The AMS Export BibTeX style [amsxport]

American Mathematical Society
Michael Downes
David M. Jones

Version 2.00, 2004/06/28

Contents

1 Introduction

`amsxport` is the collective name for a collection of BibTeX styles that form a bridge between BibTeX and `amsrefs`. It produces output in which the structural information of the original BibTeX database file is retained. No ad hoc visual formatting is introduced that would hamper the application of alternative design specifications when rendering the information on paper or screen. This makes it possible to have BibTeX handle only extraction and sorting and leave the visual appearance to be controlled entirely by LaTeX. The process of applying typical bibliography specs to exported data is illustrated in the `amsrefs` package. The authoritative description of the export format is in the `amsrefs` documentation, inasmuch as interpreting the format is the chief task of the `amsrefs` package.

Each of the five `.bst` files generated from this file corresponds to a combination of `amsrefs` package options, as follows:

amsrn: This is the default style, used when none of the following is appropriate.

amsru: This is used when the `citation-order` option is specified. It is identical to `amsrn` except that the bibliography items are not sorted.

amsra: This is when with the `alphabetic` option is specified, byt the `y2k` option is *not* also specified.

amsry: This is used when both the `alphabetic` and `y2k` options are specified.

amsrs: This is used when the `shortalphabetic` option is specified.

Readers may wish to refer also to the standard BibTeX documentation (`btxdoc.tex`, `btxhak.tex`, `btxbst.doc`) for background information.

2 Implementation

The list below gives the fields that are expected to occur. (Other fields will be discarded without comment.) suggested fields.

```
⟨*bst⟩
ENTRY
  {
    address       institution  number       status
    archive       isbn         organization subtitle
    author        issn         pages        title
    booktitle     journal      part         translator
    chapter       key          pii          type
    date          language     preprint     url
    edition       meeting      publisher    volume
    editor        month        review       xid
    eprint        mrnumber     school       year
    howpublished  note         series
  }
  { }
  { label }
```

Some comments about specific fields:

archive: The archive that holds the eprint listed in the `eprint` field.

author: BibTEX doesn't allow this field (or any field) to be used more than once, but in the output its contents will be split into a separate key-value pair for each name. Likewise for `editor` and `translator`.

crossref: This is a built-in field type, so it doesn't occur in the argument of ENTRY above.

date: This is a generalization of the `year` and `month` fields. Its value should be written in ISO 8601 format, e.g., 1987-06-05; but the day and month are omissible, so this can be easily be used instead of the `year` field.

edition: For books. The BibTEX documentation suggests that the value of this field should be an ordinal word such as "Second". This may be because the BibTEX language provides only the weakest sort of support for an *is-numeric* test.

eprint: Electronic preprint information such as for www.arXiv.org.

institution: What's the difference between an `institution` and an `organization`? The BibTEX documentation says that `institution` should be used for technical reports and `organization` for other entry types.

isbn: International Standard Book Number.

issn: International Standard Serial Number. See the discussion of journal abbreviations.

language: Language of the work. This is especially desirable when the value of the title field is a translation of the true title (most often when the original language is one like Arabic or Chinese that poses rendering problems for non-native software).

meeting: Since BibTEX doesn't allow a field name to be the same as a function name, we have to use `meeting` instead of `conference` if we want to give the title of a conference in a dedicated field.

mrnumber: An alias for `review` which we provide because MathSciNet has been using it in the BibTEX output that it provides for a couple of years already.

organization: See the comment for `institution`.

pii: Publisher Item Identifier

part: This is for a long journal article that is published in separate issues of the same journal. There should be a separate BibTEX entry for each part (though all the ones after part 1 could crossref the first one).

preprint: It is not possible quite yet to assume that all preprints should be classified as eprints. If you have a preprint that is not an eprint, use this field to give the "address" where the preprint may be found (institution, preprint number, etc.).

