BibTeX Style Files for Chemistry Journals[*]

Stephan Schenk
mail (at) schenk-stephan.de

September 17, 2008

Contents

---

[*] This file describes version 0.2.5 and has been last revised 2008/09/17.

1 Introduction

This collection of bibtex style files started with a version intended for *Chem. Eur. J.* The base version of it was created using the marvelous `makebst` program by Patrick W. Daly. Soon afterwards, a version for *J. Am. Chem. Soc.* was available, too. With changes to both files becoming more and more complex, everything was merged into a single `chembst.dtx` file to avoid maintaining several versions of the code. Using the `docstrip` utility, the different style files can now easily be generated by running `latex chembst.ins`. Currently, the following journals are supported:

- *Chem. Commun.* (`ChemCommun.bst`)
- *Inorg. Chem.* (`InorgChem.bst`), which can be used for most of the journals published by the American Chemical Society
- *J. Am. Chem. Soc.* (`JAmChemSoc.bst`)
- *Chem. Eur. J.* (`ChemEurJ.bst`), which can be used for most of the journals published by Wiley

Additionally, the following styles are also supported:

- *Curriculum vitae* (`cv.bst`), a style similar to *Chem. Eur. J.* that includes the title of an article

2 Some notes

2.1 crossref entries

The crossref feature is described in detail in the `btxdoc.dvi` BibTeX documentation. Using crossref is a way to inherit information from a parent entry. As an example consider a database containing a @book entry for citing the whole book and a @inbook entry for citing some pages from that book. Naturally, the only difference between both entries is that @inbook contains a `pages` field. All the other information is stored redundantly.

Using the crossref feature the @inbook entry can be as simple as

```
@inbook{inbook_key,
  crossref = {book_key},
  pages = {1-5},
}
```

All the other fields (author, publisher, year etc.) are inherited from the parent entry with the key `book_key`.

If you do not cite the whole book in your document, i. e. the document does not contain `\cite{book_key}`, the following will happen. If `book_key` is not cross-referenced more than once[1] everything will look perfectly like a "normal" @inbook entry. However, if `book_key` is cross-referenced more than once, the entry corresponding to the whole book will be added automatically to the bibliography and the @inbook entry will be formatted as A. Author in `\cite{book_key}`, pages 1-5.[2] If your document does contain `\cite{book_key}` you will always get the

---

[1] You can change that number by passing the argument `-min-crossrefs=number` to bibtex.

[2] In fact `\bibliographycite` is used. See section 2.3 for details.

crossref format.

This applies to entries of type @inbook, @incollection and @inproceedings.

2.2 JAmChemSoc.bst

The Journal of the American Chemical Society (ACS) now requires that for citations with more than 10 authors the list of authors is abbreviated by giving only the first author followed by *et al.* The style file `JAmChemSoc.bst` automatically takes care of that. However, ACS also requires that for those citations the full list of authors should be given as Supporting Information. The style `JAmChemSoc_all.bst` can be used for this purpose, since it does not abbreviate the list of authors.

2.3 Customization

Defaults for several commands used in the `thebibliography` environment are defined at the beginning of the `.bbl` file. To override the default settings define the command before using the `\bibliography` command.

`\url` The `\url` command is used to display urls. By default this only selects a typewriter font. However, it is strongly recommened, that you use the package `url.sty` which defines `\url` in a more sophisticated way. Due to some problems with multiline urls (spurious '%' in formatted url) you should use some reasonable new version (3.2 works fine) of `url.sty`.

`\urlprefix` Every `\url` is prefixed by `\urlprefix`. By default `\urlprefix` does nothing. This command is useful if you have to prepend every url with some text. For *Angew. Chem.*, i. e., every url must be preceded by "to be found under". This can easily be achieved by giving

```
\newcommand{\urlprefix}{to be found under }
```

`\foreignlanguage` This command is used to temporary switch the language for the title of books etc. By default this does nothing but if you load `babel.sty` then the language will be changed.

`\bibliographycite` This command is used for formatting crossref entries. It defaults to `\cite`. However, if you use `overcite.sty` you may not want superscripted citations in your bibliography. Then you can define `\bibliographycite` the following way

```
\newcommand{\bibliographycite}[1]{[\citen{#1}]}
```

`\bbl*` Inside the `.bbl` file no text is hard coded and instead the appropriate `\bbl*` command is used. The text "erratum", i. e., is produced by `\bblerratum`. Thus one can easily change all the words in the bibliography by defining the appropriate commands.

```
\newcommand{\bblerratum}{\emph{errat.}}
```

for instance will give "*errat.*". By redefining all necessary commands you can also change the language to some other language than the default english.

3 Description of important entry types

The entry types discussed in this section are those which are most commonly used in writing articles in chemistry. The types listed here refer to entries in the database (i. e. `*.bib`).

3.1 @article

This entry type is designed for an article in a journal. The entry type has been augmented with several additional fields. Other style files will simply ignore these fields. The following fields are recognized:

author The names of all authors

journal The name of the journal the article was published in

year The year the article has been published

volume The volume this article has been published in

pages The page numbers of the article

eid The electronic identifier of an article. Some journals, i. e. *J. Chem. Phys.* no longer have page numbers associated to an article but only use the EID. If an EID is present `pages` will be ignored.

numpages The number of pages of an article. This will only be used together with an EID.

germanpages When citing articles from *Angew. Chem.* one should always give both the german and the english version. If this field is present the style file assumes that this is a citation of *Angew. Chem. Int. Ed.* and automatically appends the correct citation for *Angew. Chem.* The year is always the same for both versions and the german volume is easily calculated by adding 73 to the english volume. The page range of the german version is taken from the `germanpages` field.

erratumyear, erratumvolume, erratumpages, erratumeid, erratumnumpages If any of these fields is non-empty the information will be used to append a correctly formatted citation of the erratum to the regular citation.

note A free format text that will be appended at the very end of a citation.

3.2 @book

This entry type is designed for citing a whole book. If you want to cite only some chapters or pages of a book use @inbook or @incollection (see below). The following fields are recognized:

author, editor The names of all authors or editors. You can only use one of author or editor, but not both.

title The title of the book

language The language of the title

edition The edition of the book

volume The volume of the book

series The series the book is part of

publisher The publisher of the book

address The publisher's address

year The year the book was published in

note A free format text that will be appended at the very end of the citation.

3.3 @inbook

This entry type is designed for citing chapters or pages of a book. The final format of the citation is the same as for @book except that the cited chapter(s) and/or pages are appended. For the crossref feature see section 2.1. The same fields as for @book plus the following are recognized:

chapter The cited chapters

pages The cited pages

3.4 @incollection

This entry is also designed for citing chapters or pages of a book and thus very similar to @inbook. However, there is one difference: the present entry is designed for chapter(s) written by some authors in a book edited by others. The final citation looks similar to A. Author, in book title, edited by E. Editor. The crossref feature is discussed in section 2.1. The same fields as for @inbook (except `title`) plus the following are recognized.

booktitle The name of the book

3.5 @masterthesis and @phdthesis

These entries are designed for citing a master's thesis or a PhD thesis. Both entries are very similar. The only difference is the thesis' name. The following fields are recognized:

author The author of the thesis

school The school or university where this thesis work was carried out.

address The school's address

year The year the thesis was finished in.

note A free format text that will be appended at the very end of the citation.

url An url where the thesis can be found.

3.6 @program

This entry is designed for citing a program. This is a new entry type which is not contained in the standard BIBTEX styles. It allows greater flexibility compared to the @misc entry. The following fields are recognized:

author The program's authors

title The program name

description A short description of the program

version The version/release of the program

year The year the program was published in.

publisher The publisher of the program

address The publisher's address

note A free format text that will be appended at the very end of the citation.

url An url where the program can be found.

