GB/T 7714 BibTeX style

Zeping Lee*

2025/03/11 v2.1.7

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 biblilography entry.

1 简介

GB/T 7714—2015《信息与文献　参考文献著录规则》[1]（以下简称"国标"）是中国的参考文献格式推荐标准。国内的绝大部分学术期刊、学位论文都使用了基于该标准的格式。本宏包是国标的 BibTeX[2] 实现，具有以下特性：

- 兼容 natbib 宏包[3]。
- 支持"顺序编码制"和"著者-出版年制"两种风格。
- 自动识别语言并进行相应处理。
- 提供了简单的接口供用户修改样式。
- 同时提供了 2005 版的 `.bst` 文件。

本宏包的主页：`https://github.com/zepinglee/gbt7714-bibtex-style`。

2 版本 v2.0 的重要修改

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

*zepinglee AT gmail.com

3 使用方法

以下是 gbt7714 宏包的一个简单示例。

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

按照国标的规定，参考文献的标注体系分为“顺序编码制”和“著者-出版年制”。用户应在导言区调用宏包 gbt7714，并且使用 `\bibliographystyle` 命令选择参考文献表的样式，比如：

```
\bibliographystyle{gbt7714-numerical}  % 顺序编码制
```

或者

```
\bibliographystyle{gbt7714-author-year}  % 著者-出版年制
```

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

注意，版本 v2.0 更改了设置参考文献表样式的方法，要求直接使用 `\bibliographystyle`，不再使用宏包的参数，而且更改了 `bst` 的文件名。

`\citestyle` `\citestyle{⟨citation style⟩}`

可选：`super`, `numbers`, `author-year`。使用 `\bibliography` 选择参考文献表的样式时会自动设置对应的引用样式。顺序编码制的引用标注默认使用角标式（`super`），如“张三[2] 提出”。如果要使用正文模式，如“文献 [3] 中说明”，可以使用 `\citestyle` 命令切换为数字式（`numbers`）。

```
\citestyle{numbers}
```

著者-出版年制通常不需要修改引用样式。

`sort&compress` 同一处引用多篇文献时，应当将各篇文献的 key 一同写在 `\cite` 命令中。如遇连续编号，默认会自动转为起讫序号并用短横线连接（见 natbib 的 `compress` 选项）。如果要对引用的编号进行自动排序，需要在调用 gbt7714 时加 `sort&compress` 参数，这些参数会传给 natbib 处理。

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

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

若需要标出引文的页码，可以标在`\cite`的可选参数中，如`\cite[42]{knuth84}`。更多的引用标注方法可以参考 natbib 宏包的使用说明[3]。

使用时需要注意以下几点：

- `.bib` 数据库应使用 UTF-8 编码。
- 使用著者-出版年制参考文献表时，中文的文献*必须* 在 key 域填写作者姓名的拼音，才能按照拼音排序，详见第 6 节。

4 文献类型

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

表 1: 全部文献类型

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

5 著录项目

由于国标中规定的著录项目多于 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
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 annote, chapter, crossref, month, type。

本宏包默认情况下可以自动识别文献语言，并自动处理文献类型和载体类型标识，但是在少数情况下需要用户手动指定，如：

```
@misc{citekey,
  langid = {japanese},
  mark   = {Z},
  medium = {DK},
  ...
}
```

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

6 文献列表的排序

国标规定参考文献表采用著者-出版年制组织时，各篇文献首先按文种集中，然后按著者字顺和出版年排列；中文文献可以按著者汉语拼音字顺排列，也可以按著者的笔画笔顺排列。然而由于 BibTEX 功能的局限性，无法自动获取著者姓名的拼音或笔画笔顺，所以必须在 bib 数据库中的 key 域手动录入著者姓名的拼音用于排序，如：

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

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

7 自定义样式

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

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

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

8 相关工作

TeX 社区也有其他关于 GB/T 7714 系列参考文献标准的工作。2005 年吴凯[7]发布了基于 GB/T 7714—2005 的 BibTEX 样式，支持顺序编码制和著者出版年制两种风格。李志奇[8]发布了严格遵循 GB/T 7714—2005 的 BibLaTeX 的样式。胡海星[9]提供

表 2: 参考文献表样式的配置参数

参数值	默认值	功能
uppercase.name	`#1`	将著者姓名转为大写
max.num.authors	`#3`	输出著者的最多数量
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
wave.dash.in.pages	`#0`	起止页码使用波浪号
show.urldate	`#1`	显示引用日期 urldate
show.url	`#1`	显示 url
show.doi	`#1`	显示 DOI
show.preprint	`#1`	显示预印本信息
show.note	`#0`	显示 note 域的信息
end.with.period	`#1`	结尾加句点

了另一个 BibTEX 实现，还给每行 bst 代码写了 java 语言注释。沈周[10] 基于 biblatex-caspervector[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] 中国国家标准化委员会. 信息与文献　参考文献著录规则: GB/T 7714—2015[S]. 北京: 中国标准出版社, 2015.

[2] PATASHNIK O. BibTEXing[M/OL]. 1988. http://mirrors.ctan.org/biblio/bibtex/base/btxdoc.pdf.

[3] DALY P W. Natural sciences citations and references[M/OL]. 1999. http://mirrors.ctan.org/macros/latex/contrib/natbib/natbib.pdf.

[4] PATASHNIK O. Designing BibTEX styles[M/OL]. 1988. http://mirrors.ctan.org/biblio/bibtex/base/btxhak.pdf.

[5] MARKEY N. Tame the beast[M/OL]. 2003. http://mirrors.ctan.org/info/bibtex/tamethebeast/ttb_en.pdf.

[6] MITTELBACH F, GOOSSENS M, BRAAMS J, et al. The LATEX companion[M]. 2nd ed. Reading, MA, USA: Addison-Wesley, 2004.

[7] 吴凯. 发布 GBT7714-2005.bst version1 Beta 版 [EB/OL]. 2006. CTeX 论坛（已关闭）.

[8] 李志奇. 基于 biblatex 的符合 GBT7714—2005 的中文文献生成工具 [EB/OL]. 2013. CTeX 论坛（已关闭）.

[9] 胡海星. A GB/T 7714—2005 national standard compliant BibTeX style[EB/OL]. 2013. https://github.com/Haixing-Hu/GBT7714-2005-BibTeX-Style.

[10] 沈周. 基于 caspervector 改写的符合 GB/T 7714—2005 标准的参考文献格式 [EB/OL]. 2016. https://github.com/szsdk/biblatex-gbt77142005.

[11] VECTOR C T. biblatex 参考文献和引用样式: caspervector[M/OL]. 2012. http://mirrors.ctan.org/macros/latex/contrib/biblatex-contrib/biblatex-caspervector/doc/caspervector.pdf.

[12] 胡振震. 符合 GB/T 7714—2015 标准的 biblatex 参考文献样式 [M/OL]. 2016. http://mirrors.ctan.org/macros/latex/contrib/biblatex-contrib/biblatex-gb7714-2015/biblatex-gb7714-2015.pdf.

A 宏包的代码实现

兼容过时的接口

```
⟨*package⟩
\newif\ifgbt@legacy@interface
\newif\ifgbt@mmxv
\newif\ifgbt@numerical
\newif\ifgbt@super
\newcommand\gbt@obsolete@option[1]{%
  \PackageWarning{gbt7714}{The option "#1" is obsolete}%
}
\DeclareOption{2015}{%
  \gbt@obsolete@option{2015}%
  \gbt@legacy@interfacetrue
  \gbt@mmxvtrue
}
\DeclareOption{2005}{%
  \gbt@obsolete@option{2005}%
  \gbt@legacy@interfacetrue
  \gbt@mmxvfalse
}
\DeclareOption{super}{%
  \gbt@obsolete@option{super}%
  \gbt@legacy@interfacetrue
  \gbt@numericaltrue
  \gbt@supertrue
}
\DeclareOption{numbers}{%
  \gbt@obsolete@option{numbers}%
  \gbt@legacy@interfacetrue
  \gbt@numericaltrue
  \gbt@superfalse
}
\DeclareOption{authoryear}{%
  \gbt@obsolete@option{authoryear}%
  \gbt@legacy@interfacetrue
  \gbt@numericalfalse
}
```

将选项传递给 natbib

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

调用宏包，注意只需要 `compress` 不需要 `sort`。

```
\RequirePackage{natbib}
\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。

```
\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。

```
\renewcommand\newblock{\space}
\newcommand\bibstyle@super{\bibpunct{[}{]}{,}{s}{,}{\textsuperscript{,}}}
\newcommand\bibstyle@numbers{\bibpunct{[}{]}{,}{n}{,}{,}}
\newcommand\bibstyle@authoryear{\bibpunct{(}{)}{;}{a}{,}{,}}
\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。

```
\@namedef{bibstyle@gbt7714-numerical}{\bibstyle@super}
\@namedef{bibstyle@gbt7714-author-year}{\bibstyle@authoryear}
\@namedef{bibstyle@gbt7714-2005-numerical}{\bibstyle@super}
\@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。

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

将 numbers 样式的 `\citep` 的页码置于括号外。

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

Numerical 模式的 `\citet` 的页码:

```
\def\NAT@citexnum[#1][#2]#3{%
  \NAT@reset@parser
  \NAT@sort@cites{#3}%
  \NAT@reset@citea
  \@cite{\def\NAT@num{-1}\let\NAT@last@yr\relax\let\NAT@nm\@empty
    \@for\@citeb:=\NAT@cite@list\do
    {\@safe@activestrue
     \edef\@citeb{\expandafter\@firstofone\@citeb\@empty}%
     \@safe@activesfalse
     \@ifundefined{b@\@citeb\@extra@b@citeb}{%
       {\reset@font\bfseries?}
        \NAT@citeundefined\PackageWarning{natbib}%
       {Citation `\@citeb' on page \thepage \space undefined}}%
     {\let\NAT@last@num\NAT@num\let\NAT@last@nm\NAT@nm
      \NAT@parse{\@citeb}%
      \ifNAT@longnames\@ifundefined{bv@\@citeb\@extra@b@citeb}{%
        \let\NAT@name=\NAT@all@names
        \global\@namedef{bv@\@citeb\@extra@b@citeb}{}}{}%
```

```
\fi
\ifNAT@full\let\NAT@nm\NAT@all@names\else
  \let\NAT@nm\NAT@name\fi
\ifNAT@swa
 \@ifnum{\NAT@ctype>\@ne}{%
  \@citea
  \NAT@hyper@{\@ifnum{\NAT@ctype=\tw@}{\NAT@test{\NAT@ctype}}{\NAT@alias}}%
 }{%
  \@ifnum{\NAT@cmprs>\z@}{%
   \NAT@ifcat@num\NAT@num
    {\let\NAT@nm=\NAT@num}%
    {\def\NAT@nm{-2}}%
   \NAT@ifcat@num\NAT@last@num
    {\@tempcnta=\NAT@last@num\relax}%
    {\@tempcnta\m@ne}%
   \@ifnum{\NAT@nm=\@tempcnta}{%
    \@ifnum{\NAT@merge>\@ne}{}{\NAT@last@yr@mbox}%
   }{%
     \advance\@tempcnta by\@ne
     \@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。

