The ACM Transactions (`acmtrans`) bibliography style

Glenn Paulley   Andrew W. Appel   Rebecca L. Davies
David M. Jones

22 August 1995
Version 1.2b β

Contents

1 Introduction

This is the `acmtrans` BibTeX style, which is based Glenn Paulley's `chicago` bibliography style (for further information about the history of this code, please see the comments in `chicago.bst`). The `acmtrans` style was hacked by Andrew W. Appel and Rebecca L. Davies of Princeton University, and further modified and reworked by David M. Jones.

This style implements an "author-year" bibliography format, where in-text citations have the general form "[author name(s) year]" and the Reference list is sorted alphabetically by the author or whatever passes for author in the absence of one. The style also supports abbreviated author lists in citations.

This style must be supplemented by some TeX macros to process the citation information generated by `acmtrans`. The citation information deposited in the bibliography has the following format:

`\citeauthoryear{`⟨*full-author-info*⟩`}{`⟨*abbrev-author-info*⟩`}{`⟨*year*⟩`}`

The LaTeX style should define something like the following:

```
\let\@internalcite\cite
\def\fullcite{\def\citeauthoryear##1##2##3{##1 ##3}\@internalcite}
\def\fullciteA{\def\citeauthoryear##1##2##3{##1}\@internalcite}
```

```
\def\shortcite{\def\citeauthoryear##1##2##3{##2 ##3}\@internalcite}
\def\shortciteA{\def\citeauthoryear##1##2##3{##2}\@internalcite}
\def\citeyear{\def\citeauthoryear##1##2##3{##3}\@internalcite}
```

Additional commands to manipulate different components of a citation can be defined so that, for example, you can list author's names without parentheses if using a citation as a noun or object in a sentence. See the `acmart` document class for a more complete example.

In order to provide more flexibility for use with both LaTeX 2.09 and LaTeX 2_ε, `acmtrans` uses the following macros for all font changes:

`\bibsc` for small caps

`\bibemph` for italics without an italic correction

`\bibemphic` for italics with an italic correction at the end.

For LaTeX 2_ε, the following is appropriate:

```
\let\bibsc\textsc
\let\bibemph\emph
\let\bibemphic\emph
```

For LaTeX 2.09, the following should do:

```
\newcommand{\bibsc}[1]{{\sc#1}}
\newcommand{\bibemph}[1]{{\em#1}}
\newcommand{\bibemphic}[1]{{\em#1\/}}
```

One further macro, `\bibyear`, is needed, to format the year. The following definition is used in the `acmart` class. See the comments before `output.year.check` below and before `\bibyear` in `acmart` for more information.

```
\newcommand{\bibyear}[2]{%
   \unskip\quad\ignorespaces#1\unskip
   \if#2. .\quad \else \quad#2 \fi
}
```

These TeX macro definitions are found in the `acmart` document class.

2 Further features of `acmtrans`

- full names used in citations, but abbreviated citations are available (see above)
- if an entry has a "month", then the month and year are also printed as part of that bibitem.
- all conjunctions use "and" instead of "&"

- major modification from Chicago Manual of Style (13th ed.) is that only the first author in a reference appears last name first- additional authors appear as J. Q. Public.
- pages are listed as "pp. xx-xx" in all entry types except article entries.
- book, inbook, and manual use "location: publisher" (or organization) for address and publisher. All other types list publishers separately.
- "pp." are used to identify page numbers for all entry types except articles.
- organization is used as a citation label if neither author nor editor is present (for manuals).
- "et al." is used for long author and editor lists, or when "others" is used.

3 Modifications and bug fixes from newapa.bst

- added month, year to bib entries if month is present
- fixed bug with In proceedings, added necessary comma after title
- all conjunctions changed to "and" from "&"
- fixed bug with author labels in my.full.label: "et al." now is generated when "others" is an author name
- major modification from Chicago Manual of Style (13th ed.) is that only the first author in a reference appears last name first—additional authors appear as J. Q. Public.
- pages are listed as "pp. xx-xx" in all entry types except article entries. Unnecessary (IMHO) "()" around page numbers were removed, and page numbers now don't end with a period.
- created chicago.sty for use with this bibstyle (required).
- fixed bugs in `format.vol.num.pages` for missing volume, number, and/or pages. Renamed to `format.jour.vol`.
- fixed bug in formatting booktitles: additional period an error if book has a volume.
- fixed bug: editors usually given redundant period before next clause (`format.editors.dot`) removed.
- added label support for organizations, if both author and editor are missing (from alpha.bst). If organization is too long, then the key field is used for abbreviated citations.

- In proceedings or books of several volumes, no comma was written between the "Volume x" and the page numbers (this was intentional in newapa.bst). Fixed.
- Some journals may not have volumes/numbers, only month/year (eg. IEEE Computer). Fixed bug in article style that assumed volume/number was always present.

4 To-do list

Here are a few of the many things that need to be done to finish this style file.

