GB/T 7714-2015 BibTEX style

Zeping Lee*

2018/08/05 v1.0.9

摘要

The gbt7714 package provides a BibTEX implementation for the China's bibliography style standard GB/T 7714-2015. It consists of two bst files for numerical and authoryear styles as well as a LATEX package which provides the citation style defined in the standard. It is compatible with natbib and supports language detection (Chinese and English) for each 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"）是中国的参考文献推荐标准。本宏包是国标的 BibTEX[2] 实现，具有以下特性：

- 兼容 natbib 宏包[3]
- 支持顺序编码制和著者-出版年制两种风格
- 自动识别语言并进行相应处理
- 提供了简单的接口供用户修改样式

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

2 使用方法

super
numbers
authoryear

按照国标的规定，参考文献的标注体系分为"顺序编码制"和"著者-出版年制"(authoryear)，其中顺序编码制根据引用标注样式的不同分为角标数字式（super）和与正文平排的数字式（numbers）。

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

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

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

使用时需要注意以下几点：

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

\cite 在正文中引用文献时应使用 \cite 命令。同一处引用多篇文献时，应将各篇文献的 key 一同写在 \cite 命令中，如 \cite{knuth84,lamport94,mittelbach04}。如遇连续编号，可以自动转为起讫序号并用短横线连接。它可以自动排序并用处理连续编号。若需要标出引文的页码，可以标在 \cite 的可选参数中，如 \cite[42]{knuth84}。更多的引用标注方法可以参考 natbib 宏包的使用说明[3]。

\bibliography 参考文献表可以在文中使用 \bibliography 命令调用。注意文献列表的样式已经在模板中根据选项设置，用户不再需要使用 \bibliographystyle 命令。

3 文献类型

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

表 1: 全部文献类型

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

4 著录项目

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

author 主要责任者

title 题名

mark* 文献类型标识

medium* 载体类型标识

*zepinglee AT gmail.com

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

不支持的 BibTEX 标准著录项目有 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,
  language = {japanese},
  mark     = {Z},
  medium   = {DK},
  ...
```

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

5 文献列表的排序

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

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

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
period.between.author.year	`#0`	著者和年份之间使用句点连接
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
show.mark	`#1`	显示文献类型标识
italic.jounal	`#0`	西文期刊名使用斜体
show.missing.address.publisher	`#1`	出版项缺失时显示"出版者不详"
show.url	`#1`	显示 url
show.doi	`#1`	显示 doi
show.note	`#0`	显示 note 域的信息

6 自定义样式

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

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

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

7 相关工作

TeX 社区也有其他关于 GB/T 7714 系列参考文献标准的工作。2005 年吴凯[7] 发布了基于 GB/T 7714-2005 的 BibTEX 样式，支持顺序编码制和著者出版年制两种风格。李志奇[8] 发布了严格遵循 GB/T 7714-2005 的 BibLaTeX 的样式。胡海星[9] 提供了另一个 BibTEX 实现，还给每行 bst 代码写了 java 语言注释。沈周[10] 基于 biblatex-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. http://bbs.ctex.org/forum.php?mod=viewthread&tid=33591.

[8] 李志奇. 基于 biblatex 的符合 GBT7714-2005 的中文文献生成工具[EB/OL]. 2013. http://bbs.ctex.org/forum.php?mod=viewthread&tid=74474.

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

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

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

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

版本历史

A 宏包的代码实现

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

```
⟨*package⟩
\newif\if@gbt@mmxv
\newif\if@gbt@numerical
\newif\if@gbt@super
\DeclareOption{2015}{\@gbt@mmxvtrue}
\DeclareOption{2005}{\@gbt@mmxvfalse}
\DeclareOption{super}{\@gbt@numericaltrue\@gbt@supertrue}
\DeclareOption{numbers}{\@gbt@numericaltrue\@gbt@superfalse}
\DeclareOption{authoryear}{\@gbt@numericalfalse}
\DeclareOption*{\PassOptionsToPackage{\CurrentOption}{natbib}}
\ExecuteOptions{2015,super}
\ProcessOptions\relax
```

只在顺序编码时使用 `sort&compress`。