```
       % \ifx\NAT@last@yr\relax
       %   \def@NAT@last@yr{\@citea}%
       % \else
       %   \def@NAT@last@yr{--\NAT@penalty}%
       % \fi
       \def@NAT@last@yr{-\NAT@penalty}%
     }{%
       \NAT@last@yr@mbox
     }%
   }%
  }{%
   \@tempswatrue
   \@ifnum{\NAT@merge>\@ne}{\@ifnum{\NAT@last@num=\NAT@num\relax}{\@tempswafalse}{}}{}%
   \if@tempswa\NAT@citea@mbox\fi
  }%
 }%
 \NAT@def@citea
\else
  \ifcase\NAT@ctype
    \ifx\NAT@last@nm\NAT@nm \NAT@yrsep\NAT@penalty\NAT@space\else
```

```
            \@citea \NAT@test{\@ne}\NAT@spacechar\NAT@mbox{\NAT@super@kern\NAT@@open}%
          \fi
          \if*#1*\else#1\NAT@spacechar\fi
          \NAT@mbox{\NAT@hyper@{{\citenumfont{\NAT@num}}}}%
          \NAT@def@citea@box
        \or
          \NAT@hyper@citea@space{\NAT@test{\NAT@ctype}}%
        \or
          \NAT@hyper@citea@space{\NAT@test{\NAT@ctype}}%
        \or
          \NAT@hyper@citea@space\NAT@alias
        \fi
      \fi
     }%
    }%
      \@ifnum{\NAT@cmprs>\z@}{\NAT@last@yr}{}%
      \ifNAT@swa\else
```

将页码放在括号外边，并且置于上标。

```
        % \@ifnum{\NAT@ctype=\z@}{%
        %   \if*#2*\else\NAT@cmt#2\fi
        % }{}%
        \NAT@mbox{\NAT@@close}%
        \@ifnum{\NAT@ctype=\z@}{%
          \if*#2*\else
            \textsuperscript{#2}%
          \fi
        }{}%
        \NAT@super@kern
      \fi
  }{#1}{#2}%
}%
```

Author-year 模式的 `\citep` 的页码：

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

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

Author-year 模式的 `\citet` 的页码：

```
\def\NAT@citex%
  [#1][#2]#3{%
  \NAT@reset@parser
  \NAT@sort@cites{#3}%
```

```
\NAT@reset@citea
\@cite{\let\NAT@nm\@empty\let\NAT@year\@empty
  \@for\@citeb:=\NAT@cite@list\do
  {\@safe@activestrue
   \edef\@citeb{\expandafter\@firstofone\@citeb\@empty}%
   \@safe@activesfalse
   \@ifundefined{b@\@citeb\@extra@b@citeb}{\@citea%
     {\reset@font\bfseries ?}\NAT@citeundefined
               \PackageWarning{natbib}%
     {Citation `\@citeb' on page \thepage \space undefined}\def\NAT@date{}}%
   {\let\NAT@last@nm=\NAT@nm\let\NAT@last@yr=\NAT@year
    \NAT@parse{\@citeb}%
    \ifNAT@longnames\@ifundefined{bv@\@citeb\@extra@b@citeb}{%
      \let\NAT@name=\NAT@all@names
      \global\@namedef{bv@\@citeb\@extra@b@citeb}{}}{}%
    \fi
   \ifNAT@full\let\NAT@nm\NAT@all@names\else
     \let\NAT@nm\NAT@name\fi
   \ifNAT@swa\ifcase\NAT@ctype
     \if\relax\NAT@date\relax
       \@citea\NAT@hyper@{\NAT@nmfmt{\NAT@nm}\NAT@date}%
     \else
       \ifx\NAT@last@nm\NAT@nm\NAT@yrsep
          \ifx\NAT@last@yr\NAT@year
            \def\NAT@temp{{?}}%
            \ifx\NAT@temp\NAT@exlab\PackageWarningNoLine{natbib}%
             {Multiple citation on page \thepage: same authors and
             year\MessageBreak without distinguishing extra
             letter,\MessageBreak appears as question mark}\fi
            \NAT@hyper@{\NAT@exlab}%
          \else\unskip\NAT@spacechar
            \NAT@hyper@{\NAT@date}%
          \fi
       \else
         \@citea\NAT@hyper@{%
           \NAT@nmfmt{\NAT@nm}%
           \hyper@natlinkbreak{%
             \NAT@aysep\NAT@spacechar}{\@citeb\@extra@b@citeb
           }%
           \NAT@date
         }%
       \fi
  \fi
\or\@citea\NAT@hyper@{\NAT@nmfmt{\NAT@nm}}%
\or\@citea\NAT@hyper@{\NAT@date}%
\or\@citea\NAT@hyper@{\NAT@alias}%
\fi \NAT@def@citea
\else
  \ifcase\NAT@ctype
   \if\relax\NAT@date\relax
     \@citea\NAT@hyper@{\NAT@nmfmt{\NAT@nm}}%
   \else
    \ifx\NAT@last@nm\NAT@nm\NAT@yrsep
       \ifx\NAT@last@yr\NAT@year
         \def\NAT@temp{{?}}%
         \ifx\NAT@temp\NAT@exlab\PackageWarningNoLine{natbib}%
          {Multiple citation on page \thepage: same authors and
          year\MessageBreak without distinguishing extra
          letter,\MessageBreak appears as question mark}\fi
         \NAT@hyper@{\NAT@exlab}%
       \else
         \unskip\NAT@spacechar
         \NAT@hyper@{\NAT@date}%
       \fi
    \else
      \@citea\NAT@hyper@{%
        \NAT@nmfmt{\NAT@nm}%
        \hyper@natlinkbreak{\NAT@spacechar\NAT@@open\if*#1*\else#1\NAT@spacechar\fi}%
          {\@citeb\@extra@b@citeb}%
        \NAT@date
      }%
    \fi
   \fi
  \or\@citea\NAT@hyper@{\NAT@nmfmt{\NAT@nm}}%
  \or\@citea\NAT@hyper@{\NAT@date}%
  \or\@citea\NAT@hyper@{\NAT@alias}%
  \fi
  \if\relax\NAT@date\relax
    \NAT@def@citea
  \else
    \NAT@def@citea@close
  \fi
\fi
}}\ifNAT@swa\else
```

将页码放在括号外边，并且置于上标。

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

`thebibliography` (*env.*) 参考文献列表的标签左对齐

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

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

```
\let\NAT@bare@aux\NAT@bare
\def\NAT@bare#1(#2){%
 \begingroup\edef\x{\endgroup
   \unexpanded{\NAT@bare@aux#1}(\@firstofone#2)}\x}
```

`\url` 使用 xurl 宏包的方法，增加 URL 可断行的位置。

```
\g@addto@macro\UrlBreaks{%
  \do0\do1\do2\do3\do4\do5\do6\do7\do8\do9%
  \do\A\do\B\do\C\do\D\do\E\do\F\do\G\do\H\do\I\do\J\do\K\do\L\do\M
  \do\N\do\O\do\P\do\Q\do\R\do\S\do\T\do\U\do\V\do\W\do\X\do\Y\do\Z
  \do\a\do\b\do\c\do\d\do\e\do\f\do\g\do\h\do\i\do\j\do\k\do\l\do\m
  \do\n\do\o\do\p\do\q\do\r\do\s\do\t\do\u\do\v\do\w\do\x\do\y\do\z
}
\Urlmuskip=0mu plus 0.1mu
```

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

兼容 v2.0 前过时的接口：

```
\newif\ifgbt@bib@style@written
\@ifpackageloaded{chapterbib}{}{%
  \def\bibliography#1{%
    \ifgbt@bib@style@written\else
      \bibliographystyle{gbt7714-numerical}%
    \fi
    \if@filesw
      \immediate\write\@auxout{\string\bibdata{\zap@space#1 \@empty}}%
    \fi
    \@input@{\jobname.bbl}}
  \def\bibliographystyle#1{%
    \gbt@bib@style@writtentrue
    \ifx\@begindocumenthook\@undefined\else
      \expandafter\AtBeginDocument
    \fi
    {\if@filesw
      \immediate\write\@auxout{\string\bibstyle{#1}}%
    \fi}%
  }%
}
\ifgbt@legacy@interface
  \ifgbt@numerical
    \ifgbt@super\else
      \citestyle{numbers}
    \fi
    \bibliographystyle{gbt7714-numerical}
  \else
    \bibliographystyle{gbt7714-author-year}
  \fi
\fi
⟨/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。

```
⟨*author-year | numerical⟩
INTEGERS {
  citation.et.al.min
  citation.et.al.use.first
  bibliography.et.al.min
  bibliography.et.al.use.first
  uppercase.name
  terms.in.macro
  year.after.author
  period.after.author
  italic.book.title
  sentence.case.title
  link.title
  title.in.journal
  show.patent.country
  show.mark
  space.before.mark
  show.medium.type
  short.journal
  italic.journal
  link.journal
  bold.journal.volume
  show.missing.address.publisher
  space.before.pages
  only.start.page
  wave.dash.in.pages
  show.urldate
  show.url
  show.doi
  show.preprint
  show.note
  show.english.translation
  end.with.period
⟨*author-year⟩
  lang.zh.order
  lang.ja.order
  lang.en.order
  lang.ru.order
  lang.other.order
⟨/author-year⟩
}

STRINGS {
  component.part.label
}

```

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。

```
FUNCTION {load.config}
{
```

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”。

```
⟨*!ucas⟩
  #2 'citation.et.al.min :=
  #1 'citation.et.al.use.first :=
⟨/!ucas⟩
⟨*ucas⟩
  #3 'citation.et.al.min :=
  #1 'citation.et.al.use.first :=
⟨/ucas⟩
  #4 'bibliography.et.al.min :=
  #3 'bibliography.et.al.use.first :=
```

英文姓名转为全大写：

```
⟨*!(no-uppercase | thu)⟩
  #1 'uppercase.name :=
⟨/!(no-uppercase | thu)⟩
⟨*no-uppercase | thu⟩
  #0 'uppercase.name :=
⟨/no-uppercase | thu⟩
```

使用 TeX 宏输出“和”、“等”

```
⟨*!(macro | ucas)⟩
  #0 'terms.in.macro :=
⟨/!(macro | ucas)⟩
⟨*macro | ucas⟩
```

```
  #1 'terms.in.macro :=
⟨/macro | ucas⟩
```

将年份置于著者后面（著者-出版年制默认）

```
⟨*numerical | ucas⟩
  #0 'year.after.author :=
⟨/numerical | ucas⟩
⟨*author-year&!ucas⟩
  #1 'year.after.author :=
⟨/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：

