

GB/T 7714 BibTeX style

Zeping Lee*

2026/01/15 v2.1.9

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 LaTeX 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 \bibliographystyle 命令选择参考文献样式,如 gbt7714-numerical 或 gbt7714-author-year。在早期的版本中,选择文献样式的方法是将 numbers 或 super 等参数传递给 gbt7714,而不能使用 \bibliographystyle。这跟标准的 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 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]。

`locator-inside-brackets` 国标要求在括号外以角标的形式著录引文页码。如果要将页码置于括号内，可以在调用宏包时设置 `locator-inside-brackets=true`。

```
\usepackage[locator-inside-brackets=true]{gbt7714}
```

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）。

5 著录项目

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

author 主要责任者

title 题名

mark* 文献类型标识

medium* 载体类型标识

translator* 译者

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

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

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 BibTeX 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

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	输出著者的最多数量
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
link.journal	#0	在期刊题名上添加 url 的超链接
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
page.range.delimiter	"-"	起止页码中的连接号
show.urldate	#1	显示引用日期 urldate
show.url	#1	显示 url
show.doi	#1	显示 DOI
show.note	#0	显示 note 域的信息
end.with.period	#1	结尾加句点
lowercase.word.after.colon	#1	将冒号后的单词变成小写

格。李志奇^[8]发布了严格遵循 GB/T 7714—2005 的 BibLaTeX 的样式。胡海星^[9]提供了另一个 BibTeX 实现, 还给每行 bst 代码写了 java 语言注释。沈周^[10]基于 biblatex-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] 中国国家标准委员会. 信息与文献 参考文献著录规则: GB/T 7714—2015[S]. 北京: 中国标准出版社, 2015.
- [2] PATASHNIK O. BibTeXing[M/OL]. 1988. <http://mirrors.ctan.org/biblio/bibtex/base/>

[btxdoc.pdf](#).

- [3] DALY P W. Natural sciences citations and references[M/OL]. 1999. <http://mirrors.ctan.org/macros/latex/contrib/natbib/natbib.pdf>.
- [4] PATASHNIK O. Designing BibTeX styles[M/OL]. 1988. <http://mirrors.ctan.org/biblio/bibtex/base/btxhak.pdf>.
- [5] MARKEY N. Tame the beast[M/OL]. 2003. http://mirrors.ctan.org/info/bibtex/tamethebeast/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 \DeclareKeys[gbt7714]{
10   2015 .code = {%
11     \gbt@obsolete@option{2015}%
12     \gbt@legacy@interfacetrue
13     \gbt@mmxvtrue
14   },
15   2005 .code = {%
16     \gbt@obsolete@option{2005}%
17     \gbt@legacy@interfacetrue
18     \gbt@mmxvfalse
19   },
20   super .code = {%
21     \gbt@obsolete@option{super}%
22     \gbt@legacy@interfacetrue
23     \gbt@numericaltrue
24     \gbt@supertrue
25   },
26   numbers .code = {%
27     \gbt@obsolete@option{numbers}%
28     \gbt@legacy@interfacetrue
29     \gbt@numericaltrue
30     \gbt@superfalse
31   },
32   authoryear .code = {%
33     \gbt@obsolete@option{authoryear}%
34     \gbt@legacy@interfacetrue
35     \gbt@numericalfalse
36   },
37 }
```

控制引注的页码在括号内还是在括号外。


```

38 \DeclareKeys[gbt7714]{
39   locator-inside-brackets .if = @gbt@locator@inside@affixes ,
40 }
41 \SetKeys[gbt7714]{
42   locator-inside-brackets = false ,
43 }

```

将选项传递给 **natbib**

```

44 \DeclareUnknownKeyHandler[gbt7714]{\PassOptionsToPackage{#1}{natbib}}
45 \ProcessKeyOptions[gbt7714]

```

调用宏包，注意只需要 **compress** 不需要 **sort**。

```

46 \RequirePackage{natbib}
47 \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。

```

48 \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**。

```

49 \renewcommand\newblock{\space}
50 \newcommand\bibstyle@super{\bibpunct{[ ]}{,}{s}{,}{\textsuperscript{,}}{,}}
51 \newcommand\bibstyle@numbers{\bibpunct{[ ]}{,}{n}{,}{,}{,}}
52 \newcommand\bibstyle@authoryear{\bibpunct{( )}{;}{a}{,}{,}{,}}
53 \newcommand\bibstyle@inline{\bibstyle@numbers}

```

(End of 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。

```

54 \@namedef{bibstyle@gbt7714-numerical}{\bibstyle@super}
55 \@namedef{bibstyle@gbt7714-author-year}{\bibstyle@authoryear}
56 \@namedef{bibstyle@gbt7714-2005-numerical}{\bibstyle@super}
57 \@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](#)。

```

58 \renewcommand\NAT@citesuper[3]{%
59   \ifNAT@swa
60     \if*#2*\else
61       #2\NAT@spacechar
62     \fi
63     \unskip\kern\p@
64     \textsuperscript{%

```

```

65     \NAT@@open
66     #1%
67     \if@gbt@locator@inside@affixes
68     \if*#3*\else
69         \NAT@cmt#3%
70     \fi
71     \NAT@@close
72 \else
73     \NAT@@close
74     \if*#3*\else
75         #3%
76     \fi
77 \fi
78 }%
79 \kern\p@
80 \else
81     #1%
82 \fi
83 \endgroup
84 }

```

将 **numbers** 样式的 `\citep` 的页码置于括号外。

```

85 \renewcommand\NAT@citenum[3]{%
86     \ifNAT@swa
87     \NAT@@open
88     \if*#2*\else
89         #2\NAT@spacechar
90     \fi
91     #1%
92     \if@gbt@locator@inside@affixes
93     \if*#3*\else\NAT@cmt#3\fi\NAT@@close
94 \else
95     \NAT@@close
96     \if*#3*\else
97         \textsuperscript{#3}%
98     \fi
99 \fi
100 \else
101     #1%
102 \fi
103 \endgroup
104 }

```

Numerical 模式的 \citet 的页码:

```
105 \def\NAT@citexnum[#1][#2]#3{%
106   \NAT@reset@parser
107   \NAT@sort@cites{#3}%
108   \NAT@reset@citea
109   \@cite{\def\NAT@num{-1}\let\NAT@last@yr\relax\let\NAT@nm\@empty
110     \@for\@citeb:=\NAT@cite@list\do
111       {\@safe@activestrue
112         \edef\@citeb{\expandafter\@firstofone\@citeb\@empty}%
113         \@safe@activesfalse
114         \@ifundefined{b@\@citeb\@extra@b@citeb}{%
115           {\reset@font\bfseries?}%
116           \NAT@citeundefined\PackageWarning{natbib}%
117             {Citation `'\@citeb' on page \thepage \space undefined}}%
118         {\let\NAT@last@num\NAT@num\let\NAT@last@nm\NAT@nm
119           \NAT@parse{\@citeb}%
120           \ifNAT@longnames\@ifundefined{bv@\@citeb\@extra@b@citeb}{%
121             \let\NAT@name=\NAT@all@names
122             \global\@namedef{bv@\@citeb\@extra@b@citeb}{}%
123           }%
124           \ifNAT@full\let\NAT@nm\NAT@all@names\else
125             \let\NAT@nm\NAT@name\fi
126           \ifNAT@swa
127             \@ifnum{\NAT@ctype>\@ne}{%
128               \@citea
129               \NAT@hyper@{\@ifnum{\NAT@ctype=\tw@}{\NAT@test{\NAT@ctype}}{\NAT@al
130             }%
131             \@ifnum{\NAT@cmprs>\z@}{%
132               \NAT@ifcat@num\NAT@num
133               {\let\NAT@nm=\NAT@num}%
134               {\def\NAT@nm{-2}}%
135             \NAT@ifcat@num\NAT@last@num
136               {\@tempcnta=\NAT@last@num\relax}%
137               {\@tempcnta\m@ne}%
138             \@ifnum{\NAT@nm=\@tempcnta}{%
139               \@ifnum{\NAT@merge>\@ne}{\NAT@last@yr@mbox}%
140             }%
141             \advance\@tempcnta by\@ne
142             \@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。参考 <https://tex.stackexchange.com/>

[a/86991/82731](#)。

```
143      % \ifx\NAT@last@yr\relax
144      % \def@NAT@last@yr{\@citea}%
145      % \else
146      % \def@NAT@last@yr{--\NAT@penalty}%
147      % \fi
148      \def@NAT@last@yr{-\NAT@penalty}%
149      }{%
150      \NAT@last@yr@mbox
151      }%
152      }%
153      }{%
154      \@tempswatrue
155      \@ifnum{\NAT@merge>\@ne}{\@ifnum{\NAT@last@num=\NAT@num\relax}{\@t
156      \if@tempswa\NAT@citea@mbox\fi
157      }%
158      }%
159      \NAT@def@citea
160      \else
161      \ifcase\NAT@ctype
162      \ifx\NAT@last@nm\NAT@nm \NAT@yrsep\NAT@penalty\NAT@space\else
163      \@citea \NAT@test{\@ne}\NAT@spacechar\NAT@mbox{\NAT@super@kern\
164      \fi
165      \if*#1*\else#1\NAT@spacechar\fi
166      \NAT@mbox{\NAT@hyper@{\@citenumfont{\NAT@num}}}%
167      \NAT@def@citea@box
168      \or
169      \NAT@hyper@citea@space{\NAT@test{\NAT@ctype}}%
170      \or
171      \NAT@hyper@citea@space{\NAT@test{\NAT@ctype}}%
172      \or
173      \NAT@hyper@citea@space\NAT@alias
174      \fi
175      \fi
176      }%
177      }%
178      \@ifnum{\NAT@cmprs>\z@}{\NAT@last@yr}{}%
179      \ifNAT@swa\else
```

将页码放在括号外边，并且置于上标。

```
180      \if@gbt@locator@inside@affixes
181      \@ifnum{\NAT@ctype=\z@}{%
182      \if*#2*\else\NAT@cmt#2\fi
```

```

183         {}%
184         \NAT@mbbox{\NAT@@close}%
185     \else
186         \NAT@mbbox{\NAT@@close}%
187         \@ifnum{\NAT@ctype=\z@}{%
188             \if*#2*\else
189                 \textsuperscript{#2}%
190             \fi
191         }{}%
192         \NAT@super@kern
193     \fi
194 \fi
195 }{#1}{#2}%
196 }%

```

Author-year 模式的 \citep 的页码:

```

197 \renewcommand\NAT@cite%
198     [3]{\ifNAT@swa\NAT@@open\if*#2*\else#2\NAT@spacechar\fi
199         #1%
200         \if@gbt@locator@inside@affixes
201             \if*#3*\else\NAT@cmt#3\fi\NAT@@close
202         \else
203             \NAT@@close\if*#3*\else\textsuperscript{#3}\fi
204         \fi
205         \else#1\fi\endgroup}

```

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

Author-year 模式的 \citex 的页码:

```

206 \def\NAT@citex%
207     [#1][#2]#3{%
208     \NAT@reset@parser
209     \NAT@sort@cites{#3}%
210     \NAT@reset@citea
211     \@cite{\let\NAT@nm\@empty\let\NAT@year\@empty
212         \@for\@citeb:=\NAT@cite@list\do
213         {\@safe@activestru
214             \edef\@citeb{\expandafter\@firstofone\@citeb\@empty}%
215             \@safe@activesfalse
216             \@ifundefined{b@\@citeb\@extra@b@citeb}{\@citea%
217                 {\reset@font\bfseries ?}\NAT@citeundefined
218                 \PackageWarning{natbib}%
219                 {Citation '\@citeb' on page \thepage \space undefined}}{\def\NAT@date{
220                 {\let\NAT@last@nm=\NAT@nm\let\NAT@last@yr=\NAT@year

```

```

221 \NAT@parse{\@citeb}%
222 \ifNAT@longnames\@ifundefined{bv@\@citeb\@extra@b@citeb}{%
223 \let\NAT@name=\NAT@all@names
224 \global\@namedef{bv@\@citeb\@extra@b@citeb}{}}{}%
225 \fi
226 \ifNAT@full\let\NAT@nm\NAT@all@names\else
227 \let\NAT@nm\NAT@name\fi
228 \ifNAT@swa\ifcase\NAT@ctype
229 \if\relax\NAT@date\relax
230 \@citea\NAT@hyper@{\NAT@nmfmt{\NAT@nm}\NAT@date}%
231 \else
232 \ifx\NAT@last@nm\NAT@nm\NAT@yrsep
233 \ifx\NAT@last@yr\NAT@year
234 \def\NAT@temp{?}%
235 \ifx\NAT@temp\NAT@exlab\PackageWarningNoLine{natbib}%
236 {Multiple citation on page \thepage: same authors and
237 year\MessageBreak without distinguishing extra
238 letter,\MessageBreak appears as question mark}\fi
239 \NAT@hyper@{\NAT@exlab}%
240 \else\unskip\NAT@spacechar
241 \NAT@hyper@{\NAT@date}%
242 \fi
243 \else
244 \@citea\NAT@hyper@{%
245 \NAT@nmfmt{\NAT@nm}%
246 \hyper@natlinkbreak{%
247 \NAT@aysep\NAT@spacechar}{\@citeb\@extra@b@citeb
248 }%
249 \NAT@date
250 }%
251 \fi
252 \fi
253 \or\@citea\NAT@hyper@{\NAT@nmfmt{\NAT@nm}}%
254 \or\@citea\NAT@hyper@{\NAT@date}%
255 \or\@citea\NAT@hyper@{\NAT@alias}%
256 \fi \NAT@def@citea
257 \else
258 \ifcase\NAT@ctype
259 \if\relax\NAT@date\relax
260 \@citea\NAT@hyper@{\NAT@nmfmt{\NAT@nm}}%
261 \else
262 \ifx\NAT@last@nm\NAT@nm\NAT@yrsep

```