- Figure out what, if any, copyright notice should cover this file.
- Decide what parts of the above documentation to keep. Sort out credit issues.
- Get style rules from ACM and implement them.
- Implement `format.proc.publisher` and sort out other issues mentioned in connection with `inproceedings`.
- Sort out `format.key` business.
- Munge `month` to standardize month abbreviations when authors fail to use the standard strings.
- Sort out journal abbreviations.
- Make `.bbl` file look more like what the ACM wants for its automatic conversion programs. (In fact, we could output SGML code directly for them, if they'd tell us what they want.)
- Rewrite `macrocode` to index BST macros.

5 The `docstrip` modules

This code uses the following modules:

xref	use crossrefs in the bibliography
driver	generate a driver file for the documentation

6 The macros

6.1 The field names and useful utilities

```
ENTRY
  { address
    author
```

```
    booktitle
    chapter
    edition
    editor
    howpublished
    institution
    journal
    key
    month
    note
    number
    organization
    pages
    publisher
```

In addition to the standard fields, we introduce three new fields, **pub-year**, **pub-month** and **pub-address**. These are for use in proceedings, where the **year**, **month** and **address** fields normally refer to the date and location of the conference rather than the publisher. For compatibility with the standard BibTeX styles, we introduce new fields for the publisher information rather than the conference information, although it would probably be more logical for the existing fields always to refer to the publisher and to introduce new fields for the conference information.

```
    pub-year
    pub-month
    pub-address
    school
    series
    title
    type
    volume
    year
  }
  {}
  { label.year extra.label sort.year sort.label }

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

init.state.consts

```
FUNCTION {init.state.consts}
{ #0 'before.all :=
  #1 'mid.sentence :=
  #2 'after.sentence :=
  #3 'after.block :=
}
```

output.nonnull

```
STRINGS { s t u }

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

output.nonnull.colon Use a colon to separate output. Used only for address/publisher combination in book/inbook types, address/institution for manuals, and organization:publisher for proceedings (inproceedings).

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

output

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

output.colon

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

output.check

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

output.check.colon

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

new.block

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

new.sentence

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

```
          if$
        }
    if$
}
```

output.year.check

This is somewhat nasty. We need to put a quad space around the year. Unfortunately, the obvious thing (i.e., putting out "`\quad`⟨*year*⟩`\quad`") won't work because the next call to `output` will add a period *after* the second `\quad`. Because of the way BibTeX writes things, it would be difficult to get the `\quad` in the right place, so instead we cheat by wrapping the year in the macro `\bibyear`, which is defined in `acmart` to look ahead one character and, if necessary, move the period before the second quad space.

This should be thought about some more.

```
FUNCTION {output.year.check}
{ year empty$
     { "empty year in " cite$ * warning$ }
     { add.period$ write$
       " \bibyear{" year * extra.label * "}" *
       mid.sentence 'output.state :=
     }
  if$
}
```

fin.entry

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

not

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

and

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

or

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

new.block.checka

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

new.block.checkb

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

new.sentence.checka

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

new.sentence.checkb

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

field.or.null

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

parenthesize Put parentheses around the top string on the stack.

```
FUNCTION {parenthesize}
{ duplicate$ empty$
    { pop$ "" }
    { " (" swap$ * ")" * }
```

```
  if$
}
```

`emphasize` Emphasize the top string on the stack.

It's a shame that we can't assume people are using LaTeX 2_ε, since the `\emph` command would be much more convenient here. As it is, we have to try to guess whether or not we will need an italic correction, and use either `emphasize` or `emphasize.ic` as appropriate. As a partial solution, we use the macros `\bibemph` and `\bibemphic`, which should be defined by the style file. For LaTeX 2_ε, these can both be equivalent to `\emph`; for LaTeX 2.09, `\bibemphic` should add an italic correction and `\bibemph` should not.

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

`emphasize.ic` Emphasize the top string on the stack, but add a trailing space.

```
FUNCTION {emphasize.ic}
{ duplicate$ empty$
    { pop$ "" }
    { "\bibemphic{" swap$ * "}" * }
  if$
}
```

`emphasize.comma` Emphasize the top string on stack, add a trailing comma and space.

Removed superfluous italic correction, moved comma outside of emphasis. [8/15/95—dmj]

```
FUNCTION {emphasize.comma}
{ duplicate$ empty$
    { pop$ "" }
    { emphasize "," * }
  if$
}
```

`smallcaps` Put the top string on the stack in small caps.

```
FUNCTION {smallcaps}
{ duplicate$ empty$
    { pop$ "" }
    { "\bibsc{" swap$ * "}" * }
  if$
}
```

`format.names` Format bibliographical entries with the last name first. All names are formatted in this routine.