```
⟨*numerical⟩
  #1 'period.after.author :=
⟨/numerical⟩
⟨*author-year⟩
⟨*2015&!(period | ustc)⟩
  #0 'period.after.author :=
⟨/2015&!(period | ustc)⟩
⟨*period | 2005 | ustc⟩
  #1 'period.after.author :=
⟨/period | 2005 | ustc⟩
⟨/author-year⟩
```

书名使用斜体：

```
⟨*!italic-book-title⟩
  #0 'italic.book.title :=
⟨/!italic-book-title⟩
⟨*italic-book-title⟩
  #1 'italic.book.title :=
⟨/italic-book-title⟩
```

英文标题转为 sentence case （句首字母大写，其余小写）：

```
⟨*!no-sentence-case⟩
  #1 'sentence.case.title :=
⟨/!no-sentence-case⟩
⟨*no-sentence-case⟩
  #0 'sentence.case.title :=
⟨/no-sentence-case⟩
```

在标题添加超链接：

```
⟨*!link-title⟩
  #0 'link.title :=
⟨/!link-title⟩
⟨*link-title⟩
  #1 'link.title :=
⟨/link-title⟩
```

期刊是否含标题：

```
⟨*!no-title-in-journal⟩
  #1 'title.in.journal :=
⟨/!no-title-in-journal⟩
⟨*no-title-in-journal⟩
  #0 'title.in.journal :=
⟨/no-title-in-journal⟩
```

专利题名是否含专利国别

```
⟨∗!(show-patent-country | 2005 | ustc | thu)⟩
  #0 'show.patent.country :=
⟨/!(show-patent-country | 2005 | ustc | thu)⟩
⟨∗(show-patent-country | 2005 | ustc | thu)⟩
  #1 'show.patent.country :=
⟨/(show-patent-country | 2005 | ustc | thu)⟩
```

著录文献类型标识（比如“[M/OL]“）：

```
⟨∗!no-mark⟩
  #1 'show.mark :=
⟨/!no-mark⟩
⟨∗no-mark⟩
  #0 'show.mark :=
⟨/no-mark⟩
```

文献类型标识前是否有空格：

```
⟨∗!space-before-mark⟩
  #0 'space.before.mark :=
⟨/!space-before-mark⟩
⟨∗space-before-mark⟩
  #1 'space.before.mark :=
⟨/space-before-mark⟩
```

是否显示载体类型标识（比如“/OL“）：

```
⟨∗!no-medium-type⟩
  #1 'show.medium.type :=
⟨/!no-medium-type⟩
⟨∗no-medium-type⟩
  #0 'show.medium.type :=
⟨/no-medium-type⟩
```

使用“//”表示析出文献

```
⟨∗!(in-collection | no-slash)⟩
  "slash" 'component.part.label :=
⟨/!(in-collection | no-slash)⟩
⟨∗in-collection⟩
  "in" 'component.part.label :=
⟨/in-collection⟩
⟨∗no-slash⟩
  "none" 'component.part.label :=
⟨/no-slash⟩
```

期刊名使用缩写：

```
⟨∗!short-journal⟩
  #0 'short.journal :=
⟨/!short-journal⟩
⟨∗short-journal⟩
  #1 'short.journal :=
⟨/short-journal⟩
```

期刊名使用斜体：

```
⟨∗!italic-journal⟩
  #0 'italic.journal :=
⟨/!italic-journal⟩
⟨*italic-journal⟩
  #1 'italic.journal :=
⟨/italic-journal⟩
```

在期刊题名添加超链接：

```
⟨*!link-journal⟩
  #0 'link.journal :=
⟨/!link-journal⟩
⟨*link-journal⟩
  #1 'link.journal :=
⟨/link-journal⟩
```

期刊的卷使用粗体：

```
  #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.”:

```
⟨*!sl-sn⟩
  #0 'show.missing.address.publisher :=
⟨/!sl-sn⟩
⟨*sl-sn⟩
  #1 'show.missing.address.publisher :=
⟨/sl-sn⟩
```

页码与前面的冒号之间是否有空格：

```
⟨*!no-space-before-pages⟩
  #1 'space.before.pages :=
⟨/!no-space-before-pages⟩
⟨*no-space-before-pages⟩
  #0 'space.before.pages :=
⟨/no-space-before-pages⟩
```

页码是否只含起始页：

```
⟨*!only-start-page⟩
  #0 'only.start.page :=
⟨/!only-start-page⟩
⟨*only-start-page⟩
  #1 'only.start.page :=
⟨/only-start-page⟩
```

起止页码使用波浪号：

```
⟨*!wave-dash-in-pages⟩
  #0 'wave.dash.in.pages :=
⟨/!wave-dash-in-pages⟩
⟨*wave-dash-in-pages⟩
  #1 'wave.dash.in.pages :=
⟨/wave-dash-in-pages⟩
```

是否著录非电子文献的引用日期：

```
⟨*!no-urldate⟩
  #1 'show.urldate :=
⟨/!no-urldate⟩
⟨*no-urldate⟩
  #0 'show.urldate :=
⟨/no-urldate⟩
```

是否著录 URL：

```
⟨*!no-url⟩
  #1 'show.url :=
⟨/!no-url⟩
⟨*no-url⟩
  #0 'show.url :=
⟨/no-url⟩
```

是否著录 DOI：

```
⟨*!(no-doi | 2005)⟩
  #1 'show.doi :=
⟨/!(no-doi | 2005)⟩
⟨*no-doi | 2005⟩
  #0 'show.doi :=
⟨/no-doi | 2005⟩
```

是否著录 e-print：

```
⟨*!preprint⟩
  #1 'show.preprint :=
⟨/!preprint⟩
⟨*preprint⟩
  #0 'show.preprint :=
⟨/preprint⟩
```

在每一条文献最后输出注释（note）的内容：

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

中文文献是否显示英文翻译

```
⟨*!show-english-translation⟩
  #0 'show.english.translation :=
⟨/!show-english-translation⟩
⟨*show-english-translation⟩
  #1 'show.english.translation :=
⟨/show-english-translation⟩
```

结尾加句点

```
⟨*!no-period-at-end⟩
  #1 'end.with.period :=
⟨/!no-period-at-end⟩
⟨*no-period-at-end⟩
  #0 'end.with.period :=
⟨/no-period-at-end⟩
```

参考文献表按照“著者-出版年”组织时，各个文种的顺序：

```
⟨*author-year⟩
  #1 'lang.zh.order :=
  #2 'lang.ja.order :=
  #3 'lang.en.order :=
  #4 'lang.ru.order :=
  #5 'lang.other.order :=
⟨/author-year⟩
}

```

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.

```
ENTRY
  { address
    archivePrefix
    author
    booktitle
    date
    doi
    edition
    editor
    eprint
    eprinttype
    entrysubtype
    howpublished
    institution
    journal
    journaltitle
    key
    langid
    language
    location
    mark
    medium
    note
    number
    organization
    pages
    publisher
    school
    series
    shortjournal
    title
    translation
    translator
    url
    urldate
    volume
    year
  }
  { 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.

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

```

B.3 Entry functions

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

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

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

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

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

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

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

```
output.nonnull(s) ==
 BEGIN
      s := argument on stack
      if output.state = mid.sentence then
          write$(pop() * ",␣")
                -- "pop" isn't␣a␣function:␣just␣use␣stack␣top
␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣if␣output.state␣=␣after.block␣then
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣write$(add.period$(pop()))
␣␣␣␣␣␣␣␣␣␣␣␣␣␣newline$
␣␣␣␣␣␣␣␣␣␣␣␣␣␣write$("\newblock␣")
␣␣␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣output.state␣=␣before.all␣then
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣write$(pop())
␣␣␣␣␣␣␣␣␣␣␣␣␣␣else␣␣␣␣␣␣␣␣--␣output.state␣should␣be␣after.sentence
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣write$(add.period$(pop())␣*␣"␣")
␣␣␣␣␣␣␣␣␣␣␣␣␣␣fi
␣␣␣␣␣␣␣␣␣␣fi
␣␣␣␣␣␣␣␣␣␣output.state␣:=␣mid.sentence
␣␣␣␣␣␣fi
␣␣␣␣␣␣push␣s␣on␣stack
␣END
```

The output function calls output.nonnull if its argument is non-empty; its argument may be a missing field (thus, not necessarily a string)

```
output(s) ==
 BEGIN
      if not empty$(s) then output.nonnull(s)
      fi
 END
```

The output.check function is the same as the output function except that, if necessary, output.check warns the user that the t field shouldn't be empty (this is because it probably won't be a good reference without the field; the entry functions try to make the formatting look reasonable even when such fields are empty).

```
output.check(s,t) ==
 BEGIN
      if empty$(s) then
          warning$("empty␣" * t * "␣in␣" * cite$)
      else output.nonnull(s)
      fi
 END
```

The output.bibitem function writes the `\bibitem` for the current entry (the label should already have been set up), and sets up the separator state for the output functions. And, it leaves a string on the stack as per the output convention.

```
output.bibitem ==
 BEGIN
      newline$
      write$("\bibitem[")     % for alphabetic labels,
      write$(label)           % these three lines
      write$("]{")            % are used
      write$("\bibitem{")             % this line for numeric labels
      write$(cite$)
      write$("}")
      push "" on stack
      output.state := before.all
 END
```

The fin.entry function finishes off an entry by adding a period to the string remaining on the stack. If the state is still before.all then nothing was produced for this entry, so the result will look bad, but the user deserves it. (We don't omit the whole entry because the entry was cited, and a bibitem is needed to define the citation label.)

```
fin.entry ==
 BEGIN
      write$(add.period$(pop()))
      newline$
 END
```

The new.block function prepares for a new block to be output, and new.sentence prepares for a new sentence.

```
new.block ==
 BEGIN
      if output.state <> before.all then
          output.state := after.block
      fi
 END
```

```
new.sentence ==
 BEGIN
      if output.state <> after.block then
          if output.state <> before.all then
              output.state :=  after.sentence
          fi
      fi
 END
```

```
INTEGERS { output.state before.all mid.sentence after.sentence after.block after.slash }

INTEGERS { lang.zh lang.ja lang.en lang.ru lang.other }

INTEGERS { charptr len }

FUNCTION {init.state.consts}
{ #0 'before.all :=
  #1 'mid.sentence :=
  #2 'after.sentence :=
  #3 'after.block :=
  #4 'after.slash :=
  #3 'lang.zh :=
  #4 'lang.ja :=
  #1 'lang.en :=
  #2 'lang.ru :=
  #0 'lang.other :=
}