```

263         \ifx\NAT@last@yr\NAT@year
264         \def\NAT@temp{{?}}%
265         \ifx\NAT@temp\NAT@exlab\PackageWarningNoLine{natbib}%
266         {Multiple citation on page \thepage: same authors and
267         year\MessageBreak without distinguishing extra
268         letter,\MessageBreak appears as question mark}\fi
269         \NAT@hyper@{\NAT@exlab}%
270     \else
271         \unskip\NAT@spacechar
272         \NAT@hyper@{\NAT@date}%
273     \fi
274 \else
275     \@citea\NAT@hyper@{%
276     \NAT@nmfmt{\NAT@nm}%
277     \hyper@natlinkbreak{\NAT@spacechar\NAT@@open\if*#1*\else#1\NAT
278     {\@citeb\@extra@b@citeb}%
279     \NAT@date
280     }%
281     \fi
282 \fi
283 \or\@citea\NAT@hyper@{\NAT@nmfmt{\NAT@nm}}%
284 \or\@citea\NAT@hyper@{\NAT@date}%
285 \or\@citea\NAT@hyper@{\NAT@alias}%
286 \fi
287 \if\relax\NAT@date\relax
288     \NAT@def@citea
289 \else
290     \NAT@def@citea@close
291 \fi
292 \fi
293 }}\ifNAT@swa\else
    \if@gbt@locator@inside@affixes
294     \if*#2*\else\NAT@cmt#2\fi
295     \if\relax\NAT@date\relax\else\NAT@@close\fi
296 \else
297     \if\relax\NAT@date\relax\else\NAT@@close\fi
298     \if*#2*\else\textsuperscript{#2}\fi
299 \fi
300 \fi}{#1}{#2}}
301

```

将页码放在括号外边，并且置于上标。

thebibliography (*env.*) 参考文献列表的标签左对齐

Patch `natbib` 内部命令，以支持 `\noopsort`。参考 <https://tex.stackexchange.com/a/39718/82731>。

`\url` 使用 `xurl` 宏包的方法，增加 URL 可断行的位置。

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

```

315 \newif\ifgbt@bib@style@written
316 \@ifpackageloaded{chapterbib}{%}%
317 \def\bibliography#1{%
318   \ifgbt@bib@style@written\else
319     \bibliographystyle{gbt7714-numerical}%
320   \fi
321   \if@filesw
322     \immediate\write\@auxout{\string\bibdata{\zap@space#1 \@empty}}%
323   \fi
324   \@input{\jobname.bbl}%
325 \def\bibliographystyle#1{%
326   \gbt@bib@style@writtenttrue
327   \ifx\@begindocumenthook\@undefined\else
328     \expandafter\AtBeginDocument
329   \fi
330   {\if@filesw
331     \immediate\write\@auxout{\string\bibstyle{#1}}%
332   \fi}%
333 }%
334 }
335 \ifgbt@legacy@interface
336 \ifgbt@numerical

```



```

337     \ifgbt@super\else
338         \citestyle{numbers}
339     \fi
340     \bibliographystyle{gbt7714-numerical}
341 \else
342     \bibliographystyle{gbt7714-author-year}
343 \fi
344 \fi
345 \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。

```

346 <{*author-year | numerical}
347 INTEGERS {
348     citation.et.al.min
349     citation.et.al.use.first
350     bibliography.et.al.min
351     bibliography.et.al.use.first
352     uppercase.name
353     terms.in.macro
354     year.after.author
355     period.after.author
356     italic.book.title
357     sentence.case.title
358     link.title
359     title.in.journal
360     show.patent.country
361     show.mark
362     space.before.mark
363     show.medium.type
364     short.journal
365     italic.journal
366     link.journal
367     bold.journal.volume
368     show.missing.address.publisher
369     space.before.pages
370     only.start.page
371     show.urldate
372     show.url
373     show.doi
374     show.note
375     show.english.translation
376     end.with.period
377     lowercase.word.after.colon
378 <{*author-year}

```

```

379 lang.zh.order
380 lang.ja.order
381 lang.en.order
382 lang.ru.order
383 lang.other.order
384 </author-year>
385 }
386
387 STRINGS {
388   component.part.label
389   page.range.delimiter
390 }
391

```

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。

```

392 FUNCTION {load.config}
393 {

```

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”。

```

394 <!*ucas>
395   #2 'citation.et.al.min :=
396   #1 'citation.et.al.use.first :=
397 </!*ucas>
398 <*ucas>
399   #3 'citation.et.al.min :=
400   #1 'citation.et.al.use.first :=
401 </ucas>
402   #4 'bibliography.et.al.min :=
403   #3 'bibliography.et.al.use.first :=

```

英文姓名转为全大写：

```

404 <*(no-uppercase | thu | ustc)>
405   #1 'uppercase.name :=
406 </!(no-uppercase | thu | ustc)>
407 <*no-uppercase | thu | ustc>
408   #0 'uppercase.name :=
409 </no-uppercase | thu | ustc>

```

使用 TeX 宏输出“和”、“等”

```

410 <*(macro |ucas)>
411   #0 'terms.in.macro :=
412 </!(macro |ucas)>
413 <*macro |ucas>
414   #1 'terms.in.macro :=
415 </macro |ucas>

```

将年份置于著者后面（著者-出版年制默认）

```

416 <*numerical |ucas>
417   #0 'year.after.author :=
418 </numerical |ucas>
419 <*author-year&!ucas>
420   #1 'year.after.author :=

```

```
421 </author-year&!lucas>
```

采用著者-出版年制时，作者姓名与年份之间使用句点连接：

```
422 <*numerical>
423   #1 'period.after.author :=
424 </numerical>
425 <*author-year>
426 <*2015&!(period)>
427   #0 'period.after.author :=
428 </2015&!(period)>
429 <*period | 2005>
430   #1 'period.after.author :=
431 </period | 2005>
432 </author-year>
```

书名使用斜体：

```
433 <!*italic-book-title>
434   #0 'italic.book.title :=
435 </!italic-book-title>
436 <*italic-book-title>
437   #1 'italic.book.title :=
438 </italic-book-title>
```

英文标题转为 sentence case（句首字母大写，其余小写）：

```
439 <!*no-sentence-case>
440   #1 'sentence.case.title :=
441 </!no-sentence-case>
442 <*no-sentence-case>
443   #0 'sentence.case.title :=
444 </no-sentence-case>
```

在标题添加超链接：

```
445 <!*link-title>
446   #0 'link.title :=
447 </!link-title>
448 <*link-title>
449   #1 'link.title :=
450 </link-title>
```

期刊是否含标题：

```
451 <!*no-title-in-journal>
452   #1 'title.in.journal :=
453 </!no-title-in-journal>
454 <*no-title-in-journal>
455   #0 'title.in.journal :=
456 </no-title-in-journal>
```

专利题名是否含专利国别

```
457 <*(show-patent-country | 2005 | thu)>
458   #0 'show.patent.country :=
459 </!(show-patent-country | 2005 | thu)>
460 <*(show-patent-country | 2005 | thu)>
461   #1 'show.patent.country :=
462 </(show-patent-country | 2005 | thu)>
```

著录文献类型标识（比如“[M/OL]”）：

```
463 <!*no-mark>
464     #1 'show.mark :=
465 </!no-mark>
466 <*no-mark>
467     #0 'show.mark :=
468 </no-mark>
```

文献类型标识前是否有空格：

```
469 <!*space-before-mark>
470     #0 'space.before.mark :=
471 </!space-before-mark>
472 <*space-before-mark>
473     #1 'space.before.mark :=
474 </space-before-mark>
```

是否显示载体类型标识（比如“/OL”）：

```
475 <!*no-medium-type>
476     #1 'show.medium.type :=
477 </!no-medium-type>
478 <*no-medium-type>
479     #0 'show.medium.type :=
480 </no-medium-type>
```

使用“//”表示析出文献

```
481 <*(in-collection | no-slash)>
482     "slash" 'component.part.label :=
483 </!(in-collection | no-slash)>
484 <*in-collection>
485     "in" 'component.part.label :=
486 </in-collection>
487 <*no-slash>
488     "none" 'component.part.label :=
489 </no-slash>
```

期刊名使用缩写：

```
490 <!*short-journal>
491     #0 'short.journal :=
492 </!short-journal>
493 <*short-journal>
494     #1 'short.journal :=
495 </short-journal>
```

期刊名使用斜体：

```
496 <!*italic-journal>
497     #0 'italic.journal :=
498 </!italic-journal>
499 <*italic-journal>
500     #1 'italic.journal :=
501 </italic-journal>
```

在期刊题名添加超链接：

```
502 <!*link-journal>
503     #0 'link.journal :=
```

```

504 </!link-journal>
505 <*link-journal>
506   #1 'link.journal :=
507 </link-journal>

```

期刊的卷使用粗体:

```

508   #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.”:

```

509 <!*sl-sn>
510   #0 'show.missing.address.publisher :=
511 </!sl-sn>
512 <*sl-sn>
513   #1 'show.missing.address.publisher :=
514 </sl-sn>

```

页码与前面的冒号之间是否有空格:

```

515 <!*no-space-before-pages>
516   #1 'space.before.pages :=
517 </!no-space-before-pages>
518 <*no-space-before-pages>
519   #0 'space.before.pages :=
520 </no-space-before-pages>

```

页码是否只含起始页:

```

521 <!*only-start-page>
522   #0 'only.start.page :=
523 </!only-start-page>
524 <*only-start-page>
525   #1 'only.start.page :=
526 </only-start-page>

```

起止页码中的连接号:

```

527 <*(en-dash-page-range-delimiter | wave-dash-page-range-delimiter)>
528   "-" 'page.range.delimiter :=
529 </!(en-dash-page-range-delimiter | wave-dash-page-range-delimiter)>
530 <*en-dash-page-range-delimiter>
531   "--" 'page.range.delimiter :=
532 </en-dash-page-range-delimiter>
533 <*wave-dash-page-range-delimiter>
534   "~" 'page.range.delimiter :=
535 </wave-dash-page-range-delimiter>

```

是否著录非电子文献的引用日期:

```

536 <!*no-urldate>
537   #1 'show.urldate :=
538 </!no-urldate>
539 <*no-urldate>
540   #0 'show.urldate :=
541 </no-urldate>

```

是否著录 URL:

```

542 <*(no-url | ustc)>
543   #1 'show.url :=

```

```

544 </!(no-url | ustc)>
545 <*no-url | ustc>
546     #0 'show.url :=
547 </no-url | ustc>

```

是否著录 DOI:

```

548 <*(no-doi | 2005 | ustc)>
549     #1 'show.doi :=
550 </!(no-doi | 2005 | ustc)>
551 <*no-doi | 2005 | ustc>
552     #0 'show.doi :=
553 </no-doi | 2005 | ustc>

```

在每一条文献最后输出注释 (note) 的内容:

```

554     #0 'show.note :=

```

中文文献是否显示英文翻译

```

555 <!*show-english-translation>
556     #0 'show.english.translation :=
557 </!show-english-translation>
558 <*show-english-translation>
559     #1 'show.english.translation :=
560 </show-english-translation>

```

结尾加句点

```

561 <!*no-period-at-end>
562     #1 'end.with.period :=
563 </!no-period-at-end>
564 <*no-period-at-end>
565     #0 'end.with.period :=
566 </no-period-at-end>

```

将冒号后的单词变成小写

```

567 <!*no-lowercase-word-after-colon>
568     #1 'lowercase.word.after.colon :=
569 </!no-lowercase-word-after-colon>
570 <*no-lowercase-word-after-colon>
571     #0 'lowercase.word.after.colon :=
572 </no-lowercase-word-after-colon>

```

参考文献表按照“著者-出版年”组织时, 各个文种的顺序:

```

573 <*author-year>
574     #1 'lang.zh.order :=
575     #2 'lang.ja.order :=
576     #3 'lang.en.order :=
577     #4 'lang.ru.order :=
578     #5 'lang.other.order :=
579 </author-year>
580 }
581

```

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.

```
582 ENTRY
583   { address
584     archivePrefix
585     author
586     booktitle
587     date
588     doi
589     edition
590     editor
591     eprint
592     eprinttype
593     entrysubtype
594     howpublished
595     institution
596     journal
597     journaltitle
598     key
599     langid
600     language
601     location
602     mark
603     medium
604     note
605     number
606     organization
607     pages
608     publisher
609     school
610     series
611     shortjournal
612     title
613     translation
614     translator
615     url
616     urldate
617     volume
618     year
619   }
620   { 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.

```
621   { label extra.label sort.label short.list entry.mark entry.url }
622
```

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$(pop() * ", ")
    -- "pop" isn't a function: just use stack top
  elseif
  if output.state = after.block then
    write$(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
```

```
623 INTEGERS { output.state before.all mid.sentence after.sentence after.block
624
625 INTEGERS { lang.zh lang.ja lang.en lang.ru lang.other }
626
627 INTEGERS { charptr len }
628
629 FUNCTION {init.state.consts}
630 { #0 'before.all :=
631   #1 'mid.sentence :=
632   #2 'after.sentence :=
633   #3 'after.block :=
634   #4 'after.slash :=
635   #3 'lang.zh :=
636   #4 'lang.ja :=
637   #1 'lang.en :=
638   #2 'lang.ru :=
639   #0 'lang.other :=
640 }
641
```

下面是一些常量的定义

```
642 FUNCTION {bbl.anonymous}
643 { entry.lang lang.zh =
```

```

644     { " 佚名" }
645     { "Anon" }
646   if$
647 }
648
649 FUNCTION {bbl.space}
650 { entry.lang lang.zh =
651   { "\ " }
652   { " " }
653   if$
654 }
655
656 FUNCTION {bbl.and}
657 { " " }
658
659 FUNCTION {bbl.et.al}
660 { entry.lang lang.zh =
661   { " 等" }
662   { entry.lang lang.ja =
663     { " 他" }
664     { entry.lang lang.ru =
665       { " " }
666       { "et~al." }
667       if$
668     }
669     if$
670   }
671   if$
672 }
673
674 FUNCTION {citation.and}
675 { terms.in.macro
676   { "{\biband}" }
677   'bbl.and
678   if$
679 }
680
681 FUNCTION {citation.et.al}
682 { terms.in.macro
683   { "{\bibetal}" }
684   'bbl.et.al
685   if$
686 }
687
688 FUNCTION {bbl.colon} { ": " }
689
690 FUNCTION {bbl.pages.colon}
691 { space.before.pages
692   { ": " }
693   { ":\allowbreak " }
694   if$
695 }
696
697 <!*2005>
698 FUNCTION {bbl.wide.space} { "\quad " }

```

[illegible]

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

```

735 FUNCTION {not}
736 { { #0 }
737   { #1 }
738   if$
739 }
740
741 FUNCTION {and}
742 { 'skip$
743   { pop$ #0 }
744   if$
745 }
746
747 FUNCTION {or}
748 { { pop$ #1 }

```

```

749     'skip$
750   if$
751 }
752
753 STRINGS { x y }
754
755 FUNCTION {contains}
756 { 'y :=
757   'x :=
758   y text.length$ 'len :=
759   x text.length$ len - #1 + 'charptr :=
760   { charptr #0 >
761     x charptr len substring$ y = not
762     and
763   }
764   { charptr #1 - 'charptr := }
765   while$
766   charptr #0 >
767 }
768

```

the variables s and t are temporary string holders

```

769 STRINGS { s t }
770
771 FUNCTION {output.nonnull}
772 { 's :=
773   output.state mid.sentence =
774   { ", " * write$ }
775   { output.state after.block =
776     { add.period$ write$
777       newline$
778       "\newblock " write$
779     }
780     { output.state before.all =
781       'write$
782       { output.state after.slash =
783         { bbl.slash * write$
784           newline$
785         }
786         { add.period$ " " * write$ }
787       if$
788     }
789     if$
790   }
791   if$
792   mid.sentence 'output.state :=
793 }
794 if$
795 s
796 }
797
798 FUNCTION {output}
799 { duplicate$ empty$
800   'pop$
801   'output.nonnull

```

```

802   if$
803 }
804
805 FUNCTION {output.after}
806 { 't :=
807   duplicate$ empty$
808   'pop$
809   { 's :=
810     output.state mid.sentence =
811     { t * write$ }
812     { output.state after.block =
813       { add.period$ write$
814         newline$
815         "\newblock " write$
816       }
817       { output.state before.all =
818         'write$
819         { output.state after.slash =
820           { bbl.slash * write$ }
821           { add.period$ " " * write$ }
822           if$
823         }
824         if$
825       }
826       if$
827       mid.sentence 'output.state :=
828     }
829     if$
830     s
831   }
832   if$
833 }
834
835 FUNCTION {output.check}
836 { 't :=
837   duplicate$ empty$
838   { pop$ "empty " t * " in " * cite$ * warning$ }
839   'output.nonnull
840   if$
841 }
842

```

This function finishes all entries.

