

GB/T 7714 BibTeX style

Zeping Lee*

2025/06/22 v2.1.8

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.preprint	#1	显示预印本信息
show.note	#0	显示 note 域的信息
end.with.period	#1	结尾加句点
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           \fi
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
    将页码放在括号外边，并且置于上标。
294     \if@gbt@locator@inside@affixes
295     \if*#2*\else\NAT@cmt#2\fi
296     \if\relax\NAT@date\relax\else\NAT@@close\fi
297 \else
298     \if\relax\NAT@date\relax\else\NAT@@close\fi
299     \if*#2*\else\textsuperscript{#2}\fi
300 \fi
301 \fi}{#1}{#2}}

```

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.preprint
375     show.note
376     show.english.translation
377     end.with.period
378     lowercase.word.after.colon

```

```

379 <*author-year>
380   lang.zh.order
381   lang.ja.order
382   lang.en.order
383   lang.ru.order
384   lang.other.order
385 </author-year>
386 }
387
388 STRINGS {
389   component.part.label
390   page.range.delimiter
391 }
392

```

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。

```

393 FUNCTION {load.config}
394 {

```

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”。

```

395 <*!ucas>
396   #2 'citation.et.al.min :=
397   #1 'citation.et.al.use.first :=
398 </!ucas>
399 <*ucas>
400   #3 'citation.et.al.min :=
401   #1 'citation.et.al.use.first :=
402 </ucas>
403   #4 'bibliography.et.al.min :=
404   #3 'bibliography.et.al.use.first :=

```

英文姓名转为全大写：

```

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

```

使用 TeX 宏输出“和”、“等”

```

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

```

将年份置于著者后面（著者-出版年制默认）

```

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

```

```

421   #1 'year.after.author :=
422 </author-year&!ucas>

```

采用著者-出版年制时，作者姓名与年份之间使用句点连接：

```

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

```

书名使用斜体：

```

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

```

英文标题转为 sentence case（句首字母大写，其余小写）：

```

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

```

在标题添加超链接：

```

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

```

期刊是否含标题：

```

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

```

专利题名是否含专利国别

```

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

```

著录文献类型标识（比如“[M/OL]”）：

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

文献类型标识前是否有空格：

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

是否显示载体类型标识（比如“/OL”）：

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

使用“//”表示析出文献

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

期刊名使用缩写：

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

期刊名使用斜体：

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

在期刊题名添加超链接：

```
503 <!*link-journal>
504     #0 'link.journal :=
```

```

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

```

期刊的卷使用粗体:

```

509   #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.”:

```

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

```

页码与前面的冒号之间是否有空格:

```

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

```

页码是否只含起始页:

```

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

```

起止页码中的连接号:

```

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

```

是否著录非电子文献的引用日期:

```

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

```

是否著录 URL:

```

543 <*(no-url | ustc)>
544   #1 'show.url :=

```

```

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

```

是否著录 DOI:

```

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

```

是否著录 e-print:

```

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

```

在每一条文献最后输出注释 (note) 的内容:

```

561     #0 'show.note :=

```

中文文献是否显示英文翻译

```

562 <!*show-english-translation>
563     #0 'show.english.translation :=
564 </!*show-english-translation>
565 <*show-english-translation>
566     #1 'show.english.translation :=
567 </show-english-translation>

```

结尾加句点

```

568 <!*no-period-at-end>
569     #1 'end.with.period :=
570 </!*no-period-at-end>
571 <*no-period-at-end>
572     #0 'end.with.period :=
573 </no-period-at-end>

```

将冒号后的单词变成小写

```

574 <!*no-lowercase-word-after-colon>
575     #1 'lowercase.word.after.colon :=
576 </!*no-lowercase-word-after-colon>
577 <*no-lowercase-word-after-colon>
578     #0 'lowercase.word.after.colon :=
579 </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:

```

580 <*author-year>
581     #1 'lang.zh.order :=
582     #2 'lang.ja.order :=
583     #3 'lang.en.order :=
584     #4 'lang.ru.order :=

```

```

585     #5 'lang.other.order :=
586 </author-year>
587 }
588

```

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.

```

589 ENTRY
590 { address
591   archivePrefix
592   author
593   booktitle
594   date
595   doi
596   edition
597   editor
598   eprint
599   eprinttype
600   entrysubtype
601   howpublished
602   institution
603   journal
604   journaltitle
605   key
606   langid
607   language
608   location
609   mark
610   medium
611   note
612   number
613   organization
614   pages
615   publisher
616   school
617   series
618   shortjournal
619   title
620   translation
621   translator
622   url
623   urldate
624   volume
625   year
626 }
627 { 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.

```
628 { label extra.label sort.label short.list entry.mark entry.url }  
629
```

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() * ", ")
```


[illegible]

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

```

```

630 INTEGERS { output.state before.all mid.sentence after.sentence after.block
631
632 INTEGERS { lang.zh lang.ja lang.en lang.ru lang.other }
633
634 INTEGERS { charptr len }
635
636 FUNCTION {init.state.consts}
637 { #0 'before.all :=
638   #1 'mid.sentence :=
639   #2 'after.sentence :=
640   #3 'after.block :=
641   #4 'after.slash :=
642   #3 'lang.zh :=
643   #4 'lang.ja :=
644   #1 'lang.en :=
645   #2 'lang.ru :=
646   #0 'lang.other :=

```

```

647 }
648
        下面是一些常量的定义
649 FUNCTION {bbl.anonymous}
650 { entry.lang lang.zh =
651   { " 佚名" }
652   { "Anon" }
653   if$
654 }
655
656 FUNCTION {bbl.space}
657 { entry.lang lang.zh =
658   { "\ " }
659   { " " }
660   if$
661 }
662
663 FUNCTION {bbl.and}
664 { "" }
665
666 FUNCTION {bbl.et.al}
667 { entry.lang lang.zh =
668   { " 等" }
669   { entry.lang lang.ja =
670     { " 他" }
671     { entry.lang lang.ru =
672       { " " }
673       { "et~al." }
674       if$
675     }
676     if$
677   }
678   if$
679 }
680
681 FUNCTION {citation.and}
682 { terms.in.macro
683   { "{\biband}" }
684   'bbl.and
685   if$
686 }
687
688 FUNCTION {citation.et.al}
689 { terms.in.macro
690   { "{\bibetal}" }
691   'bbl.et.al
692   if$
693 }
694
695 FUNCTION {bbl.colon} { ": " }
696
697 FUNCTION {bbl.pages.colon}
698 { space.before.pages
699   { ": " }

```

[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

```

742 FUNCTION {not}
743 {
744   { #0 }
745   { #1 }
746   if$
747 }
748 FUNCTION {and}
749 {
750   'skip$

```

```

750     { pop$ #0 }
751   if$
752 }
753
754 FUNCTION {or}
755 {   { pop$ #1 }
756     'skip$
757   if$
758 }
759
760 STRINGS { x y }
761
762 FUNCTION {contains}
763 { 'y :=
764   'x :=
765   y text.length$ 'len :=
766   x text.length$ len - #1 + 'charptr :=
767   { charptr #0 >
768     x charptr len substring$ y = not
769     and
770   }
771   { charptr #1 - 'charptr := }
772   while$
773   charptr #0 >
774 }
775

```

the variables s and t are temporary string holders

```

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

```

```

803 }
804
805 FUNCTION {output}
806 { duplicate$ empty$
807   'pop$
808   'output.nonnull
809   if$
810 }
811
812 FUNCTION {output.after}
813 { 't :=
814   duplicate$ empty$
815   'pop$
816   { 's :=
817     output.state mid.sentence =
818     { t * write$ }
819     { output.state after.block =
820       { add.period$ write$
821         newline$
822         "\newblock " write$
823       }
824       { output.state before.all =
825         'write$
826         { output.state after.slash =
827           { bbl.slash * write$ }
828           { add.period$ " " * write$ }
829         if$
830       }
831       if$
832     }
833     if$
834     mid.sentence 'output.state :=
835   }
836   if$
837   s
838 }
839 if$
840 }
841
842 FUNCTION {output.check}
843 { 't :=
844   duplicate$ empty$
845   { pop$ "empty " t * " in " * cite$ * warning$ }
846   'output.nonnull
847   if$
848 }
849

```

This function finishes all entries.

```

850 FUNCTION {fin.entry}
851 { end.with.period
852   'add.period$
853   'skip$
854   if$
855   write$

```

```

856     show.english.translation entry.lang lang.zh = and
857         { ")"
858             write$
859         }
860     'skip$
861     if$
862     newline$
863 }
864
865 FUNCTION {new.block}
866 { output.state before.all =
867     'skip$
868     { output.state after.slash =
869         'skip$
870         { after.block 'output.state := }
871     if$
872 }
873 if$
874 }
875
876 FUNCTION {new.sentence}
877 { output.state after.block =
878     'skip$
879     { output.state before.all =
880         'skip$
881         { output.state after.slash =
882             'skip$
883             { after.sentence 'output.state := }
884         if$
885     }
886     if$
887 }
888 if$
889 }
890
891 FUNCTION {new.slash}
892 { output.state before.all =
893     'skip$
894     { component.part.label "slash" =
895         { after.slash 'output.state := }
896         { new.block
897             component.part.label "in" =
898                 { entry.lang lang.en =
899                     { "In: " output
900                         write$
901                         ""
902                         before.all 'output.state :=
903                     }
904                     'skip$
905                 if$
906             }
907             'skip$
908         if$
909     }
910     if$

```

```

911     }
912     if$
913 }
914

```

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.

```

915 FUNCTION {new.block.checka}
916 { empty$
917   'skip$
918   'new.block
919   if$
920 }
921
922 FUNCTION {new.block.checkb}
923 { empty$
924   swap$ empty$
925   and
926   'skip$
927   'new.block
928   if$
929 }
930

```

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

```

931 FUNCTION {new.sentence.checka}
932 { empty$
933   'skip$
934   'new.sentence
935   if$
936 }
937
938 FUNCTION {new.sentence.checkb}
939 { empty$
940   swap$ empty$
941   and
942   'skip$
943   'new.sentence
944   if$
945 }
946

```

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.