```

下面是一些常量的定义

```
FUNCTION {bbl.anonymous}
{ entry.lang lang.zh =
    { "佚名" }
    { "Anon" }
  if$
}

FUNCTION {bbl.space}
{ entry.lang lang.zh =
    { "\ " }
    { " " }
  if$
}

FUNCTION {bbl.and}
{ "" }

FUNCTION {bbl.et.al}
{ entry.lang lang.zh =
    { "等" }
    { entry.lang lang.ja =
        { "他" }
        { entry.lang lang.ru =
            { "идр" }
            { "et~al." }
          if$
        }
      if$
    }
  if$
}

FUNCTION {citation.and}
{ terms.in.macro
    { "{\biband}" }
    'bbl.and
  if$
}

FUNCTION {citation.et.al}
{ terms.in.macro
    { "{\bibetal}" }
    'bbl.et.al
  if$
}

FUNCTION {bbl.colon} { ": " }

FUNCTION {bbl.pages.colon}
{ space.before.pages
    { ": " }
    { ":\allowbreak " }
  if$
}
```

```
⟨*!2005⟩
FUNCTION {bbl.wide.space} { "\quad " }
⟨/!2005⟩
⟨*2005⟩
FUNCTION {bbl.wide.space} { "\ " }
⟨/2005⟩

FUNCTION {bbl.slash} { "//\allowbreak " }

FUNCTION {bbl.sine.loco}
{ entry.lang lang.zh =
    { "[出版地不详]" }
    { "[S.l.]" }
  if$
}

FUNCTION {bbl.sine.nomine}
{ entry.lang lang.zh =
    { "[出版者不详]" }
    { "[s.n.]" }
  if$
}

FUNCTION {bbl.sine.loco.sine.nomine}
{ entry.lang lang.zh =
    { "[出版地不详: 出版者不详]" }
    { "[S.l.: s.n.]" }
  if$
}

```

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

```
FUNCTION {not}
{   { #0 }
    { #1 }
  if$
}

FUNCTION {and}
{   'skip$
    { pop$ #0 }
  if$
}

FUNCTION {or}
{   { pop$ #1 }
    'skip$
  if$
}

STRINGS { x y }
```

```
FUNCTION {contains}
{ 'y :=
  'x :=
  y text.length$ 'len :=
  x text.length$ len - #1 + 'charptr :=
    { charptr #0 >
      x charptr len substring$ y = not
      and
    }
    { charptr #1 - 'charptr := }
  while$
  charptr #0 >
}

```

the variables s and t are temporary string holders

```
STRINGS { s t }

FUNCTION {output.nonnull}
{ 's :=
  output.state mid.sentence =
    { ", " * write$ }
    { output.state after.block =
        { add.period$ write$
          newline$
          "\newblock " write$
        }
        { output.state before.all =
            'write$
            { output.state after.slash =
                { bbl.slash * write$
                  newline$
                }
                { add.period$ " " * write$ }
              if$
            }
          if$
        }
      if$
      mid.sentence 'output.state :=
    }
  if$
  s
}

FUNCTION {output}
{ duplicate$ empty$
    'pop$
    'output.nonnull
  if$
}

FUNCTION {output.after}
{ 't :=
  duplicate$ empty$
    'pop$
    { 's :=
      output.state mid.sentence =
        { t * write$ }
        { output.state after.block =
            { add.period$ write$
              newline$
              "\newblock " write$
            }
            { output.state before.all =
                'write$
                { output.state after.slash =
                    { bbl.slash * write$ }
                    { add.period$ " " * write$ }
                  if$
                }
              if$
            }
          if$
          mid.sentence 'output.state :=
        }
      if$
      s
    }
  if$
}

FUNCTION {output.check}
{ 't :=
  duplicate$ empty$
    { pop$ "empty " t * " in " * cite$ * warning$ }
    'output.nonnull
  if$
}

```

This function finishes all entries.

```
FUNCTION {fin.entry}
{ end.with.period
    'add.period$
    'skip$
  if$
  write$
  show.english.translation entry.lang lang.zh = and
    { ")"
      write$
    }
    'skip$
  if$
  newline$
}

FUNCTION {new.block}
{ output.state before.all =
    'skip$
    { output.state after.slash =
        'skip$
        { after.block 'output.state := }
      if$
    }
  if$
}

FUNCTION {new.sentence}
{ output.state after.block =
    'skip$
    { output.state before.all =
        'skip$
        { output.state after.slash =
            'skip$
            { after.sentence 'output.state := }
          if$
        }
      if$
    }
  if$
}

FUNCTION {new.slash}
{ output.state before.all =
    'skip$
    { component.part.label "slash" =
        { after.slash 'output.state := }
        { new.block
          component.part.label "in" =
            { entry.lang lang.en =
                { "In: " output
                  write$
                  ""
                  before.all 'output.state :=
                }
                'skip$
              if$
            }
            'skip$
          if$
        }
      if$
    }
  if$
}

```

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.

```
FUNCTION {new.block.checka}
{ empty$
```

```
    'skip$
    'new.block
  if$
}

FUNCTION {new.block.checkb}
{ empty$
  swap$ empty$
  and
    'skip$
    'new.block
  if$
}

```

The new.sentence.check functions are analogous.

```
FUNCTION {new.sentence.checka}
{ empty$
    'skip$
    'new.sentence
  if$
}

FUNCTION {new.sentence.checkb}
{ empty$
  swap$ empty$
  and
    'skip$
    'new.sentence
  if$
}

```

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 emphazised, if that is non-empty, otherwise it returns the null string. Italic corrections aren't used, so this function should be used when punctation 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

```
FUNCTION {field.or.null}
{ duplicate$ empty$
    { pop$ "" }
    'skip$
  if$
}

FUNCTION {emphasize}
{ duplicate$ empty$
    { pop$ "" }
    { "\emph{" swap$ * "}" * }
  if$
}

FUNCTION {format.btitle}
{ italic.book.title
  entry.lang lang.en = and
    'emphasize
    'skip$
  if$
}

```

B.4.1 Detect Language

```
INTEGERS { byte second.byte }

INTEGERS { char.lang tmp.lang }

STRINGS { tmp.str }

FUNCTION {get.str.lang}
{ 'tmp.str :=
  lang.other 'tmp.lang :=
  #1 'charptr :=
  tmp.str text.length$ #1 + 'len :=
    { charptr len < }
    { tmp.str charptr #1 substring$ chr.to.int$ 'byte :=
      byte #128 <
        { charptr #1 + 'charptr :=
          byte #64 > byte #91 < and byte #96 > byte #123 < and or
            { lang.en 'char.lang := }
            { lang.other 'char.lang := }
          if$
        }
        { tmp.str charptr #1 + #1 substring$ chr.to.int$ 'second.byte :=
          byte #224 <
```

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

```
    { charptr #2 + 'charptr :=
      byte #207 > byte #212 < and
      byte #212 = second.byte #176 < and or
        { lang.ru 'char.lang := }
        { lang.other 'char.lang := }
      if$
    }
    { byte #240 <
```

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

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

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

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

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

```
                    { second.byte #128 > second.byte #132 < and
                        { lang.ja 'char.lang := }
                        { lang.other 'char.lang := }
                      if$
                    }
                  if$
                }
```

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

```
                { byte #239 =
                  second.byte #163 > second.byte #172 < and and
                    { lang.zh 'char.lang := }
                    { lang.other 'char.lang := }
                  if$
                }
              if$
            }
          if$
        }
```

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.

```
        { charptr #4 + 'charptr :=
          byte #240 = second.byte #159 > and
            { lang.zh 'char.lang := }
            { lang.other 'char.lang := }
          if$
```

```
                }
              if$
            }
          if$
        }
      if$
      char.lang tmp.lang >
        { char.lang 'tmp.lang := }
        'skip$
      if$
    }
  while$
  tmp.lang
}

FUNCTION {check.entry.lang}
{ author field.or.null
  title field.or.null *
  get.str.lang
}

STRINGS { entry.langid }

FUNCTION {set.entry.lang}
{ "" 'entry.langid :=
  language empty$ not
    { language 'entry.langid := }
    'skip$
  if$
  langid empty$ not
    { langid 'entry.langid := }
    'skip$
  if$
  entry.langid empty$
    { check.entry.lang }
    { entry.langid "english" = entry.langid "american" = or entry.langid "british" = or
        { lang.en }
        { entry.langid "chinese" =
            { lang.zh }
            { entry.langid "japanese" =
                { lang.ja }
                { entry.langid "russian" =
                    { lang.ru }
                    { check.entry.lang }
                  if$
                }
              if$
            }
          if$
        }
      if$
    }
  if$
  'entry.lang :=
}
```

```

FUNCTION {set.entry.numbered}
{ type$ "patent" =
  type$ "standard" = or
  type$ "techreport" = or
    { #1 'entry.numbered := }
    { #0 'entry.numbered := }
  if$
}

```

B.4.2 Format names

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

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

format.names(s) ==
 BEGIN
      nameptr := 1
      numnames := num.names$(s)
      namesleft := numnames
      while namesleft > 0
        do
                              % for full names:
          t := format.name$(s, nameptr, "{ff~}{vv~}{ll}{,␣jj}")
                              % for abbreviated first names:
          t := format.name$(s, nameptr, "{f.~}{vv~}{ll}{,␣jj}")
          if nameptr > 1 then
              if namesleft > 1 then nameresult := nameresult ∗ ",␣" ∗ t
              else if numnames > 2
                     then nameresult := nameresult ∗ ","
                   fi
                   if t = "others"
                     then nameresult := nameresult ∗ "␣et~al."
                     else nameresult := nameresult ∗ "␣and␣" ∗ t
                   fi
              fi
          else nameresult := t
          fi
          nameptr := nameptr + 1
          namesleft := namesleft - 1
        od
      return nameresult
 END
```

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

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

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

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

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

```
INTEGERS { nameptr namesleft numnames name.lang }

FUNCTION {format.name}
{ "{vv~}{ll}{, jj}{, ff}" format.name$ 't :=
  t "others" =
    { bbl.et.al }
    { t get.str.lang 'name.lang :=
      name.lang lang.en =
        { t #1 "{vv~}{ll}{ f{~}}" format.name$
          uppercase.name
            { "u" change.case$ }
            'skip$
          if$
          t #1 "{, jj}" format.name$ *
        }
        { t #1 "{ll}{ff}" format.name$ }
      if$
    }
  if$
}

FUNCTION {format.names}
{ 's :=
  #1 'nameptr :=
  s num.names$ 'numnames :=
  ""
  numnames 'namesleft :=
    { namesleft #0 > }
    { s nameptr format.name bbl.et.al =
      numnames bibliography.et.al.min #1 - > nameptr bibliography.et.al.use.first > and or
```

```
        { ", " *
          bbl.et.al *
          #1 'namesleft :=
        }
        { nameptr #1 >
            { namesleft #1 = bbl.and "" = not and
                { bbl.and * }
                { ", " * }
              if$
            }
            'skip$
          if$
          s nameptr format.name *
        }
      if$
      nameptr #1 + 'nameptr :=
      namesleft #1 - 'namesleft :=
    }
  while$
}