Changed so that entire author list is surrounded by `{\sc` and `}`, instead of surrounding each individual element. Also moved "Junior" part to the very end, so that it appears after the first initial instead of before.

```
INTEGERS { nameptr namesleft numnames }

FUNCTION {format.names}
{ 's :=
  #1 'nameptr :=               % nameptr = 1;
  s num.names$ 'numnames :=    % numnames = num.name$(s);
  numnames 'namesleft :=
    { namesleft #0 > }
    {
%      nameptr #1 =
%        {s nameptr "{vv~}{ll}{, f.}{, jj}" format.name$ 't := } {
        s nameptr "{vv~}{ll}{, f.}{, jj}" format.name$ 't :=
%      } if$
      nameptr #1 >
        { namesleft #1 >
              { ", " * t * }
              { numnames #2 >
                  { "," * }
                  'skip$
                if$
                t "others" =
                    { " et~al." * }
                    { " and " * t * } % from Chicago Manual of Style
                  if$
               }
               if$
             }
            't
        if$
        nameptr #1 + 'nameptr :=          % nameptr += 1;
        namesleft #1 - 'namesleft :=      % namesleft =- 1;
    }
  while$
}
```

my.full.label

```
FUNCTION {my.full.label}
{ 's :=
  #1 'nameptr :=               % nameptr = 1;
  s num.names$ 'numnames :=    % numnames = num.name$(s);
  numnames 'namesleft :=
    { namesleft #0 > }
    { s nameptr "{vv~}{ll}" format.name$ 't :=  % get the next name
      nameptr #1 >
        { namesleft #1 >
              { ", " * t * }
              { numnames #2 >
                  { "," * }
                  'skip$
                if$
```

```
                t "others" =
                    { " et~al." * }
                    { " and " * t * } % from Chicago Manual of Style
                  if$
               }
               if$
             }
           't
       if$
       nameptr #1 + 'nameptr :=          % nameptr += 1;
       namesleft #1 - 'namesleft :=      % namesleft =- 1;
   }
  while$
}
```

format.names.fml

Format names in "familiar" format, with first initial followed by last name. Like format.names, ALL names are formatted.

Changed so that entire author list is surrounded by `{\sc` and `}`, instead of surrounding each individual element.

```
FUNCTION {format.names.fml}
{ 's :=
  #1 'nameptr :=               % nameptr = 1;
  s num.names$ 'numnames :=    % numnames = num.name$(s);
  numnames 'namesleft :=
    { namesleft #0 > }
    { s nameptr "{f.~}{vv~}{ll}{, jj}" format.name$ 't :=
      nameptr #1 >
        { namesleft #1 >
              { ", " * t * }
              { numnames #2 >
                  { "," * }
                  'skip$
                if$
                t "others" =
                    { " et~al." * }
                    { " and " * t * }
                  if$
               }
               if$
             }
           't
       if$
       nameptr #1 + 'nameptr :=          % nameptr += 1;
       namesleft #1 - 'namesleft :=      % namesleft =- 1;
   }
  while$
}
```

format.authors

```
FUNCTION {format.authors}
{ author empty$
    { "" }
    { author format.names smallcaps }
  if$
}
```

`format.key`

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

`format.editors.fml` Format editor names for use in the "in" types: inbook, incollection, inproceedings: first initial, then last names. When editors are the LABEL for an entry, then **format.editor** is used which lists editors by last name first.

Removed parens around "Ed." since that seems to be desired by the ACM. [8/18/95—dmj]

```
FUNCTION {format.editors.fml}
{ editor empty$
    { "" }
    { editor format.names.fml
      editor num.names$ #1 >
    { ", Eds." * }
    { ", Ed." * }
      if$
    }
  if$
}
```

`format.editors` Format editor names for use in labels, last names first.

Removed parens around "Ed." since that seems to be desired by the ACM. [8/18/95—dmj]

```
FUNCTION {format.editors}
{ editor empty$
    { "" }
    { editor format.names smallcaps
      editor num.names$ #1 >
    { ", Eds." * }
    { ", Ed." * }
      if$
    }
  if$
}
```

`format.title`

```
FUNCTION {format.title}
```

```
{ title empty$
    { "" }
    { title "t" change.case$ }
  if$
}
```

n.dashify

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

format.btitle

```
FUNCTION {format.btitle}
{ edition empty$
  { title emphasize }
  { title empty$
```

The following branch is somewhat eccentric, but I'll leave it for now since it's harmless, and in the current code, `format.btitle` is always followed by a call to `output.check`, which will generate a warning message if necessary.

```
    { title emphasize }
    { volume empty$     % gnp - check for volume, then don't need period
       { title emphasize.ic edition " ed." * parenthesize * "." * }
       { title emphasize.ic edition " ed." * parenthesize * }
      if$
    }
    if$
  }
  if$
```

```
}
```

format.emphasize.booktitle

```
FUNCTION {format.emphasize.booktitle}
{ edition empty$
  { booktitle emphasize.ic }
  { booktitle empty$
```