3.7 @misc

This entry is designed for everything that does not fit into one of the other entry categories. This is for instance useful to cite websites. The following fields are recognized:

author The author of the cited work

title The title of the cited work

howpublished A free format text describing how this work has been published. This is usually empty.

year The year the cited work was published in.

note A free format text that will be appended at the very end of the citation.

url An url where the cited work can be found.

3.8 @unpublished

This entry is designed for unpublished results. The following fields are recognized:

author The author(s) of the unpublished work

title The title of the unpublished work

year The year the unpublished work was carried out.

note A free format text explaining what kind of unpublished work this is (i. e. "unpublished results").

url An url where this work can be found.

4 The implementation

Write the journal this style file is intended for to the file header.

```
%% This file is intended for use with:
⟨ChemCommun⟩%%       Chem. Commun.
⟨ChemEurJ⟩%%       Chem.-Eur. J.
⟨InorgChem⟩%%       Inorg. Chem.
⟨JAmChemSoc⟩%%       J. Am. Chem. Soc.
⟨cv⟩%%       a curriculum vitae
%%
```

4.1 Setup

ENTRY Define all the fields an entry can contain.

```
ENTRY
  {
    address
    author
    booktitle
    chapter
    collaboration
    description
    edition
    editor
    eid
    erratumeid
    erratumgermanpages
    erratumnumpages
    erratumpages
    erratumvolume
    erratumyear
    germanpages
    howpublished
    institution
    journal
    key
    language
    month
    note
    number
    numpages
    organization
    pages
    publisher
    school
    series
    title
    type
    url
    version
    volume
    year
  }
  {}
  { label }
```

`INTEGERS` Define all integer variables.

```
INTEGERS {
  before.all
  i
  j
  longest.label.width
  mid.sentence
  multiresult
  nameptr
  namesleft
  new.sentence
  number.label
  numnames
  o
  output.state
}
```

`STRINGS` Define all string variables.

```
STRINGS {
  bibinfo
  delimiter
  longest.label
  s
  t
}
```

`bbl.*` Define all the functions for the LaTeX code which returns the words used.

```
FUNCTION {bbl.and} { "\bbland{}" }
FUNCTION {bbl.chap} { "\bblchap{}" }
FUNCTION {bbl.chapter} { "\bblchapter{}" }
FUNCTION {bbl.edition} { "\bbledn{}" }
FUNCTION {bbl.editor} { "\bbled{}" }
FUNCTION {bbl.editors} { "\bbleds{}" }
FUNCTION {bbl.eidp} { "\bbleidp{}" }
FUNCTION {bbl.eidpp} { "\bbleidpp{}" }
FUNCTION {bbl.erratum} { "\bblerratum{}" }
FUNCTION {bbl.etal} { "\bbletal{}" }
FUNCTION {bbl.fifth} { "\bblfiftho{}" }
FUNCTION {bbl.first} { "\bblfirsto{}" }
FUNCTION {bbl.fourth} { "\bblfourtho{}" }
FUNCTION {bbl.in} { "\bblin{}" }
FUNCTION {bbl.mthesis} { "\bblmthesis{}" }
FUNCTION {bbl.nd} { "\bblnd{}" }
FUNCTION {bbl.nr} { "\bblno{}" }
FUNCTION {bbl.number} { "\bblno{}" }
FUNCTION {bbl.of} { "\bblof{}" }
FUNCTION {bbl.page}{ "\bblp{}" }
FUNCTION {bbl.pages} { "\bblpp{}" }
FUNCTION {bbl.phdthesis} { "\bblphdthesis{}" }
FUNCTION {bbl.rd} { "\bblrd{}" }
FUNCTION {bbl.second} { "\bblsecondo{}" }
FUNCTION {bbl.st} { "\bblst{}" }
FUNCTION {bbl.techrep} { "\bbltechrep{}" }
FUNCTION {bbl.th} { "\bblth{}" }
FUNCTION {bbl.third} { "\bblthirdo{}" }
FUNCTION {bbl.volume} { "\bblvol{}" }
MACRO {jan} {"\bbljan{}"}
MACRO {feb} {"\bblfeb{}"}
MACRO {mar} {"\bblmar{}"}
MACRO {apr} {"\bblapr{}"}
MACRO {may} {"\bblmay{}"}
MACRO {jun} {"\bbljun{}"}
MACRO {jul} {"\bbljul{}"}
MACRO {aug} {"\bblaug{}"}
MACRO {sep} {"\bblsep{}"}
MACRO {oct} {"\bbloct{}"}
MACRO {nov} {"\bblnov{}"}
MACRO {dec} {"\bbldec{}"}
```

`delimiter.*` Define some delimiters used to separate different parts of a citation.

```
FUNCTION {delimiter.blank} { " " }
FUNCTION {delimiter.colon} { ": " }
FUNCTION {delimiter.comma} { ", " }
FUNCTION {delimiter.semicolon} { "; " }
FUNCTION {delimiter.default}
{
⟨ChemCommun | ChemEurJ | cv⟩  delimiter.comma
⟨JAmChemSoc | InorgChem⟩  delimiter.semicolon
}
```

4.2 Output related functions

`output.bibitem` Write `\bibitem` and setup new citation.

```
FUNCTION {output.bibitem}
{
  newline$
  "\bibitem{" write$
  cite$ write$
  "}" write$
  newline$
  ""
  before.all 'output.state :=
}
```

`output.internal` This function is finally called by all the other output functions. It first pops the delimiter from the stack. If the (now) top string is non-empty the function appends the delimiter to the (top-1) string and writes it. The old top string is pushed back on the stack at the very end thus leaving it untouched.

```
FUNCTION {output.internal}
{
  'delimiter :=
```

write only if top string is non-empty

```
  duplicate$ empty$
    'pop$
    {
```

backup top string

```
      's :=
      output.state mid.sentence =
        {
          delimiter *
          write$
        }
        {
          output.state before.all =
            'write$
            { add.period$ " " * write$ }
          if$
          mid.sentence 'output.state :=
        }
      if$
      s
    }
  if$
}
```

`output.check.internal`

The function first pops the delimiter from the stack. If the (now) top string is empty it issues a warning, otherwise it calls `output.internal` for writing.

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

`output`
`output.blank`
`output.comma`
`output.semicolon`

These functions just push the appropriate delimiter on the stack and then call `output.internal`.

```
FUNCTION {output} { delimiter.default output.internal }
FUNCTION {output.blank} { delimiter.blank output.internal }
FUNCTION {output.comma} { delimiter.comma output.internal }
FUNCTION {output.semicolon} { delimiter.semicolon output.internal }
```

`output.check`
`output.check.blank`
`output.check.comma`
`output.check.semicolon`

These functions just push the appropriate delimiter and then call `output.check.internal`.

```
FUNCTION {output.check} { delimiter.default output.check.internal }
FUNCTION {output.check.blank} { delimiter.blank output.check.internal }
FUNCTION {output.check.comma} { delimiter.comma output.check.internal }
FUNCTION {output.check.semicolon} { delimiter.semicolon output.check.internal }
```

4.3 Operators

`not`

Define a logical `not`.

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

`and` Define a logical `and`.

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

`or` Define a logical `or`.

