

GB/T 7714-2015 BibTeX style

Zeping Lee*

2019/03/21 v1.1.1

摘要

The `gbt7714` package provides a BibTeX implementation for the China's bibliography style standard GB/T 7714-2015. It consists of two `bst` files for numerical and `authoryear` styles as well as a `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”）是中国的参考文献推荐标准。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

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

2 使用方法

`super` 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”(authoryear)，
`numbers` 其中顺序编码制根据引用标注样式的不同分为角标数字式 (`super`) 和与正文平排的数字式
`authoryear` (`numbers`)。

用户应在导言区调用宏包 `gbt7714`，并在参数中选择参考文献的标注样式。默认的参数是 `super`，额外的参数会传递给 `natbib` 宏包，比如：

```
\usepackage[authoryear]{gbt7714}
```

然后不再需要调用 `\bibliographystyle` 命令设置参考文献列表风格。

使用时需要注意以下几点：

- 不再需要调用 `\bibliographystyle` 命令选择参考文献表的格式。
- `bib` 数据库应使用 UTF-8 编码。
- 使用著者-出版年制参考文献表时，中文的文献必须在 `key` 域填写作者姓名的拼音，才能按照拼音排序，详见第 5 节。

- `\cite` 在正文中引用文献时应使用 `\cite` 命令。同一处引用多篇文献时，应将各篇文献的 `key` 一同写在 `\cite` 命令中，如 `\cite{knuth84,lamport94,mittelbach04}`。如遇连续编号，可以自动转为起讫序号并用短横线连接。它可以自动排序并用处理连续编号。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]。
- `\bibliography` 参考文献表可以在文中使用 `\bibliography` 命令调用。注意文献列表的样式已经在模板中根据选项设置，用户不再需要使用 `\bibliographystyle` 命令。

3 文献类型

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

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

4 著录项目

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

- author** 主要责任者
- title** 题名
- mark*** 文献类型标识
- medium*** 载体类型标识

* zepinglee AT gmail.com

translator* 译者
editor 编辑
organization 组织（用于会议）
booktitle 图书题名
series 系列
journal 期刊题名
edition 版本
address 出版地
publisher 出版者
school 学校（用于 phdthesis）
institution 机构（用于 techreport）
year 出版年
volume 卷
number 期（或者专利号）
pages 引文页码
date* 更新或修改日期
urldate* 引用日期
url 获取和访问路径
doi 数字对象唯一标识符
language* 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,
  language = {japanese},
  mark      = {Z},
  medium    = {DK},
  ...
  
```

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

5 文献列表的排序

国标规定参考文献表采用著者-出版年制组织时，各篇文献首先按文种集中，然后按著者字顺和出版年排列；中文文献可以按著者汉语拼音字顺排列，也可以按著者的笔画笔顺排列。然而由于 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，如：

```

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

6 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。

表 2: 参考文献表样式的配置参数

参数值	默认值	功能
<code>uppercase.name</code>	<code>#1</code>	将著者姓名转为大写
<code>max.num.authors</code>	<code>#3</code>	输出著者的最多数量
<code>period.between.author.year</code>	<code>#0</code>	著者和年份之间使用句点连接
<code>sentence.case.title</code>	<code>#1</code>	将西文的题名转为 <code>sentence case</code>
<code>link.title</code>	<code>#0</code>	在题名上添加 <code>url</code> 的超链接
<code>show.mark</code>	<code>#1</code>	显示文献类型标识
<code>italic.journal</code>	<code>#0</code>	西文期刊名使用斜体
<code>show.missing.address.publisher</code>	<code>#1</code>	出版项缺失时显示“出版者不详”
<code>show.url</code>	<code>#1</code>	显示 <code>url</code>
<code>show.doi</code>	<code>#1</code>	显示 <code>doi</code>
<code>show.note</code>	<code>#0</code>	显示 <code>note</code> 域的信息

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

7 相关工作

TeX 社区也有其他关于 GB/T 7714 系列参考文献标准的工作。2005 年吴凯^[7] 发布了基于 GB/T 7714-2005 的 BibTeX 样式，支持顺序编码制和著者出版年制两种风格。李志奇^[8] 发布了严格遵循 GB/T 7714-2005 的 BibLaTeX 的样式。胡海星^[9] 提供了另一个 BibTeX 实现，还给每行 `bst` 代码写了 `java` 语言注释。沈周^[10] 基于 `biblatex-casperi`^[11] 进行修改，以符合国标的格式。胡振震发布了符合 GB/T 7714-2015 标准的 BibLaTeX 参考文献样式^[12]，并进行了比较完善的持续维护。

参考文献

- [1] 中国国家标准委员会. 信息与文献 参考文献著录规则: GB/T 7714-2015[S]. 北京: 中国标准出版社, 2015.
- [2] PATASHNIK O. BibTeXing[M/OL]. 1988. <http://mirrors.ctan.org/biblio/bibtex/base/btxdoc.pdf>.
- [3] DALY P W. Natural sciences citations and references[M/OL]. 1999. <http://mirrors.ctan.org/macros/latex/contrib/natbib/natbib.pdf>.
- [4] PATASHNIK O. Designing BibTeX styles[M/OL]. 1988. <http://mirrors.ctan.org/biblio/bibtex/base/btxhak.pdf>.

- [5] MARKEY N. Tame the beast[M/OL]. 2003. http://mirrors.ctan.org/info/bibtex/tamethebeast/tt_b_en.pdf.
- [6] MITTELBAACH F, GOOSSENS M, BRAAMS J, et al. The L^AT_EX companion[M]. 2nd ed. Reading, MA, USA: Addison-Wesley, 2004.
- [7] 吴凯. 发布 GBT7714-2005.bst version1 Beta 版[EB/OL]. 2006. <http://bbs.ctex.org/forum.php?mod=viewthread&tid=33591>.
- [8] 李志奇. 基于 biblatex 的符合 GBT7714-2005 的中文文献生成工具[EB/OL]. 2013. <http://bbs.ctex.org/forum.php?mod=viewthread&tid=74474>.
- [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>.

版本历史

v1.0 (2018/01/01)	v1.0.7 (2018/05/12)
General: Initial release. 1	bst: 修正了检测 Unicode 语言 18
v1.0.1 (2018/03/09)	v1.0.8 (2018/06/23)
General: 著者出版年制的文献引用不再排序 . . . 6	bst: 使用“~”连接英文姓名 21
v1.0.2 (2018/03/16)	支持 howpublished 中的 url 34
bst: 正确识别姓名中的“others” 21	新增接口供用户自定义样式 8
v1.0.3 (2018/03/29)	\url: 使用 xurl 的方法改进 URL 断行 8
\cite: 顺序编码制连续两个文献引用之间使用 连接号 7	v1.0.9 (2018/08/05)
v1.0.4 (2018/04/12)	bst: 不再转换题名 volume 的大小写 24
\cite: 页码的连接号由 en dash 改为 hyphen . . . 7	修正不显示 url 的选项 40
v1.0.5 (2018/04/18)	增加选项在题名添加超链接 24
bst: 允许著录多个 DOI 35	v1.1 (2019/01/02)
v1.0.6 (2018/05/10)	bst: 修正 series 的 bug 28
thebibliography: 文献列表的数字标签左对齐 . . . 8	允许自定义“et al” 13
bst: 不再处理中文标题的英文单词的大小写 . . . 24	v1.1.1 (2019/03/21)
	bst: 允许自定义文种的顺序 10

A 宏包的代码实现

下面声明和处理宏包的选项，有 `authoryear` 和 `numbers`。

```
1 \langle *package\rangle
2 \newif\if@gbt@mmxv
3 \newif\if@gbt@numerical
4 \newif\if@gbt@super
5 \DeclareOption{2015}{\@gt@mmxvtrue}
6 \DeclareOption{2005}{\@gt@mmxvfalse}
7 \DeclareOption{super}{\@gt@numericaltrue\@gt@supertrue}
8 \DeclareOption{numbers}{\@gt@numericaltrue\@gt@superfalse}
9 \DeclareOption{authoryear}{\@gt@numericalfalse}
10 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{natbib}}
11 \ExecuteOptions{2015,super}
12 \ProcessOptions\relax
```

只在顺序编码时使用 `sort&compress`。

```
13 \if@gbt@numerical
14   \PassOptionsToPackage{sort&compress}{natbib}
15 \fi
16 \RequirePackage{natbib}
17 \RequirePackage{etoolbox}
18 \RequirePackage{url}
```

`\citestyle` 定义接口切换引用文献的标注法，可用 `\citestyle` 调用 `numerical` 或 `authoryear`，参见 `natbib`。

```
19 \newcommand\bibstyle@super{\bibpunct{[ ]{ }{, }{s}{, }{\textsuperscript{,}}{,}}
20 \newcommand\bibstyle@numbers{\bibpunct{[ ]{ }{, }{n}{, }{, }{,}}
21 \newcommand\bibstyle@authoryear{\bibpunct{({ }{ )}{; }{a}{, }{, }{,}}
```