```

843 FUNCTION {fin.entry}
844 { end.with.period
845   'add.period$
846   'skip$
847   if$
848   write$
849   show.english.translation entry.lang lang.zh = and
850   { " ) "
851     write$
852   }
853   'skip$
854   if$

```

```

855     newline$
856 }
857
858 FUNCTION {new.block}
859 { output.state before.all =
860   'skip$
861   { output.state after.slash =
862     'skip$
863     { after.block 'output.state := }
864     if$
865   }
866   if$
867 }
868
869 FUNCTION {new.sentence}
870 { output.state after.block =
871   'skip$
872   { output.state before.all =
873     'skip$
874     { output.state after.slash =
875       'skip$
876       { after.sentence 'output.state := }
877       if$
878     }
879     if$
880   }
881   if$
882 }
883
884 FUNCTION {new.slash}
885 { output.state before.all =
886   'skip$
887   { component.part.label "slash" =
888     { after.slash 'output.state := }
889     { new.block
890       component.part.label "in" =
891       { entry.lang lang.en =
892         { "In: " output
893           write$
894           ""
895           before.all 'output.state :=
896         }
897         'skip$
898         if$
899       }
900       'skip$
901       if$
902     }
903     if$
904   }
905   if$
906 }
907

```

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.

```

908 FUNCTION {new.block.checka}
909 { empty$
910   'skip$
911   'new.block
912   if$
913 }
914
915 FUNCTION {new.block.checkb}
916 { empty$
917   swap$ empty$
918   and
919   'skip$
920   'new.block
921   if$
922 }
923

```

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

```

924 FUNCTION {new.sentence.checka}
925 { empty$
926   'skip$
927   'new.sentence
928   if$
929 }
930
931 FUNCTION {new.sentence.checkb}
932 { empty$
933   swap$ empty$
934   and
935   'skip$
936   'new.sentence
937   if$
938 }
939

```

In order to support UTF-8 encoding, we need some auxiliary functions. Below are a series of such functions. We try to make functions loosely-coupled as much as possible. Where the use of variables is inevitable in functions, we generally assume it is the caller's responsibility to save and restore those variables. Exceptions are made for some unary functions, where it is convenient for the callee to do so.

```

940 INTEGERS { b }
941

```

Function `is.int.in.range` takes a codepoint and two integers and check if the codepoint is between these two integers (inclusive).

```

942 % codepoint: int, a: int, b: int -> bool
943 % variable used: b
944 FUNCTION {is.int.in.range}
945 {

```



```

946 'b :=
947 #1 +
948 b >
949 { #1 - b < }
950 { pop$ #0 }
951 if$
952 }
953

```

Function `mult.power2` takes two integers and returns 2^nm .

```

954 % m: int, n: int -> int
955 FUNCTION {mult.power2}
956 {
957   { duplicate$ #0 > }
958   {
959     swap$
960     duplicate$ +
961     swap$ #1 -
962   }
963   while$
964   pop$
965 }
966

```

Function `find.match.brace` takes two strings, the first of which is assumed to be "{", and find the matching brace in the second string. It returns a token (or subtoken) and the rest of the string after the matching brace. When braces are unmatched, it issues a warning and complete the brace automatically, following the convention of the original **BIBTEX**.

```

967 % "{", str -> subtoken: str, rest: str
968 % variables used: s, t
969 FUNCTION {find.match.brace}
970 {
971   's :=
972   't :=
973
974   #1
975   { duplicate$ #0 >
976     s empty$ not and }
977   {
978     s #1 #1 substring$ "{" =
979     { #1 + }
980     {
981       s #1 #1 substring$ "}" =
982       { #1 - }
983       'skip$
984       if$
985     }
986     if$
987     t s #1 #1 substring$ * 't :=
988     s #2 global.max$ substring$ 's :=
989   }
990   while$

```

```

991
992 duplicate$ #0 >
993 {
994   "Unbalanced brace(s): one or more closing braces are missing" warning
995   { duplicate$ #0 > }
996   {
997     t "]" * 't :=
998     #1 -
999   }
1000   while$
1001   }
1002   'skip$
1003 if$
1004 pop$
1005
1006 t
1007 s
1008 }
1009

```

Function `split.first.char.from.str` takes a UTF-8 string and return the first UTF-8 character and the rest of the string in reverse order.

```

1010 % str -> str, char
1011 FUNCTION {split.first.char.from.str}
1012 {
1013   duplicate$ "" =
1014   {
1015     "split.first.char.from.str: Trying to split an empty string!" warning
1016     ""
1017   }
1018   {
1019     duplicate$ #1 #1 substring$ chr.to.int$ #128 <
1020     {
1021       duplicate$ #1 #1 substring$ swap$
1022       #2 global.max$ substring$ swap$
1023     }
1024     {
1025       duplicate$ #1 #1 substring$ chr.to.int$ #224 <
1026       {
1027         duplicate$ #1 #2 substring$ swap$
1028         #3 global.max$ substring$ swap$
1029       }
1030       {
1031         duplicate$ #1 #1 substring$ chr.to.int$ #240 <
1032         {
1033           duplicate$ #1 #3 substring$ swap$
1034           #4 global.max$ substring$ swap$
1035         }
1036         {
1037           duplicate$ #1 #4 substring$ swap$
1038           #5 global.max$ substring$ swap$
1039         }
1040       if$
1041     }
1042   }
1043 }

```

```

1042         if$
1043     }
1044     if$
1045 }
1046 if$
1047 }
1048

```

Function `get.first.char.from.str` takes a UTF-8 string and return the first UTF-8 character.

```

1049 % str -> char
1050 FUNCTION {get.first.char.from.str}
1051 {
1052     split.first.char.from.str swap$ pop$
1053 }
1054

```

Function `split.first.tex.char.from.str` is like `split.first.char.from.str`. It takes a UTF-8 string and return the first UTF-8 character or first \TeX group and the rest of string in reverse order.

```

1055 % str -> rest: str, texchar
1056 FUNCTION {split.first.tex.char.from.str}
1057 {
1058     duplicate$ #1 #1 substring$ "{" =
1059     {
1060         split.first.char.from.str swap$
1061         find.match.brace swap$
1062     }
1063     'split.first.char.from.str
1064     if$
1065 }
1066

```

Function `char.to.unicode` takes a UTF-8 character and returns its codepoint in Unicode. It issues a warning and returns -1 if the presumed character is an empty string. For other invalid input, the behavior is undefined.

```

1067 % char -> int
1068 FUNCTION {char.to.unicode}
1069 {
1070     duplicate$ #4 #1 substring$ "" =
1071     {
1072         duplicate$ #3 #1 substring$ "" =
1073         {
1074             duplicate$ #2 #1 substring$ "" =
1075             {
1076                 duplicate$ "" =
1077                 {
1078                     "Empty string is not a char!" warning$
1079                     pop$ #-1
1080                 }
1081                 { #1 #1 substring$ chr.to.int$ }
1082             if$

```

```

1083         }
1084         {
1085             duplicate$ #2 #1 substring$ chr.to.int$ #128 - swap$
1086             #1 #1 substring$ chr.to.int$ #192 -
1087             #6 mult.power2 +
1088         }
1089         if$
1090     }
1091     {
1092         duplicate$ #3 #1 substring$ chr.to.int$ #128 - swap$
1093         duplicate$ #2 #1 substring$ chr.to.int$ #128 - swap$
1094         #1 #1 substring$ chr.to.int$ #224 -
1095         #6 mult.power2 +
1096         #6 mult.power2 +
1097     }
1098     if$
1099 }
1100 {
1101     duplicate$ #4 #1 substring$ chr.to.int$ #128 - swap$
1102     duplicate$ #3 #1 substring$ chr.to.int$ #128 - swap$
1103     duplicate$ #2 #1 substring$ chr.to.int$ #128 - swap$
1104     #1 #1 substring$ chr.to.int$ #240 -
1105     #6 mult.power2 +
1106     #6 mult.power2 +
1107     #6 mult.power2 +
1108 }
1109 if$
1110 }
1111

```

Function `is.char.in.str` takes a string and a UTF-8 character. It checks whether the character is in the string. It issues a warning and returns 0 if the presumed character is an empty string. It also returns 0 if the string itself is empty. For other input, the behavior is undefined.

```

1112 % str, char -> bool
1113 % variable used: t
1114 FUNCTION {is.char.in.str}
1115 {
1116     't :=
1117     t "" =
1118     { "is.char.in.str: Empty string is not a char!" warning$ }
1119     'skip$
1120     if$
1121     #0 swap$
1122     { duplicate$ "" = not }
1123     {
1124         split.first.char.from.str t =
1125         { pop$ pop$ #1 "" }
1126         'skip$
1127         if$
1128     }
1129 }
1130

```

```

1131 while$
1132 pop$
1133 }
1134

```

Function `is.upper.ascii` takes a UTF-8 character and checks whether it is an uppercase ASCII letter.

```

1135 % char -> bool
1136 % variable used: b
1137 FUNCTION {is.upper.ascii}
1138 {
1139   char.to.unicode #65 swap$ #90 swap$ is.int.in.range
1140 }
1141

```

Function `is.upper` takes a UTF-8 character and checks whether it is uppercase in the range from U+0000 to U+017F.

```

1142 % char -> bool
1143 % variable used: b
1144 FUNCTION {is.upper}
1145 {
1146   duplicate$ is.upper.ascii
1147   { pop$ #1 }
1148   { latin.upper swap$ is.char.in.str }
1149   if$
1150 }
1151

```

Function `is.lower.ascii` takes a UTF-8 character and checks whether it is a lowercase ASCII letter.

```

1152 % char -> bool
1153 % variable used: b
1154 FUNCTION {is.lower.ascii}
1155 {
1156   char.to.unicode #97 swap$ #122 swap$ is.int.in.range
1157 }
1158

```

Function `is.lower` takes a UTF-8 character and checks whether it is lowercase in the range from U+0000 to U+017F.

```

1159 % char -> bool
1160 % variable used: b
1161 FUNCTION {is.lower}
1162 {
1163   duplicate$ is.lower.ascii
1164   { pop$ #1 }
1165   { latin.lower swap$ is.char.in.str }
1166   if$
1167 }
1168

```

Function `is.printable.ascii` takes a UTF-8 character and checks whether it is a printable ASCII character.

```
1169 % char -> bool
1170 % variable used: b
1171 FUNCTION {is.printable.ascii}
1172 {
1173   char.to.unicode #32 swap$ #126 swap$ is.int.in.range
1174 }
1175
```

Function `is.letter.ascii` takes a UTF-8 character and checks whether it is an ASCII letter.

```
1176 % char -> bool
1177 % variable used: b
1178 FUNCTION {is.letter.ascii}
1179 {
1180   duplicate$ is.upper.ascii swap$ is.lower.ascii or
1181 }
1182
```

Function `is.symbol.ascii` takes a UTF-8 character and checks whether it is a printable ASCII character but not an ASCII letter.

```
1183 % char -> bool
1184 % variable used: b
1185 FUNCTION {is.symbol.ascii}
1186 {
1187   duplicate$ is.printable.ascii swap$ is.letter.ascii not and
1188 }
1189
```

Function `is.all.lower` takes a string and checks whether every character in it is lowercase in the range from U+0000 to U+017F.