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

`multiply` Define a function for multiplying two integers.

```
FUNCTION {multiply}
{
```

`i` is the multiplicator and will be used as a counter

```
  'i :=
```

`j` is the value to multiplicate by i, thus will be added i times to itself

```
  'j :=
  #0
  j #0 =
```

if `j==0`, nothing has to be done since the product will always be 0; `i==0` will be handled gracefully by while loop below

```
    'skip$
    {
```

now add `j` `i` times to the 0 on the stack

```
        { i }
        {
          j +
          i #1 - 'i :=
        }
      while$
    }
  if$
}
```

4.4 Small helper functions

`bibinfo.check` This function checks whether the top value is a missing field. If so it replaces the top value by an empty string, otherwise the top value is left unchanged.

```
FUNCTION {bibinfo.check}
{
  duplicate$ missing$
```

```
    { pop$ "" }
    'skip$
  if$
}
```

`bibinfo.warn`

This functions first checks whether a field is empty. If so it issues a warning. The topmost value is a description of the current field. If the field is missing, `bibinfo.warn` pushes an empty string, otherwise the value is left unchanged. The behaviour is very similar to `bibinfo.check`.

```
FUNCTION {bibinfo.warn}
{
  swap$
  duplicate$ missing$
    {
      swap$ "missing " swap$ * " in " * cite$ * warning$ pop$
      ""
    }
    { duplicate$ empty$
        {
          swap$ "empty " swap$ * " in " * cite$ * warning$
        }
        { swap$
          pop$
        }
      if$
    }
  if$
}
```

`bolden`

This function returns the LaTeX code for boldening the top string.

```
FUNCTION {bolden}
{
  duplicate$ empty$
    { pop$ "" }
    { "\textbf{" swap$ * "}" * }
  if$
}
```

`capitalize`

This function capitalizes the first letter of a word.

```
FUNCTION {capitalize}
{
  "\capitalize" swap$ *
}
```

`cat.internal`

This function catenates two strings using the delimiter on top of the stack. The second string is at (top-1) position, the first at (top-2). If any of both strings is empty the function just returns the other string without any delimiter.

```
FUNCTION {cat.internal}
{
  'delimiter :=
  duplicate$ empty$
    'pop$
    {
      swap$
      duplicate$ empty$
        'skip$
        { delimiter * }
      if$
      swap$
      *
    }
  if$
}
```

`cat.blank`
`cat.colon`
`cat.comma`
`cat.default`
`cat.semicolon`

These functions just push the appropriate delimiter and call `cat.internal`.

```
FUNCTION {cat.blank} { delimiter.blank cat.internal }
FUNCTION {cat.colon} { delimiter.colon cat.internal }
FUNCTION {cat.comma} { delimiter.comma cat.internal }
FUNCTION {cat.default} { delimiter.default cat.internal }
FUNCTION {cat.semicolon} { delimiter.semicolon cat.internal }
```

`eng.ord`

This function formats an english ordinal by appending the appropriate string.

```
FUNCTION {eng.ord}
{
  duplicate$ "1" swap$ *
  #-2 #1 substring$ "1" =
     { bbl.th * }
     { duplicate$ #-1 #1 substring$
       duplicate$ "1" =
         { pop$ bbl.st * }
         { duplicate$ "2" =
             { pop$ bbl.nd * }
             { "3" =
                 { bbl.rd * }
                 { bbl.th * }
               if$
             }
           if$
          }
       if$
     }
   if$
}
```

`is.num`

This function checks whether the top character is a number.

```
FUNCTION {is.num}
{
  chr.to.int$
  duplicate$ "0" chr.to.int$ < not
  swap$ "9" chr.to.int$ > not and
}
```

`extract.num`

This function extracts a number (as string) from the top string.

```
FUNCTION {extract.num}
{
  duplicate$ 't :=
  "" 's :=
    { t empty$ not }
    {
      t #1 #1 substring$
      t #2 global.max$ substring$ 't :=
      duplicate$ is.num
        { s swap$ * 's := }
        { pop$ "" 't := }
      if$
    }
  while$
  s empty$
    'skip$
    { pop$ s }
  if$
}
```

`convert.edition` This functions converts the `edition` field into the appropriate ordinal.

```
FUNCTION {convert.edition}
{
  extract.num "l" change.case$ 's :=
  s "first" = s "1" = or
    { bbl.first 't := }
    { s "second" = s "2" = or
        { bbl.second 't := }
        { s "third" = s "3" = or
            { bbl.third 't := }
            { s "fourth" = s "4" = or
                { bbl.fourth 't := }
                { s "fifth" = s "5" = or
                    { bbl.fifth 't := }
                    { s #1 #1 substring$ is.num
                        { s eng.ord 't := }
                        { edition 't := }
                      if$
                    }
                  if$
                }
              if$
            }
          if$
        }
      if$
    }
  if$
  t
}
```

`either.or.check` This function checks whether mutually exclusive fields are present at the same time.

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

emphasize This function returns the LaTeX code for emphasizing the top string.

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

fin.entry This function finalizes a citation. It appends a dot and writes the last text chunk.

```
FUNCTION {fin.entry}
{
  add.period$
  write$ newline$
}
```

format.names This function formats a list of names.

```
FUNCTION {format.names}
{
% bibinfo is the description of the names, i.e. author, editor
  'bibinfo :=
  duplicate$ empty$
    'skip$
    {
```

`s` is the full list of names

```
      's :=
```

`t` is the formatted name

```
      "" 't :=
```

`nameptr` is the index of the current name
`numnames` is the total number of names
`namesleft` is the number of names yet to format

```
      #1 'nameptr :=
      s num.names$ 'numnames :=
      numnames 'namesleft :=
        { namesleft #0 > }
        {
          s nameptr
⟨ChemCommun | ChemEurJ | cv⟩            "{f.~}{vv~}{ll}{, jj}"
⟨JAmChemSoc | InorgChem⟩            "{vv~}{ll}{, f.}{, jj}"
          format.name$
          bibinfo.check
          't :=
          nameptr #1 >
            {
```

The following code chunk checks whether the list of names should be abbreviated by et al.

```
                numnames #0
⟨tennames⟩                #10 +
⟨fifteennames⟩          #15 +
       >
```

If no appropriate `docstrip` option is given, the next expression will always be false (`nameptr` is always > 1, see above). Thus the check is effectively disabled.

```
       nameptr #0
⟨namesone⟩          #2 +
       =
       and
        {
```

list of names should be truncated, set formatted name to “others”

```
         "others" 't :=
         #1 'namesleft :=
        }
        'skip$
       if$
       delimiter.default *
       namesleft #1 >
        { t * }
        {
```

check whether current name is “others”, if so set formatted name to “others”

```
         s nameptr "{ll}" format.name$
         duplicate$ "others" =
          { 't := }
          { pop$ }
         if$
```

check whether formatted name is “others”, if so print et al.

```
         t "others" =
          { bbl.etal * }
          { t * }
         if$
        }
       if$
      }
```

this is the first name

```
      't
     if$
     nameptr #1 + 'nameptr :=
     namesleft #1 - 'namesleft :=
    }
   while$
  }
 if$
}
```

`get.bbl.editor` Return “editors” if `editor` contains more than one name, “editor” otherwise.

```
FUNCTION {get.bbl.editor}
{
 editor num.names$ #1 >
  'bbl.editors
  'bbl.editor
 if$
}
```

`get.bbl.erratum` Return the formatted string for “erratum”.

```
FUNCTION {get.bbl.erratum}
{
  bbl.erratum
  ":" *
}
```

`multi.page.check` Check whether the top string is only a single page or a range of pages.

```
FUNCTION {multi.page.check}
{
  't :=
  #0 'multiresult :=
    {
      multiresult not
      t empty$ not and
    }
    { t #1 #1 substring$
      duplicate$ "-" =
          swap$ duplicate$ "," =
          swap$ "+" =
          or or
        { #1 'multiresult := }
        { t #2 global.max$ substring$ 't := }
      if$
    }
  while$
  multiresult
}
```

`get.bbl.page` Return a formatted prefix for pages.

```
FUNCTION {get.bbl.page}
{
  duplicate$ multi.page.check
    { bbl.pages }
    { bbl.page  }
  if$
⟨ChemCommun | ChemEurJ | cv⟩   "." *
}
```