`\gbtbibstyle` 定义接口切换参考文献表的风格，可选 `authoryear` 和 `numerical`，这个仅用于 `chapterbib`。

```
22 \newcommand\gbtbibstyle[1]{%
23   \ifstrequal{#1}{numerical}{%
24     \if@gbt@mmxv
25       \bibliographystyle{gbt7714-unsrt}%
26     \else
27       \bibliographystyle{gbt7714-2005-unsrt}%
28     \fi
29   }{%
30     \ifstrequal{#1}{authoryear}{%
31       \if@gbt@mmxv
32         \bibliographystyle{gbt7714-plain}%
33       \else
34         \bibliographystyle{gbt7714-2005-plain}%
35       \fi
36     }{%
37       \PackageError{gbt7714}{Unknown argument #1.}%
38       {It should be `numerical' or `authoryear'.}%
39     }%
40   }%
41 }
```

处理宏包选项。

```
42 \if@gbt@numerical
43   \if@gbt@super
44     \citestyle{super}%
45     \gbtbibstyle{numerical}%
46   \else
47     \citestyle{numbers}
48     \gbtbibstyle{numerical}%
49   \fi
50 \else
51   \citestyle{authoryear}
52   \gbtbibstyle{authoryear}%
53 \fi
```

`\cite` 下面修改 `natbib` 的引用格式，主要是将页码写在上标位置。Numerical 模式的 `\citet` 的页码：

```
54 \newcommand\gbt@patchfailure[1]{%
55   \PackageError{gbt7714}{Failed to patch command \protect#1}{}%
56 }
57 \patchcmd{\NAT@citexnum}{%
58   \@ifnum{\NAT@ctype=\z@}{%
59     \if*#2*\else\NAT@cmt#2\fi
60   }{%
61     \NAT@mbox{\NAT@close}%
62   }{%
63     \NAT@mbox{\NAT@close}%
64     \@ifnum{\NAT@ctype=\z@}{%
65       \if*#2*\else\textsuperscript{#2}\fi
66     }{%
67   }}{\gbt@patchfailure{\NAT@citexnum}}
```

Numerical 模式的 `\citep` 的页码：

```
68 \renewcommand\NAT@citesuper[3]{\ifNAT@swa
69   \if*#2*\else#2\NAT@spacechar\fi
70   \unskip\kern\p@\textsuperscript{\NAT@open#1\NAT@close\if*#3*\else#3\fi}%
71   \else #1\fi\endgroup}
```

Author-year 模式的 `\citet` 的页码：

```
72 \patchcmd{\NAT@citex}{%
73   \if*#2*\else\NAT@cmt#2\fi
74   \if\relax\NAT@date\relax\else\NAT@close\fi
75 }{%
76   \if\relax\NAT@date\relax\else\NAT@close\fi
77   \if*#2*\else\textsuperscript{#2}\fi
78 }}{\gbt@patchfailure{\NAT@citex}}
```

Author-year 模式的 `\citep` 的页码：

```
79 \renewcommand\NAT@cite%
80   [3]{\ifNAT@swa\NAT@open\if*#2*\else#2\NAT@spacechar\fi
81     #1\NAT@close\if*#3*\else\textsuperscript{#3}\fi\else#1\fi\endgroup}
```

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。

```

82 \patchcmd{\NAT@citexnum}{%
83 \ifx\NAT@last@yr\relax
84 \def@NAT@last@yr{\@citea}%
85 \else
86 \def@NAT@last@yr{--\NAT@penalty}%
87 \fi
88 }{%
89 \def@NAT@last@yr{-\NAT@penalty}%
90 }{\gbt@patchfailure{\NAT@citexnum}}

```

thebibliography 参考文献列表的标签左对齐

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

\url 使用 xurl 宏包的方法，增加 URL 可断行的位置。

```

92 \def\UrlBreaks{%
93 \do\/%
94 \do\a\do\b\do\c\do\d\do\e\do\f\do\g\do\h\do\i\do\j\do\k\do\l%
95 \do\m\do\n\do\o\do\p\do\q\do\r\do\s\do\t\do\u\do\v\do\w\do\x\do\y\do\z%
96 \do\A\do\B\do\C\do\D\do\E\do\F\do\G\do\H\do\I\do\J\do\K\do\L%
97 \do\M\do\N\do\O\do\P\do\Q\do\R\do\S\do\T\do\U\do\V\do\W\do\X\do\Y\do\Z%
98 \do\0\do1\do2\do3\do4\do5\do6\do7\do8\do9\do=\do/\do.\do:%
99 \do\*\do-\do\~\do\' \do\" \do\~}
100 \Urlmuskip=0mu plus 0.1mu
101 \</package>

```

B BibTeX 样式的代码实现

B.1 自定义选项

bst 这里定义了一些变量用于定制样式，可以在下面的 load.config 函数中选择是否启用。

```

102 (*authoryear | numerical)
103 INTEGERS {
104 uppercase.name
105 max.num.authors
106 period.between.author.year
107 sentence.case.title
108 link.title
109 show.mark
110 slash.for.extraction
111 in.booktitle
112 italic.journal
113 bold.journal.volume
114 show.missing.address.publisher
115 show.url
116 show.doi
117 show.note
118 (*authoryear)
119 lang.zh.order
120 lang.ja.order
121 lang.en.order
122 lang.ru.order
123 lang.other.order
124 \</authoryear>
125 }
126

```


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。

```
127 FUNCTION {load.config}
128 {
```

英文姓名转为全大写：

```
129 #1 'uppercase.name :=
130 (*nouppercase | thu)
131 #0 'uppercase.name :=
132 (/nouppercase | thu)
```

最多显示的作者数量：

```
133 #3 'max.num.authors :=
```

采用著者-出版年制时，作者姓名与年份之间使用句点连接：

```
134 (*authoryear)
135 #0 'period.between.author.year :=
136 (*period | 2005 | usth)
137 #1 'period.between.author.year :=
138 (/period | 2005 | usth)
139 (/authoryear)
```

英文标题转为 sentence case（句首字母大写，其余小写）：

```
140 #1 'sentence.case.title :=
141 (*nosentencecase)
142 #0 'sentence.case.title :=
143 (/nosentencecase)
```

在标题添加超链接：

```
144 #0 'link.title :=
145 (*linktitle)
146 #1 'link.title :=
147 (/linktitle)
```

著录文献类型标识（比如“[M/OL]”）：

```
148 #1 'show.mark :=
149 (*nomark)
150 #0 'show.mark :=
151 (/nomark)
```

使用“/”表示析出文献

```
152 #1 'slash.for.extraction :=
153 (*noslash)
154 #0 'slash.for.extraction :=
155 (/noslash)
```

使用“In:”表示析出文献

```
156 #0 'in.booktitle :=
```

期刊名使用斜体：

```
157 #0 'italic.journal :=
158 (*italicjournal)
159 #1 'italic.journal :=
160 (/italicjournal)
```

期刊的卷使用粗体：

```
161 #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.”：

```
162 #1 'show.missing.address.publisher :=
163 (*noslsn | usth)
164 #0 'show.missing.address.publisher :=
165 (/noslsn | usth)
```

是否著录 URL:

```
166 #1 'show.url :=
167 <*nourl>
168 #0 'show.url :=
169 </nourl>
```

是否著录 DOI:

```
170 #1 'show.doi :=
171 <*nodoi | 2005>
172 #0 'show.doi :=
173 </nodoi | 2005>
```

在每一条文献最后输出注释 (note) 的内容:

```
174 #0 'show.note :=
```

参考文献表按照“著者-出版年”组织时, 各个文种的顺序:

```
175 <*authoryear>
176 #1 'lang.zh.order :=
177 #2 'lang.ja.order :=
178 #3 'lang.en.order :=
179 #4 'lang.ru.order :=
180 #5 'lang.other.order :=
181 </authoryear>
182 }
183
```

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 annotate 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.

```
184 ENTRY
185 { address
186   author
187   booktitle
188   date
189   doi
190   edition
191   editor
192   howpublished
193   institution
194   journal
195   key
196   language
197   mark
198   medium
199   note
200   number
201   organization
202   pages
203   publisher
204   school
205   series
206   title
207   translator
208   url
209   urldate
210   volume
211   year
212 }
213 { entry.lang entry.is.electronic entry.numbered }
```

These string entry variables are used to form the citation label. In a storage pinch, `sort.label` can be easily computed on the fly.