review: Review numbers or similar pointers, e.g., for Mathematical Reviews or Zentralblatt.

school: The analog of "organization", for a thesis.

status: Typically used for notes such as "to appear" or "in preparation" with journal articles.

subtitle: Typically used with a multipart journal article to give a subtitle for each part.

translator: This needs no explanation, surely? Except perhaps to note that the standard BibTEX styles don't provide it.

url: Universal Resource Locator.

xid: This is used by a cross-referenced item to pass its identity to child entries that refer to it. It would not be necessary if BibTEX left the `crossref` field value accessible, but for some reason that I have not found yet (skimming in the source code) it is cleared internally before there is any opportunity to save it.

Practically speaking: For every `crossref` in your `.bib` files, the target should contain a matching `xid` field.

More on extra fields. In the Shelah bibliography, the `author` field is given without any accents in the author names, and the normal TEX version of the author names is given in a separate field called `trueauthor`.

Also, for MR reviews:

```
review = {MR 48:3735},
```

And for each author a country is indicated with the `fromwhere` field, e.g.,

```
fromwhere = {UK, IL},
```

This means the first author is from the United Kingdom and the second one from Israel.

3 Macros for month names

```
MACRO {jan} {"-01"}
MACRO {feb} {"-02"}
MACRO {mar} {"-03"}
MACRO {apr} {"-04"}
MACRO {may} {"-05"}
MACRO {jun} {"-06"}
MACRO {jul} {"-07"}
MACRO {aug} {"-08"}
MACRO {sep} {"-09"}
MACRO {oct} {"-10"}
MACRO {nov} {"-11"}
MACRO {dec} {"-12"}
```

Some scratch variables and global variables.

```
STRINGS { p s t l }
```

```
INTEGERS { len ptr }
⟨bst⟩
```

Note: BibTEX 0.99 requires a blank line in certain contexts, such as after the `READ` *command, and will choke if it is missing. When it is necessary to add a blank line at the boundary of a macro code section we use a* `<bst>` *marker.*

Some utility functions.

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

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

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

FUNCTION {TRUE} { #1 }

FUNCTION {FALSE} { #0 }
⟨bst⟩
```

`incr.ptr`

```
FUNCTION {incr.ptr} { ptr #1 + 'ptr := }
⟨bst⟩
```

`current.char` Returns: first character of STRING

```
FUNCTION {current.char} { ptr #1 substring$ }
```

`copy.two.chars` The function `copy.two.chars` appends the next two characters from s to STRING.

Arg: STRING (partial copy of pages string)
Returns: Modified version of STRING
Side effects: advances ptr

```
FUNCTION {copy.two.chars} { t * incr.ptr s current.char * incr.ptr }
⟨bst⟩
```

`is.digit`

```
FUNCTION {is.digit} {
    chr.to.int$
    duplicate$
    "0" chr.to.int$ #1 - >
    swap$
    "9" chr.to.int$ #1 + <
    AND
}
⟨bst⟩
```

`is.lowercase.letter` In a previous implementation the `is.letter` function first lowercased the character using `change.case$`, then tested it against the range a–z. But `change.case$` issues a warning if it is applied to a lone brace character.

```
FUNCTION {is.lowercase.letter} {
    chr.to.int$ duplicate$
    "a" chr.to.int$ #1 - >
    swap$
    "z" chr.to.int$ #1 + <
    AND
}
```

is.uppercase.letter

```
FUNCTION {is.uppercase.letter} {
    chr.to.int$ duplicate$
    "A" chr.to.int$ #1 - >
    swap$
    "Z" chr.to.int$ #1 + <
    AND
}
```

is.letter

```
FUNCTION {is.letter} {
    duplicate$
    is.lowercase.letter
        { pop$ TRUE }
        { is.uppercase.letter }
    if$
}
```

skip.accent This implementation of `skip.accent` simply skips ahead one character whenever a backslash is found. This handles things like `\v`, `\u`, `\k`, `\r`. Although this is overly simplistic it is still an improvement over the previous method which was to not even make the attempt. [mjd,2001-10-27]