The following branch is also somewhat eccentric, but I'll leave it for now since it should never be taken in the current code.

```
    { booktitle emphasize }
    { volume empty$  % gnp - extra period an error if book has a volume
        { booktitle emphasize.ic edition " ed." * parenthesize * "." *}
        { booktitle emphasize.ic edition " ed." * parenthesize * }
      if$
      }
    if$
    }
  if$
}
```

tie.or.space.connect

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

either.or.check

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

format.bvolume

```
FUNCTION {format.bvolume}
{
    series empty$
        { "" }
        { series emphasize }
    if$
    volume empty$
        'skip$
        { " vol." volume tie.or.space.connect * }
    if$
}
```

```
% FUNCTION {format.bvolume}
% { volume empty$
%     { "" }
%     { "Volume" volume tie.or.space.connect % gnp - changed to mixed case
%       series empty$
%         'skip$
%         { " of " * series emphasize * }
%       if$
%       "volume and number" number either.or.check
%     }
%   if$
% }
```

format.number.series

```
FUNCTION {format.number.series}
{
    volume empty$
        { number empty$
            { series field.or.null }
            { "Number" number tie.or.space.connect
              series empty$
                { "there's a number but no series in " cite$ * warning$ }
                { " in " * series * }
              if$
            }
          if$
        }
        { "" }
    if$
}
```

multi.page.check

```
INTEGERS { multiresult }

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
}
```

format.pages gnp - removed ()

```
FUNCTION {format.pages}
{ pages empty$
  { "" }
  {
%    pages multi.page.check {
        "pp.\ " pages n.dashify tie.or.space.connect
%       } { "pp.\ " pages tie.or.space.connect }
%    if$
  }
  if$
}
```

format.jour.vol By Young (and Spencer) GNP - fixed bugs with missing volume, number, and/or pages

Format journal, volume, number, pages for article types.

```
FUNCTION {format.jour.vol}
{ journal empty$
    { "no journal in " cite$ * warning$
      "" }
    { journal emphasize.ic }
    if$
  number empty$
    { volume empty$
       { "no number and no volume in " cite$ * warning$
         "" * }
       { "~" * volume emphasize * }
      if$
    }
    { volume empty$
      {"no volume for " cite$ * warning$
       "~" * number * }
      { "~" *
        volume emphasize.comma
        "~" * number * * }
      if$
     month empty$
     'skip$
     {month parenthesize *}
     if$
    }
  if$
  pages empty$
    {"page numbers missing in " cite$ * warning$
```

```
     "" * } % gnp - place a null string on the stack for output
    { duplicate$ empty$
      { pop$ format.pages }
      { ", " *  pages n.dashify * } % gnp - removed pp. for articles
      if$
    }
  if$
}
```

format.chapter.pages

```
FUNCTION {format.chapter.pages}
{ chapter empty$
    'format.pages
    { type empty$
        { "Chapter" } % gnp - changed to mixed case
        { type "t" change.case$ }
      if$
      chapter tie.or.space.connect
      pages empty$
        {"page numbers missing in " cite$ * warning$}%gnp - added check
        { ", " * format.pages * }
      if$
    }
  if$
}
```

format.in.ed.booktitle

```
FUNCTION {format.in.ed.booktitle}
{ booktitle empty$
  { "" }
  { editor empty$
    { "In " format.emphasize.booktitle * }
    { "In " format.editors.fml * ", " * format.emphasize.booktitle * }
    if$
  }
  if$
}
```

format.thesis.title

```
FUNCTION {format.thesis.title}
{ title empty$
    'skip$
%   {pop$
   { { "" }
     title "t" change.case$
   }
  if$
}
```

format.tr.number

```
FUNCTION {format.tr.number}
{ type empty$
    { "Technical Report" }
    'type
  if$
  number empty$
    { "t" change.case$ }
    { number tie.or.space.connect }
  if$
}
```

format.crossref

Merged `format.article.crossref` and `format.incoll.inproc.crossref` into a single macro, `format.crossref`, which has proper spacing.

```
⟨*xref⟩
FUNCTION {format.crossref}
{
    "See \citeN{" crossref * "}" *
}
⟨/xref⟩
```

format.crossref.editor

```
⟨*xref⟩
FUNCTION {format.crossref.editor}
{ editor #1 "{vv~}{ll}" format.name$
  editor num.names$ duplicate$
  #2 >
    { pop$ " et~al." * }
    { #2 <
    'skip$
    { editor #2 "{ff }{vv }{ll}{ jj}" format.name$ "others" =
        { " et~al." * }
        { " and " * editor #2 "{vv~}{ll}" format.name$ * }
      if$
    }
      if$
    }
  if$
}
⟨/xref⟩
```

format.book.crossref

```
⟨*xref⟩
FUNCTION {format.book.crossref}
{ volume empty$
    { "empty volume in " cite$ * "'s crossref of " * crossref * warning$
      "In "
    }
    { "Volume" volume tie.or.space.connect % gnp - changed to mixed case
```

```
      " of " *
    }
  if$
  editor empty$
  editor field.or.null author field.or.null =
  or
    { key empty$
    { series empty$
        { "need editor, key, or series for " cite$ * " to crossref " *
          crossref * warning$
          "" *
        }
        { series emphasize.ic }
      if$
    }
    { key * }
      if$
    }
    { format.crossref.editor * }
  if$
  " \citeN{" * crossref * "}" *
}
⟨/xref⟩
```

`format.lab.names` Determines “short” names for the abbreviated author information. “Long” labels are created in calc.label, using the routine my.full.label to format author and editor fields.