```
214 { label extra.label sort.label short.list entry.mark entry.url }
215
```

B.3 Entry functions

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

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

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

To tell which separator is needed, we maintain an `output.state`. It will be one of these values: `before.all` just after the `\bibitem` `mid.sentence` in the middle of a sentence: comma needed if more sentence is output `after.sentence` just after a sentence: period needed `after.block` just after a block (and sentence): period and `\newblock` needed. Note: These styles don't use `after.sentence`

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

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

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

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

```

```

216 INTEGERS { output.state before.all mid.sentence after.sentence after.block after.slash }
217
218 INTEGERS { lang.zh lang.ja lang.en lang.ru lang.other }
219
220 INTEGERS { charptr len }
221
222 FUNCTION {init.state.consts}
223 { #0 'before.all :=
224   #1 'mid.sentence :=
225   #2 'after.sentence :=
226   #3 'after.block :=
227   #4 'after.slash :=
228   #3 'lang.zh :=
229   #4 'lang.ja :=
230   #1 'lang.en :=
231   #2 'lang.ru :=
232   #0 'lang.other :=
233 }
234

```

下面是一些常量的定义

```

235 FUNCTION {bbl.anonymous}
236 { entry.lang lang.zh =
237   { "佚名" }
238   { "Anon" }
239 if$
240 }
241
242 FUNCTION {bbl.space}
243 { entry.lang lang.zh =
244   { "\ " }
245   { " " }
246 if$
247 }
248
249 FUNCTION {bbl.et.al}
250 { entry.lang lang.zh =
251   { "等" }
252   { entry.lang lang.ja =
253     { "他" }
254     { entry.lang lang.ru =
255       { "идр" }
256       { "et~al." }
257     if$
258     }
259   if$
260   }
261 if$
262 }

```

```

263
264 FUNCTION {citation.et.al}
265 { bbl.et.al }
266
267 FUNCTION {bbl.colon} { ": " }
268
269 ⟨*2015⟩
270 FUNCTION {bbl.wide.space} { "\quad " }
271 ⟨/2015⟩
272 ⟨*2005⟩
273 FUNCTION {bbl.wide.space} { "\ " }
274 ⟨/2005⟩
275
276 FUNCTION {bbl.slash} { "//\allowbreak " }
277
278 FUNCTION {bbl.sine.loco}
279 { entry.lang lang.zh =
280   { "[出版地不详]" }
281   { "[S.l.]" }
282   if$
283 }
284
285 FUNCTION {bbl.sine.nomine}
286 { entry.lang lang.zh =
287   { "[出版者不详]" }
288   { "[s.n.]" }
289   if$
290 }
291
292 FUNCTION {bbl.sine.loco.sine.nomine}
293 { entry.lang lang.zh =
294   { "[出版地不详: 出版者不详]" }
295   { "[S.l.: s.n.]" }
296   if$
297 }
298

```

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

```

299 FUNCTION {not}
300 { { #0 }
301   { #1 }
302   if$
303 }
304
305 FUNCTION {and}
306 { 'skip$
307   { pop$ #0 }
308   if$
309 }
310
311 FUNCTION {or}
312 { { pop$ #1 }
313   'skip$
314   if$
315 }
316

```

the variables s and t are temporary string holders

```

317 STRINGS { s t }
318
319 FUNCTION {output.nonnull}

```

```

320 { 's :=
321   output.state mid.sentence =
322     { ", " * write$ }
323     { output.state after.block =
324       { add.period$ write$
325         newline$
326         "\newblock " write$
327       }
328       { output.state before.all =
329         'write$
330         { output.state after.slash =
331           { bbl.slash * write$
332             newline$
333           }
334           { add.period$ " " * write$ }
335         if$
336       }
337     if$
338   }
339   if$
340   mid.sentence 'output.state :=
341 }
342 if$
343 s
344 }
345
346 FUNCTION {output}
347 { duplicate$ empty$
348   'pop$
349   'output.nonnull
350   if$
351 }
352
353 FUNCTION {output.after}
354 { 't :=
355   duplicate$ empty$
356   'pop$
357   { 's :=
358     output.state mid.sentence =
359     { t * write$ }
360     { output.state after.block =
361       { add.period$ write$
362         newline$
363         "\newblock " write$
364       }
365       { output.state before.all =
366         'write$
367         { output.state after.slash =
368           { bbl.slash * write$ }
369           { add.period$ " " * write$ }
370         if$
371       }
372     if$
373   }
374   if$
375   mid.sentence 'output.state :=
376 }
377 if$
378 s
379 }
380 if$
381 }
382
383 FUNCTION {output.check}

```

```

384 { 't :=
385   duplicate$ empty$
386   { pop$ "empty " t * " in " * cite$ * warning$ }
387   'output.nonnull
388   if$
389 }
390

```

This function finishes all entries.

Tsinghua requires no period at the end of book-like entries.

```

391 FUNCTION {fin.entry}
392 <!*thu>
393 { add.period$
394 </!thu>
395 <!*thu>
396 { type$ "book" =
397   type$ "inbook" = or
398   type$ "incollection" = or
399   type$ "collection" = or
400   'skip$
401   'add.period$
402   if$
403 </!thu>
404   write$
405   newline$
406 }
407
408 FUNCTION {new.block}
409 { output.state before.all =
410   'skip$
411   { output.state after.slash =
412     'skip$
413     { after.block 'output.state := }
414     if$
415   }
416   if$
417 }
418
419 FUNCTION {new.sentence}
420 { output.state after.block =
421   'skip$
422   { output.state before.all =
423     'skip$
424     { output.state after.slash =
425       'skip$
426       { after.sentence 'output.state := }
427       if$
428     }
429     if$
430   }
431   if$
432 }
433
434 FUNCTION {new.slash}
435 { output.state before.all =
436   'skip$
437   { slash.for.extraction
438     { after.slash 'output.state := }
439     { after.block 'output.state := }
440     if$
441   }
442   if$
443 }
444

```


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.

```
445 FUNCTION {new.block.checka}
446 { empty$
447   'skip$
448   'new.block
449   if$
450 }
451
452 FUNCTION {new.block.checkb}
453 { empty$
454   swap$ empty$
455   and
456   'skip$
457   'new.block
458   if$
459 }
460
```

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

```
461 FUNCTION {new.sentence.checka}
462 { empty$
463   'skip$
464   'new.sentence
465   if$
466 }
467
468 FUNCTION {new.sentence.checkb}
469 { empty$
470   swap$ empty$
471   and
472   'skip$
473   'new.sentence
474   if$
475 }
476
```

B.4 Formatting chunks

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

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

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

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

```

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

```

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

```

477 FUNCTION {field.or.null}
478 { duplicate$ empty$
479   { pop$ "" }
480   'skip$
481   if$
482 }
483
484 FUNCTION {italicize}
485 { duplicate$ empty$
486   { pop$ "" }
487   { "\textit{" swap$ * "" * }
488   if$
489 }
490

```

B.4.1 Detect Language

```

491 INTEGERS { byte second.byte }
492
493 INTEGERS { char.lang tmp.lang }
494
495 STRINGS { tmp.str }
496
497 FUNCTION {get.str.lang}
498 { 'tmp.str :=
499   lang.other 'tmp.lang :=
500   #1 'charptr :=
501   tmp.str text.length$ #1 + 'len :=
502   { charptr len < }
503   { tmp.str charptr #1 substring$ chr.to.int$ 'byte :=
504     byte #128 <
505     { charptr #1 + 'charptr :=
506       byte #64 > byte #91 < and byte #96 > byte #123 < and or
507       { lang.en 'char.lang := }
508       { lang.other 'char.lang := }
509     if$
510   }
511   { tmp.str charptr #1 + #1 substring$ chr.to.int$ 'second.byte :=
512     byte #224 <

```

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

```

513     { charptr #2 + 'charptr :=
514       byte #207 > byte #212 < and
515       byte #212 = second.byte #176 < and or
516       { lang.ru 'char.lang := }
517       { lang.other 'char.lang := }
518     if$
519   }
520   { byte #240 <

```

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

```

521     { charptr #3 + 'charptr :=
522       byte #227 > byte #234 < and
523       { lang.zh 'char.lang := }

```

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

```

524         { byte #227 =
525             { second.byte #143 >
526                 { lang.zh 'char.lang := }

```

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

```

527             { second.byte #128 > second.byte #132 < and
528                 { lang.ja 'char.lang := }
529                 { lang.other 'char.lang := }
530             if$
531         }
532     if$
533 }

```

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

```

534     { byte #239 =
535         second.byte #163 > second.byte #172 < and and
536         { lang.zh 'char.lang := }
537         { lang.other 'char.lang := }
538     if$
539     }
540     if$
541     }
542     if$
543 }

```

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.

```

544     { charptr #4 + 'charptr :=
545         byte #240 = second.byte #159 > and
546         { lang.zh 'char.lang := }
547         { lang.other 'char.lang := }
548     if$
549     }
550     if$
551     }
552     if$
553     }
554     if$
555     char.lang tmp.lang >
556     { char.lang 'tmp.lang := }
557     'skip$
558     if$
559     }
560 while$
561 tmp.lang
562 }

```

```

563
564 FUNCTION {check.entry.lang}
565 { author field.or.null
566   title field.or.null *
567   get.str.lang
568 }
569

```

```

570 FUNCTION {set.entry.lang}
571 { language empty$
572   { check.entry.lang }
573   { language "english" = language "american" = or language "british" = or
574     { lang.en }
575     { language "chinese" =
576       { lang.zh }
577       { language "japanese" =
578         { lang.ja }
579         { language "russian" =
580           { lang.ru }
581           { check.entry.lang }

```

```

582             if$
583             }
584         if$
585     }
586     if$
587     }
588     if$
589 }
590 if$
591 'entry.lang :=
592 }
593
594 FUNCTION {set.entry.numbered}
595 { type$ "patent" =
596   type$ "standard" = or
597   type$ "techreport" = or
598   { #1 'entry.numbered := }
599   { #0 'entry.numbered := }
600 if$
601 }
602

```

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
    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.

```

603 INTEGERS { nameptr namesleft numnames name.lang }
604
605 FUNCTION {format.names}
606 { 's :=
607   #1 'nameptr :=
608   s num.names$ 'numnames :=
609   numnames 'namesleft :=
610   { namesleft #0 > }
611   { s nameptr "{vv~}{ll}{, jj}{, ff}" format.name$ 't :=
612     nameptr max.num.authors >
613     { bbl.et.al
614       #1 'namesleft :=
615     }
616     { t "others" =
617       { bbl.et.al }
618       { t get.str.lang 'name.lang :=
619         name.lang lang.en =
620         { t #1 "{vv~}{ll}{~f{~}}" format.name$
621           uppercase.name
622           { "u" change.case$ }
623           'skip$
624           if$
625           t #1 "{, jj}" format.name$ *
626         }
627         { t #1 "{ll}{ff}" format.name$ }
628         if$
629       }
630     }
631   }
632   if$
633   nameptr #1 >
634   { ", " swap$ * * }
635   'skip$
636   if$
637   nameptr #1 + 'nameptr :=
638   namesleft #1 - 'namesleft :=
639 }
640 while$
641 }
642
643 FUNCTION {format.key}

