

GB/T 7714—2015 BibT_EX style

Zeping Lee*

2020/12/17 v2.1

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 authoryear styles as well as a L^AT_EX package which provides the citation style defined in the standard. It is compatible with natbib and supports language detection (Chinese and English) for each bibliography entry.

1 简介

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

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

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

2 版本 v2.0 的重要修改

从 v2.0 版本开始(2020-03-04), 用户必须在文档中使用 `\bibliographystyle` 命令选择参考文献样式, 如 `gbt7714-numerical` 或 `gbt7714-author-year`。在早期的版本中, 选择文献样式的方法是将 `numbers` 或 `super` 等参数传递给 `gbt7714`, 而不能使用 `\bibliographystyle`。这跟标准的 L^AT_EX 接口不一致, 所以将被弃用。

* zepinglee AT gmail.com

3 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
```

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 6 节。

4 文献类型

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

表 1: 全部文献类型

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

5 著录项目

由于国标中规定的著录项目多于 Bib_TE_X 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 BibLa_TE_X，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

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
langid* 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,  
  langid = {japanese},  
  mark   = {Z},  
  medium = {DK},  
  ...  
}
```

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

6 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},  
  ...  
}
```

7 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
<code>uppercase.name</code>	<code>#1</code>	将著者姓名转为大写
<code>max.num.authors</code>	<code>#3</code>	输出著者的最多数目
<code>year.after.author</code>	<code>#0</code>	年份置于著者之后
<code>period.after.author</code>	<code>#0</code>	著者和年份之间使用句点连接
<code>sentence.case.title</code>	<code>#1</code>	将西文的题名转为 <code>sentence case</code>
<code>link.title</code>	<code>#0</code>	在题名上添加 <code>url</code> 的超链接
<code>title.in.journal</code>	<code>#1</code>	期刊是否显示标题
<code>show.mark</code>	<code>#1</code>	显示文献类型标识
<code>show.medium.type</code>	<code>#1</code>	显示载体类型标识
<code>italic.journal</code>	<code>#0</code>	西文期刊名使用斜体
<code>show.missing.address.publisher</code>	<code>#1</code>	出版项缺失时显示“出版者不详”
<code>space.before.pages</code>	<code>#1</code>	页码与前面的冒号之间有空格
<code>only.start.page</code>	<code>#0</code>	只显示起始页码
<code>show.urldate</code>	<code>#1</code>	显示引用日期 <code>urldate</code>
<code>show.url</code>	<code>#1</code>	显示 <code>url</code>
<code>show.doi</code>	<code>#1</code>	显示 DOI
<code>show.preprint</code>	<code>#0</code>	显示预印本信息
<code>show.note</code>	<code>#0</code>	显示 <code>note</code> 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。

8 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 Bib_T_EX 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 Bib_T_EX 实现，还给每行 `bst` 代码写了 `java` 语言注释。沈周^[10]基于 `biblatex-casperson`^[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. CTeX 论坛 (已关闭) .
- [8] 李志奇. 基于 biblatex 的符合 GBT7714—2005 的中文文献生成工具 [EB/OL]. 2013. CTeX 论坛 (已关闭) .
- [9] 胡海星. A GB/T 7714—2005 national standard compliant BibTeX style[EB/OL]. 2013. <https://github.com/Haixing-Hu/GBT7714-2005-BibTeX-Style>.
- [10] 沈周. 基于 caspervector 改写的符合 GB/T 7714—2005 标准的参考文献格式 [EB/OL]. 2016. <https://github.com/szsdk/biblatex-gbt77142005>.
- [11] VECTOR C T. biblatex 参考文献和引用样式: caspervector[M/OL]. 2012. <http://mirrors.ctan.org/macros/latex/contrib/biblatex-contrib/biblatex-caspervector/doc/caspervector.pdf>.
- [12] 胡振震. 符合 GB/T 7714—2015 标准的 biblatex 参考文献样式 [M/OL]. 2016. <http://mirrors.ctan.org/macros/latex/contrib/biblatex-contrib/biblatex-gb7714-2015/biblatex-gb7714-2015.pdf>.

A 宏包的代码实现

兼容过时的接口

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

将选项传递给 natbib

```
36 \PassOptionsToPackage{compress}{natbib}
37 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{natbib}}
38 \ProcessOptions\relax
```

调用宏包，注意只需要 `compress` 不需要 `sort`。

```
39 \RequirePackage{natbib}
```

```
40 \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`。

```
41 \renewcommand\newblock{\space}
```

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

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

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

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

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

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

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

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

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

`\cite` 下面修改 `natbib` 的引用格式。为了减少依赖的宏包，这里直接重定义命令不使用 `\patchcmd`。

Super 样式的 `\citep` 的页码也为上标。

```
50 \renewcommand\NAT@citesuper[3]{\ifNAT@swa
```

```
51 \if*#2*\else#2\NAT@spacechar\fi
```

```
52 % \unskip\kern\p@ \textsuperscript{\NAT@@open#1\NAT@@close}%
```

```
53 % \if*#3*\else\NAT@spacechar#3\fi\else #1\fi\endgroup}
```

```
54 \unskip\kern\p@ \textsuperscript{\NAT@@open#1\NAT@@close\if*#3*\else#3\fi}%
```

```
55 \else #1\fi\endgroup}
```

将 `numbers` 样式的 `\citep` 的页码置于括号外。

```
56 \renewcommand\NAT@citenum%
```

```
57 [3]{\ifNAT@swa\NAT@@open\if*#2*\else#2\NAT@spacechar\fi
```

```
58 % #1\if*#3*\else\NAT@cmt#3\fi\NAT@@close\else#1\fi\endgroup}
```

```
59 #1\NAT@@close\textsuperscript{\if*#3*\else#3\fi}\else#1\fi\endgroup}
```

Numerical 模式的 `\citet` 的页码:

```
60 \def\NAT@citexnum[#1][#2]#3{%
```

```
61 \NAT@reset@parser
```

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

```
63 \NAT@reset@citea
```

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

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

```
66 {\@safe@activestrue
```

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



```

68 \safe@activesfalse
69 \ifundefined{b@\@citeb\@extra@b@citeb}{%
70   {\reset@font\bfseries?}
71   \NAT@citeundefined\PackageWarning{natbib}%
72   {Citation '\@citeb' on page \thepage \space undefined}}%
73 {\let\NAT@last@num\NAT@num\let\NAT@last@nm\NAT@nm
74   \NAT@parse{\@citeb}%
75   \ifNAT@longnames\@ifundefined{bv@\@citeb\@extra@b@citeb}{%
76     \let\NAT@name=\NAT@all@names
77     \global\@namedef{bv@\@citeb\@extra@b@citeb}{}}{%
78   \fi
79   \ifNAT@full\let\NAT@nm\NAT@all@names\else
80     \let\NAT@nm\NAT@name\fi
81   \ifNAT@swa
82     \@ifnum{\NAT@ctype>\@ne}{%
83       \citea
84       \NAT@hyper@{\@ifnum{\NAT@ctype=\tw@}{\NAT@test{\NAT@ctype}}{\NAT@alias}}%
85     }{%
86       \@ifnum{\NAT@cmprs>\z@}{%
87         \NAT@ifcat@num\NAT@num
88         {\let\NAT@nm=\NAT@num}%
89         {\def\NAT@nm{-2}}%
90         \NAT@ifcat@num\NAT@last@num
91         {\@tempcnta=\NAT@last@num\relax}%
92         {\@tempcnta\m@ne}%
93         \@ifnum{\NAT@nm=\@tempcnta}{%
94           \@ifnum{\NAT@merge>\@ne}{\NAT@last@yr@mbx}%
95         }{%
96           \advance\@tempcnta by\@ne
97           \@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。

```

98 % \ifx\NAT@last@yr\relax
99 %   \def\NAT@last@yr{\@citea}%
100 % \else
101 %   \def\NAT@last@yr{--\NAT@penalty}%
102 % \fi
103 \def\NAT@last@yr{-\NAT@penalty}%
104 }{%
105   \NAT@last@yr@mbx
106 }%
107 }%

```

```

108     }{%
109     \@tempwattrue
110     \@ifnum{\NAT@merge>\@ne}{\@ifnum{\NAT@last@num=\NAT@num\relax}{\@tempwafalse}{}}{%
111     \if@tempwa\NAT@citea@mbox\fi
112     }%
113     }%
114     \NAT@def@citea
115     \else
116     \ifcase\NAT@ctype
117     \ifx\NAT@last@nm\NAT@nm \NAT@yrsep\NAT@penalty\NAT@space\else
118     \@citea \NAT@test{\@ne}\NAT@spacechar\NAT@mbox{\NAT@super@kern\NAT@@open}%
119     \fi
120     \if*#1*\else#1\NAT@spacechar\fi
121     \NAT@mbox{\NAT@hyper@{\@citenumfont{\NAT@num}}}%
122     \NAT@def@citea@box
123     \or
124     \NAT@hyper@citea@space{\NAT@test{\NAT@ctype}}%
125     \or
126     \NAT@hyper@citea@space{\NAT@test{\NAT@ctype}}%
127     \or
128     \NAT@hyper@citea@space\NAT@alias
129     \fi
130     \fi
131     }%
132     }%
133     \@ifnum{\NAT@cmprs>\z@}{\NAT@last@yr}{}%
134     \ifNAT@swa\else

```

将页码放在括号外边，并且置于上标。

```

135     % \@ifnum{\NAT@ctype=\z@}{%
136     %   \if*#2*\else\NAT@cmt#2\fi
137     % }{%
138     \NAT@mbox{\NAT@@close}%
139     \@ifnum{\NAT@ctype=\z@}{%
140     \if*#2*\else\textsuperscript{#2}\fi
141     }{%
142     \fi
143     }{#1}{#2}%
144 }%

```

Author-year 模式的 \citep 的页码：

```

145 \renewcommand\NAT@cite%
146 [3]{\ifNAT@swa\NAT@@open\if*#2*\else#2\NAT@spacechar\fi

```

147 #1\NAT@@close\if*#3*\else#3\fi\else#1\fi\endgroup}

Author-year 模式的 \citete 的页码:

```

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

```

```

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

```



```

265         \immediate\write\@auxout{\string\bibstyle{#1}}%
266     \fi}%
267 }%
268 }
269 \ifgbt@legacy@interface
270 \ifgbt@numerical
271     \ifgbt@super\else
272         \citestyle{numbers}
273     \fi
274     \bibliographystyle{gbt7714-numerical}
275 \else
276     \bibliographystyle{gbt7714-author-year}
277 \fi
278 \fi
279 </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。

```

280 <(*authoryear | numerical)>
281 INTEGERS {
282     citation.et.al.min
283     citation.et.al.use.first
284     bibliography.et.al.min
285     bibliography.et.al.use.first
286     uppercase.name
287     terms.in.macro
288     year.after.author
289     period.after.author
290     sentence.case.title
291     link.title
292     title.in.journal
293     show.mark
294     show.medium.type
295     slash.for.extraction
296     in.booktitle
297     short.journal
298     italic.journal
299     bold.journal.volume
300     show.missing.address.publisher
301     space.before.pages
302     only.start.page
303     show.urldate
304     show.url
305     show.doi

```

```

306 show.preprint
307 show.note
308 show.english.translation
309 (*authoryear)
310 lang.zh.order
311 lang.ja.order
312 lang.en.order
313 lang.ru.order
314 lang.other.order
315 </authoryear>
316 }
317

```

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。

```

318 FUNCTION {load.config}
319 {

```

如果姓名的数量大于等于 **et.al.min**，只著录前 **et.al.use.first** 个，其后加“et al.”或“等”。

```

320 (*!ucas)
321 #2 'citation.et.al.min :=
322 #1 'citation.et.al.use.first :=
323 </!ucas)
324 (*ucas)
325 #3 'citation.et.al.min :=
326 #1 'citation.et.al.use.first :=
327 </ucas)
328 #4 'bibliography.et.al.min :=
329 #3 'bibliography.et.al.use.first :=

```

英文姓名转为全大写：

```

330 (*!(nouppercase | thu))
331 #1 'uppercase.name :=
332 </!(nouppercase | thu))
333 (*nouppercase | thu)
334 #0 'uppercase.name :=
335 </nouppercase | thu)

```

使用 TeX 宏输出“和”、“等”

```

336 (*!(macro | ucas))
337 #0 'terms.in.macro :=
338 </!(macro | ucas))
339 (*macro | ucas)
340 #1 'terms.in.macro :=
341 </macro | ucas)

```

将年份置于著者后面（著者-出版年制默认）

```

342 (*numerical | ucas)
343 #0 'year.after.author :=
344 </numerical | ucas)
345 (*authoryear&!ucas)
346 #1 'year.after.author :=
347 </authoryear&!ucas)

```

采用著者-出版年制时，作者姓名与年份之间使用句点连接：

```
348 <!*numerical>
349   #1 'period.after.author :=
350 </numerical>
351 <!*authoryear>
352 <!*2015&!(period | thu | ustc)>
353   #0 'period.after.author :=
354 </2015&!(period | thu | ustc)>
355 <!*period | 2005 | thu | ustc>
356   #1 'period.after.author :=
357 </period | 2005 | thu | ustc>
358 </authoryear>
```

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

```
359 </!nosentencecase>
360   #1 'sentence.case.title :=
361 <!*nosentencecase>
362   #0 'sentence.case.title :=
363 </nosentencecase>
```

在标题添加超链接：

```
364 <!*linktitle>
365   #0 'link.title :=
366 </!linktitle>
367 <!*linktitle>
368   #1 'link.title :=
369 </linktitle>
```

期刊是否含标题：

```
370 <!*!(title-in-journal | npr)>
371   #1 'title.in.journal :=
372 </!(title-in-journal | npr)>
373 <!*title-in-journal | npr>
374   #0 'title.in.journal :=
375 </title-in-journal | npr>
```

著录文献类型标识（比如“[M/OL]”）：

```
376 <!*nomark>
377   #1 'show.mark :=
378 </!nomark>
379 <!*nomark>
380   #0 'show.mark :=
381 </nomark>
```

是否显示载体类型标识（比如“/OL”）：

```
382 <!*no.medium.type>
383   #1 'show.medium.type :=
384 </!no.medium.type>
385 <!*no.medium.type>
386   #0 'show.medium.type :=
387 </no.medium.type>
```

使用“//”表示析出文献

```
388 <!*noslash>
```



```

389 #1 'slash.for.extraction :=
390 </!noslash>
391 < *noslash>
392 #0 'slash.for.extraction :=
393 </noslash>

```

使用“In:”表示析出文献

```

394 #0 'in.booktitle :=

```

期刊名使用缩写:

```

395 < *!(short-journal | npr)>
396 #0 'short.journal :=
397 </!(short-journal | npr)>
398 < *short-journal | npr>
399 #1 'short.journal :=
400 </short-journal | npr>

```

期刊名使用斜体:

```

401 < *!italicjournal>
402 #0 'italic.journal :=
403 </!italicjournal>
404 < *italicjournal>
405 #1 'italic.journal :=
406 </italicjournal>

```

期刊的卷使用粗体:

```

407 #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.”:

```

408 < *!(noslsn | thu | ustc |ucas | npr)>
409 #1 'show.missing.address.publisher :=
410 </!(noslsn | thu | ustc |ucas | npr)>
411 < *noslsn | thu | ustc |ucas | npr>
412 #0 'show.missing.address.publisher :=
413 </noslsn | thu | ustc |ucas | npr>

```

页码是否只含起始页:

```

414 < *!(no-space-before-pages | thu)>
415 #1 'space.before.pages :=
416 </!(no-space-before-pages | thu)>
417 < *no-space-before-pages | thu>
418 #0 'space.before.pages :=
419 </no-space-before-pages | thu>

```

页码是否只含起始页:

```

420 < *!(only-start-page | npr)>
421 #0 'only.start.page :=
422 </!(only-start-page | npr)>
423 < *only-start-page | npr>
424 #1 'only.start.page :=
425 </only-start-page | npr>

```

是否著录非电子文献的引用日期:

```

426 < *!no-urldate>
427 #1 'show.urldate :=

```

```

428 </!no-urldate>
429 <*no-urldate>
430 #0 'show.urldate :=
431 </no-urldate>

```

是否著录 URL:

```

432 <*!nourl>
433 #1 'show.url :=
434 </!nourl>
435 <*nourl>
436 #0 'show.url :=
437 </nourl>

```

是否著录 DOI:

```

438 <*!nodoi & 2015>
439 #1 'show.doi :=
440 </!nodoi & 2015>
441 <*nodoi | 2005>
442 #0 'show.doi :=
443 </nodoi | 2005>

```

是否著录 e-print:

```

444 <*!(preprint | npr)>
445 #0 'show.preprint :=
446 </!(preprint | npr)>
447 <*preprint | npr>
448 #1 'show.preprint :=
449 </preprint | npr>

```

在每一条文献最后输出注释 (note) 的内容:

```

450 #0 'show.note :=

```

中文文献是否显示英文翻译

```

451 <*!(show-english-translation | npr)>
452 #0 'show.english.translation :=
453 </!(show-english-translation | npr)>
454 <*show-english-translation | npr>
455 #1 'show.english.translation :=
456 </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:

```

457 <*authoryear>
458 #1 'lang.zh.order :=
459 #2 'lang.ja.order :=
460 #3 'lang.en.order :=
461 #4 'lang.ru.order :=
462 #5 'lang.other.order :=
463 </authoryear>
464 }
465

```

B.2 The ENTRY declaration

Like Scribe's (according to pages 231-2 of the April '84 edition), but no fullauthor or editors fields because BibTeX does name handling. The 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.

```
466 ENTRY
467 { address
468   archivePrefix
469   author
470   booktitle
471   date
472   doi
473   edition
474   editor
475   eprint
476   eprinttype
477   howpublished
478   institution
479   journal
480   journaltitle
481   key
482   langid
483   language
484   location
485   mark
486   medium
487   note
488   number
489   organization
490   pages
491   publisher
492   school
493   series
494   shortjournal
495   title
496   translation
497   translator
498   url
499   urldate
500   volume
501   year
502 }
503 { entry.lang entry.is.electronic is.pure.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.

```
504 { label extra.label sort.label short.label short.list entry.mark entry.url }
505
```

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
```

```
506 INTEGERS { output.state before.all mid.sentence after.sentence after.block after.slash }
507
508 INTEGERS { lang.zh lang.ja lang.en lang.ru lang.other }
509
510 INTEGERS { charptr len }
511
512 FUNCTION {init.state.consts}
513 { #0 'before.all :=
514   #1 'mid.sentence :=
515   #2 'after.sentence :=
516   #3 'after.block :=
517   #4 'after.slash :=
518   #3 'lang.zh :=
519   #4 'lang.ja :=
520   #1 'lang.en :=
521   #2 'lang.ru :=
522   #0 'lang.other :=
523 }
524
```

下面是一些常量的定义

```
525 FUNCTION {bbl.anonymous}
526 { entry.lang lang.zh =
```

```

527 { " 佚名" }
528 { "Anon" }
529 if$
530 }
531
532 FUNCTION {bbl.space}
533 { entry.lang lang.zh =
534   { "\ " }
535   { " " }
536   if$
537 }
538
539 FUNCTION {bbl.and}
540 { "" }
541
542 FUNCTION {bbl.et.al}
543 { entry.lang lang.zh =
544   { " 等" }
545   { entry.lang lang.ja =
546     { " 他" }
547     { entry.lang lang.ru =
548       { "идр" }
549       { "et~al." }
550       if$
551     }
552     if$
553   }
554   if$
555 }
556
557 FUNCTION {citation.and}
558 { terms.in.macro
559   { "{\biband}" }
560   'bbl.and
561   if$
562 }
563
564 FUNCTION {citation.et.al}
565 { terms.in.macro
566   { "{\bibetal}" }
567   'bbl.et.al
568   if$
569 }
570
571 FUNCTION {bbl.colon} { ": " }
572
573 FUNCTION {bbl.pages.colon}
574 { space.before.pages
575   { ": " }
576   { ":\allowbreak " }
577   if$
578 }
579
580 (*2015)
581 FUNCTION {bbl.wide.space} { "\quad " }

```

```

582 </2015>
583 <*2005>
584 FUNCTION {bbl.wide.space} { "\ " }
585 </2005>
586
587 FUNCTION {bbl.slash} { "//\allowbreak " }
588
589 FUNCTION {bbl.sine.loco}
590 { entry.lang lang.zh =
591   { "[出版地不详]" }
592   { "[S.l.]" }
593   if$
594 }
595
596 FUNCTION {bbl.sine.nomine}
597 { entry.lang lang.zh =
598   { "[出版者不详]" }
599   { "[s.n.]" }
600   if$
601 }
602
603 FUNCTION {bbl.sine.loco.sine.nomine}
604 { entry.lang lang.zh =
605   { "[出版地不详: 出版者不详]" }
606   { "[S.l.: s.n.]" }
607   if$
608 }
609