```
\if@gbt@numerical
  \PassOptionsToPackage{sort&compress}{natbib}
\fi
\RequirePackage{natbib}
\RequirePackage{etoolbox}
\RequirePackage{url}
```

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

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

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

```
\newcommand\gbtbibstyle[1]{%
  \ifstrequal{#1}{numerical}{%
    \if@gbt@mmxv
      \bibliographystyle{gbt7714-unsrt}%
    \else
      \bibliographystyle{gbt7714-2005-unsrt}%
    \fi
  }{%
    \ifstrequal{#1}{authoryear}{%
      \if@gbt@mmxv
        \bibliographystyle{gbt7714-plain}%
      \else
        \bibliographystyle{gbt7714-2005-plain}%
      \fi
    }{%
      \PackageError{gbt7714}{Unknown argument #1.}%
      {It should be `numerical' or `authoryear'.}%
    }%
  }%
}
```

处理宏包选项。

```
\if@gbt@numerical
  \if@gbt@super
    \citestyle{super}%
    \gbtbibstyle{numerical}%
  \else
    \citestyle{numbers}
    \gbtbibstyle{numerical}%
  \fi
\else
  \citestyle{authoryear}
  \gbtbibstyle{authoryear}%
\fi
```

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

```
\newcommand\gbt@patchfailure[1]{%
  \ClassError{ustcthesis}{Failed to patch command \protect#1.\MessageBreak
    Please contact the template author.%
  }{}%
}
\patchcmd{\NAT@citexnum}{%
  \@ifnum{\NAT@ctype=\z@}{%
    \if*#2*\else\NAT@cmt#2\fi
  }{}%
  \NAT@mbox{\NAT@@close}%
}{%
  \NAT@mbox{\NAT@@close}%
  \@ifnum{\NAT@ctype=\z@}{%
    \if*#2*\else\textsuperscript{#2}\fi
  }{}%
}{}{\gbt@patchfailure{\NAT@citexnum}}
```

Numerical 模式的 `\citep` 的页码：

```
\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#3\fi}%
   \else #1\fi\endgroup}
```

Author-year 模式的 `\citet` 的页码：

```
\patchcmd{\NAT@citex}{%
  \if*#2*\else\NAT@cmt#2\fi
  \if\relax\NAT@date\relax\else\NAT@@close\fi
}{%
  \if\relax\NAT@date\relax\else\NAT@@close\fi
  \if*#2*\else\textsuperscript{#2}\fi
}{}{\gbt@patchfailure{\NAT@citex}}
```

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}
```

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。

```
\patchcmd{\NAT@citexnum}{%
  \ifx\NAT@last@yr\relax
    \def@NAT@last@yr{\@citea}%
  \else
    \def@NAT@last@yr{--\NAT@penalty}%
  \fi
}{%
  \def@NAT@last@yr{-\NAT@penalty}%
}{}{\gbt@patchfailure{\NAT@citexnum}}
```

thebibliography 参考文献列表的标签左对齐

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

\url 使用 xurl 宏包的方法，增加 URL 可断行的位置。

```
\def\UrlBreaks{%
  \do\/%
  \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%
  \do0\do1\do2\do3\do4\do5\do6\do7\do8\do9\do=\do/\do.\do:%
  \do\*\do\-\do\~\do\'\do\"\do\-}
\Urlmuskip=0mu plus 0.1mu
⟨/package⟩
```

B BibTeX 样式的代码实现

B.1 自定义选项

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

```
⟨*authoryear | numerical⟩
INTEGERS {
  uppercase.name
  max.num.authors
  period.between.author.year
  sentence.case.title
  link.title
  show.mark
  italic.jounal
  show.missing.address.publisher
  show.url
  show.doi
  show.note
}