```

```

644 { empty$
645   { key field.or.null }
646   { "" }
647   if$
648 }
649
650 FUNCTION {format.authors}
651 { author empty$ not
652   { author format.names }
653   { "empty author in " cite$ * warnings$
654   (*authoryear)
655     bbl.anonymous
656   (/authoryear)
657   (*numerical)
658     ""
659   (/numerical)
660   }
661   if$
662 }
663
664 FUNCTION {format.editors}
665 { editor empty$
666   { "" }
667   { editor format.names }
668   if$
669 }
670
671 FUNCTION {format.translators}
672 { translator empty$
673   { "" }
674   { translator format.names
675     entry.lang lang.zh =
676     { translator num.names$ #3 >
677       { " 译" * }
678       { ", 译" * }
679       if$
680     }
681     'skip$
682     if$
683   }
684   if$
685 }
686
687 FUNCTION {format.full.names}
688 { 's :=
689   #1 'nameptr :=
690   s num.names$ 'numnames :=
691   numnames 'namesleft :=
692   { namesleft #0 > }
693   { s nameptr "{vv~}{ll}{, jj}{, ff}" format.name$ 't :=
694     t get.str.lang 'name.lang :=
695     name.lang lang.en =
696     { t #1 "{vv~}{ll}" format.name$ 't := }
697     { t #1 "{ll}{ff}" format.name$ 't := }
698     if$
699     nameptr #1 >
700     {
701       namesleft #1 >
702       { ", " * t * }
703       {
704         numnames #2 >
705         { ", " * }
706         'skip$
707         if$

```

```

708         t "others" =
709             { " et~al." * }
710             { " and " * t * }
711         if$
712     }
713     if$
714 }
715 't
716 if$
717     nameptr #1 + 'nameptr :=
718     namesleft #1 - 'namesleft :=
719 }
720 while$
721 }
722
723 FUNCTION {author.editor.full}
724 { author empty$
725     { editor empty$
726         { "" }
727         { editor format.full.names }
728     if$
729 }
730 { author format.full.names }
731 if$
732 }
733
734 FUNCTION {author.full}
735 { author empty$
736     { "" }
737     { author format.full.names }
738 if$
739 }
740
741 FUNCTION {editor.full}
742 { editor empty$
743     { "" }
744     { editor format.full.names }
745 if$
746 }
747
748 FUNCTION {make.full.names}
749 { type$ "book" =
750     type$ "inbook" =
751     or
752     'author.editor.full
753     { type$ "collection" =
754         type$ "proceedings" =
755         or
756         'editor.full
757         'author.full
758     if$
759 }
760 if$
761 }
762
763 FUNCTION {output.bibitem}
764 { newline$
765     "\bibitem[" write$
766     label write$
767     ")" make.full.names duplicate$ short.list =
768     { pop$ }
769     { * }
770 if$
771 's :=

```

```

772 s text.length$ 'charptr :=
773   { charptr #0 > }
774   { s charptr #1 substring$ "]" =
775     { #0 'charptr := }
776     { charptr #1 - 'charptr := }
777     if$
778   }
779 while$
780 charptr #0 >
781   { "{" s * "}" * }
782   { s }
783 if$
784 "]" * write$
785 cite$ write$
786 "]" write$
787 newline$
788 ""
789 before.all 'output.state :=
790 }
791

```

B.4.3 Format title

The `format.title` function is used for non-book-like titles. For most styles we convert to lower-case (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.

```

792 FUNCTION {change.sentence.case}
793 { entry.lang lang.en =
794   { "t" change.case$ }
795   'skip$
796   if$
797 }
798
799 FUNCTION {add.link}
800 { url empty$ not
801   { "\href{" url * "}" * swap$ * "}" * }
802   { doi empty$ not
803     { "\href{http://dx.doi.org/" doi * "}" * swap$ * "}" * }
804     'skip$
805     if$
806   }
807 if$
808 }
809
810 FUNCTION {format.title}
811 { title empty$
812   { "" }
813   { title
814     sentence.case.title
815     'change.sentence.case
816     'skip$
817     if$
818     entry.numbered number empty$ not and
819     { bbl.colon * number * }
820     'skip$
821     if$
822     link.title
823     'add.link
824     'skip$
825     if$
826   }

```



```

827 if$
828 }
829

```

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

```

```

830 FUNCTION {tie.or.space.connect}
831 { duplicate$ text.length$ #3 <
832   { "~" }
833   { " " }
834   if$
835   swap$ * *
836 }
837

```

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

```

```

838 FUNCTION {either.or.check}
839 { empty$
840   'pop$
841   { "can't use both " swap$ * " fields in " * cite$ * warning$ }
842   if$
843 }
844

```

The `format.bvolume` function is for formatting the volume and perhaps series name of a multivolume work. If both a volume and a series field are there, we assume the series field is the title of the whole multivolume work (the title field should be the title of the thing being referred to), and we add an "of <series>". This function is called in mid-sentence.

The `format.number.series` function is for formatting the series name and perhaps number of a work in a series. This function is similar to `format.bvolume`, although for this one the series must exist (and the volume must not exist). If the number field is empty we output either the series field unchanged if it exists or else the null string. If both the number and series fields are there we assume the series field gives the name of the whole series (the title field should be the title of the work being one referred to), and we add an "in <series>". We capitalize Number when this function is used at the beginning of a block.

```

845 FUNCTION {is.digit}
846 { duplicate$ empty$
847   { pop$ #0 }
848   { chr.to.int$
849     duplicate$ "0" chr.to.int$ <
850     { pop$ #0 }

```

```

851     { "9" chr.to.int$ >
852       { #0 }
853       { #1 }
854     if$
855   }
856   if$
857 }
858 if$
859 }
860
861 FUNCTION {is.number}
862 { 's :=
863   s empty$
864   { #0 }
865   { s text.length$ 'charptr :=
866     { charptr #0 >
867       s charptr #1 substring$ is.digit
868       and
869     }
870     { charptr #1 - 'charptr := }
871     while$
872     charptr not
873   }
874   if$
875 }
876
877 FUNCTION {format.volume}
878 { volume empty$ not
879   { volume is.number
880     { entry.lang lang.zh =
881       { " 第 " volume * " 卷" * }
882       { "volume" volume tie.or.space.connect }
883     if$
884   }
885   { volume }
886   if$
887 }
888 { "" }
889 if$
890 }
891
892 FUNCTION {format.number}
893 { number empty$ not
894   { number is.number
895     { entry.lang lang.zh =
896       { " 第 " number * " 册" * }
897       { "number" number tie.or.space.connect }
898     if$
899   }
900   { number }
901   if$
902 }
903 { "" }
904 if$
905 }
906
907 FUNCTION {format.volume.number}
908 { volume empty$ not
909   { format.volume }
910   { format.number }
911   if$
912 }
913
914 FUNCTION {format.title.vol.num}

```

```

915 { title
916   sentence.case.title
917   'change.sentence.case
918   'skip$
919   if$
920   entry.numbered
921     { number.empty$ not
922       { bbl.colon * number * }
923       'skip$
924     if$
925     }
926     { format.volume.number 's :=
927       s.empty$ not
928       { bbl.colon * s * }
929       'skip$
930     if$
931     }
932   if$
933 }
934
935 FUNCTION {format.series.vol.num.title}
936 { format.volume.number 's :=
937   series.empty$ not
938   { series
939     sentence.case.title
940     'change.sentence.case
941     'skip$
942     if$
943     entry.numbered
944       { bbl.wide.space * }
945       { bbl.colon *
946         s.empty$ not
947         { s * bbl.wide.space * }
948         'skip$
949       if$
950     }
951     if$
952     title *
953     sentence.case.title
954     'change.sentence.case
955     'skip$
956     if$
957     entry.numbered.number.empty$ not and
958     { bbl.colon * number * }
959     'skip$
960     if$
961   }
962   { format.title.vol.num }
963   if$
964   link.title
965   'add.link
966   'skip$
967   if$
968 }
969
970 FUNCTION {format.booktitle.vol.num}
971 { booktitle
972   entry.numbered
973   'skip$
974   { format.volume.number 's :=
975     s.empty$ not
976     { bbl.colon * s * }
977     'skip$
978   if$

```

```

979     }
980   if$
981 }
982
983 FUNCTION {format.series.vol.num.booktitle}
984 { format.volume.number 's :=
985   series empty$ not
986   { series bbl.colon *
987     entry.numbered not s empty$ not and
988     { s * bbl.wide.space * }
989     'skip$
990     if$
991     booktitle *
992   }
993   { format.booktitle.vol.num }
994   if$
995   in.booktitle
996   { duplicate$ empty$ not entry.lang lang.en = and
997     { "In: " swap$ * }
998     'skip$
999     if$
1000   }
1001   'skip$
1002   if$
1003 }
1004
1005 FUNCTION {format.journal}
1006 { journal
1007   italic.journal entry.lang lang.en = and
1008   'italicize
1009   'skip$
1010   if$
1011 }
1012