```

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

```

610 FUNCTION {not}
611 { { #0 }
612   { #1 }
613   if$
614 }
615
616 FUNCTION {and}
617 { 'skip$
618   { pop$ #0 }
619   if$
620 }
621
622 FUNCTION {or}
623 { { pop$ #1 }
624   'skip$
625   if$
626 }
627

```

the variables s and t are temporary string holders

```

628 STRINGS { s t }
629

```



```

630 FUNCTION {output.nonnull}
631 { 's :=
632   output.state mid.sentence =
633   { ", " * write$ }
634   { output.state after.block =
635     { add.period$ write$
636       newline$
637       "\newblock " write$
638     }
639     { output.state before.all =
640       'write$
641       { output.state after.slash =
642         { bbl.slash * write$
643           newline$
644         }
645         { add.period$ " " * write$ }
646       }
647     }
648     if$
649   }
650   if$
651   mid.sentence 'output.state :=
652 }
653 if$
654 s
655 }
656
657 FUNCTION {output}
658 { duplicate$ empty$
659   'pop$
660   'output.nonnull
661   if$
662 }
663
664 FUNCTION {output.after}
665 { 't :=
666   duplicate$ empty$
667   'pop$
668   { 's :=
669     output.state mid.sentence =
670     { t * write$ }
671     { output.state after.block =
672       { add.period$ write$
673         newline$
674         "\newblock " write$
675       }
676       { output.state before.all =
677         'write$
678         { output.state after.slash =
679           { bbl.slash * write$ }
680           { add.period$ " " * write$ }
681         }
682       }
683       if$
684     }

```

```

685         if$
686         mid.sentence 'output.state :=
687     }
688     if$
689     s
690 }
691 if$
692 }
693
694 FUNCTION {output.check}
695 { 't :=
696   duplicate$ empty$
697   { pop$ "empty " t * " in " * cite$ * warning$ }
698   'output.nonnull
699   if$
700 }
701

```

This function finishes all entries.

```

702 FUNCTION {fin.entry}
703 { add.period$
704   write$
705   show.english.translation entry.lang lang.zh = and
706   { ")"
707     write$
708   }
709   'skip$
710   if$
711   newline$
712 }
713
714 FUNCTION {new.block}
715 { output.state before.all =
716   'skip$
717   { output.state after.slash =
718     'skip$
719     { after.block 'output.state := }
720     if$
721   }
722   if$
723 }
724
725 FUNCTION {new.sentence}
726 { output.state after.block =
727   'skip$
728   { output.state before.all =
729     'skip$
730     { output.state after.slash =
731       'skip$
732       { after.sentence 'output.state := }
733       if$
734     }
735     if$
736   }
737   if$

```

```

738 }
739
740 FUNCTION {new.slash}
741 { output.state before.all =
742   'skip$
743   { slash.for.extraction
744     { after.slash 'output.state := }
745     { after.block 'output.state := }
746     if$
747   }
748   if$
749 }
750

```

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.

```

751 FUNCTION {new.block.checka}
752 { empty$
753   'skip$
754   'new.block
755   if$
756 }
757
758 FUNCTION {new.block.checkb}
759 { empty$
760   swap$ empty$
761   and
762   'skip$
763   'new.block
764   if$
765 }
766

```

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

```

767 FUNCTION {new.sentence.checka}
768 { empty$
769   'skip$
770   'new.sentence
771   if$
772 }
773
774 FUNCTION {new.sentence.checkb}
775 { empty$
776   swap$ empty$
777   and
778   'skip$
779   'new.sentence
780   if$
781 }
782

```

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

```
783 FUNCTION {field.or.null}  
784 { duplicate$ empty$  
785   { pop$ "" }  
786   'skip$  
787   if$  
788 }  
789  
790 FUNCTION {italicize}  
791 { duplicate$ empty$  
792   { pop$ "" }  
793   { "\textit{" swap$ * "}" * }  
794   if$  
795 }  
796
```

B.4.1 Detect Language

```
797 INTEGERS { byte second.byte }  
798  
799 INTEGERS { char.lang tmp.lang }
```

```

800
801 STRINGS { tmp.str }
802
803 FUNCTION {get.str.lang}
804 { 'tmp.str :=
805   lang.other 'tmp.lang :=
806   #1 'charptr :=
807   tmp.str text.length$ #1 + 'len :=
808   { charptr len < }
809   { tmp.str charptr #1 substring$ chr.to.int$ 'byte :=
810     byte #128 <
811     { charptr #1 + 'charptr :=
812       byte #64 > byte #91 < and byte #96 > byte #123 < and or
813       { lang.en 'char.lang := }
814       { lang.other 'char.lang := }
815       if$
816     }
817     { tmp.str charptr #1 + #1 substring$ chr.to.int$ 'second.byte :=
818       byte #224 <

```

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

```

819     { charptr #2 + 'charptr :=
820       byte #207 > byte #212 < and
821       byte #212 = second.byte #176 < and or
822       { lang.ru 'char.lang := }
823       { lang.other 'char.lang := }
824       if$
825     }
826     { byte #240 <

```

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

```

827     { charptr #3 + 'charptr :=
828       byte #227 > byte #234 < and
829       { lang.zh 'char.lang := }

```

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

```

830     { byte #227 =
831       { second.byte #143 >
832       { lang.zh 'char.lang := }

```

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

```

833     { second.byte #128 > second.byte #132 < and
834     { lang.ja 'char.lang := }
835     { lang.other 'char.lang := }
836     if$
837   }
838   if$
839 }

```

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

```

840     { byte #239 =
841     second.byte #163 > second.byte #172 < and and
842     { lang.zh 'char.lang := }
843     { lang.other 'char.lang := }

```

```

844             if$
845         }
846     if$
847 }
848 if$
849 }

```

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.

```

850         { charptr #4 + 'charptr :=
851         byte #240 = second.byte #159 > and
852         { lang.zh 'char.lang := }
853         { lang.other 'char.lang := }
854     if$
855 }
856     if$
857 }
858     if$
859 }
860     if$
861     char.lang tmp.lang >
862     { char.lang 'tmp.lang := }
863     'skip$
864     if$
865 }
866 while$
867 tmp.lang
868 }
869
870 FUNCTION {check.entry.lang}
871 { author field.or.null
872   title field.or.null *
873   get.str.lang
874 }
875
876 STRINGS { entry.langid }
877
878 FUNCTION {set.entry.lang}
879 { "" 'entry.langid :=
880   language empty$ not
881   { language 'entry.langid := }
882   'skip$
883   if$
884   langid empty$ not
885   { langid 'entry.langid := }
886   'skip$
887   if$
888   entry.langid empty$
889   { check.entry.lang }
890   { entry.langid "english" = entry.langid "american" = or entry.langid "british" = or
891     { lang.en }
892     { entry.langid "chinese" =
893       { lang.zh }

```

```

894         { entry.langid "japanese" =
895           { lang.ja }
896           { entry.langid "russian" =
897             { lang.ru }
898             { check.entry.lang }
899             if$
900           }
901         if$
902       }
903     if$
904   }
905   if$
906 }
907 if$
908 'entry.lang :=
909 }
910
911 FUNCTION {set.entry.numbered}
912 { type$ "patent" =
913   type$ "standard" = or
914   type$ "techreport" = or
915   { #1 'entry.numbered := }
916   { #0 'entry.numbered := }
917   if$
918 }
919

```

B.4.2 Format names

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

```

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

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

```

```

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

```

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

```

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

```

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

```

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

```

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

```

920 INTEGERS { nameptr namesleft numnames name.lang }
921
922 FUNCTION {format.name}
923 { "{vv~}{ll}{, jj}{, ff}" format.name$ 't :=
924   t "others" =
925     { bbl.et.al }
926   { t get.str.lang 'name.lang :=
927     name.lang lang.en =
928       { t #1 "{vv~}{ll}{~f{~}}" format.name$
929         uppercase.name
930         { "u" change.case$ }
931         'skip$
932       if$
933       t #1 "{, jj}" format.name$ *

```



```

934     }
935     { t #1 "{ll}{ff}" format.name$ }
936     if$
937   }
938   if$
939 }
940
941 FUNCTION {format.names}
942 { 's :=
943   #1 'nameptr :=
944   s num.names$ 'numnames :=
945   ""
946   numnames 'namesleft :=
947   { namesleft #0 > }
948   { s nameptr format.name bbl.et.al =
949     numnames bibliography.et.al.min #1 - > nameptr bibliography.et.al.use.first > and or
950     { ", " *
951       bbl.et.al *
952       #1 'namesleft :=
953     }
954     { nameptr #1 >
955       { namesleft #1 = bbl.and "" = not and
956         { bbl.and * }
957         { ", " * }
958       }
959       if$
960       'skip$
961       if$
962       s nameptr format.name *
963     }
964     if$
965     nameptr #1 + 'nameptr :=
966     namesleft #1 - 'namesleft :=
967   }
968   while$
969 }
970
971 FUNCTION {format.key}
972 { empty$
973   { key field.or.null }
974   { "" }
975   if$
976 }
977
978 FUNCTION {format.authors}
979 { author empty$ not
980   { author format.names }
981   { "empty author in " cite$ * warning$
982     (*authoryear)
983     bbl.anonymous
984     (/authoryear)
985     (*numerical)
986     ""
987     (/numerical)
988   }

```

```

989  if$
990 }
991
992 FUNCTION {format.editors}
993 { editor empty$
994   { "" }
995   { editor format.names }
996   if$
997 }
998
999 FUNCTION {format.translators}
1000 { translator empty$
1001   { "" }
1002   { translator format.names
1003     entry.lang lang.zh =
1004     { translator num.names$ #3 >
1005       { " 译" * }
1006       { ", 译" * }
1007       if$
1008     }
1009     'skip$
1010     if$
1011   }
1012   if$
1013 }
1014
1015 FUNCTION {format.full.names}
1016 { 's :=
1017   #1 'nameptr :=
1018   s num.names$ 'numnames :=
1019   numnames 'namesleft :=
1020   { namesleft #0 > }
1021   { s nameptr "{vv~}{ll}{, jj}{, ff}" format.name$ 't :=
1022     t get.str.lang 'name.lang :=
1023     name.lang lang.en =
1024     { t #1 "{vv~}{ll}" format.name$ 't := }
1025     { t #1 "{ll}{ff}" format.name$ 't := }
1026     if$
1027     nameptr #1 >
1028     {
1029       namesleft #1 >
1030       { ", " * t * }
1031       {
1032         numnames #2 >
1033         { ", " * }
1034         'skip$
1035         if$
1036         t "others" =
1037         { " et~al." * }
1038         { " and " * t * }
1039         if$
1040       }
1041       if$
1042     }
1043     't