```

947 INTEGERS { b }
948

```


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

```

949 % codepoint: int, a: int, b: int -> bool
950 % variable used: b
951 FUNCTION {is.int.in.range}
952 {
953   'b :=
954   #1 +
955   b >
956   { #1 - b < }
957   { pop$ #0 }
958   if$
959 }
960

```

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

```

961 % m: int, n: int -> int
962 FUNCTION {mult.power2}
963 {
964   { duplicate$ #0 > }
965   {
966     swap$
967     duplicate$ +
968     swap$ #1 -
969   }
970   while$
971   pop$
972 }
973

```

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 **B_WTeX**.

```

974 % "{", str -> subtoken: str, rest: str
975 % variables used: s, t
976 FUNCTION {find.match.brace}
977 {
978   's :=
979   't :=
980
981   #1
982   { duplicate$ #0 >
983     s empty$ not and }
984   {
985     s #1 #1 substring$ "{" =
986     { #1 + }
987     {
988       s #1 #1 substring$ "}" =
989       { #1 - }
990       'skip$

```

```

991         if$
992     }
993     if$
994     t s #1 #1 substring$ * 't :=
995     s #2 global.max$ substring$ 's :=
996 }
997 while$
998
999 duplicate$ #0 >
1000 {
1001     "Unbalanced brace(s): one or more closing braces are missing" warning
1002     { duplicate$ #0 > }
1003     {
1004         t "}" * 't :=
1005         #1 -
1006     }
1007     while$
1008 }
1009 'skip$
1010 if$
1011 pop$
1012
1013 t
1014 s
1015 }
1016

```

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.

```

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

```

```

1042         }
1043         {
1044             duplicate$ #1 #4 substring$ swap$
1045             #5 global.max$ substring$ swap$
1046         }
1047         if$
1048     }
1049     if$
1050 }
1051 if$
1052 }
1053 if$
1054 }
1055

```

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

```

1056 % str -> char
1057 FUNCTION {get.first.char.from.str}
1058 {
1059     split.first.char.from.str swap$ pop$
1060 }
1061

```

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 $\text{T}_{\text{E}}\text{X}$ group and the rest of string in reverse order.

```

1062 % str -> rest: str, texchar
1063 FUNCTION {split.first.tex.char.from.str}
1064 {
1065     duplicate$ #1 #1 substring$ "{" =
1066     {
1067         split.first.char.from.str swap$
1068         find.match.brace swap$
1069     }
1070     'split.first.char.from.str
1071     if$
1072 }
1073

```

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.

```

1074 % char -> int
1075 FUNCTION {char.to.unicode}
1076 {
1077     duplicate$ #4 #1 substring$ "" =
1078     {
1079         duplicate$ #3 #1 substring$ "" =
1080         {
1081             duplicate$ #2 #1 substring$ "" =
1082             {

```

```

1083         duplicate$ "" =
1084         {
1085             "Empty string is not a char!" warning$
1086             pop$ #-1
1087         }
1088         { #1 #1 substring$ chr.to.int$ }
1089     if$
1090 }
1091 {
1092     duplicate$ #2 #1 substring$ chr.to.int$ #128 - swap$
1093     #1 #1 substring$ chr.to.int$ #192 -
1094     #6 mult.power2 +
1095 }
1096 if$
1097 }
1098 {
1099     duplicate$ #3 #1 substring$ chr.to.int$ #128 - swap$
1100     duplicate$ #2 #1 substring$ chr.to.int$ #128 - swap$
1101     #1 #1 substring$ chr.to.int$ #224 -
1102     #6 mult.power2 +
1103     #6 mult.power2 +
1104 }
1105 if$
1106 }
1107 {
1108     duplicate$ #4 #1 substring$ chr.to.int$ #128 - swap$
1109     duplicate$ #3 #1 substring$ chr.to.int$ #128 - swap$
1110     duplicate$ #2 #1 substring$ chr.to.int$ #128 - swap$
1111     #1 #1 substring$ chr.to.int$ #240 -
1112     #6 mult.power2 +
1113     #6 mult.power2 +
1114     #6 mult.power2 +
1115 }
1116 if$
1117 }
1118

```

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.

```

1119 % str, char -> bool
1120 % variable used: t
1121 FUNCTION {is.char.in.str}
1122 {
1123     't :=
1124
1125     t "" =
1126     { "is.char.in.str: Empty string is not a char!" warning$ }
1127     'skip$
1128     if$
1129
1130     #0 swap$

```

```

1131 { duplicate$ "" = not }
1132 {
1133     split.first.char.from.str t =
1134     { pop$ pop$ #1 "" }
1135     'skip$
1136     if$
1137 }
1138 while$
1139 pop$
1140 }
1141

```

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

```

1142 % char -> bool
1143 % variable used: b
1144 FUNCTION {is.upper.ascii}
1145 {
1146     char.to.unicode #65 swap$ #90 swap$ is.int.in.range
1147 }
1148

```

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

```

1149 % char -> bool
1150 % variable used: b
1151 FUNCTION {is.upper}
1152 {
1153     duplicate$ is.upper.ascii
1154     { pop$ #1 }
1155     { latin.upper swap$ is.char.in.str }
1156     if$
1157 }
1158

```

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

```

1159 % char -> bool
1160 % variable used: b
1161 FUNCTION {is.lower.ascii}
1162 {
1163     char.to.unicode #97 swap$ #122 swap$ is.int.in.range
1164 }
1165

```

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

```

1166 % char -> bool
1167 % variable used: b
1168 FUNCTION {is.lower}
1169 {
1170     duplicate$ is.lower.ascii
1171     { pop$ #1 }

```

```

1172     { latin.lower swap$ is.char.in.str }
1173   if$
1174 }
1175

```

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

```

1176 % char -> bool
1177 % variable used: b
1178 FUNCTION {is.printable.ascii}
1179 {
1180   char.to.unicode #32 swap$ #126 swap$ is.int.in.range
1181 }
1182

```

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

```

1183 % char -> bool
1184 % variable used: b
1185 FUNCTION {is.letter.ascii}
1186 {
1187   duplicate$ is.upper.ascii swap$ is.lower.ascii or
1188 }
1189

```

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

```

1190 % char -> bool
1191 % variable used: b
1192 FUNCTION {is.symbol.ascii}
1193 {
1194   duplicate$ is.printable.ascii swap$ is.letter.ascii not and
1195 }
1196

```

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.

```

1197 % str -> bool
1198 % variable used: b
1199 % return true if str is empty
1200 FUNCTION {is.all.lower}
1201 {
1202   #1 swap$
1203   { duplicate$ "" = not }
1204   {
1205     split.first.char.from.str is.lower
1206     'skip$
1207     { pop$ pop$ #0 "" }
1208     if$
1209   }
1210   while$
1211   pop$
1212 }

```

```

1213
1214 % str -> bool
1215 % variable used: b
1216 FUNCTION {is.tex.str.in.title.case}
1217 {
1218   duplicate$ "" =
1219   { pop$ #0 }
1220   {
1221     split.first.tex.char.from.str purify$
1222     duplicate$ "" =
1223     { pop$ pop$ #0 }
1224     {
1225       split.first.char.from.str is.upper
1226       {
1227         duplicate$ is.all.lower
1228         {
1229           empty$
1230           {
1231             duplicate$ "" =
1232             { pop$ #0 }
1233             'is.all.lower
1234             if$
1235           }
1236           'is.all.lower
1237           if$
1238         }
1239         { pop$ pop$ #0 }
1240         if$
1241       }
1242       { pop$ pop$ #0 }
1243       if$
1244     }
1245     if$
1246   }
1247   if$
1248 }
1249
1250 % char, int -> bool
1251 % variables used: t, b
1252 FUNCTION {is.in.inter.token.chars}
1253 {
1254   duplicate$ #0 =
1255   { pop$ " " = }
1256   {
1257     #1 =
1258     { " " range.delimiters * swap$ is.char.in.str }
1259     'is.letter.ascii
1260     if$
1261   }
1262   if$
1263 }
1264
1265 % str, int -> intertoken: str, rest: str
1266 % variable used: t, b
1267 FUNCTION {skip.inter.token.chars.by}

```

```

1268 {
1269     'b :=
1270     't :=
1271
1272     "" t
1273     { duplicate$ "" = not }
1274     {
1275         split.first.char.from.str
1276         duplicate$ b is.in.inter.token.chars
1277         { swap$ 't := * t }
1278         { swap$ * 't := "" }
1279         if$
1280     }
1281     while$
1282
1283     pop$ t
1284 }
1285
1286 % str -> intertoken: str, rest: str
1287 % variable used: t, b
1288 FUNCTION {skip.inter.token.chars}
1289 {
1290     #0 skip.inter.token.chars.by
1291 }
1292
1293 % str -> intertoken: str, rest: str
1294 % variable used: t, b
1295 FUNCTION {skip.inter.token.command}
1296 {
1297     duplicate$ "" =
1298     { "" }
1299     {
1300         duplicate$ #1 #1 substring$ is.symbol.ascii
1301         { split.first.char.from.str swap$ }
1302         { #2 skip.inter.token.chars.by }
1303         if$
1304     }
1305     if$
1306 }
1307
1308 % cmdstr -> cmdstr
1309 FUNCTION {is.special.char.command}
1310 {
1311     #2 global.max$ substring$ skip.inter.token.command
1312
1313     empty$
1314     'skip$
1315     { "is.special.char.command: cmdstr has extra components!" warning$ }
1316     if$
1317
1318     duplicate$ duplicate$ duplicate$ duplicate$ duplicate$ duplicate$
1319     "oOlLij" swap$ is.char.in.str
1320     swap$ "oe" = or
1321     swap$ "OE" = or
1322     swap$ "ae" = or

```



```

1323 swap$ "AE" = or
1324 swap$ "aa" = or
1325 swap$ "AA" = or
1326 }
1327
1328 % str, str, char -> char
1329 % variable used: t
1330 FUNCTION {map.char}
1331 {
1332   't :=
1333   split.first.char.from.str
1334   { swap$ duplicate$ "" = not }
1335   {
1336     swap$ t =
1337     { pop$ "" t }
1338     {
1339       swap$ split.first.char.from.str pop$ swap$
1340       split.first.char.from.str
1341     }
1342     if$
1343   }
1344   while$
1345   pop$ t =
1346   'get.first.char.from.str
1347   { pop$ t }
1348   if$
1349 }
1350
1351 % char -> char
1352 % variables used: t, b
1353 FUNCTION {to.lower}
1354 {
1355   duplicate$ is.upper.ascii
1356   { chr.to.int$ #32 + int.to.chr$ }
1357   { latin.lower swap$ latin.upper swap$ map.char }
1358   if$
1359 }
1360
1361 % char -> char
1362 % variables used: t, b
1363 FUNCTION {to.upper}
1364 {
1365   duplicate$ is.lower.ascii
1366   { chr.to.int$ #32 - int.to.chr$ }
1367   { latin.upper swap$ latin.lower swap$ map.char }
1368   if$
1369 }
1370
1371 % str -> str
1372 % variables used: t, b
1373 FUNCTION {all.to.lower}
1374 {
1375   "" swap$
1376   { duplicate$ empty$ not }
1377   { split.first.char.from.str to.lower swap$ 't := * t }

```