```

B.4.4 Format entry type mark

```

1013 FUNCTION {set.entry.mark}
1014 { entry.mark empty$ not
1015   'pop$
1016   { mark empty$ not
1017     { pop$ mark 'entry.mark := }
1018     { 'entry.mark := }
1019     if$
1020   }
1021   if$
1022 }
1023
1024 FUNCTION {format.mark}
1025 { show.mark
1026   { medium empty$ not
1027     { entry.mark "/" * medium * 'entry.mark := }
1028     { entry.is.electronic
1029       { entry.mark "/OL" * 'entry.mark := }
1030       'skip$
1031     }
1032     if$
1033     "\allowbreak[" entry.mark * "]" *
1034   }
1035   { "" }
1036   if$
1037 }
1038 }
1039

```

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.

```
1040 FUNCTION {num.to.ordinal}
1041 { duplicate$ text.length$ 'charptr :=
1042   duplicate$ charptr #1 substring$ 's :=
1043   s "1" =
1044     { "st" * }
1045     { s "2" =
1046       { "nd" * }
1047       { s "3" =
1048         { "rd" * }
1049         { "th" * }
1050         if$
1051       }
1052     }
1053   }
1054   if$
1055 }
1056
1057 FUNCTION {format.edition}
1058 { edition empty$
1059   { "" }
1060   { edition is.number
1061     { entry.lang lang.zh =
1062       { edition " 版" * }
1063       { edition num.to.ordinal " ed." * }
1064       if$
1065     }
1066     { entry.lang lang.en =
1067       { edition change.sentence.case 's :=
1068         s "Revised" = s "Revised edition" = or
1069         { "Rev. ed." }
1070         { s " ed." * }
1071         if$
1072       }
1073       { edition }
1074       if$
1075     }
1076   }
1077 }
1078 if$
1079 }
1080
```

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。

```
1081 FUNCTION {format.publisher}
1082 { publisher empty$ not
1083   { publisher }
1084   { school empty$ not
1085     { school }
1086     { organization empty$ not
1087       { organization }
1088       { institution empty$ not
1089         { institution }
1090         { "" }
1091       }
1092     }
1093   }
1094   if$
1095 }
```

```

1094     }
1095     if$
1096   }
1097   if$
1098 }
1099
1100 FUNCTION {format.address.publisher}
1101 { address empty$ not
1102   { address
1103     format.publisher empty$ not
1104     { bbl.colon * format.publisher * }
1105     { entry.is.electronic not show.missing.address.publisher and
1106       { bbl.colon * bbl.sine.nomine * }
1107       'skip$
1108     if$
1109   }
1110   if$
1111 }
1112 { entry.is.electronic not show.missing.address.publisher and
1113   { format.publisher empty$ not
1114     { bbl.sine.loco bbl.colon * format.publisher * }
1115     { bbl.sine.loco.sine.nomine }
1116   if$
1117 }
1118 { format.publisher empty$ not
1119   { format.publisher }
1120   { "" }
1121   if$
1122 }
1123 if$
1124 }
1125 if$
1126 }
1127

```

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.

Newsraer 和 paptent 要显示完整的日期，同时不再显示修改日期。但是在 `author-year` 模式下，需要单独设置 `format.year`。

```

1128 FUNCTION {extract.before.dash}
1129 { duplicate$ empty$
1130   { pop$ "" }
1131   { 's :=
1132     #1 'charptr :=
1133     s text.length$ #1 + 'len :=
1134     { charptr len <
1135       s charptr #1 substring$ "-" = not
1136       and
1137     }
1138     { charptr #1 + 'charptr := }
1139     while$
1140     s #1 charptr #1 - substring$
1141   }
1142   if$
1143 }
1144
1145 FUNCTION {extract.after.dash}
1146 { duplicate$ empty$
1147   { pop$ "" }
1148   { 's :=

```

```

1149 #1 'charptr :=
1150 s text.length$ #1 + 'len :=
1151 { charptr len <
1152   s charptr #1 substring$ "-" = not
1153   and
1154 }
1155 { charptr #1 + 'charptr := }
1156 while$
1157 { charptr len <
1158   s charptr #1 substring$ "-" =
1159   and
1160 }
1161 { charptr #1 + 'charptr := }
1162 while$
1163 s charptr global.max$ substring$
1164 }
1165 if$
1166 }
1167
1168 FUNCTION {contains.dash}
1169 { duplicate$ empty$
1170   { pop$ #0 }
1171   { 's :=
1172     { s empty$ not
1173       s #1 #1 substring$ "-" = not
1174       and
1175     }
1176     { s #2 global.max$ substring$ 's := }
1177     while$
1178     s empty$ not
1179   }
1180   if$
1181 }
1182

```

著者-出版年制必须提取出年份

```

1183 FUNCTION {format.year}
1184 { year empty$ not
1185   { year extract.before.dash }
1186   { date empty$ not
1187     { date extract.before.dash }
1188     { "empty year in " cite$ * warning$
1189       urldate empty$ not
1190       { "[" urldate extract.before.dash * "]" * }
1191       { "" }
1192     }
1193     if$
1194   }
1195   }
1196   if$
1197   extra.label *
1198 }
1199

```

专利和报纸都是使用日期而不是年

```

1200 FUNCTION {format.date}
1201 { type$ "patent" = type$ "newspaper" = or
1202   date empty$ not and
1203   { date }
1204   { year }
1205   if$
1206 }
1207

```

更新、修改日期只用于电子资源 `elctronic`

```
1208 FUNCTION {format.editdate}
1209 { date empty$ not
1210   { "\allowbreak(" date * ")" * }
1211   { "" }
1212   if$
1213 }
1214
```

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。

```
1215 FUNCTION {format.urldate}
1216 { urldate empty$ not entry.is.electronic and
1217   { "\allowbreak[" urldate * "]" * }
1218   { "" }
1219   if$
1220 }
1221
```

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`。

```
1222 FUNCTION {hyphenate}
1223 { 't :=
1224   ""
```



```

1225 { t empty$ not }
1226 { t #1 #1 substring$ "-" =
1227   { "-" *
1228     { t #1 #1 substring$ "-" = }
1229     { t #2 global.max$ substring$ 't := }
1230     while$
1231   }
1232   { t #1 #1 substring$ *
1233     t #2 global.max$ substring$ 't :=
1234   }
1235   if$
1236 }
1237 while$
1238 }
1239

```

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

```

1240 FUNCTION {format.pages}
1241 { pages empty$
1242   { "" }
1243   { pages hyphenate }
1244   if$
1245 }
1246

```

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。

```

1247 FUNCTION {format.journal.volume}
1248 { volume empty$ not
1249   { bold.journal.volume
1250     { "\textbf{" volume * "}" * }
1251     { volume }
1252     if$
1253   }
1254   { "" }
1255   if$
1256 }
1257
1258 FUNCTION {format.journal.number}
1259 { number empty$ not
1260   { "\penalty0 (" number * ")" * }
1261   { "" }
1262   if$
1263 }
1264
1265 FUNCTION {format.journal.pages}
1266 { pages empty$
1267   { "" }
1268   { ":\penalty0 " pages hyphenate * }
1269   if$
1270 }
1271

```

连续出版物的年卷期有起止范围，需要特殊处理

```

1272 FUNCTION {format.periodical.year.volume.number}
1273 { year empty$ not
1274   { year extract.before.dash }
1275   { "empty year in periodical " cite$ * warnings }
1276   if$

```

```

1277 volume empty$ not
1278   { ", " * volume extract.before.dash * }
1279   'skip$
1280 if$
1281 number empty$ not
1282   { "\penalty0 (" * number extract.before.dash * ")" * }
1283   'skip$
1284 if$
1285 year contains.dash
1286   { "--" *
1287     year extract.after.dash empty$
1288     volume extract.after.dash empty$ and
1289     number extract.after.dash empty$ and not
1290     { year extract.after.dash empty$ not
1291       { year extract.after.dash * }
1292       { year extract.before.dash * }
1293       if$
1294         volume empty$ not
1295         { ", " * volume extract.after.dash * }
1296         'skip$
1297         if$
1298           number empty$ not
1299           { "\penalty0 (" * number extract.after.dash * ")" * }
1300           'skip$
1301           if$
1302             }
1303             'skip$
1304           if$
1305         }
1306         'skip$
1307       if$
1308     }
1309

```

B.4.9 Format url and doi

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

```

1310 FUNCTION {check.url}
1311 { url empty$ not
1312   { "\url{" url * "}" * 'entry.url :=
1313     #1 'entry.is.electronic :=
1314   }
1315   { howpublished empty$ not
1316     { howpublished #1 #5 substring$ "\url{" =
1317       { howpublished 'entry.url :=
1318         #1 'entry.is.electronic :=
1319       }
1320       'skip$
1321     if$
1322   }
1323   { note empty$ not
1324     { note #1 #5 substring$ "\url{" =
1325       { note 'entry.url :=
1326         #1 'entry.is.electronic :=
1327       }
1328       'skip$
1329     if$
1330   }
1331   'skip$
1332 if$
1333 }
1334 if$
1335 }

```

```

1336 if$
1337 }
1338
1339 FUNCTION {format.url}
1340 { entry.url empty$ not
1341   { new.block entry.url }
1342   { "" }
1343 if$
1344 }
1345