```

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}
{
```

英文姓名转为全大写：

```
  #1 'uppercase.name :=
```

最多显示的作者数量：

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

采用著者-出版年制时，作者姓名与年份之间使用句点连接：

```
  #0 'period.between.author.year :=
```

英文标题转为 sentence case （句首字母大写，其余小写）：

```
  #1 'sentence.case.title :=
```

在标题添加超链接：

```
  #0 'link.title :=
```

著录文献类型标识（比如“[M/OL]“）：

```
  #1 'show.mark :=
```

期刊名使用斜体：

```
  #0 'italic.jounal :=
```

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.”：

```
  #1 'show.missing.address.publisher :=
```

是否著录 URL：

```
  #1 'show.url :=
```

是否著录 DOI：

```
⟨*2015⟩
  #1 'show.doi :=
⟨/2015⟩
⟨*2005⟩
  #0 'show.doi :=
⟨/2005⟩
```

在每一条文献最后输出注释（note）的内容：

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

```

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
    author
    booktitle
    date
    doi
    edition
    editor
    howpublished
    institution
    journal
    key
    language
    mark
    medium
  note
  number
  organization
  pages
  publisher
  school
  series
  title
  translator
  url
  urldate
  volume
  year
}
{ entry.lang entry.is.electronic }
```

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}
{ lang.zh entry.lang =
    { " 佚名" }
    { "Anon" }
  if$
}

FUNCTION {bbl.space} { "\ " }

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

FUNCTION {bbl.colon} { ": " }

⟨*2015⟩
FUNCTION {bbl.wide.space} { "\quad " }
⟨/2015⟩
⟨*2005⟩
FUNCTION {bbl.wide.space} { "\ " }
⟨/2005⟩

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

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

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

FUNCTION {bbl.sine.loco.sine.nomine}
{ lang.zh entry.lang =
    { "[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$
}

```

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$ }
                { 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}
{ add.period$
  write$
  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$
    { after.slash 'output.state := }
  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 {italicize}
{ duplicate$ empty$
    { pop$ "" }
    { "\textit{" swap$ * "}" * }
  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
}