`n.dashify` Replace a single “-” in a range of pages by “--”.

```
FUNCTION {n.dashify}
{
  't :=
  ""
    { t empty$ not }
    {
      t #1 #1 substring$ "-" =
        {
          t #1 #2 substring$ "--" = not
            {
              "--" *
              t #2 global.max$ substring$ 't :=
            }
            {
```

there is more than one '-', therefore append all of them to the string on the stack

```
                { t #1 #1 substring$ "-" = }
                {
                  "-" *
                  t #2 global.max$ substring$ 't :=
                }
              while$
            }
          if$
        }
        {
```

the current char is not '-', therefore just append it

```
          t #1 #1 substring$ *
          t #2 global.max$ substring$ 't :=
        }
      if$
    }
  while$
}
```

`select.language` Return LaTeX code for changing the language for the string on top of the stack.

```
FUNCTION {select.language}
{
  duplicate$ empty$
    'skip$
    {
      language empty$
        'skip$
        { "\foreignlanguage{" language * "}{" * swap$ * "}" * }
      if$
    }
  if$
}
```

`space.word` This function puts spaces around a word.

```
FUNCTION {space.word}
{
  " " swap$ * " " *
}
```

`str.to.int.warn` Print a warning if top string is not a representation of a valid integer (used by `str.to.int`).

```
FUNCTION {str.to.int.warn}
{
  "str.to.int: '" swap$ * "' is not a valid integer" * warning$
}
```

`str.to.int` This function converts a string into an integer. A warning is issued if the string is not a valid representation of an integer.

```
FUNCTION {str.to.int}
{
  duplicate$ empty$
    {
      str.to.int.warn
      #0
    }
    {
```

assign the original string to `t` for parsing `t` from the end

```
      duplicate$ 't :=
```

check for sign

```
      t #1 #1 substring$ "-" =
        {
```

be sure that “-” is followed by at least one more character

```
          t #2 global.max$ substring$ 't :=
          t empty$
            {
              duplicate$ str.to.int.warn
              #0
            }
            { #-1 }
          if$
        }
        { #1 }
      if$
```

the top stack position contains now -1 or 1 depending on sign
`o` stores the offset for position inside the number

```
      #1 'o :=
```

push starting value on stack

```
      #0
        { t empty$ not}
        {
```

get last character

```
          t #-1 #1 substring$
          duplicate$ is.num
            {
```

character is in range [0–9], now multiply by offset and add to value already on stack

```
              chr.to.int$ #48 -
              o multiply
              +
```

remove last character from string, increment offset `o`

```
              t #-2 global.max$ substring$ 't :=
              o #10 multiply 'o :=
            }
            {
```

the last character was not a digit, therefore pop last character and sum

```
              pop$ pop$
```

swap sign and original string, duplicate string, print warning

```
              swap$ duplicate$ str.to.int.warn
```

swap original string and sign, push 0

```
              swap$ #0
```

break `while` loop

```
              "" 't :=
            }
          if$
        }
      while$
```

stack holds value and sign, multiply combines them

```
      multiply
```

pop copy of original string

```
      swap$ pop$
    }
  if$
}
```

`tie.or.space.prefix` This function prepends a string with either a space or a “~” depending on the length of the string.

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

`word.in` This function returns the word “in” followed by a space.

```
FUNCTION {word.in}
{
  bbl.in delimiter.blank *
}
```

4.5 Field formatting functions

`format.year.internal` This function applies all necessary formatting for a year on the stack.

```
FUNCTION {format.year.internal}
{
⟨ChemEurJ | cv⟩   bolden
}
```

`format.volume.internal` This function applies all necessary formating for a volume on the stack.

```
FUNCTION {format.volume.internal}
{
  duplicate$ empty$
    'skip$
    {
      bbl.volume
      swap$
      tie.or.space.prefix
```

stack (top-down): volume, tie/space, bbl.volume

```
      * *
    }
```

```
  if$
}
```

format.authors This function formats the list of authors. If a `collaboration` field is present its contents is printed and the list of authors is appended in parentheses.

```
FUNCTION {format.authors}
{
  author "author" format.names
  duplicate$ empty$
    'skip$
    {
      collaboration bibinfo.check
      duplicate$ empty$
        'skip$
        { " (" * swap$ * ")" * }
      if$
      *
    }
  if$
}
```

format.booktitle This function formats the title of a book and switches to an optional foreign language for this title.

```
FUNCTION {format.booktitle}
{
  booktitle bibinfo.check
  emphasize
  select.language
}
```

format.chapter This function formats a chapter. If a `type` field is present its contents is used as a prefix. If it is absent then the default (return value of `bbl.chapter`) will be used.

```
FUNCTION {format.chapter}
{
  chapter bibinfo.check
  duplicate$ empty$
    'skip$
    {
      type bibinfo.check
      duplicate$ empty$
        { pop$ bbl.chapter }
        { "l" change.case$ }
      if$
⟨JAmChemSoc | InorgChem⟩        capitalize
      swap$
```

stack (top-down): chapter, type/bbl.chapter

```
      tie.or.space.prefix
      * *
    }
  if$
}
```

format.date This function formats the date of a citation. All currently supported styles only use the year. It is pushed onto the stack and `format.year.internal` is called to format it.

```
FUNCTION {format.date}
{
  year bibinfo.check
  format.year.internal
}
```

format.edition This function formats the edition. It is converted into the appropriate english ordinal and the return value of `bbl.edition` will be appended.

```
FUNCTION {format.edition}
{
  edition bibinfo.check
  duplicate$ empty$
    'skip$
    {
      convert.edition
      output.state mid.sentence =
        { "l" }
        { "t" }
      if$ change.case$
      " " * bbl.edition *
⟨ChemCommun | ChemEurJ | cv⟩        emphasize
    }
  if$
}
```

format.editors This function formats the list of editors.

```
FUNCTION {format.editors}
{
  editor "editor" format.names
  duplicate$ empty$
    'skip$
    {
⟨JAmChemSoc | InorgChem⟩        author empty$
⟨JAmChemSoc | InorgChem⟩          'skip$
⟨JAmChemSoc | InorgChem⟩          {
⟨JAmChemSoc | InorgChem⟩        "," *
      " " *
      get.bbl.editor
      capitalize
⟨ChemCommun | ChemEurJ | cv⟩    "(" swap$ * ")" *
      *
⟨JAmChemSoc | InorgChem⟩          }
⟨JAmChemSoc | InorgChem⟩        if$
    }
  if$
}
```

format.in.booktitle This function formats a booktitle an prepends it with “in”.

```
FUNCTION {format.in.booktitle}
{
  format.booktitle
  duplicate$ empty$
    'skip$
    {
      word.in
⟨JAmChemSoc | InorgChem⟩      capitalize
      swap$ *
    }
  if$
}
```

`format.note` This function formats the `note` field of a citation. Since this is a free format field is content is used unchanged.

```
FUNCTION {format.note}
{
  note bibinfo.check
}
```

`format.number.series` This function formats number and series if the `volume` field is empty.

```
FUNCTION {format.number.series}
{
  volume bibinfo.check
  duplicate$ empty$
    {
      number empty$
        { series bibinfo.check }
        {
          series empty$
            { number bibinfo.check }
            {
              output.state mid.sentence =
                { bbl.number }
                { bbl.number capitalize }
              if$
              number bibinfo.check tie.or.space.prefix * *
              word.in *
              series bibinfo.check *
            }
          if$
        }
      if$
    }
    'skip$
  if$
}
```

`format.org.or.pub` This function formats an organization or publisher (on the stack) and its address.

```
FUNCTION {format.org.or.pub}
{
  't :=
  address empty$ t empty$ and
    { "" }
    {
      t
      address bibinfo.check
      duplicate$ empty$
        'pop$
        {
⟨ChemCommun | ChemEurJ | cv⟩            cat.comma
⟨JAmChemSoc | InorgChem⟩            cat.colon
        }
      if$
    }
  if$
}
```