FUNCTION {format.key}
{ empty$
    { key field.or.null }
    { "" }
  if$
}

FUNCTION {format.authors}
{ author empty$ not
    { author format.names }
    { "empty author in " cite$ * warning$
⟨*author-year⟩
      bbl.anonymous
⟨/author-year⟩
⟨*numerical⟩
      ""
⟨/numerical⟩
    }
  if$
}

FUNCTION {format.editors}
{ editor empty$
    { "" }
    { editor format.names }
  if$
}

FUNCTION {format.translators}
{ translator empty$
    { "" }
    { translator format.names
      entry.lang lang.zh =
        { translator num.names$ #3 >
```

```
          { " 译" * }
          { ", 译" * }
        if$
      }
      'skip$
    if$
  }
  if$
}

FUNCTION {format.full.names}
{'s :=
  #1 'nameptr :=
  s num.names$ 'numnames :=
  numnames 'namesleft :=
    { namesleft #0 > }
    { s nameptr "{vv~}{ll}{, jj}{, ff}" format.name$ 't :=
      t get.str.lang 'name.lang :=
      name.lang lang.en =
        { t #1 "{vv~}{ll}" format.name$ 't := }
        { t #1 "{ll}{ff}" format.name$ 't := }
      if$
      nameptr #1 >
        {
          namesleft #1 >
            { ", " * t * }
            {
              numnames #2 >
                { "," * }
                'skip$
              if$
              t "others" =
                { " et~al." * }
                { " and " * t * }
              if$
            }
          if$
        }
        't
      if$
      nameptr #1 + 'nameptr :=
      namesleft #1 - 'namesleft :=
    }
  while$
}

FUNCTION {author.editor.full}
{ author empty$
    { editor empty$
        { "" }
        { editor format.full.names }
      if$
    }
    { author format.full.names }
  if$
```

```
}

FUNCTION {author.full}
{ author empty$
    { "" }
    { author format.full.names }
  if$
}

FUNCTION {editor.full}
{ editor empty$
    { "" }
    { editor format.full.names }
  if$
}

FUNCTION {make.full.names}
{ type$ "book" =
  type$ "inbook" = booktitle empty$ not and
  or
    'author.editor.full
    { type$ "collection" =
      type$ "proceedings" =
      or
        'editor.full
        'author.full
      if$
    }
  if$
}

FUNCTION {output.bibitem}
{ newline$
  "\bibitem[" write$
  label ")" *
  make.full.names duplicate$ short.list =
    { pop$ }
    { duplicate$ "]" contains
        { "{" swap$ * "}" * }
        'skip$
      if$
      *
    }
  if$
  "]{" * write$
  cite$ write$
  "}" write$
  newline$
  ""
  before.all 'output.state :=
}

```

B.4.3 Format title

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

```
FUNCTION {change.sentence.case}
{ entry.lang lang.en =
    { "t" change.case$ }
    'skip$
  if$
}

FUNCTION {add.link}
{ url empty$ not
    { "\href{" url * "}{" * swap$ * "}" * }
    { doi empty$ not
        { "\href{https://doi.org/" doi * "}{" * swap$ * "}" * }
        'skip$
      if$
    }
  if$
}

FUNCTION {format.title}
{ title empty$
    { "" }
    { title
      sentence.case.title
        'change.sentence.case
        'skip$
      if$
      entry.numbered number empty$ not and
        { bbl.colon *
          type$ "patent" = show.patent.country and
            { address empty$ not
                { address * ", " * }
                { location empty$ not
                    { location * ", " * }
                    { entry.lang lang.zh =
                        { " 中国" * ", " * }
                        'skip$
                      if$
                    }
                  if$
                }
              if$
            }
            'skip$
          if$
          number *
        }
        'skip$
      if$
      link.title
        'add.link
        'skip$
      if$
    }
  if$
}

```

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
```

```
FUNCTION {tie.or.space.connect}
{ duplicate$ text.length$ #3 <
    { "~" }
    { " " }
  if$
  swap$ * *
}

```

The either.or.check function complains if both fields or an either-or pair are nonempty.

```
either.or.check(t,s) ==
 BEGIN
      if empty$(s) then
          warning$(can't␣use␣both␣"␣*␣t␣*␣"␣fields␣in␣"␣*␣cite$)
␣␣␣␣␣␣fi
␣END
```

```
FUNCTION {either.or.check}
{ empty$
    'pop$
    { "can't use both " swap$ * " fields in " * cite$ * warning$ }
  if$
}

```

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

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

```
FUNCTION {is.digit}
{ duplicate$ empty$
    { pop$ #0 }
    { chr.to.int$
      duplicate$ "0" chr.to.int$ <
      { pop$ #0 }
      { "9" chr.to.int$ >
          { #0 }
          { #1 }
        if$
      }
    if$
    }
  if$
}

FUNCTION {is.number}
{ 's :=
  s empty$
    { #0 }
    { s text.length$ 'charptr :=
        { charptr #0 >
          s charptr #1 substring$ is.digit
          and
        }
        { charptr #1 - 'charptr := }
      while$
      charptr not
    }
  if$
}

FUNCTION {format.volume}
{ volume empty$ not
    { volume is.number
        { entry.lang lang.zh =
            { "第" volume * "卷" * }
            { "Vol." volume tie.or.space.connect }
          if$
        }
        { volume }
      if$
    }
    { "" }
  if$
}

FUNCTION {format.number}
{ number empty$ not
    { number is.number
        { entry.lang lang.zh =
            { " 第 " number * " 册" * }
            { "No." number tie.or.space.connect }
          if$
        }
        { number }
      if$
    }
    { "" }
  if$
}

FUNCTION {format.volume.number}
{ volume empty$ not
    { format.volume }
    { format.number }
  if$
}

FUNCTION {format.title.vol.num}
{ title
  sentence.case.title
    'change.sentence.case
    'skip$
  if$
  entry.numbered
    { number empty$ not
        { bbl.colon * number * }
        'skip$
      if$
    }
    { format.volume.number 's :=
      s empty$ not
        { bbl.colon * s * }
        'skip$
      if$
    }
  if$
}

FUNCTION {format.series.vol.num.title}
{ format.volume.number 's :=
  series empty$ not
```

```
    { series
      sentence.case.title
        'change.sentence.case
        'skip$
      if$
      entry.numbered
        { bbl.wide.space * }
        { bbl.colon *
          s empty$ not
            { s * bbl.wide.space * }
            'skip$
          if$
        }
      if$
      title *
      sentence.case.title
        'change.sentence.case
        'skip$
      if$
      entry.numbered number empty$ not and
        { bbl.colon * number * }
        'skip$
      if$
    }
    { format.title.vol.num }
  if$
  format.btitle
  link.title
    'add.link
    'skip$
  if$
}

FUNCTION {format.booktitle.vol.num}
{ booktitle
  entry.numbered
    'skip$
    { format.volume.number 's :=
      s empty$ not
        { bbl.colon * s * }
        'skip$
      if$
    }
  if$
}

FUNCTION {format.series.vol.num.booktitle}
{ format.volume.number 's :=
  series empty$ not
    { series bbl.colon *
      entry.numbered not s empty$ not and
        { s * bbl.wide.space * }
        'skip$
      if$
      booktitle *
    }
    { format.booktitle.vol.num }
  if$
  format.btitle
}

FUNCTION {remove.period}
{ 't :=
  "" 's :=
    { t empty$ not }
    { t #1 #1 substring$ 'tmp.str :=
      tmp.str "." = not
        { s tmp.str * 's := }
        'skip$
      if$
      t #2 global.max$ substring$ 't :=
    }
  while$
  s
}

FUNCTION {abbreviate}
{ remove.period
  't :=
  t "l" change.case$ 's :=
  ""
  s "physical review letters" =
    { "Phys Rev Lett" }
    'skip$
  if$
  's :=
  s empty$
    { t }
    { pop$ s }
  if$
}

FUNCTION {get.journal.title}
{ short.journal
    { shortjournal empty$ not
        { shortjournal }
        { journal empty$ not
            { journal abbreviate }
            { journaltitle empty$ not
                { journaltitle abbreviate }
                { "" }
              if$
            }
          if$
        }
      if$
    }
    { journal empty$ not
        { journal }
        { journaltitle empty$ not
```

```
            { journaltitle }
            { shortjournal empty$ not
                { shortjournal }
                { "" }
              if$
            }
          if$
        }
      if$
    }
  if$
}

FUNCTION {check.arxiv.preprint}
{ #1 #5 substring$ purify$ "l" change.case$ "arxiv" =
    { #1 }
    { #0 }
  if$
}

FUNCTION {format.journal}
{ get.journal.title
  duplicate$ empty$ not
    { italic.journal entry.lang lang.en = and
        'emphasize
        'skip$
      if$
      link.journal
        'add.link
        'skip$
      if$
    }
    'skip$
  if$
}

```

B.4.4 Format entry type mark

```
FUNCTION {set.entry.mark}
{ entry.mark empty$ not
    'pop$
    { mark empty$ not
        { pop$ mark 'entry.mark := }
        { 'entry.mark := }
      if$
    }
  if$
}

FUNCTION {format.mark}
{ show.mark
    { entry.mark
      show.medium.type
    { medium empty$ not
      { "/" * medium * }
      { entry.is.electronic
        { "/OL" * }
        'skip$
       if$
      }
     if$
    }
    'skip$
   if$
   'entry.mark :=
   space.before.mark
    { " " }
    { "\allowbreak" }
   if$
   "[" * entry.mark * "]" *
  }
  { "" }
 if$
}

```

B.4.5 Format edition

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

```
FUNCTION {num.to.ordinal}
{ duplicate$ text.length$ 'charptr :=
 duplicate$ charptr #1 substring$ 's :=
 s "1" =
  { "st" * }
  { s "2" =
    { "nd" * }
    { s "3" =
      { "rd" * }
      { "th" * }
     if$
    }
   if$
  }
 if$
}

FUNCTION {format.edition}
{ edition empty$
  { "" }
  { edition is.number
    { edition "1" = not
      { entry.lang lang.zh =
        { edition " 版" * }
        { edition num.to.ordinal " ed." * }
       if$
```

```
            }
            'skip$
          if$
        }
        { entry.lang lang.en =
            { edition change.sentence.case 's :=
              s "Revised" = s "Revised edition" = or
                { "Rev. ed." }
                { s " ed." * }
              if$
            }
            { edition }
          if$
        }
      if$
    }
  if$
}

```

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。

```
FUNCTION {format.publisher}
{ publisher empty$ not
    { publisher }
    { school empty$ not
        { school }
        { organization empty$ not
            { organization }
            { institution empty$ not
                { institution }
                { "" }
              if$
            }
          if$
        }
      if$
    }
  if$
}

FUNCTION {format.address.publisher}
{ address empty$ not
    { address }
    { location empty$ not
        { location }
        { "" }
      if$
    }
  if$
  duplicate$ empty$ not
    { format.publisher empty$ not
        { bbl.colon * format.publisher * }
        { entry.is.electronic not show.missing.address.publisher and
            { bbl.colon * bbl.sine.nomine * }
            'skip$
          if$
        }
      if$
    }
    { pop$
      entry.is.electronic not show.missing.address.publisher and
        { format.publisher empty$ not
            { bbl.sine.loco bbl.colon * format.publisher * }
            { bbl.sine.loco.sine.nomine }
          if$
        }
        { format.publisher empty$ not
            { format.publisher }
            { "" }
          if$
        }
      if$
    }
  if$
}