```

1378 while$
1379 *
1380 }
1381
1382 % texchar -> texchar
1383 % variables used: t, b
1384 FUNCTION {command.to.lower}
1385 {
1386   duplicate$ "" =
1387   { "command.to.lower: Empty string is not a texchar!" warning$ }
1388   {
1389     duplicate$ #1 #1 substring$ #92 int.to.chr$ =
1390     {
1391       duplicate$ is.special.char.command
1392       'all.to.lower
1393       'skip$
1394       if$
1395     }
1396     'to.lower
1397     if$
1398   }
1399   if$
1400 }
1401
1402 % texchar -> texchar
1403 % variables used: t, b
1404 FUNCTION {tex.to.lower}
1405 {
1406   duplicate$ #1 #2 substring$ "{" #92 int.to.chr$ * =
1407   {
1408     "" swap$
1409     { duplicate$ "" = not }
1410     {
1411       split.first.char.from.str
1412       duplicate$ #92 int.to.chr$ =
1413       {
1414         swap$ skip.inter.token.command 't := * t
1415         swap$ command.to.lower
1416       }
1417       'to.lower
1418       if$
1419       swap$ 't := * t
1420     }
1421     while$
1422     pop$
1423   }
1424   {
1425     duplicate$ #1 #1 substring$ "{" =
1426     { split.first.char.from.str swap$ find.match.brace pop$ }
1427     'command.to.lower
1428     if$
1429   }
1430   if$
1431 }
1432

```

```

1433 % str -> str
1434 % variables used: t, b
1435 FUNCTION {all.to.upper}
1436 {
1437   "" swap$
1438   { duplicate$ empty$ not }
1439   { split.first.char.from.str to.upper swap$ 't := * t }
1440   while$
1441   *
1442 }
1443
1444 % texchar -> texchar
1445 % variables used: t, b
1446 FUNCTION {command.to.upper}
1447 {
1448   duplicate$ "" =
1449   { "command.to.lower: Empty string is not a texchar!" warning$ }
1450   {
1451     duplicate$ #1 #1 substring$ #92 int.to.chr$ =
1452     {
1453       duplicate$ is.special.char.command
1454       'all.to.upper
1455       'skip$
1456       if$
1457     }
1458     'to.upper
1459     if$
1460   }
1461   if$
1462 }
1463
1464 % texchar -> texchar
1465 % variables used: t, b
1466 FUNCTION {tex.to.upper}
1467 {
1468   duplicate$ #1 #2 substring$ "{" #92 int.to.chr$ * =
1469   {
1470     "" swap$
1471     { duplicate$ "" = not }
1472     {
1473       split.first.char.from.str
1474       duplicate$ #92 int.to.chr$ =
1475       {
1476         swap$ skip.inter.token.command 't := * t
1477         swap$ command.to.upper
1478       }
1479       'to.upper
1480       if$
1481       swap$ 't := * t
1482     }
1483     while$
1484     pop$
1485   }
1486   {
1487     duplicate$ #1 #1 substring$ "{" =

```

```

1488         { split.first.char.from.str swap$ find.match.brace pop$ }
1489         'command.to.upper
1490     if$
1491 }
1492 if$
1493 }
1494
1495 % texstr -> texstr
1496 % variable used: t, b
1497 FUNCTION {lower.token.if.in.title.case}
1498 {
1499     duplicate$ is.tex.str.in.title.case
1500     { split.first.tex.char.from.str tex.to.lower swap$ * }
1501     'skip$
1502     if$
1503 }
1504
1505 % int -> str
1506 FUNCTION {self.tokens}
1507 {
1508     #0 =
1509     'default.self.tokens
1510     'range.delimiters
1511     if$
1512 }
1513
1514 % str, int -> token: str, rest: str
1515 % variables used: s, t, b
1516 FUNCTION {tokenize.by}
1517 {
1518     'b :=
1519     's :=
1520
1521     s "" =
1522     { "" "" }
1523     {
1524         s split.first.char.from.str
1525         duplicate$ b self.tokens swap$ is.char.in.str
1526         'swap$
1527         {
1528             duplicate$ #92 int.to.chr$ =
1529             { swap$ skip.inter.token.command 's := * s }
1530             {
1531                 pop$ pop$ "" s
1532                 { duplicate$ "" = not }
1533                 {
1534                     split.first.char.from.str
1535                     duplicate$ "\ " b self.tokens * swap$ is.char.in.str
1536                     { pop$ pop$ "" }
1537                     {
1538                         duplicate$ "{" =
1539                         { swap$ find.match.brace }
1540                         'swap$
1541                         if$
1542                         's := * s

```

```

1543         }
1544         if$
1545     }
1546     while$
1547     pop$ s
1548 }
1549 if$
1550 }
1551 if$
1552 }
1553 if$
1554 }
1555
1556 % str -> str
1557 % variables used: s, t, b
1558 FUNCTION {tokenize}
1559 {
1560     #0 tokenize.by
1561 }
1562
1563 % str -> str
1564 % variables used: s, t, b
1565 FUNCTION {smart.sentence.case}
1566 {
1567     tokenize 's :=
1568
1569     { s "" = not }
1570     {
1571         s skip.inter.token.chars 's := * s
1572         tokenize swap$
1573         duplicate$ ":" =
1574         {
1575             swap$ 's := *
1576             s skip.inter.token.chars 's := * s
1577             tokenize swap$
1578             lowercase.word.after.colon
1579             {
1580                 duplicate$ "A" =
1581                 { pop$ "a" }
1582                 'lower.token.if.in.title.case
1583                 if$
1584             }
1585             'skip$
1586             if$
1587         }
1588         'lower.token.if.in.title.case
1589         if$
1590         swap$ 's := *
1591     }
1592     while$
1593 }
1594
1595 % str -> str
1596 % variables used: s, t, b
1597 FUNCTION {smart.upper.case}

```

```

1598 {
1599   s swap$ t swap$
1600
1601   "" swap$
1602   { duplicate$ "" = not }
1603   {
1604     tokenize swap$
1605     duplicate$ #1 #1 substring$ #92 int.to.chr$ =
1606     'command.to.upper
1607     {
1608       "" swap$
1609       { duplicate$ "" = not }
1610       {
1611         split.first.tex.char.from.str tex.to.upper
1612         swap$ 't := * t
1613       }
1614       while$
1615       pop$
1616     }
1617     if$
1618     swap$ 't := * t
1619     skip.inter.token.chars 't := * t
1620   }
1621   while$
1622   pop$
1623
1624   swap$ 't :=
1625   swap$ 's :=
1626 }
1627

```

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

```
1628 FUNCTION {field.or.null}
1629 { duplicate$ empty$
1630   { pop$ "" }
1631   'skip$
1632   if$
1633 }
1634
1635 FUNCTION {emphasize}
1636 { duplicate$ empty$
1637   { pop$ "" }
1638   { "\emph{" swap$ * "}" * }
1639   if$
1640 }
1641
1642 FUNCTION {format.btitle}
1643 { italic.book.title
1644   entry.lang lang.en = and
1645   'emphasize
1646   'skip$
1647   if$
1648 }
1649
```

B.4.1 Detect Language

```
1650 INTEGERS { byte second.byte }
1651
1652 INTEGERS { char.lang tmp.lang }
1653
1654 STRINGS { tmp.str }
1655
1656 FUNCTION {get.str.lang}
1657 { 'tmp.str :=
1658   lang.other 'tmp.lang :=
1659   #1 'charptr :=
1660   tmp.str text.length$ #1 + 'len :=
1661   { charptr len < }
1662   { tmp.str charptr #1 substring$ chr.to.int$ 'byte :=
1663     byte #128 <
1664     { charptr #1 + 'charptr :=
1665       byte #64 > byte #91 < and byte #96 > byte #123 < and or
1666       { lang.en 'char.lang := }
```

```

1667         { lang.other 'char.lang := }
1668     if$
1669 }
1670 { tmp.str charptr #1 + #1 substring$ chr.to.int$ 'second.byte :=
1671     byte #224 <

```

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

```

1672     { charptr #2 + 'charptr :=
1673         byte #207 > byte #212 < and
1674         byte #212 = second.byte #176 < and or
1675         { lang.ru 'char.lang := }
1676         { lang.other 'char.lang := }
1677     if$
1678 }
1679 { byte #240 <

```

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

```

1680     { charptr #3 + 'charptr :=
1681         byte #227 > byte #234 < and
1682         { lang.zh 'char.lang := }

```

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

```

1683         { byte #227 =
1684             { second.byte #143 >
1685                 { lang.zh 'char.lang := }

```

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

```

1686             { second.byte #128 > second.byte #132 < and
1687                 { lang.ja 'char.lang := }
1688                 { lang.other 'char.lang := }
1689             if$
1690         }
1691     if$
1692 }

```

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

```

1693         { byte #239 =
1694             second.byte #163 > second.byte #172 < and and
1695             { lang.zh 'char.lang := }
1696             { lang.other 'char.lang := }
1697         if$
1698     }
1699     if$
1700 }
1701 if$
1702 }

```

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.