`format.organization.address` This function formats an organization and its address. It pushes the organization onto the stack and calls `format.org.or.pub`.

```
FUNCTION {format.organization.address}
{
  organization bibinfo.check
  format.org.or.pub
}
```

`format.pages` This function formats a list of pages. The list is prepended with `bbl.page(s)`.

```
FUNCTION {format.pages}
{
  pages bibinfo.check
  duplicate$ empty$
    'skip$
    {
      n.dashify
      get.bbl.page
```

stack top-down: pages prefix, pages

```
      swap$
      tie.or.space.prefix *
      *
    }
  if$
}
```

`format.publisher.address` This function formats a publisher and its address. It pushes the publisher onto the stack and calls `format.org.or.pub`.

```
FUNCTION {format.publisher.address}
{
  publisher "publisher" bibinfo.warn
  format.org.or.pub
}
```

`format.thesis.type` This function formats the type of a thesis.

```
FUNCTION {format.thesis.type}
{
  type
  duplicate$ empty$
    'pop$
    {
      swap$ pop$
      "t" change.case$ bibinfo.check
    }
  if$
}
```

`format.title` This function formats a title. It also switches the language temporarily.

```
FUNCTION {format.title}
{
  title bibinfo.check
  duplicate$ empty$
    'skip$
    {
      emphasize
      select.language
    }
  if$
}
```

`format.tr.number` This function formats the number of a technical report.

```
FUNCTION {format.tr.number}
{
  number bibinfo.check
  type
  duplicate$ empty$
    { pop$ bbl.techrep }
    'skip$
  if$
  bibinfo.check
  swap$
  duplicate$ empty$
    { pop$ "t" change.case$ }
    { tie.or.space.prefix * * }
  if$
}
```

`format.url` This function formats an url. Each url is prefixed with `\urlprefix`. See section 2.3 for more information.

```
FUNCTION {format.url}
{
  url bibinfo.check
  duplicate$ empty$
    'skip$
    {
      "\urlprefix\url{" swap$ * "}" *
      new.sentence 'output.state :=
    }
  if$
}
```

`format.volume` This function formats the `volume` field. It merely pushes the field's value onto the stack and calls `format.volume.internal` to format it.

```
FUNCTION {format.volume}
{
  volume bibinfo.check
  format.volume.internal
}
```

`format.volume.and.series` This function will format the `volume` and `series`, but only if *both* of them are not empty. Otherwise and empty string is returned.

```
FUNCTION {format.volume.and.series}
{
  volume empty$ series empty$ or
    { "" }
    {
      volume format.volume.internal
```

prepend "of"

```
      swap$ bbl.of space.word * swap$
      emphasize
      *
      "volume and number" number either.or.check
    }
  if$
}
```

`format.volume.noseries` This function formats the volume. If the `series` field is non-empty an empty string is returned.

```
FUNCTION {format.volume.noseries}
{
  series empty$
    {
      volume bibinfo.check
      format.volume.internal
⟨ChemCommun | ChemEurJ | cv⟩        emphasize
    }
    { "" }
  if$
}
```

4.6 Functions related to cross-referenced entries

`bibliography.cite` This function returns the string for citing the document the `crossref` field points to.

```
FUNCTION {bibliography.cite}
{
  "\bibliographycite{" swap$ * "}" *
}
```

`format.crossref` This function formats the cross-reference.

```
FUNCTION {format.crossref}
{
  bbl.in " " *
⟨JAmChemSoc | InorgChem⟩   capitalize
  crossref bibliography.cite *
}
```

4.7 @article related funtions

`rmat.article.cat.journal.year` This function catenates the journal name and the year. It is used when formatting an erratum and/or articles in *Angew. Chem.*

```
FUNCTION {format.article.cat.journal.year}
{
⟨ChemCommun⟩  cat.comma
⟨ChemEurJ | cv | JAmChemSoc | InorgChem⟩  cat.blank
}
```

`format.article.year.internal` This function provides all formatting required for an article which is not already done in `format.date.internal`.

```
FUNCTION {format.article.year.internal}
{
⟨JAmChemSoc | InorgChem⟩  bolden
}
```

`ormat.article.volume.internal` This function applies all necessary formatting to the volume an article is published in.

```
FUNCTION {format.article.volume.internal}
{
⟨ChemCommun⟩  bolden
⟨ChemEurJ | cv | JAmChemSoc | InorgChem⟩  emphasize
}
```

`e.germanpages.volume.internal` This function is used while formatting citations of *Angew. Chem. Int. Ed.* It calculates the volume of the german edition from the volume of the english one (on the stack).

```
FUNCTION {format.article.germanpages.volume.internal}
{
  duplicate$ empty$
    'skip$
    {
      str.to.int
      duplicate$ #1 <
        {
          pop$
          "volume in " cite$ * " is not a positive integer value" * warning$
          ""
        }
        {
```

vol. 1 of the english edition corresponds to vol. 74 of the german one

```
          #73 +
          int.to.str$
          format.article.volume.internal
        }
      if$
    }
  if$
}
```

`le.germanpages.pages.internal` This function applies all necessary formatting to pages of *Angew. Chem.*, the german edition of *Angew. Chem. Engl. Ed.*

```
FUNCTION {format.article.germanpages.pages.internal}
{
  n.dashify
}
```

`format.article.date` This function formats the date of an article.

```
FUNCTION {format.article.date}
{
  format.date
  format.article.year.internal
}
```

`format.article.numpages` This function formats the number of pages of an article. The number of pages (`numpages` field) is only used in conjunction with an EID. The number of pages is assumed to be already on the stack.

```
FUNCTION {format.article.numpages}
{
  duplicate$ empty$
    'skip$
    {
      duplicate$ "1" =
        { "~" * bbl.eidp * }
        { "~" * bbl.eidpp * }
      if$
      "(" swap$ * ")" *
    }
  if$
}
```

`format.article.eid` This function formats the EID of an article.

```
FUNCTION {format.article.eid}
{
  eid bibinfo.check
  duplicate$ empty$
    'pop$
    {
      cat.comma
⟨cv⟩      numpages bibinfo.check
⟨cv⟩      format.article.numpages
⟨cv⟩      cat.blank
    }
  if$
}
```

`format.article.journal` This function formats the journal an article was published in.

```
FUNCTION {format.article.journal}
{
  journal bibinfo.check
  duplicate$ empty$
    'skip$
    { emphasize }
  if$
}
```

`ticle.germanpages.journalname` This function formats *Angew. Chem.* as the name of the german version of *Angew. Chem. Int. Ed.*

```
FUNCTION {format.article.germanpages.journalname}
{
  "Angew.\ Chem."
  emphasize
}
```

`cle.erratum.germanpages.pages` This function is used for an erratum in *Angew. Chem. Int. Ed.* It formats the pages of the german version.

```
FUNCTION {format.article.erratum.germanpages.pages}
{
  erratumgermanpages bibinfo.check
  format.article.germanpages.pages.internal
}
```

`format.article.erratum.year` This function formats the year an erratum was published.

```
FUNCTION {format.article.erratum.year}
{
  erratumyear bibinfo.check
  format.year.internal
  format.article.year.internal
}
```

`le.erratum.germanpages.volume` This function is used for an erratum in *Angew. Chem. Int. Ed.* It formats the erratum's volume in the german version.

```
FUNCTION {format.article.erratum.germanpages.volume}
{
  erratumvolume "erratumvolume" bibinfo.warn
  format.article.germanpages.volume.internal
}
```

`t.article.erratum.germanpages` This function formats the german version of an erratum published in *Angew. Chem. Int. Ed.*

```
FUNCTION {format.article.erratum.germanpages}
{
  erratumgermanpages empty$
    { "" }
    {
      format.article.germanpages.journalname
      format.article.erratum.year format.article.cat.journal.year
      format.article.erratum.germanpages.volume cat.comma
      format.article.erratum.germanpages.pages cat.comma
    }
  if$
}
```