```
FUNCTION {skip.accent} { pop$ incr.ptr }
```

strip.label

```
FUNCTION {strip.label} {
    's :=
    #1 'ptr :=
    ""
    { s ptr #1 substring$ duplicate$ empty$ NOT }
        { duplicate$ is.letter
            { * }
            { duplicate$ "\" =
                { skip.accent }
                'pop$
              if$
            }
          if$
          incr.ptr
        }
    while$
```

When the loop ends we have an extra empty-string on top of the stack to get rid of.

```
    pop$
}
```

sortify

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

rpad Warning: The use of `text.length$` here and in `lpad` restricts their use to strings that we know do not contain any special characters. Since we only use `lpad` for field names and `rpad` for `purify$`ed strings, this is ok.

⟨*string*⟩ ⟨*int*⟩ `rpad`

```
FUNCTION {rpad} {
    swap$
    'l :=
    l text.length$ - 'len :=
    { len #0 > }
        {
            l " " * 'l :=
            len #1 - 'len :=
        }
    while$
    l
}
```

lpad ⟨*string*⟩ ⟨*int*⟩ `lpad`

```
FUNCTION {lpad} {
    swap$
    'l :=
    l text.length$ - 'len :=
    { len #0 > }
        {
            " " l * 'l :=
            len #1 - 'len :=
        }
    while$
    l
}
```

`field.or.null`

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

`missing.or.empty`

```
FUNCTION {missing.or.empty} { duplicate$ missing$ swap$ empty$ OR }
```

⟨bst⟩

`ndash.and.skip.hyphens` The function `append.ndash.and.skip.hyphens` adds `\ndash` to STRING when a hyphen is found and advances ptr to the next non-hyphen character in `s`.

Arg: STRING (partial copy of pages string)
Returns: modified version of STRING
Side effects: advances ptr

```
FUNCTION {append.ndash.and.skip.hyphens} {
    "\ndash " *
    incr.ptr
    { s current.char "-" = }
        { incr.ptr }
    while$
}
```

`n.dashify` In the `n.dashify` function we store the given string in `s`, push an empty string on the stack, and start examining the characters of `s`. For a non-hyphen character, we just append a copy of it to the top string. For a hyphen, we append `\ndash` to the top string and advance the pointer until we reach a non-hyphen character. And there is one more exceptional case: for a backslash, we copy two characters instead of one. This keeps us from erroneously translating `\-` to `\\ndash`. One would scarcely expect to see an instance of `\-` in a page number field but let's face it, in actual use nearly anything can and does happen.

Arg: STRING (value of `pages` field)

Returns: STRING (dashified version of input string)

```
FUNCTION {n.dashify} {
    's :=
    #1 'ptr :=
    ""
    { s current.char 't :=  t "" = NOT }
      { t "\" =
          { copy.two.chars }
          { t "-" =
              { append.ndash.and.skip.hyphens }
              { t * incr.ptr }
            if$
          }
        if$
      }
    while$
}
```

`missing.warning` Standard warning message for a missing or empty field. For the user we call any such field 'missing' without respect to the distinction made by BibTEX between missing and empty.

```
FUNCTION {missing.warning} {
    "missing " swap$ * " in " * cite$ * warning$
}
⟨bst⟩
```

`string.length` Because BibTEX doesn't provide a straightforward `string.length` function (`text.length$` counts "special characters" as a single character), it appears necessary to implement one the hard way.