```
1190 % str -> bool
1191 % variable used: b
1192 % return true if str is empty
1193 FUNCTION {is.all.lower}
1194 {
1195   #1 swap$
1196   { duplicate$ "" = not }
1197   {
1198     split.first.char.from.str is.lower
1199     'skip$
1200     { pop$ pop$ #0 "" }
1201     if$
1202   }
1203   while$
1204   pop$
1205 }
1206
1207 % str -> bool
1208 % variable used: b
1209 FUNCTION {is.tex.str.in.title.case}
```

```

1210 {
1211     duplicate$ "" =
1212     { pop$ #0 }
1213     {
1214         split.first.tex.char.from.str purify$
1215         duplicate$ "" =
1216         { pop$ pop$ #0 }
1217         {
1218             split.first.char.from.str is.upper
1219             {
1220                 duplicate$ is.all.lower
1221                 {
1222                     empty$
1223                     {
1224                         duplicate$ "" =
1225                         { pop$ #0 }
1226                         'is.all.lower
1227                         if$
1228                         }
1229                         'is.all.lower
1230                         if$
1231                         }
1232                         { pop$ pop$ #0 }
1233                         if$
1234                         }
1235                         { pop$ pop$ #0 }
1236                         if$
1237                     }
1238                 if$
1239             }
1240         if$
1241     }
1242
1243 % char, int -> bool
1244 % variables used: t, b
1245 FUNCTION {is.in.inter.token.chars}
1246 {
1247     duplicate$ #0 =
1248     { pop$ " " = }
1249     {
1250         #1 =
1251         { " " range.delimiters * swap$ is.char.in.str }
1252         'is.letter.ascii
1253         if$
1254     }
1255     if$
1256 }
1257
1258 % str, int -> intertoken: str, rest: str
1259 % variable used: t, b
1260 FUNCTION {skip.inter.token.chars.by}
1261 {
1262     'b :=
1263     't :=
1264

```

```

1265     "" t
1266     { duplicate$ "" = not }
1267     {
1268         split.first.char.from.str
1269         duplicate$ b is.in.inter.token.chars
1270         { swap$ 't := * t }
1271         { swap$ * 't := "" }
1272         if$
1273     }
1274     while$
1275
1276     pop$ t
1277 }
1278
1279 % str -> intertoken: str, rest: str
1280 % variable used: t, b
1281 FUNCTION {skip.inter.token.chars}
1282 {
1283     #0 skip.inter.token.chars.by
1284 }
1285
1286 % str -> intertoken: str, rest: str
1287 % variable used: t, b
1288 FUNCTION {skip.inter.token.command}
1289 {
1290     duplicate$ "" =
1291     { "" }
1292     {
1293         duplicate$ #1 #1 substring$ is.symbol.ascii
1294         { split.first.char.from.str swap$ }
1295         { #2 skip.inter.token.chars.by }
1296         if$
1297     }
1298     if$
1299 }
1300
1301 % cmdstr -> cmdstr
1302 FUNCTION {is.special.char.command}
1303 {
1304     #2 global.max$ substring$ skip.inter.token.command
1305
1306     empty$
1307     'skip$
1308     { "is.special.char.command: cmdstr has extra components!" warning$ }
1309     if$
1310
1311     duplicate$ duplicate$ duplicate$ duplicate$ duplicate$ duplicate$
1312     "oOllIj" swap$ is.char.in.str
1313     swap$ "oe" = or
1314     swap$ "OE" = or
1315     swap$ "ae" = or
1316     swap$ "AE" = or
1317     swap$ "aa" = or
1318     swap$ "AA" = or
1319 }

```



```

1320
1321 % str, str, char -> char
1322 % variable used: t
1323 FUNCTION {map.char}
1324 {
1325   't :=
1326   split.first.char.from.str
1327   { swap$ duplicate$ "" = not }
1328   {
1329     swap$ t =
1330     { pop$ "" t }
1331     {
1332       swap$ split.first.char.from.str pop$ swap$
1333       split.first.char.from.str
1334     }
1335     if$
1336   }
1337   while$
1338   pop$ t =
1339   'get.first.char.from.str
1340   { pop$ t }
1341   if$
1342 }
1343
1344 % char -> char
1345 % variables used: t, b
1346 FUNCTION {to.lower}
1347 {
1348   duplicate$ is.upper.ascii
1349   { chr.to.int$ #32 + int.to.chr$ }
1350   { latin.lower swap$ latin.upper swap$ map.char }
1351   if$
1352 }
1353
1354 % char -> char
1355 % variables used: t, b
1356 FUNCTION {to.upper}
1357 {
1358   duplicate$ is.lower.ascii
1359   { chr.to.int$ #32 - int.to.chr$ }
1360   { latin.upper swap$ latin.lower swap$ map.char }
1361   if$
1362 }
1363
1364 % str -> str
1365 % variables used: t, b
1366 FUNCTION {all.to.lower}
1367 {
1368   "" swap$
1369   { duplicate$ empty$ not }
1370   { split.first.char.from.str to.lower swap$ 't := * t }
1371   while$
1372   *
1373 }
1374

```

```

1375 % texchar -> texchar
1376 % variables used: t, b
1377 FUNCTION {command.to.lower}
1378 {
1379   duplicate$ "" =
1380   { "command.to.lower: Empty string is not a texchar!" warning$ }
1381   {
1382     duplicate$ #1 #1 substring$ #92 int.to.chr$ =
1383     {
1384       duplicate$ is.special.char.command
1385       'all.to.lower
1386       'skip$
1387       if$
1388     }
1389     'to.lower
1390   if$
1391   }
1392   if$
1393 }
1394
1395 % texchar -> texchar
1396 % variables used: t, b
1397 FUNCTION {tex.to.lower}
1398 {
1399   duplicate$ #1 #2 substring$ "{" #92 int.to.chr$ * =
1400   {
1401     "" swap$
1402     { duplicate$ "" = not }
1403     {
1404       split.first.char.from.str
1405       duplicate$ #92 int.to.chr$ =
1406       {
1407         swap$ skip.inter.token.command 't := * t
1408         swap$ command.to.lower
1409       }
1410       'to.lower
1411       if$
1412       swap$ 't := * t
1413     }
1414     while$
1415     pop$
1416   }
1417   {
1418     duplicate$ #1 #1 substring$ "{" =
1419     { split.first.char.from.str swap$ find.match.brace pop$ }
1420     'command.to.lower
1421     if$
1422   }
1423   if$
1424 }
1425
1426 % str -> str
1427 % variables used: t, b
1428 FUNCTION {all.to.upper}
1429 {

```

```

1430 "" swap$
1431 { duplicate$ empty$ not }
1432 { split.first.char.from.str to.upper swap$ 't := * t }
1433 while$
1434 *
1435 }
1436
1437 % texchar -> texchar
1438 % variables used: t, b
1439 FUNCTION {command.to.upper}
1440 {
1441   duplicate$ "" =
1442   { "command.to.lower: Empty string is not a texchar!" warning$ }
1443   {
1444     duplicate$ #1 #1 substring$ #92 int.to.chr$ =
1445     {
1446       duplicate$ is.special.char.command
1447       'all.to.upper
1448       'skip$
1449       if$
1450     }
1451     'to.upper
1452     if$
1453   }
1454   if$
1455 }
1456
1457 % texchar -> texchar
1458 % variables used: t, b
1459 FUNCTION {tex.to.upper}
1460 {
1461   duplicate$ #1 #2 substring$ "{" #92 int.to.chr$ * =
1462   {
1463     "" swap$
1464     { duplicate$ "" = not }
1465     {
1466       split.first.char.from.str
1467       duplicate$ #92 int.to.chr$ =
1468       {
1469         swap$ skip.inter.token.command 't := * t
1470         swap$ command.to.upper
1471       }
1472       'to.upper
1473       if$
1474       swap$ 't := * t
1475     }
1476     while$
1477     pop$
1478   }
1479   {
1480     duplicate$ #1 #1 substring$ "{" =
1481     { split.first.char.from.str swap$ find.match.brace pop$ }
1482     'command.to.upper
1483     if$
1484   }

```

```

1485   if$
1486 }
1487
1488 % texstr -> texstr
1489 % variable used: t, b
1490 FUNCTION {lower.token.if.in.title.case}
1491 {
1492   duplicate$ is.tex.str.in.title.case
1493   { split.first.tex.char.from.str tex.to.lower swap$ * }
1494   'skip$
1495   if$
1496 }
1497
1498 % int -> str
1499 FUNCTION {self.tokens}
1500 {
1501   #0 =
1502   'default.self.tokens
1503   'range.delimiters
1504   if$
1505 }
1506
1507 % str, int -> token: str, rest: str
1508 % variables used: s, t, b
1509 FUNCTION {tokenize.by}
1510 {
1511   'b :=
1512   's :=
1513
1514   s "" =
1515   { "" "" }
1516   {
1517     s split.first.char.from.str
1518     duplicate$ b self.tokens swap$ is.char.in.str
1519     'swap$
1520     {
1521       duplicate$ #92 int.to.chr$ =
1522       { swap$ skip.inter.token.command 's := * s }
1523       {
1524         pop$ pop$ "" s
1525         { duplicate$ "" = not }
1526         {
1527           split.first.char.from.str
1528           duplicate$ "\ " b self.tokens * swap$ is.char.in.str
1529           { pop$ pop$ "" }
1530           {
1531             duplicate$ "{" =
1532             { swap$ find.match.brace }
1533             'swap$
1534             if$
1535             's := * s
1536           }
1537           if$
1538         }
1539         while$

```

```

1540             pop$ s
1541         }
1542         if$
1543     }
1544     if$
1545 }
1546 if$
1547 }
1548
1549 % str -> str
1550 % variables used: s, t, b
1551 FUNCTION {tokenize}
1552 {
1553     #0 tokenize.by
1554 }
1555
1556 % str -> str
1557 % variables used: s, t, b
1558 FUNCTION {smart.sentence.case}
1559 {
1560     tokenize 's :=
1561     { s "" = not }
1562     {
1563         s skip.inter.token.chars 's := * s
1564         tokenize swap$
1565         duplicate$ ":" =
1566         {
1567             swap$ 's := *
1568             s skip.inter.token.chars 's := * s
1569             tokenize swap$
1570             lowercase.word.after.colon
1571             {
1572                 duplicate$ "A" =
1573                 { pop$ "a" }
1574                 'lower.token.if.in.title.case
1575                 if$
1576             }
1577             'skip$
1578             if$
1579         }
1580         'lower.token.if.in.title.case
1581         if$
1582         swap$ 's := *
1583     }
1584     while$
1585 }
1586 }
1587
1588 % str -> str
1589 % variables used: s, t, b
1590 FUNCTION {smart.upper.case}
1591 {
1592     s swap$ t swap$
1593     "" swap$

```

```

1595 { duplicate$ "" = not }
1596 {
1597   tokenize swap$
1598   duplicate$ #1 #1 substring$ #92 int.to.chr$ =
1599   'command.to.upper
1600   {
1601     "" swap$
1602     { duplicate$ "" = not }
1603     {
1604       split.first.tex.char.from.str tex.to.upper
1605       swap$ 't := * t
1606     }
1607     while$
1608     pop$
1609   }
1610   if$
1611   swap$ 't := * t
1612   skip.inter.token.chars 't := * t
1613 }
1614 while$
1615 pop$
1616
1617 swap$ 't :=
1618 swap$ 's :=
1619 }
1620

```

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_U" * s * "}"

```

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

```

1621 FUNCTION {field.or.null}
1622 { duplicate$ empty$
1623   { pop$ "" }
1624   'skip$
1625   if$
1626 }
1627
1628 FUNCTION {emphasize}
1629 { duplicate$ empty$
1630   { pop$ "" }
1631   { "\emph{" swap$ * "}" * }
1632   if$
1633 }
1634
1635 FUNCTION {format.btitle}
1636 { italic.book.title
1637   entry.lang lang.en = and
1638   'emphasize
1639   'skip$
1640   if$
1641 }
1642

```

B.4.1 Detect Language

```

1643 INTEGERS { byte second.byte }
1644
1645 INTEGERS { char.lang tmp.lang }
1646
1647 STRINGS { tmp.str }
1648
1649 FUNCTION {get.str.lang}
1650 { 'tmp.str :=
1651   lang.other 'tmp.lang :=
1652   #1 'charptr :=
1653   tmp.str text.length$ #1 + 'len :=
1654   { charptr len < }
1655   { tmp.str charptr #1 substring$ chr.to.int$ 'byte :=
1656     byte #128 <
1657     { charptr #1 + 'charptr :=
1658       byte #64 > byte #91 < and byte #96 > byte #123 < and or
1659       { lang.en 'char.lang := }
1660       { lang.other 'char.lang := }
1661       if$
1662     }
1663   { tmp.str charptr #1 + #1 substring$ chr.to.int$ 'second.byte :=

```

```
1664         byte #224 <
```

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

```
1665         { charptr #2 + 'charptr :=
1666           byte #207 > byte #212 < and
1667           byte #212 = second.byte #176 < and or
1668             { lang.ru 'char.lang := }
1669             { lang.other 'char.lang := }
1670         if$
1671       }
1672     { byte #240 <
```

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

```
1673         { charptr #3 + 'charptr :=
1674           byte #227 > byte #234 < and
1675           { lang.zh 'char.lang := }
```

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

```
1676         { byte #227 =
1677           { second.byte #143 >
1678             { lang.zh 'char.lang := }
```

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

```
1679           { second.byte #128 > second.byte #132 < and
1680             { lang.ja 'char.lang := }
1681             { lang.other 'char.lang := }
1682         if$
1683       }
1684     if$
1685   }
```

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

```
1686         { byte #239 =
1687           second.byte #163 > second.byte #172 < and and
1688             { lang.zh 'char.lang := }
1689             { lang.other 'char.lang := }
1690         if$
1691       }
1692     if$
1693   }
1694   if$
1695 }
```

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.

```
1696         { charptr #4 + 'charptr :=
1697           byte #240 = second.byte #159 > and
1698             { lang.zh 'char.lang := }
1699             { lang.other 'char.lang := }
1700         if$
1701       }
1702     if$
1703   }
1704   if$
```



```

1705         }
1706         if$
1707         char.lang tmp.lang >
1708         { char.lang 'tmp.lang := }
1709         'skip$
1710     if$
1711 }
1712 while$
1713 tmp.lang
1714 }
1715
1716 FUNCTION {check.entry.lang}
1717 { author field.or.null
1718   title field.or.null *
1719   get.str.lang
1720 }
1721
1722 STRINGS { entry.langid }
1723
1724 FUNCTION {set.entry.lang}
1725 { "" 'entry.langid :=
1726   language empty$ not
1727   { language 'entry.langid := }
1728   'skip$
1729   if$
1730   langid empty$ not
1731   { langid 'entry.langid := }
1732   'skip$
1733   if$
1734   entry.langid empty$
1735   { check.entry.lang }
1736   { entry.langid "english" = entry.langid "american" = or entry.langid "b
1737     { lang.en }
1738     { entry.langid "chinese" =
1739       { lang.zh }
1740       { entry.langid "japanese" =
1741         { lang.ja }
1742         { entry.langid "russian" =
1743           { lang.ru }
1744           { check.entry.lang }
1745           if$
1746         }
1747         if$
1748       }
1749       if$
1750     }
1751     if$
1752   }
1753   if$
1754   'entry.lang :=
1755 }
1756
1757 FUNCTION {set.entry.numbered}
1758 { type$ "patent" =
1759   type$ "standard" = or

```

```

1760 type$ "techreport" = or
1761     { #1 'entry.numbered := }
1762     { #0 'entry.numbered := }
1763 if$
1764 }
1765