```

1703     { charptr #4 + 'charptr :=
1704         byte #240 = second.byte #159 > and
1705         { lang.zh 'char.lang := }
1706         { lang.other 'char.lang := }
1707     if$

```



```

1708         }
1709         if$
1710     }
1711     if$
1712 }
1713 if$
1714 char.lang tmp.lang >
1715 { char.lang 'tmp.lang := }
1716 'skip$
1717 if$
1718 }
1719 while$
1720 tmp.lang
1721 }
1722
1723 FUNCTION {check.entry.lang}
1724 { author field.or.null
1725   title field.or.null *
1726   get.str.lang
1727 }
1728
1729 STRINGS { entry.langid }
1730
1731 FUNCTION {set.entry.lang}
1732 { "" 'entry.langid :=
1733   language empty$ not
1734     { language 'entry.langid := }
1735   'skip$
1736   if$
1737   langid empty$ not
1738     { langid 'entry.langid := }
1739   'skip$
1740   if$
1741   entry.langid empty$
1742     { check.entry.lang }
1743     { entry.langid "english" = entry.langid "american" = or entry.langid "b
1744       { lang.en }
1745       { entry.langid "chinese" =
1746         { lang.zh }
1747         { entry.langid "japanese" =
1748           { lang.ja }
1749           { entry.langid "russian" =
1750             { lang.ru }
1751             { check.entry.lang }
1752             if$
1753           }
1754         if$
1755       }
1756     if$
1757   }
1758   if$
1759 }
1760 if$
1761 'entry.lang :=
1762 }

```

```

1763
1764 FUNCTION {set.entry.numbered}
1765 { type$ "patent" =
1766   type$ "standard" = or
1767   type$ "techreport" = or
1768   { #1 'entry.numbered := }
1769   { #0 'entry.numbered := }
1770   if$
1771 }
1772

```

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}{,ujj}")
    % for abbreviated first names:
    t := format.name$(s, nameptr, "{f.~}{vv~}{ll}{,ujj}")
    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.

```

1773 INTEGERS { nameptr namesleft numnames name.lang }
1774
1775 FUNCTION {format.name}
1776 { "{vv~}{ll}{, jj}{, ff}" format.name$ 't :=
1777   t "others" =
1778     { bbl.et.al }
1779     { t get.str.lang 'name.lang :=
1780       name.lang lang.en =
1781         { t #1 "{vv~}{ll}{ f{~}}" format.name$
1782           uppercase.name
1783           'smart.upper.case
1784           'skip$
1785           if$
1786           t #1 "{, jj}" format.name$ *
1787         }
1788         { t #1 "{ll}{ff}" format.name$ }
1789       if$
1790     }
1791   if$
1792 }
1793
1794 FUNCTION {format.names}
1795 { 's :=
1796   #1 'nameptr :=
1797   s num.names$ 'numnames :=
1798   ""
1799   numnames 'namesleft :=
1800   { namesleft #0 > }
1801   { s nameptr format.name bbl.et.al =
1802     numnames bibliography.et.al.min #1 - > nameptr bibliography.et.al.use

```

```

1803         { ", " *
1804           bbl.et.al *
1805           #1 'namesleft :=
1806         }
1807         { nameptr #1 >
1808           { namesleft #1 = bbl.and "" = not and
1809             { bbl.and * }
1810             { ", " * }
1811           if$
1812         }
1813         'skip$
1814         if$
1815         s nameptr format.name *
1816       }
1817       if$
1818       nameptr #1 + 'nameptr :=
1819       namesleft #1 - 'namesleft :=
1820     }
1821   while$
1822 }
1823
1824 FUNCTION {format.key}
1825 { empty$
1826   { key field.or.null }
1827   { "" }
1828   if$
1829 }
1830
1831 FUNCTION {format.authors}
1832 { author empty$ not
1833   { author format.names }
1834   { "empty author in " cite$ * warning$
1835   <*author-year>
1836     bbl.anonymous
1837   </author-year>
1838   <*numerical>
1839     ""
1840   </numerical>
1841   }
1842   if$
1843 }
1844
1845 FUNCTION {format.editors}
1846 { editor empty$
1847   { "" }
1848   { editor format.names }
1849   if$
1850 }
1851
1852 FUNCTION {format.translators}
1853 { translator empty$
1854   { "" }
1855   { translator format.names
1856     entry.lang lang.zh =
1857     { translator num.names$ #3 >

```

```

1858         { " 译" * }
1859         { ", 译" * }
1860         if$
1861     }
1862     'skip$
1863     if$
1864 }
1865 if$
1866 }
1867
1868 FUNCTION {format.full.names}
1869 { 's :=
1870   #1 'nameptr :=
1871   s num.names$ 'numnames :=
1872   numnames 'namesleft :=
1873   { namesleft #0 > }
1874   { s nameptr "{vv~}{ll}{, jj}{, ff}" format.name$ 't :=
1875     t get.str.lang 'name.lang :=
1876     name.lang lang.en =
1877       { t #1 "{vv~}{ll}" format.name$ 't := }
1878       { t #1 "{ll}{ff}" format.name$ 't := }
1879     if$
1880     nameptr #1 >
1881     {
1882       namesleft #1 >
1883       { ", " * t * }
1884       {
1885         numnames #2 >
1886         { ", " * }
1887         'skip$
1888         if$
1889         t "others" =
1890         { " et~al." * }
1891         { " and " * t * }
1892         if$
1893       }
1894       if$
1895     }
1896     't
1897     if$
1898     nameptr #1 + 'nameptr :=
1899     namesleft #1 - 'namesleft :=
1900   }
1901   while$
1902 }
1903
1904 FUNCTION {author.editor.full}
1905 { author empty$
1906   { editor empty$
1907     { "" }
1908     { editor format.full.names }
1909     if$
1910   }
1911   { author format.full.names }
1912   if$

```

```

1913 }
1914
1915 FUNCTION {author.full}
1916 { author empty$
1917   { "" }
1918   { author format.full.names }
1919   if$
1920 }
1921
1922 FUNCTION {editor.full}
1923 { editor empty$
1924   { "" }
1925   { editor format.full.names }
1926   if$
1927 }
1928
1929 FUNCTION {make.full.names}
1930 { type$ "book" =
1931   type$ "inbook" = booktitle empty$ not and
1932   or
1933   'author.editor.full
1934   { type$ "collection" =
1935     type$ "proceedings" =
1936     or
1937     'editor.full
1938     'author.full
1939     if$
1940   }
1941   if$
1942 }
1943
1944 FUNCTION {output.bibitem}
1945 { newline$
1946   "\bibitem[" write$
1947   label "]" *
1948   make.full.names duplicate$ short.list =
1949   { pop$ }
1950   { duplicate$ "]" contains
1951     { "{" swap$ * "]" * }
1952     'skip$
1953     if$
1954     *
1955   }
1956   if$
1957   "]" * write$
1958   cite$ write$
1959   "]" write$
1960   newline$
1961   ""
1962   before.all 'output.state :=
1963 }
1964

```

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.

```
1965 FUNCTION {change.sentence.case}
1966 { entry.lang lang.en =
1967   'smart.sentence.case
1968   'skip$
1969   if$
1970 }
1971
1972 FUNCTION {add.link}
1973 { url empty$ not
1974   { "\href{" url * "}" * swap$ * "}" * }
1975   { doi empty$ not
1976     { "\href{https://doi.org/" doi * "}" * swap$ * "}" * }
1977     'skip$
1978     if$
1979   }
1980   if$
1981 }
1982
1983 FUNCTION {format.title}
1984 { title empty$
1985   { "" }
1986   { title
1987     sentence.case.title
1988     'change.sentence.case
1989     'skip$
1990     if$
1991     entry.numbered number empty$ not and
1992     { bbl.colon *
1993       type$ "patent" = show.patent.country and
1994       { address empty$ not
1995         { address * ", " * }
1996         { location empty$ not
1997           { location * ", " * }
1998           { entry.lang lang.zh =
1999             { " 中国" * ", " * }
2000             'skip$
2001             if$
2002           }
2003           if$
2004         }
2005         if$
2006       }
2007       'skip$
2008       if$
2009       number *
2010     }
2011     'skip$
```

```

2012         if$
2013         link.title
2014         'add.link
2015         'skip$
2016     if$
2017 }
2018 if$
2019 }
2020

```

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

```

```

2021 FUNCTION {tie.or.space.connect}
2022 { duplicate$ text.length$ #3 <
2023   { "~" }
2024   { " " }
2025   if$
2026   swap$ * *
2027 }
2028

```

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*s" fields in "s" cite$)
    fi
END

```

```

2029 FUNCTION {either.or.check}
2030 { empty$
2031   'pop$
2032   { "can't use both " swap$ * " fields in " * cite$ * warning$ }
2033   if$
2034 }
2035

```

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.

```

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

```

```

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

```

```

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

```

```

2190 FUNCTION {remove.period}
2191 { 't :=
2192   "" 's :=
2193     { t empty$ not }
2194     { t #1 #1 substring$ 'tmp.str :=
2195       tmp.str "." = not
2196         { s tmp.str * 's := }
2197         'skip$
2198       if$
2199       t #2 global.max$ substring$ 't :=
2200     }
2201   while$
2202   s
2203 }
2204
2205 FUNCTION {abbreviate}
2206 { remove.period
2207   't :=
2208   t "l" change.case$ 's :=
2209   ""
2210   s "physical review letters" =
2211     { "Phys Rev Lett" }
2212     'skip$
2213   if$
2214   's :=
2215   s empty$
2216     { t }
2217     { pop$ s }
2218   if$
2219 }
2220
2221 FUNCTION {get.journal.title}
2222 { short.journal
2223   { shortjournal empty$ not
2224     { shortjournal }
2225     { journal empty$ not
2226       { journal abbreviate }
2227       { journaltitle empty$ not
2228         { journaltitle abbreviate }
2229         { "" }
2230       if$
2231     }
2232     if$
2233   }
2234   if$
2235 }
2236 { journal empty$ not
2237   { journal }
2238   { journaltitle empty$ not
2239     { journaltitle }
2240     { shortjournal empty$ not
2241       { shortjournal }
2242       { "" }
2243     if$
2244   }

```

```

2245         if$
2246     }
2247     if$
2248 }
2249 if$
2250 }
2251
2252 FUNCTION {check.arxiv.preprint}
2253 { #1 #5 substring$ purify$ "1" change.case$ "arxiv" =
2254   { #1 }
2255   { #0 }
2256   if$
2257 }
2258
2259 FUNCTION {format.journal}
2260 { get.journal.title
2261   duplicate$ empty$ not
2262     { italic.journal entry.lang lang.en = and
2263       'emphasize
2264       'skip$
2265       if$
2266       link.journal
2267       'add.link
2268       'skip$
2269       if$
2270     }
2271     'skip$
2272   if$
2273 }
2274

```

B.4.4 Format entry type mark

```

2275 FUNCTION {set.entry.mark}
2276 { entry.mark empty$ not
2277   'pop$
2278   { mark empty$ not
2279     { pop$ mark 'entry.mark := }
2280     { 'entry.mark := }
2281     if$
2282   }
2283   if$
2284 }
2285
2286 FUNCTION {format.mark}
2287 { show.mark
2288   { entry.mark
2289     show.medium.type
2290     { medium empty$ not
2291       { "/" * medium * }
2292       { entry.is.electronic
2293         { "/OL" * }
2294         'skip$
2295         if$
2296       }

```

```

2297         if$
2298     }
2299     'skip$
2300 if$
2301 'entry.mark :=
2302 space.before.mark
2303     { " " }
2304     { "\allowbreak" }
2305 if$
2306 "[" * entry.mark * "]" *
2307 }
2308 { "" }
2309 if$
2310 }
2311

```

B.4.5 Format edition

The `format.edition` function appends `edition` to the `edition`, if present. We lower-case the `edition` (it should be something like `Third`), because this doesn't start a sentence.