```
INTEGERS { string.ptr }

FUNCTION {string.length} {
    #1 'string.ptr :=
    { duplicate$ string.ptr #1 substring$ "" = 'FALSE 'TRUE if$ }
        { string.ptr #1 + 'string.ptr := }
    while$
    pop$
    string.ptr #1 -
}
```

`format.title`

```
FUNCTION {format.title}{
    duplicate$
    missing.or.empty
        { pop$ "" }
```

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

start.field

```
FUNCTION {start.field} {
    #12 lpad "={" * write$
}
```

fin.field

```
FUNCTION {fin.field} { "}," write$ newline$ }
```

⟨bst⟩

write.field

```
FUNCTION {write.field} {
    duplicate$
    missing$
        { pop$ missing.warning }
        { duplicate$ empty$
            { pop$ missing.warning }
            { swap$ start.field write$ fin.field }
          if$
        }
    if$
}
```

optional.field

```
FUNCTION {optional.field} {
    duplicate$
    missing.or.empty
        { pop$ pop$ }
        { swap$ start.field write$ fin.field }
    if$
}
```

optional.title.field

```
FUNCTION {optional.title.field} {
    duplicate$
    missing.or.empty
        { pop$ pop$ }
        { format.title swap$ start.field write$ fin.field }
    if$
}
```

optional.pages.field

```
FUNCTION {optional.pages.field} {
    duplicate$
    missing.or.empty
        { pop$ pop$ }
        { swap$ start.field
          n.dashify
          write$ fin.field
        }
    if$
}
```

optional.mr.field

```
FUNCTION {optional.mr.field} {
    duplicate$
    missing.or.empty
        { pop$ pop$ }
        {
          swap$ start.field
          "\MR{" swap$ * "}" * write$ fin.field
        }
    if$
}
```

remove.ordinal.suffix

```
FUNCTION {remove.ordinal.suffix} {
    's :=
    s string.length 'len :=
    #1 'ptr :=
    { ptr len < s ptr #1 substring$ is.digit AND }
        'incr.ptr
    while$
    s ptr global.max$ substring$ 'p :=
    p "st" =
        { TRUE }
        { p "nd" =
            { TRUE }
            { p "rd" =
                { TRUE }
                { p "th" =
                    { TRUE }
                    { FALSE }
                  if$
                }
              if$
            }
          if$
        }
    if$
    { s #1 ptr #1 - substring$ }
    { s }
  if$
}
```

optional.edition.field

```
FUNCTION {optional.edition.field} {
    duplicate$ missing.or.empty
        { pop$ pop$ }
        { swap$ start.field
          remove.ordinal.suffix
          write$ fin.field
        }
    if$
}
```

4 Formatting author names and editor names

Take a name list in BibTEX form (names separated by the word "and") and output the desired form of each name.

```
INTEGERS { nameptr numnames namesleft }
STRINGS { namelist fieldname }
```

`format.name` The `format.name` function operates on a single name, producing a string of the form

```
von Last, First, Jr.
```

Args: *namelist*, *index* (top; integer)

Returns: formatted version of the *n*th name in *namelist* where integer *index* specifies *n*.

```
FUNCTION {format.name} { "{vv~}{ll}{, ff}{, jj}" format.name$ }
⟨bst⟩
```

`optional.name.field` The function `optional.name.field` handles the task of splitting a multiple-name field value into multiple fields with single values.

Args: *fieldname*, *namelist* (top)

Side effects: writes a key/value pair to the output file for each name in *namelist*.

```
FUNCTION {optional.name.field}{
    duplicate$ missing.or.empty
        { pop$ pop$ }
        { swap$ 'fieldname :=
          duplicate$ num.names$
          'namesleft :=
          #1 'ptr :=
          { namesleft #0 > }
            { fieldname start.field
```

Copy the name string before running `format.name` on it.

```
              duplicate$ ptr format.name write$
              fin.field
              ptr #1 + 'ptr :=
              namesleft #1 - 'namesleft :=
            }
          while$
```

At this point we have an extra copy of the name string on the stack.

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

5 Handling year, month, date

`assemble.date.field` In the output we produce a `date` field instead of the year and month fields normally used in BibTEX files. In the date field we use ISO date notation (e.g., 1987-06-05) to facilitate switching between full and abbreviated month names. The month and day parts are frequently absent, making the date field equivalent to a year field.

