106   electronic
2107 }
2108
```

A phdthesis is like a mastersthesis.

Required: author, title, school, year

Optional: type, address, month, note

```
2109 <!*thu>
2110 FUNCTION {phdthesis} { mastersthesis }
2111 </!thu>
2112 <!*thu>
2113 FUNCTION {phdthesis}
```

```

2114 { lang.zh entry.lang =
2115     { " 博士学位论文" }
2116     { "D" }
2117   if$
2118   set.entry.mark
2119   monograph
2120 }
2121 </thu>
2122

```

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

```

2123 FUNCTION {proceedings}
2124 { "C" set.entry.mark
2125   monograph
2126 }
2127
2128 FUNCTION {software}
2129 { "CP" set.entry.mark
2130   electronic
2131 }
2132
2133 FUNCTION {standard}
2134 { "S" set.entry.mark
2135   misc
2136 }
2137

```

A techreport is a technical report.

Required: author, title, institution, year

Optional: type, number, address, month, note

```

2138 FUNCTION {techreport}
2139 { "R" set.entry.mark
2140   misc
2141 }
2142

```

An unpublished is something that hasn't been published.

Required: author, title, note

Optional: month, year

```

2143 FUNCTION {unpublished}
2144 { "Z" set.entry.mark
2145   misc
2146 }
2147

```

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

```
2148 FUNCTION {default.type} { misc }
2149
```

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

```
2150 MACRO {jan} {"January"}
2151
2152 MACRO {feb} {"February"}
2153
2154 MACRO {mar} {"March"}
2155
2156 MACRO {apr} {"April"}
2157
2158 MACRO {may} {"May"}
2159
2160 MACRO {jun} {"June"}
2161
2162 MACRO {jul} {"July"}
2163
2164 MACRO {aug} {"August"}
2165
2166 MACRO {sep} {"September"}
2167
2168 MACRO {oct} {"October"}
2169
2170 MACRO {nov} {"November"}
2171
2172 MACRO {dec} {"December"}
2173
```

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

```
2174 MACRO {acmcs} {"ACM Computing Surveys"}
2175
2176 MACRO {acta} {"Acta Informatica"}
2177
2178 MACRO {cacm} {"Communications of the ACM"}
2179
2180 MACRO {ibmjrd} {"IBM Journal of Research and Development"}
2181
2182 MACRO {ibmsj} {"IBM Systems Journal"}
2183
2184 MACRO {ieeese} {"IEEE Transactions on Software Engineering"}
```

```

2185
2186 MACRO {ieetc} {"IEEE Transactions on Computers"}
2187
2188 MACRO {ieetcad}
2189 {"IEEE Transactions on Computer-Aided Design of Integrated Circuits"}
2190
2191 MACRO {ipl} {"Information Processing Letters"}
2192
2193 MACRO {jacm} {"Journal of the ACM"}
2194
2195 MACRO {jcss} {"Journal of Computer and System Sciences"}
2196
2197 MACRO {scp} {"Science of Computer Programming"}
2198
2199 MACRO {sicomp} {"SIAM Journal on Computing"}
2200
2201 MACRO {tocs} {"ACM Transactions on Computer Systems"}
2202
2203 MACRO {tods} {"ACM Transactions on Database Systems"}
2204
2205 MACRO {tog} {"ACM Transactions on Graphics"}
2206
2207 MACRO {toms} {"ACM Transactions on Mathematical Software"}
2208
2209 MACRO {toois} {"ACM Transactions on Office Information Systems"}
2210
2211 MACRO {toplas} {"ACM Transactions on Programming Languages and Systems"}
2212
2213 MACRO {tcs} {"Theoretical Computer Science"}
2214

```

B.7 Format labels

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

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

```

2215 FUNCTION {sortify}
2216 {  purify$
2217   "l" change.case$
2218 }
2219

```

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

```

2220 FUNCTION {chop.word}
2221 { 's :=
2222   'len :=
2223   s #1 len substring$ =
2224   { s len #1 + global.max$ substring$ }
2225   's
2226   if$

```

2227 }
2228

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
```

```
2229 FUNCTION {format.lab.names}
2230 { 's :=
2231 s #1 "{vv~}{ll}{, jj}{, ff}" format.name$ 't :=
2232 t get.str.lang 'name.lang :=
2233 name.lang lang.en =
2234 { t #1 "{vv~}{ll}" format.name$}
2235 { t #1 "{ll}{ff}" format.name$}
2236 if$
2237 s num.names$ #1 >
2238 { bbl.space * citation.et.al * }
```

```

2239     'skip$
2240   if$
2241 }
2242
2243 FUNCTION {author.key.label}
2244 { author empty$
2245   { key empty$
2246     { cite$ #1 #3 substring$ }
2247     'key
2248     if$
2249   }
2250   { author format.lab.names }
2251   if$
2252 }
2253
2254 FUNCTION {author.editor.key.label}
2255 { author empty$
2256   { editor empty$
2257     { key empty$
2258       { cite$ #1 #3 substring$ }
2259       'key
2260       if$
2261     }
2262     { editor format.lab.names }
2263     if$
2264   }
2265   { author format.lab.names }
2266   if$
2267 }
2268
2269 FUNCTION {author.key.organization.label}
2270 { author empty$
2271   { key empty$
2272     { organization empty$
2273       { cite$ #1 #3 substring$ }
2274       { "The " #4 organization chop.word #3 text.prefix$ }
2275       if$
2276     }
2277     'key
2278     if$
2279   }
2280   { author format.lab.names }
2281   if$
2282 }
2283
2284 FUNCTION {editor.key.organization.label}
2285 { editor empty$
2286   { key empty$
2287     { organization empty$
2288       { cite$ #1 #3 substring$ }
2289       { "The " #4 organization chop.word #3 text.prefix$ }
2290       if$
2291     }
2292     'key
2293     if$

