

GB/T 7714 BibTeX style

Zeping Lee*

2022/10/03 v2.1.5

摘要

The gbt7714 package provides a BibTeX implementation for the China's national bibliography style standard GB/T 7714. It consists of .bst files for numeric and author-date 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”）是中国的参考文献格式推荐标准。国内的绝大部分学术期刊、学位论文都使用了基于该标准的格式。本宏包是国标的 BibTeX^[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。
- 同时提供了 2005 版的 .bst 文件。

本宏包的主页：<https://github.com/zepinglee/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 `\biblilographystyle` 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 LaTeX 接口不一致，所以将被弃用。

*zepinglee AT gmail.com

3 使用方法

以下是 gbt7714 宏包的一个简单示例。

```
\documentclass{ctexart}
\usepackage{gbt7714}
\bibliographystyle{gbt7714-numerical}
\begin{document}
  \cite{...}
  ...
  \bibliography{bibfile}
\end{document}
```

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
```

此外还可以使用 2005 版的格式 gbt7714-2005-numerical 和 gbt7714-2005-author-year。

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。

```
\citestyle \citestyle{<citation style>}
```

可选：super, numbers, author-year。使用 \bibliography 选择参考文献表的样式时会自动设置对应的引用样式。顺序编码制的引用标注默认使用角标式（super），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 命令切换为数字式（numbers）。

```
\citestyle{numbers}
```

著者-出版年制通常不需要修改引用样式。

sort&compress (env.) 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，这些参数会传给 natbib 处理。

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

注意国标中要求 2 个或以上的连续编号用连接号，不同于 `natbib` 默认的 3 个或以上。宏包中已经作了修改。

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` 和 `biblatex` 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_T_EX 的标准域，必须新增一些著录项目（带 * 号），这些新增的类型在设计时参考了 BibLaTeX，如 date 和 urldate。本宏包支持的全部域如下：

author 主要责任者
title 题名
mark* 文献类型标识
medium* 载体类型标识
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 BibTeX 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用于排序，如：

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

对于著者-出版年的样式，如果中文文献较多时更推荐使用 `biblatex` 宏包，其后端 `biber` 可以自动处理中文按照拼音排序，无须手动填写拼音。

7 自定义样式

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

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

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 BibTeX 样式，支持顺序编码制和著者出版年制两种风格。李志奇^[8]发布了严格遵循 GB/T 7714—2005 的 BibLaTeX 的样式。胡海星^[9]提供

表 2: 参考文献表样式的配置参数

参数值	默认值	功能
uppercase.name	#1	将著者姓名转为大写
max.num.authors	#3	输出著者的最多数
year.after.author	#0	年份置于著者之后
period.after.author	#0	著者和年份之间使用句点连接
italic.book.title	#0	西文书籍名使用斜体
sentence.case.title	#1	将西文的题名转为 sentence case
link.title	#0	在题名上添加 url 的超链接
title.in.journal	#1	期刊是否显示标题
show.patent.country	#0	专利题名是否含国别
space.before.mark	#0	文献类型标识前是否有空格
show.mark	#1	显示文献类型标识
show.medium.type	#1	显示载体类型标识
component.part.label	"slash"	表示析出文献的符号, 可选: "in", "none"
italic.journal	#0	西文期刊名使用斜体
show.missing.address.publisher	#0	出版项缺失时显示“出版者不详”
space.before.pages	#1	页码与前面的冒号之间有空格
only.start.page	#0	只显示起始页码
wave.dash.in.pages	#0	起止页码使用波浪号
show.urldate	#1	显示引用日期 urldate
show.url	#1	显示 url
show.doi	#1	显示 DOI
show.preprint	#1	显示预印本信息
show.note	#0	显示 note 域的信息
end.with.period	#1	结尾加句点

了另一个 Bib_TE_X 实现, 还给每行 bst 代码写了 java 语言注释。沈周^[10]基于 biblatex-caservector^[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. Bib_TE_Xing[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 BibT_EX styles[M/OL]. 1988. <http://mirrors.ctan.org/biblio/bibtex/base/btxhak.pdf>.
- [5] MARKEY N. Tame the beast[M/OL]. 2003. http://mirrors.ctan.org/info/bibtex/tamethebeast/ttb_en.pdf.
- [6] MITTELBAACH F, GOOSSENS M, BRAAMS J, et al. The L^AT_EX companion[M]. 2nd ed. Reading, MA, USA: Addison-Wesley, 2004.
- [7] 吴凯. 发布 GBT7714-2005.bst version1 Beta 版 [EB/OL]. 2006. CTeX 论坛（已关闭）.
- [8] 李志奇. 基于 biblatex 的符合 GB/T7714—2005 的中文文献生成工具 [EB/OL]. 2013. CTeX 论坛（已关闭）.
- [9] 胡海星. A GB/T 7714—2005 national standard compliant BibTeX style[EB/OL]. 2013. <https://github.com/Haixing-Hu/GBT7714-2005-BibTeX-Style>.
- [10] 沈周. 基于 caspervector 改写的符合 GB/T 7714—2005 标准的参考文献格式 [EB/OL]. 2016. <https://github.com/szsdk/biblatex-gbt77142005>.
- [11] VECTOR C T. biblatex 参考文献和引用样式: caspervector[M/OL]. 2012. <http://mirrors.ctan.org/macros/latex/contrib/biblatex-contrib/biblatex-caspervector/doc/caspervector.pdf>.
- [12] 胡振震. 符合 GB/T 7714—2015 标准的 biblatex 参考文献样式 [M/OL]. 2016. <http://mirrors.ctan.org/macros/latex/contrib/biblatex-contrib/biblatex-gb7714-2015/biblatex-gb7714-2015.pdf>.

A 宏包的代码实现

兼容过时的接口

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

将选项传递给 natbib

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

调用宏包，注意只需要 `compress` 不需要 `sort`。

```
38 \RequirePackage{natbib}
```

```
39 \RequirePackage{url}
```

如果将 `compress` 传给 `natbib` 容易导致 `option clash`。这里直接修改内部命令。

```
40 \def\NAT@cmprs{\@ne}
```

`\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}
```

(End definition for \citestyle. This function is documented on page 2.)

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 `etoolbox` 的 `\patchcmd`。

`Super` 样式的 `\citep` 的页码也为上标。另外加上 `\kern\p@` 去掉上标式引用后与中文之间多余的空格，参考 [tuna/thuthesis#624](#)。

```
50 \renewcommand\NAT@citesuper[3]{%
```

```
51 \ifNAT@swa
```

```
52 \if*#2*\else
```

```
53 #2\NAT@spacechar
```

```
54 \fi
```

```
55 % \unskip\kern\p@\textsuperscript{\NAT@open#1\NAT@close}%
```

```
56 % \if*#3*\else\NAT@spacechar#3\fi\else #1\fi\endgroup}
```

```
57 \unskip\kern\p@
```

```
58 \textsuperscript{%
```

```
59 \NAT@open
```

```
60 #1%
```

```
61 \NAT@close
```

```
62 \if*#3*\else
```

```
63 #3%
```

```
64 \fi
```

```
65 }%
```

```
66 \kern\p@
```

```

67 \else
68   #1%
69 \fi
70 \endgroup
71 }

```

将 numbers 样式的 \citep 的页码置于括号外。

```

72 \renewcommand\NAT@citenum[3]{%
73   \ifNAT@swa
74     \NAT@@open
75     \if*#2*\else
76       #2\NAT@spacechar
77     \fi
78     % #1\if*#3*\else\NAT@cmt#3\fi\NAT@@close\else#1\fi\endgroup}
79     #1\NAT@@close
80     \if*#3*\else
81       \textsuperscript{#3}%
82     \fi
83   \else
84     #1%
85   \fi
86 \endgroup
87 }

```

Numerical 模式的 \citet 的页码:

```

88 \def\NAT@citexnum[#1][#2]#3{%
89   \NAT@reset@parser
90   \NAT@sort@cites{#3}%
91   \NAT@reset@citea
92   \@cite{\def\NAT@num{-1}\let\NAT@last@yr\relax\let\NAT@nm\@empty
93     \@for\@citeb:=\NAT@cite@list\do
94       {\@safe@activestrue
95         \edef\@citeb{\expandafter\@firstofone\@citeb\@empty}%
96         \@safe@activesfalse
97         \ifundefined{b@\@citeb\@extra@b@citeb}{%
98           {\reset@font\bfseries?}
99           \NAT@citeundefined\PackageWarning{natbib}%
100             {Citation ``\@citeb' on page \thepage \space undefined}}%
101         {\let\NAT@last@num\NAT@num\let\NAT@last@nm\NAT@nm
102           \NAT@parse{\@citeb}%
103           \ifNAT@longnames\ifundefined{bv@\@citeb\@extra@b@citeb}{%
104             \let\NAT@name=\NAT@all@names
105             \global\@namedef{bv@\@citeb\@extra@b@citeb}{}}{}}}%

```

```

106 \fi
107 \ifNAT@full\let\NAT@nm\NAT@all@names\else
108 \let\NAT@nm\NAT@name\fi
109 \ifNAT@swa
110 \@ifnum{\NAT@ctype>\@ne}{%
111 \citea
112 \NAT@hyper@{\@ifnum{\NAT@ctype=\tw@}{\NAT@test{\NAT@ctype}}{\NAT@alias}}%
113 }{%
114 \@ifnum{\NAT@cmprs>\z@}{%
115 \NAT@ifcat@num\NAT@num
116 {\let\NAT@nm=\NAT@num}%
117 {\def\NAT@nm{-2}}%
118 \NAT@ifcat@num\NAT@last@num
119 {\@tempcnta=\NAT@last@num\relax}%
120 {\@tempcnta\m@ne}%
121 \@ifnum{\NAT@nm=\@tempcnta}{%
122 \@ifnum{\NAT@merge>\@ne}{\NAT@last@yr@mbbox}%
123 }{%
124 \advance\@tempcnta by\@ne
125 \@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。

```

126 % \ifx\NAT@last@yr\relax
127 % \def@NAT@last@yr{\@citea}%
128 % \else
129 % \def@NAT@last@yr{--\NAT@penalty}%
130 % \fi
131 \def@NAT@last@yr{-\NAT@penalty}%
132 }{%
133 \NAT@last@yr@mbbox
134 }%
135 }%
136 }{%
137 \@tempswatrue
138 \@ifnum{\NAT@merge>\@ne}{\@ifnum{\NAT@last@num=\NAT@num\relax}{\@tempswafalse}{}}{%
139 \if@tempswa\NAT@citea@mbbox\fi
140 }%
141 }%
142 \NAT@def@citea
143 \else
144 \ifcase\NAT@ctype
145 \ifx\NAT@last@nm\NAT@nm \NAT@yrsep\NAT@penalty\NAT@space\else

```

```

146      \@citea \NAT@test{\@ne}\NAT@spacechar\NAT@mbox{\NAT@super@kern\NAT@@open}%
147      \fi
148      \if*#1*\else#1\NAT@spacechar\fi
149      \NAT@mbox{\NAT@hyper@{\@citenumfont{\NAT@num}}}%
150      \NAT@def@citea@box
151      \or
152      \NAT@hyper@citea@space{\NAT@test{\NAT@ctype}}%
153      \or
154      \NAT@hyper@citea@space{\NAT@test{\NAT@ctype}}%
155      \or
156      \NAT@hyper@citea@space\NAT@alias
157      \fi
158    \fi
159  }%
160}%
161  \@ifnum{\NAT@cmprs>\z@}{\NAT@last@yr}{}%
162  \ifNAT@swa\else

```

将页码放在括号外边，并且置于上标。

```

163    % \@ifnum{\NAT@ctype=\z@}{%
164    %   \if*#2*\else\NAT@cmt#2\fi
165    % }{%
166    \NAT@mbox{\NAT@@close}%
167    \@ifnum{\NAT@ctype=\z@}{%
168      \if*#2*\else
169        \textsuperscript{#2}%
170      \fi
171    }{%
172      \NAT@super@kern
173    \fi
174    }{#1}{#2}%
175  }%

```

Author-year 模式的 \citep 的页码：