```

```

1044     if$
1045     nameptr #1 + 'nameptr :=
1046     namesleft #1 - 'namesleft :=
1047     }
1048 while$
1049 }
1050
1051 FUNCTION {author.editor.full}
1052 { author empty$
1053   { editor empty$
1054     { "" }
1055     { editor format.full.names }
1056     if$
1057   }
1058   { author format.full.names }
1059   if$
1060 }
1061
1062 FUNCTION {author.full}
1063 { author empty$
1064   { "" }
1065   { author format.full.names }
1066   if$
1067 }
1068
1069 FUNCTION {editor.full}
1070 { editor empty$
1071   { "" }
1072   { editor format.full.names }
1073   if$
1074 }
1075
1076 FUNCTION {make.full.names}
1077 { type$ "book" =
1078   type$ "inbook" =
1079   or
1080   'author.editor.full
1081   { type$ "collection" =
1082     type$ "proceedings" =
1083     or
1084     'editor.full
1085     'author.full
1086     if$
1087   }
1088   if$
1089 }
1090
1091 FUNCTION {output.bibitem}
1092 { newline$
1093   "\bibitem[" writes$
1094   label "]" *
1095   make.full.names duplicates$ short.list =
1096   { pop$ }
1097   { * }
1098   if$

```

```

1099 's :=
1100 s text.length$ 'charptr :=
1101 { charptr #0 > s charptr #1 substring$ "[" = not and }
1102 { charptr #1 - 'charptr := }
1103 while$
1104 charptr #0 >
1105   { "{" s * "}" * }
1106   { s }
1107 if$
1108 "]"{" * write$
1109 cite$ write$
1110 "}" write$
1111 newline$
1112 ""
1113 before.all 'output.state :=
1114 }
1115

```

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.

```

1116 FUNCTION {change.sentence.case}
1117 { entry.lang lang.en =
1118   { "t" change.case$ }
1119   'skip$
1120   if$
1121 }
1122
1123 FUNCTION {add.link}
1124 { url empty$ not
1125   { "\href{" url * "}" * swap$ * "}" * }
1126   { doi empty$ not
1127     { "\href{http://dx.doi.org/" doi * "}" * swap$ * "}" * }
1128     'skip$
1129     if$
1130   }
1131   if$
1132 }
1133
1134 FUNCTION {format.title}
1135 { title empty$
1136   { "" }
1137   { title
1138     sentence.case.title
1139     'change.sentence.case
1140     'skip$
1141     if$
1142     entry.numbered number empty$ not and
1143     { bbl.colon * number * }

```

```

1144         'skip$
1145     if$
1146         link.title
1147         'add.link
1148         'skip$
1149     if$
1150 }
1151 if$
1152 }
1153

```

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

```

```

1154 FUNCTION {tie.or.space.connect}
1155 { duplicate$ text.length$ #3 <
1156   { "~" }
1157   { " " }
1158   if$
1159   swap$ * *
1160 }
1161

```

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

```

```

1162 FUNCTION {either.or.check}
1163 { empty$
1164   'pop$
1165   { "can't use both " swap$ * " fields in " * cite$ * warning$ }
1166   if$
1167 }
1168

```

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.

```

1169 FUNCTION {is.digit}
1170 { duplicate$ empty$
1171   { pop$ #0 }
1172   { chr.to.int$
1173     duplicate$ "0" chr.to.int$ <
1174     { pop$ #0 }
1175     { "9" chr.to.int$ >
1176       { #0 }
1177       { #1 }
1178     if$
1179   }
1180   if$
1181 }
1182 if$
1183 }
1184
1185 FUNCTION {is.number}
1186 { 's :=
1187   s empty$
1188   { #0 }
1189   { s text.length$ 'charptr :=
1190     { charptr #0 >
1191       s charptr #1 substring$ is.digit
1192       and
1193     }
1194     { charptr #1 - 'charptr := }
1195     while$
1196     charptr not
1197   }
1198   if$
1199 }
1200
1201 FUNCTION {format.volume}
1202 { volume empty$ not
1203   { volume is.number
1204     { entry.lang lang.zh =
1205       { " 第 " volume * " 卷" * }
1206       { "volume" volume tie.or.space.connect }
1207     if$
1208   }

```

```

1209         { volume }
1210     if$
1211 }
1212 { "" }
1213 if$
1214 }
1215
1216 FUNCTION {format.number}
1217 { number empty$ not
1218   { number is.number
1219     { entry.lang lang.zh =
1220       { " 第 " number * " 册" * }
1221       { "number" number tie.or.space.connect }
1222     if$
1223   }
1224   { number }
1225   if$
1226 }
1227 { "" }
1228 if$
1229 }
1230
1231 FUNCTION {format.volume.number}
1232 { volume empty$ not
1233   { format.volume }
1234   { format.number }
1235   if$
1236 }
1237
1238 FUNCTION {format.title.vol.num}
1239 { title
1240   sentence.case.title
1241   'change.sentence.case
1242   'skip$
1243   if$
1244   entry.numbered
1245   { number empty$ not
1246     { bbl.colon * number * }
1247     'skip$
1248     if$
1249   }
1250   { format.volume.number 's :=
1251     s empty$ not
1252     { bbl.colon * s * }
1253     'skip$
1254     if$
1255   }
1256   if$
1257 }
1258
1259 FUNCTION {format.series.vol.num.title}
1260 { format.volume.number 's :=
1261   series empty$ not
1262   { series
1263     sentence.case.title

```

```

1264         'change.sentence.case
1265         'skip$
1266     if$
1267     entry.numbered
1268         { bbl.wide.space * }
1269         { bbl.colon *
1270         s empty$ not
1271         { s * bbl.wide.space * }
1272         'skip$
1273     if$
1274     }
1275     if$
1276     title *
1277     sentence.case.title
1278         'change.sentence.case
1279         'skip$
1280     if$
1281     entry.numbered number empty$ not and
1282         { bbl.colon * number * }
1283         'skip$
1284     if$
1285     }
1286     { format.title.vol.num }
1287 if$
1288 link.title
1289     'add.link
1290     'skip$
1291 if$
1292 }
1293
1294 FUNCTION {format.booktitle.vol.num}
1295 { booktitle
1296     entry.numbered
1297     'skip$
1298     { format.volume.number 's :=
1299     s empty$ not
1300     { bbl.colon * s * }
1301     'skip$
1302     if$
1303     }
1304     if$
1305 }
1306
1307 FUNCTION {format.series.vol.num.booktitle}
1308 { format.volume.number 's :=
1309     series empty$ not
1310     { series bbl.colon *
1311     entry.numbered not s empty$ not and
1312     { s * bbl.wide.space * }
1313     'skip$
1314     if$
1315     booktitle *
1316     }
1317     { format.booktitle.vol.num }
1318     if$

```



```

1319 in.booktitle
1320 { duplicate$ empty$ not entry.lang lang.en = and
1321   { "In: " swap$ * }
1322   'skip$
1323   if$
1324 }
1325 'skip$
1326 if$
1327 }
1328
1329 FUNCTION {remove.period}
1330 { 't :=
1331   "" 's :=
1332   { t empty$ not }
1333   { t #1 #1 substring$ 'tmp.str :=
1334     tmp.str "." = not
1335     { s tmp.str * 's := }
1336     'skip$
1337     if$
1338     t #2 global.max$ substring$ 't :=
1339   }
1340   while$
1341   s
1342 }
1343
1344 FUNCTION {abbreviate}
1345 { remove.period
1346   't :=
1347   t "l" change.case$ 's :=
1348   ""
1349   s "physical review letters" =
1350   { "Phys Rev Lett" }
1351   'skip$
1352   if$
1353   (*npr)
1354   s "china physics c" =
1355   { "Chin Phys C" }
1356   'skip$
1357   if$
1358   s "chinese physics letters" =
1359   { "Chin Phys Lett" }
1360   'skip$
1361   if$
1362   s "nuclear instruments and methods in physics research section a" =
1363   { "Nucl Instr and Meth A" }
1364   'skip$
1365   if$
1366   s "nuclear instruments and methods in physics research section a: accelerators, spectrometers,"
1367   { "Nucl Instr and Meth A" }
1368   'skip$
1369   if$
1370   s "nuclear instruments and methods in physics research section b" =
1371   { "Nucl Instr and Meth B" }
1372   'skip$
1373   if$

```

```

1374 s "nuclear instruments and methods in physics research section b: beam interactions with materi
1375 { "Nucl Instr and Meth B" }
1376 'skip$
1377 if$
1378 s "physical review c" =
1379 { "Phys Rev C" }
1380 'skip$
1381 if$
1382 s "physical review d" =
1383 { "Phys Rev D" }
1384 'skip$
1385 if$
1386 s "physical review e" =
1387 { "Phys Rev E" }
1388 'skip$
1389 if$
1390 s "physics letters b" =
1391 { "Phys Lett B" }
1392 'skip$
1393 if$
1394 </npr>
1395 's :=
1396 s empty$
1397 { t }
1398 { pop$ s }
1399 if$
1400 }
1401
1402 FUNCTION {format.journal}
1403 { ""
1404   short.journal
1405   { shortjournal empty$ not
1406     { shortjournal * }
1407     { journal empty$ not
1408       { journal * abbreviate }
1409       { journaltitle empty$ not
1410         { journaltitle * abbreviate }
1411         'skip$
1412         if$
1413       }
1414       if$
1415     }
1416     if$
1417   }
1418   { journal empty$ not
1419     { journal * }
1420     { journaltitle empty$ not
1421       { journaltitle * }
1422       'skip$
1423       if$
1424     }
1425     if$
1426   }
1427   if$
1428   duplicates$ empty$ not

```

```

1429 { italic.journal entry.lang lang.en = and
1430     'italicize
1431     'skip$
1432     if$
1433 }
1434 'skip$
1435 if$
1436 }
1437

```

B.4.4 Format entry type mark

```

1438 FUNCTION {set.entry.mark}
1439 { entry.mark empty$ not
1440     'pop$
1441     { mark empty$ not
1442         { pop$ mark 'entry.mark := }
1443         { 'entry.mark := }
1444         if$
1445     }
1446     if$
1447 }
1448
1449 FUNCTION {format.mark}
1450 { show.mark
1451     { entry.mark
1452         show.medium.type
1453         { medium empty$ not
1454             { "/" * medium * }
1455             { entry.is.electronic
1456                 { "/OL" * }
1457                 'skip$
1458                 if$
1459             }
1460             if$
1461         }
1462         'skip$
1463         if$
1464         'entry.mark :=
1465         "[" entry.mark * "]" *
1466     }
1467     { "" }
1468     if$
1469 }
1470