```



```

2294     }
2295     { editor format.lab.names }
2296     if$
2297 }
2298
2299 FUNCTION {calc.short.authors}
2300 { type$ "book" =
2301   type$ "inbook" =
2302   or
2303     'author.editor.key.label
2304     { type$ "collection" =
2305       type$ "proceedings" =
2306       or
2307         { editor empty$ not
2308           'editor.key.organization.label
2309           'author.key.organization.label
2310           if$
2311         }
2312         'author.key.label
2313       if$
2314     }
2315   if$
2316   'short.list :=
2317 }
2318
2319 FUNCTION {calc.label}
2320 { calc.short.authors
2321   short.list
2322   "("
2323   *
2324   format.year duplicates$ empty$
2325   short.list key field.or.null = or
2326     { pop$ "" }
2327     'skip$
2328   if$
2329   *
2330   'label :=
2331 }
2332

```

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`.

```

2333 (*authoryear)
2334 FUNCTION {sort.language.label}
2335 { entry.lang lang.zh =
2336   { lang.zh.order }
2337   { entry.lang lang.ja =
2338     { lang.ja.order }
2339     { entry.lang lang.en =
2340       { lang.en.order }
2341       { entry.lang lang.ru =
2342         { lang.ru.order }
2343         { lang.other.order }
2344         if$
2345       }
2346     }
2347   }
2348   if$
2349 }
2350 if$
2351 int.to.chr$
2352 }
2353
2354 FUNCTION {sort.format.names}
2355 { 's :=
2356   #1 'nameptr :=
2357   ""
2358   s num.names$ 'numnames :=
2359   numnames 'namesleft :=
2360   { namesleft #0 > }
2361   {
2362     s nameptr "{vv{ } }{ll{ }}{ ff{ }}{ jj{ }}" format.name$ 't :=
2363     nameptr #1 >
2364     {
2365       " " *
2366       namesleft #1 = t "others" = and
2367       { "zzzzz" * }
2368       { numnames #2 > nameptr #2 = and
2369         { "zz" * year field.or.null * " " * }
2370         'skip$
2371       }
2372       if$
2373       t sortify *
2374     }
2375     if$
2376   }

```

```

2376         { t sortify * }
2377     if$
2378         nameptr #1 + 'nameptr :=
2379         namesleft #1 - 'namesleft :=
2380     }
2381 while$
2382 }
2383

```

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

```

2384 FUNCTION {sort.format.title}
2385 { 't :=
2386   "A " #2
2387   "An " #3
2388   "The " #4 t chop.word
2389   chop.word
2390   chop.word
2391   sortify
2392   #1 global.max$ substring$
2393 }
2394

```

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.

```

2395 FUNCTION {anonymous.sort}
2396 { entry.lang lang.zh =
2397   { "yi4 ming2" }
2398   { "anon" }
2399   if$
2400 }
2401
2402 FUNCTION {warn.empty.key}
2403 { entry.lang lang.zh =
2404   { "empty key in " cite$ * warning$ }
2405   'skip$
2406   if$
2407 }
2408
2409 FUNCTION {author.sort}
2410 { key empty$
2411   { warn.empty.key
2412     author empty$
2413     { anonymous.sort }
2414     { author sort.format.names }
2415     if$
2416   }
2417   { key sortify }
2418   if$

```

```

2419 }
2420
2421 FUNCTION {author.editor.sort}
2422 { key empty$
2423   { warn.empty.key
2424     author empty$
2425       { editor empty$
2426         { anonymous.sort }
2427         { editor sort.format.names }
2428       if$
2429     }
2430     { author sort.format.names }
2431   if$
2432 }
2433 { key sortify }
2434 if$
2435 }
2436
2437 FUNCTION {author.organization.sort}
2438 { key empty$
2439   { warn.empty.key
2440     author empty$
2441       { organization empty$
2442         { anonymous.sort }
2443         { "The " #4 organization chop.word sortify }
2444       if$
2445     }
2446     { author sort.format.names }
2447   if$
2448 }
2449 { key sortify }
2450 if$
2451 }
2452
2453 FUNCTION {editor.organization.sort}
2454 { key empty$
2455   { warn.empty.key
2456     editor empty$
2457       { organization empty$
2458         { anonymous.sort }
2459         { "The " #4 organization chop.word sortify }
2460       if$
2461     }
2462     { editor sort.format.names }
2463   if$
2464 }
2465 { key sortify }
2466 if$
2467 }
2468
2469 </authoryear>

```

顺序编码制的排序要简单得多

```

2470 (*numerical)
2471 INTEGERS { seq.num }

```

```

2472
2473 FUNCTION {init.seq}
2474 { #0 'seq.num :=}
2475
2476 FUNCTION {int.to.fix}
2477 { "00000000" swap$ int.to.str$ *
2478   #-1 #10 substring$
2479 }
2480
2481 </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!