```

176 \renewcommand\NAT@cite%
177   [3]{\ifNAT@swa\NAT@@open\if*#2*\else#2\NAT@spacechar\fi
178     #1\NAT@@close\if*#3*\else\textsuperscript{#3}\fi\else#1\fi\endgroup}

```

(End definition for \cite. This function is documented on page ??.)

Author-year 模式的 \citete 的页码：

```

179 \def\NAT@citex%
180   [#1][#2]#3{%
181   \NAT@reset@parser
182   \NAT@sort@cites{#3}%

```

```

183 \NAT@reset@citea
184 \@cite{\let\NAT@nm\@empty\let\NAT@year\@empty
185 \@for\@citeb:=\NAT@cite@list\do
186 {\@safe@activestrue
187 \edef\@citeb{\expandafter\@firstofone\@citeb\@empty}%
188 \@safe@activesfalse
189 \@ifundefined{b@\@citeb\@extra@b@citeb}{\@citea%
190 {\reset@font\bfseries ?}\NAT@citeundefined
191 \PackageWarning{natbib}%
192 {Citation '\@citeb' on page \thepage \space undefined}\def\NAT@date{}}%
193 {\let\NAT@lastnm=\NAT@nm\let\NAT@lastyr=\NAT@year
194 \NAT@parse{\@citeb}%
195 \ifNAT@longnames\ifundefined{bv@\@citeb\@extra@b@citeb}{%
196 \let\NAT@name=\NAT@all@names
197 \global\@namedef{bv@\@citeb\@extra@b@citeb}{}}}%
198 \fi
199 \ifNAT@full\let\NAT@nm\NAT@all@names\else
200 \let\NAT@nm\NAT@name\fi
201 \ifNAT@swa\ifcase\NAT@ctype
202 \if\relax\NAT@date\relax
203 \@citea\NAT@hyper@{\NAT@nmfmt{\NAT@nm}\NAT@date}%
204 \else
205 \ifx\NAT@lastnm\NAT@nm\NAT@yrsep
206 \ifx\NAT@lastyr\NAT@year
207 \def\NAT@temp{?}%
208 \ifx\NAT@temp\NAT@exlab\PackageWarningNoLine{natbib}%
209 {Multiple citation on page \thepage: same authors and
210 year\MessageBreak without distinguishing extra
211 letter,\MessageBreak appears as question mark}\fi
212 \NAT@hyper@{\NAT@exlab}%
213 \else\unskip\NAT@spacechar
214 \NAT@hyper@{\NAT@date}%
215 \fi
216 \else
217 \@citea\NAT@hyper@{%
218 \NAT@nmfmt{\NAT@nm}%
219 \hyper@natlinkbreak{%
220 \NAT@aysep\NAT@spacechar}{\@citeb\@extra@b@citeb
221 }%
222 \NAT@date
223 }%
224 \fi

```

```

225     \fi
226     \or\@citea\NAT@hyper@{\NAT@nmfmt{\NAT@nm}}%
227     \or\@citea\NAT@hyper@{\NAT@date}%
228     \or\@citea\NAT@hyper@{\NAT@alias}%
229     \fi \NAT@def@citea
230   \else
231     \ifcase\NAT@ctype
232     \if\relax\NAT@date\relax
233       \@citea\NAT@hyper@{\NAT@nmfmt{\NAT@nm}}%
234     \else
235       \ifx\NAT@last@nm\NAT@nm\NAT@yrsep
236         \ifx\NAT@last@yr\NAT@year
237           \def\NAT@temp{{?}}%
238           \ifx\NAT@temp\NAT@exlab\PackageWarningNoLine{natbib}%
239             {Multiple citation on page \thepage: same authors and
240             year\MessageBreak without distinguishing extra
241             letter,\MessageBreak appears as question mark}\fi
242           \NAT@hyper@{\NAT@exlab}%
243         \else
244           \unskip\NAT@spacechar
245           \NAT@hyper@{\NAT@date}%
246         \fi
247       \else
248         \@citea\NAT@hyper@{%
249           \NAT@nmfmt{\NAT@nm}%
250           \hyper@natlinkbreak{\NAT@spacechar\NAT@@open\if*#1*\else#1\NAT@spacechar\fi}%
251           {\@citeb\@extra@b@citeb}%
252           \NAT@date
253         }%
254       \fi
255     \fi
256     \or\@citea\NAT@hyper@{\NAT@nmfmt{\NAT@nm}}%
257     \or\@citea\NAT@hyper@{\NAT@date}%
258     \or\@citea\NAT@hyper@{\NAT@alias}%
259     \fi
260     \if\relax\NAT@date\relax
261       \NAT@def@citea
262     \else
263       \NAT@def@citea@close
264     \fi
265   \fi
266 }}\ifNAT@swa\else

```

将页码放在括号外边，并且置于上标。

```
267 % \if*#2*\else\NAT@cmt#2\fi
268 \if\relax\NAT@date\relax\else\NAT@close\fi
269 \if*#2*\else\textsuperscript{#2}\fi
270 \fi}{#1}{#2}}
```

thebibliography (env.) 参考文献列表的标签左对齐

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

\url 使用 xurl 宏包的方法，增加 URL 可断行的位置。

```
272 \g@addto@macro\UrlBreaks{%
273 \do0\do1\do2\do3\do4\do5\do6\do7\do8\do9%
274 \doA\doB\doC\doD\doE\doF\doG\doH\doI\doJ\doK\doL\doM
275 \doN\doO\doP\doQ\doR\doS\doT\doU\doV\doW\doX\doY\doZ
276 \doa\dob\doc\do d\doe\do f\dog\do h\doi\do j\do k\do l\do m
277 \do n\do o\do p\do q\do r\do s\do t\do u\do v\do w\do x\do y\do z
278 }
279 \Urlmuskip=0mu plus 0.1mu
```

(End definition for \url. This function is documented on page ??.)

兼容 v2.0 前过时的接口：

```
280 \newif\ifgbt@bib@style@written
281 \@ifpackageloaded{chapterbib}{}%
282 \def\bibliography#1{%
283 \ifgbt@bib@style@written\else
284 \bibliographystyle{gbt7714-numerical}%
285 \fi
286 \if@filesw
287 \immediate\write\@auxout{\string\bibdata{\zap@space#1 \@empty}}%
288 \fi
289 \@input@{\jobname.bbl}}
290 \def\bibliographystyle#1{%
291 \gbt@bib@style@writtenttrue
292 \ifx\@begindocumenthook\@undefined\else
293 \expandafter\AtBeginDocument
294 \fi
295 {\if@filesw
296 \immediate\write\@auxout{\string\bibstyle{#1}}%
297 \fi}%
298 }%
299 }
300 \ifgbt@legacy@interface
301 \ifgbt@numerical
```

```

302     \ifgbt@super\else
303         \citestyle{numbers}
304     \fi
305     \bibliographystyle{gbt7714-numerical}
306 \else
307     \bibliographystyle{gbt7714-author-year}
308 \fi
309 \fi
310 \end{package}

```

B BibTeX 样式的代码实现

B.1 自定义选项

`bst (env.)` 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。

```

311 (*author-year | numerical)
312 INTEGERS {
313     citation.et.al.min
314     citation.et.al.use.first
315     bibliography.et.al.min
316     bibliography.et.al.use.first
317     uppercase.name
318     terms.in.macro
319     year.after.author
320     period.after.author
321     italic.book.title
322     sentence.case.title
323     link.title
324     title.in.journal
325     show.patent.country
326     show.mark
327     space.before.mark
328     show.medium.type
329     short.journal
330     italic.journal
331     bold.journal.volume
332     show.missing.address.publisher
333     space.before.pages
334     only.start.page
335     wave.dash.in.pages
336     show.urldate
337     show.url
338     show.doi
339     show.preprint
340     show.note
341     show.english.translation
342     end.with.period
343 (*author-year)

```

```

344 lang.zh.order
345 lang.ja.order
346 lang.en.order
347 lang.ru.order
348 lang.other.order
349 </author-year>
350 }
351
352 STRINGS {
353   component.part.label
354 }
355

```

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。

```

356 FUNCTION {load.config}
357 {

```

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”。

```

358 <!*lucas>
359   #2 'citation.et.al.min :=
360   #1 'citation.et.al.use.first :=
361 </!lucas>
362 <*ucas>
363   #3 'citation.et.al.min :=
364   #1 'citation.et.al.use.first :=
365 </ucas>
366   #4 'bibliography.et.al.min :=
367   #3 'bibliography.et.al.use.first :=

```

英文姓名转为全大写：

```

368 <*(no-uppercase | thu)>
369   #1 'uppercase.name :=
370 </!(no-uppercase | thu)>
371 <*no-uppercase | thu>
372   #0 'uppercase.name :=
373 </no-uppercase | thu>

```

使用 TeX 宏输出“和”、“等”

```

374 <*(macro |ucas)>
375   #0 'terms.in.macro :=
376 </!(macro |ucas)>
377 <*macro |ucas>
378   #1 'terms.in.macro :=
379 </macro |ucas>

```

将年份置于著者后面（著者-出版年制默认）

```

380 <*numerical |ucas>
381   #0 'year.after.author :=
382 </numerical |ucas>
383 <*author-year&!ucas>
384   #1 'year.after.author :=
385 </author-year&!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：

```
386 <*numerical>
387   #1 'period.after.author :=
388 </numerical>
389 <*author-year>
390 <*2015&!(period | ustc)>
391   #0 'period.after.author :=
392 </2015&!(period | ustc)>
393 <*period | 2005 | ustc>
394   #1 'period.after.author :=
395 </period | 2005 | ustc>
396 </author-year>
```

书名使用斜体：

```
397 <*!italic-book-title>
398   #0 'italic.book.title :=
399 </!italic-book-title>
400 <*italic-book-title>
401   #1 'italic.book.title :=
402 </italic-book-title>
```

英文标题转为 sentence case（句首字母大写，其余小写）：

```
403 <*!no-sentence-case>
404   #1 'sentence.case.title :=
405 </!no-sentence-case>
406 <*no-sentence-case>
407   #0 'sentence.case.title :=
408 </no-sentence-case>
```

在标题添加超链接：

```
409 <*!link-title>
410   #0 'link.title :=
411 </!link-title>
412 <*link-title>
413   #1 'link.title :=
414 </link-title>
```

期刊是否含标题：

```
415 <*!no-title-in-journal>
416   #1 'title.in.journal :=
417 </!no-title-in-journal>
418 <*no-title-in-journal>
419   #0 'title.in.journal :=
420 </no-title-in-journal>
```

专利题名是否含专利国别

```
421 <*!(show-patent-country | 2005 | ustc | thu)>
422   #0 'show.patent.country :=
423 </!(show-patent-country | 2005 | ustc | thu)>
424 <*(show-patent-country | 2005 | ustc | thu)>
425   #1 'show.patent.country :=
426 </(show-patent-country | 2005 | ustc | thu)>
```

著录文献类型标识（比如“[M/OL]”）：

```
427 <!*no-mark>
428   #1 'show.mark :=
429 </!no-mark>
430 <*no-mark>
431   #0 'show.mark :=
432 </no-mark>
```

文献类型标识前是否有空格：

```
433 <!*space-before-mark>
434   #0 'space.before.mark :=
435 </!space-before-mark>
436 <*space-before-mark>
437   #1 'space.before.mark :=
438 </space-before-mark>
```

是否显示载体类型标识（比如“/OL”）：

```
439 <!*no-medium-type>
440   #1 'show.medium.type :=
441 </!no-medium-type>
442 <*no-medium-type>
443   #0 'show.medium.type :=
444 </no-medium-type>
```