```
FUNCTION {assemble.date.field}
{ date missing$
    { year missing.or.empty
```

```
        { status missing.or.empty
            { "No year or other date information for "
               cite$ * warning$ }
            'skip$
          if$
          "status"
        }
        { year month missing$ { "" } { month } if$ * }
      if$
    }
    { date }
  if$
```

If the date string is now equal to "status", it means that we have a status field or we already gave a warning about lack of date info; in either case omit the date field.

```
  duplicate$ "status" =
    { pop$ pop$ }
    { swap$ start.field write$ fin.field }
  if$
}
```

6 Journal abbreviations

Use of journal abbreviations is recommended to make supplying ISSN numbers easier. To use this feature, you need a STRING definition of the form

```
@STRING{cpam="cpam/0010-3640/"
        #"Communications in Pure and Applied Mathematics"}
```

Then when you write

```
journal=cpam,
```

exporting with `amsxport` will produce

```
journal={cpam},
ISSN={0010-3640},
```

The text before the ISSN number in the STRING definition is preferably a repetition of the abbreviation name (leaving expansion to be done on the LaTeX side); or it could be the full journal name.

Warning: If you include braces and write `journal={cpam},` BibTeX will not expand the abbreviation but leave it as the field value; and then you will not get automatic lookup of the ISSN number.

optional.journal.field

```
FUNCTION {optional.journal.field} {
    duplicate$ missing.or.empty
        { pop$ pop$ }
        {
            's :=
            start.field
            #1 'ptr :=
            s current.char is.lowercase.letter
                {
                    s "/" * 's :=
                    { s ptr #1 substring$ "/" = NOT }
                        { incr.ptr }
                    while$
```

Journal name started with a lowercase letter, but we couldn't find no slash? All right, just let it go through as it is.

```
                    s ptr #1 + #1 substring$ empty$
                        { s #1 ptr #1 - substring$ }
```

But if we did find a slash, then we probably have found us an ISSN number that we can write out.

```
                        {
                            s #1 ptr #1 - substring$
                            write$
                            fin.field
```

ISSNs are always nine characters in length (four digits, hyphen, four digits).

```
                            "ISSN" start.field
                            s ptr #1 + #9 substring$
                        }
                    if$
                }
                { s }
            if$
            write$
            fin.field
        }
    if$
}
```

7 Type info

optional.type.field

Putz around with the type info a little, for theses mainly. Two types here: the type field, and the entry type.

```
FUNCTION {optional.type.field}
{ duplicate$ missing.or.empty
    { type$ "mastersthesis" = { "Master's Thesis" }
        { type$ "phdthesis" = { "Ph.D. Thesis" }
          { "" } if$ } if$
      swap$ pop$
    }
    'skip$
  if$
  duplicate$ empty$
    { pop$ pop$ }
    { swap$ start.field write$ fin.field }
  if$
}
```

8 Miscellaneous tail-end information

url.note.status.review

The following items are common to all entry types are output at the tail end of the entry.

```
FUNCTION {url.note.status.review} {
    "url" url optional.field
    "note" note optional.field
    "status" status optional.field
```

The `review` field should perhaps be translated into multiple fields like author names, if more than one review is given.

```
    "review" review optional.field
```

Data from MathSciNet will have the Math Reviews number in an mrnumber field. We assume that a given entry will have a review field or an mrnumber field but not both.

```
    "review" mrnumber optional.mr.field
}
```

9 Wrapper for each entry

start.entry

```
FUNCTION {start.entry} {
    newline$
    "\bib{" cite$ * "}{" *
```

Merge Master's thesis and Ph.D. thesis into a single type.