```

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.

```

1471 FUNCTION {num.to.ordinal}
1472 { duplicate$ text.length$ 'charptr :=
1473     duplicate$ charptr #1 substring$ 's :=

```

```

1474 s "1" =
1475 { "st" * }
1476 { s "2" =
1477   { "nd" * }
1478   { s "3" =
1479     { "rd" * }
1480     { "th" * }
1481   if$
1482   }
1483   if$
1484 }
1485 if$
1486 }
1487
1488 FUNCTION {format.edition}
1489 { edition empty$
1490   { "" }
1491   { edition is.number
1492     { entry.lang lang.zh =
1493       { edition " 版" * }
1494       { edition num.to.ordinal " ed." * }
1495     if$
1496   }
1497   { entry.lang lang.en =
1498     { edition change.sentence.case 's :=
1499       s "Revised" = s "Revised edition" = or
1500       { "Rev. ed." }
1501       { s " ed." *}
1502     if$
1503   }
1504   { edition }
1505   if$
1506 }
1507 if$
1508 }
1509 if$
1510 }
1511

```

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。

```

1512 FUNCTION {format.publisher}
1513 { publisher empty$ not
1514   { publisher }
1515   { school empty$ not
1516     { school }
1517     { organization empty$ not
1518       { organization }
1519       { institution empty$ not
1520         { institution }
1521         { "" }
1522       if$
1523     }

```

```

1524         if$
1525     }
1526     if$
1527 }
1528 if$
1529 }
1530
1531 FUNCTION {format.address.publisher}
1532 { address empty$ not
1533   { address }
1534   { location empty$ not
1535     { location }
1536     { "" }
1537     if$
1538   }
1539   if$
1540   duplicates$ empty$ not
1541   { format.publisher empty$ not
1542     { bbl.colon * format.publisher * }
1543     { entry.is.electronic not show.missing.address.publisher and
1544       { bbl.colon * bbl.sine.nomine * }
1545       'skip$
1546       if$
1547     }
1548     if$
1549   }
1550   { pop$
1551     entry.is.electronic not show.missing.address.publisher and
1552     { format.publisher empty$ not
1553       { bbl.sine.loco bbl.colon * format.publisher * }
1554       { bbl.sine.loco.sine.nomine }
1555       if$
1556     }
1557     { format.publisher empty$ not
1558       { format.publisher }
1559       { "" }
1560       if$
1561     }
1562     if$
1563   }
1564   if$
1565 }
1566

```

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.

期刊需要著录起止范围，其中年份使用“/”分隔，卷和期使用“-”分隔。版本 v2.0.2 前的年份也使用“-”分隔，仅提供兼容性，不再推荐。

```

1567 FUNCTION {extract.before.dash}

```

```

1568 { duplicate$ empty$
1569   { pop$ "" }
1570   { 's :=
1571     #1 'charptr :=
1572     s text.length$ #1 + 'len :=
1573     { charptr len <
1574       s charptr #1 substring$ "-" = not
1575       and
1576     }
1577     { charptr #1 + 'charptr := }
1578     while$
1579     s #1 charptr #1 - substring$
1580   }
1581   if$
1582 }
1583
1584 FUNCTION {extract.after.dash}
1585 { duplicate$ empty$
1586   { pop$ "" }
1587   { 's :=
1588     #1 'charptr :=
1589     s text.length$ #1 + 'len :=
1590     { charptr len <
1591       s charptr #1 substring$ "-" = not
1592       and
1593     }
1594     { charptr #1 + 'charptr := }
1595     while$
1596     { charptr len <
1597       s charptr #1 substring$ "-" =
1598       and
1599     }
1600     { charptr #1 + 'charptr := }
1601     while$
1602     s charptr global.max$ substring$
1603   }
1604   if$
1605 }
1606
1607 FUNCTION {contains.dash}
1608 { duplicate$ empty$
1609   { pop$ #0 }
1610   { 's :=
1611     { s empty$ not
1612       s #1 #1 substring$ "-" = not
1613       and
1614     }
1615     { s #2 global.max$ substring$ 's := }
1616     while$
1617     s empty$ not
1618   }
1619   if$
1620 }
1621
1622 FUNCTION {extract.before.slash}

```

```

1623 { duplicate$ empty$
1624   { pop$ "" }
1625   { 's :=
1626     #1 'charptr :=
1627     s text.length$ #1 + 'len :=
1628     { charptr len <
1629       s charptr #1 substring$ "/" = not
1630       and
1631     }
1632     { charptr #1 + 'charptr := }
1633     while$
1634     s #1 charptr #1 - substring$
1635   }
1636   if$
1637 }
1638
1639 FUNCTION {extract.after.slash}
1640 { duplicate$ empty$
1641   { pop$ "" }
1642   { 's :=
1643     #1 'charptr :=
1644     s text.length$ #1 + 'len :=
1645     { charptr len <
1646       s charptr #1 substring$ "-" = not
1647       and
1648       s charptr #1 substring$ "/" = not
1649       and
1650     }
1651     { charptr #1 + 'charptr := }
1652     while$
1653     { charptr len <
1654       s charptr #1 substring$ "-" =
1655       s charptr #1 substring$ "/" =
1656       or
1657       and
1658     }
1659     { charptr #1 + 'charptr := }
1660     while$
1661     s charptr global.max$ substring$
1662   }
1663   if$
1664 }
1665
1666 FUNCTION {contains.slash}
1667 { duplicate$ empty$
1668   { pop$ #0 }
1669   { 's :=
1670     { s empty$ not
1671       s #1 #1 substring$ "-" = not
1672       and
1673       s #1 #1 substring$ "/" = not
1674       and
1675     }
1676     { s #2 global.max$ substring$ 's := }
1677     while$

```

```

1678     s empty$ not
1679   }
1680   if$
1681 }
1682

```

著者-出版年制必须提取出年份

```

1683 FUNCTION {format.year}
1684 { year empty$ not
1685   { year extract.before.slash extra.label * }
1686   { date empty$ not
1687     { date extract.before.dash extra.label * }
1688     { "empty year in " cite$ * warning$
1689       urldate empty$ not
1690       { "[" urldate extract.before.dash * extra.label * "]" * }
1691       { "" }
1692       if$
1693     }
1694     if$
1695   }
1696   if$
1697 }
1698
1699 FUNCTION {format.periodical.year}
1700 { year empty$ not
1701   { year extract.before.slash
1702     "--" *
1703     year extract.after.slash
1704     duplicate$ empty$
1705     'pop$
1706     { * }
1707     if$
1708   }
1709   { date empty$ not
1710     { date extract.before.dash }
1711     { "empty year in " cite$ * warning$
1712       urldate empty$ not
1713       { "[" urldate extract.before.dash * "]" * }
1714       { "" }
1715       if$
1716     }
1717     if$
1718   }
1719   if$
1720 }
1721

```

专利和报纸都是使用日期而不是年

```

1722 FUNCTION {format.date}
1723 { type$ "patent" = type$ "newspaper" = or
1724   date empty$ not and
1725   { date }
1726   { year field.or.null
1727     extra.label *
1728   }

```



```

1729 if$
1730 }
1731

```

更新、修改日期只用于电子资源 `electronic`

```

1732 FUNCTION {format.editeddate}
1733 { date empty$ not
1734   { "\allowbreak(" date * ")" * }
1735   { "" }
1736 if$
1737 }
1738

```

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。

```

1739 FUNCTION {format.urldate}
1740 { urldate empty$ not
1741   show.urldate show.url and is.pure.electronic or and
1742   url empty$ not and
1743   { "\allowbreak[" urldate * "]" * }
1744   { "" }
1745 if$
1746 }
1747

```

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 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。

```

1748 FUNCTION {hyphenate}
1749 { 't :=
1750   ""
1751   { t empty$ not }
1752   { t #1 #1 substring$ "-" =
1753     { "-" *
1754       { t #1 #1 substring$ "-" = }
1755       { t #2 global.max$ substring$ 't := }
1756       while$
1757     }
1758     { t #1 #1 substring$ *
1759       t #2 global.max$ substring$ 't :=
1760     }
1761     if$
1762   }
1763   while$
1764 }
1765

```

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

```

1766 FUNCTION {format.pages}
1767 { pages empty$
1768   { "" }
1769   { pages hyphenate }
1770   if$
1771 }
1772
1773 FUNCTION {format.extracted.pages}
1774 { pages empty$
1775   { "" }
1776   { pages
1777     only.start.page
1778     'extract.before.dash
1779     'hyphenate
1780     if$
1781   }
1782   if$
1783 }
1784

```

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。

```
1785 FUNCTION {format.journal.volume}
1786 { volume empty$ not
1787   { bold.journal.volume
1788     { "\textbf{" volume * "}" * }
1789     { volume }
1790     if$
1791   }
1792   { "" }
1793   if$
1794 }
1795
1796 FUNCTION {format.journal.number}
1797 { number empty$ not
1798   { "\allowbreak (" number * ")" * }
1799   { "" }
1800   if$
1801 }
1802
1803 FUNCTION {format.journal.pages}
1804 { pages empty$
1805   { "" }
1806   { format.extracted.pages }
1807   if$
1808 }
1809
```

连续出版物的年卷期有起止范围，需要特殊处理

```
1810 FUNCTION {format.periodical.year.volume.number}
1811 { year empty$ not
1812   { year extract.before.slash }
1813   { "empty year in periodical " cite$ * warning$ }
1814   if$
1815   volume empty$ not
1816     { ", " * volume extract.before.dash * }
1817     'skip$
1818   if$
1819   number empty$ not
1820     { "\allowbreak (" * number extract.before.dash * ")" * }
1821     'skip$
1822   if$
1823   "--" *
1824   year extract.after.slash empty$
1825   volume extract.after.dash empty$ and
1826   number extract.after.dash empty$ and not
1827     { year extract.after.slash empty$ not
1828       { year extract.after.slash * }
1829       { year extract.before.slash * }
1830       if$
1831       volume empty$ not
```

```

1832         { ", " * volume extract.after.dash * }
1833         'skip$
1834     if$
1835     number empty$ not
1836     { "\allowbreak (" * number extract.after.dash * ")" * }
1837     'skip$
1838     if$
1839 }
1840 'skip$
1841 if$
1842 }
1843