```

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: namerresult: 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 namerresult := namerresult * ",~" * t
            else if numnames > 2
                then namerresult := namerresult * ",~"
            fi
            if t = "others"
                then namerresult := namerresult * "~et~al."
                else namerresult := namerresult * "~and~" * t
            fi
        fi
        else namerresult := t
        fi
        nameptr := nameptr + 1
        namesleft := namesleft - 1
    od
    return namerresult
END

```

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

```

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

```

```
END
```

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

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

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

```
1766 INTEGERS { nameptr namesleft numnames name.lang }
1767
1768 FUNCTION {format.name}
1769 { "{vv~}{ll}{, jj}{, ff}" format.name$ 't :=
1770   t "others" =
1771     { bbl.et.al }
1772     { t get.str.lang 'name.lang :=
1773       name.lang lang.en =
1774         { t #1 "{vv~}{ll}{ f{~}}" format.name$
1775           uppercase.name
1776           'smart.upper.case
1777           'skip$
1778           if$
1779             t #1 "{, jj}" format.name$ *
1780           }
1781         { t #1 "{ll}{ff}" format.name$ }
1782       if$
1783     }
1784   if$
1785 }
1786
1787 FUNCTION {format.names}
1788 { 's :=
1789   #1 'nameptr :=
1790   s num.names$ 'numnames :=
1791   ""
1792   numnames 'namesleft :=
1793   { namesleft #0 > }
1794   { s nameptr format.name bbl.et.al =
1795     numnames bibliography.et.al.min #1 - > nameptr bibliography.et.al.use
1796     { ", " *
1797       bbl.et.al *
1798       #1 'namesleft :=
1799     }
```

```

1800         { nameptr #1 >
1801             { namesleft #1 = bbl.and "" = not and
1802                 { bbl.and * }
1803                 { ", " * }
1804             if$
1805         }
1806         'skip$
1807         if$
1808             s nameptr format.name *
1809         }
1810         if$
1811             nameptr #1 + 'nameptr :=
1812             namesleft #1 - 'namesleft :=
1813         }
1814     while$
1815 }
1816
1817 FUNCTION {format.key}
1818 { empty$
1819     { key field.or.null }
1820     { "" }
1821     if$
1822 }
1823
1824 FUNCTION {format.authors}
1825 { author empty$ not
1826     { author format.names }
1827     { "empty author in " cite$ * warning$
1828     <*author-year>
1829         bbl.anonymous
1830     </author-year>
1831     <*numerical>
1832         ""
1833     </numerical>
1834     }
1835     if$
1836 }
1837
1838 FUNCTION {format.editors}
1839 { editor empty$
1840     { "" }
1841     { editor format.names }
1842     if$
1843 }
1844
1845 FUNCTION {format.translators}
1846 { translator empty$
1847     { "" }
1848     { translator format.names
1849         entry.lang lang.zh =
1850         { translator num.names$ #3 >
1851             { " 译" * }
1852             { ", 译" * }
1853             if$
1854         }

```

```

1855         'skip$
1856     if$
1857     }
1858     if$
1859 }
1860
1861 FUNCTION {format.full.names}
1862 { 's :=
1863   #1 'nameptr :=
1864   s num.names$ 'numnames :=
1865   numnames 'namesleft :=
1866   { namesleft #0 > }
1867   { s nameptr "{vv~}{ll}{, jj}{, ff}" format.name$ 't :=
1868     t get.str.lang 'name.lang :=
1869     name.lang lang.en =
1870     { t #1 "{vv~}{ll}" format.name$ 't := }
1871     { t #1 "{ll}{ff}" format.name$ 't := }
1872   if$
1873   nameptr #1 >
1874   {
1875     namesleft #1 >
1876     { ", " * t * }
1877     {
1878       numnames #2 >
1879       { ", " * }
1880       'skip$
1881       if$
1882       t "others" =
1883       { " et~al." * }
1884       { " and " * t * }
1885       if$
1886     }
1887     if$
1888   }
1889   't
1890   if$
1891   nameptr #1 + 'nameptr :=
1892   namesleft #1 - 'namesleft :=
1893 }
1894 while$
1895 }
1896
1897 FUNCTION {author.editor.full}
1898 { author empty$
1899   { editor empty$
1900     { "" }
1901     { editor format.full.names }
1902   if$
1903   }
1904   { author format.full.names }
1905 if$
1906 }
1907
1908 FUNCTION {author.full}
1909 { author empty$

```

```

1910     { "" }
1911     { author format.full.names }
1912   if$
1913 }
1914
1915 FUNCTION {editor.full}
1916 { editor empty$
1917   { "" }
1918   { editor format.full.names }
1919   if$
1920 }
1921
1922 FUNCTION {make.full.names}
1923 { type$ "book" =
1924   type$ "inbook" = booktitle empty$ not and
1925   or
1926   'author.editor.full
1927   { type$ "collection" =
1928     type$ "proceedings" =
1929     or
1930     'editor.full
1931     'author.full
1932     if$
1933   }
1934   if$
1935 }
1936
1937 FUNCTION {output.bibitem}
1938 { newline$
1939   "\bibitem[" write$
1940   label "]" *
1941   make.full.names duplicate$ short.list =
1942   { pop$ }
1943   { duplicate$ "]" contains
1944     { "{" swap$ * "]" * }
1945     'skip$
1946     if$
1947     *
1948   }
1949   if$
1950   "]" * write$
1951   cite$ write$
1952   "]" write$
1953   newline$
1954   ""
1955   before.all 'output.state :=
1956 }
1957

```

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.

```

1958 FUNCTION {change.sentence.case}
1959 { entry.lang lang.en =
1960   'smart.sentence.case
1961   'skip$
1962   if$
1963 }
1964
1965 FUNCTION {add.link}
1966 { url empty$ not
1967   { "\href{" url * "}" * swap$ * "}" * }
1968   { doi empty$ not
1969     { "\href{https://doi.org/" doi * "}" * swap$ * "}" * }
1970     'skip$
1971     if$
1972   }
1973   if$
1974 }
1975
1976 FUNCTION {format.title}
1977 { title empty$
1978   { "" }
1979   { title
1980     sentence.case.title
1981     'change.sentence.case
1982     'skip$
1983     if$
1984     entry.numbered number empty$ not and
1985     { bbl.colon *
1986       type$ "patent" = show.patent.country and
1987       { address empty$ not
1988         { address * ", " * }
1989         { location empty$ not
1990           { location * ", " * }
1991           { entry.lang lang.zh =
1992             { " 中国" * ", " * }
1993             'skip$
1994             if$
1995           }
1996           if$
1997         }
1998         if$
1999       }
2000       'skip$
2001       if$
2002       number *
2003     }
2004     'skip$
2005     if$
2006     link.title
2007     'add.link
2008     'skip$
2009     if$

```

```

2010     }
2011   if$
2012 }
2013

```

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

```

```

2014 FUNCTION {tie.or.space.connect}
2015 { duplicate$ text.length$ #3 <
2016   { "~" }
2017   { " " }
2018   if$
2019   swap$ * *
2020 }
2021

```

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

```

```

2022 FUNCTION {either.or.check}
2023 { empty$
2024   'pop$
2025   { "can't use both " swap$ * " fields in " * cite$ * warning$ }
2026   if$
2027 }
2028

```

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 capitilize Number when this function is used at the beginning of a block.

```

2029 FUNCTION {is.digit}
2030 { duplicate$ empty$
2031   { pop$ #0 }
2032   { chr.to.int$
2033     duplicate$ "0" chr.to.int$ <
2034     { pop$ #0 }
2035     { "9" chr.to.int$ >
2036       { #0 }
2037       { #1 }
2038     if$
2039   }
2040   if$
2041 }
2042 if$
2043 }
2044
2045 FUNCTION {is.number}
2046 { 's :=
2047   s empty$
2048   { #0 }
2049   { s text.length$ 'charptr :=
2050     { charptr #0 >
2051       s charptr #1 substring$ is.digit
2052       and
2053     }
2054     { charptr #1 - 'charptr := }
2055     while$
2056     charptr not
2057   }
2058   if$
2059 }
2060
2061 FUNCTION {format.volume}
2062 { volume empty$ not
2063   { volume is.number
2064     { entry.lang lang.zh =
2065       { " 第 " volume * " 卷" * }
2066       { "Vol." volume tie.or.space.connect }
2067     if$
2068   }
2069   { volume }
2070   if$
2071 }
2072 { "" }
2073 if$
2074 }
2075
2076 FUNCTION {format.number}
2077 { number empty$ not
2078   { number is.number
2079     { entry.lang lang.zh =

```

```

2080             { " 第 " number * " 册 " * }
2081             { "No." number tie.or.space.connect }
2082         if$
2083     }
2084     { number }
2085     if$
2086 }
2087 { "" }
2088 if$
2089 }
2090
2091 FUNCTION {format.volume.number}
2092 { volume empty$ not
2093   { format.volume }
2094   { format.number }
2095   if$
2096 }
2097
2098 FUNCTION {format.title.vol.num}
2099 { title
2100   sentence.case.title
2101   'change.sentence.case
2102   'skip$
2103   if$
2104   entry.numbered
2105   { number empty$ not
2106     { bbl.colon * number * }
2107     'skip$
2108     if$
2109   }
2110   { format.volume.number 's :=
2111     s empty$ not
2112     { bbl.colon * s * }
2113     'skip$
2114     if$
2115   }
2116   if$
2117 }
2118
2119 FUNCTION {format.series.vol.num.title}
2120 { format.volume.number 's :=
2121   series empty$ not
2122   { series
2123     sentence.case.title
2124     'change.sentence.case
2125     'skip$
2126     if$
2127     entry.numbered
2128     { bbl.wide.space * }
2129     { bbl.colon *
2130       s empty$ not
2131       { s * bbl.wide.space * }
2132       'skip$
2133       if$
2134     }

```

```

2135         if$
2136         title *
2137         sentence.case.title
2138         'change.sentence.case
2139         'skip$
2140     if$
2141     entry.numbered number empty$ not and
2142     { bbl.colon * number * }
2143     'skip$
2144     if$
2145     }
2146     { format.title.vol.num }
2147 if$
2148 format.btitle
2149 link.title
2150     'add.link
2151     'skip$
2152 if$
2153 }
2154
2155 FUNCTION {format.booktitle.vol.num}
2156 { booktitle
2157     entry.numbered
2158     'skip$
2159     { format.volume.number 's :=
2160     s empty$ not
2161     { bbl.colon * s * }
2162     'skip$
2163     if$
2164     }
2165     if$
2166 }
2167
2168 FUNCTION {format.series.vol.num.booktitle}
2169 { format.volume.number 's :=
2170     series empty$ not
2171     { series bbl.colon *
2172     entry.numbered not s empty$ not and
2173     { s * bbl.wide.space * }
2174     'skip$
2175     if$
2176     booktitle *
2177     }
2178     { format.booktitle.vol.num }
2179     if$
2180     format.btitle
2181 }
2182
2183 FUNCTION {remove.period}
2184 { 't :=
2185     "" 's :=
2186     { t empty$ not }
2187     { t #1 #1 substring$ 'tmp.str :=
2188     tmp.str "." = not
2189     { s tmp.str * 's := }

```

```

2190         'skip$
2191     if$
2192         t #2 global.max$ substring$ 't :=
2193     }
2194     while$
2195     s
2196 }
2197
2198 FUNCTION {abbreviate}
2199 { remove.period
2200     't :=
2201     t "l" change.case$ 's :=
2202     ""
2203     s "physical review letters" =
2204     { "Phys Rev Lett" }
2205     'skip$
2206     if$
2207     's :=
2208     s empty$
2209     { t }
2210     { pop$ s }
2211     if$
2212 }
2213
2214 FUNCTION {get.journal.title}
2215 { short.journal
2216     { shortjournal empty$ not
2217         { shortjournal }
2218         { journal empty$ not
2219             { journal abbreviate }
2220             { journaltitle empty$ not
2221                 { journaltitle abbreviate }
2222                 { "" }
2223             }
2224             if$
2225         }
2226         if$
2227     }
2228     { journal empty$ not
2229         { journal }
2230         { journaltitle empty$ not
2231             { journaltitle }
2232             { shortjournal empty$ not
2233                 { shortjournal }
2234                 { "" }
2235             }
2236             if$
2237         }
2238         if$
2239     }
2240     if$
2241 }
2242 if$
2243 }
2244

```

```

2245 FUNCTION {check.arxiv.preprint}
2246 { "l" change.case$
2247   duplicate$
2248   "arxiv:" 'y :=
2249   'x :=
2250   y text.length$ 'len :=
2251   x text.length$ len - #1 + 'charptr :=
2252   { charptr #0 >
2253     x charptr len substring$ y = not
2254     and
2255   }
2256   { charptr #1 - 'charptr := }
2257   while$
2258   charptr #0 >
2259   { x charptr #6 + global.max$ substring$ 'x :=
2260     x text.length$ #1 + 'len :=
2261     #1 'charptr :=
2262     { charptr len <
2263       x charptr #1 substring$ " " = not and
2264       x charptr #1 substring$ "[" = not and
2265     }
2266     { charptr #1 + 'charptr := }
2267     while$
2268     x #1 charptr substring$
2269     duplicate$ empty$
2270     { pop$ }
2271     { "https://arxiv.org/abs/" swap$ * 'entry.url :=
2272       #1 'entry.is.electronic :=
2273       #1 'is.pure.electronic :=
2274     }
2275     if$
2276   }
2277   'skip$
2278   if$
2279   purify$ #1 #5 substring$ "arxiv" =
2280   { #1 }
2281   { #0 }
2282   if$
2283 }
2284
2285 FUNCTION {format.journal}
2286 { get.journal.title
2287   duplicate$ empty$ not
2288   { italic.journal entry.lang lang.en = and
2289     'emphasize
2290     'skip$
2291     if$
2292     link.journal
2293     'add.link
2294     'skip$
2295     if$
2296   }
2297   'skip$
2298   if$
2299 }

```

2300

B.4.4 Format entry type mark

```
2301 FUNCTION {set.entry.mark}
2302 { entry.mark empty$ not
2303   'pop$
2304   { mark empty$ not
2305     { pop$ mark 'entry.mark := }
2306     { 'entry.mark := }
2307     if$
2308   }
2309   if$
2310 }
2311
2312 FUNCTION {format.mark}
2313 { show.mark
2314   { entry.mark
2315     show.medium.type
2316     { medium empty$ not
2317       { "/" * medium * }
2318       { entry.is.electronic
2319         { "/OL" * }
2320         'skip$
2321         if$
2322       }
2323       if$
2324     }
2325     'skip$
2326     if$
2327     'entry.mark :=
2328     space.before.mark
2329     { " " }
2330     { "\allowbreak" }
2331     if$
2332     "[" * entry.mark * "]" *
2333   }
2334   { " " }
2335   if$
2336 }
2337
```

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.