```

2482 FUNCTION {presort}
2483 { set.entry.lang
2484   set.entry.numbered
2485   show.url show.doi check.electronic
2486   calc.label
2487   label sortify
2488   " "
2489   *
2490   <*authoryear>
2491   sort.language.label
2492   type$ "book" =
2493   type$ "inbook" =
2494   or
2495     'author.editor.sort
2496     { type$ "collection" =
2497       type$ "proceedings" =
2498       or
2499         'editor.organization.sort
2500         'author.sort
2501       if$
2502     }
2503   if$
2504   *
2505   " "
2506   *
2507   year field.or.null sortify
2508   *
2509   " "
2510   *
2511   cite$
2512   *
2513   #1 entry.max$ substring$
2514   </authoryear>
2515   <*numerical>
2516   seq.num #1 + 'seq.num :=
2517   seq.num int.to.fix
2518   </numerical>
2519   'sort.label :=
2520   sort.label *
2521   #1 entry.max$ substring$

```

```

2522 'sort.key$ :=
2523 }
2524

```

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

```

```

2525 STRINGS { longest.label last.label next.extra }
2526
2527 INTEGERS { longest.label.width last.extra.num number.label }
2528
2529 FUNCTION {initialize.longest.label}
2530 { "" 'longest.label :=
2531   #0 int.to.chr$ 'last.label :=

```

```

2532 "" 'next.extra :=
2533 #0 'longest.label.width :=
2534 #0 'last.extra.num :=
2535 #0 'number.label :=
2536 }
2537
2538 FUNCTION {forward.pass}
2539 { last.label label =
2540   { last.extra.num #1 + 'last.extra.num :=
2541     last.extra.num int.to.chr$ 'extra.label :=
2542   }
2543   { "a" chr.to.int$ 'last.extra.num :=
2544     "" 'extra.label :=
2545     label 'last.label :=
2546   }
2547   if$
2548   number.label #1 + 'number.label :=
2549 }
2550
2551 FUNCTION {reverse.pass}
2552 { next.extra "b" =
2553   { "a" 'extra.label := }
2554   'skip$
2555   if$
2556   extra.label 'next.extra :=
2557   extra.label
2558   duplicate$ empty$
2559   'skip$
2560   { "{\natexlab{" swap$ * "}}" * }
2561   if$
2562   'extra.label :=
2563   label extra.label * 'label :=
2564 }
2565
2566 FUNCTION {bib.sort.order}
2567 { sort.label 'sort.key$ :=
2568 }
2569

```

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.

```

2570 FUNCTION {begin.bib}
2571 { preamble$ empty$
2572   'skip$
2573   { preamble$ write$ newline$ }
2574   if$
2575   "\begin{thebibliography}{ number.label int.to.str$ * "}" *

```

```

2576 write$ newline$
2577 "\providecommand{\natexlab}[1]{#1}"
2578 write$ newline$
2579 "\providecommand{\url}[1]{#1}"
2580 write$ newline$
2581 "\expandafter\ifx\csname urlstyle\endcsname\relax\else"
2582 write$ newline$
2583 " \urlstyle{same}\fi"
2584 write$ newline$
2585 show.doi
2586 { "\expandafter\ifx\csname href\endcsname\relax"
2587   write$ newline$
2588   " \DeclareUrlCommand\doi{\urlstyle{rm}}\else"
2589   write$ newline$
2590   " \providecommand\doi[1]{\href{https://doi.org/#1}{\nolinkurl{#1}}}\fi"
2591   write$ newline$
2592 }
2593 'skip$
2594 if$
2595 }
2596

```

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

```

2597 FUNCTION {end.bib}
2598 { newline$
2599   "\end{thebibliography}" write$ newline$
2600 }
2601

```

B.10 Main execution

Now we read in the .BIB entries.

```

2602 READ
2603
2604 EXECUTE {init.state.consts}
2605
2606 EXECUTE {load.config}
2607
2608 (*numerical)
2609 EXECUTE {init.seq}
2610
2611 (/numerical)
2612 ITERATE {presort}
2613

```

And now we can sort

```

2614 SORT
2615
2616 EXECUTE {initialize.longest.label}
2617
2618 ITERATE {forward.pass}
2619
2620 REVERSE {reverse.pass}
2621

```



```
2622 ITERATE {bib.sort.order}  
2623  
2624 SORT  
2625  
2626 EXECUTE {begin.bib}  
2627
```

Now we produce the output for all the entries

```
2628 ITERATE {call.type$}  
2629  
2630 EXECUTE {end.bib}  
2631 </authoryear | numerical>
```