```

需要检测 DOI 是否已经包含在 URL 中。

```

1346 FUNCTION {check.doi}
1347 { doi empty$ not
1348   { #1 'entry.is.electronic := }
1349   'skip$
1350 if$
1351 }
1352
1353 FUNCTION {is.in.url}
1354 { 's :=
1355   s empty$
1356   { #1 }
1357   { entry.url empty$
1358     { #0 }
1359     { s text.length$ 'len :=
1360       entry.url text.length$ 'charptr :=
1361       { entry.url charptr len substring$ s = not
1362         charptr #0 >
1363         and
1364         }
1365       { charptr #1 - 'charptr := }
1366       while$
1367       charptr
1368     }
1369     if$
1370   }
1371 if$
1372 }
1373
1374 FUNCTION {format.doi}
1375 { ""
1376   doi empty$ not show.doi and
1377   { "" 's :=
1378     doi 't :=
1379     #0 'numnames :=
1380     { t empty$ not}
1381     { t #1 #1 substring$ 'tmp.str :=
1382       tmp.str "," = tmp.str " " = or t #2 #1 substring$ empty$ or
1383       { t #2 #1 substring$ empty$
1384         { s tmp.str * 's := }
1385         'skip$
1386       if$
1387       s empty$ s is.in.url or
1388       'skip$
1389       { numnames #1 + 'numnames :=
1390         numnames #1 >
1391         { ", " * }
1392         { "DOI: " * }
1393         if$
1394         "\doi{" s * "}" * *
1395       }
1396       if$
1397       "" 's :=

```

```

1398     }
1399     { s tmp.str * 's := }
1400     if$
1401     t #2 global.max$ substring$ 't :=
1402     }
1403     while$
1404     's :=
1405     s empty$ not
1406     { new.block s }
1407     { "" }
1408     if$
1409     }
1410     'skip$
1411     if$
1412 }
1413
1414 FUNCTION {check.electronic}
1415 { "" 'entry.url :=
1416   #0 'entry.is.electronic :=
1417   'check.doi
1418   'skip$
1419   if$
1420   'check.url
1421   'skip$
1422   if$
1423   medium empty$ not
1424   { medium "MT" = medium "DK" = or medium "CD" = or medium "OL" = or
1425     { #1 'entry.is.electronic := }
1426     'skip$
1427     if$
1428     }
1429     'skip$
1430   if$
1431 }
1432
1433 FUNCTION {format.note}
1434 { note empty$ not show.note and
1435   { note }
1436   { "" }
1437   if$
1438 }
1439

```

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

```

1440 FUNCTION {empty.misc.check}
1441 { author empty$ title empty$
1442   year empty$
1443   and and
1444   key empty$ not and
1445   { "all relevant fields are empty in " cite$ * warnings$ }
1446   'skip$
1447   if$
1448 }
1449

```

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

```
1450 FUNCTION {monograph}
1451 { output.bibitem
1452   author empty$ not
1453     { format.authors }
1454     { editor empty$ not
1455       { format.editors }
1456       { "empty author and editor in " cite$ * warning$
1457 < *authoryear >
1458       bbl.anonymous
1459 < /authoryear >
1460 < *numerical >
1461     ""
1462 < /numerical >
1463   }
1464   if$
1465 }
1466 if$
1467 output
1468 < *authoryear >
1469   period.between.author.year
1470   'new.sentence
1471   'skip$
1472   if$
1473   format.year "year" output.check
1474 < /authoryear >
1475 new.block
1476 format.series.vol.num.title "title" output.check
1477 "M" set.entry.mark
1478 format.mark "" output.after
1479 new.block
1480 format.translators output
1481 new.sentence
1482 format.edition output
1483 new.block
1484 format.address.publisher output
1485 < *numerical >
1486   format.year "year" output.check
1487 < /numerical >
1488 format.pages bbl.colon output.after
1489 format.urldate "" output.after
1490 format.url output
1491 format.doi output
1492 new.block
1493 format.note output
1494 fin.entry
1495 }
1496
```

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

```
1497 FUNCTION {incollection}
1498 { output.bibitem
1499   format.authors output
```

```

1500 author format.key output
1501 <{*authoryear}
1502 period.between.author.year
1503 'new.sentence
1504 'skip$
1505 if$
1506 format.year "year" output.check
1507 </authoryear>
1508 new.block
1509 format.title "title" output.check
1510 "M" set.entry.mark
1511 format.mark "" output.after
1512 new.block
1513 format.translators output
1514 new.slash
1515 format.editors output
1516 new.block
1517 format.series.vol.num.booktitle "booktitle" output.check
1518 new.block
1519 format.edition output
1520 new.block
1521 format.address.publisher output
1522 <{*numerical}
1523 format.year "year" output.check
1524 </numerical>
1525 format.pages bbl.colon output.after
1526 format.urldate "" output.after
1527 format.url output
1528 format.doi output
1529 new.block
1530 format.note output
1531 fin.entry
1532 }
1533

```

B.5.3 连续出版物

```

1534 FUNCTION {periodical}
1535 { output.bibitem
1536   format.authors output
1537   author format.key output
1538 <{*authoryear}
1539 period.between.author.year
1540 'new.sentence
1541 'skip$
1542 if$
1543 format.year "year" output.check
1544 </authoryear>
1545 new.block
1546 format.title "title" output.check
1547 "J" set.entry.mark
1548 format.mark "" output.after
1549 new.block
1550 format.periodical.year.volume.number output
1551 new.block
1552 format.address.publisher output
1553 <{*numerical}
1554 format.date "year" output.check
1555 </numerical>
1556 format.urldate "" output.after
1557 format.url output
1558 format.doi output
1559 new.block
1560 format.note output

```

```
1561 fin.entry
1562 }
1563
```

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.

```
1564 FUNCTION {article}
1565 { output.bibitem
1566   format.authors output
1567   author format.key output
1568   <*authoryear>
1569   period.between.author.year
1570   'new.sentence
1571   'skip$
1572   if$
1573   format.year "year" output.check
1574 </authoryear>
1575   new.block
1576   format.title "title" output.check
1577   "J" set.entry.mark
1578   format.mark "" output.after
1579   new.block
1580   format.journal "journal" output.check
1581 <*numerical>
1582   format.date "year" output.check
1583 </numerical>
1584   format.journal.volume output
1585   format.journal.number "" output.after
1586   format.journal.pages "" output.after
1587   format.urldate "" output.after
1588   format.url output
1589   format.doi output
1590   new.block
1591   format.note output
1592   fin.entry
1593 }
1594
```

B.5.5 专利文献

number 域也可以用来表示专利号。

```
1595 FUNCTION {patent}
1596 { output.bibitem
1597   format.authors output
1598   author format.key output
1599 <*authoryear>
1600   period.between.author.year
1601   'new.sentence
1602   'skip$
1603   if$
1604   format.year "year" output.check
1605 </authoryear>
1606   new.block
1607   format.title "title" output.check
1608   "P" set.entry.mark
1609   format.mark "" output.after
1610   new.block
1611   format.date "year" output.check
```

```

1612 format.urldate "" output.after
1613 format.url output
1614 format.doi output
1615 new.block
1616 format.note output
1617 fin.entry
1618 }
1619

```

B.5.6 电子资源

```

1620 FUNCTION {electronic}
1621 { #1 #1 check.electronic
1622   #1 'entry.is.electronic :=
1623   output.bibitem
1624   format.authors output
1625   author format.key output
1626   (*authoryear)
1627   period.between.author.year
1628   'new.sentence
1629   'skip$
1630   if$
1631   format.year "year" output.check
1632 }/authoryear)
1633 new.block
1634 format.series.vol.num.title "title" output.check
1635 "EB" set.entry.mark
1636 format.mark "" output.after
1637 new.block
1638 format.address.publisher output
1639 (*numerical)
1640 date empty$
1641   { format.date output }
1642   'skip$
1643   if$
1644 }/numerical)
1645 format.pages bbl.colon output.after
1646 format.editedate "" output.after
1647 format.urldate "" output.after
1648 format.url output
1649 format.doi output
1650 new.block
1651 format.note output
1652 fin.entry
1653 }
1654

```

B.5.7 其他文献类型

A misc is something that doesn't fit elsewhere.

Required: at least one of the 'optional' fields

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

Misc 用来自动判断类型。

```

1655 FUNCTION {misc}
1656 { journal empty$ not
1657   'article
1658   { booktitle empty$ not
1659     'incollection
1660     { publisher empty$ not
1661       'monograph
1662       { entry.is.electronic
1663         'electronic
1664         { "Z" set.entry.mark

```



```

1665             monograph
1666         }
1667     if$
1668 }
1669 if$
1670 }
1671 if$
1672 }
1673 if$
1674 empty.misc.check
1675 }
1676
1677 FUNCTION {archive}
1678 { "A" set.entry.mark
1679   misc
1680 }
1681

```

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

```

1682 FUNCTION {book} { monograph }
1683

```

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

Required: title

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

```

1684 FUNCTION {booklet} { book }
1685
1686 FUNCTION {collection}
1687 { "G" set.entry.mark
1688   monograph
1689 }
1690
1691 FUNCTION {database}
1692 { "DB" set.entry.mark
1693   electronic
1694 }
1695
1696 FUNCTION {dataset}
1697 { "DS" set.entry.mark
1698   electronic
1699 }
1700

```

An inbook is a piece of a book: either a chapter and/or a page range. It may CROSSREF a book. If there's no volume field, the type field will come before number and series.