```
    type$ "mastersthesis" = type$ "phdthesis" =  OR
        { "thesis" }
        { type$ }
    if$
    * "}{" * write$
    newline$
⟨*debug⟩
    "sort.label" label sortify optional.field
    "sort.key" sort.key$ optional.field
⟨/debug⟩
}
```

fin.entry

```
FUNCTION {fin.entry} {
    "}" write$
    newline$
}
```

10 Entry types

Here are the types of entries that are normally allowed in a BibTEX file:

book: A book with an explicit publisher.

booklet: A work that is printed and bound, but without a named publisher or sponsoring institution.

inbook: A part of a book, which may be a chapter (or section or whatever) and/or a range of pages.

incollection: A part of a book having its own title.

manual: Technical documentation.

mastersthesis: A Master's thesis.

phdthesis: A PhD thesis.

proceedings: The proceedings of a conference.

techreport: A report published by a school or other institution, usually numbered within a series.

article: An article from a journal or magazine.

inproceedings: An article in a conference proceedings.

conference: An alias for `inproceedings`.

unpublished: A document having an author and title, but not formally published.

misc: Use this type when nothing else fits.

`article` A journal article differs from an inproceedings article by not having booktitle, publisher, editor, and other such info.

```
FUNCTION {article} {
    start.entry
    "author" author optional.name.field
    "translator" translator optional.name.field
    "title" title format.title write.field
    "subtitle" subtitle optional.title.field
    "language" language optional.field
    "organization" organization optional.field
    "how" howpublished optional.field
```

Construct a date value from date, year, month fields

```
    "date" assemble.date.field
    "ISSN" issn optional.field
    "journal" journal optional.journal.field
    "volume" volume optional.field
    "number" number optional.field
    "pages" pages optional.pages.field
    "PII" pii optional.field
    "archive" archive optional.field
    "eprint" eprint optional.field
    "preprint" preprint optional.field
    url.note.status.review
    fin.entry
}
```

`inproceedings` An inproceedings entry may have xid, booktitle, meeting, publisher info; it is not expected to have journal, eprint, or preprint info.

```
FUNCTION {inproceedings} {
    start.entry
    "author" author optional.name.field
    "translator" translator optional.name.field
    "title" title format.title write.field
    "subtitle" subtitle optional.title.field
    "language" language optional.field
    "organization" organization optional.field
    "how" howpublished optional.field
```

Construct a date value from date, year, month fields

```
    "date" assemble.date.field
    "xid" xid optional.field
```

The following fields might be inherited from the parent of an article in a proceedings volume or collection.

```
    "conference" meeting optional.field
    "booktitle" booktitle optional.title.field
```

Include the edition in case this is an inbook entry.

```
    "edition" edition optional.edition.field
    "editor" editor optional.name.field
    "series" series optional.field
    "volume" volume optional.field
    "publisher" publisher optional.field
    "address" address optional.field
    "pages" pages optional.pages.field
```

```
    url.note.status.review
    fin.entry
}
```

inbook

```
FUNCTION {inbook} { inproceedings }
```

incollection

```
FUNCTION {incollection} { inproceedings }
```

inproceedings

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

book The book type includes a meeting field because otherwise we'd have to repeat the entire definition for the proceedings type with that as the sole difference.

```
FUNCTION {book} {
    start.entry
    "author" author optional.name.field
    "editor" editor optional.name.field
    "translator" translator optional.name.field
    "title" title missing$ { booktitle } { title } if$
       format.title write.field
    "subtitle" subtitle optional.title.field
    "type" type optional.type.field
    "language" language optional.field
    "conference" meeting optional.field
    "edition" edition optional.edition.field
    "series" series optional.field
    "publisher" publisher optional.field
    organization missing.or.empty
        { "institution" institution optional.field }
        { "organization" organization optional.field }
    if$
    "address" address optional.field
    "how" howpublished optional.field
    "date" year write.field
    "volume" volume optional.field
```

The number here is intended for a tech report.