FUNCTION {set.entry.lang}
{ language empty$
    { check.entry.lang }
    { language "english" = language "american" = or language "british" = or
        { lang.en }
        { language "chinese" =
            { lang.zh }
            { language "japanese" =
                { lang.ja }
                { language "russian" =
                    { lang.ru }
                    { check.entry.lang }
                  if$
                }
              if$
            }
          if$
        }
      if$
    }
  if$
  'entry.lang :=
}

```

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.names}
{ 's :=
  #1 'nameptr :=
  s num.names$ 'numnames :=
  numnames 'namesleft :=
    { namesleft #0 > }
    { s nameptr "{vv~}{ll}{, jj}{, ff}" format.name$ 't :=
      nameptr max.num.authors >
        { bbl.et.al
          #1 'namesleft :=
        }
        { 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$
        }
      if$
      nameptr #1 >
        { ", " swap$ ∗ ∗ }
        'skip$
      if$
      nameptr #1 + 'nameptr :=
      namesleft #1 - 'namesleft :=
    }
  while$
}

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

FUNCTION {format.authors}
{ author empty$
⟨*authoryear⟩
    { bbl.anonymous }
⟨/authoryear⟩
⟨*numerical⟩
    { "" }
⟨/numerical⟩
    { author format.names }
  if$
}
```

```

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

FUNCTION {format.translators}
{ translator empty$
    { "" }
    { translator format.names
      lang.zh entry.lang =
        { 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" =
  or
    'author.editor.full
    { type$ "collection" =
      type$ "proceedings" =
      or
        'editor.full
        'author.full
      if$
    }
  if$
}

FUNCTION {output.bibitem}
{ newline$
  "\bibitem[" write$
  label write$
  ")" make.full.names duplicate$ short.list =
     { pop$ }
     { * }
   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{http://dx.doi.org/" doi * "}{" * swap$ * "}" * }
         'skip$
       if$
     }
   if$
 }

FUNCTION {format.title}
{ title empty$
    { "" }
    { title
      sentence.case.title
        'change.sentence.case
        '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$
    { "" }
    { lang.zh entry.lang =
        { " 第 " volume * " 卷" * }
        { "volume" volume tie.or.space.connect }
      if$
    }
  if$
}
```

```

FUNCTION {format.number}
{ number empty$
    { "" }
    { lang.zh entry.lang =
        { " 第 " number * " 册" * }
        { "number" number tie.or.space.connect }
      if$
    }
  if$
}

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

FUNCTION {format.series.vol.num.title}
{ format.volume.number 's :=
  series empty$ not
    { series
      sentence.case.title
        'change.sentence.case
        'skip$
      if$
      bbl.colon *
      s empty$ not
        { s * bbl.wide.space * }
        'skip$
      if$
      title
      sentence.case.title
        'change.sentence.case
        'skip$
      if$
      *
    }
    { title
      sentence.case.title
        'change.sentence.case
        'skip$
      if$
      s empty$ not
        { bbl.colon * s * }
        'skip$
      if$
    }
  if$
  link.title
    'add.link
    'skip$
  if$
}

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

FUNCTION {format.journal}
{ journal
  italic.jounal
    'italicize
    '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
    { medium empty$ not
        { entry.mark "/" * medium * 'entry.mark := }
        { entry.is.electronic
            { entry.mark "/OL" * 'entry.mark := }
            'skip$
          if$
        }
      if$
      "\allowbreak[" 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
        { lang.zh entry.lang =
            { edition " 版" * }
            { edition num.to.ordinal " ed." * }
          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
      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$
    }
    { 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.

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

```
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 {contains.dash}
{ duplicate$ empty$
    { pop$ #0 }
    { 's :=
        { s empty$ not
          s #1 #1 substring$ "-" = not
          and
        }
        { s #2 global.max$ substring$ 's := }
      while$
      s empty$ not
    }
  if$
}

```

著者-出版年制必须提取出年份

```
FUNCTION {format.year}
{ year empty$ not
    { year extract.before.dash }
    { date empty$ not
        { date extract.before.dash }
        { "empty year in " cite$ * warning$
          ""
        }
      if$
    }
  if$
  extra.label *
}

```

专利和报纸都是使用日期而不是年

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

```

更新、修改日期只用于电子资源 elctronic

```
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}
{ urldate empty$ not entry.is.electronic 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)
                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$ "-" =
        { "-" *
            { 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$
}

```

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.number}
{ number empty$ not
    { "\penalty0 (" number * ")" * }
    { "" }
  if$
}

FUNCTION {format.journal.pages}
{ pages empty$
    { "" }
    { ":\penalty0 " pages hyphenate * }
  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.dash }
    { "empty year in periodical " cite$ * warning$ }
  if$
  volume empty$ not
    { ", " * volume extract.before.dash * }
    'skip$
  if$
  number empty$ not
    { "\penalty0 (" * number extract.before.dash * ")" * }
    'skip$
  if$
  year contains.dash
    { "--" *
      year extract.after.dash empty$
      volume extract.after.dash empty$ and
      number extract.after.dash empty$ and not
        { year extract.after.dash empty$ not
            { year extract.after.dash * }
            { year extract.before.dash * }
          if$
          volume empty$ not
            { ", " * volume extract.after.dash * }
            'skip$
          if$
          number empty$ not
            { "\penalty0 (" * number extract.after.dash * ")" * }
            'skip$
          if$
        }
        '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{" 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 {format.url}
{ entry.url empty$ not
    { new.block entry.url }
    { "" }
  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 text.length$ 'charptr :=
            { entry.url charptr len substring$ s = not
              charptr #0 >
              and
            }
            { charptr #1 - 'charptr := }
          while$
          charptr
        }
      if$
    }
  if$
}