Required: author or editor, title, chapter and/or pages, publisher, year

Optional: volume or number, series, type, address, edition, month, note

inbook 类是不含 booktitle 域的，所以不应该适用于“专著中的析出文献”，而应该是专著，即 book 类。

```

1701 FUNCTION {inbook} { book }
1702

```

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

```
1703 FUNCTION {inproceedings}
1704 { "C" set.entry.mark
1705   incollection
1706 }
1707
```

The conference function is included for Scribe compatibility.

```
1708 FUNCTION {conference} { inproceedings }
1709
1710 FUNCTION {map}
1711 { "CM" set.entry.mark
1712   misc
1713 }
1714
```

A manual is technical documentation.

Required: title

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

```
1715 FUNCTION {manual} { monograph }
1716
```

A mastersthesis is a Master's thesis.

Required: author, title, school, year

Optional: type, address, month, note

```
1717 FUNCTION {mastersthesis}
1718 <!*thu>
1719 { "D" set.entry.mark
1720 </!thu>
1721 <!*thu>
1722 { lang.zh entry.lang =
1723   { " 硕士学位论文" }
1724   { "D" }
1725   if$
1726   set.entry.mark
1727 </thu>
1728   monograph
1729 }
1730
1731 FUNCTION {newspaper}
1732 { "N" set.entry.mark
1733   article
1734 }
1735
1736 FUNCTION {online}
1737 { "EB" set.entry.mark
1738   electronic
1739 }
1740
```

A phdthesis is like a mastersthesis.

Required: author, title, school, year

Optional: type, address, month, note

```
1741 <!*thu>
1742 FUNCTION {phdthesis} { mastersthesis }
1743 </!thu>
1744 <!*thu>
1745 FUNCTION {phdthesis}
1746 { lang.zh entry.lang =
1747   { " 博士学位论文" }
1748   { "D" }
1749   if$
```

```

1750 set.entry.mark
1751 monograph
1752 }
1753 \</thu>
1754

```

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

```

1755 FUNCTION {proceedings}
1756 { "C" set.entry.mark
1757   monograph
1758 }
1759
1760 FUNCTION {software}
1761 { "CP" set.entry.mark
1762   electronic
1763 }
1764
1765 FUNCTION {standard}
1766 { "S" set.entry.mark
1767   misc
1768 }
1769

```

A techreport is a technical report.

Required: author, title, institution, year

Optional: type, number, address, month, note

```

1770 FUNCTION {techreport}
1771 { "R" set.entry.mark
1772   misc
1773 }
1774

```

An unpublished is something that hasn't been published.

Required: author, title, note

Optional: month, year

```

1775 FUNCTION {unpublished}
1776 { "Z" set.entry.mark
1777   misc
1778 }
1779

```

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

```

1780 FUNCTION {default.type} { misc }
1781

```

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

```

1782 MACRO {jan} {"January"}
1783
1784 MACRO {feb} {"February"}
1785

```

```

1786 MACRO {mar} {"March"}
1787
1788 MACRO {apr} {"April"}
1789
1790 MACRO {may} {"May"}
1791
1792 MACRO {jun} {"June"}
1793
1794 MACRO {jul} {"July"}
1795
1796 MACRO {aug} {"August"}
1797
1798 MACRO {sep} {"September"}
1799
1800 MACRO {oct} {"October"}
1801
1802 MACRO {nov} {"November"}
1803
1804 MACRO {dec} {"December"}
1805

```

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

```

1806 MACRO {acmcs} {"ACM Computing Surveys"}
1807
1808 MACRO {acta} {"Acta Informatica"}
1809
1810 MACRO {cacm} {"Communications of the ACM"}
1811
1812 MACRO {ibmjrd} {"IBM Journal of Research and Development"}
1813
1814 MACRO {ibmsj} {"IBM Systems Journal"}
1815
1816 MACRO {ieeese} {"IEEE Transactions on Software Engineering"}
1817
1818 MACRO {ieeetc} {"IEEE Transactions on Computers"}
1819
1820 MACRO {ieeetcad}
1821 {"IEEE Transactions on Computer-Aided Design of Integrated Circuits"}
1822
1823 MACRO {ipl} {"Information Processing Letters"}
1824
1825 MACRO {jacm} {"Journal of the ACM"}
1826
1827 MACRO {jcss} {"Journal of Computer and System Sciences"}
1828
1829 MACRO {scp} {"Science of Computer Programming"}
1830
1831 MACRO {sicomp} {"SIAM Journal on Computing"}
1832
1833 MACRO {tocs} {"ACM Transactions on Computer Systems"}
1834
1835 MACRO {tods} {"ACM Transactions on Database Systems"}
1836
1837 MACRO {tog} {"ACM Transactions on Graphics"}
1838
1839 MACRO {toms} {"ACM Transactions on Mathematical Software"}
1840
1841 MACRO {toois} {"ACM Transactions on Office Information Systems"}

```

```

1842
1843 MACRO {toplas} {"ACM Transactions on Programming Languages and Systems"}
1844
1845 MACRO {tcs} {"Theoretical Computer Science"}
1846

```

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`.

```

1847 FUNCTION {sortify}
1848 { purify$
1849   "l" change.case$
1850 }
1851

```

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

```

1852 FUNCTION {chop.word}
1853 { 's :=
1854   'len :=
1855   s #1 len substring$ =
1856   { s len #1 + global.max$ substring$ }
1857   's
1858   if$
1859 }
1860

```

The `format.lab.names` function makes a short label by using the initials of the von and Last parts of the names (but if there are more than four names, (i.e., people) it truncates after three and adds a superscripted "+"; it also adds such a "+" if the last of multiple authors is "others"). If there is only one name, and its von and Last parts combined have just a single name-token ("Knuth" has a single token, "Brinch Hansen" has two), we take the first three letters of the last name. The boolean `et.al.char.used` tells whether we've used a superscripted "+", so that we know whether to include a LaTeX macro for it.

```

format.lab.names(s) ==
BEGIN
  numnames := num.names$(s)
  if numnames > 1 then
    if numnames > 4 then
      namesleft := 3
    else
      namesleft := numnames
    nameptr := 1
    namerresult := ""
    while namesleft > 0
    do
      if (name_ptr = numnames) and
        format.name$(s, nameptr, "{ff }{vv }{ll}{ jj}") = "others"
      then namerresult := namerresult * "{\etalchar{+}}"
        et.al.char.used := true
      else namerresult := namerresult *
        format.name$(s, nameptr, "{v}{l}")
        nameptr := nameptr + 1
        namesleft := namesleft - 1
    od
  if numnames > 4 then
    namerresult := namerresult * "{\etalchar{+}}"
    et.al.char.used := true

```

```

else
  t := format.name$(s, 1, "{v~}{l}")
  if text.length$(t) < 2 then % there's just one name-token
    namerresult := text.prefix$(format.name$(s,1,"{ll}"),3)
  else
    namerresult := t
  fi
fi
return namerresult
END

```

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

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

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

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

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

```

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

```

```

1861 FUNCTION {format.lab.names}
1862 { 's :=
1863 s #1 "{vv~}{ll}{, jj}{, ff}" format.name$ 't :=
1864 t get.str.lang 'name.lang :=
1865 name.lang lang.en =
1866 { t #1 "{vv~}{ll}" format.name$}
1867 { t #1 "{ll}{ff}" format.name$}
1868 if$

```

```

1869 s num.names$ #1 >
1870 { bbl.space * citation.et.al * }
1871 'skip$
1872 if$
1873 }
1874
1875 FUNCTION {author.key.label}
1876 { author empty$
1877   { key empty$
1878     { cite$ #1 #3 substring$ }
1879     'key
1880     if$
1881   }
1882   { author format.lab.names }
1883   if$
1884 }
1885
1886 FUNCTION {author.editor.key.label}
1887 { author empty$
1888   { editor empty$
1889     { key empty$
1890       { cite$ #1 #3 substring$ }
1891       'key
1892       if$
1893     }
1894     { editor format.lab.names }
1895     if$
1896   }
1897   { author format.lab.names }
1898   if$
1899 }
1900
1901 FUNCTION {author.key.organization.label}
1902 { author empty$
1903   { key empty$
1904     { organization empty$
1905       { cite$ #1 #3 substring$ }
1906       { "The " #4 organization chop.word #3 text.prefix$ }
1907       if$
1908     }
1909     'key
1910     if$
1911   }
1912   { author format.lab.names }
1913   if$
1914 }
1915
1916 FUNCTION {editor.key.organization.label}
1917 { editor empty$
1918   { key empty$
1919     { organization empty$
1920       { cite$ #1 #3 substring$ }
1921       { "The " #4 organization chop.word #3 text.prefix$ }
1922       if$
1923     }
1924     'key
1925     if$
1926   }
1927   { editor format.lab.names }
1928   if$
1929 }
1930
1931 FUNCTION {calc.short.authors}
1932 { type$ "book" =

```

```

1933 type$ "inbook" =
1934 or
1935   'author.editor.key.label
1936   { type$ "collection" =
1937     type$ "proceedings" =
1938     or
1939     { editor.empty$ not
1940       'editor.key.organization.label
1941       'author.key.organization.label
1942     }
1943     'author.key.label
1944   }
1945   if$
1946 }
1947 if$
1948 'short.list :=
1949 }
1950
1951 FUNCTION {calc.label}
1952 { calc.short.authors
1953   short.list
1954   "("
1955   *
1956   format.year duplicate$ empty$
1957   short.list key field.or.null = or
1958   { pop$ "" }
1959   'skip$
1960   if$
1961   *
1962   'label :=
1963 }
1964

```

B.8 Sorting

When sorting, we compute the sortkey by executing "presort" on each entry. The presort key contains a number of "sortify"ed strings, concatenated with multiple blanks between them. This makes things like "brinch per" come before "brinch hansen per".