```

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。

```
FUNCTION {extract.before.dash}
{ duplicate$ empty$
    { pop$ "" }
    { 's :=
      #1 'charptr :=
      s text.length$ #1 + 'len :=
        { charptr len <
          s charptr #1 substring$ "-" = not
          and
        }
        { charptr #1 + 'charptr := }
      while$
      s #1 charptr #1 - substring$
    }
  if$
}

FUNCTION {extract.after.dash}
{ duplicate$ empty$
    { pop$ "" }
    { 's :=
      #1 'charptr :=
      s text.length$ #1 + 'len :=
        { charptr len <
          s charptr #1 substring$ "-" = not
          and
        }
        { charptr #1 + 'charptr := }
      while$
        { charptr len <
          s charptr #1 substring$ "-" =
          and
        }
        { charptr #1 + 'charptr := }
      while$
      s charptr global.max$ substring$
    }
  if$
}

FUNCTION {extract.before.slash}
{ duplicate$ empty$
    { pop$ "" }
    { 's :=
      #1 'charptr :=
      s text.length$ #1 + 'len :=
        { charptr len <
          s charptr #1 substring$ "/" = not
          and
        }
        { charptr #1 + 'charptr := }
      while$
      s #1 charptr #1 - substring$
    }
  if$
}

FUNCTION {extract.after.slash}
{ duplicate$ empty$
    { pop$ "" }
    { 's :=
      #1 'charptr :=
      s text.length$ #1 + 'len :=
        { charptr len <
          s charptr #1 substring$ "-" = not
          and
          s charptr #1 substring$ "/" = not
          and
        }
        { charptr #1 + 'charptr := }
      while$
        { charptr len <
          s charptr #1 substring$ "-" =
          s charptr #1 substring$ "/" =
          or
          and
        }
        { charptr #1 + 'charptr := }
      while$
      s charptr global.max$ substring$
    }
  if$
}

```

著者-出版年制必须提取出年份

```
FUNCTION {format.year}
{ year empty$ not
    { year extra.label * }
    { date empty$ not
        { date extract.before.dash extra.label * }
        { entry.is.electronic not
            { "empty year in " cite$ * warning$ }
            'skip$
          if$
          urldate empty$ not
            { "[" urldate extract.before.dash * extra.label * "]" * }
            { "" }
          if$
        }
      if$
    }
  if$
}

FUNCTION {format.periodical.year}
{ year empty$ not
    { year extract.before.slash
      "--" *
      year extract.after.slash
      duplicate$ empty$
        'pop$
        { * }
      if$
    }
    { date empty$ not
        { date extract.before.dash }
        { "empty year in " cite$ * warning$
          urldate empty$ not
            { "[" urldate extract.before.dash * "]" * }
            { "" }
          if$
        }
      if$
    }
  if$
}

```

专利和报纸都是使用日期而不是年

```
FUNCTION {format.date}
{ date empty$ not
    { type$ "patent" = type$ "newspaper" = or
        { date }
        { entrysubtype empty$ not
            { type$ "article" = entrysubtype "newspaper" = and
                { date }
                { format.year }
              if$
            }
            { format.year }
          if$
        }
      if$
    }
    { year empty$ not
        { format.year }
        { "" }
      if$
    }
  if$
}

```

更新、修改日期只用于电子资源 electronic

```
FUNCTION {format.editdate}
{ date empty$ not
    { "\allowbreak(" date * ")" * }
    { "" }
  if$
}

```

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。

```
FUNCTION {format.urldate}
{ show.urldate show.url and entry.url empty$ not and
  is.pure.electronic or
  urldate empty$ not and
    { "\allowbreak[" urldate * "]" * }
    { "" }
  if$
}

```

B.4.8 Format pages

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

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

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

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

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

```
FUNCTION {hyphenate}
{ 't :=
  ""
    { t empty$ not }
    { t #1 #1 substring$ "-" =
        { wave.dash.in.pages
            { "~" * }
            { "-" * }
          if$
            { t #1 #1 substring$ "-" = }
            { t #2 global.max$ substring$ 't := }
          while$
        }
        { t #1 #1 substring$ *
          t #2 global.max$ substring$ 't :=
        }
      if$
    }
  while$
}

```

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

```
FUNCTION {format.pages}
{ pages empty$
    { "" }
    { pages hyphenate }
  if$
}

FUNCTION {format.extracted.pages}
{ pages empty$
    { "" }
    { pages
      only.start.page
        'extract.before.dash
        'hyphenate
      if$
    }
  if$
}

```

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。

```
FUNCTION {format.journal.volume}
{ volume empty$ not
    { bold.journal.volume
        { "\textbf{" volume * "}" * }
        { volume }
      if$
    }
    { "" }
  if$
}

FUNCTION {format.journal.number}
{ number empty$ not
    { "\allowbreak (" number * ")" * }
    { "" }
  if$
}

FUNCTION {format.journal.pages}
{ pages empty$
    { "" }
    { format.extracted.pages }
  if$
}

```

连续出版物的年卷期有起止范围，需要特殊处理

```
FUNCTION {format.periodical.year.volume.number}
{ year empty$ not
    { year extract.before.slash }
    { "empty year in periodical " cite$ * warning$ }
  if$
  volume empty$ not
    { ", " * volume extract.before.dash * }
    'skip$
  if$
  number empty$ not
    { "\allowbreak (" * number extract.before.dash * ")" * }
    'skip$
  if$
  "--" *
  year extract.after.slash empty$
  volume extract.after.dash empty$ and
  number extract.after.dash empty$ and not
    { year extract.after.slash empty$ not
        { year extract.after.slash * }
        { year extract.before.slash * }
      if$
      volume empty$ not
        { ", " * volume extract.after.dash * }
        'skip$
      if$
      number empty$ not
        { "\allowbreak (" * number extract.after.dash * ")" * }
        'skip$
      if$
    }
    'skip$
  if$
}

```

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。

```
FUNCTION {check.url}
{ url empty$ not
    { url 'entry.url :=
      #1 'entry.is.electronic :=
    }
    { howpublished empty$ not
        { howpublished #1 #5 substring$ "\url{" =
            { howpublished 'entry.url :=
              #1 'entry.is.electronic :=
            }
            'skip$
          if$
        }
        { note empty$ not
            { note #1 #5 substring$ "\url{" =
                { note 'entry.url :=
                  #1 'entry.is.electronic :=
                }
                'skip$
              if$
            }
            'skip$
          if$
        }
      if$
    }
  if$
}

FUNCTION {output.url}
{ show.url is.pure.electronic or
  entry.url empty$ not and
    { new.block
      entry.url #1 #5 substring$ "\url{" =
        { entry.url }
        { "\url{" entry.url * "}" * }
      if$
      output
    }
    'skip$
  if$
}

```

需要检测 DOI 是否已经包含在 URL 中。

```
FUNCTION {check.doi}
{ doi empty$ not
    { #1 'entry.is.electronic := }
    'skip$
  if$
}

FUNCTION {is.in.url}
{ 's :=
  s empty$
    { #1 }
    { entry.url empty$
        { #0 }
        { s text.length$ 'len :=
          entry.url "l" change.case$ text.length$ 'charptr :=
            { entry.url "l" change.case$ charptr len substring$ s "l" change.case$ = not
              charptr #0 >
              and
            }
            { charptr #1 - 'charptr := }
          while$
          charptr
        }
      if$
    }
  if$
```

```
}

FUNCTION {format.doi}
{ ""
  doi empty$ not
    { "" 's :=
      doi 't :=
      #0 'numnames :=
        { t empty$ not}
        { t #1 #1 substring$ 'tmp.str :=
          tmp.str "," = tmp.str " " = or t #2 #1 substring$ empty$ or
            { t #2 #1 substring$ empty$
                { s tmp.str * 's := }
                'skip$
              if$
              s empty$ s is.in.url or
                'skip$
                { numnames #1 + 'numnames :=
                  numnames #1 >
                    { ", " * }
                    { "DOI: " * }
                  if$
                  "\doi{" s * "}" * *
                }
              if$
              "" 's :=
            }
            { s tmp.str * 's := }
          if$
          t #2 global.max$ substring$ 't :=
        }
      while$
    }
    'skip$
  if$
}

FUNCTION {output.doi}
{ doi empty$ not show.doi and
  show.english.translation entry.lang lang.zh = and not and
    { new.block
      format.doi output
    }
    'skip$
  if$
}

FUNCTION {check.electronic}
{ "" 'entry.url :=
  #0 'entry.is.electronic :=
    'check.doi
    'skip$
  if$
    'check.url
    'skip$
  if$
  medium empty$ not
    { medium "MT" = medium "DK" = or medium "CD" = or medium "OL" = or
        { #1 'entry.is.electronic := }
        'skip$
      if$
    }
    'skip$
  if$
}

FUNCTION {format.eprint}
{ archivePrefix empty$ not
    { archivePrefix }
    { eprinttype empty$ not
        { archivePrefix }
        { "" }
      if$
    }
  if$
  's :=
  s empty$ not
    { s ": \eprint{" *
      url empty$ not
        { url }
        { "https://" s "l" change.case$ * ".org/abs/" * eprint * }
      if$
      * "}{" *
      eprint * "}" *
    }
    { eprint }
  if$
}

FUNCTION {output.eprint}
{ show.preprint eprint empty$ not and
    { new.block
      format.eprint output
    }
    'skip$
  if$
}

FUNCTION {format.note}
{ note empty$ not show.note and
    { note }
    { "" }
  if$
}

FUNCTION {output.translation}
{ show.english.translation entry.lang lang.zh = and
    { translation empty$ not
        { translation }
        { "[English translation missing!]" }
      if$
      " (in Chinese)" * output
      write$
      format.doi duplicate$ empty$ not
        { newline$
          write$
        }
        'pop$
      if$
      " \\" write$
      newline$
      "(" write$
      ""
      before.all 'output.state :=
    }
    'skip$
  if$
}

```

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

```
FUNCTION {empty.misc.check}
{ author empty$ title empty$
  year empty$
  and and
  key empty$ not and
    { "all relevant fields are empty in " cite$ * warning$ }
    'skip$
  if$
}