```

2312 FUNCTION {num.to.ordinal}
2313 { duplicate$ text.length$ 'charptr :=
2314   duplicate$ charptr #1 substring$ 's :=
2315   s "1" =
2316     { "st" * }
2317     { s "2" =
2318       { "nd" * }
2319       { s "3" =
2320         { "rd" * }
2321         { "th" * }
2322         if$
2323       }
2324     if$
2325   }
2326   if$
2327 }
2328
2329 FUNCTION {format.edition}
2330 { edition empty$
2331   { "" }
2332   { edition is.number
2333     { edition "1" = not
2334       { entry.lang lang.zh =
2335         { edition " 版" * }
2336         { edition num.to.ordinal " ed." * }
2337       if$
2338     }
2339     'skip$
2340   if$
2341   }
2342   { entry.lang lang.en =
2343     { edition change.sentence.case 's :=
2344       s "Revised" = s "Revised edition" = or
2345       { "Rev. ed." }

```

```

2346             { s " ed." * }
2347             if$
2348         }
2349         { edition }
2350     if$
2351 }
2352 if$
2353 }
2354 if$
2355 }
2356

```

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。

```

2357 FUNCTION {format.publisher}
2358 { publisher empty$ not
2359   { publisher }
2360   { school empty$ not
2361     { school }
2362     { organization empty$ not
2363       { organization }
2364       { institution empty$ not
2365         { institution }
2366         { "" }
2367       if$
2368     }
2369   if$
2370 }
2371 if$
2372 }
2373 if$
2374 }
2375
2376 FUNCTION {format.address.publisher}
2377 { address empty$ not
2378   { address }
2379   { location empty$ not
2380     { location }
2381     { "" }
2382   if$
2383 }
2384 if$
2385 duplicate$ empty$ not
2386   { format.publisher empty$ not
2387     { bbl.colon * format.publisher * }
2388     { entry.is.electronic not show.missing.address.publisher and
2389       { bbl.colon * bbl.sine.nomine * }
2390       'skip$
2391     if$
2392   }
2393   if$
2394 }
2395 { pop$

```

```

2396     entry.is.electronic not show.missing.address.publisher and
2397     { format.publisher empty$ not
2398       { bbl.sine.loco bbl.colon * format.publisher * }
2399       { bbl.sine.loco.sine.nomine }
2400     if$
2401   }
2402   { format.publisher empty$ not
2403     { format.publisher }
2404     { "" }
2405   if$
2406   }
2407   if$
2408 }
2409 if$
2410 }
2411

```

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。

```

2412 FUNCTION {extract.before.dash}
2413 { duplicate$ empty$
2414   { pop$ "" }
2415   { 's :=
2416     #1 'charptr :=
2417     s text.length$ #1 + 'len :=
2418     { charptr len <
2419       s charptr #1 substring$ "-" = not
2420       and
2421     }
2422     { charptr #1 + 'charptr := }
2423     while$
2424     s #1 charptr #1 - substring$
2425   }
2426   if$
2427 }
2428
2429 FUNCTION {extract.after.dash}
2430 { duplicate$ empty$
2431   { pop$ "" }
2432   { 's :=
2433     #1 'charptr :=
2434     s text.length$ #1 + 'len :=
2435     { charptr len <
2436       s charptr #1 substring$ "-" = not
2437       and
2438     }
2439     { charptr #1 + 'charptr := }
2440     while$

```



```

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

```

著者-出版年制必须提取出年份

```
2496 FUNCTION {format.year}
2497 { year empty$ not
2498   { year extra.label * }
2499   { date empty$ not
2500     { date extract.before.dash extra.label * }
2501     { entry.is.electronic not
2502       { "empty year in " cite$ * warning$ }
2503       'skip$
2504       if$
2505       urldate empty$ not
2506       { "[" urldate extract.before.dash * extra.label * "]" * }
2507       { "" }
2508       if$
2509     }
2510     if$
2511   }
2512   if$
2513 }
2514
2515 FUNCTION {format.periodical.year}
2516 { year empty$ not
2517   { year extract.before.slash
2518     "--" *
2519     year extract.after.slash
2520     duplicate$ empty$
2521     'pop$
2522     { * }
2523     if$
2524   }
2525   { date empty$ not
2526     { date extract.before.dash }
2527     { "empty year in " cite$ * warning$
2528       urldate empty$ not
2529       { "[" urldate extract.before.dash * "]" * }
2530       { "" }
2531       if$
2532     }
2533     if$
2534   }
2535   if$
2536 }
2537
```

专利和报纸都是使用日期而不是年

```
2538 FUNCTION {format.date}
2539 { date empty$ not
2540   { type$ "patent" = type$ "newspaper" = or
2541     { date }
2542     { entrysubtype empty$ not
2543       { type$ "article" = entrysubtype "newspaper" = and
2544         { date }
2545         { format.year }
2546       if$
2547     }
2548   }
2549 }
```

```

2548         { format.year }
2549     if$
2550 }
2551 if$
2552 }
2553 { year empty$ not
2554     { format.year }
2555     { "" }
2556 if$
2557 }
2558 if$
2559 }
2560

```

更新、修改日期只用于电子资源 **electronic**

```

2561 FUNCTION {format.editdate}
2562 { date empty$ not
2563     { "\allowbreak(" date * ")" * }
2564     { "" }
2565 if$
2566 }
2567

```

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。

```

2568 FUNCTION {format.urldate}
2569 { show.urldate show.url and entry.url empty$ not and
2570   is.pure.electronic or
2571   urldate empty$ not and
2572   { "\allowbreak[" urldate * "]" * }
2573   { "" }
2574 if$
2575 }
2576

```

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

<pre> pseudoVAR: pageresult: STRING n.dashify(s) == BEGIN t := s pageresult := "" </pre>	(it's what's accumulated on the stack)
--	--

```

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**。

```

2577 % str -> str
2578 % variable used: s, t, b
2579 FUNCTION {normalize.page.range}
2580 {
2581   " swap$
2582   { duplicate$ empty$ not }
2583   {
2584     #1 skip.inter.token.chars.by 't :=
2585     empty$
2586     { "" }
2587     'page.range.delimiter
2588     if$
2589     * t
2590     #1 tokenize.by 't :=
2591     * t
2592   }
2593   while$
2594   pop$
2595 }
2596

```

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

```

2597 FUNCTION {format.pages}
2598 {
2599   pages normalize.page.range
2600 }
2601
2602 FUNCTION {format.extracted.pages}
2603 { pages empty$

```

```

2604     { "" }
2605     { pages
2606       only.start.page
2607       { #1 tokenize.by pop$ }
2608       'normalize.page.range
2609     if$
2610   }
2611   if$
2612 }
2613

```

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。

```

2614 FUNCTION {format.journal.volume}
2615 { volume empty$ not
2616   { bold.journal.volume
2617     { "\textbf{" volume * "}" * }
2618     { volume }
2619   if$
2620   }
2621   { "" }
2622 if$
2623 }
2624
2625 FUNCTION {format.journal.number}
2626 { number empty$ not
2627   { "\allowbreak (" number * ")" * }
2628   { "" }
2629 if$
2630 }
2631
2632 FUNCTION {format.journal.pages}
2633 { pages empty$
2634   { "" }
2635   { format.extracted.pages }
2636 if$
2637 }
2638

```

连续出版物的年卷期有起止范围，需要特殊处理

```

2639 FUNCTION {format.periodical.year.volume.number}
2640 { year empty$ not
2641   { year extract.before.slash }
2642   { "empty year in periodical " cite$ * warning$ }
2643 if$
2644 volume empty$ not
2645   { ", " * volume extract.before.dash * }
2646   'skip$
2647 if$
2648 number empty$ not
2649   { "\allowbreak (" * number extract.before.dash * ")" * }

```

```

2650     'skip$
2651   if$
2652   "--" *
2653   year extract.after.slash empty$
2654   volume extract.after.dash empty$ and
2655   number extract.after.dash empty$ and not
2656   { year extract.after.slash empty$ not
2657     { year extract.after.slash * }
2658     { year extract.before.slash * }
2659   if$
2660   volume empty$ not
2661   { ", " * volume extract.after.dash * }
2662   'skip$
2663   if$
2664   number empty$ not
2665   { "\allowbreak (" * number extract.after.dash * ")" * }
2666   'skip$
2667   if$
2668   }
2669   'skip$
2670   if$
2671 }
2672

```

B.4.9 Format url and doi

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

```

2673 FUNCTION {check.url}
2674 { url empty$ not
2675   { url 'entry.url :=
2676     #1 'entry.is.electronic :=
2677   }
2678   { howpublished empty$ not
2679     { howpublished #1 #5 substring$ "\url{" =
2680       { howpublished 'entry.url :=
2681         #1 'entry.is.electronic :=
2682       }
2683       'skip$
2684     if$
2685   }
2686   { note empty$ not
2687     { note #1 #5 substring$ "\url{" =
2688       { note 'entry.url :=
2689         #1 'entry.is.electronic :=
2690       }
2691       'skip$
2692     if$
2693   }
2694   'skip$
2695   if$
2696 }
2697   if$
2698 }
2699   if$