`ormat.article.erratum.journal` This function formats the journal of an erratum. Normally this is the same journal the original article was published in.

```
FUNCTION {format.article.erratum.journal} { format.article.journal }
```

`format.article.erratum.pages` This function formats the pages of an erratum.

```
FUNCTION {format.article.erratum.pages}
```

```
{
  eratumpages bibinfo.check
  n.dashify
}
```

`format.article.erratum.eid` This function formats the EID of an erratum.

```
FUNCTION {format.article.erratum.eid}
{
  erratumeid bibinfo.check
  erratumnumpages bibinfo.check format.article.numpages *
}
```

`format.article.erratum.volume` This function formats the volume of an erratum.

```
FUNCTION {format.article.erratum.volume}
{
  erratumvolume bibinfo.check
  format.article.volume.internal
}
```

`format.article.erratum` This function formats an erratum of an article. The erratum will be formatted if one of `erratumyear`, `erratumvolume`, `erratumpages` or `erratumeid` is present.

```
FUNCTION {format.article.erratum}
{
  erratumyear bibinfo.check empty$
      erratumvolume bibinfo.check empty$ and
      erratumpages bibinfo.check empty$ and
      erratumeid bibinfo.check empty$ and
    { "" }
    {
```

An erratum was detected. Since at least one field is present, “erratum:” can already be printed.

```
      get.bbl.erratum
      format.article.erratum.journal cat.blank
      format.article.erratum.year format.article.cat.journal.year
      format.article.erratum.volume cat.comma
      erratumeid empty$
        { format.article.erratum.pages }
        { format.article.erratum.eid }
      if$
      cat.comma
      format.article.erratum.germanpages cat.semicolon
    }
  if$
}
```

`mat.article.germanpages.pages` This function formats the pages of the german version of an article in *Angew. Chem. Int. Ed.*

```
FUNCTION {format.article.germanpages.pages}
{
  germanpages bibinfo.check
  format.article.germanpages.pages.internal
}
```

`at.article.germanpages.volume` This function formats the volume of the german version of an article in *Angew. Chem. Int. Ed.* It pushes the volume of the english version onto the stack and calls `format.article.germanpages.volume.internal` which automatically calculates and formats the german volume.

```
FUNCTION {format.article.germanpages.volume}
{
  volume "volume" bibinfo.warn
  format.article.germanpages.volume.internal
}
```

`rmat.article.germanpages.year` This function formats the year of the german version of an article in *Angew. Chem. Int. Ed.* This is always the same as the english version.

```
FUNCTION {format.article.germanpages.year} { format.article.date }
```

`format.article.germanpages` This function formats the german version of an article in *Angew. Chem. Int. Ed.*

```
FUNCTION {format.article.germanpages}
{
  germanpages empty$
    { "" }
    {
      format.article.germanpages.journalname
      format.article.germanpages.year format.article.cat.journal.year
      format.article.germanpages.volume cat.comma
      format.article.germanpages.pages cat.comma
    }
  if$
}
```

`format.article.pages` This function formats the pages of an article.

```
FUNCTION {format.article.pages}
{
```

The stack contains whatever is the text directly preceding pages.

```
  pages
  duplicate$ empty$
    'pop$
    {
```

Check whether preceding string is empty.

```
      swap$
      duplicate$ empty$
```

If the preceding string is empty, pop it and the pages from stack and call `format.pages`. This means that `pages` is the only text so far (except bibitem).

```
        { pop$ pop$ format.pages }
```

The preceding string is not empty. Therefore format the pages and append them.

```
        {
          swap$
          n.dashify
          cat.comma
        }
      if$
    }
  if$
}
```

`format.article.title` This function formats the title of an article.

```
FUNCTION {format.article.title}
{
  title bibinfo.check
  duplicate$ empty$
    'skip$
    {
      new.sentence 'output.state :=
    }
  if$
}
```

`mat.article.volume.and.number` This function formats the volume an article was published in. Optionally it appends the issue to the volume.

```
FUNCTION {format.article.volume.and.number}
{
  volume bibinfo.check
  duplicate$ empty$
    'skip$
    { bibinfo.check }
  if$
  format.article.volume.internal
⟨*number⟩
  number bibinfo.check
  duplicate$ empty$
    'skip$
    {
      swap$
      duplicate$ empty$
        { "there's a number but no volume in " cite$ * warning$ }
        'skip$
      if$
      swap$
      "(" swap$ * ")" *
    }
  if$
  *
⟨/number⟩
}
```

4.8 @book related functions

`format.book.authors` This function formats the authors of a book. If no authors were given, the editors are used instead.

```
FUNCTION {format.book.authors}
{
  author empty$
    { format.editors }
    {
      format.authors
      "author and editor" editor either.or.check
    }
  if$
}
```

`format.book.editors` This function formats the editors of a book. If the book has no authors then the editors were already given as a substitute for the authors. In this case, nothing is printed, since the editors should not be given twice.

```
FUNCTION {format.book.editors}
{
  author empty$
    { "" }
    { format.editors }
  if$
}
```

`format.book.volume.internal` This function formats the volume of a book.

```
FUNCTION {format.book.volume.internal}
{
  volume bibinfo.check
  duplicate$ empty$
    'skip$
    {
      tie.or.space.prefix *
      bbl.volume swap$ *
    }
  if$
}
```

`format.book.volume.and.series` This function formats the volume of a book and the series it is part of. For this function to produce any non-empty output both `volume` and `series` must be non-empty. To format the volume without a series the `format.book.volume.noseries` function is used (see below).

```
FUNCTION {format.book.volume.and.series}
{
  series bibinfo.check
  duplicate$ empty$
    'skip$
    {
⟨ChemCommun | ChemEurJ | cv⟩      emphasize
```

Now the formatted series is on the stack.

```
      format.book.volume.internal
⟨ChemCommun | ChemEurJ | cv⟩      bbl.of space.word *
```

stack top-down: formatted volume; formatted series

```
⟨ChemCommun | ChemEurJ | cv⟩      swap$ *
⟨JAmChemSoc | InorgChem⟩      cat.comma
    }
  if$
}
```

`format.book.volume.noseries` This function formats the volume of a book if the `series` field is empty.

```
FUNCTION {format.book.volume.noseries}
{
  series empty$ not
    { "" }
    {
      format.book.volume.internal
⟨ChemCommun | ChemEurJ | cv⟩        emphasize
    }
  if$
}
```

4.9 @misc related functions

`misc.empty.check`

This function checks whether fields of an `@misc` entry are empty and issues a warning.

```
FUNCTION {misc.empty.check}
{
  author empty$ title empty$ howpublished empty$
  month empty$ year empty$ note empty$ url empty$
  and and and and and and
    { "all relevant fields are empty in " cite$ * warning$ }
    'skip$
  if$
}
```

4.10 @program related functions

`format.program.description`

This function formats the description of a program. It also temporarily switches the language.

```
FUNCTION {format.program.description}
{
  description bibinfo.check
  duplicate$ empty$
    'skip$
    { select.language }
  if$
}
```

`mat.program.publisher.address`

This function formats the publisher of a program and its address.

```
FUNCTION {format.program.publisher.address}
{
  publisher bibinfo.check
  format.org.or.pub
}
```

`format.program.title`

This function formats the title of a program.

```
FUNCTION {format.program.title}
{
  title "title" bibinfo.warn
  duplicate$ empty$
    'skip$
    {
      "t" change.case$
      " " swap$ *
      capitalize
      emphasize
    }
  if$
}
```

`format.program.version` This function formats the version of a program.

```
FUNCTION {format.program.version}
{
  version
}
```