```

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

```
FUNCTION {monograph}
{ output.bibitem
  output.translation
  author empty$ not
    { format.authors }
    { editor empty$ not
```

```
          { format.editors }
          { "empty author and editor in " cite$ * warning$
⟨*author-year⟩
            bbl.anonymous
⟨/author-year⟩
⟨*numerical⟩
            ""
⟨/numerical⟩
          }
        if$
      }
    if$
    output
    year.after.author
      { period.after.author
          'new.sentence
          'skip$
        if$
        format.year "year" output.check
      }
      'skip$
    if$
    new.block
    format.series.vol.num.title "title" output.check
    "M" set.entry.mark
    format.mark "" output.after
    new.block
    format.translators output
    new.sentence
    format.edition output
    new.block
    format.address.publisher output
    year.after.author not
      { format.year "year" output.check }
      'skip$
    if$
    format.pages bbl.pages.colon output.after
    format.urldate "" output.after
    output.url
    output.doi
    new.block
    format.note output
    fin.entry
}

```

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

```
FUNCTION {incollection}
{ output.bibitem
  output.translation
  format.authors output
  author format.key output
  year.after.author
    { period.after.author
        'new.sentence
        'skip$
      if$
      format.year "year" output.check
    }
    'skip$
  if$
  new.block
  format.title "title" output.check
  "M" set.entry.mark
  format.mark "" output.after
  new.block
  format.translators output
  new.slash
  format.editors output
  new.block
  format.series.vol.num.booktitle "booktitle" output.check
  new.block
  format.edition output
  new.block
  format.address.publisher output
  year.after.author not
    { format.year "year" output.check }
    'skip$
  if$
  format.extracted.pages bbl.pages.colon output.after
  format.urldate "" output.after
  output.url
  output.doi
  new.block
  format.note output
  fin.entry
}

```

B.5.3 连续出版物

```
FUNCTION {periodical}
{ output.bibitem
  output.translation
  format.authors output
  author format.key output
  year.after.author
    { period.after.author
        'new.sentence
        'skip$
```

```
      if$
      format.year "year" output.check
    }
    'skip$
  if$
  new.block
  format.title "title" output.check
  "J" set.entry.mark
  format.mark "" output.after
  new.block
  format.periodical.year.volume.number output
  new.block
  format.address.publisher output
  year.after.author not
    { format.periodical.year "year" output.check }
    'skip$
  if$
  format.urldate "" output.after
  output.url
  output.doi
  new.block
  format.note output
  fin.entry
}

```

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.

```
FUNCTION {journal.article}
{ output.bibitem
  output.translation
  format.authors output
  author format.key output
  year.after.author
    { period.after.author
        'new.sentence
        'skip$
      if$
      format.year "year" output.check
    }
    'skip$
  if$
  new.block
  title.in.journal
    { format.title "title" output.check
      entrysubtype empty$ not
```

```
        {
          entrysubtype "newspaper" =
            { "N" set.entry.mark }
            { "J" set.entry.mark }
          if$
        }
        { "J" set.entry.mark }
      if$
      format.mark "" output.after
      new.block
    }
    'skip$
  if$
  format.journal "journal" output.check
  year.after.author not
    { format.date "year" output.check }
    'skip$
  if$
  format.journal.volume output
  format.journal.number "" output.after
  format.journal.pages bbl.pages.colon output.after
  format.urldate "" output.after
  output.url
  output.doi
  new.block
  format.note output
  fin.entry
}

```

B.5.5 专利文献

number 域也可以用来表示专利号。

```
FUNCTION {patent}
{ output.bibitem
  output.translation
  format.authors output
  author format.key output
  year.after.author
    { period.after.author
        'new.sentence
        'skip$
      if$
      format.year "year" output.check
    }
    'skip$
  if$
  new.block
  format.title "title" output.check
  "P" set.entry.mark
  format.mark "" output.after
  new.block
  format.date "year" output.check
  format.urldate "" output.after
  output.url
  output.doi
  new.block
  format.note output
  fin.entry
}

```

B.5.6 电子资源

```
FUNCTION {electronic}
{ #1 #1 check.electronic
  #1 'entry.is.electronic :=
  #1 'is.pure.electronic :=
  output.bibitem
  output.translation
  format.authors output
  author format.key output
  year.after.author
    { period.after.author
        'new.sentence
        'skip$
      if$
      format.year "year" output.check
    }
    'skip$
  if$
  new.block
  format.series.vol.num.title "title" output.check
  "EB" set.entry.mark
  format.mark "" output.after
  new.block
  format.address.publisher output
  year.after.author not
    { date empty$
        { format.date output }
        'skip$
      if$
    }
    'skip$
  if$
  format.pages bbl.pages.colon output.after
  format.editdate "" output.after
  format.urldate "" output.after
  output.url
  output.doi
  new.block
  format.note output
  fin.entry
}

```

B.5.7 预印本

```
FUNCTION {preprint}
```

```
{ output.bibitem
  output.translation
  author empty$ not
    { format.authors }
    { editor empty$ not
        { format.editors }
        { "empty author and editor in " cite$ * warning$
⟨*author-year⟩
          bbl.anonymous
⟨/author-year⟩
⟨*numerical⟩
          ""
⟨/numerical⟩
        }
      if$
    }
  if$
  output
  year.after.author
    { period.after.author
        'new.sentence
        'skip$
      if$
      format.year "year" output.check
    }
    'skip$
  if$
  new.block
  title.in.journal
    { format.series.vol.num.title "title" output.check
⟨*2015⟩
      "A" set.entry.mark
⟨/2015⟩
⟨*!2015⟩
      "Z" set.entry.mark
⟨/!2015⟩
      format.mark "" output.after
      new.block
    }
    'skip$
  if$
  format.translators output
  new.sentence
  format.edition output
  new.block
  year.after.author not
    { date empty$
        { format.date output }
        'skip$
      if$
    }
    'skip$
  if$
  format.pages bbl.pages.colon output.after
  format.editdate "" output.after
  format.urldate "" output.after
  output.eprint
  output.url
  show.preprint not eprint empty$ or
    'output.doi
    'skip$
  if$
  new.block
  format.note output
  fin.entry
}

```

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。

```
FUNCTION {misc}
{ get.journal.title
  duplicate$ empty$ not
    { check.arxiv.preprint
        'preprint
        'journal.article
      if$
    }
    { pop$
      booktitle empty$ not
        'incollection
        { publisher empty$ not
            'monograph
            { eprint empty$ not archivePrefix empty$ not or
                'preprint
                { entry.is.electronic
                    'electronic
                    {
⟨*!2005⟩
                      "Z" set.entry.mark
⟨/!2005⟩
⟨*2005⟩
                      "M" set.entry.mark
⟨/2005⟩
                      monograph
                    }
                  if$
                }
              if$
            }
          if$
        }
      if$
    }
  if$
  empty.misc.check
}

FUNCTION {archive}
{ "A" set.entry.mark
  misc
}

FUNCTION {article} { misc }

```

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

```
FUNCTION {book} { monograph }

```

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

Required: title

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

```
FUNCTION {booklet} { book }

FUNCTION {collection}
{ "G" set.entry.mark
  monograph
}

FUNCTION {database}
{ "DB" set.entry.mark
  electronic
}

FUNCTION {dataset}
{ "DS" set.entry.mark
  electronic
}

```

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。

```
FUNCTION {inbook} {
  booktitle empty$
    'book
    'incollection
  if$
}

```

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

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

```

The conference function is included for Scribe compatibility.

```
FUNCTION {conference} { inproceedings }

FUNCTION {legislation} { archive }


FUNCTION {map}
{ "CM" set.entry.mark
  misc
}

```

A manual is technical documentation.

Required: title

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

```
FUNCTION {manual} { monograph }

```

A mastersthesis is a Master's thesis.

Required: author, title, school, year

Optional: type, address, month, note

```
FUNCTION {mastersthesis}
{ "D" set.entry.mark
  monograph
}

FUNCTION {newspaper}
{ "N" set.entry.mark
  article
}

FUNCTION {online}
{ "EB" set.entry.mark
  electronic
}

```

A phdthesis is like a mastersthesis.

Required: author, title, school, year

Optional: type, address, month, note

```
FUNCTION {phdthesis} { mastersthesis }

```

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

```
FUNCTION {proceedings}
{ "C" set.entry.mark
  monograph
}

FUNCTION {software}
{ "CP" set.entry.mark
  electronic
}

FUNCTION {standard}
{ "S" set.entry.mark
  misc
}

```

A techreport is a technical report.

Required: author, title, institution, year

Optional: type, number, address, month, note

```
FUNCTION {techreport}
{ "R" set.entry.mark
  misc
}

```

An unpublished is something that hasn't been published.

Required: author, title, note

Optional: month, year

```
FUNCTION {unpublished} { misc }

```

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

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

```

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

```
MACRO {jan} {"January"}

MACRO {feb} {"February"}

MACRO {mar} {"March"}

MACRO {apr} {"April"}

MACRO {may} {"May"}

MACRO {jun} {"June"}

MACRO {jul} {"July"}

MACRO {aug} {"August"}

MACRO {sep} {"September"}

MACRO {oct} {"October"}

MACRO {nov} {"November"}

MACRO {dec} {"December"}

```

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

```
MACRO {acmcs} {"ACM Computing Surveys"}

MACRO {acta} {"Acta Informatica"}

MACRO {cacm} {"Communications of the ACM"}

MACRO {ibmjrd} {"IBM Journal of Research and Development"}

MACRO {ibmsj} {"IBM Systems Journal"}

MACRO {ieeese} {"IEEE Transactions on Software Engineering"}

MACRO {ieeetc} {"IEEE Transactions on Computers"}

MACRO {ieeetcad}
 {"IEEE Transactions on Computer-Aided Design of Integrated Circuits"}

MACRO {ipl} {"Information Processing Letters"}

MACRO {jacm} {"Journal of the ACM"}

MACRO {jcss} {"Journal of Computer and System Sciences"}

MACRO {scp} {"Science of Computer Programming"}

MACRO {sicomp} {"SIAM Journal on Computing"}

MACRO {tocs} {"ACM Transactions on Computer Systems"}

MACRO {tods} {"ACM Transactions on Database Systems"}

MACRO {tog} {"ACM Transactions on Graphics"}

MACRO {toms} {"ACM Transactions on Mathematical Software"}

MACRO {toois} {"ACM Transactions on Office Information Systems"}

MACRO {toplas} {"ACM Transactions on Programming Languages and Systems"}

MACRO {tcs} {"Theoretical Computer Science"}

```

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.

```
FUNCTION {sortify}
{ purify$
  "l" change.case$
}

```

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

```
FUNCTION {chop.word}
{ 's :=
  'len :=
  s #1 len substring$ =
    { s len #1 + global.max$ substring$ }
    's
  if$
}

```

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

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

Exactly what fields we look at in constructing the primary part of the label depends on the entry type; this selectivity (as opposed to, say, always looking at author, then editor, then key) helps ensure that `ignored` fields, as described in the

LaTeX book, really are ignored. Note that MISC is part of the deepest 'else' clause in the nested part of calc.label; thus, any unrecognized entry type in the database is handled correctly.