The fields used here are: the `sort.label` for alphabetic labels (as set by `calc.label`), followed by the author names (or editor names or organization (with a leading "The " removed) or key field, depending on entry type and on what's empty), followed by year, followed by the first bit of the title (chopping off a leading "The ", "A ", or "An "). Names are formatted: Von Last First Junior. The names within a part will be separated by a single blank (such as "brinch hansen"), two will separate the name parts themselves (except the von and last), three will separate the names, four will separate the names from year (and from label, if alphabetic), and four will separate year from title.

The `sort.format.names` function takes an argument that should be in BibTeX name format, and returns a string containing " "-separated names in the format described above. The function is almost the same as `format.names`.

```

1965 (*authorityyear)
1966 FUNCTION {sort.language.label}
1967 { entry.lang lang.zh =
1968   { lang.zh.order }
1969   { entry.lang lang.ja =
1970     { lang.ja.order }
1971     { entry.lang lang.en =
1972       { lang.en.order }
1973       { entry.lang lang.ru =

```



```

1974         { lang.ru.order }
1975         { lang.other.order }
1976     if$
1977     }
1978     if$
1979     }
1980     if$
1981     }
1982     if$
1983     int.to.chr$
1984 }
1985
1986 FUNCTION {sort.format.names}
1987 { 's :=
1988   #1 'nameptr :=
1989   ""
1990   s num.names$ 'numnames :=
1991   numnames 'namesleft :=
1992   { namesleft #0 > }
1993   {
1994     s nameptr "{vv{ }}{ll{ }}{ ff{ }}{ jj{ }}" format.name$ 't :=
1995     nameptr #1 >
1996     {
1997       " " *
1998       namesleft #1 = t "others" = and
1999       { "zzzz" * }
2000       { numnames #2 > nameptr #2 = and
2001         { "zz" * year field.or.null * " " * }
2002         'skip$
2003         if$
2004         t sortify *
2005       }
2006     }
2007     { t sortify * }
2008     if$
2009     nameptr #1 + 'nameptr :=
2010     namesleft #1 - 'namesleft :=
2011   }
2012 }
2013 while$
2014 }
2015

```

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`

```

2016 FUNCTION {sort.format.title}
2017 { 't :=
2018   "A " #2
2019   "An " #3
2020   "The " #4 t chop.word
2021   chop.word
2022   chop.word
2023   sortify
2024   #1 global.max$ substring$
2025 }
2026

```

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.

```

2027 FUNCTION {anonymous.sort}
2028 { entry.lang lang.zh =
2029   { "yi4 ming2" }

```

```

2030     { "anon" }
2031   if$
2032 }
2033
2034 FUNCTION {warn.empty.key}
2035 { entry.lang lang.zh =
2036   { "empty key in " cite$ * warning$ }
2037   'skip$
2038   if$
2039 }
2040
2041 FUNCTION {author.sort}
2042 { key empty$
2043   { warn.empty.key
2044     author empty$
2045       { anonymous.sort }
2046       { author sort.format.names }
2047     if$
2048   }
2049   { key sortify }
2050   if$
2051 }
2052
2053 FUNCTION {author.editor.sort}
2054 { key empty$
2055   { warn.empty.key
2056     author empty$
2057       { editor empty$
2058         { anonymous.sort }
2059         { editor sort.format.names }
2060       if$
2061     }
2062     { author sort.format.names }
2063   if$
2064 }
2065 { key sortify }
2066 if$
2067 }
2068
2069 FUNCTION {author.organization.sort}
2070 { key empty$
2071   { warn.empty.key
2072     author empty$
2073       { organization empty$
2074         { anonymous.sort }
2075         { "The " #4 organization chop.word sortify }
2076       if$
2077     }
2078     { author sort.format.names }
2079   if$
2080 }
2081 { key sortify }
2082 if$
2083 }
2084
2085 FUNCTION {editor.organization.sort}
2086 { key empty$
2087   { warn.empty.key
2088     editor empty$
2089       { organization empty$
2090         { anonymous.sort }
2091         { "The " #4 organization chop.word sortify }
2092       if$
2093     }

```

```

2094     { editor sort.format.names }
2095     if$
2096   }
2097   { key sortify }
2098   if$
2099 }
2100
2101 </authoryear>

```

顺序编码制的排序要简单得多

```

2102 (*numerical)
2103 INTEGERS { seq.num }
2104
2105 FUNCTION {init.seq}
2106 { #0 'seq.num :=}
2107
2108 FUNCTION {int.to.fix}
2109 { "00000000" swap$ int.to.str$ *
2110   #-1 #10 substring$
2111 }
2112
2113 </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!

```

2114 FUNCTION {presort}
2115 { set.entry.lang
2116   set.entry.numbered
2117   show.url show.doi check.electronic
2118   calc.label
2119   label sortify
2120   " "
2121   *
2122 (*authoryear)
2123   sort.language.label
2124   type$ "book" =
2125   type$ "inbook" =
2126   or
2127     'author.editor.sort
2128     { type$ "collection" =
2129       type$ "proceedings" =
2130       or
2131         'editor.organization.sort
2132         'author.sort
2133       if$
2134     }
2135   if$
2136   *
2137   " "
2138   *
2139   year field.or.null sortify
2140   *
2141   " "
2142   *
2143   cite$
2144   *
2145   #1 entry.max$ substring$
2146 </authoryear>
2147 (*numerical)
2148   seq.num #1 + 'seq.num :=
2149   seq.num int.to.fix
2150 </numerical>

```

```

2151 'sort.label :=
2152 sort.label *
2153 #1 entry.max$ substrings$
2154 'sort.key$ :=
2155 }
2156

```

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

```

```

2157 STRINGS { longest.label last.label next.extra }
2158
2159 INTEGERS { longest.label.width last.extra.num number.label }
2160
2161 FUNCTION {initialize.longest.label}
2162 { "" 'longest.label :=
2163 #0 int.to.chr$ 'last.label :=
2164 "" 'next.extra :=
2165 #0 'longest.label.width :=
2166 #0 'last.extra.num :=
2167 #0 'number.label :=
2168 }
2169
2170 FUNCTION {forward.pass}

```

```

2171 { last.label label =
2172   { last.extra.num #1 + 'last.extra.num :=
2173     last.extra.num int.to.chr$ 'extra.label :=
2174   }
2175   { "a" chr.to.int$ 'last.extra.num :=
2176     "" 'extra.label :=
2177     label 'last.label :=
2178   }
2179   if$
2180   number.label #1 + 'number.label :=
2181 }
2182
2183 FUNCTION {reverse.pass}
2184 { next.extra "b" =
2185   { "a" 'extra.label := }
2186   'skip$
2187   if$
2188   extra.label 'next.extra :=
2189   extra.label
2190   duplicate$ empty$
2191   'skip$
2192   { "{\natexlab{" swap$ * "}" * }
2193   if$
2194   'extra.label :=
2195   label extra.label * 'label :=
2196 }
2197
2198 FUNCTION {bib.sort.order}
2199 { sort.label 'sort.key$ :=
2200 }
2201

```

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.

```

2202 FUNCTION {begin.bib}
2203 { preamble$ empty$
2204   'skip$
2205   { preamble$ write$ newline$ }
2206   if$
2207   "\begin{thebibliography}{ number.label int.to.str$ * "}" *
2208   write$ newline$
2209   "\providecommand{\natexlab}[1]{#1}"
2210   write$ newline$
2211   "\providecommand{\url}[1]{#1}"
2212   write$ newline$
2213   "\expandafter\ifx\curname urlstyle\endcsname\relax\relax\else"
2214   write$ newline$
2215   " \urlstyle{same}\fi"
2216   write$ newline$
2217   show.doi
2218   { "\providecommand{\href}[2]{\url{#2}}"
2219     write$ newline$
2220     "\providecommand{\doi}[1]{\href{https://doi.org/#1}{#1}}"
2221     write$ newline$
2222   }
2223   'skip$

```

```
2224 if$
2225 }
2226
```

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

```
2227 FUNCTION {end.bib}
2228 { newline$
2229   "\end{thebibliography}" write$ newline$
2230 }
2231
```

B.10 Main execution

Now we read in the .BIB entries.

```
2232 READ
2233
2234 EXECUTE {init.state.consts}
2235
2236 EXECUTE {load.config}
2237
2238 (*numerical)
2239 EXECUTE {init.seq}
2240
2241 
```

And now we can sort

```
2244 SORT
2245
2246 EXECUTE {initialize.longest.label}
2247
2248 ITERATE {forward.pass}
2249
2250 REVERSE {reverse.pass}
2251
2252 ITERATE {bib.sort.order}
2253
2254 SORT
2255
2256 EXECUTE {begin.bib}
2257
```

Now we produce the output for all the entries

```
2258 ITERATE {call.type$}
2259
2260 EXECUTE {end.bib}
2261 
```