4.11 Entry types

`article`

```
FUNCTION {article}
{
  output.bibitem
  format.authors "author"
⟨ChemCommun | ChemEurJ | cv⟩  output.check
⟨JAmChemSoc | InorgChem⟩  output.check.blank
⟨cv⟩  format.article.title "title" output.check
⟨cv⟩  new.sentence 'output.state :=
  format.article.journal "journal"
⟨ChemCommun | ChemEurJ | cv⟩  output.check
⟨JAmChemSoc | InorgChem⟩  output.check.blank
  format.article.date "year"
```

If you change the way journal name and publication year are catenated, do not forget to change `format.article.cat.journal.year`.

```
⟨ChemCommun⟩  output.check
⟨ChemEurJ | cv | JAmChemSoc | InorgChem⟩  output.check.blank
  format.article.volume.and.number output.comma
  eid empty$
    { format.article.pages }
    { format.article.eid }
  if$
  format.article.germanpages output.semicolon
  format.article.erratum output.semicolon
  format.note output
  fin.entry
}
```

`book`

```
FUNCTION {book}
{
  output.bibitem
  format.book.authors "author and editor" output.check
  format.title "title"
⟨ChemCommun | ChemEurJ | cv⟩  output.check.comma
⟨JAmChemSoc | InorgChem⟩  output.check.blank
  format.edition
⟨ChemCommun | ChemEurJ | cv⟩  output.blank
⟨JAmChemSoc | InorgChem⟩  output.comma
⟨JAmChemSoc | InorgChem⟩  format.book.editors output.semicolon
  format.book.volume.and.series output
⟨ChemCommun | ChemEurJ | cv⟩  format.book.volume.noseries output
⟨ChemCommun | ChemEurJ | cv⟩  format.book.editors output.comma
  format.publisher.address output
  format.date "year" output.check.comma
⟨JAmChemSoc | InorgChem⟩  format.volume.noseries output.semicolon
  format.note output
  fin.entry
}
```

booklet

```
FUNCTION {booklet}
{
  output.bibitem
  format.authors output
  format.title "title" output.check
  howpublished bibinfo.check output
  address bibinfo.check output
  format.date output
  format.note output
  fin.entry
}
```

inbook

```
FUNCTION {inbook}
{
  output.bibitem
  format.book.authors "author and editor" output.check
  crossref missing$
    {
      format.title "title"
⟨ChemCommun | ChemEurJ | cv⟩      output.check.comma
⟨JAmChemSoc | InorgChem⟩      output.check.blank
      format.edition
⟨ChemCommun | ChemEurJ | cv⟩      output.blank
⟨JAmChemSoc | InorgChem⟩      output.comma
⟨JAmChemSoc | InorgChem⟩      format.book.editors output.semicolon
      format.book.volume.and.series output
⟨ChemCommun | ChemEurJ | cv⟩      format.book.volume.noseries output
⟨ChemCommun | ChemEurJ | cv⟩      format.book.editors output.comma
      format.publisher.address output
      format.date "year" output.check.comma
```

The *J. Am. Chem. Soc.* requires "; volume, chapter, pages". To handle empty entries gracefully we cat them together and write the entire string afterwards.

```
⟨JAmChemSoc | InorgChem⟩       format.volume.noseries
⟨ChemCommun | ChemEurJ | cv⟩       ""
    }
    {
      format.crossref output.blank
```

Push an empty string since there will not be a volume.

```
      ""
    }
  if$
  format.chapter cat.comma
  format.pages cat.comma
  output
  format.note output
  fin.entry
}
```

incollection

```
FUNCTION {incollection}
{
  output.bibitem
  format.authors "author" output.check
⟨cv⟩  format.title "title" output.check
  crossref missing$
    {
      format.in.booktitle "booktitle" output.check.blank
      format.edition
⟨ChemCommun | ChemEurJ | cv⟩      output.blank
⟨JAmChemSoc | InorgChem⟩      output.comma
⟨JAmChemSoc | InorgChem⟩      format.book.editors output.semicolon
      format.book.volume.and.series output
⟨ChemCommun | ChemEurJ | cv⟩      format.book.volume.noseries output
⟨ChemCommun | ChemEurJ | cv⟩      format.book.editors output.comma
      format.publisher.address output
      format.date "year" output.check.comma
```

The *J. Am. Chem. Soc.* requires "; volume, chapter, pages". To handle empty entries gracefully we cat them together and write the entire string afterwards.

```
⟨JAmChemSoc | InorgChem⟩      format.volume.noseries
⟨ChemCommun | ChemEurJ | cv⟩      ""
    }
    {
      format.crossref output.blank
```

Push an empty string since there will not be a volume.

```
      ""
    }
  if$
  format.chapter cat.comma
  format.pages cat.comma
  output
  format.note output
  fin.entry
}
```

inproceedings

```
FUNCTION {inproceedings}
{
  output.bibitem
  format.authors "author" output.check
⟨cv⟩  format.title "title" output.check
  crossref missing$
    {
      format.in.booktitle "booktitle" output.check.blank
      publisher empty$
        { format.organization.address output }
        {
          organization bibinfo.check output
          format.publisher.address output
        }
      if$
      format.book.volume.and.series output
    }
    { format.crossref output.blank }
  if$
  format.pages "pages" output.check
  format.note output
  format.url output
  fin.entry
}
```

manual

```
FUNCTION {manual}
{
  output.bibitem
  author empty$
    {
      organization bibinfo.check
      duplicate$ empty$
        'pop$
        {
          output
          address bibinfo.check output
        }
      if$
    }
    { format.authors output }
  if$
  format.title "title" output.check
  author empty$
    {
      organization empty$
        { address bibinfo.check output }
        'skip$
      if$
    }
    {
      organization bibinfo.check output
      address bibinfo.check output
    }
  if$
  format.edition output
  format.date output
  format.note output
  format.url output
  fin.entry
}
```

mastersthesis

```
FUNCTION {mastersthesis}
{
  output.bibitem
  format.authors "author" output.check
  bbl.mthesis format.thesis.type output
  school "school" bibinfo.warn output
  address bibinfo.check output
  format.date "year" output.check
  format.note output
  format.url output
  fin.entry
}
```

misc

```
FUNCTION {misc}
{
  output.bibitem
  format.authors output
  format.title output
  howpublished bibinfo.check output
  format.date output
  format.note output
  format.url output
  fin.entry
  misc.empty.check
}
```

phdthesis

```
FUNCTION {phdthesis}
{
  output.bibitem
  format.authors "author" output.check
⟨cv⟩  format.title "title" output.check
  bbl.phdthesis format.thesis.type output
  school "school" bibinfo.warn output
  address bibinfo.check output
  format.date "year" output.check
  format.note output
  format.url output
  fin.entry
}
```

proceedings

```
FUNCTION {proceedings}
{
  output.bibitem
  editor empty$
    { organization bibinfo.check output }
    { format.editors output }
  if$
  format.title "title" output.check
  format.volume output
  format.number.series output
  editor empty$
    {
      publisher empty$
```

```
        'skip$
        { format.publisher.address output }
      if$
    }
    {
      publisher empty$
        { format.organization.address output }
        {
          organization bibinfo.check output
          format.publisher.address output
        }
      if$
     }
  if$
⟨ChemCommun | ChemEurJ | cv⟩  format.date "year" output.check
  format.note output
  format.url output
  fin.entry
}
```

program

```
FUNCTION {program}
{
  output.bibitem
  format.authors output
  format.program.title "title"
⟨ChemCommun | ChemEurJ | cv⟩  output.check
⟨JAmChemSoc | InorgChem⟩  output.check.blank
  format.program.description output.comma
  format.program.version output
⟨ChemCommun | ChemEurJ | cv⟩  format.date output
  format.program.publisher.address output
⟨JAmChemSoc | InorgChem⟩  format.date output.comma
  format.note output
  format.url output
  fin.entry
}
```

techreport

```
FUNCTION {techreport}
{
  output.bibitem
  format.authors "author" output.check
  format.title
  "title" output.check
  format.tr.number output
  institution "institution" bibinfo.warn output
  address bibinfo.check output
  format.date "year" output.check
  format.note output
  format.url output
  fin.entry
}
```

`unpublished`

```
FUNCTION {unpublished}
{
  output.bibitem
  format.authors "author" output.check
  format.title "title" output
  format.date output
  format.note "note" output.check
  format.url output
  fin.entry
}
```

`conference` `conference` is an alias for `inproceedings`.

```
FUNCTION {conference} { inproceedings }
```

`default.type` The default type is `misc`.