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

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

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

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

```
FUNCTION {format.lab.name}
{ "{vv~}{ll}{, jj}{, ff}" format.name$ 't :=
  t "others" =
    { citation.et.al }
    { t get.str.lang 'name.lang :=
      name.lang lang.zh = name.lang lang.ja = or
        { t #1 "{ll}{ff}" format.name$ }
        { t #1 "{vv~}{ll}" format.name$ }
      if$
    }
  if$
}

FUNCTION {format.lab.names}
{ 's :=
  #1 'nameptr :=
  s num.names$ 'numnames :=
  ""
  numnames 'namesleft :=
    { namesleft #0 > }
    { s nameptr format.lab.name citation.et.al =
      numnames citation.et.al.min #1 - > nameptr citation.et.al.use.first > and or
        { bbl.space *
          citation.et.al *
          #1 'namesleft :=
        }
        { nameptr #1 >
            { namesleft #1 = citation.and "" = not and
                { citation.and * }
                { ", " * }
              if$
            }
            'skip$
          if$
          s nameptr format.lab.name *
        }
      if$
      nameptr #1 + 'nameptr :=
      namesleft #1 - 'namesleft :=
    }
  while$
}

FUNCTION {author.key.label}
{ author empty$
    { key empty$
        { cite$ #1 #3 substring$ }
        'key
      if$
    }
    { author format.lab.names }
  if$
}

FUNCTION {author.editor.key.label}
{ author empty$
    { editor empty$
        { key empty$
            { cite$ #1 #3 substring$ }
            'key
          if$
        }
        { editor format.lab.names }
      if$
    }
    { author format.lab.names }
  if$
}

FUNCTION {author.key.organization.label}
{ author empty$
    { key empty$
        { organization empty$
            { cite$ #1 #3 substring$ }
            { "The " #4 organization chop.word #3 text.prefix$ }
          if$
        }
        'key
      if$
    }
    { author format.lab.names }
  if$
}

FUNCTION {editor.key.organization.label}
{ editor empty$
    { key empty$
        { organization empty$
            { cite$ #1 #3 substring$ }
            { "The " #4 organization chop.word #3 text.prefix$ }
          if$
        }
        'key
      if$
    }
    { editor format.lab.names }
  if$
}

FUNCTION {calc.short.authors}
{ type$ "book" =
  type$ "inbook" = booktitle empty$ not and
  or
    'author.editor.key.label
    { type$ "collection" =
      type$ "proceedings" =
      or
        { editor empty$ not
            'editor.key.organization.label
            'author.key.organization.label
          if$
        }
        'author.key.label
      if$
    }
  if$
  'short.list :=
}
```

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。

```
FUNCTION {calc.label}
{ calc.short.authors
  short.list "]" contains
    { "{" short.list * "}" * }
    { short.list }
  if$
  "("
  *
  format.year duplicate$ empty$
  short.list key field.or.null = or
     { pop$ "" }
     'skip$
  if$
  duplicate$ "]" contains
    { "{" swap$ * "}" * }
    'skip$
  if$
  *
  'label :=
}

```

B.8 Sorting

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

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

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

⟨*author-year⟩

```
FUNCTION {sort.language.label}
{ entry.lang lang.zh =
    { lang.zh.order }
    { entry.lang lang.ja =
        { lang.ja.order }
        { entry.lang lang.en =
            { lang.en.order }
            { entry.lang lang.ru =
                { lang.ru.order }
                { lang.other.order }
              if$
            }
          if$
        }
      if$
    }
  if$
  #64 +
  int.to.chr$
}

FUNCTION {sort.format.names}
{ 's :=
  #1 'nameptr :=
  ""
  s num.names$ 'numnames :=
  numnames 'namesleft :=
    { namesleft #0 > }
    {
      s nameptr "{vv{ } }{ll{ }}{  ff{ }}{  jj{ }}" format.name$ 't :=
      nameptr #1 >
        {
          "   "  *
          namesleft #1 = t "others" = and
            { "zzzzz" * }
            { numnames #2 > nameptr #2 = and
                { "zz" * year field.or.null * "   " * }
                'skip$
              if$
              t sortify *
            }
          if$
        }
        { t sortify * }
      if$
      nameptr #1 + 'nameptr :=
      namesleft #1 - 'namesleft :=
    }
  while$
}
```

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

```
FUNCTION {sort.format.title}
{ 't :=
  "A " #2
    "An " #3
      "The " #4 t chop.word
    chop.word
  chop.word
  sortify
  #1 global.max$ substring$
}

```

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.

```
FUNCTION {anonymous.sort}
{ entry.lang lang.zh =
    { "yi4 ming2" }
    { "anon" }
  if$
}

FUNCTION {warn.empty.key}
{ entry.lang lang.zh =
    { "empty key in " cite$ * warning$ }
    'skip$
  if$
}

FUNCTION {author.sort}
{ key empty$
    { warn.empty.key
      author empty$
        { anonymous.sort }
        { author sort.format.names }
      if$
    }
    { key }
  if$
}

FUNCTION {author.editor.sort}
{ key empty$
    { warn.empty.key
      author empty$
        { editor empty$
            { anonymous.sort }
            { editor sort.format.names }
          if$
        }
        { author sort.format.names }
      if$
    }
    { key }
  if$
}

FUNCTION {author.organization.sort}
{ key empty$
    { warn.empty.key
      author empty$
        { organization empty$
            { anonymous.sort }
            { "The " #4 organization chop.word sortify }
          if$
        }
        { author sort.format.names }
      if$
    }
    { key }
  if$
}

FUNCTION {editor.organization.sort}
{ key empty$
    { warn.empty.key
      editor empty$
        { organization empty$
            { anonymous.sort }
            { "The " #4 organization chop.word sortify }
          if$
        }
        { editor sort.format.names }
      if$
    }
    { key }
  if$
}

⟨/author-year⟩
```

顺序编码制的排序要简单得多

```
⟨*numerical⟩
INTEGERS { seq.num }

FUNCTION {init.seq}
{ #0 'seq.num :=}

FUNCTION {int.to.fix}
{ "000000000" swap$ int.to.str$ *
  #-1 #10 substring$
}

⟨/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!

```
FUNCTION {presort}
{ set.entry.lang
  set.entry.numbered
  show.url show.doi check.electronic
  #0 'is.pure.electronic :=
  calc.label
  label sortify
  "    "
  *
⟨*author-year⟩
  sort.language.label
  "    "
  *
  type$ "book" =
  type$ "inbook" = booktitle empty$ not and
  or
    'author.editor.sort
    { type$ "collection" =
      type$ "proceedings" =
      or
        'editor.organization.sort
        'author.sort
      if$
    }
  if$
  *
  "    "
  *
  year field.or.null sortify
  *
  "    "
  *
  cite$
  *
  #1 entry.max$ substring$
⟨/author-year⟩
⟨*numerical⟩
  seq.num #1 + 'seq.num :=
  seq.num  int.to.fix
⟨/numerical⟩
  'sort.label :=
  sort.label *
  #1 entry.max$ substring$
  'sort.key$ :=
}

```

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
```

```
STRINGS { longest.label last.label next.extra last.extra.label }

INTEGERS { longest.label.width number.label }

FUNCTION {initialize.longest.label}
{ "" 'longest.label :=
  #0 int.to.chr$ 'last.label :=
  "" 'next.extra :=
  #0 'longest.label.width :=
  #0 'number.label :=
  "" 'last.extra.label :=
}

FUNCTION {forward.pass}
{
⟨*author-year⟩
```

```
  last.label label =
    { "" 'extra.label :=
      last.extra.label text.length$ 'charptr :=
        { last.extra.label charptr #1 substring$ "z" =
          charptr #0 > and
        }
        { "a" extra.label * 'extra.label :=
          charptr #1 - 'charptr :=
        }
      while$
      charptr #0 >
        { last.extra.label charptr #1 substring$ chr.to.int$ #1 + int.to.chr$
          extra.label * 'extra.label :=
          last.extra.label #1 charptr #1 - substring$
          extra.label * 'extra.label :=
        }
        { "a" extra.label * 'extra.label := }
      if$
      extra.label 'last.extra.label :=
    }
    { "a" 'last.extra.label :=
      "" 'extra.label :=
      label 'last.label :=
    }
  if$
⟨/author-year⟩
  number.label #1 + 'number.label :=
}

FUNCTION {reverse.pass}
{
⟨*author-year⟩
  next.extra "b" =
    { "a" 'extra.label := }
    'skip$
  if$
  extra.label 'next.extra :=
  extra.label
  duplicate$ empty$
    'skip$
    { "{\natexlab{" swap$ * "}}" * }
  if$
  'extra.label :=
⟨/author-year⟩
  label extra.label * 'label :=
}

FUNCTION {bib.sort.order}
{ sort.label  'sort.key$ :=
}
```

B.9 Write bbl file

Now we're ready to start writing the .BBL file. We begin, if necessary, with a LaTeX macro for unnamed names in an alphabetic label; next comes stuff from the 'preamble' command in the database files. Then we give an incantation containing the command `\begin{thebibliography}{...}` where the '...' is the longest label.

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

```
FUNCTION {begin.bib}
{   preamble$ empty$
    'skip$
    { preamble$ write$ newline$ }
  if$
  "\begin{thebibliography}{" number.label int.to.str$ * "}" *
  write$ newline$
  terms.in.macro
    { "\providecommand{\biband}{和}"
      write$ newline$
      "\providecommand{\bibetal}{等}"
      write$ newline$
    }
    'skip$
  if$
  "\providecommand{\natexlab}[1]{#1}"
  write$ newline$
  "\providecommand{\url}[1]{#1}"
  write$ newline$
  "\expandafter\ifx\csname urlstyle\endcsname\relax\else"
  write$ newline$
  "  \urlstyle{same}\fi"
  write$ newline$
  "\expandafter\ifx\csname href\endcsname\relax"
  write$ newline$
  "  \DeclareUrlCommand\doi{\urlstyle{rm}}"
  write$ newline$
  "  \def\eprint#1#2{#2}"
      write$ newline$
  "\else"
  write$ newline$
  "  \def\doi#1{\href{https://doi.org/#1}{\nolinkurl{#1}}}"
  write$ newline$
  "  \let\eprint\href"
      write$ newline$
  "\fi"
      write$ newline$
    }

```

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

```
FUNCTION {end.bib}
{ newline$
  "\end{thebibliography}" write$ newline$
}
```

B.10 Main execution

Now we read in the .BIB entries.

```
READ

EXECUTE {init.state.consts}

EXECUTE {load.config}

⟨*numerical⟩
EXECUTE {init.seq}

⟨/numerical⟩
ITERATE {presort}

```

And now we can sort

```
SORT

EXECUTE {initialize.longest.label}

ITERATE {forward.pass}

REVERSE {reverse.pass}

ITERATE {bib.sort.order}

SORT

EXECUTE {begin.bib}

```

Now we produce the output for all the entries

```
ITERATE {call.type$}

EXECUTE {end.bib}
⟨/author-year | numerical⟩
```