There are 4 cases for labels ($n = 3$ in the example):

1. one author: Foo
2. one to n: Foo, Bar and Baz
3. use of “and others”: Foo, Bar et al.
4. more than n: Foo et al.

```
FUNCTION {format.lab.names}
{ 's :=
  s num.names$ 'numnames :=
  numnames #2 >    % change number to number of others allowed before
                   % forcing "et al".
    { s #1 "{vv~}{ll}" format.name$ " et~al." * }
    {
      numnames #1 - 'namesleft :=
      #2 'nameptr :=
      s #1 "{vv~}{ll}" format.name$
        { namesleft #0 > }
        { nameptr numnames =
            { s nameptr "{ff }{vv }{ll}{ jj}" format.name$ "others" =
                { " et~al." * }
```

```
        { " and " * s nameptr "{vv~}{ll}" format.name$ * }
       if$
      }
      { ", " * s nameptr "{vv~}{ll}" format.name$ * }
     if$
     nameptr #1 + 'nameptr :=
     namesleft #1 - 'namesleft :=
    }
   while$
  }
 if$
}
```

author.key.label

```
FUNCTION {author.key.label}
{ author empty$
  { key empty$
     { "no key, author in " cite$ * warning$
      cite$ #1 #3 substring$ }
    'key
   if$
  }
  { author format.lab.names }
 if$
}
```

editor.key.label

```
FUNCTION {editor.key.label}
{ editor empty$
  { key empty$
     { "no key, editor in " cite$ * warning$
      cite$ #1 #3 substring$ }
     'key
    if$
   }
   { editor format.lab.names }
 if$
}
```

author.key.org.label added - gnp. Provide label formatting by organization if author is null.

```
FUNCTION {author.key.org.label}
{ author empty$
  { organization empty$
    { key empty$
      { "no key, author or organization in " cite$ * warning$
       cite$ #1 #3 substring$ }
      'key
     if$
    }
```

```
        'organization
      if$
    }
    { author format.lab.names }
  if$
}
```

`editor.key.org.label` added - gnp. Provide label formatting by organization if editor is null.

```
FUNCTION {editor.key.org.label}
{ editor empty$
    { organization empty$
        { key empty$
            { "no key, editor or organization in " cite$ * warning$
              cite$ #1 #3 substring$ }
            'key
          if$
        }
        'organization
      if$
    }
    { editor format.lab.names }
  if$
}
```

`author.editor.key.label`

```
FUNCTION {author.editor.key.label}
{ author empty$
    { editor empty$
          { key empty$
               { "no key, author, or editor in " cite$ * warning$
                 cite$ #1 #3 substring$ }
             'key
           if$
         }
          { editor format.lab.names }
      if$
    }
    { author format.lab.names }
  if$
}
```

`calc.label` GNP: changed. Form label for BibTeX entry. The classification of which fields are used for which type of entry (book, inbook, etc.) are taken from alpha.bst. The change here from newapa is to also include organization as a citation label if author or editor is missing. See also **author.organization.sort**, **editor.organization.sort**.

```
FUNCTION {calc.label}
{ type$ "book" =
  type$ "inbook" =
```

```
  or
    'author.editor.key.label
    { type$ "proceedings" =
        'editor.key.org.label
        { type$ "manual" =
            'author.key.org.label
            'author.key.label
          if$
        }
      if$
    }
  if$
  author empty$  % generate the full label citation information.
    { editor empty$
        { organization empty$
           { "no author, editor, or organization in " cite$ * warning$
             "??" }
           'organization
           if$
        }
        { editor my.full.label }
        if$
    }
    { author my.full.label }
  if$
```

leave label on the stack, to be popped when required.

```
  "}{" * swap$ * "}{" *
%  year field.or.null purify$ #-1 #4 substring$ *
```

save the year for sort processing afterwards (adding a, b, c, etc.)

```
  year field.or.null purify$ #-1 #4 substring$
  'label.year :=
}
```

output.bibitem

```
FUNCTION {output.bibitem}
{ newline$
  "\bibitem[\protect\citeauthoryear{" write$
  calc.label write$
  sort.year write$
  "}]{" write$
  cite$ write$
  "}" write$
  newline$
  ""
  before.all 'output.state :=
}
```

6.2 The entry types

6.2.1 The article type

article

```
FUNCTION {article}
{
    output.bibitem
    format.authors "author" output.check
```

Some but not all types contain the following line. It's questionable whether it belongs there at all, but if it does, shouldn't it be in *all* the types?

```
    author format.key output          % added
    output.year.check                 % added
    new.block
    format.title "title" output.check
    new.block
⟨*xref⟩
    crossref missing$
        {
⟨/xref⟩
            format.jour.vol output
⟨*xref⟩
        }
        {
            format.crossref output.nonnull
            format.pages output
        }
    if$
⟨/xref⟩
  new.block
  note output
  fin.entry
}
```