```

```

2700 }
2701
2702 FUNCTION {output.url}
2703 { show.url is.pure.electronic or
2704   entry.url empty$ not and
2705   { new.block
2706     entry.url #1 #5 substring$ "\url{" =
2707     { entry.url }
2708     { "\url{" entry.url * "}" * }
2709     if$
2710     output
2711   }
2712   'skip$
2713   if$
2714 }
2715

```

需要检测 DOI 是否已经包含在 URL 中。

```

2716 FUNCTION {check.doi}
2717 { doi empty$ not
2718   { #1 'entry.is.electronic := }
2719   'skip$
2720   if$
2721 }
2722
2723 FUNCTION {is.in.url}
2724 { 's :=
2725   s empty$
2726   { #1 }
2727   { entry.url empty$
2728     { #0 }
2729     { s text.length$ 'len :=
2730       entry.url "1" change.case$ text.length$ 'charptr :=
2731       { entry.url "1" change.case$ charptr len substring$ s "1" chang
2732         charptr #0 >
2733         and
2734       }
2735       { charptr #1 - 'charptr := }
2736       while$
2737       charptr
2738     }
2739     if$
2740   }
2741   if$
2742 }
2743
2744 FUNCTION {format.doi}
2745 { ""
2746   doi empty$ not
2747   { "" 's :=
2748     doi 't :=
2749     #0 'numnames :=
2750     { t empty$ not}
2751     { t #1 #1 substring$ 'tmp.str :=
2752       tmp.str "," = tmp.str " " = or t #2 #1 substring$ empty$ or

```

```

2753         { t #2 #1 substring$ empty$
2754           { s tmp.str * 's := }
2755           'skip$
2756         if$
2757         s empty$ s is.in.url or
2758         'skip$
2759         { numnames #1 + 'numnames :=
2760           numnames #1 >
2761           { ", " * }
2762           { "DOI: " * }
2763           if$
2764           "\doi{" s * "}" * *
2765           }
2766         if$
2767         "" 's :=
2768         }
2769         { s tmp.str * 's := }
2770       if$
2771       t #2 global.max$ substring$ 't :=
2772     }
2773     while$
2774   }
2775   'skip$
2776 if$
2777 }
2778
2779 FUNCTION {output.doi}
2780 { doi empty$ not show.doi and
2781   show.english.translation entry.lang lang.zh = and not and
2782   { new.block
2783     format.doi output
2784   }
2785   'skip$
2786 if$
2787 }
2788
2789 FUNCTION {check.electronic}
2790 { "" 'entry.url :=
2791   #0 'entry.is.electronic :=
2792   'check.doi
2793   'skip$
2794 if$
2795   'check.url
2796   'skip$
2797 if$
2798 medium empty$ not
2799   { medium "MT" = medium "DK" = or medium "CD" = or medium "OL" = or
2800     { #1 'entry.is.electronic := }
2801     'skip$
2802   if$
2803   }
2804   'skip$
2805 if$
2806 }
2807

```



```

2808 FUNCTION {format.eprint}
2809 { archivePrefix empty$ not
2810   { archivePrefix }
2811   { eprinttype empty$ not
2812     { archivePrefix }
2813     { "" }
2814     if$
2815   }
2816   if$
2817   's :=
2818   s empty$ not
2819   { s ": \eprint{" *
2820     url empty$ not
2821     { url }
2822     { "https://" s "l" change.case$ * ".org/abs/" * eprint * }
2823     if$
2824     * "){ " *
2825     eprint * "}" *
2826   }
2827   { eprint }
2828   if$
2829 }
2830
2831 FUNCTION {output.eprint}
2832 { show.preprint eprint empty$ not and
2833   { new.block
2834     format.eprint output
2835   }
2836   'skip$
2837   if$
2838 }
2839
2840 FUNCTION {format.note}
2841 { note empty$ not show.note and
2842   { note }
2843   { "" }
2844   if$
2845 }
2846
2847 FUNCTION {output.translation}
2848 { show.english.translation entry.lang lang.zh = and
2849   { translation empty$ not
2850     { translation }
2851     { "[English translation missing!]" }
2852     if$
2853     " (in Chinese)" * output
2854     write$
2855     format.doi duplicate$ empty$ not
2856     { newline$
2857       write$
2858     }
2859     'pop$
2860     if$
2861     " \\" write$
2862     newline$

```

```

2863         "(" write$
2864         ""
2865         before.all 'output.state :=
2866     }
2867     'skip$
2868     if$
2869 }
2870

```

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

```

2871 FUNCTION {empty.misc.check}
2872 { author empty$ title empty$
2873   year empty$
2874   and and
2875   key empty$ not and
2876   { "all relevant fields are empty in " cite$ * warning$ }
2877   'skip$
2878   if$
2879 }
2880

```

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

```

2881 FUNCTION {monograph}
2882 { output.bibitem
2883   output.translation
2884   author empty$ not
2885     { format.authors }
2886     { editor empty$ not
2887       { format.editors }
2888       { "empty author and editor in " cite$ * warning$
2889 <*author-year>
2890       bbl.anonymous
2891 </author-year>
2892 <*numerical>
2893       ""
2894 </numerical>
2895     }
2896     if$
2897   }
2898   if$

```

```

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

```

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

```

2932 FUNCTION {incollection}
2933 { output.bibitem
2934   output.translation
2935   format.authors output
2936   author format.key output
2937   year.after.author
2938       { period.after.author
2939           'new.sentence
2940           'skip$
2941           if$
2942           format.year "year" output.check
2943       }

```

```

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

```

B.5.3 连续出版物

```

2973 FUNCTION {periodical}
2974 { output.bibitem
2975   output.translation
2976   format.authors output
2977   author format.key output
2978   year.after.author
2979     { period.after.author
2980       'new.sentence
2981       'skip$
2982       if$
2983       format.year "year" output.check
2984     }
2985   'skip$
2986   if$
2987   new.block
2988   format.title "title" output.check
2989   "J" set.entry.mark
2990   format.mark "" output.after
2991   new.block
2992   format.periodical.year.volume.number output
2993   new.block
2994   format.address.publisher output
2995   year.after.author not

```

```

2996     { format.periodical.year "year" output.check }
2997     'skip$
2998   if$
2999   format.urldate "" output.after
3000   output.url
3001   output.doi
3002   new.block
3003   format.note output
3004   fin.entry
3005 }
3006

```

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.

```

3007 FUNCTION {journal.article}
3008 { output.bibitem
3009   output.translation
3010   format.authors output
3011   author format.key output
3012   year.after.author
3013     { period.after.author
3014       'new.sentence
3015       'skip$
3016       if$
3017       format.year "year" output.check
3018     }
3019   'skip$
3020   if$
3021   new.block
3022   title.in.journal
3023     { format.title "title" output.check
3024       entrysubtype empty$ not
3025       {
3026         entrysubtype "newspaper" =
3027         { "N" set.entry.mark }
3028         { "J" set.entry.mark }
3029         if$
3030       }
3031       { "J" set.entry.mark }
3032     if$
3033     format.mark "" output.after
3034     new.block
3035   }
3036   'skip$
3037   if$
3038   format.journal "journal" output.check

```

```

3039 year.after.author not
3040     { format.date "year" output.check }
3041     'skip$
3042 if$
3043 format.journal.volume output
3044 format.journal.number "" output.after
3045 format.journal.pages bbl.pages.colon output.after
3046 format.urldate "" output.after
3047 output.url
3048 output.doi
3049 new.block
3050 format.note output
3051 fin.entry
3052 }
3053

```

B.5.5 专利文献

number 域也可以用来表示专利号。

```

3054 FUNCTION {patent}
3055 { output.bibitem
3056   output.translation
3057   format.authors output
3058   author format.key output
3059   year.after.author
3060     { period.after.author
3061       'new.sentence
3062       'skip$
3063       if$
3064       format.year "year" output.check
3065     }
3066   'skip$
3067   if$
3068   new.block
3069   format.title "title" output.check
3070   "P" set.entry.mark
3071   format.mark "" output.after
3072   new.block
3073   format.date "year" output.check
3074   format.urldate "" output.after
3075   output.url
3076   output.doi
3077   new.block
3078   format.note output
3079   fin.entry
3080 }
3081

```

B.5.6 电子资源

```

3082 FUNCTION {electronic}
3083 { #1 #1 check.electronic
3084   #1 'entry.is.electronic :=
3085   #1 'is.pure.electronic :=

```

```

3086 output.bibitem
3087 output.translation
3088 format.authors output
3089 author format.key output
3090 year.after.author
3091 { period.after.author
3092     'new.sentence
3093     'skip$
3094     if$
3095         format.year "year" output.check
3096     }
3097     'skip$
3098 if$
3099 new.block
3100 format.series.vol.num.title "title" output.check
3101 "EB" set.entry.mark
3102 format.mark "" output.after
3103 new.block
3104 format.address.publisher output
3105 year.after.author not
3106 { date empty$
3107     { format.date output }
3108     'skip$
3109     if$
3110     }
3111     'skip$
3112 if$
3113 format.pages bbl.pages.colon output.after
3114 format.editdate "" output.after
3115 format.urldate "" output.after
3116 output.url
3117 output.doi
3118 new.block
3119 format.note output
3120 fin.entry
3121 }
3122

```

B.5.7 预印本

```

3123 FUNCTION {preprint}
3124 { output.bibitem
3125   output.translation
3126   author empty$ not
3127     { format.authors }
3128     { editor empty$ not
3129       { format.editors }
3130       { "empty author and editor in " cite$ * warning$
3131 <*author-year>
3132         bbl.anonymous
3133 </author-year>
3134 <*numerical>
3135         ""
3136 </numerical>
3137       }
3138     if$

```

```

3139     }
3140     if$
3141     output
3142     year.after.author
3143     { period.after.author
3144       'new.sentence
3145       'skip$
3146       if$
3147       format.year "year" output.check
3148     }
3149     'skip$
3150     if$
3151     new.block
3152     title.in.journal
3153     { format.series.vol.num.title "title" output.check
3154     (*2015)
3155       "A" set.entry.mark
3156     (/2015)
3157     (*!2015)
3158       "Z" set.entry.mark
3159     (/!2015)
3160       format.mark "" output.after
3161       new.block
3162     }
3163     'skip$
3164     if$
3165     format.translators output
3166     new.sentence
3167     format.edition output
3168     new.block
3169     year.after.author not
3170     { date empty$
3171       { format.date output }
3172       'skip$
3173       if$
3174     }
3175     'skip$
3176     if$
3177     format.pages bbl.pages.colon output.after
3178     format.editdate "" output.after
3179     format.urldate "" output.after
3180     output.eprint
3181     output.url
3182     show.preprint not eprint empty$ or
3183       'output.doi
3184       'skip$
3185     if$
3186     new.block
3187     format.note output
3188     fin.entry
3189   }
3190