```

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。

```

1844 FUNCTION {check.url}
1845 { url empty$ not
1846   { "\url{" url * "}" * 'entry.url :=
1847     #1 'entry.is.electronic :=
1848   }
1849   { howpublished empty$ not
1850     { howpublished #1 #5 substring$ "\url{" =
1851       { howpublished 'entry.url :=
1852         #1 'entry.is.electronic :=
1853       }
1854       'skip$
1855     if$
1856   }
1857   { note empty$ not
1858     { note #1 #5 substring$ "\url{" =
1859       { note 'entry.url :=
1860         #1 'entry.is.electronic :=
1861       }
1862       'skip$
1863     if$
1864   }
1865   'skip$
1866   if$
1867 }
1868   if$
1869 }
1870 if$
1871 }
1872
1873 FUNCTION {format.url}
1874 { entry.url
1875 }
1876
1877 FUNCTION {output.url}
1878 { entry.url empty$ not
1879   { new.block
1880     entry.url output
1881   }

```

```

1882     'skip$
1883   if$
1884 }
1885

```

需要检测 DOI 是否已经包含在 URL 中。

```

1886 FUNCTION {check.doi}
1887 { doi empty$ not
1888   { #1 'entry.is.electronic := }
1889   'skip$
1890   if$
1891 }
1892
1893 FUNCTION {is.in.url}
1894 { 's :=
1895   s empty$
1896   { #1 }
1897   { entry.url empty$
1898     { #0 }
1899     { s text.length$ 'len :=
1900       entry.url text.length$ 'charptr :=
1901       { entry.url charptr len substring$ s = not
1902         charptr #0 >
1903         and
1904       }
1905       { charptr #1 - 'charptr := }
1906       while$
1907       charptr
1908     }
1909     if$
1910   }
1911   if$
1912 }
1913
1914 FUNCTION {format.doi}
1915 { ""
1916   doi empty$ not
1917   { "" 's :=
1918     doi 't :=
1919     #0 'numnames :=
1920     { t empty$ not}
1921     { t #1 #1 substring$ 'tmp.str :=
1922       tmp.str "," = tmp.str " " = or t #2 #1 substring$ empty$ or
1923       { t #2 #1 substring$ empty$
1924         { s tmp.str * 's := }
1925         'skip$
1926       if$
1927       s empty$ s is.in.url or
1928       'skip$
1929       { numnames #1 + 'numnames :=
1930         numnames #1 >
1931         { ", " * }
1932         { "DOI: " * }
1933         if$
1934         "\doi{" s * "}" * *

```

```

1935         }
1936         if$
1937         "" 's :=
1938     }
1939     { s tmp.str * 's := }
1940     if$
1941     t #2 global.max$ substring$ 't :=
1942 }
1943 while$
1944 }
1945 'skip$
1946 if$
1947 }
1948
1949 FUNCTION {output.doi}
1950 { doi empty$ not show.doi and
1951   show.english.translation entry.lang lang.zh = and not and
1952   { new.block
1953     format.doi output
1954   }
1955   'skip$
1956   if$
1957 }
1958
1959 FUNCTION {check.electronic}
1960 { "" 'entry.url :=
1961   #0 'entry.is.electronic :=
1962   'check.doi
1963   'skip$
1964   if$
1965   'check.url
1966   'skip$
1967   if$
1968   medium empty$ not
1969   { medium "MT" = medium "DK" = or medium "CD" = or medium "OL" = or
1970     { #1 'entry.is.electronic := }
1971     'skip$
1972     if$
1973   }
1974   'skip$
1975   if$
1976 }
1977
1978 FUNCTION {format.eprint}
1979 { eprinttype empty$ not
1980   { eprinttype }
1981   { archivePrefix empty$ not
1982     { archivePrefix }
1983     { "" }
1984     if$
1985   }
1986   if$
1987   's :=
1988   s empty$ not
1989   { s ": \eprint{" *

```

```

1990     url empty$ not
1991     { url }
1992     { "https://" s "l" change.case$ * ".org/abs/" * eprint * }
1993     if$
1994     * "}" *
1995     eprint * "}" *
1996     }
1997     { eprint }
1998     if$
1999 }
2000
2001 FUNCTION {output.eprint}
2002 { show.preprint eprint empty$ not and
2003   { new.block
2004     format.eprint output
2005   }
2006   'skip$
2007   if$
2008 }
2009
2010 FUNCTION {format.note}
2011 { note empty$ not show.note and
2012   { note }
2013   { "" }
2014   if$
2015 }
2016
2017 FUNCTION {output.translation}
2018 { show.english.translation entry.lang lang.zh = and
2019   { translation empty$ not
2020     { translation }
2021     { "[English translation missing!]" }
2022     if$
2023     " (in Chinese)" * output
2024     write$
2025     format.doi duplicate$ empty$ not
2026     { newline$
2027       write$
2028     }
2029     'pop$
2030     if$
2031     " \\" write$
2032     newline$
2033     "(" write$
2034     ""
2035     before.all 'output.state :=
2036   }
2037   'skip$
2038   if$
2039 }
2040

```

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

```

2041 FUNCTION {empty.misc.check}
2042 { author empty$ title empty$
2043   year empty$
2044   and and
2045   key empty$ not and
2046   { "all relevant fields are empty in " cite$ * warning$ }
2047   'skip$
2048   if$
2049 }
2050

```

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

```

2051 FUNCTION {monograph}
2052 { output.bibitem
2053   output.translation
2054   author empty$ not
2055     { format.authors }
2056     { editor empty$ not
2057       { format.editors }
2058       { "empty author and editor in " cite$ * warning$
2059 (*authoryear)
2060       bbl.anonymous
2061 (/authoryear)
2062 (*numerical)
2063       ""
2064 (/numerical)
2065     }
2066     if$
2067   }
2068   if$
2069   output
2070   year.after.author
2071   { period.after.author
2072     'new.sentence
2073     'skip$
2074     if$
2075     format.year "year" output.check
2076   }
2077   'skip$
2078   if$
2079   new.block

```



```

2080 format.series.vol.num.title "title" output.check
2081 "M" set.entry.mark
2082 format.mark "" output.after
2083 new.block
2084 format.translators output
2085 new.sentence
2086 format.edition output
2087 new.block
2088 format.address.publisher output
2089 year.after.author not
2090 { format.year "year" output.check }
2091 'skip$
2092 if$
2093 format.pages bbl.pages.colon output.after
2094 format.urldate "" output.after
2095 output.url
2096 output.doi
2097 new.block
2098 format.note output
2099 fin.entry
2100 }
2101

```

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

```

2102 FUNCTION {incollection}
2103 { output.bibitem
2104   output.translation
2105   format.authors output
2106   author format.key output
2107   year.after.author
2108     { period.after.author
2109       'new.sentence
2110       'skip$
2111       if$
2112       format.year "year" output.check
2113     }
2114   'skip$
2115   if$
2116   new.block
2117   format.title "title" output.check
2118   "M" set.entry.mark
2119   format.mark "" output.after
2120   new.block
2121   format.translators output
2122   new.slash

```

```

2123 format.editors output
2124 new.block
2125 format.series.vol.num.booktitle "booktitle" output.check
2126 new.block
2127 format.edition output
2128 new.block
2129 format.address.publisher output
2130 year.after.author not
2131     { format.year "year" output.check }
2132     'skip$
2133 if$
2134 format.extracted.pages bbl.pages.colon output.after
2135 format.urldate "" output.after
2136 output.url
2137 output.doi
2138 new.block
2139 format.note output
2140 fin.entry
2141 }
2142

```

B.5.3 连续出版物

```

2143 FUNCTION {periodical}
2144 { output.bibitem
2145   output.translation
2146   format.authors output
2147   author format.key output
2148   year.after.author
2149     { period.after.author
2150       'new.sentence
2151       'skip$
2152       if$
2153       format.year "year" output.check
2154     }
2155     'skip$
2156   if$
2157   new.block
2158   format.title "title" output.check
2159   "J" set.entry.mark
2160   format.mark "" output.after
2161   new.block
2162   format.periodical.year.volume.number output
2163   new.block
2164   format.address.publisher output
2165   year.after.author not
2166     { format.periodical.year "year" output.check }
2167     'skip$
2168   if$
2169   format.urldate "" output.after
2170   output.url
2171   output.doi
2172   new.block
2173   format.note output
2174   fin.entry

```

2175 }
2176

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.

```
2177 FUNCTION {article}  
2178 { output.bibitem  
2179   output.translation  
2180   format.authors output  
2181   author format.key output  
2182   year.after.author  
2183     { period.after.author  
2184       'new.sentence  
2185       'skip$  
2186       if$  
2187         format.year "year" output.check  
2188       }  
2189     'skip$  
2190   if$  
2191   new.block  
2192   title.in.journal  
2193     { format.title "title" output.check  
2194       "J" set.entry.mark  
2195       format.mark "" output.after  
2196       new.block  
2197     }  
2198   'skip$  
2199   if$  
2200   format.journal "journal" output.check  
2201   year.after.author not  
2202     { format.date "year" output.check }  
2203   'skip$  
2204   if$  
2205   format.journal.volume output  
2206   format.journal.number "" output.after  
2207   format.journal.pages bbl.pages.colon output.after  
2208   format.urldate "" output.after  
2209   output.url  
2210   output.doi  
2211   new.block  
2212   format.note output  
2213   fin.entry  
2214 }  
2215
```

B.5.5 专利文献

number 域也可以用来表示专利号。

```
2216 FUNCTION {patent}
2217 { output.bibitem
2218   output.translation
2219   format.authors output
2220   author format.key output
2221   year.after.author
2222     { period.after.author
2223       'new.sentence
2224       'skip$
2225       if$
2226       format.year "year" output.check
2227     }
2228     'skip$
2229   if$
2230   new.block
2231   format.title "title" output.check
2232   "P" set.entry.mark
2233   format.mark "" output.after
2234   new.block
2235   format.date "year" output.check
2236   format.urldate "" output.after
2237   output.url
2238   output.doi
2239   new.block
2240   format.note output
2241   fin.entry
2242 }
2243
```

B.5.6 电子资源

```
2244 FUNCTION {electronic}
2245 { #1 #1 check.electronic
2246   #1 'entry.is.electronic :=
2247   #1 'is.pure.electronic :=
2248   output.bibitem
2249   output.translation
2250   format.authors output
2251   author format.key output
2252   year.after.author
2253     { period.after.author
2254       'new.sentence
2255       'skip$
2256       if$
2257       format.year "year" output.check
2258     }
2259     'skip$
2260   if$
2261   new.block
2262   format.series.vol.num.title "title" output.check
2263   "EB" set.entry.mark