使用“//”表示析出文献

```
445 <*(in-collection | no-slash)>
446   "slash" 'component.part.label :=
447 </!(in-collection | no-slash)>
448 <*in-collection>
449   "in" 'component.part.label :=
450 </in-collection>
451 <*no-slash>
452   "none" 'component.part.label :=
453 </no-slash>
```

期刊名使用缩写：

```
454 <!*short-journal>
455   #0 'short.journal :=
456 </!short-journal>
457 <*short-journal>
458   #1 'short.journal :=
459 </short-journal>
```

期刊名使用斜体：

```
460 <!*italic-journal>
461   #0 'italic.journal :=
462 </!italic-journal>
463 <*italic-journal>
464   #1 'italic.journal :=
465 </italic-journal>
```

期刊的卷使用粗体：

```
466   #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.”：

```
467 <!*sl-sn>
468   #0 'show.missing.address.publisher :=
469 </!sl-sn>
470 <*sl-sn>
471   #1 'show.missing.address.publisher :=
472 </sl-sn>
```

页码与前面的冒号之间是否有空格：

```
473 <!*no-space-before-pages>
474   #1 'space.before.pages :=
475 </!no-space-before-pages>
476 <*no-space-before-pages>
477   #0 'space.before.pages :=
478 </no-space-before-pages>
```

页码是否只含起始页：

```
479 <!*only-start-page>
480   #0 'only.start.page :=
481 </!only-start-page>
482 <*only-start-page>
483   #1 'only.start.page :=
484 </only-start-page>
```

起止页码使用波浪号：

```
485 <!*wave-dash-in-pages>
486   #0 'wave.dash.in.pages :=
487 </!wave-dash-in-pages>
488 <*wave-dash-in-pages>
489   #1 'wave.dash.in.pages :=
490 </wave-dash-in-pages>
```

是否著录非电子文献的引用日期：

```
491 <!*no-urldate>
492   #1 'show.urldate :=
493 </!no-urldate>
494 <*no-urldate>
495   #0 'show.urldate :=
496 </no-urldate>
```

是否著录 URL：

```
497 <!*no-url>
498   #1 'show.url :=
499 </!no-url>
500 <*no-url>
501   #0 'show.url :=
502 </no-url>
```

是否著录 DOI：

```
503 <*(no-doi | 2005)>
504   #1 'show.doi :=
505 </!(no-doi | 2005)>
506 <*no-doi | 2005>
507   #0 'show.doi :=
508 </no-doi | 2005>
```

是否著录 e-print:

```
509 <!*preprint>
510 #1 'show.preprint :=
511 </!preprint>
512 <*preprint>
513 #0 'show.preprint :=
514 </preprint>
```

在每一条文献最后输出注释 (note) 的内容:

```
515 #0 'show.note :=
```

中文文献是否显示英文翻译

```
516 <!*show-english-translation>
517 #0 'show.english.translation :=
518 </!show-english-translation>
519 <*show-english-translation>
520 #1 'show.english.translation :=
521 </show-english-translation>
```

结尾加句点

```
522 <!*no-period-at-end>
523 #1 'end.with.period :=
524 </!no-period-at-end>
525 <*no-period-at-end>
526 #0 'end.with.period :=
527 </no-period-at-end>
```

参考文献表按照“著者-出版年”组织时, 各个文种的顺序:

```
528 <*author-year>
529 #1 'lang.zh.order :=
530 #2 'lang.ja.order :=
531 #3 'lang.en.order :=
532 #4 'lang.ru.order :=
533 #5 'lang.other.order :=
534 </author-year>
535 }
536
```

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.

```
537 ENTRY
538 { address
539   archivePrefix
540   author
541   booktitle
542   date
543   doi
544   edition
```

```

545     editor
546     eprint
547     eprinttype
548     entrysubtype
549     howpublished
550     institution
551     journal
552     journaltitle
553     key
554     langid
555     language
556     location
557     mark
558     medium
559     note
560     number
561     organization
562     pages
563     publisher
564     school
565     series
566     shortjournal
567     title
568     translation
569     translator
570     url
571     urldate
572     volume
573     year
574 }
575 { 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.

```

576 { label extra.label sort.label short.label short.list entry.mark entry.url }
577

```

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

```

```

578 INTEGERS { output.state before.all mid.sentence after.sentence after.block after.slash }
579
580 INTEGERS { lang.zh lang.ja lang.en lang.ru lang.other }
581
582 INTEGERS { charptr len }
583
584 FUNCTION {init.state.consts}
585 { #0 'before.all :=
586   #1 'mid.sentence :=
587   #2 'after.sentence :=
588   #3 'after.block :=
589   #4 'after.slash :=
590   #3 'lang.zh :=
591   #4 'lang.ja :=
592   #1 'lang.en :=
593   #2 'lang.ru :=
594   #0 'lang.other :=
595 }
596

```

下面是一些常量的定义

```

597 FUNCTION {bbl.anonymous}
598 { entry.lang lang.zh =
599   { "佚名" }
600   { "Anon" }
601   if$
602 }
603
604 FUNCTION {bbl.space}
605 { entry.lang lang.zh =
606   { "\ " }
607   { " " }
608   if$
609 }
610
611 FUNCTION {bbl.and}
612 { "" }
613
614 FUNCTION {bbl.et.al}

```

```

615 { entry.lang lang.zh =
616     { " 等" }
617     { entry.lang lang.ja =
618         { " 他" }
619         { entry.lang lang.ru =
620             { "идр" }
621             { "et~al." }
622             if$
623         }
624         if$
625     }
626     if$
627 }
628
629 FUNCTION {citation.and}
630 { terms.in.macro
631     { "{\biband}" }
632     'bbl.and
633     if$
634 }
635
636 FUNCTION {citation.et.al}
637 { terms.in.macro
638     { "{\bibetal}" }
639     'bbl.et.al
640     if$
641 }
642
643 FUNCTION {bbl.colon} { ": " }
644
645 FUNCTION {bbl.pages.colon}
646 { space.before.pages
647     { ": " }
648     { ":\allowbreak " }
649     if$
650 }
651
652 <!*2005>
653 FUNCTION {bbl.wide.space} { "\quad " }
654 </!*2005>
655 <*2005>
656 FUNCTION {bbl.wide.space} { "\ " }
657 </2005>
658
659 FUNCTION {bbl.slash} { "//\allowbreak " }
660
661 FUNCTION {bbl.sine.loco}
662 { entry.lang lang.zh =
663     { "[出版地不详]" }
664     { "[S.l.]" }
665     if$
666 }
667
668 FUNCTION {bbl.sine.nomine}
669 { entry.lang lang.zh =

```

```

670     { "[出版者不详]" }
671     { "[s.n.]" }
672     if$
673 }
674
675 FUNCTION {bbl.sine.loco.sine.nomine}
676 { entry.lang lang.zh =
677   { "[出版地不详: 出版者不详]" }
678   { "[S.l.: s.n.]" }
679   if$
680 }
681

```

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

```

682 FUNCTION {not}
683 {   { #0 }
684     { #1 }
685     if$
686 }
687
688 FUNCTION {and}
689 {   'skip$
690     { pop$ #0 }
691     if$
692 }
693
694 FUNCTION {or}
695 {   { pop$ #1 }
696     'skip$
697     if$
698 }
699
700 STRINGS { x y }
701
702 FUNCTION {contains}
703 { 'y :=
704   'x :=
705   y text.length$ 'len :=
706   x text.length$ len - #1 + 'charptr :=
707   { charptr #0 >
708     x charptr len substring$ y = not
709     and
710   }
711   { charptr #1 - 'charptr := }
712   while$
713   charptr #0 >
714 }
715

```

the variables s and t are temporary string holders

```

716 STRINGS { s t }
717

```

```

718 FUNCTION {output.nonnull}
719 { 's :=
720   output.state mid.sentence =
721   { ", " * write$ }
722   { output.state after.block =
723     { add.period$ write$
724       newline$
725       "\newblock " write$
726     }
727     { output.state before.all =
728       'write$
729       { output.state after.slash =
730         { bbl.slash * write$
731           newline$
732         }
733         { add.period$ " " * write$ }
734       }
735     }
736   }
737   if$
738   mid.sentence 'output.state :=
739 }
740 if$
741 s
742 }
743
744
745 FUNCTION {output}
746 { duplicate$ empty$
747   'pop$
748   'output.nonnull
749   if$
750 }
751
752 FUNCTION {output.after}
753 { 't :=
754   duplicate$ empty$
755   'pop$
756   { 's :=
757     output.state mid.sentence =
758     { t * write$ }
759     { output.state after.block =
760       { add.period$ write$
761         newline$
762         "\newblock " write$
763       }
764       { output.state before.all =
765         'write$
766         { output.state after.slash =
767           { bbl.slash * write$ }
768           { add.period$ " " * write$ }
769         }
770       }
771     }
772     if$
773   }

```

```

773         if$
774         mid.sentence 'output.state :=
775     }
776     if$
777     s
778 }
779 if$
780 }
781
782 FUNCTION {output.check}
783 { 't :=
784   duplicate$ empty$
785   { pop$ "empty " t * " in " * cite$ * warning$ }
786   'output.nonnull
787   if$
788 }
789

```

This function finishes all entries.

```

790 FUNCTION {fin.entry}
791 { end.with.period
792   'add.period$
793   'skip$
794   if$
795   write$
796   show.english.translation entry.lang lang.zh = and
797   { ")"
798     write$
799   }
800   'skip$
801   if$
802   newline$
803 }
804
805 FUNCTION {new.block}
806 { output.state before.all =
807   'skip$
808   { output.state after.slash =
809     'skip$
810     { after.block 'output.state := }
811     if$
812   }
813   if$
814 }
815
816 FUNCTION {new.sentence}
817 { output.state after.block =
818   'skip$
819   { output.state before.all =
820     'skip$
821     { output.state after.slash =
822       'skip$
823       { after.sentence 'output.state := }
824       if$
825     }

```

```

826         if$
827     }
828     if$
829 }
830
831 FUNCTION {new.slash}
832 { output.state before.all =
833   'skip$
834   { component.part.label "slash" =
835     { after.slash 'output.state := }
836     { new.block
837       component.part.label "in" =
838         { entry.lang lang.en =
839           { "In: " output
840             writes$
841             ""
842             before.all 'output.state :=
843             }
844             'skip$
845             if$
846           }
847           'skip$
848         if$
849       }
850     if$
851   }
852   if$
853 }
854

```

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.

```

855 FUNCTION {new.block.checka}
856 { empty$
857   'skip$
858   'new.block
859   if$
860 }
861
862 FUNCTION {new.block.checkb}
863 { empty$
864   swap$ empty$
865   and
866   'skip$
867   'new.block
868   if$
869 }
870

```

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

```

871 FUNCTION {new.sentence.checka}
872 { empty$
873   'skip$

```

```

874     'new.sentence
875   if$
876 }
877
878 FUNCTION {new.sentence.checkb}
879 { empty$
880   swap$ empty$
881   and
882   'skip$
883   'new.sentence
884   if$
885 }
886

```

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

```

887 FUNCTION {field.or.null}
888 { duplicate$ empty$
889   { pop$ "" }
890   'skip$
891   if$

```

```

892 }
893
894 FUNCTION {emphasize}
895 { duplicate$ empty$
896   { pop$ "" }
897   { "\emph{" swap$ * "}" * }
898   if$
899 }
900
901 FUNCTION {format.btitle}
902 { italic.book.title
903   entry.lang lang.en = and
904     'emphasize
905     'skip$
906   if$
907 }
908

```

B.4.1 Detect Language

```

909 INTEGERS { byte second.byte }
910
911 INTEGERS { char.lang tmp.lang }
912
913 STRINGS { tmp.str }
914
915 FUNCTION {get.str.lang}
916 { 'tmp.str :=
917   lang.other 'tmp.lang :=
918   #1 'charptr :=
919   tmp.str text.length$ #1 + 'len :=
920   { charptr len < }
921   { tmp.str charptr #1 substring$ chr.to.int$ 'byte :=
922     byte #128 <
923     { charptr #1 + 'charptr :=
924       byte #64 > byte #91 < and byte #96 > byte #123 < and or
925       { lang.en 'char.lang := }
926       { lang.other 'char.lang := }
927     if$
928   }
929   { tmp.str charptr #1 + #1 substring$ chr.to.int$ 'second.byte :=
930     byte #224 <

```

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

```

931   { charptr #2 + 'charptr :=
932     byte #207 > byte #212 < and
933     byte #212 = second.byte #176 < and or
934     { lang.ru 'char.lang := }
935     { lang.other 'char.lang := }
936   if$
937 }
938 { byte #240 <

```

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

```

939   { charptr #3 + 'charptr :=

```

```

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

```

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

```

942         { byte #227 =
943         { second.byte #143 >
944         { lang.zh 'char.lang := }

```

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

```

945         { second.byte #128 > second.byte #132 < and
946         { lang.ja 'char.lang := }
947         { lang.other 'char.lang := }
948         if$
949         }
950         if$
951         }

```

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

```

952         { byte #239 =
953         second.byte #163 > second.byte #172 < and and
954         { lang.zh 'char.lang := }
955         { lang.other 'char.lang := }
956         if$
957         }
958         if$
959         }
960         if$
961         }

```

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.