```
2338 FUNCTION {num.to.ordinal}
2339 { duplicate$ text.length$ 'charptr :=
2340   duplicate$ charptr #1 substring$ 's :=
2341   s "1" =
2342     { "st" * }
2343     { s "2" =
2344       { "nd" * }
2345       { s "3" =
```

```

2346         { "rd" * }
2347         { "th" * }
2348     if$
2349 }
2350 if$
2351 }
2352 if$
2353 }
2354
2355 FUNCTION {format.edition}
2356 { edition empty$
2357   { "" }
2358   { edition is.number
2359     { edition "1" = not
2360       { entry.lang lang.zh =
2361         { edition " 版" * }
2362         { edition num.to.ordinal " ed." * }
2363         if$
2364       }
2365       { "" }
2366       if$
2367     }
2368     { entry.lang lang.en =
2369       { edition change.sentence.case 's :=
2370         s "Revised" = s "Revised edition" = or
2371         { "Rev. ed." }
2372         { s " ed." * }
2373         if$
2374       }
2375       { edition }
2376       if$
2377     }
2378   if$
2379 }
2380 if$
2381 }
2382

```

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。

```

2383 FUNCTION {format.publisher}
2384 { publisher empty$ not
2385   { publisher }
2386   { school empty$ not
2387     { school }
2388     { organization empty$ not
2389       { organization }
2390       { institution empty$ not
2391         { institution }
2392         { "" }
2393         if$
2394       }
2395     if$

```

```

2396     }
2397     if$
2398   }
2399   if$
2400 }
2401
2402 FUNCTION {format.address.publisher}
2403 { address empty$ not
2404   { address }
2405   { location empty$ not
2406     { location }
2407     { "" }
2408     if$
2409   }
2410   if$
2411   duplicate$ empty$ not
2412   { format.publisher empty$ not
2413     { bbl.colon * format.publisher * }
2414     { entry.is.electronic not show.missing.address.publisher and
2415       { bbl.colon * bbl.sine.nomine * }
2416       'skip$
2417       if$
2418     }
2419     if$
2420   }
2421   { pop$
2422     entry.is.electronic not show.missing.address.publisher and
2423     { format.publisher empty$ not
2424       { bbl.sine.loco bbl.colon * format.publisher * }
2425       { bbl.sine.loco.sine.nomine }
2426       if$
2427     }
2428     { format.publisher empty$ not
2429       { format.publisher }
2430       { "" }
2431       if$
2432     }
2433     if$
2434   }
2435   if$
2436 }
2437

```

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。

```

2438 FUNCTION {extract.before.dash}
2439 { duplicate$ empty$
2440   { pop$ "" }

```



```

2441     { 's :=
2442       #1 'charptr :=
2443       s text.length$ #1 + 'len :=
2444         { charptr len <
2445           s charptr #1 substring$ "-" = not
2446           and
2447         }
2448         { charptr #1 + 'charptr := }
2449       while$
2450       s #1 charptr #1 - substring$
2451     }
2452   if$
2453 }
2454
2455 FUNCTION {extract.after.dash}
2456 { duplicate$ empty$
2457   { pop$ "" }
2458   { 's :=
2459     #1 'charptr :=
2460     s text.length$ #1 + 'len :=
2461     { charptr len <
2462       s charptr #1 substring$ "-" = not
2463       and
2464     }
2465     { charptr #1 + 'charptr := }
2466   while$
2467   { charptr len <
2468     s charptr #1 substring$ "-" =
2469     and
2470   }
2471   { charptr #1 + 'charptr := }
2472   while$
2473   s charptr global.max$ substring$
2474 }
2475 if$
2476 }
2477
2478 FUNCTION {extract.before.slash}
2479 { duplicate$ empty$
2480   { pop$ "" }
2481   { 's :=
2482     #1 'charptr :=
2483     s text.length$ #1 + 'len :=
2484     { charptr len <
2485       s charptr #1 substring$ "/" = not
2486       and
2487     }
2488     { charptr #1 + 'charptr := }
2489   while$
2490   s #1 charptr #1 - substring$
2491 }
2492 if$
2493 }
2494
2495 FUNCTION {extract.after.slash}

```

```

2496 { duplicate$ empty$
2497   { pop$ "" }
2498   { 's :=
2499     #1 'charptr :=
2500     s text.length$ #1 + 'len :=
2501     { charptr len <
2502       s charptr #1 substring$ "-" = not
2503       and
2504       s charptr #1 substring$ "/" = not
2505       and
2506     }
2507     { charptr #1 + 'charptr := }
2508   while$
2509     { charptr len <
2510       s charptr #1 substring$ "-" =
2511       s charptr #1 substring$ "/" =
2512       or
2513       and
2514     }
2515     { charptr #1 + 'charptr := }
2516   while$
2517   s charptr global.max$ substring$
2518 }
2519 if$
2520 }
2521

```

著者-出版年制必须提取出年份

```

2522 FUNCTION {format.year}
2523 { year empty$ not
2524   { year extra.label * }
2525   { date empty$ not
2526     { date extract.before.dash extra.label * }
2527     { entry.is.electronic not
2528       { "empty year in " cite$ * warning$ }
2529       'skip$
2530     }
2531     if$
2532     urldate empty$ not
2533     { "[" urldate extract.before.dash * extra.label * "]" * }
2534     { "" }
2535     if$
2536   }
2537   if$
2538 }
2539 }
2540
2541 FUNCTION {format.periodical.year}
2542 { year empty$ not
2543   { year extract.before.slash
2544     "__" *
2545     year extract.after.slash
2546     duplicate$ empty$
2547     'pop$
2548     { * }

```

```

2549         if$
2550     }
2551     { date empty$ not
2552         { date extract.before.dash }
2553         { "empty year in " cite$ * warning$
2554           urldate empty$ not
2555           { "[" urldate extract.before.dash * "]" * }
2556           { "" }
2557         if$
2558     }
2559     if$
2560 }
2561 if$
2562 }
2563

```

专利和报纸都是使用日期而不是年

```

2564 FUNCTION {format.date}
2565 { date empty$ not
2566   { type$ "patent" = type$ "newspaper" = or
2567     { date }
2568     { entrysubtype empty$ not
2569       { type$ "article" = entrysubtype "newspaper" = and
2570         { date }
2571         { format.year }
2572       if$
2573     }
2574     { format.year }
2575   if$
2576 }
2577 if$
2578 }
2579 { year empty$ not
2580   { format.year }
2581   { "" }
2582 if$
2583 }
2584 if$
2585 }
2586

```

更新、修改日期只用于电子资源 **electronic**

```

2587 FUNCTION {format.editdate}
2588 { date empty$ not
2589   { "\allowbreak(" date * ")" * }
2590   { "" }
2591 if$
2592 }
2593

```

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。

```

2594 FUNCTION {format.urldate}
2595 { show.urldate show.url and entry.url empty$ not and

```

```

2596 is.pure.electronic or
2597 urldate empty$ not and
2598 { "\allowbreak[" urldate * "]" * }
2599 { "" }
2600 if$
2601 }
2602

```

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`。

```

2603 % str -> str
2604 % variable used: s, t, b
2605 FUNCTION {normalize.page.range}

```

```

2606 {
2607     "" swap$
2608     { duplicate$ empty$ not }
2609     {
2610         #1 skip.inter.token.chars.by 't :=
2611         empty$
2612         { "" }
2613         'page.range.delimiter
2614         if$
2615         * t
2616         #1 tokenize.by 't :=
2617         * t
2618     }
2619     while$
2620     pop$
2621 }
2622

```

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

```

2623 FUNCTION {format.pages}
2624 {
2625     pages normalize.page.range
2626 }
2627
2628 FUNCTION {format.extracted.pages}
2629 { pages empty$
2630     { "" }
2631     { pages
2632         only.start.page
2633         { #1 tokenize.by pop$ }
2634         'normalize.page.range
2635         if$
2636     }
2637     if$
2638 }
2639

```

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。

```

2640 FUNCTION {format.journal.volume}
2641 { volume empty$ not
2642     { bold.journal.volume
2643         { "\textbf{" volume * "}" * }
2644         { volume }
2645         if$
2646     }
2647     { "" }
2648     if$
2649 }
2650

```

```

2651 FUNCTION {format.journal.number}
2652 { number empty$ not
2653   { "\allowbreak (" number * ")" * }
2654   { "" }
2655   if$
2656 }
2657
2658 FUNCTION {format.journal.pages}
2659 { pages empty$
2660   { "" }
2661   { format.extracted.pages }
2662   if$
2663 }
2664

```

连续出版物的年卷期有起止范围，需要特殊处理

```

2665 FUNCTION {format.periodical.year.volume.number}
2666 { year empty$ not
2667   { year extract.before.slash }
2668   { "empty year in periodical " cite$ * warning$ }
2669   if$
2670   volume empty$ not
2671     { ", " * volume extract.before.dash * }
2672     'skip$
2673   if$
2674   number empty$ not
2675     { "\allowbreak (" * number extract.before.dash * ")" * }
2676     'skip$
2677   if$
2678   "--" *
2679   year extract.after.slash empty$
2680   volume extract.after.dash empty$ and
2681   number extract.after.dash empty$ and not
2682     { year extract.after.slash empty$ not
2683       { year extract.after.slash * }
2684       { year extract.before.slash * }
2685       if$
2686       volume empty$ not
2687         { ", " * volume extract.after.dash * }
2688         'skip$
2689       if$
2690       number empty$ not
2691         { "\allowbreak (" * number extract.after.dash * ")" * }
2692         'skip$
2693       if$
2694     }
2695     'skip$
2696   if$
2697 }
2698

```

B.4.9 Format url and doi

传统的 \LaTeX 习惯使用 `howpublished` 著录 url，这里提供支持。

```

2699 FUNCTION {check.url}
2700 { url empty$ not
2701   { url 'entry.url :=
2702     #1 'entry.is.electronic :=
2703   }
2704   { howpublished empty$ not
2705     { howpublished #1 #5 substring$ "\url{" =
2706       { howpublished 'entry.url :=
2707         #1 'entry.is.electronic :=
2708       }
2709       'skip$
2710     if$
2711   }
2712   { note empty$ not
2713     { note #1 #5 substring$ "\url{" =
2714       { note 'entry.url :=
2715         #1 'entry.is.electronic :=
2716       }
2717       'skip$
2718     if$
2719   }
2720   'skip$
2721 if$
2722 }
2723 if$
2724 }
2725 if$
2726 }
2727
2728 FUNCTION {output.url}
2729 { show.url is.pure.electronic or
2730   entry.url empty$ not and
2731   { new.block
2732     entry.url #1 #5 substring$ "\url{" =
2733     { entry.url }
2734     { "\url{" entry.url * "}" * }
2735   if$
2736   output
2737 }
2738 'skip$
2739 if$
2740 }
2741
2742 FUNCTION {check.doi}
2743 { doi empty$ not
2744   { #1 'entry.is.electronic := }
2745   'skip$
2746 if$
2747 }
2748
2749 FUNCTION {is.in.url}
2750 { 's :=
2751   s empty$

```

需要检测 DOI 是否已经包含在 URL 中。

```

2752     { #1 }
2753     { entry.url empty$
2754       { #0 }
2755         { s text.length$ 'len :=
2756           entry.url "1" change.case$ text.length$ 'charptr :=
2757             { entry.url "1" change.case$ charptr len substring$ s "1" chang
2758               charptr #0 >
2759               and
2760             }
2761             { charptr #1 - 'charptr := }
2762           while$
2763             charptr
2764         }
2765       if$
2766     }
2767   if$
2768 }
2769
2770 FUNCTION {format.doi}
2771 { ""
2772   doi empty$ not
2773   { "" 's :=
2774     doi 't :=
2775     #0 'numnames :=
2776     { t empty$ not}
2777     { t #1 #1 substring$ 'tmp.str :=
2778       tmp.str "," = tmp.str " " = or t #2 #1 substring$ empty$ or
2779       { t #2 #1 substring$ empty$
2780         { s tmp.str * 's := }
2781         'skip$
2782       if$
2783       s empty$ s is.in.url or
2784       'skip$
2785       { numnames #1 + 'numnames :=
2786         numnames #1 >
2787         { ", " * }
2788         { "DOI: " * }
2789         if$
2790         "\doi{" s * "}" * *
2791       }
2792       if$
2793       "" 's :=
2794     }
2795     { s tmp.str * 's := }
2796   if$
2797   t #2 global.max$ substring$ 't :=
2798 }
2799 while$
2800 }
2801 'skip$
2802 if$
2803 }
2804
2805 FUNCTION {output.doi}
2806 { doi empty$ not show.doi and

```



```

2807     show.english.translation entry.lang lang.zh = and not and
2808     { new.block
2809       format.doi output
2810     }
2811     'skip$
2812   if$
2813 }
2814
2815 FUNCTION {check.electronic}
2816 { "" 'entry.url :=
2817   #0 'entry.is.electronic :=
2818     'check.doi
2819     'skip$
2820   if$
2821     'check.url
2822     'skip$
2823   if$
2824   medium empty$ not
2825     { medium "MT" = medium "DK" = or medium "CD" = or medium "OL" = or
2826       { #1 'entry.is.electronic := }
2827       'skip$
2828     if$
2829     }
2830   'skip$
2831   if$
2832 }
2833
2834 FUNCTION {format.eprinttype}
2835 { archivePrefix empty$ not
2836   { archivePrefix }
2837   { eprinttype empty$ not
2838     { eprinttype }
2839     { type$ "article" = get.journal.title check.arxiv.preprint and
2840       { "arXiv" }
2841       { "" }
2842     if$
2843     }
2844   if$
2845   }
2846   if$
2847 }
2848
2849 FUNCTION {format.note}
2850 { note empty$ not show.note and
2851   { note }
2852   { "" }
2853   if$
2854 }
2855
2856 FUNCTION {output.translation}
2857 { show.english.translation entry.lang lang.zh = and
2858   { translation empty$ not
2859     { translation }
2860     { "[English translation missing!]" }
2861     if$

```