FUNCTION {format.doi}
{ ""
  doi empty$ not show.doi and
    { "" '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$
      's :=
      s empty$ not
        { new.block s }
        { "" }
      if$
    }
    '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.note}
{ note empty$ not show.note and
    { note }
    { "" }
  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
  author empty$ not
    { format.authors }
    { editor empty$ not
        { format.editors }
⟨*authoryear⟩
        { bbl.anonymous }
⟨/authoryear⟩
⟨*numerical⟩
        { "" }
⟨/numerical⟩
      if$
    }
  if$
  output
⟨*authoryear⟩
  period.between.author.year
    'new.sentence
    'skip$
  if$
  format.year "year" output.check
⟨/authoryear⟩
  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
⟨*numerical⟩
  format.year "year" output.check
⟨/numerical⟩
  format.pages bbl.colon output.after
  format.urldate "" output.after
  format.url output
  format.doi output
  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
  format.authors "author" output.check
  author format.key output
⟨*authoryear⟩
  period.between.author.year
    'new.sentence
    'skip$
  if$
  format.year "year" output.check
⟨/authoryear⟩
  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
⟨*numerical⟩
  format.year "year" output.check
⟨/numerical⟩
  format.pages bbl.colon output.after
  format.urldate "" output.after
  format.url output
  format.doi output
  new.block
  format.note output
  fin.entry
}

```

B.5.3 连续出版物

```
FUNCTION {periodical}
{ output.bibitem
  format.authors "author" output.check
  author format.key output
⟨*authoryear⟩
  period.between.author.year
    'new.sentence
    'skip$
  if$
  format.year "year" output.check
⟨/authoryear⟩
  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
⟨*numerical⟩
  format.date "year" output.check
⟨/numerical⟩
  format.urldate "" output.after
  format.url output
  format.doi output
  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 {article}
{ output.bibitem
  format.authors "author" output.check
  author format.key output
⟨*authoryear⟩
  period.between.author.year
    'new.sentence
    'skip$
  if$
  format.year "year" output.check
⟨/authoryear⟩
  new.block
  format.title "title" output.check
  "J" set.entry.mark
  format.mark "" output.after
  new.block
  format.journal "journal" output.check
⟨*numerical⟩
  format.date "year" output.check
⟨/numerical⟩
  volume output
  format.journal.number "" output.after
  format.journal.pages "" output.after
  format.urldate "" output.after
  format.url output
  format.doi output
  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
  format.authors output
  author format.key output
⟨*authoryear⟩
  period.between.author.year
    'new.sentence
    'skip$
  if$
  format.year "year" output.check
⟨/authoryear⟩
  new.block
  format.title
  number empty$ not
    { bbl.colon * number * }
    'skip$
  if$
  "title" output.check
  "P" set.entry.mark
  format.mark "" output.after
  new.block
  format.date "year" output.check
  format.urldate "" output.after
  format.url output
  format.doi output
  new.block
  format.note output
  fin.entry
}

```

B.5.6 电子资源

```
FUNCTION {electronic}
{ #1 #1 check.electronic
  #1 'entry.is.electronic :=
  output.bibitem
  format.authors output
  author format.key output
⟨*authoryear⟩
  period.between.author.year
    'new.sentence
    'skip$
  if$
  format.year "year" output.check
⟨/authoryear⟩
  new.block
  format.series.vol.num.title "title" output.check
  "EB" set.entry.mark
  format.mark "" output.after
  new.block
  format.address.publisher output
⟨*numerical⟩
  date empty$
    { format.date output }
    'skip$
  if$
⟨/numerical⟩
  format.pages bbl.colon output.after
  format.editdate "" output.after
  format.urldate "" output.after
  format.url output
  format.doi output
  new.block
  format.note output
  fin.entry
}

```

B.5.7 其他文献类型

A misc is something that doesn't fit elsewhere.

Required: at least one of the 'optional' fields

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

Misc 用来自动判断类型。