```

962         { charptr #4 + 'charptr :=
963         byte #240 = second.byte #159 > and
964         { lang.zh 'char.lang := }
965         { lang.other 'char.lang := }
966         if$
967         }
968         if$
969         }
970         if$
971         }
972         if$
973         char.lang tmp.lang >
974         { char.lang 'tmp.lang := }
975         'skip$
976         if$
977         }
978         while$
979         tmp.lang
980     }
981
982 FUNCTION {check.entry.lang}
983 { author field.or.null
984   title field.or.null *

```

```

985   get.str.lang
986 }
987
988 STRINGS { entry.langid }
989
990 FUNCTION {set.entry.lang}
991 { "" 'entry.langid :=
992   language empty$ not
993   { language 'entry.langid := }
994   'skip$
995   if$
996   langid empty$ not
997   { langid 'entry.langid := }
998   'skip$
999   if$
1000  entry.langid empty$
1001  { check.entry.lang }
1002  { entry.langid "english" = entry.langid "american" = or entry.langid "british" = or
1003    { lang.en }
1004    { entry.langid "chinese" =
1005      { lang.zh }
1006      { entry.langid "japanese" =
1007        { lang.ja }
1008        { entry.langid "russian" =
1009          { lang.ru }
1010          { check.entry.lang }
1011          if$
1012        }
1013        if$
1014      }
1015      if$
1016    }
1017    if$
1018  }
1019  if$
1020  'entry.lang :=
1021 }
1022
1023 FUNCTION {set.entry.numbered}
1024 { type$ "patent" =
1025   type$ "standard" = or
1026   type$ "techreport" = or
1027   { #1 'entry.numbered := }
1028   { #0 'entry.numbered := }
1029   if$
1030 }
1031

```

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

```

```

        fi
    fi
END

```

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

```

1032 INTEGERS { nameptr namesleft numnames name.lang }
1033
1034 FUNCTION {format.name}
1035 { "{vv~}{ll}{, jj}{, ff}" format.name$ 't :=
1036   t "others" =
1037     { bbl.et.al }
1038     { t get.str.lang 'name.lang :=
1039       name.lang lang.en =
1040         { t #1 "{vv~}{ll}{ f{~}}" format.name$
1041           uppercase.name
1042             { "u" change.case$ }
1043             'skip$
1044             if$
1045             t #1 "{, jj}" format.name$ *
1046           }
1047         { t #1 "{ll}{ff}" format.name$ }
1048       }
1049     if$
1050   }
1051 }
1052
1053 FUNCTION {format.names}
1054 { 's :=
1055   #1 'nameptr :=
1056   s num.names$ 'numnames :=
1057   ""
1058   numnames 'namesleft :=
1059   { namesleft #0 > }
1060   { s nameptr format.name bbl.et.al =
1061     numnames bibliography.et.al.min #1 - > nameptr bibliography.et.al.use.first > and or
1062     { ", " *
1063       bbl.et.al *
1064       #1 'namesleft :=
1065     }
1066     { nameptr #1 >
1067       { namesleft #1 = bbl.and "" = not and
1068         { bbl.and * }
1069         { ", " * }
1070       }
1071       if$
1072       'skip$
1073       if$
1074       s nameptr format.name *
1075     }
1076     if$
1077     nameptr #1 + 'nameptr :=
1078     namesleft #1 - 'namesleft :=
1079   }
1080   while$

```

```

1081 }
1082
1083 FUNCTION {format.key}
1084 { empty$
1085   { key field.or.null }
1086   { "" }
1087   if$
1088 }
1089
1090 FUNCTION {format.authors}
1091 { author empty$ not
1092   { author format.names }
1093   { "empty author in " cite$ * warning$
1094   <*author-year>
1095     bbl.anonymous
1096   </author-year>
1097   <*numerical>
1098     ""
1099   </numerical>
1100   }
1101   if$
1102 }
1103
1104 FUNCTION {format.editors}
1105 { editor empty$
1106   { "" }
1107   { editor format.names }
1108   if$
1109 }
1110
1111 FUNCTION {format.translators}
1112 { translator empty$
1113   { "" }
1114   { translator format.names
1115     entry.lang lang.zh =
1116     { translator num.names$ #3 >
1117       { " 译" * }
1118       { ", 译" * }
1119       if$
1120     }
1121     'skip$
1122     if$
1123   }
1124   if$
1125 }
1126
1127 FUNCTION {format.full.names}
1128 {'s :=
1129   #1 'nameptr :=
1130   s num.names$ 'numnames :=
1131   numnames 'namesleft :=
1132   { namesleft #0 > }
1133   { s nameptr "{vv~}{ll}{, jj}{, ff}" format.names$ 't :=
1134     t get.str.lang 'name.lang :=
1135     name.lang lang.en =

```

```

1136     { t #1 "{vv~}{ll}" format.name$ 't := }
1137     { t #1 "{ll}{ff}" format.name$ 't := }
1138   if$
1139   nameptr #1 >
1140   {
1141     namesleft #1 >
1142     { ", " * t * }
1143     {
1144       numnames #2 >
1145       { ", " * }
1146       'skip$
1147       if$
1148       t "others" =
1149       { " et~al." * }
1150       { " and " * t * }
1151       if$
1152     }
1153     if$
1154   }
1155   't
1156   if$
1157   nameptr #1 + 'nameptr :=
1158   namesleft #1 - 'namesleft :=
1159 }
1160 while$
1161 }
1162
1163 FUNCTION {author.editor.full}
1164 { author empty$
1165   { editor empty$
1166     { "" }
1167     { editor format.full.names }
1168     if$
1169   }
1170   { author format.full.names }
1171   if$
1172 }
1173
1174 FUNCTION {author.full}
1175 { author empty$
1176   { "" }
1177   { author format.full.names }
1178   if$
1179 }
1180
1181 FUNCTION {editor.full}
1182 { editor empty$
1183   { "" }
1184   { editor format.full.names }
1185   if$
1186 }
1187
1188 FUNCTION {make.full.names}
1189 { type$ "book" =
1190   type$ "inbook" =

```

```

1191 or
1192 'author.editor.full
1193 { type$ "collection" =
1194   type$ "proceedings" =
1195   or
1196     'editor.full
1197     'author.full
1198   if$
1199 }
1200 if$
1201 }
1202
1203 FUNCTION {output.bibitem}
1204 { newline$
1205   "\bibitem[" write$
1206   label "]" *
1207   make.full.names duplicate$ short.list =
1208   { pop$ }
1209   { duplicate$ "]" contains
1210     { "{" swap$ * "}" * }
1211     'skip$
1212     if$
1213     *
1214   }
1215   if$
1216   "{" * write$
1217   cite$ write$
1218   "}" write$
1219   newline$
1220   ""
1221   before.all 'output.state :=
1222 }
1223

```

B.4.3 Format title

The `format.title` function is used for non-book-like titles. For most styles we convert to lowercase (except for the very first letter, and except for the first one after a colon (followed by whitespace)), and hope the user has brace-surrounded words that need to stay capitalized; for some styles, however, we leave it as it is in the database.

```

1224 FUNCTION {change.sentence.case}
1225 { entry.lang lang.en =
1226   { "t" change.case$ }
1227   'skip$
1228   if$
1229 }
1230
1231 FUNCTION {add.link}
1232 { url empty$ not
1233   { "\href{" url * "}" * swap$ * "}" * }
1234   { doi empty$ not
1235     { "\href{https://doi.org/" doi * "}" * swap$ * "}" * }

```

```

1236         'skip$
1237     if$
1238     }
1239     if$
1240 }
1241
1242 FUNCTION {format.title}
1243 { title empty$
1244   { "" }
1245   { title
1246     sentence.case.title
1247     'change.sentence.case
1248     'skip$
1249     if$
1250     entry.numbered number empty$ not and
1251     { bbl.colon *
1252       type$ "patent" = show.patent.country and
1253       { address empty$ not
1254         { address * ", " * }
1255         { location empty$ not
1256           { location * ", " * }
1257           { entry.lang lang.zh =
1258             { " 中国" * ", " * }
1259             'skip$
1260             if$
1261           }
1262         if$
1263       }
1264       if$
1265     }
1266     'skip$
1267     if$
1268     number *
1269   }
1270   'skip$
1271   if$
1272   link.title
1273   'add.link
1274   'skip$
1275   if$
1276 }
1277 if$
1278 }
1279

```

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

```

```

1280 FUNCTION {tie.or.space.connect}
1281 { duplicate$ text.length$ #3 <
1282   { "~" }
1283   { " " }
1284   if$
1285   swap$ * *
1286 }
1287

```

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

```

```

1288 FUNCTION {either.or.check}
1289 { empty$
1290   'pop$
1291   { "can't use both " swap$ * " fields in " * cite$ * warning$ }
1292   if$
1293 }
1294

```

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.

```

1295 FUNCTION {is.digit}
1296 { duplicate$ empty$
1297   { pop$ #0 }
1298   { chr.to.int$
1299     duplicate$ "0" chr.to.int$ <
1300     { pop$ #0 }
1301     { "9" chr.to.int$ >
1302       { #0 }
1303       { #1 }
1304     if$

```

```

1305     }
1306     if$
1307     }
1308     if$
1309 }
1310
1311 FUNCTION {is.number}
1312 { 's :=
1313   s empty$
1314   { #0 }
1315   { s text.length$ 'charptr :=
1316     { charptr #0 >
1317       s charptr #1 substring$ is.digit
1318       and
1319     }
1320     { charptr #1 - 'charptr := }
1321     while$
1322     charptr not
1323   }
1324   if$
1325 }
1326
1327 FUNCTION {format.volume}
1328 { volume empty$ not
1329   { volume is.number
1330     { entry.lang lang.zh =
1331       { " 第 " volume * " 卷" * }
1332       { "Vol." volume tie.or.space.connect }
1333     if$
1334   }
1335   { volume }
1336   if$
1337 }
1338 { "" }
1339 if$
1340 }
1341
1342 FUNCTION {format.number}
1343 { number empty$ not
1344   { number is.number
1345     { entry.lang lang.zh =
1346       { " 第 " number * " 册" * }
1347       { "No." number tie.or.space.connect }
1348     if$
1349   }
1350   { number }
1351   if$
1352 }
1353 { "" }
1354 if$
1355 }
1356
1357 FUNCTION {format.volume.number}
1358 { volume empty$ not
1359   { format.volume }