```
    "number" number optional.field
    "ISBN" isbn optional.field
    url.note.status.review
    fin.entry
}
```

booklet

```
{booklet}
FUNCTION {booklet} { book }
```

manual

```
FUNCTION {manual} { book }
```

masterthesis

```
FUNCTION {mastersthesis} { book }

```

```
FUNCTION {phdthesis} { book }

FUNCTION {proceedings} { book }

FUNCTION {collection} { book }

FUNCTION {techreport} { book }

FUNCTION {unpublished} { book }

FUNCTION {misc} { book }

FUNCTION {default.type} { misc }
```
⟨bst⟩

BibTEX 0.99 keels over if READ does not have a blank line after it.

⟨bst⟩
```
READ
```
⟨bst⟩

11 Sorting the entries

Since the final labels will be produced by the `amsrefs` package, we don't need to worry about generating suffixes to disambiguate between equal stems. All we have to do is strive to generate stems that are identical to the ones that `amsrefs` will generate, so that `amsxport` will sort its output in a fashion that will be consistent with the labels.

Overall, this simplifies our job, since we don't have to keep track of duplicate labels, so we can process each entry in isolation. However, we do have to be more careful about the handling of text accents and text symbols in order to maintain compatibility with `amsrefs`.

See (*inter alia*) the sections "Lexical structure of names" and "Generating alphabetic lables" in the `amsrefs` implementation documentation for more information.

⟨∗sort⟩

`chop.word`

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

`sort.format.title`

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

```
489     sortify
490     #1 global.max$ substring$
491 }
```

format.lab.names

```
492 ⟨∗alpha | short⟩
493 FUNCTION {format.lab.names} {
494   's :=
495     s num.names$ 'numnames :=
496     numnames #1 >
497       { numnames #4 >
498           { #3 'namesleft := }
499           { numnames 'namesleft := }
500         if$
501         #1 'nameptr :=
502         ""
503           { namesleft #0 > }
504           { nameptr numnames =
505               { s nameptr "{ff }{vv }{ll}{ jj}" format.name$ "others" =
506                   { "{\etalchar{+}}" * }
507                   { s nameptr "{v{}}{l{}}" format.name$ * }
508                 if$
509               }
510               { s nameptr "{v{}}{l{}}" format.name$ * }
511             if$
512             nameptr #1 + 'nameptr :=
513             namesleft #1 - 'namesleft :=
514           }
515         while$
516         numnames #4 >
517           { "{\etalchar{+}}" * }
518           'skip$
519         if$
520       }
521       { s #1 "{v{}}{l{}}" format.name$
522         duplicate$ text.length$ #2 <
523 ⟨alpha⟩           { pop$ s #1 "{ll}" format.name$ #3 text.prefix$ }
524 ⟨short⟩           { pop$ s #1 "{ll}" format.name$ #1 text.prefix$ }
525           'skip$
526         if$
527       }
528     if$
529 }
530
531 FUNCTION {author.key.label}
532 { author empty$
533     { key empty$
534 { cite$ #1 #3 substring$ }
535 { key #3 text.prefix$ }
536       if$
537     }
538     { author format.lab.names }
539   if$
540 }
```

```
541
542 FUNCTION {author.editor.key.label}
543 { author empty$
544     { editor empty$
545 { key empty$
546     { cite$ #1 #3 substring$ }
547     { key #3 text.prefix$ }
548   if$
549 }
550 { editor format.lab.names }
551       if$
552     }
553     { author format.lab.names }
554   if$
555 }
556
557 FUNCTION {author.key.organization.label}
558 { author empty$
559     { key empty$
560 { organization empty$
561     { cite$ #1 #3 substring$ }
562     { "The " #4 organization chop.word #3 text.prefix$ }
563   if$
564 }
565 { key #3 text.prefix$ }
566       if$
567     }
568     { author format.lab.names }
569   if$
570 }
571
572 FUNCTION {editor.key.organization.label}
573 { editor empty$
574     { key empty$
575 { organization empty$
576     { cite$ #1 #3 substring$ }
577     { "The " #4 organization chop.word #3 text.prefix$ }
578   if$
579 }
580 { key #3 text.prefix$ }
581       if$
582     }
583     { editor format.lab.names }
584   if$
585 }
586
587 FUNCTION {calc.label}{
588     type$ "book" =  type$ "inbook" =  OR
589     'author.editor.key.label
590     { type$ "proceedings" =
591 'editor.key.organization.label
592 { type$ "manual" =
593     'author.key.organization.label
594     'author.key.label
```