```


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。

```
3191 FUNCTION {misc}
3192 { get.journal.title
3193   duplicate$ empty$ not
3194     { check.arxiv.preprint
3195       'preprint
3196       'journal.article
3197     } if$
3198   }
3199   { pop$
3200     booktitle empty$ not
3201     'incollection
3202     { eprint empty$ not archivePrefix empty$ not or
3203       'preprint
3204       { publisher empty$ not
3205         'monograph
3206         { entry.is.electronic
3207           'electronic
3208           {
3209             <*/2005>
3210               "Z" set.entry.mark
3211             </!2005>
3212             <*2005>
3213               "M" set.entry.mark
3214             </2005>
3215             monograph
3216           }
3217         } if$
3218       }
3219     } if$
3220   }
3221   if$
3222 }
3223   if$
3224 }
3225   if$
3226   empty.misc.check
3227 }
3228
3229 FUNCTION {archive}
3230 { "A" set.entry.mark
3231   misc
3232 }
3233
3234 FUNCTION {article} { misc }
3235
```

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

```
3236 FUNCTION {book} { monograph }
3237
```

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

Required: title

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

```
3238 FUNCTION {booklet} { book }
3239
3240 FUNCTION {collection}
3241 { "G" set.entry.mark
3242   monograph
3243 }
3244
3245 FUNCTION {database}
3246 { "DB" set.entry.mark
3247   electronic
3248 }
3249
3250 FUNCTION {dataset}
3251 { "DS" set.entry.mark
3252   electronic
3253 }
3254
```

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。

```
3255 FUNCTION {inbook} {
3256   booktitle empty$
3257   'book
3258   'incollection
3259   if$
3260 }
3261
```

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

```
3262 FUNCTION {inproceedings}
3263 { "C" set.entry.mark
3264   incollection
```

3265 }
3266

The conference function is included for Scribe compatibility.

3267 FUNCTION {conference} { inproceedings }
3268
3269 FUNCTION {legislation} { archive }
3270
3271
3272 FUNCTION {map}
3273 { "CM" set.entry.mark
3274 misc
3275 }
3276

A manual is technical documentation.

Required: title

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

3277 FUNCTION {manual} { monograph }
3278

A mastersthesis is a Master's thesis.

Required: author, title, school, year

Optional: type, address, month, note

3279 FUNCTION {mastersthesis}
3280 { "D" set.entry.mark
3281 monograph
3282 }
3283
3284 FUNCTION {newspaper}
3285 { "N" set.entry.mark
3286 article
3287 }
3288
3289 FUNCTION {online}
3290 { "EB" set.entry.mark
3291 electronic
3292 }
3293

A phdthesis is like a mastersthesis.

Required: author, title, school, year

Optional: type, address, month, note

3294 FUNCTION {phdthesis} { mastersthesis }
3295

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

```
3296 FUNCTION {proceedings}
3297 { "C" set.entry.mark
3298   monograph
3299 }
3300
3301 FUNCTION {software}
3302 { "CP" set.entry.mark
3303   electronic
3304 }
3305
3306 FUNCTION {standard}
3307 { "S" set.entry.mark
3308   misc
3309 }
3310
```

A techreport is a technical report.

Required: author, title, institution, year

Optional: type, number, address, month, note

```
3311 FUNCTION {techreport}
3312 { "R" set.entry.mark
3313   misc
3314 }
3315
```

An unpublished is something that hasn't been published.

Required: author, title, note

Optional: month, year

```
3316 FUNCTION {unpublished} { misc }
3317
```

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

```
3318 FUNCTION {default.type} { misc }
3319
```

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

```
3320 MACRO {jan} {"January"}
3321
3322 MACRO {feb} {"February"}
3323
3324 MACRO {mar} {"March"}
3325
3326 MACRO {apr} {"April"}
3327
```

```

3328 MACRO {may} {"May"}
3329
3330 MACRO {jun} {"June"}
3331
3332 MACRO {jul} {"July"}
3333
3334 MACRO {aug} {"August"}
3335
3336 MACRO {sep} {"September"}
3337
3338 MACRO {oct} {"October"}
3339
3340 MACRO {nov} {"November"}
3341
3342 MACRO {dec} {"December"}
3343

```

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

```

3344 MACRO {acmcs} {"ACM Computing Surveys"}
3345
3346 MACRO {acta} {"Acta Informatica"}
3347
3348 MACRO {cacm} {"Communications of the ACM"}
3349
3350 MACRO {ibmjrd} {"IBM Journal of Research and Development"}
3351
3352 MACRO {ibmsj} {"IBM Systems Journal"}
3353
3354 MACRO {ieeese} {"IEEE Transactions on Software Engineering"}
3355
3356 MACRO {ieeetc} {"IEEE Transactions on Computers"}
3357
3358 MACRO {ieeetcad}
3359 {"IEEE Transactions on Computer-Aided Design of Integrated Circuits"}
3360
3361 MACRO {ipl} {"Information Processing Letters"}
3362
3363 MACRO {jacm} {"Journal of the ACM"}
3364
3365 MACRO {jcss} {"Journal of Computer and System Sciences"}
3366
3367 MACRO {scp} {"Science of Computer Programming"}
3368
3369 MACRO {sicomp} {"SIAM Journal on Computing"}
3370
3371 MACRO {tocs} {"ACM Transactions on Computer Systems"}
3372
3373 MACRO {tods} {"ACM Transactions on Database Systems"}
3374

```

```

3375 MACRO {tog} {"ACM Transactions on Graphics"}
3376
3377 MACRO {toms} {"ACM Transactions on Mathematical Software"}
3378
3379 MACRO {toois} {"ACM Transactions on Office Information Systems"}
3380
3381 MACRO {toplas} {"ACM Transactions on Programming Languages and Systems"}
3382
3383 MACRO {tcs} {"Theoretical Computer Science"}
3384

```

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`.

```

3385 FUNCTION {sortify}
3386 { purify$
3387   "l" change.case$
3388 }
3389

```

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

```

3390 FUNCTION {chop.word}
3391 { 's :=
3392   'len :=
3393   s #1 len substring$ =
3394     { s len #1 + global.max$ substring$ }
3395   's
3396   if$
3397 }
3398

```

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}}{l_{j}}")
            nameptr := nameptr + 1
            namesleft := namesleft - 1
        od
    if numnames > 4 then
        nameresult := nameresult * "{\etalchar{+}}"
        et.al.char.used := true
    else
        t := format.name$(s, 1, "{v_{l}}{l_{j}}")
        if text.length$(t) < 2 then % there's just one name-token
            nameresult := text.prefix$(format.name$(s, 1, "{ll}"), 3)
        else
            nameresult := t
        fi
    fi
    return nameresult
END

```

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

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

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

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

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

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

```
3399 FUNCTION {format.lab.name}
3400 { "{vv~}{ll}{, jj}{, ff}" format.name$ 't :=
3401   t "others" =
3402     { citation.et.al }
3403     { t get.str.lang 'name.lang :=
3404       name.lang lang.zh = name.lang lang.ja = or
3405         { t #1 "{ll}{ff}" format.name$ }
3406         { t #1 "{vv~}{ll}" format.name$ }
3407       if$
3408     }
3409   if$
3410 }
3411
3412 FUNCTION {format.lab.names}
3413 { 's :=
3414   #1 'nameptr :=
3415   s num.names$ 'numnames :=
3416   ""
3417   numnames 'namesleft :=
3418   { namesleft #0 > }
3419   { s nameptr format.lab.name citation.et.al =
3420     numnames citation.et.al.min #1 - > nameptr citation.et.al.use.first >
3421     { bbl.space *
3422       citation.et.al *
3423       #1 'namesleft :=
3424     }
3425     { nameptr #1 >
3426       { namesleft #1 = citation.and "" = not and
3427         { citation.and * }
3428         { ", " * }
3429       if$
3430     }
3431     'skip$
3432   if$
```



```

3433         s nameptr format.lab.name *
3434     }
3435     if$
3436     nameptr #1 + 'nameptr :=
3437     namesleft #1 - 'namesleft :=
3438 }
3439 while$
3440 }
3441
3442 FUNCTION {author.key.label}
3443 { author empty$
3444   { key empty$
3445     { cite$ #1 #3 substring$ }
3446     'key
3447     if$
3448   }
3449   { author format.lab.names }
3450   if$
3451 }
3452
3453 FUNCTION {author.editor.key.label}
3454 { author empty$
3455   { editor empty$
3456     { key empty$
3457       { cite$ #1 #3 substring$ }
3458       'key
3459       if$
3460     }
3461     { editor format.lab.names }
3462     if$
3463   }
3464   { author format.lab.names }
3465   if$
3466 }
3467
3468 FUNCTION {author.key.organization.label}
3469 { author empty$
3470   { key empty$
3471     { organization empty$
3472       { cite$ #1 #3 substring$ }
3473       { "The " #4 organization chop.word #3 text.prefix$ }
3474       if$
3475     }
3476     'key
3477     if$
3478   }
3479   { author format.lab.names }
3480   if$
3481 }
3482
3483 FUNCTION {editor.key.organization.label}
3484 { editor empty$
3485   { key empty$
3486     { organization empty$
3487       { cite$ #1 #3 substring$ }

```

```

3488         { "The " #4 organization chop.word #3 text.prefix$ }
3489         if$
3490     }
3491     'key
3492     if$
3493 }
3494 { editor format.lab.names }
3495 if$
3496 }
3497
3498 FUNCTION {calc.short.authors}
3499 { type$ "book" =
3500   type$ "inbook" = booktitle empty$ not and
3501   or
3502   'author.editor.key.label
3503   { type$ "collection" =
3504     type$ "proceedings" =
3505     or
3506     { editor empty$ not
3507       'editor.key.organization.label
3508       'author.key.organization.label
3509       if$
3510     }
3511     'author.key.label
3512     if$
3513   }
3514   if$
3515   'short.list :=
3516 }
3517

```

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)。

```

3518 FUNCTION {calc.label}
3519 { calc.short.authors
3520   short.list "]" contains
3521   { "{" short.list * "}" * }
3522   { short.list }
3523   if$
3524   "("
3525   *
3526   format.year duplicate$ empty$
3527   short.list key field.or.null = or
3528   { pop$ "" }
3529   'skip$
3530   if$
3531   duplicate$ "]" contains
3532   { "{" swap$ * "}" * }
3533   'skip$
3534   if$
3535   *
3536   'label :=
3537 }

```

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 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`.