```

```

1360     { format.number }
1361   if$
1362 }
1363
1364 FUNCTION {format.title.vol.num}
1365 { title
1366   sentence.case.title
1367   'change.sentence.case
1368   'skip$
1369   if$
1370   entry.numbered
1371     { number empty$ not
1372       { bbl.colon * number * }
1373       'skip$
1374       if$
1375     }
1376     { format.volume.number 's :=
1377       s empty$ not
1378       { bbl.colon * s * }
1379       'skip$
1380       if$
1381     }
1382   if$
1383 }
1384
1385 FUNCTION {format.series.vol.num.title}
1386 { format.volume.number 's :=
1387   series empty$ not
1388   { series
1389     sentence.case.title
1390     'change.sentence.case
1391     'skip$
1392     if$
1393     entry.numbered
1394       { bbl.wide.space * }
1395       { bbl.colon *
1396         s empty$ not
1397         { s * bbl.wide.space * }
1398         'skip$
1399       if$
1400     }
1401     if$
1402     title *
1403     sentence.case.title
1404     'change.sentence.case
1405     'skip$
1406     if$
1407     entry.numbered number empty$ not and
1408     { bbl.colon * number * }
1409     'skip$
1410     if$
1411   }
1412   { format.title.vol.num }
1413   if$
1414   format.btitle

```

```

1415     link.title
1416     'add.link
1417     'skip$
1418     if$
1419 }
1420
1421 FUNCTION {format.booktitle.vol.num}
1422 { booktitle
1423   entry.numbered
1424   'skip$
1425   { format.volume.number 's :=
1426     s empty$ not
1427     { bbl.colon * s * }
1428     'skip$
1429     if$
1430   }
1431   if$
1432 }
1433
1434 FUNCTION {format.series.vol.num.booktitle}
1435 { format.volume.number 's :=
1436   series empty$ not
1437   { series bbl.colon *
1438     entry.numbered not s empty$ not and
1439     { s * bbl.wide.space * }
1440     'skip$
1441     if$
1442     booktitle *
1443   }
1444   { format.booktitle.vol.num }
1445   if$
1446   format.btitle
1447 }
1448
1449 FUNCTION {remove.period}
1450 { 't :=
1451   "" 's :=
1452   { t empty$ not }
1453   { t #1 #1 substring$ 'tmp.str :=
1454     tmp.str "." = not
1455     { s tmp.str * 's := }
1456     'skip$
1457     if$
1458     t #2 global.max$ substring$ 't :=
1459   }
1460   while$
1461   s
1462 }
1463
1464 FUNCTION {abbreviate}
1465 { remove.period
1466   't :=
1467   t "l" change.case$ 's :=
1468   ""
1469   s "physical review letters" =

```

```

1470     { "Phys Rev Lett" }
1471     'skip$
1472     if$
1473     's :=
1474     s empty$
1475     { t }
1476     { pop$ s }
1477     if$
1478 }
1479
1480 FUNCTION {get.journal.title}
1481 { short.journal
1482   { shortjournal empty$ not
1483     { shortjournal }
1484     { journal empty$ not
1485       { journal abbreviate }
1486       { journaltitle empty$ not
1487         { journaltitle abbreviate }
1488         { "" }
1489         if$
1490       }
1491       if$
1492     }
1493     if$
1494   }
1495   { journal empty$ not
1496     { journal }
1497     { journaltitle empty$ not
1498       { journaltitle }
1499       { shortjournal empty$ not
1500         { shortjournal }
1501         { "" }
1502         if$
1503       }
1504       if$
1505     }
1506     if$
1507   }
1508   if$
1509 }
1510
1511 FUNCTION {check.arxiv.preprint}
1512 { #1 #5 substring$ "l" change.case$ "arxiv" =
1513   { #1 }
1514   { #0 }
1515   if$
1516 }
1517
1518 FUNCTION {format.journal}
1519 { get.journal.title
1520   duplicate$ empty$ not
1521   { italic.journal entry.lang lang.en = and
1522     'emphasize
1523     'skip$
1524     if$

```

```

1525     }
1526     'skip$
1527   if$
1528 }
1529

```

B.4.4 Format entry type mark

```

1530 FUNCTION {set.entry.mark}
1531 { entry.mark empty$ not
1532   'pop$
1533   { mark empty$ not
1534     { pop$ mark 'entry.mark := }
1535     { 'entry.mark := }
1536   if$
1537   }
1538   if$
1539 }
1540
1541 FUNCTION {format.mark}
1542 { show.mark
1543   { entry.mark
1544     show.medium.type
1545     { medium empty$ not
1546       { "/" * medium * }
1547       { entry.is.electronic
1548         { "/OL" * }
1549         'skip$
1550       if$
1551     }
1552     if$
1553   }
1554   'skip$
1555   if$
1556   'entry.mark :=
1557   space.before.mark
1558   { " " }
1559   { "\allowbreak" }
1560   if$
1561   "[" * entry.mark * "]" *
1562   }
1563   { "" }
1564   if$
1565 }
1566

```

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.

```

1567 FUNCTION {num.to.ordinal}
1568 { duplicate$ text.length$ 'charptr :=
1569   duplicate$ charptr #1 substring$ 's :=
1570   s "1" =

```

```

1571     { "st" * }
1572     { s "2" =
1573         { "nd" * }
1574         { s "3" =
1575             { "rd" * }
1576             { "th" * }
1577         if$
1578         }
1579     if$
1580 }
1581 if$
1582 }
1583
1584 FUNCTION {format.edition}
1585 { edition empty$
1586   { "" }
1587   { edition is.number
1588     { edition "1" = not
1589       { entry.lang lang.zh =
1590         { edition " 版" * }
1591         { edition num.to.ordinal " ed." * }
1592       if$
1593       }
1594       'skip$
1595     if$
1596   }
1597   { entry.lang lang.en =
1598     { edition change.sentence.case 's :=
1599       s "Revised" = s "Revised edition" = or
1600       { "Rev. ed." }
1601       { s " ed." * }
1602     if$
1603   }
1604   { edition }
1605   if$
1606 }
1607 if$
1608 }
1609 if$
1610 }
1611

```

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。

```

1612 FUNCTION {format.publisher}
1613 { publisher empty$ not
1614   { publisher }
1615   { school empty$ not
1616     { school }
1617     { organization empty$ not
1618       { organization }
1619       { institution empty$ not
1620         { institution }

```

```

1621         { "" }
1622     if$
1623 }
1624 if$
1625 }
1626 if$
1627 }
1628 if$
1629 }
1630
1631 FUNCTION {format.address.publisher}
1632 { address empty$ not
1633   { address }
1634   { location empty$ not
1635     { location }
1636     { "" }
1637   if$
1638 }
1639 if$
1640 duplicates$ empty$ not
1641   { format.publisher empty$ not
1642     { bbl.colon * format.publisher * }
1643     { entry.is.electronic not show.missing.address.publisher and
1644       { bbl.colon * bbl.sine.nomine * }
1645       'skip$
1646     if$
1647   }
1648   if$
1649 }
1650 { pop$
1651   entry.is.electronic not show.missing.address.publisher and
1652   { format.publisher empty$ not
1653     { bbl.sine.loco bbl.colon * format.publisher * }
1654     { bbl.sine.loco.sine.nomine }
1655   if$
1656   }
1657   { format.publisher empty$ not
1658     { format.publisher }
1659     { "" }
1660   if$
1661   }
1662   if$
1663 }
1664 if$
1665 }
1666

```

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。

```
1667 FUNCTION {extract.before.dash}
1668 { duplicate$ empty$
1669   { pop$ "" }
1670   { 's :=
1671     #1 'charptr :=
1672     s text.length$ #1 + 'len :=
1673     { charptr len <
1674       s charptr #1 substring$ "-" = not
1675       and
1676     }
1677     { charptr #1 + 'charptr := }
1678     while$
1679     s #1 charptr #1 - substring$
1680   }
1681   if$
1682 }
1683
1684 FUNCTION {extract.after.dash}
1685 { duplicate$ empty$
1686   { pop$ "" }
1687   { 's :=
1688     #1 'charptr :=
1689     s text.length$ #1 + 'len :=
1690     { charptr len <
1691       s charptr #1 substring$ "-" = not
1692       and
1693     }
1694     { charptr #1 + 'charptr := }
1695     while$
1696     { charptr len <
1697       s charptr #1 substring$ "-" =
1698       and
1699     }
1700     { charptr #1 + 'charptr := }
1701     while$
1702     s charptr global.max$ substring$
1703   }
1704   if$
1705 }
1706
1707 FUNCTION {extract.before.slash}
1708 { duplicate$ empty$
1709   { pop$ "" }
1710   { 's :=
1711     #1 'charptr :=
1712     s text.length$ #1 + 'len :=
1713     { charptr len <
1714       s charptr #1 substring$ "/" = not
1715       and
1716     }
1717     { charptr #1 + 'charptr := }
1718     while$
1719     s #1 charptr #1 - substring$
1720   }
```

```

1721     if$
1722 }
1723
1724 FUNCTION {extract.after.slash}
1725 { duplicate$ empty$
1726   { pop$ "" }
1727   { 's :=
1728     #1 'charptr :=
1729     s text.length$ #1 + 'len :=
1730     { charptr len <
1731       s charptr #1 substring$ "-" = not
1732       and
1733       s charptr #1 substring$ "/" = not
1734       and
1735     }
1736     { charptr #1 + 'charptr := }
1737     while$
1738     { charptr len <
1739       s charptr #1 substring$ "-" =
1740       s charptr #1 substring$ "/" =
1741       or
1742       and
1743     }
1744     { charptr #1 + 'charptr := }
1745     while$
1746     s charptr global.max$ substring$
1747   }
1748   if$
1749 }
1750

```

著者-出版年制必须提取出年份

```

1751 FUNCTION {format.year}
1752 { year empty$ not
1753   { year extract.before.slash extra.label * }
1754   { date empty$ not
1755     { date extract.before.dash extra.label * }
1756     { entry.is.electronic not
1757       { "empty year in " cite$ * warning$ }
1758       'skip$
1759       if$
1760       urldate empty$ not
1761       { "[" urldate extract.before.dash * extra.label * "]" * }
1762       { "" }
1763       if$
1764     }
1765     if$
1766   }
1767   if$
1768 }
1769
1770 FUNCTION {format.periodical.year}
1771 { year empty$ not
1772   { year extract.before.slash
1773     "--" *

```

```

1774     year extract.after.slash
1775     duplicate$ empty$
1776     'pop$
1777     { * }
1778     if$
1779 }
1780 { date empty$ not
1781     { date extract.before.dash }
1782     { "empty year in " cite$ * warning$
1783       urldate empty$ not
1784         { "[" urldate extract.before.dash * "]" * }
1785         { "" }
1786       if$
1787     }
1788     if$
1789 }
1790 if$
1791 }
1792

```

专利和报纸都是使用日期而不是年

```

1793 FUNCTION {format.date}
1794 { date empty$ not
1795   { type$ "patent" = type$ "newspaper" = or
1796     { date }
1797     { entrysubtype empty$ not
1798       { type$ "article" = entrysubtype "newspaper" = and
1799         { date }
1800         { format.year }
1801       if$
1802     }
1803     { format.year }
1804   if$
1805 }
1806 if$
1807 }
1808 { year empty$ not
1809   { format.year }
1810   { "" }
1811 if$
1812 }
1813 if$
1814 }
1815

```

更新、修改日期只用于电子资源 electronic

```

1816 FUNCTION {format.editeddate}
1817 { date empty$ not
1818   { "\allowbreak(" date * ")" * }
1819   { "" }
1820 if$
1821 }
1822

```

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。

```

1823 FUNCTION {format.urldate}
1824 { show.urldate show.url and entry.url empty$ not and
1825   is.pure.electronic or
1826   urldate empty$ not and
1827   { "\allowbreak[" urldate * "]" * }
1828   { "" }
1829   if$
1830 }
1831

```

B.4.8 Format pages

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

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

```

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

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

```

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

```

1832 FUNCTION {hyphenate}