```

```

2264 format.mark "" output.after
2265 new.block
2266 format.address.publisher output
2267 year.after.author not
2268   { date empty$
2269     { format.date output }
2270     'skip$
2271     if$
2272   }
2273   'skip$
2274 if$
2275 format.pages bbl.pages.colon output.after
2276 format.editdate "" output.after
2277 format.urldate "" output.after
2278 output.url
2279 output.doi
2280 new.block
2281 format.note output
2282 fin.entry
2283 }
2284

```

B.5.7 预印本

```

2285 FUNCTION {preprint}
2286 { output.bibitem
2287   output.translation
2288   author empty$ not
2289     { format.authors }
2290     { editor empty$ not
2291       { format.editors }
2292       { "empty author and editor in " cite$ * warning$
2293 (*authoryear)
2294       bbl.anonymous
2295 (/authoryear)
2296 (*numerical)
2297       ""
2298 (/numerical)
2299     }
2300     if$
2301   }
2302   if$
2303   output
2304   year.after.author
2305     { period.after.author
2306       'new.sentence
2307       'skip$
2308       if$
2309       format.year "year" output.check
2310     }
2311     'skip$
2312   if$
2313   new.block
2314   title.in.journal
2315     { format.series.vol.num.title "title" output.check
2316 (*2015)

```

```

2317         "Z" set.entry.mark
2318 </2015>
2319 <*2005>
2320         "M" set.entry.mark
2321 </2005>
2322         format.mark "" output.after
2323         new.block
2324     }
2325     'skip$
2326 if$
2327 format.translators output
2328 new.sentence
2329 format.edition output
2330 new.block
2331 output.eprint
2332 year.after.author not
2333     { format.year "year" output.check }
2334     'skip$
2335 if$
2336 format.pages bbl.pages.colon output.after
2337 format.urldate "" output.after
2338 output.url
2339 new.block
2340 format.note output
2341 fin.entry
2342 }
2343

```

B.5.8 其他文献类型

A misc is something that doesn't fit elsewhere.

Required: at least one of the 'optional' fields

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

Misc 用来自动判断类型。

```

2344 FUNCTION {misc}
2345 { journal empty$ not
2346     'article
2347     { booktitle empty$ not
2348         'incollection
2349         { publisher empty$ not
2350             'monograph
2351             { eprint empty$ not show.preprint and
2352                 'preprint
2353                 { entry.is.electronic
2354                     'electronic
2355                     {
2356 <*2015>
2357                 "Z" set.entry.mark
2358 </2015>
2359 <*2005>
2360                 "M" set.entry.mark
2361 </2005>
2362                 monograph

```

```

2363         }
2364     if$
2365 }
2366     if$
2367 }
2368     if$
2369 }
2370     if$
2371 }
2372 if$
2373 empty.misc.check
2374 }
2375
2376 FUNCTION {archive}
2377 { "A" set.entry.mark
2378   misc
2379 }
2380

```

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

```

2381 FUNCTION {book} { monograph }
2382

```

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

Required: title

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

```

2383 FUNCTION {booklet} { book }
2384
2385 FUNCTION {collection}
2386 { "G" set.entry.mark
2387   monograph
2388 }
2389
2390 FUNCTION {database}
2391 { "DB" set.entry.mark
2392   electronic
2393 }
2394
2395 FUNCTION {dataset}
2396 { "DS" set.entry.mark
2397   electronic
2398 }
2399

```

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。

```
2400 FUNCTION {inbook} { book }
2401
```

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

```
2402 FUNCTION {inproceedings}
2403 { "C" set.entry.mark
2404   incollection
2405 }
2406
```

The conference function is included for Scribe compatibility.

```
2407 FUNCTION {conference} { inproceedings }
2408
2409 FUNCTION {map}
2410 { "CM" set.entry.mark
2411   misc
2412 }
2413
```

A manual is technical documentation.

Required: title

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

```
2414 FUNCTION {manual} { monograph }
2415
```

A mastersthesis is a Master's thesis.

Required: author, title, school, year

Optional: type, address, month, note

```
2416 FUNCTION {mastersthesis}
2417 { "D" set.entry.mark
2418   monograph
2419 }
2420
2421 FUNCTION {newspaper}
2422 { "N" set.entry.mark
2423   article
2424 }
2425
2426 FUNCTION {online}
2427 { "EB" set.entry.mark
2428   electronic
2429 }
2430
```


A phdthesis is like a mastersthesis.

Required: author, title, school, year

Optional: type, address, month, note

```
2431 FUNCTION {phdthesis} { mastersthesis }  
2432
```

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

```
2433 FUNCTION {proceedings}  
2434 { "C" set.entry.mark  
2435   monograph  
2436 }  
2437  
2438 FUNCTION {software}  
2439 { "CP" set.entry.mark  
2440   electronic  
2441 }  
2442  
2443 FUNCTION {standard}  
2444 { "S" set.entry.mark  
2445   misc  
2446 }  
2447
```

A techreport is a technical report.

Required: author, title, institution, year

Optional: type, number, address, month, note

```
2448 FUNCTION {techreport}  
2449 { "R" set.entry.mark  
2450   misc  
2451 }  
2452
```

An unpublished is something that hasn't been published.

Required: author, title, note

Optional: month, year

```
2453 FUNCTION {unpublished} { misc }  
2454
```

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

```
2455 FUNCTION {default.type} { misc }  
2456
```

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

```
2457 MACRO {jan} {"January"}
2458
2459 MACRO {feb} {"February"}
2460
2461 MACRO {mar} {"March"}
2462
2463 MACRO {apr} {"April"}
2464
2465 MACRO {may} {"May"}
2466
2467 MACRO {jun} {"June"}
2468
2469 MACRO {jul} {"July"}
2470
2471 MACRO {aug} {"August"}
2472
2473 MACRO {sep} {"September"}
2474
2475 MACRO {oct} {"October"}
2476
2477 MACRO {nov} {"November"}
2478
2479 MACRO {dec} {"December"}
2480
```

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

```
2481 MACRO {acmcs} {"ACM Computing Surveys"}
2482
2483 MACRO {acta} {"Acta Informatica"}
2484
2485 MACRO {cacm} {"Communications of the ACM"}
2486
2487 MACRO {ibmjrd} {"IBM Journal of Research and Development"}
2488
2489 MACRO {ibmsj} {"IBM Systems Journal"}
2490
2491 MACRO {ieeese} {"IEEE Transactions on Software Engineering"}
2492
2493 MACRO {ieeetc} {"IEEE Transactions on Computers"}
2494
2495 MACRO {ieeetcad} {"IEEE Transactions on Computer-Aided Design of Integrated Circuits"}
2496
```

```

2497
2498 MACRO {ipl} {"Information Processing Letters"}
2499
2500 MACRO {jacm} {"Journal of the ACM"}
2501
2502 MACRO {jcss} {"Journal of Computer and System Sciences"}
2503
2504 MACRO {scp} {"Science of Computer Programming"}
2505
2506 MACRO {sicomp} {"SIAM Journal on Computing"}
2507
2508 MACRO {tocs} {"ACM Transactions on Computer Systems"}
2509
2510 MACRO {tods} {"ACM Transactions on Database Systems"}
2511
2512 MACRO {tog} {"ACM Transactions on Graphics"}
2513
2514 MACRO {toms} {"ACM Transactions on Mathematical Software"}
2515
2516 MACRO {toois} {"ACM Transactions on Office Information Systems"}
2517
2518 MACRO {toplas} {"ACM Transactions on Programming Languages and Systems"}
2519
2520 MACRO {tcs} {"Theoretical Computer Science"}
2521

```

B.7 Format labels

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

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

```

2522 FUNCTION {sortify}
2523 { purify$
2524   "l" change.case$
2525 }
2526

```

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

```

2527 FUNCTION {chop.word}
2528 { 's :=
2529   'len :=
2530   s #1 len substring$ =
2531     { s len #1 + global.max$ substring$ }
2532     's
2533   if$
2534 }
2535

```

The `format.lab.names` function makes a short label by using the initials of the von and Last parts of the names (but if there are more than four names, (i.e.,

people) it truncates after three and adds a superscripted ”+”; it also adds such a ”+” if the last of multiple authors is ”others”). If there is only one name, and its von and Last parts combined have just a single name-token (”Knuth” has a single token, ”Brinch Hansen” has two), we take the first three letters of the last name. The boolean `et.al.char.used` tells whether we’ve used a superscripted ”+”, so that we know whether to include a LaTeX macro for it.

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

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

There is one auxiliary function for each of the four different sequences of fields

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

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

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

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

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

```
2536 FUNCTION {format.lab.name}
2537 { "{vv~}{ll}{, jj}{, ff}" format.name$ 't :=
2538   t "others" =
2539     { citation.et.al }
2540     { t get.str.lang 'name.lang :=
2541       name.lang lang.zh = name.lang lang.ja = or
2542         { t #1 "{ll}{ff}" format.name$ }
2543         { t #1 "{vv~}{ll}" format.name$ }
2544       if$
2545     }
2546   if$
2547 }
```

第一作者姓名相同、年份相同但作者数量不同时，也需要年份标签区分。比如“王临惠 等, 2010a”和“王临惠, 2010b”，所以使用 `short.label` 存储不带“et

al”的版本。

```
2549 FUNCTION {format.lab.names}
2550 { 's :=
2551   s #1 format.lab.name 'short.label :=
2552   #1 'nameptr :=
2553   s num.names$ 'numnames :=
2554   ""
2555   numnames 'namesleft :=
2556   { namesleft #0 > }
2557   { s nameptr format.lab.name citation.et.al =
2558     numnames citation.et.al.min #1 - > nameptr citation.et.al.use.first > and or
2559     { bbl.space *
2560       citation.et.al *
2561       #1 'namesleft :=
2562     }
2563     { nameptr #1 >
2564       { namesleft #1 = citation.and "" = not and
2565         { citation.and * }
2566         { ", " * }
2567         if$
2568       }
2569       'skip$
2570       if$
2571       s nameptr format.lab.name *
2572     }
2573     if$
2574     nameptr #1 + 'nameptr :=
2575     namesleft #1 - 'namesleft :=
2576   }
2577   while$
2578 }
2579
2580 FUNCTION {author.key.label}
2581 { author empty$
2582   { key empty$
2583     { cite$ #1 #3 substring$ }
2584     'key
2585     if$
2586   }
2587   { author format.lab.names }
2588   if$
2589 }
2590
2591 FUNCTION {author.editor.key.label}
2592 { author empty$
2593   { editor empty$
2594     { key empty$
2595       { cite$ #1 #3 substring$ }
2596       'key
2597       if$
2598     }
2599     { editor format.lab.names }
2600     if$
2601   }
2602   { author format.lab.names }
```

```

2603 if$
2604 }
2605
2606 FUNCTION {author.key.organization.label}
2607 { author empty$
2608   { key empty$
2609     { organization empty$
2610       { cite$ #1 #3 substring$ }
2611       { "The " #4 organization chop.word #3 text.prefix$ }
2612       if$
2613     }
2614     'key
2615     if$
2616   }
2617   { author format.lab.names }
2618   if$
2619 }
2620
2621 FUNCTION {editor.key.organization.label}
2622 { editor empty$
2623   { key empty$
2624     { organization empty$
2625       { cite$ #1 #3 substring$ }
2626       { "The " #4 organization chop.word #3 text.prefix$ }
2627       if$
2628     }
2629     'key
2630     if$
2631   }
2632   { editor format.lab.names }
2633   if$
2634 }
2635
2636 FUNCTION {calc.short.authors}
2637 { "" 'short.label :=
2638   type$ "book" =
2639   type$ "inbook" =
2640   or
2641   'author.editor.key.label
2642   { type$ "collection" =
2643     type$ "proceedings" =
2644     or
2645     { editor empty$ not
2646       'editor.key.organization.label
2647       'author.key.organization.label
2648       if$
2649     }
2650     'author.key.label
2651     if$
2652   }
2653   if$
2654   'short.list :=
2655   short.label empty$
2656   { short.list 'short.label := }
2657   'skip$