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

4.12 Main program

`begin.bib` This function starts the bibliography by opening the `thebibliography` environment.

```
FUNCTION {begin.bib}
{
  "\begin{thebibliography}{"  longest.label  * "}" * write$ newline$
}
```

`end.bib` This function ends the bibliography by closing the `thebibliography` environment.

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

`initialize.longest.label` This function sets up the variables for the longest label.

```
FUNCTION {initialize.longest.label}
{
  "" 'longest.label :=
  #1 'number.label :=
  #0 'longest.label.width :=
}
```

`init.consts` This function sets up some constant used while formatting the bibliography.

```
FUNCTION {init.consts}
{
  #0 'before.all :=
  #1 'mid.sentence :=
  #2 'new.sentence :=
}
```

`longest.label.pass` This function is used to determine the length of the longest label. It checks whether the current label is longer than the current longest label. If so, it updates `longest.label.width`.

```
FUNCTION {longest.label.pass}
{
  number.label int.to.str$ 'label :=
  number.label #1 + 'number.label :=
  label width$ longest.label.width >
    {
      label 'longest.label :=
      label width$ 'longest.label.width :=
    }
    'skip$
  if$
}
```

`write.babel.misc` This function writes the default definitions of some commands used in formatting a citation.

```
FUNCTION {write.babel.misc}
{
  "\providecommand{\bbland}{and}" write$ newline$
  "\providecommand{\bblchap}{chap.}" write$ newline$
  "\providecommand{\bblchapter}{chapter}" write$ newline$
  "\providecommand{\bbletal}{et~al.}" write$ newline$
  "\providecommand{\bbleditors}{editors}" write$ newline$
  "\providecommand{\bbleds}{eds.}" write$ newline$
  "\providecommand{\bbleditor}{editor}" write$ newline$
  "\providecommand{\bbled}{ed.}" write$ newline$
  "\providecommand{\bbledition}{edition}" write$ newline$
  "\providecommand{\bbledn}{ed.}" write$ newline$
  "\providecommand{\bbleidp}{page}" write$ newline$
  "\providecommand{\bbleidpp}{pages}" write$ newline$
  "\providecommand{\bblerratum}{erratum}" write$ newline$
  "\providecommand{\bblin}{in}" write$ newline$
  "\providecommand{\bblmthesis}{Master's thesis}" write$ newline$
  "\providecommand{\bblno}{no.}" write$ newline$
  "\providecommand{\bblnumber}{number}" write$ newline$
  "\providecommand{\bblof}{of}" write$ newline$
  "\providecommand{\bblpage}{page}" write$ newline$
  "\providecommand{\bblpages}{pages}" write$ newline$
  "\providecommand{\bblp}{p}" write$ newline$
  "\providecommand{\bblphdthesis}{Ph.D. thesis}" write$ newline$
  "\providecommand{\bblpp}{pp}" write$ newline$
  "\providecommand{\bbltechrep}{Tech. Rep.}" write$ newline$
  "\providecommand{\bbltechreport}{Technical Report}" write$ newline$
  "\providecommand{\bblvolume}{volume}" write$ newline$
  "\providecommand{\bblvol}{Vol.}" write$ newline$
}
```

`write.babel.months` This function writes the default names of the months.

```
FUNCTION {write.babel.months}
{
  "\providecommand{\bbljan}{January}" write$ newline$
  "\providecommand{\bblfeb}{February}" write$ newline$
  "\providecommand{\bblmar}{March}" write$ newline$
  "\providecommand{\bblapr}{April}" write$ newline$
  "\providecommand{\bblmay}{May}" write$ newline$
  "\providecommand{\bbljun}{June}" write$ newline$
  "\providecommand{\bbljul}{July}" write$ newline$
  "\providecommand{\bblaug}{August}" write$ newline$
  "\providecommand{\bblsep}{September}" write$ newline$
  "\providecommand{\bbloct}{October}" write$ newline$
  "\providecommand{\bblnov}{November}" write$ newline$
  "\providecommand{\bbldec}{December}" write$ newline$
}
```

`write.babel.ordinals` This function writes the default names of the ordinals.

```
FUNCTION {write.babel.ordinals}
{
  "\providecommand{\bblfirst}{First}" write$ newline$
  "\providecommand{\bblfirsto}{1st}" write$ newline$
  "\providecommand{\bblsecond}{Second}" write$ newline$
  "\providecommand{\bblsecondo}{2nd}" write$ newline$
  "\providecommand{\bblthird}{Third}" write$ newline$
  "\providecommand{\bblthirdo}{3rd}" write$ newline$
  "\providecommand{\bblfourth}{Fourth}" write$ newline$
  "\providecommand{\bblfourtho}{4th}" write$ newline$
  "\providecommand{\bblfifth}{Fifth}" write$ newline$
  "\providecommand{\bblfiftho}{5th}" write$ newline$
  "\providecommand{\bblst}{st}" write$ newline$
  "\providecommand{\bblnd}{nd}" write$ newline$
  "\providecommand{\bblrd}{rd}" write$ newline$
  "\providecommand{\bblth}{th}" write$ newline$
}
```

`write.babel` This function writes the default of all words used for formatting a citation.

```
FUNCTION {write.babel}
{
  write.babel.misc
  write.babel.months
  write.babel.ordinals
}
```

`write.commands` This function writes the commands used in a citation.

```
FUNCTION {write.commands}
{
  "\providecommand{\url}[1]{\texttt{#1}}" write$ newline$
  "\providecommand{\urlprefix}{}" write$ newline$
  "\providecommand{\foreignlanguage}[2]{#2}" write$ newline$
  "\providecommand{\Capitalize}[1]{\uppercase{#1}}" write$ newline$
  "\providecommand{\capitalize}[1]{\expandafter\Capitalize#1}" write$ newline$
  "\providecommand{\bibliographycite}[1]{\cite{#1}}" write$ newline$
  write.babel
}
```

`write.header` This function writes the header of the `.bbl` file. It does not, however, start the `thebibliography` environment. The contents of `@preamble` is always written first.

```
FUNCTION {write.header}
{
  preamble$ empty$
    'skip$
    { preamble$ write$ newline$ }
  if$
  write.commands
}
```

`main` Read all required entries from the database. The entries are in citation order.

```
READ
```

Determine the length of the longest label.

```
EXECUTE {initialize.longest.label}
ITERATE {longest.label.pass}
```

Write the header to the file and open the `thebibliography` environment.

```
EXECUTE {init.consts}
EXECUTE {write.header}
EXECUTE {begin.bib}
```

Format all entries

```
ITERATE {call.type$}
```

Close the `thebibliography` environment.

```
EXECUTE {end.bib}
```

Change History

- `format.names`: Added fifteennames option . . . 15

0.2.3

- General: Fixed some minor issues with ACS styles. . . . 1
- `format.editors`: ACS does not print 'Eds.' if there are no authors. . . . 22
- `format.url`: Put URL into a new sentence. . . . 25
- `program`: ACS does not like a delimiter after last author name . . . 40

0.2.4

- General: Removed number of pages from papers with EID in all styles except 'cv'. . . . 1
- `format.article.eid`: Include number of pages only for style 'cv' 28

0.2.5

- General: Minor documentation updates . . . 1

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.