```
FUNCTION {misc}
{ journal empty$ not
    'article
    { booktitle empty$ not
        'incollection
        { publisher empty$ not
            'monograph
            { entry.is.electronic
                'electronic
                { "Z" set.entry.mark
                  monograph
                }
              if$
            }
          if$
        }
      if$
    }
  if$
  empty.misc.check
}

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

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

```
FUNCTION {inbook} { book }

```

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 {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}
{ "Z" set.entry.mark
  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.label”s until after sorting.

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

```
FUNCTION {format.lab.names}
{ 's :=
  s #1 "{vv~}{ll}{, jj}{, ff}" format.name$ 't :=
  t get.str.lang 'name.lang :=
  name.lang lang.en =
    { t #1 "{vv~}{ll}" format.name$}
    { t #1 "{ll}{ff}" format.name$}
  if$
  s num.names$ #1 >
    { bbl.space * bbl.et.al * }
    'skip$
  if$
}

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" =
  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 :=
}

FUNCTION {calc.label}
{ calc.short.authors
  short.list
  "("
  *
  format.year duplicate$ empty$
  short.list key field.or.null = or
     { pop$ "" }
     '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.

```
⟨*authoryear⟩
FUNCTION {sort.language.label}
{ entry.lang lang.zh =
    { "a zh " }
    { entry.lang lang.ja =
        { "b ja " }
        { entry.lang lang.en =
            { "c en " }
            { entry.lang lang.ru =
                { "d ru " }
                { "e other " }
              if$
            }
          if$
        }
      if$
    }
  if$
}

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}
{ lang.zh entry.lang =
    { "yi4 ming2" }
    { "anon" }
  if$
}

FUNCTION {author.sort}
{ key empty$
    { entry.lang lang.zh =
        { "empty key in " cite$ * warning$  }
        'skip$
      if$
      author empty$
        { anonymous.sort }
        { author sort.format.names }
      if$
    }
    { key sortify }
  if$
}

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

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

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

⟨/authoryear⟩
```

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
  show.url show.doi check.electronic
  calc.label
  label sortify
  "    "
  *
⟨*authoryear⟩
  sort.language.label
  type$ "book" =
  type$ "inbook" =
  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$
⟨/authoryear⟩
⟨*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 }

INTEGERS { longest.label.width last.extra.num number.label }

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

```

```
FUNCTION {forward.pass}
{ last.label label =
    { last.extra.num #1 + 'last.extra.num :=
      last.extra.num int.to.chr$ 'extra.label :=
    }
    { "a" chr.to.int$ 'last.extra.num :=
      "" 'extra.label :=
      label 'last.label :=
    }
  if$
  number.label #1 + 'number.label :=
}

FUNCTION {reverse.pass}
{ next.extra "b" =
    { "a" 'extra.label := }
    'skip$
  if$
  extra.label 'next.extra :=
  extra.label
  duplicate$ empty$
    'skip$
    { "{\natexlab{" swap$ * "}}" * }
  if$
  'extra.label :=
  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$
  "\providecommand{\natexlab}[1]{#1}"
  write$ newline$
  show.url show.doi or
    { "\providecommand{\url}[1]{#1}"
      write$ newline$
      "\expandafter\ifx\csname urlstyle\endcsname\relax\relax\else"
      write$ newline$
      "  \urlstyle{same}\fi"
      write$ newline$
    }
    'skip$
  if$
  show.doi
    { "\providecommand{\href}[2]{\url{#2}}"
```

```
      write$ newline$
      "\providecommand{\doi}[1]{\href{https://doi.org/#1}{#1}}"
      write$ newline$
    }
    'skip$
  if$
}

```

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}
⟨/authoryear | numerical⟩
```