```

```

1833 { 't :=
1834     ""
1835     { t empty$ not }
1836     { t #1 #1 substring$ "-" =
1837       { wave.dash.in.pages
1838         { "~" * }
1839         { "-" * }
1840         if$
1841           { t #1 #1 substring$ "-" = }
1842           { t #2 global.max$ substring$ 't := }
1843         while$
1844       }
1845       { t #1 #1 substring$ *
1846         t #2 global.max$ substring$ 't :=
1847       }
1848     if$
1849   }
1850 while$
1851 }
1852

```

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

```

1853 FUNCTION {format.pages}
1854 { pages empty$
1855   { "" }
1856   { pages hyphenate }
1857   if$
1858 }
1859
1860 FUNCTION {format.extracted.pages}
1861 { pages empty$
1862   { "" }
1863   { pages
1864     only.start.page
1865     'extract.before.dash
1866     'hyphenate
1867     if$
1868   }
1869   if$
1870 }
1871

```

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。

```

1872 FUNCTION {format.journal.volume}
1873 { volume empty$ not
1874   { bold.journal.volume
1875     { "\textbf{" volume * "}" * }
1876     { volume }
1877     if$

```

```

1878     }
1879     { "" }
1880     if$
1881 }
1882
1883 FUNCTION {format.journal.number}
1884 { number empty$ not
1885   { "\allowbreak (" number * ")" * }
1886   { "" }
1887   if$
1888 }
1889
1890 FUNCTION {format.journal.pages}
1891 { pages empty$
1892   { "" }
1893   { format.extracted.pages }
1894   if$
1895 }
1896

```

连续出版物的年卷期有起止范围，需要特殊处理

```

1897 FUNCTION {format.periodical.year.volume.number}
1898 { year empty$ not
1899   { year extract.before.slash }
1900   { "empty year in periodical " cite$ * warning$ }
1901   if$
1902   volume empty$ not
1903     { ", " * volume extract.before.dash * }
1904     'skip$
1905   if$
1906   number empty$ not
1907     { "\allowbreak (" * number extract.before.dash * ")" * }
1908     'skip$
1909   if$
1910   "--" *
1911   year extract.after.slash empty$
1912   volume extract.after.dash empty$ and
1913   number extract.after.dash empty$ and not
1914     { year extract.after.slash empty$ not
1915       { year extract.after.slash * }
1916       { year extract.before.slash * }
1917       if$
1918       volume empty$ not
1919         { ", " * volume extract.after.dash * }
1920         'skip$
1921       if$
1922       number empty$ not
1923         { "\allowbreak (" * number extract.after.dash * ")" * }
1924         'skip$
1925       if$
1926     }
1927     'skip$
1928   if$
1929 }
1930

```

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。

```
1931 FUNCTION {check.url}
1932 { url empty$ not
1933   { "\url{" url * "}" * 'entry.url :=
1934     #1 'entry.is.electronic :=
1935   }
1936   { howpublished empty$ not
1937     { howpublished #1 #5 substring$ "\url{" =
1938       { howpublished 'entry.url :=
1939         #1 'entry.is.electronic :=
1940       }
1941       'skip$
1942     if$
1943   }
1944   { note empty$ not
1945     { note #1 #5 substring$ "\url{" =
1946       { note 'entry.url :=
1947         #1 'entry.is.electronic :=
1948       }
1949       'skip$
1950     if$
1951   }
1952   'skip$
1953   if$
1954 }
1955 if$
1956 }
1957 if$
1958 }
1959
1960 FUNCTION {output.url}
1961 { show.url is.pure.electronic or
1962   entry.url empty$ not and
1963   { new.block
1964     entry.url output
1965   }
1966   'skip$
1967   if$
1968 }
1969
```

需要检测 DOI 是否已经包含在 URL 中。

```
1970 FUNCTION {check.doi}
1971 { doi empty$ not
1972   { #1 'entry.is.electronic := }
1973   'skip$
1974   if$
1975 }
1976
1977 FUNCTION {is.in.url}
1978 { 's :=
1979   s empty$
```

```

1980     { #1 }
1981     { entry.url empty$
1982       { #0 }
1983       { s text.length$ 'len :=
1984         entry.url text.length$ 'charptr :=
1985         { entry.url charptr len substring$ s = not
1986           charptr #0 >
1987           and
1988           }
1989         { charptr #1 - 'charptr := }
1990       while$
1991       charptr
1992     }
1993   if$
1994 }
1995 if$
1996 }
1997
1998 FUNCTION {format.doi}
1999 { ""
2000   doi empty$ not
2001   { "" 's :=
2002     doi 't :=
2003     #0 'numnames :=
2004     { t empty$ not}
2005     { t #1 #1 substring$ 'tmp.str :=
2006       tmp.str "," = tmp.str " " = or t #2 #1 substring$ empty$ or
2007       { t #2 #1 substring$ empty$
2008         { s tmp.str * 's := }
2009         'skip$
2010       if$
2011       s empty$ s is.in.url or
2012       'skip$
2013       { numnames #1 + 'numnames :=
2014         numnames #1 >
2015         { ", " * }
2016         { "DOI: " * }
2017         if$
2018         "\doi{" s * "}" * *
2019       }
2020       if$
2021       "" 's :=
2022     }
2023     { s tmp.str * 's := }
2024     if$
2025     t #2 global.max$ substring$ 't :=
2026   }
2027   while$
2028 }
2029 'skip$
2030 if$
2031 }
2032
2033 FUNCTION {output.doi}
2034 { doi empty$ not show.doi and

```

```

2035     show.english.translation entry.lang lang.zh = and not and
2036     { new.block
2037       format.doi output
2038     }
2039     'skip$
2040     if$
2041 }
2042
2043 FUNCTION {check.electronic}
2044 { "" 'entry.url :=
2045   #0 'entry.is.electronic :=
2046   'check.doi
2047   'skip$
2048   if$
2049   'check.url
2050   'skip$
2051   if$
2052   medium empty$ not
2053   { medium "MT" = medium "DK" = or medium "CD" = or medium "OL" = or
2054     { #1 'entry.is.electronic := }
2055     'skip$
2056     if$
2057   }
2058   'skip$
2059   if$
2060 }
2061
2062 FUNCTION {format.eprint}
2063 { archivePrefix empty$ not
2064   { archivePrefix }
2065   { eprinttype empty$ not
2066     { archivePrefix }
2067     { "" }
2068     if$
2069   }
2070   if$
2071   's :=
2072   s empty$ not
2073   { s ": \eprint{" *
2074     url empty$ not
2075     { url }
2076     { "https://" s "l" change.case$ * ".org/abs/" * eprint * }
2077     if$
2078     * "}" *
2079     eprint * "}" *
2080   }
2081   { eprint }
2082   if$
2083 }
2084
2085 FUNCTION {output.eprint}
2086 { show.preprint eprint empty$ not and
2087   { new.block
2088     format.eprint output
2089   }

```

```

2090     'skip$
2091   if$
2092 }
2093
2094 FUNCTION {format.note}
2095 { note empty$ not show.note and
2096   { note }
2097   { "" }
2098   if$
2099 }
2100
2101 FUNCTION {output.translation}
2102 { show.english.translation entry.lang lang.zh = and
2103   { translation empty$ not
2104     { translation }
2105     { "[English translation missing!]" }
2106     if$
2107     " (in Chinese)" * output
2108     write$
2109     format.doi duplicate$ empty$ not
2110     { newline$
2111       write$
2112     }
2113     'pop$
2114     if$
2115     " \\" writes$
2116     newline$
2117     "(" writes$
2118     ""
2119     before.all 'output.state :=
2120   }
2121   'skip$
2122   if$
2123 }
2124

```

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

```

2125 FUNCTION {empty.misc.check}
2126 { author empty$ title empty$
2127   year empty$
2128   and and
2129   key empty$ not and
2130   { "all relevant fields are empty in " cite$ * warning$ }
2131   'skip$
2132   if$
2133 }
2134

```

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

```

2135 FUNCTION {monograph}
2136 { output.bibitem
2137   output.translation
2138   author empty$ not
2139     { format.authors }
2140     { editor empty$ not
2141       { format.editors }
2142       { "empty author and editor in " cite$ * warning$
2143 (*author-year)
2144   bbl.anonymous
2145 (/author-year)
2146 (*numerical)
2147   ""
2148 (/numerical)
2149   }
2150   if$
2151   }
2152   if$
2153   output
2154   year.after.author
2155     { period.after.author
2156       'new.sentence
2157       'skip$
2158       if$
2159         format.year "year" output.check
2160       }
2161       'skip$
2162   if$
2163   new.block
2164   format.series.vol.num.title "title" output.check
2165   "M" set.entry.mark
2166   format.mark "" output.after
2167   new.block
2168   format.translators output
2169   new.sentence
2170   format.edition output
2171   new.block
2172   format.address.publisher output
2173   year.after.author not
2174     { format.year "year" output.check }
2175     'skip$
2176   if$
2177   format.pages bbl.pages.colon output.after
2178   format.urldate "" output.after
2179   output.url
2180   output.doi

```

```

2181 new.block
2182 format.note output
2183 fin.entry
2184 }
2185

```

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

```

2186 FUNCTION {incollection}
2187 { output.bibitem
2188   output.translation
2189   format.authors output
2190   author format.key output
2191   year.after.author
2192     { period.after.author
2193       'new.sentence
2194       'skip$
2195       if$
2196       format.year "year" output.check
2197     }
2198     'skip$
2199   if$
2200   new.block
2201   format.title "title" output.check
2202   "M" set.entry.mark
2203   format.mark "" output.after
2204   new.block
2205   format.translators output
2206   new.slash
2207   format.editors output
2208   new.block
2209   format.series.vol.num.booktitle "booktitle" output.check
2210   new.block
2211   format.edition output
2212   new.block
2213   format.address.publisher output
2214   year.after.author not
2215     { format.year "year" output.check }
2216     'skip$
2217   if$
2218   format.extracted.pages bbl.pages.colon output.after
2219   format.urldate "" output.after
2220   output.url
2221   output.doi
2222   new.block
2223   format.note output
2224   fin.entry
2225 }

```

B.5.3 连续出版物

```

2227 FUNCTION {periodical}
2228 { output.bibitem
2229   output.translation
2230   format.authors output
2231   author format.key output
2232   year.after.author
2233     { period.after.author
2234       'new.sentence
2235       'skip$
2236       if$
2237       format.year "year" output.check
2238     }
2239   'skip$
2240   if$
2241   new.block
2242   format.title "title" output.check
2243   "J" set.entry.mark
2244   format.mark "" output.after
2245   new.block
2246   format.periodical.year.volume.number output
2247   new.block
2248   format.address.publisher output
2249   year.after.author not
2250     { format.periodical.year "year" output.check }
2251   'skip$
2252   if$
2253   format.urldate "" output.after
2254   output.url
2255   output.doi
2256   new.block
2257   format.note output
2258   fin.entry
2259 }
2260

```

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.

```

2261 FUNCTION {journal.article}
2262 { output.bibitem
2263   output.translation
2264   format.authors output
2265   author format.key output

```

```

2266 year.after.author
2267 { period.after.author
2268   'new.sentence
2269   'skip$
2270   if$
2271     format.year "year" output.check
2272   }
2273   'skip$
2274 if$
2275 new.block
2276 title.in.journal
2277 { format.title "title" output.check
2278   entrysubtype empty$ not
2279   {
2280     entrysubtype "newspaper" =
2281       { "N" set.entry.mark }
2282       { "J" set.entry.mark }
2283     if$
2284     { "J" set.entry.mark }
2285     if$
2286     format.mark "" output.after
2287     new.block
2288   }
2289   'skip$
2290 if$
2291 format.journal "journal" output.check
2292 year.after.author not
2293 { format.date "year" output.check }
2294   'skip$
2295 if$
2296 format.journal.volume output
2297 format.journal.number "" output.after
2298 format.journal.pages bbl.pages.colon output.after
2299 format.urldate "" output.after
2300 output.url
2301 output.doi
2302 new.block
2303 format.note output
2304 fin.entry
2305 }
2306 }
2307