6.2.2 The book type

book

```
FUNCTION {book}
{ output.bibitem
  author empty$
    { format.editors "author and editor" output.check }
    { format.authors
          output.nonnull
      crossref missing$
        { "author and editor" editor either.or.check }
        'skip$
      if$
    }
  if$
```

```
  output.year.check       % added
  new.block
  format.btitle "title" output.check
⟨*xref⟩
  crossref missing$
    {
⟨/xref⟩
      format.bvolume output
      new.block
      format.number.series output
      new.sentence
      publisher "publisher" output.check
      address output
⟨*xref⟩
    }
    {
      new.block
      format.book.crossref output.nonnull
    }
  if$
⟨/xref⟩
  new.block
  note output
  fin.entry
}
```

6.2.3 The booklet type

booklet

```
FUNCTION {booklet}
{ output.bibitem
  format.authors output
  author format.key output          % added
  output.year.check                 % added
  new.block
  format.title "title" output.check
  new.block
  howpublished output
  address output
  new.block
  note output
  fin.entry
}
```

6.2.4 The inbook type

inbook

```
FUNCTION {inbook}
{ output.bibitem
```

```
  author empty$
    { format.editors "author and editor" output.check
    }
    { format.authors output.nonnull
      crossref missing$
    { "author and editor" editor either.or.check }
    'skip$
      if$
    }
  if$
  output.year.check                 % added
  new.block
  format.btitle "title" output.check
⟨*xref⟩
  crossref missing$
    {
⟨/xref⟩
      format.bvolume output
      format.chapter.pages "chapter and pages" output.check
      new.block
      format.number.series output
      new.sentence
      publisher "publisher" output.check.colon
      address output
⟨*xref⟩
    }
    {
      format.chapter.pages "chapter and pages" output.check
      new.block
      format.book.crossref output.nonnull
    }
  if$
⟨/xref⟩
  new.block
  note output
  fin.entry
}
```

6.2.5 The incollection type

incollection

```
FUNCTION {incollection}
{ output.bibitem
  format.authors "author" output.check
  author format.key output       % added
  output.year.check              % added
  new.block
  format.title "title" output.check
  new.block
```

```
⟨*xref⟩
  crossref missing$
  {
⟨/xref⟩
    format.in.ed.booktitle "booktitle" output.check
    format.bvolume output
    format.number.series output
    format.chapter.pages output % gnp - was special.output.nonnull
%                                 left out comma before page numbers
    new.sentence
    address output
    publisher "publisher" output.check.colon
⟨*xref⟩
  }
  {
    format.crossref output.nonnull
    format.chapter.pages output
  }
  if$
⟨/xref⟩
  new.block
  note output
  fin.entry
}
```

6.2.6 The inproceedings type

format.date

```
FUNCTION {format.date}
{ year empty$
    { month empty$
        { "" }
        { "there's a month but no year in " cite$ * warning$
          month
        }
      if$
    }
    { month empty$
        'year
        { month " " * year * }
      if$
    }
  if$
}
```

format.proc.loc.date

```
FUNCTION {format.proc.loc.date}
{
    address field.or.null
```

```
    format.date
    duplicate$ empty$
        'pop$
        { swap$
          duplicate$ empty$
            'pop$
            { ", " * swap$ * }
          if$
        }
    if$
    parenthesize
    *
}
```

inproceedings Added `format.proc.loc.date` to format proceedings location and date. Should also add something like `format.proc.publisher` to handle the `pub-year`, `pub-month` and `pub-address` fields, but should first find out from ACM how such things should be handled. In particular, if year of publication is different from year of conference, which should be used in citation (presumably the year of publication).

A related problem is to decide exactly what the `year` refers to. In the standard styles, if there is a `year` but no `address`, the year is assumed to refer to the date of publication, while if there is an `address`, the address and date are assumed to refer to the conference. For true compatibility with the standard styles, we should probably preserve this distinction.

```
FUNCTION {inproceedings}
{
  output.bibitem
  format.authors "author" output.check
  author format.key output            % added
  output.year.check                   % added
  new.block
  format.title "title" output.check
  new.block
⟨*xref⟩
  crossref missing$
    {
⟨/xref⟩
      format.in.ed.booktitle "booktitle" output.check
      format.bvolume output
      format.number.series output
%      address output
      format.proc.loc.date %output
%      format.pages output
      new.sentence
      organization output
      publisher output %.colon
      format.pages output
⟨*xref⟩
     }
```

```
    {
      format.crossref output.nonnull
      format.pages output
    }
  if$
⟨/xref⟩
  new.block
  note output
  fin.entry
}
```

6.2.7 The conference type

conference

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

6.2.8 The manual type

manual

```
FUNCTION {manual}
{ output.bibitem
  author empty$
    { editor empty$
      { organization "organization" output.check
        organization format.key output }  % if all else fails, use key
      { format.editors "author and editor" output.check }
      if$
    }
    { format.authors output.nonnull }
    if$
  output.year.check                  % added
  new.block
  format.btitle "title" output.check
  organization address new.block.checkb
```