```

2862     " (in Chinese)" * output
2863     write$
2864     format.doi duplicate$ empty$ not
2865         { newline$
2866             write$
2867         }
2868     'pop$
2869     if$
2870     " \\" write$
2871     newline$
2872     "(" write$
2873     ""
2874     before.all 'output.state :=
2875     }
2876     'skip$
2877     if$
2878 }
2879

```

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

```

2880 FUNCTION {empty.misc.check}
2881 { author empty$ title empty$
2882   year empty$
2883   and and
2884   key empty$ not and
2885   { "all relevant fields are empty in " cite$ * warning$ }
2886   'skip$
2887   if$
2888 }
2889

```

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

```

2890 FUNCTION {monograph}
2891 { output.bibitem
2892   output.translation
2893   author empty$ not
2894     { format.authors }
2895     { editor empty$ not
2896       { format.editors }
2897       { "empty author and editor in " cite$ * warning$

```

```

2898 <*author-year>
2899         bbl.anonymous
2900 </author-year>
2901 <*numerical>
2902         " "
2903 </numerical>
2904     }
2905     if$
2906 }
2907 if$
2908 output
2909 year.after.author
2910 { period.after.author
2911     'new.sentence
2912     'skip$
2913     if$
2914     format.year "year" output.check
2915 }
2916 'skip$
2917 if$
2918 new.block
2919 format.series.vol.num.title "title" output.check
2920 "M" set.entry.mark
2921 format.mark "" output.after
2922 new.block
2923 format.translators output
2924 new.sentence
2925 format.edition output
2926 new.block
2927 format.address.publisher output
2928 year.after.author not
2929 { format.year "year" output.check }
2930 'skip$
2931 if$
2932 format.pages bbl.pages.colon output.after
2933 format.urldate "" output.after
2934 output.url
2935 output.doi
2936 new.block
2937 format.note output
2938 fin.entry
2939 }
2940

```

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

```

2941 FUNCTION {incollection}
2942 { output.bibitem

```

```

2943 output.translation
2944 format.authors output
2945 author format.key output
2946 year.after.author
2947     { period.after.author
2948         'new.sentence
2949         'skip$
2950         if$
2951             format.year "year" output.check
2952     }
2953     'skip$
2954 if$
2955 new.block
2956 format.title "title" output.check
2957 "M" set.entry.mark
2958 format.mark "" output.after
2959 new.block
2960 format.translators output
2961 new.slash
2962 format.editors output
2963 new.block
2964 format.series.vol.num.booktitle "booktitle" output.check
2965 new.block
2966 format.edition output
2967 new.block
2968 format.address.publisher output
2969 year.after.author not
2970     { format.year "year" output.check }
2971     'skip$
2972 if$
2973 format.extracted.pages bbl.pages.colon output.after
2974 format.urldate "" output.after
2975 output.url
2976 output.doi
2977 new.block
2978 format.note output
2979 fin.entry
2980 }
2981

```

B.5.3 连续出版物

```

2982 FUNCTION {periodical}
2983 { output.bibitem
2984   output.translation
2985   format.authors output
2986   author format.key output
2987   year.after.author
2988       { period.after.author
2989           'new.sentence
2990           'skip$
2991           if$
2992               format.year "year" output.check
2993       }
2994   'skip$

```

```

2995     if$
2996     new.block
2997     format.title "title" output.check
2998     "J" set.entry.mark
2999     format.mark "" output.after
3000     new.block
3001     format.periodical.year.volume.number output
3002     new.block
3003     format.address.publisher output
3004     year.after.author not
3005         { format.periodical.year "year" output.check }
3006         'skip$
3007     if$
3008     format.urldate "" output.after
3009     output.url
3010     output.doi
3011     new.block
3012     format.note output
3013     fin.entry
3014 }
3015

```

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.

```

3016 FUNCTION {journal.article}
3017 { output.bibitem
3018   output.translation
3019   format.authors output
3020   author format.key output
3021   year.after.author
3022       { period.after.author
3023         'new.sentence
3024         'skip$
3025         if$
3026         format.year "year" output.check
3027       }
3028       'skip$
3029   if$
3030   new.block
3031   title.in.journal
3032       { format.title "title" output.check
3033         entrysubtype empty$ not
3034         {
3035             entrysubtype "newspaper" =
3036                 { "N" set.entry.mark }
3037                 { "J" set.entry.mark }

```

```

3038         if$
3039     }
3040     { "J" set.entry.mark }
3041     if$
3042     format.mark "" output.after
3043     new.block
3044 }
3045 'skip$
3046 if$
3047 format.journal "journal" output.check
3048 year.after.author not
3049 { format.date "year" output.check }
3050 'skip$
3051 if$
3052 format.journal.volume output
3053 format.journal.number "" output.after
3054 format.journal.pages bbl.pages.colon output.after
3055 format.urldate "" output.after
3056 output.url
3057 output.doi
3058 new.block
3059 format.note output
3060 fin.entry
3061 }
3062

```

B.5.5 专利文献

`number` 域也可以用来表示专利号。

```

3063 FUNCTION {patent}
3064 { output.bibitem
3065   output.translation
3066   format.authors output
3067   author format.key output
3068   year.after.author
3069   { period.after.author
3070     'new.sentence
3071     'skip$
3072     if$
3073     format.year "year" output.check
3074   }
3075   'skip$
3076   if$
3077   new.block
3078   format.title "title" output.check
3079   "P" set.entry.mark
3080   format.mark "" output.after
3081   new.block
3082   format.date "year" output.check
3083   format.urldate "" output.after
3084   output.url
3085   output.doi
3086   new.block
3087   format.note output

```

```

3088     fin.entry
3089 }
3090

```

B.5.6 电子资源

```

3091 FUNCTION {electronic}
3092 { #1 #1 check.electronic
3093   #1 'entry.is.electronic :=
3094   #1 'is.pure.electronic :=
3095   output.bibitem
3096   output.translation
3097   format.authors output
3098   author format.key output
3099   year.after.author
3100   { period.after.author
3101     'new.sentence
3102     'skip$
3103     if$
3104     format.year "year" output.check
3105   }
3106   'skip$
3107   if$
3108   new.block
3109   format.series.vol.num.title "title" output.check
3110   "EB" set.entry.mark
3111   format.mark "" output.after
3112   new.block
3113   format.address.publisher output
3114   year.after.author not
3115   { date empty$
3116     { format.date output }
3117     'skip$
3118     if$
3119   }
3120   'skip$
3121   if$
3122   format.pages bbl.pages.colon output.after
3123   format.editdate "" output.after
3124   format.urldate "" output.after
3125   output.url
3126   output.doi
3127   new.block
3128   format.note output
3129   fin.entry
3130 }
3131

```

B.5.7 预印本

```

3132 FUNCTION {preprint}
3133 { url empty$ not
3134   { url 'entry.url :=
3135     #1 'entry.is.electronic :=
3136     #1 'is.pure.electronic :=
3137   }

```

```

3138     { eprint empty$
3139       'skip$
3140       { archivePrefix empty$
3141         { eprinttype field.or.null }
3142         { archivePrefix }
3143         if$
3144         "1" change.case$ "arxiv" =
3145         { "https://arxiv.org/abs/" eprint * 'entry.url :=
3146           #1 'entry.is.electronic :=
3147           #1 'is.pure.electronic :=
3148         }
3149         'skip$
3150       }
3151     }
3152   if$
3153 }
3154 if$
3155 output.bibitem
3156 output.translation
3157 author empty$ not
3158 { format.authors }
3159 { editor empty$ not
3160   { format.editors }
3161   { "empty author and editor in " cite$ * warning$
3162 <*author-year>
3163   bbl.anonymous
3164 </author-year>
3165 <*numerical>
3166   ""
3167 </numerical>
3168   }
3169   if$
3170   }
3171   if$
3172   output
3173   year.after.author
3174   { period.after.author
3175     'new.sentence
3176     'skip$
3177     if$
3178     format.year "year" output.check
3179   }
3180   'skip$
3181   if$
3182   new.block
3183   title.in.journal
3184   { format.series.vol.num.title "title" output.check
3185 <*2015>
3186     "A" set.entry.mark
3187 </2015>
3188 <!*2015>
3189     "Z" set.entry.mark
3190 </!2015>
3191     format.mark "" output.after
3192     new.block

```



```

3193     }
3194     'skip$
3195     if$
3196     format.edition output
3197     new.block
3198     format.eprinttype output
3199     date empty$ not
3200     { "(" date * ")" * }
3201     { year empty$ not
3202       { "(" year * ")" * }
3203       { "" }
3204     if$
3205   }
3206   if$
3207   " " output.after
3208   format.urldate "" output.after
3209   output.url
3210   output.doi
3211   new.block
3212   format.note output
3213   fin.entry
3214 }
3215

```

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。

```

3216 FUNCTION {misc}
3217 { get.journal.title
3218   duplicate$ empty$ not
3219   { check.arxiv.preprint
3220     'preprint
3221     'journal.article
3222   if$
3223   }
3224   { pop$
3225     booktitle empty$ not
3226     'incollection
3227     { archivePrefix empty$ not
3228       eprinttype empty$ not or
3229       'preprint
3230       { publisher empty$ not
3231         'monograph
3232         { entry.is.electronic
3233           'electronic
3234         {
3235           <!*2005>
3236               "Z" set.entry.mark
3237           </!*2005>
3238           <*2005>

```

```

3239                                     "M" set.entry.mark
3240 </2005>
3241                                     monograph
3242                                     }
3243                                     if$
3244                                     }
3245                                     if$
3246                                     }
3247                                     if$
3248                                     }
3249                                     if$
3250                                     }
3251                                     if$
3252                                     empty.misc.check
3253 }
3254
3255 FUNCTION {archive}
3256 { "A" set.entry.mark
3257   misc
3258 }
3259
3260 FUNCTION {article} { misc }
3261

```

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

```

3262 FUNCTION {book} { monograph }
3263

```

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

Required: title

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

```

3264 FUNCTION {booklet} { book }
3265
3266 FUNCTION {collection}
3267 { "G" set.entry.mark
3268   monograph
3269 }
3270
3271 FUNCTION {database}
3272 { "DB" set.entry.mark
3273   electronic
3274 }
3275
3276 FUNCTION {dataset}
3277 { "DS" set.entry.mark
3278   electronic
3279 }
3280

```

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

原生 BibTeX 的数据模型中 @inbook 不含 booktitle，按照“专著”处理。而 biblatex 的 @inbook 跟 incollection 一样。按照“专著的析出文献”处理。

```
3281 FUNCTION {inbook} {  
3282   booktitle empty$  
3283   'book  
3284   'incollection  
3285   if$  
3286 }  
3287
```

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

```
3288 FUNCTION {inproceedings}  
3289 { "C" set.entry.mark  
3290   incollection  
3291 }  
3292
```

The conference function is included for Scribe compatibility.

```
3293 FUNCTION {conference} { inproceedings }  
3294  
3295 FUNCTION {legislation} { archive }  
3296  
3297  
3298 FUNCTION {map}  
3299 { "CM" set.entry.mark  
3300   misc  
3301 }  
3302
```

A manual is technical documentation.

Required: title

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

```
3303 FUNCTION {manual} { monograph }  
3304
```

A mastersthesis is a Master's thesis.

Required: author, title, school, year

Optional: type, address, month, note

```
3305 FUNCTION {mastersthesis}  
3306 { "D" set.entry.mark  
3307   monograph  
3308 }  
3309
```

```

3310 FUNCTION {newspaper}
3311 { "N" set.entry.mark
3312   article
3313 }
3314
3315 FUNCTION {online}
3316 { "EB" set.entry.mark
3317   electronic
3318 }
3319

```

A phdthesis is like a mastersthesis.

Required: author, title, school, year

Optional: type, address, month, note

```

3320 FUNCTION {phdthesis} { mastersthesis }
3321
3322 FUNCTION {thesis} { mastersthesis }
3323

```

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

```

3324 FUNCTION {proceedings}
3325 { "C" set.entry.mark
3326   monograph
3327 }
3328
3329 FUNCTION {software}
3330 { "CP" set.entry.mark
3331   electronic
3332 }
3333
3334 FUNCTION {standard}
3335 { "S" set.entry.mark
3336   misc
3337 }
3338

```

A techreport is a technical report.

Required: author, title, institution, year

Optional: type, number, address, month, note

```

3339 FUNCTION {techreport}
3340 { "R" set.entry.mark
3341   misc
3342 }
3343

```

An unpublished is something that hasn't been published.

Required: author, title, note

Optional: month, year

```
3344 FUNCTION {unpublished} { misc }  
3345
```

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

```
3346 FUNCTION {default.type} { misc }  
3347
```

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

```
3348 MACRO {jan} {"January"}  
3349  
3350 MACRO {feb} {"February"}  
3351  
3352 MACRO {mar} {"March"}  
3353  
3354 MACRO {apr} {"April"}  
3355  
3356 MACRO {may} {"May"}  
3357  
3358 MACRO {jun} {"June"}  
3359  
3360 MACRO {jul} {"July"}  
3361  
3362 MACRO {aug} {"August"}  
3363  
3364 MACRO {sep} {"September"}  
3365  
3366 MACRO {oct} {"October"}  
3367  
3368 MACRO {nov} {"November"}  
3369  
3370 MACRO {dec} {"December"}  
3371
```

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

```
3372 MACRO {acmcs} {"ACM Computing Surveys"}  
3373  
3374 MACRO {acta} {"Acta Informatica"}  
3375
```

```

3376 MACRO {cacm} {"Communications of the ACM"}
3377
3378 MACRO {ibmjrd} {"IBM Journal of Research and Development"}
3379
3380 MACRO {ibmsj} {"IBM Systems Journal"}
3381
3382 MACRO {ieeese} {"IEEE Transactions on Software Engineering"}
3383
3384 MACRO {ieeetc} {"IEEE Transactions on Computers"}
3385
3386 MACRO {ieeetcad}
3387 {"IEEE Transactions on Computer-Aided Design of Integrated Circuits"}
3388
3389 MACRO {ipl} {"Information Processing Letters"}
3390
3391 MACRO {jacm} {"Journal of the ACM"}
3392
3393 MACRO {jcss} {"Journal of Computer and System Sciences"}
3394
3395 MACRO {scp} {"Science of Computer Programming"}
3396
3397 MACRO {sicomp} {"SIAM Journal on Computing"}
3398
3399 MACRO {tocs} {"ACM Transactions on Computer Systems"}
3400
3401 MACRO {tods} {"ACM Transactions on Database Systems"}
3402
3403 MACRO {tog} {"ACM Transactions on Graphics"}
3404
3405 MACRO {toms} {"ACM Transactions on Mathematical Software"}
3406
3407 MACRO {toois} {"ACM Transactions on Office Information Systems"}
3408
3409 MACRO {toplas} {"ACM Transactions on Programming Languages and Systems"}
3410
3411 MACRO {tcs} {"Theoretical Computer Science"}
3412

```

B.7 Format labels

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

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