```

3539 <*author-year>
3540 FUNCTION {sort.language.label}
3541 { entry.lang lang.zh =
3542   { lang.zh.order }
3543   { entry.lang lang.ja =
3544     { lang.ja.order }
3545     { entry.lang lang.en =
3546       { lang.en.order }
3547       { entry.lang lang.ru =
3548         { lang.ru.order }
3549         { lang.other.order }
3550       if$
3551     }
3552   if$
3553 }
3554 if$
3555 }
3556 if$
3557 #64 +
3558 int.to.chr$
3559 }
3560
3561 FUNCTION {sort.format.names}
3562 { 's :=
3563   #1 'nameptr :=
3564   ""
3565   s num.names$ 'numnames :=
3566   numnames 'namesleft :=
3567   { namesleft #0 > }

```

```

3568     {
3569         s nameptr "{vv{ } }{ll{ }}{ ff{ }}{ jj{ }}" format.name$ 't :=
3570         nameptr #1 >
3571         {
3572             " " *
3573             namesleft #1 = t "others" = and
3574             { "zzzzz" * }
3575             { numnames #2 > nameptr #2 = and
3576                 { "zz" * year field.or.null * " " * }
3577                 'skip$
3578                 if$
3579                 t sortify *
3580             }
3581             if$
3582             {
3583                 { t sortify * }
3584                 if$
3585                 nameptr #1 + 'nameptr :=
3586                 namesleft #1 - 'namesleft :=
3587             }
3588         while$
3589     }
3590

```

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`

```

3591 FUNCTION {sort.format.title}
3592 { 't :=
3593     "A " #2
3594     "An " #3
3595     "The " #4 t chop.word
3596     chop.word
3597     chop.word
3598     sortify
3599     #1 global.max$ substring$
3600 }
3601

```

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.

```

3602 FUNCTION {anonymous.sort}
3603 { entry.lang lang.zh =
3604     { "yi4 ming2" }
3605     { "anon" }
3606     if$
3607 }
3608
3609 FUNCTION {warn.empty.key}
3610 { entry.lang lang.zh =

```

```

3611     { "empty key in " cite$ * warning$ }
3612     'skip$
3613     if$
3614 }
3615
3616 FUNCTION {author.sort}
3617 { key empty$
3618   { warn.empty.key
3619     author empty$
3620     { anonymous.sort }
3621     { author sort.format.names }
3622     if$
3623   }
3624   { key }
3625   if$
3626 }
3627
3628 FUNCTION {author.editor.sort}
3629 { key empty$
3630   { warn.empty.key
3631     author empty$
3632     { editor empty$
3633       { anonymous.sort }
3634       { editor sort.format.names }
3635       if$
3636     }
3637     { author sort.format.names }
3638     if$
3639   }
3640   { key }
3641   if$
3642 }
3643
3644 FUNCTION {author.organization.sort}
3645 { key empty$
3646   { warn.empty.key
3647     author empty$
3648     { organization empty$
3649       { anonymous.sort }
3650       { "The " #4 organization chop.word sortify }
3651       if$
3652     }
3653     { author sort.format.names }
3654     if$
3655   }
3656   { key }
3657   if$
3658 }
3659
3660 FUNCTION {editor.organization.sort}
3661 { key empty$
3662   { warn.empty.key
3663     editor empty$
3664     { organization empty$
3665       { anonymous.sort }

```

```

3666         { "The " #4 organization chop.word sortify }
3667     if$
3668 }
3669 { editor sort.format.names }
3670 if$
3671 }
3672 { key }
3673 if$
3674 }
3675
3676 </author-year>

```

顺序编码制的排序要简单得多

```

3677 <*numerical>
3678 INTEGERS { seq.num }
3679
3680 FUNCTION {init.seq}
3681 { #0 'seq.num :=}
3682
3683 FUNCTION {int.to.fix}
3684 { "000000000" swap$ int.to.str$ *
3685   #-1 #10 substring$
3686 }
3687
3688 </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!

```

3689 FUNCTION {presort}
3690 { set.entry.lang
3691   set.entry.numbered
3692   show.url show.doi check.electronic
3693   #0 'is.pure.electronic :=
3694   calc.label
3695   label sortify
3696   " "
3697   *
3698 <*author-year>
3699   sort.language.label
3700   " "
3701   *
3702   type$ "book" =
3703   type$ "inbook" = booktitle empty$ not and
3704   or
3705   'author.editor.sort
3706   { type$ "collection" =
3707     type$ "proceedings" =
3708     or
3709     'editor.organization.sort
3710     'author.sort
3711   }
3712   if$
3713 }

```

```

3714      *
3715      "      "
3716      *
3717      year field.or.null sortify
3718      *
3719      "      "
3720      *
3721      cite$
3722      *
3723      #1 entry.max$ substring$
3724      </author-year>
3725      <*numerical>
3726          seq.num #1 + 'seq.num :=
3727          seq.num int.to.fix
3728      </numerical>
3729      'sort.label :=
3730      sort.label *
3731      #1 entry.max$ substring$
3732      'sort.key$ :=
3733      }
3734

```

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

3735 STRINGS { longest.label last.label next.extra last.extra.label }
3736
3737 INTEGERS { longest.label.width number.label }
3738
3739 FUNCTION {initialize.longest.label}
3740 { "" 'longest.label :=
3741   #0 int.to.chr$ 'last.label :=
3742   "" 'next.extra :=
3743   #0 'longest.label.width :=
3744   #0 'number.label :=
3745   "" 'last.extra.label :=
3746 }
3747
3748 FUNCTION {forward.pass}
3749 {
3750   (*author-year)
3751   last.label label =
3752   { "" 'extra.label :=
3753     last.extra.label text.length$ 'charptr :=
3754     { last.extra.label charptr #1 substring$ "z" =
3755       charptr #0 > and
3756     }
3757     { "a" extra.label * 'extra.label :=
3758       charptr #1 - 'charptr :=
3759     }
3760     while$
3761     charptr #0 >
3762     { last.extra.label charptr #1 substring$ chr.to.int$ #1 + int.to.ch
3763       extra.label * 'extra.label :=
3764       last.extra.label #1 charptr #1 - substring$
3765       extra.label * 'extra.label :=
3766     }
3767     { "a" extra.label * 'extra.label := }
3768     if$
3769     extra.label 'last.extra.label :=
3770   }
3771   { "a" 'last.extra.label :=
3772     "" 'extra.label :=
3773     label 'last.label :=
3774   }
3775   if$
3776   (/author-year)
3777   number.label #1 + 'number.label :=

```



```

3778 }
3779
3780 FUNCTION {reverse.pass}
3781 {
3782   <*author-year>
3783   next.extra "b" =
3784     { "a" 'extra.label := }
3785   'skip$
3786   if$
3787   extra.label 'next.extra :=
3788   extra.label
3789   duplicate$ empty$
3790     'skip$
3791     { "{\natexlab{" swap$ * "}}" * }
3792   if$
3793   'extra.label :=
3794   </author-year>
3795   label extra.label * 'label :=
3796 }
3797
3798 FUNCTION {bib.sort.order}
3799 { sort.label 'sort.key$ :=
3800 }
3801

```

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 $\backslash\text{begin}\{\text{thebibliography}\}\{\dots\}$ where the '...' is the longest label.

We also call `init.state.consts`, for use by the output routines.

```

3802 FUNCTION {begin.bib}
3803 {   preamble$ empty$
3804     'skip$
3805     { preamble$ write$ newline$ }
3806   if$
3807   "\begin{thebibliography}\{" number.label int.to.str$ * "\}" *
3808   write$ newline$
3809   terms.in.macro
3810     { "\providecommand{\biband}\{和\}"
3811       write$ newline$
3812       "\providecommand{\bibetal}\{等\}"
3813       write$ newline$
3814     }
3815   'skip$
3816   if$
3817   "\providecommand{\natexlab}[1]\{#1\}"
3818   write$ newline$
3819   "\providecommand{\url}[1]\{#1\}"
3820   write$ newline$
3821   "\expandafter\ifx\csname urlstyle\endcsname\relax\else"

```

```

3822 write$ newline$
3823 " \urlstyle{same}\fi"
3824 write$ newline$
3825 "\expandafter\ifx\csname href\endcsname\relax"
3826 write$ newline$
3827 " \DeclareUrlCommand\doi{\urlstyle{rm}} "
3828 write$ newline$
3829 " \def\eprint#1#2{#2}"
3830 write$ newline$
3831 "\else"
3832 write$ newline$
3833 " \def\doi#1{\href{https://doi.org/#1}{\nolinkurl{#1}}}"
3834 write$ newline$
3835 " \let\eprint\href"
3836 write$ newline$
3837 "\fi"
3838 write$ newline$
3839 }
3840

```

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

```

3841 FUNCTION {end.bib}
3842 { newline$
3843   "\end{thebibliography}" write$ newline$
3844 }
3845

```

B.10 Main execution

Now we read in the .BIB entries.

```

3846 READ
3847
3848 EXECUTE {init.state.consts}
3849
3850 EXECUTE {load.config}
3851
3852 ⟨*numerical⟩
3853 EXECUTE {init.seq}
3854
3855 ⟨/numerical⟩
3856 ITERATE {presort}
3857

```

And now we can sort

```

3858 SORT
3859
3860 EXECUTE {initialize.longest.label}
3861
3862 ITERATE {forward.pass}
3863
3864 REVERSE {reverse.pass}
3865
3866 ITERATE {bib.sort.order}
3867

```

```
3868 SORT
3869
3870 EXECUTE {begin.bib}
3871
```

Now we produce the output for all the entries

```
3872 ITERATE {call.type$}
3873
3874 EXECUTE {end.bib}
3875 </author-year | numerical>
```