Reversed the order of “address” and “organization”, added the “:”.

```
  address output
  organization "organization" output.check.colon
%  address output
%  ":" output
%  organization output
  new.block
  note output
  fin.entry
}
```

6.2.9 The mastersthesis type

mastersthesis

```
FUNCTION {mastersthesis}
{ output.bibitem
  format.authors "author" output.check
  author format.key output          % added
  output.year.check                 % added
  new.block
  format.title "title" output.check
  new.block
  "Master's thesis" format.thesis.title output.nonnull
  school "school" output.check
  address output
  new.block
  note output
  fin.entry
}
```

6.2.10 The misc type

misc

```
FUNCTION {misc}
{ output.bibitem
  format.authors output
  author format.key output            % added
  output.year.check                   % added
  title howpublished new.block.checkb
  format.title output
  new.block
  howpublished output
  new.block
  note output
  fin.entry
}
```

6.2.11 The phdthesis type

phdthesis

```
FUNCTION {phdthesis}
{ output.bibitem
  format.authors "author" output.check
  author format.key output            % added
  output.year.check                   % added
  new.block
  format.btitle "title" output.check
  new.block
  format.thesis.title "Ph.\ D. thesis" output.nonnull
  school "school" output.check
  address output
  new.block
  note output
```

```
  fin.entry
}
```

6.2.12 The proceedings type

proceedings

```
FUNCTION {proceedings}
{ output.bibitem
  editor empty$
    { organization output
      organization format.key output } % gnp - changed from
    { format.editors output.nonnull }  % author format.key
  if$
% author format.key output             % gnp - removed (should be either
%                                        editor or organization
  output.year.check                    % added (newapa)
  new.block
  format.btitle "title" output.check
  format.bvolume output
  format.number.series output
%  address output
    format.proc.loc.date % output
  new.sentence
  organization output
  publisher output.colon
  new.block
  note output
  fin.entry
}
```

6.2.13 The techreport type

techreport

```
FUNCTION {techreport}
{ output.bibitem
  format.authors "author" output.check
  author format.key output             % added
  output.year.check                    % added
  new.block
  format.title "title" output.check
  new.block
  format.tr.number
     month empty$
     'skip$
     { month parenthesize * }
     if$
  output.nonnull
  institution "institution" output.check
  address output
```

```
  new.block
  note output
  fin.entry
}
```

6.2.14 The `unpublished` type

`unpublished`

```
FUNCTION {unpublished}
{ output.bibitem
  format.authors "author" output.check
  author format.key output              % added
  output.year.check                      % added
  new.block
  format.title "title" output.check
  new.block
  note "note" output.check
  fin.entry
}
```

6.2.15 The `default.type` type

`default.type`

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

6.3 Standard abbreviations

The following abbreviations should all be checked to make sure they correspond to what the ACM wants.

```
MACRO {jan} {"Jan."}

MACRO {feb} {"Feb."}

MACRO {mar} {"March"}

MACRO {apr} {"April"}

MACRO {may} {"May"}

MACRO {jun} {"June"}

MACRO {jul} {"July"}

MACRO {aug} {"Aug."}

MACRO {sep} {"Sept."}

MACRO {oct} {"Oct."}

```

```
MACRO {nov} {"Nov."}
MACRO {dec} {"Dec."}
MACRO {acmcs} {"ACM Computing Surveys"}
MACRO {acta} {"Acta Informatica"}
MACRO {ai} {"Artificial Intelligence"}
MACRO {cacm} {"Commun. ACM"}
MACRO {ibmjrd} {"IBM Journal of Research and Development"}
MACRO {ibmsj} {"IBM Systems Journal"}
MACRO {ieeese} {"IEEE Trans. Softw. Eng."}
MACRO {ieeetc} {"IEEE Transactions on Computers"}
MACRO {ieeetcad}
 {"IEEE Transactions on Computer-Aided Design of Integrated Circuits"}
MACRO {ipl} {"Inf. Process. Lett."}
MACRO {ic} {"Inf. Comput."}
MACRO {jacm} {"Journal of the ACM"}
MACRO {jcss} {"J. Comput. Syst. Sci."}
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 Trans. on Programm. Lang. Syst."}
MACRO {tcs} {"Theor. Comp. Sci."}
READ
```

6.4 Sorting the entries

sortify

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

chop.word

```
INTEGERS { len }

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

sort.format.names

```
FUNCTION {sort.format.names}
{ 's :=
  #1 'nameptr :=
  ""
  s num.names$ 'numnames :=
  numnames 'namesleft :=
    { namesleft #0 > }
    { nameptr #1 >
          { "   " * }
         'skip$
      if$
      s nameptr "{vv{ } }{ll{ }}{  f{ }}{  jj{ }}" format.name$ 't :=
      nameptr numnames = t "others" = and
          { " et~al" * }
          { t sortify * }
      if$
      nameptr #1 + 'nameptr :=
      namesleft #1 - 'namesleft :=
    }
  while$
}
```

sort.format.title

```
FUNCTION {sort.format.title}
{ 't :=
  "A " #2
    "An " #3
      "The " #4 t chop.word
```