```

3413 FUNCTION {sortify}
3414 { purify$
3415   "l" change.case$
3416 }
3417

```

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

```

3418 FUNCTION {chop.word}
3419 { 's :=
3420   'len :=
3421   s #1 len substring$ =
3422   { s len #1 + global.max$ substring$ }
3423   's
3424   if$
3425 }
3426

```

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
  fi
  return nameresult
END

```

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

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

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

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

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

```
3427 FUNCTION {format.lab.name}
3428 { "{vv~}{ll}{, jj}{, ff}" format.name$ 't :=
3429   t "others" =
3430   { citation.et.al }
3431   { t get.str.lang 'name.lang :=
3432     name.lang lang.zh = name.lang lang.ja = or
3433     { t #1 "{ll}{ff}" format.name$ }
3434     { t #1 "{vv~}{ll}" format.name$ }
```



```

3435         if$
3436     }
3437     if$
3438 }
3439
3440 FUNCTION {format.lab.names}
3441 { 's :=
3442   #1 'nameptr :=
3443   s num.names$ 'numnames :=
3444   ""
3445   numnames 'namesleft :=
3446   { namesleft #0 > }
3447   { s nameptr format.lab.name citation.et.al =
3448     numnames citation.et.al.min #1 - > nameptr citation.et.al.use.first >
3449     { bbl.space *
3450       citation.et.al *
3451       #1 'namesleft :=
3452     }
3453     { nameptr #1 >
3454       { namesleft #1 = citation.and "" = not and
3455         { citation.and * }
3456         { ", " * }
3457         if$
3458       }
3459       'skip$
3460       if$
3461       s nameptr format.lab.name *
3462     }
3463     if$
3464     nameptr #1 + 'nameptr :=
3465     namesleft #1 - 'namesleft :=
3466   }
3467   while$
3468 }
3469
3470 FUNCTION {author.key.label}
3471 { author empty$
3472   { key empty$
3473     { cite$ #1 #3 substring$ }
3474     'key
3475     if$
3476   }
3477   { author format.lab.names }
3478   if$
3479 }
3480
3481 FUNCTION {author.editor.key.label}
3482 { author empty$
3483   { editor empty$
3484     { key empty$
3485       { cite$ #1 #3 substring$ }
3486       'key
3487       if$
3488     }
3489     { editor format.lab.names }

```

```

3490         if$
3491     }
3492     { author format.lab.names }
3493     if$
3494 }
3495
3496 FUNCTION {author.key.organization.label}
3497 { author empty$
3498     { key empty$
3499         { organization empty$
3500             { cite$ #1 #3 substring$ }
3501             { "The " #4 organization chop.word #3 text.prefix$ }
3502             if$
3503         }
3504         'key
3505         if$
3506     }
3507     { author format.lab.names }
3508     if$
3509 }
3510
3511 FUNCTION {editor.key.organization.label}
3512 { editor empty$
3513     { key empty$
3514         { organization empty$
3515             { cite$ #1 #3 substring$ }
3516             { "The " #4 organization chop.word #3 text.prefix$ }
3517             if$
3518         }
3519         'key
3520         if$
3521     }
3522     { editor format.lab.names }
3523     if$
3524 }
3525
3526 FUNCTION {calc.short.authors}
3527 { type$ "book" =
3528     type$ "inbook" = booktitle empty$ not and
3529     or
3530     'author.editor.key.label
3531     { type$ "collection" =
3532         type$ "proceedings" =
3533         or
3534         { editor empty$ not
3535             'editor.key.organization.label
3536             'author.key.organization.label
3537             if$
3538         }
3539         'author.key.label
3540         if$
3541     }
3542     if$
3543     'short.list :=
3544 }

```

3545

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/thuthesis/#630](http://tuna.thuthesis/#630)。

```

3546 FUNCTION {calc.label}
3547 { calc.short.authors
3548   short.list "]" contains
3549     { "{" short.list * "}" * }
3550     { short.list }
3551   if$
3552   "("
3553   *
3554   format.year duplicate$ empty$
3555   short.list key field.or.null = or
3556     { pop$ "" }
3557     'skip$
3558   if$
3559   duplicate$ "]" contains
3560     { "{" swap$ * "}" * }
3561     'skip$
3562   if$
3563   *
3564   'label :=
3565 }
3566
```

B.8 Sorting

When sorting, we compute the sortkey by executing `presort` on each entry. The `presort` key contains a number of `sortifyed` strings, concatenated with multiple blanks between them. This makes things like `brinch percome beforebrinch 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`.

```

3567 <*author-year>
3568 FUNCTION {sort.language.label}
3569 { entry.lang lang.zh =
```

```

3570     { lang.zh.order }
3571     { entry.lang lang.ja =
3572       { lang.ja.order }
3573       { entry.lang lang.en =
3574         { lang.en.order }
3575         { entry.lang lang.ru =
3576           { lang.ru.order }
3577           { lang.other.order }
3578         if$
3579       }
3580     if$
3581   }
3582 if$
3583 }
3584 if$
3585 #64 +
3586 int.to.chr$
3587 }
3588
3589 FUNCTION {sort.format.names}
3590 { 's :=
3591   #1 'nameptr :=
3592   ""
3593   s num.names$ 'numnames :=
3594   numnames 'namesleft :=
3595   { namesleft #0 > }
3596   {
3597     s nameptr "{vv{ } }{ll{ }}{ ff{ }}{ jj{ }}" format.name$ 't :=
3598     nameptr #1 >
3599     {
3600       " " *
3601       namesleft #1 = t "others" = and
3602       { "zzzzz" * }
3603       { numnames #2 > nameptr #2 = and
3604         { "zz" * year field.or.null * " " * }
3605         'skip$
3606       if$
3607       t sortify *
3608     }
3609     if$
3610   }
3611   { t sortify * }
3612   if$
3613   nameptr #1 + 'nameptr :=
3614   namesleft #1 - 'namesleft :=
3615   }
3616   while$
3617 }
3618

```

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

```

3619 FUNCTION {sort.format.title}

```

```

3620 { 't :=
3621     "A " #2
3622     "An " #3
3623     "The " #4 t chop.word
3624     chop.word
3625     chop.word
3626     sortify
3627     #1 global.max$ substring$
3628 }
3629

```

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.

```

3630 FUNCTION {anonymous.sort}
3631 { entry.lang lang.zh =
3632   { "yi4 ming2" }
3633   { "anon" }
3634   if$
3635 }
3636
3637 FUNCTION {warn.empty.key}
3638 { entry.lang lang.zh =
3639   { "empty key in " cite$ * warning$ }
3640   'skip$
3641   if$
3642 }
3643
3644 FUNCTION {author.sort}
3645 { key empty$
3646   { warn.empty.key
3647     author empty$
3648     { anonymous.sort }
3649     { author sort.format.names }
3650     if$
3651   }
3652   { key }
3653   if$
3654 }
3655
3656 FUNCTION {author.editor.sort}
3657 { key empty$
3658   { warn.empty.key
3659     author empty$
3660     { editor empty$
3661       { anonymous.sort }
3662       { editor sort.format.names }
3663       if$
3664     }
3665     { author sort.format.names }
3666     if$
3667   }

```

```

3668     { key }
3669   if$
3670 }
3671
3672 FUNCTION {author.organization.sort}
3673 { key empty$
3674   { warn.empty.key
3675     author empty$
3676       { organization empty$
3677         { anonymous.sort }
3678         { "The " #4 organization chop.word sortify }
3679         if$
3680       }
3681       { author sort.format.names }
3682     if$
3683   }
3684   { key }
3685   if$
3686 }
3687
3688 FUNCTION {editor.organization.sort}
3689 { key empty$
3690   { warn.empty.key
3691     editor empty$
3692       { organization empty$
3693         { anonymous.sort }
3694         { "The " #4 organization chop.word sortify }
3695         if$
3696       }
3697       { editor sort.format.names }
3698     if$
3699   }
3700   { key }
3701   if$
3702 }
3703
3704 </author-year>

```

顺序编码制的排序要简单得多

```

3705 <numerical>
3706 INTEGERS { seq.num }
3707
3708 FUNCTION {init.seq}
3709 { #0 'seq.num :=}
3710
3711 FUNCTION {int.to.fix}
3712 { "000000000" swap$ int.to.str$ *
3713   #-1 #10 substring$
3714 }
3715
3716 </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!

```

3717 FUNCTION {presort}
3718 { set.entry.lang
3719   set.entry.numbered
3720   show.url show.doi check.electronic
3721   #0 'is.pure.electronic :=
3722   calc.label
3723   label sortify
3724   "      "
3725   *
3726   <*author-year>
3727   sort.language.label
3728   "      "
3729   *
3730   type$ "book" =
3731   type$ "inbook" = booktitle empty$ not and
3732   or
3733   'author.editor.sort
3734   { type$ "collection" =
3735     type$ "proceedings" =
3736     or
3737     'editor.organization.sort
3738     'author.sort
3739     if$
3740   }
3741   if$
3742   *
3743   "      "
3744   *
3745   year field.or.null sortify
3746   *
3747   "      "
3748   *
3749   cite$
3750   *
3751   #1 entry.max$ substring$
3752   </author-year>
3753   <*numerical>
3754   seq.num #1 + 'seq.num :=
3755   seq.num int.to.fix
3756   </numerical>
3757   'sort.label :=
3758   sort.label *
3759   #1 entry.max$ substring$
3760   'sort.key$ :=
3761 }
3762

```

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

```

```

3763 STRINGS { longest.label last.label next.extra last.extra.label }
3764
3765 INTEGERS { longest.label.width number.label }
3766
3767 FUNCTION {initialize.longest.label}
3768 { "" 'longest.label :=
3769   #0 int.to.chr$ 'last.label :=
3770   "" 'next.extra :=
3771   #0 'longest.label.width :=
3772   #0 'number.label :=
3773   "" 'last.extra.label :=
3774 }
3775
3776 FUNCTION {forward.pass}
3777 {
3778   *author-year
3779   last.label label =

```



```

3780 { "" 'extra.label :=
3781 last.extra.label text.length$ 'charptr :=
3782 { last.extra.label charptr #1 substring$ "z" =
3783 charptr #0 > and
3784 }
3785 { "a" extra.label * 'extra.label :=
3786 charptr #1 - 'charptr :=
3787 }
3788 while$
3789 charptr #0 >
3790 { last.extra.label charptr #1 substring$ chr.to.int$ #1 + int.to.ch
3791 extra.label * 'extra.label :=
3792 last.extra.label #1 charptr #1 - substring$
3793 extra.label * 'extra.label :=
3794 }
3795 { "a" extra.label * 'extra.label := }
3796 if$
3797 extra.label 'last.extra.label :=
3798 }
3799 { "a" 'last.extra.label :=
3800 "" 'extra.label :=
3801 label 'last.label :=
3802 }
3803 if$
3804 </author-year>
3805 number.label #1 + 'number.label :=
3806 }
3807
3808 FUNCTION {reverse.pass}
3809 {
3810 <*author-year>
3811 next.extra "b" =
3812 { "a" 'extra.label := }
3813 'skip$
3814 if$
3815 extra.label 'next.extra :=
3816 extra.label
3817 duplicate$ empty$
3818 'skip$
3819 { "{\natexlab{" swap$ * "}} " * }
3820 if$
3821 'extra.label :=
3822 </author-year>
3823 label extra.label * 'label :=
3824 }
3825
3826 FUNCTION {bib.sort.order}
3827 { sort.label 'sort.key$ :=
3828 }
3829

```

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.

```
3830 FUNCTION {begin.bib}
3831 {   preamble$ empty$
3832     'skip$
3833     { preamble$ write$ newline$ }
3834     if$
3835     "\begin{thebibliography}{ " number.label int.to.str$ * " }" *
3836     write$ newline$
3837     terms.in.macro
3838     { "\providecommand{\biband}{和}"
3839       write$ newline$
3840       "\providecommand{\bibetal}{等}"
3841       write$ newline$
3842     }
3843     'skip$
3844     if$
3845     "\providecommand{\natexlab}[1]{#1}"
3846     write$ newline$
3847     "\providecommand{\url}[1]{#1}"
3848     write$ newline$
3849     "\expandafter\ifx\csname urlstyle\endcsname\relax\else"
3850     write$ newline$
3851     "  \urlstyle{same}\fi"
3852     write$ newline$
3853     "\expandafter\ifx\csname href\endcsname\relax"
3854     write$ newline$
3855     "  \DeclareUrlCommand\doi{\urlstyle{rm}}"
3856     write$ newline$
3857     "  \def\epri#1#2{#2}"
3858     write$ newline$
3859     "\else"
3860     write$ newline$
3861     "  \def\doi#1{\href{https://doi.org/#1}{\nolinkurl{#1}}}"
3862     write$ newline$
3863     "  \let\epri\href"
3864     write$ newline$
3865     "\fi"
3866     write$ newline$
3867   }
3868
```

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

```
3869 FUNCTION {end.bib}
3870 {   newline$
3871     "\end{thebibliography}" write$ newline$
3872 }
```

3873

B.10 Main execution

Now we read in the .BIB entries.

```
3874 READ
3875
3876 EXECUTE {init.state.consts}
3877
3878 EXECUTE {load.config}
3879
3880 <numerical>
3881 EXECUTE {init.seq}
3882
3883 </numerical>
3884 ITERATE {presort}
3885
```

And now we can sort

```
3886 SORT
3887
3888 EXECUTE {initialize.longest.label}
3889
3890 ITERATE {forward.pass}
3891
3892 REVERSE {reverse.pass}
3893
3894 ITERATE {bib.sort.order}
3895
3896 SORT
3897
3898 EXECUTE {begin.bib}
3899
```

Now we produce the output for all the entries

```
3900 ITERATE {call.type$}
3901
3902 EXECUTE {end.bib}
3903 </author-year | numerical>
```