```

B.5.5 专利文献

number 域也可以用来表示专利号。

```

2308 FUNCTION {patent}
2309 { output.bibitem
2310   output.translation
2311   format.authors output
2312   author format.key output
2313   year.after.author
2314   { period.after.author
2315     'new.sentence

```

```

2316         'skip$
2317         if$
2318             format.year "year" output.check
2319     }
2320     'skip$
2321     if$
2322     new.block
2323     format.title "title" output.check
2324     "P" set.entry.mark
2325     format.mark "" output.after
2326     new.block
2327     format.date "year" output.check
2328     format.urldate "" output.after
2329     output.url
2330     output.doi
2331     new.block
2332     format.note output
2333     fin.entry
2334 }
2335

```

B.5.6 电子资源

```

2336 FUNCTION {electronic}
2337 { #1 #1 check.electronic
2338   #1 'entry.is.electronic :=
2339   #1 'is.pure.electronic :=
2340   output.bibitem
2341   output.translation
2342   format.authors output
2343   author format.key output
2344   year.after.author
2345     { period.after.author
2346       'new.sentence
2347       'skip$
2348       if$
2349         format.year "year" output.check
2350     }
2351   'skip$
2352   if$
2353   new.block
2354   format.series.vol.num.title "title" output.check
2355   "EB" set.entry.mark
2356   format.mark "" output.after
2357   new.block
2358   format.address.publisher output
2359   year.after.author not
2360     { date empty$
2361       { format.date output }
2362       'skip$
2363     if$
2364   }
2365   'skip$
2366   if$
2367   format.pages bbl.pages.colon output.after

```

```

2368 format.editdate "" output.after
2369 format.urldate "" output.after
2370 output.url
2371 output.doi
2372 new.block
2373 format.note output
2374 fin.entry
2375 }
2376

```

B.5.7 预印本

```

2377 FUNCTION {preprint}
2378 { output.bibitem
2379   output.translation
2380   author empty$ not
2381     { format.authors }
2382     { editor empty$ not
2383       { format.editors }
2384       { "empty author and editor in " cite$ * warning$
2385 <*author-year>
2386       bbl.anonymous
2387 </author-year>
2388 <*numerical>
2389       ""
2390 </numerical>
2391     }
2392     if$
2393   }
2394   if$
2395   output
2396   year.after.author
2397     { period.after.author
2398       'new.sentence
2399       'skip$
2400     if$
2401     format.year "year" output.check
2402   }
2403   'skip$
2404   if$
2405   new.block
2406   title.in.journal
2407     { format.series.vol.num.title "title" output.check
2408 <*2015>
2409     "A" set.entry.mark
2410 </2015>
2411 <!*2015>
2412     "Z" set.entry.mark
2413 </!2015>
2414     format.mark "" output.after
2415     new.block
2416   }
2417   'skip$
2418   if$
2419   format.translators output
2420   new.sentence

```

```

2421 format.edition output
2422 new.block
2423 year.after.author not
2424   { date empty$
2425     { format.date output }
2426     'skip$
2427   if$
2428   }
2429   'skip$
2430 if$
2431 format.pages bbl.pages.colon output.after
2432 format.editeddate "" output.after
2433 format.urldate "" output.after
2434 output.eprint
2435 output.url
2436 new.block
2437 format.note output
2438 fin.entry
2439 }
2440

```

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。

```

2441 FUNCTION {misc}
2442 { get.journal.title
2443   duplicate$ empty$ not
2444   { check.arxiv.preprint
2445     'preprint
2446     'journal.article
2447   if$
2448   }
2449   { pop$
2450     booktitle empty$ not
2451     'incollection
2452     { publisher empty$ not
2453       'monograph
2454       { eprint empty$ not archivePrefix empty$ not or
2455         'preprint
2456         { entry.is.electronic
2457           'electronic
2458           {
2459             <!*2005>
2460               "Z" set.entry.mark
2461             </!2005>
2462             <*2005>
2463               "M" set.entry.mark
2464             </2005>
2465             monograph
2466           }

```

```

2467         if$
2468     }
2469     if$
2470 }
2471 if$
2472 }
2473 if$
2474 }
2475 if$
2476 empty.misc.check
2477 }
2478
2479 FUNCTION {archive}
2480 { "A" set.entry.mark
2481   misc
2482 }
2483
2484 FUNCTION {article} { misc }
2485

```

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

```

2486 FUNCTION {book} { monograph }
2487

```

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

Required: title

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

```

2488 FUNCTION {booklet} { book }
2489
2490 FUNCTION {collection}
2491 { "G" set.entry.mark
2492   monograph
2493 }
2494
2495 FUNCTION {database}
2496 { "DB" set.entry.mark
2497   electronic
2498 }
2499
2500 FUNCTION {dataset}
2501 { "DS" set.entry.mark
2502   electronic
2503 }
2504

```

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。

```
2505 FUNCTION {inbook} { book }
2506
```

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

Required: author, title, booktitle, year

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

```
2507 FUNCTION {inproceedings}
2508 { "C" set.entry.mark
2509   incollection
2510 }
2511
```

The conference function is included for Scribe compatibility.

```
2512 FUNCTION {conference} { inproceedings }
2513
2514 FUNCTION {legislation} { archive }
2515
2516
2517 FUNCTION {map}
2518 { "CM" set.entry.mark
2519   misc
2520 }
2521
```

A manual is technical documentation.

Required: title

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

```
2522 FUNCTION {manual} { monograph }
2523
```

A mastersthesis is a Master's thesis.

Required: author, title, school, year

Optional: type, address, month, note

```
2524 FUNCTION {mastersthesis}
2525 { "D" set.entry.mark
2526   monograph
2527 }
2528
2529 FUNCTION {newspaper}
2530 { "N" set.entry.mark
2531   article
2532 }
2533
2534 FUNCTION {online}
2535 { "EB" set.entry.mark
2536   electronic

```

2537 }

2538

A phdthesis is like a mastersthesis.

Required: author, title, school, year

Optional: type, address, month, note

2539 FUNCTION {phdthesis} { mastersthesis }

2540

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

2541 FUNCTION {proceedings}

2542 { "C" set.entry.mark

2543 monograph

2544 }

2545

2546 FUNCTION {software}

2547 { "CP" set.entry.mark

2548 electronic

2549 }

2550

2551 FUNCTION {standard}

2552 { "S" set.entry.mark

2553 misc

2554 }

2555

A techreport is a technical report.

Required: author, title, institution, year

Optional: type, number, address, month, note

2556 FUNCTION {techreport}

2557 { "R" set.entry.mark

2558 misc

2559 }

2560

An unpublished is something that hasn't been published.

Required: author, title, note

Optional: month, year

2561 FUNCTION {unpublished} { misc }

2562

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

2563 FUNCTION {default.type} { misc }

2564

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

```
2565 MACRO {jan} {"January"}
2566
2567 MACRO {feb} {"February"}
2568
2569 MACRO {mar} {"March"}
2570
2571 MACRO {apr} {"April"}
2572
2573 MACRO {may} {"May"}
2574
2575 MACRO {jun} {"June"}
2576
2577 MACRO {jul} {"July"}
2578
2579 MACRO {aug} {"August"}
2580
2581 MACRO {sep} {"September"}
2582
2583 MACRO {oct} {"October"}
2584
2585 MACRO {nov} {"November"}
2586
2587 MACRO {dec} {"December"}
2588
```

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

```
2589 MACRO {acmcs} {"ACM Computing Surveys"}
2590
2591 MACRO {acta} {"Acta Informatica"}
2592
2593 MACRO {cacm} {"Communications of the ACM"}
2594
2595 MACRO {ibmjrd} {"IBM Journal of Research and Development"}
2596
2597 MACRO {ibmsj} {"IBM Systems Journal"}
2598
2599 MACRO {ieeese} {"IEEE Transactions on Software Engineering"}
2600
2601 MACRO {ieeetc} {"IEEE Transactions on Computers"}
2602
2603 MACRO {ieeetcad}
2604 {"IEEE Transactions on Computer-Aided Design of Integrated Circuits"}
```

```

2605
2606 MACRO {ipl} {"Information Processing Letters"}
2607
2608 MACRO {jacm} {"Journal of the ACM"}
2609
2610 MACRO {jcss} {"Journal of Computer and System Sciences"}
2611
2612 MACRO {scp} {"Science of Computer Programming"}
2613
2614 MACRO {sicmp} {"SIAM Journal on Computing"}
2615
2616 MACRO {tocs} {"ACM Transactions on Computer Systems"}
2617
2618 MACRO {tods} {"ACM Transactions on Database Systems"}
2619
2620 MACRO {tog} {"ACM Transactions on Graphics"}
2621
2622 MACRO {toms} {"ACM Transactions on Mathematical Software"}
2623
2624 MACRO {toois} {"ACM Transactions on Office Information Systems"}
2625
2626 MACRO {toplas} {"ACM Transactions on Programming Languages and Systems"}
2627
2628 MACRO {tcs} {"Theoretical Computer Science"}
2629

```

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`.

```

2630 FUNCTION {sortify}
2631 { purify$
2632   "l" change.case$
2633 }
2634

```

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

```

2635 FUNCTION {chop.word}
2636 { 's :=
2637   'len :=
2638   s #1 len substring$ =
2639     { s len #1 + global.max$ substring$ }
2640     's
2641   if$
2642 }
2643

```

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

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

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

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

There is one auxiliary function for each of the four different sequences of fields we use. The first of these functions looks at the author field, and then, if necessary, the key field. The other three functions, which might look at two fields and the key field, are similar, except

that the key field takes precedence over the organization field (for labels—not for sorting).

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

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

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

```
2644 FUNCTION {format.lab.name}
2645 { "{vv~}{ll}{, jj}{, ff}" format.name$ 't :=
2646   t "others" =
2647     { citation.et.al }
2648     { t get.str.lang 'name.lang :=
2649       name.lang lang.zh = name.lang lang.ja = or
2650         { t #1 "{ll}{ff}" format.name$ }
2651         { t #1 "{vv~}{ll}" format.name$ }
2652       if$
2653     }
2654   if$
2655 }
```

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。

```
2657 FUNCTION {format.lab.names}
2658 { 's :=
2659   s #1 format.lab.name 'short.label :=
2660   #1 'nameptr :=
```

```

2661 s num.names$ 'numnames :=
2662 ""
2663 numnames 'namesleft :=
2664 { namesleft #0 > }
2665 { s nameptr format.lab.name citation.et.al =
2666   numnames citation.et.al.min #1 - > nameptr citation.et.al.use.first > and or
2667   { bbl.space *
2668     citation.et.al *
2669     #1 'namesleft :=
2670   }
2671   { nameptr #1 >
2672     { namesleft #1 = citation.and "" = not and
2673       { citation.and * }
2674       { ", " * }
2675       if$
2676     }
2677     'skip$
2678     if$
2679     s nameptr format.lab.name *
2680   }
2681   if$
2682   nameptr #1 + 'nameptr :=
2683   namesleft #1 - 'namesleft :=
2684 }
2685 while$
2686 }
2687
2688 FUNCTION {author.key.label}
2689 { author empty$
2690   { key empty$
2691     { cite$ #1 #3 substring$ }
2692     'key
2693     if$
2694   }
2695   { author format.lab.names }
2696   if$
2697 }
2698
2699 FUNCTION {author.editor.key.label}
2700 { author empty$
2701   { editor empty$
2702     { key empty$
2703       { cite$ #1 #3 substring$ }
2704       'key
2705       if$
2706     }
2707     { editor format.lab.names }
2708     if$
2709   }
2710   { author format.lab.names }
2711   if$
2712 }
2713
2714 FUNCTION {author.key.organization.label}
2715 { author empty$