```
    chop.word
  chop.word
  sortify
  #1 global.max$ substring$
}
```

author.sort

```
FUNCTION {author.sort}
{ author empty$
    { key empty$
        { "to sort, need author or key in " cite$ * warning$
          "" }
        { key sortify }
      if$
    }
    { author sort.format.names }
  if$
}
```

editor.sort

```
FUNCTION {editor.sort}
{ editor empty$
    { key empty$
        { "to sort, need editor or key in " cite$ * warning$
          ""
        }
        { key sortify }
      if$
    }
    { editor sort.format.names }
  if$
}
```

author.editor.sort

```
FUNCTION {author.editor.sort}
{
    author empty$
        { "missing author in " cite$ * warning$
          editor empty$
            { key empty$
                { "to sort, need author, editor, or key in "
                  cite$ * warning$
                  ""
                }
                {
                  key sortify
                }
              if$
            }
```

```
      {
       editor sort.format.names
      }
     if$
    }
    {
     author sort.format.names
    }
  if$
}
```

`author.organization.sort` added - GNP. Stack author or organization for sorting (from alpha.bst). Unlike alpha.bst, we need entire names, not abbreviations

```
FUNCTION {author.organization.sort}
{ author empty$
  { organization empty$
    { key empty$
      { "to sort, need author, organization, or key in "
       cite$ * warning$
       ""
      }
      { key sortify }
     if$
    }
    { organization sortify }
   if$
  }
  { author sort.format.names }
 if$
}
```

`editor.organization.sort` added - GNP. Stack editor or organization for sorting (from alpha.bst). Unlike alpha.bst, we need entire names, not abbreviations

```
FUNCTION {editor.organization.sort}
{ editor empty$
  { organization empty$
    { key empty$
      { "to sort, need editor, organization, or key in "
       cite$ * warning$
       ""
      }
      { key sortify }
     if$
    }
    { organization sortify }
   if$
  }
  { editor sort.format.names }
 if$
}
```

`presort` Presort creates the bibentry's label via a call to `calc.label`, and then sorts the entries based on entry type. Chicago.bst adds support for including organizations as the sort key; the following is stolen from alpha.bst.

```
FUNCTION {presort}
{ calc.label sortify % recalculate bibitem label
  year field.or.null purify$ #-1 #4 substring$ * % add year
  "    "
  *
  type$ "book" =
  type$ "inbook" =
  or
    'author.editor.sort
    { type$ "proceedings" =
        'editor.organization.sort
        { type$ "manual" =
            'author.organization.sort
            'author.sort
          if$
        }
      if$
    }
  if$
  #1 entry.max$ substring$        % added for newapa
  'sort.label :=                  % added for newapa
  sort.label                      % added for newapa
  *
  "    "
  *
  title field.or.null
  sort.format.title
  *
  #1 entry.max$ substring$
  'sort.key$ :=
}

ITERATE {presort}

SORT    % by label, year, author/editor, title

```

`init.extra.label.stuff`

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

INTEGERS { last.extra.num }

FUNCTION {init.extra.label.stuff}
{ #0 int.to.chr$ 'last.label :=
  "" 'next.extra :=
  #0 'last.extra.num :=
}
```

`forward.pass` Pass through all entries, comparing current entry to last one. Need to concatenate year to the stack (done by `calc.label`) to determine if two entries are the same (see `presort`)

```
FUNCTION {forward.pass}
{ last.label
  calc.label year field.or.null purify$ #-1 #4 substring$ * % add year
  #1 entry.max$ substring$ =     % are they equal?
     { last.extra.num #1 + 'last.extra.num :=
       last.extra.num int.to.chr$ 'extra.label :=
     }
     { "a" chr.to.int$ 'last.extra.num :=
       "" 'extra.label :=
       calc.label
       year field.or.null purify$ #-1 #4 substring$ * % add year
       #1 entry.max$ substring$ 'last.label := % assign to last.label
     }
  if$
}
```

`reverse.pass`

```
FUNCTION {reverse.pass}
{ next.extra "b" =
    { "a" 'extra.label := }
      'skip$
  if$
  label.year extra.label * 'sort.year :=
  extra.label 'next.extra :=
}
```

```
EXECUTE {init.extra.label.stuff}

ITERATE {forward.pass}

REVERSE {reverse.pass}
```

`bib.sort.order`

```
FUNCTION {bib.sort.order}
{ sort.label
  "    "
  *
  year field.or.null sortify
  *
  "    "
  *
  title field.or.null
  sort.format.title
  *
  #1 entry.max$ substring$
  'sort.key$ :=
}
```

```
ITERATE {bib.sort.order}

SORT % by sort.label, year, title --- giving final bib. order.
```

begin.bib

```
FUNCTION {begin.bib}
{ preamble$ empty$
    'skip$
    { preamble$ write$ newline$ }
  if$
  "\begin{thebibliography}{}" write$ newline$
}
```

```
EXECUTE {begin.bib}

EXECUTE {init.state.consts}

ITERATE {call.type$}
```

end.bib

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

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