```

```

2658 if$
2659 }
2660
2661 FUNCTION {calc.label}
2662 { calc.short.authors
2663   short.list
2664   "("
2665   *
2666   format.year duplicate$ empty$
2667   short.list key field.or.null = or
2668     { pop$ "" }
2669     'skip$
2670   if$
2671   *
2672   'label :=
2673   short.label
2674   "("
2675   *
2676   format.year duplicate$ empty$
2677   short.list key field.or.null = or
2678     { pop$ "" }
2679     'skip$
2680   if$
2681   *
2682   'short.label :=
2683 }
2684

```

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`.


```

2685 (*authoryear)
2686 FUNCTION {sort.language.label}
2687 { entry.lang lang.zh =
2688   { lang.zh.order }
2689   { entry.lang lang.ja =
2690     { lang.ja.order }
2691     { entry.lang lang.en =
2692       { lang.en.order }
2693       { entry.lang lang.ru =
2694         { lang.ru.order }
2695         { lang.other.order }
2696       if$
2697     }
2698   if$
2699 }
2700 if$
2701 }
2702 if$
2703 #64 +
2704 int.to.chr$
2705 }
2706
2707 FUNCTION {sort.format.names}
2708 { 's :=
2709   #1 'nameptr :=
2710   ""
2711   s num.names$ 'numnames :=
2712   numnames 'namesleft :=
2713   { namesleft #0 > }
2714   {
2715     s nameptr "{vv{ } }{ll{ }}{ ff{ }}{ jj{ }}" format.name$ 't :=
2716     nameptr #1 >
2717     {
2718       " " *
2719       namesleft #1 = t "others" = and
2720       { "zzzz" * }
2721       { numnames #2 > nameptr #2 = and
2722         { "zz" * year field.or.null * " " * }
2723         'skip$
2724       if$
2725       t sortify *
2726     }
2727     if$
2728   }
2729   { t sortify * }
2730   if$
2731   nameptr #1 + 'nameptr :=
2732   namesleft #1 - 'namesleft :=
2733 }
2734 while$
2735 }
2736

```

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
2737 FUNCTION {sort.format.title}
2738 { 't :=
2739   "A " #2
2740   "An " #3
2741   "The " #4 t chop.word
2742   chop.word
2743   chop.word
2744   sortify
2745   #1 global.max$ substring$
2746 }
2747

```

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.

```

2748 FUNCTION {anonymous.sort}
2749 { entry.lang lang.zh =
2750   { "yi4 ming2" }
2751   { "anon" }
2752   if$
2753 }
2754
2755 FUNCTION {warn.empty.key}
2756 { entry.lang lang.zh =
2757   { "empty key in " cite$ * warning$ }
2758   'skip$
2759   if$
2760 }
2761
2762 FUNCTION {author.sort}
2763 { key empty$
2764   { warn.empty.key
2765     author empty$
2766       { anonymous.sort }
2767       { author sort.format.names }
2768       if$
2769   }
2770   { key }
2771   if$
2772 }
2773
2774 FUNCTION {author.editor.sort}
2775 { key empty$
2776   { warn.empty.key
2777     author empty$
2778       { editor empty$
2779         { anonymous.sort }
2780         { editor sort.format.names }
2781         if$
2782       }
2783       { author sort.format.names }

```

```

2784     if$
2785   }
2786   { key }
2787   if$
2788 }
2789
2790 FUNCTION {author.organization.sort}
2791 { key empty$
2792   { warn.empty.key
2793     author empty$
2794       { organization empty$
2795         { anonymous.sort }
2796         { "The " #4 organization chop.word sortify }
2797         if$
2798       }
2799       { author sort.format.names }
2800     if$
2801   }
2802   { key }
2803   if$
2804 }
2805
2806 FUNCTION {editor.organization.sort}
2807 { key empty$
2808   { warn.empty.key
2809     editor empty$
2810       { organization empty$
2811         { anonymous.sort }
2812         { "The " #4 organization chop.word sortify }
2813         if$
2814       }
2815       { editor sort.format.names }
2816     if$
2817   }
2818   { key }
2819   if$
2820 }
2821
2822 </authoryear>

```

顺序编码制的排序要简单得多

```

2823 < *numerical>
2824 INTEGERS { seq.num }
2825
2826 FUNCTION {init.seq}
2827 { #0 'seq.num :=}
2828
2829 FUNCTION {int.to.fix}
2830 { "000000000" swap$ int.to.str$ *
2831   #-1 #10 substring$
2832 }
2833
2834 </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!

```

2835 FUNCTION {presort}
2836 { set.entry.lang
2837   set.entry.numbered
2838   show.url show.doi check.electronic
2839   #0 'is.pure.electronic :=
2840   calc.label
2841   label sortify
2842   " "
2843   *
2844   (*authoryear)
2845   sort.language.label
2846   " "
2847   *
2848   type$ "book" =
2849   type$ "inbook" =
2850   or
2851     'author.editor.sort
2852     { type$ "collection" =
2853       type$ "proceedings" =
2854       or
2855         'editor.organization.sort
2856         'author.sort
2857       if$
2858     }
2859   if$
2860   *
2861   " "
2862   *
2863   year field.or.null sortify
2864   *
2865   " "
2866   *
2867   cite$
2868   *
2869   #1 entry.max$ substring$
2870   (/authoryear)
2871   (*numerical)
2872   seq.num #1 + 'seq.num :=
2873   seq.num int.to.fix
2874   (/numerical)
2875   'sort.label :=
2876   sort.label *
2877   #1 entry.max$ substring$
2878   'sort.key$ :=
2879 }
2880

```

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
```

```
2881 STRINGS { longest.label last.label next.extra }
2882
2883 INTEGERS { longest.label.width last.extra.num number.label }
2884
2885 FUNCTION {initialize.longest.label}
2886 { "" 'longest.label :=
2887   #0 int.to.chr$ 'last.label :=
2888   "" 'next.extra :=
2889   #0 'longest.label.width :=
2890   #0 'last.extra.num :=
2891   #0 'number.label :=
2892 }
2893
2894 FUNCTION {forward.pass}
2895 { last.label short.label =
2896   { last.extra.num #1 + 'last.extra.num :=
```

```

2897     last.extra.num int.to.chr$ 'extra.label :=
2898   }
2899   { "a" chr.to.int$ 'last.extra.num :=
2900     "" 'extra.label :=
2901     short.label 'last.label :=
2902   }
2903   if$
2904   number.label #1 + 'number.label :=
2905 }
2906
2907 FUNCTION {reverse.pass}
2908 { next.extra "b" =
2909   { "a" 'extra.label := }
2910   'skip$
2911   if$
2912   extra.label 'next.extra :=
2913   extra.label
2914   duplicate$ empty$
2915     'skip$
2916     { "{\natexlab{" swap$ * "}" * }
2917     if$
2918     'extra.label :=
2919     label extra.label * 'label :=
2920 }
2921
2922 FUNCTION {bib.sort.order}
2923 { sort.label 'sort.key$ :=
2924 }
2925

```

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.

```

2926 FUNCTION {begin.bib}
2927 { preamble$ empty$
2928   'skip$
2929   { preamble$ write$ newline$ }
2930   if$
2931   "\begin{thebibliography}{ " number.label int.to.str$ * "}" *
2932   write$ newline$
2933   terms.in.macro
2934   { "\providecommand{\biband}{和}"
2935     write$ newline$
2936     "\providecommand{\bibetal}{等}"
2937     write$ newline$
2938   }
2939   'skip$
2940   if$

```

```

2941 "\providecommand{\natexlab}[1]{#1}"
2942 write$ newline$
2943 "\providecommand{\url}[1]{#1}"
2944 write$ newline$
2945 "\expandafter\ifx\csname urlstyle\endcsname\relax\else"
2946 write$ newline$
2947 " \urlstyle{same}\fi"
2948 write$ newline$
2949 "\expandafter\ifx\csname href\endcsname\relax"
2950 write$ newline$
2951 " \DeclareUrlCommand\doi{\urlstyle{rm}}"
2952 write$ newline$
2953 " \def\eprint#1#2{#2}"
2954 write$ newline$
2955 "\else"
2956 write$ newline$
2957 " \def\doi#1{\href{https://doi.org/#1}{\nolinkurl{#1}}}"
2958 write$ newline$
2959 " \let\eprint\href"
2960 write$ newline$
2961 "\fi"
2962 write$ newline$
2963 }
2964

```

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

```

2965 FUNCTION {end.bib}
2966 { newline$
2967 "\end{thebibliography}" write$ newline$
2968 }
2969

```

B.10 Main execution

Now we read in the .BIB entries.

```

2970 READ
2971
2972 EXECUTE {init.state.consts}
2973
2974 EXECUTE {load.config}
2975
2976 {*numerical}
2977 EXECUTE {init.seq}
2978
2979 {/numerical}
2980 ITERATE {presort}
2981

```

And now we can sort

```

2982 SORT
2983
2984 EXECUTE {initialize.longest.label}
2985
2986 ITERATE {forward.pass}

```

```
2987
2988 REVERSE {reverse.pass}
2989
2990 ITERATE {bib.sort.order}
2991
2992 SORT
2993
2994 EXECUTE {begin.bib}
2995
```

Now we produce the output for all the entries

```
2996 ITERATE {call.type$}
2997
2998 EXECUTE {end.bib}
2999 </authoryear | numerical>
```