```

```

2716 { key empty$
2717   { organization empty$
2718     { cite$ #1 #3 substring$ }
2719     { "The " #4 organization chop.word #3 text.prefix$ }
2720     if$
2721   }
2722   'key
2723   if$
2724 }
2725 { author format.lab.names }
2726 if$
2727 }
2728
2729 FUNCTION {editor.key.organization.label}
2730 { editor empty$
2731   { key empty$
2732     { organization empty$
2733       { cite$ #1 #3 substring$ }
2734       { "The " #4 organization chop.word #3 text.prefix$ }
2735       if$
2736     }
2737     'key
2738     if$
2739   }
2740   { editor format.lab.names }
2741   if$
2742 }
2743
2744 FUNCTION {calc.short.authors}
2745 { "" 'short.label :=
2746   type$ "book" =
2747   type$ "inbook" =
2748   or
2749   'author.editor.key.label
2750   { type$ "collection" =
2751     type$ "proceedings" =
2752     or
2753     { editor empty$ not
2754       'editor.key.organization.label
2755       'author.key.organization.label
2756       if$
2757     }
2758     'author.key.label
2759     if$
2760   }
2761   if$
2762   'short.list :=
2763   short.label empty$
2764   { short.list 'short.label := }
2765   'skip$
2766   if$
2767 }
2768

```

如果 label 中有中括号“[”，分别用大括号保护起来，防止 \bibitem 处理出错。另外

为了兼容 bibunits, “name(year)fullname”的每一项都要分别保护起来, 参考 [tuna/thuthe-sis/#630](https://tuna.thuthe-sis/#630)。

```

2769 FUNCTION {calc.label}
2770 { calc.short.authors
2771   short.list "]" contains
2772   { "{" short.list * "}" * }
2773   { short.list }
2774   if$
2775   "("
2776   *
2777   format.year duplicate$ empty$
2778   short.list key field.or.null = or
2779   { pop$ "" }
2780   'skip$
2781   if$
2782   duplicate$ "]" contains
2783   { "{" swap$ * "}" * }
2784   'skip$
2785   if$
2786   *
2787   'label :=
2788   short.label
2789   "("
2790   *
2791   format.year duplicate$ empty$
2792   short.list key field.or.null = or
2793   { pop$ "" }
2794   'skip$
2795   if$
2796   *
2797   'short.label :=
2798 }
2799

```

B.8 Sorting

When sorting, we compute the sortkey by executing `presort` on each entry. The `presort` key contains a number of sortified 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`.

```

2800 (*author-year)
2801 FUNCTION {sort.language.label}
2802 { entry.lang lang.zh =
2803   { lang.zh.order }
2804   { entry.lang lang.ja =
2805     { lang.ja.order }
2806     { entry.lang lang.en =
2807       { lang.en.order }
2808       { entry.lang lang.ru =
2809         { lang.ru.order }
2810         { lang.other.order }
2811         if$
2812       }
2813     }
2814   }
2815   if$
2816 }
2817 if$
2818 #64 +
2819 int.to.chr$
2820 }
2821
2822 FUNCTION {sort.format.names}
2823 { 's :=
2824   #1 'nameptr :=
2825   ""
2826   s num.names$ 'numnames :=
2827   numnames 'namesleft :=
2828   { namesleft #0 > }
2829   {
2830     s nameptr "{vv{ } }{ll{ }}{ ff{ }}{ jj{ }}" format.name$ 't :=
2831     nameptr #1 >
2832     {
2833       " " *
2834       namesleft #1 = t "others" = and
2835       { "zzzzz" * }
2836       { numnames #2 > nameptr #2 = and
2837         { "zz" * year field.or.null * " " * }
2838         'skip$
2839       }
2840       if$
2841       t sortify *
2842     }
2843     if$
2844     { t sortify * }
2845     if$
2846     nameptr #1 + 'nameptr :=
2847     namesleft #1 - 'namesleft :=
2848   }
2849   while$
2850 }

```

2851

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`

```
2852 FUNCTION {sort.format.title}
2853 { 't :=
2854   "A " #2
2855   "An " #3
2856   "The " #4 t chop.word
2857   chop.word
2858   chop.word
2859   sortify
2860   #1 global.max$ substring$
2861 }
2862
```

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.

```
2863 FUNCTION {anonymous.sort}
2864 { entry.lang lang.zh =
2865   { "yi4 ming2" }
2866   { "anon" }
2867   if$
2868 }
2869
2870 FUNCTION {warn.empty.key}
2871 { entry.lang lang.zh =
2872   { "empty key in " cite$ * warning$ }
2873   'skip$
2874   if$
2875 }
2876
2877 FUNCTION {author.sort}
2878 { key empty$
2879   { warn.empty.key
2880     author empty$
2881     { anonymous.sort }
2882     { author sort.format.names }
2883     if$
2884   }
2885   { key }
2886   if$
2887 }
2888
2889 FUNCTION {author.editor.sort}
2890 { key empty$
2891   { warn.empty.key
2892     author empty$
2893     { editor empty$
2894       { anonymous.sort }
2895       { editor sort.format.names }
```

```

2896         if$
2897     }
2898     { author sort.format.names }
2899     if$
2900 }
2901 { key }
2902 if$
2903 }
2904
2905 FUNCTION {author.organization.sort}
2906 { key empty$
2907     { warn.empty.key
2908         author empty$
2909         { organization empty$
2910             { anonymous.sort }
2911             { "The " #4 organization chop.word sortify }
2912             if$
2913         }
2914         { author sort.format.names }
2915         if$
2916     }
2917     { key }
2918     if$
2919 }
2920
2921 FUNCTION {editor.organization.sort}
2922 { key empty$
2923     { warn.empty.key
2924         editor empty$
2925         { organization empty$
2926             { anonymous.sort }
2927             { "The " #4 organization chop.word sortify }
2928             if$
2929         }
2930         { editor sort.format.names }
2931         if$
2932     }
2933     { key }
2934     if$
2935 }
2936
2937 </author-year>

```

顺序编码制的排序要简单得多

```

2938 < *numerical>
2939 INTEGERS { seq.num }
2940
2941 FUNCTION {init.seq}
2942 { #0 'seq.num :=}
2943
2944 FUNCTION {int.to.fix}
2945 { "000000000" swap$ int.to.str$ *
2946     #-1 #10 substring$
2947 }
2948

```

2949 </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!

```

2950 FUNCTION {presort}
2951 { set.entry.lang
2952   set.entry.numbered
2953   show.url show.doi check.electronic
2954   #0 'is.pure.electronic :=
2955   calc.label
2956   label sortify
2957   " "
2958   *
2959   <*author-year>
2960   sort.language.label
2961   " "
2962   *
2963   type$ "book" =
2964   type$ "inbook" =
2965   or
2966     'author.editor.sort
2967     { type$ "collection" =
2968       type$ "proceedings" =
2969       or
2970         'editor.organization.sort
2971         'author.sort
2972       if$
2973     }
2974   if$
2975   *
2976   " "
2977   *
2978   year field.or.null sortify
2979   *
2980   " "
2981   *
2982   cite$
2983   *
2984   #1 entry.max$ substring$
2985   </author-year>
2986   <*numerical>
2987   seq.num #1 + 'seq.num :=
2988   seq.num int.to.fix
2989   </numerical>
2990   'sort.label :=
2991   sort.label *
2992   #1 entry.max$ substring$
2993   'sort.key$ :=
2994 }
2995

```

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

```

```

2996 STRINGS { longest.label last.label next.extra last.extra.label }
2997
2998 INTEGERS { longest.label.width number.label }
2999
3000 FUNCTION {initialize.longest.label}
3001 { "" 'longest.label :=
3002   #0 int.to.chr$ 'last.label :=
3003   "" 'next.extra :=
3004   #0 'longest.label.width :=
3005   #0 'number.label :=
3006   "" 'last.extra.label :=
3007 }
3008

```

```

3009 FUNCTION {forward.pass}
3010 {
3011   ⟨*author-year⟩
3012   last.label short.label =
3013   { "" 'extra.label :=
3014     last.extra.label text.length$ 'charptr :=
3015     { last.extra.label charptr #1 substring$ "z" =
3016       charptr #0 > and
3017     }
3018     { "a" extra.label * 'extra.label :=
3019       charptr #1 - 'charptr :=
3020     }
3021     while$
3022     charptr #0 >
3023     { last.extra.label charptr #1 substring$ chr.to.int$ #1 + int.to.chr$
3024       extra.label * 'extra.label :=
3025       last.extra.label #1 charptr #1 - substring$
3026       extra.label * 'extra.label :=
3027     }
3028     { "a" extra.label * 'extra.label := }
3029     if$
3030     extra.label 'last.extra.label :=
3031   }
3032   { "a" 'last.extra.label :=
3033     "" 'extra.label :=
3034     short.label 'last.label :=
3035   }
3036   if$
3037   ⟨/author-year⟩
3038   number.label #1 + 'number.label :=
3039 }
3040
3041 FUNCTION {reverse.pass}
3042 {
3043   ⟨*author-year⟩
3044   next.extra "b" =
3045   { "a" 'extra.label := }
3046   'skip$
3047   if$
3048   extra.label 'next.extra :=
3049   extra.label
3050   duplicate$ empty$
3051   'skip$
3052   { "{\natexlab{" swap$ * "}" * }
3053   if$
3054   'extra.label :=
3055   ⟨/author-year⟩
3056   label extra.label * 'label :=
3057 }
3058
3059 FUNCTION {bib.sort.order}
3060 { sort.label 'sort.key$ :=
3061 }
3062

```

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.

```
3063 FUNCTION {begin.bib}
3064 {   preamble$ empty$
3065     'skip$
3066     { preamble$ write$ newline$ }
3067     if$
3068     "\begin{thebibliography}{\" number.label int.to.str$ * \"} *
3069     write$ newline$
3070     terms.in.macro
3071     { "\providecommand{\biband}{和}"
3072       write$ newline$
3073       "\providecommand{\bibetal}{等}"
3074       write$ newline$
3075     }
3076     'skip$
3077     if$
3078     "\providecommand{\natexlab}[1]{#1}"
3079     write$ newline$
3080     "\providecommand{\url}[1]{#1}"
3081     write$ newline$
3082     "\expandafter\ifx\csname urlstyle\endcsname\relax\else"
3083     write$ newline$
3084     " \urlstyle{same}\fi"
3085     write$ newline$
3086     "\expandafter\ifx\csname href\endcsname\relax"
3087     write$ newline$
3088     " \DeclareUrlCommand\doi{\urlstyle{rm}}"
3089     write$ newline$
3090     " \def\epri#1#2{#2}"
3091     write$ newline$
3092     "\else"
3093     write$ newline$
3094     " \def\doi#1{\href{https://doi.org/#1}{\nolinkurl{#1}}}"
3095     write$ newline$
3096     " \let\epri\href"
3097     write$ newline$
3098     "\fi"
3099     write$ newline$
3100   }
3101
```

Finally, we finish up by writing the '`\end{thebibliography}`' command.

```
3102 FUNCTION {end.bib}
3103 { newline$
3104   "\end{thebibliography}" write$ newline$
3105 }
```

3106

B.10 Main execution

Now we read in the .BIB entries.

```
3107 READ
3108
3109 EXECUTE {init.state.consts}
3110
3111 EXECUTE {load.config}
3112
3113 ⟨*numerical⟩
3114 EXECUTE {init.seq}
3115
3116 ⟨/numerical⟩
3117 ITERATE {presort}
3118
```

And now we can sort

```
3119 SORT
3120
3121 EXECUTE {initialize.longest.label}
3122
3123 ITERATE {forward.pass}
3124
3125 REVERSE {reverse.pass}
3126
3127 ITERATE {bib.sort.order}
3128
3129 SORT
3130
3131 EXECUTE {begin.bib}
3132
```

Now we produce the output for all the entries

```
3133 ITERATE {call.type$}
3134
3135 EXECUTE {end.bib}
3136 ⟨/author-year | numerical⟩
```