```
595   if$
596 }
597       if$
598     }
599     if$
600     strip.label
601     sortify
602 ⟨*alpha⟩
603     year field.or.null
604 ⟨!y2k⟩     #3 #8 substring$
605     purify$ *
606 ⟨/alpha⟩
607     'label :=
608 }
609 ⟨/alpha | short⟩
```

sort.format.names

```
610 FUNCTION {sort.format.names} {
611     's :=
612     #1 'ptr :=
613     ""
614     s num.names$ 'numnames :=
615     numnames 'namesleft :=
616       { namesleft #0 > }
617       { ptr #1 >
618           { " " * }
619           'skip$
620         if$
621         s ptr "{ll}{ ff}{ jj}" format.name$ 't :=
622         ptr numnames = t "others" = AND
623           { "et al" }
624           { t sortify }
625         if$
626         #20 rpad *
627         ptr #1 + 'ptr :=
628         namesleft #1 - 'namesleft :=
629       }
630     while$
631 }
```

sort.warning

```
632 FUNCTION {sort.warning} {
633     "to sort, need " swap$ * " or key in " * cite$ * warning$
634 }
```

author.sort

```
635 FUNCTION {author.sort}
636 { author empty$
637     { key empty$
638         { "author" sort.warning
639           ""
640         }
641         { key sortify }
642       if$
```

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

author.editor.sort

```
FUNCTION {author.editor.sort}
{ author empty$
    { editor empty$
        { key empty$
            { "author, editor," sort.warning
              ""
            }
            { key sortify }
          if$
        }
        { editor sort.format.names }
      if$
    }
    { author sort.format.names }
  if$
}
```

thor.organization.sort

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

itor.organization.sort

```
FUNCTION {editor.organization.sort}
{ editor empty$
    { organization empty$
        { key empty$
            { "editor, organization," sort.warning
              ""
            }
            { key sortify }
          if$
        }
        { "The " #4 organization chop.word sortify }
      if$
```

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

presort

```
FUNCTION {presort} {
⟨*alpha | short⟩
    calc.label
    label #20 rpad
⟨/alpha | short⟩
    type$ "book" = type$ "inbook" = OR type$ "collection" = OR
        'author.editor.sort
        { type$ "proceedings" =
            'editor.organization.sort
            { type$ "manual" =
                'author.organization.sort
                'author.sort
              if$
            }
          if$
        }
    if$
⟨alpha | short⟩     * " " *
⟨!alpha | short⟩     year field.or.null sortify #12 rpad * " " *
    title field.or.null sort.format.title *
    #1 entry.max$ substring$
    'sort.key$ :=
}

ITERATE {presort}

SORT
⟨/sort⟩
```

12 Final pass for output

write.with.newlines

The `write.with.newlines` functions stores a string in `s` and runs through it looking for occurrences of `^^M`; when one is found, the substring before it is written out with a newline, the remainder is left in `s`, and the cycle begins again.

```
FUNCTION {write.with.newlines}
{ 's :=  #1 'ptr :=
  { s ptr #2 + #1 substring$ "" = NOT }
  { s ptr #3 substring$ "^^M" =
    { s #1 ptr #1 - substring$ write$ newline$
      ptr #3 + 'ptr :=
      s ptr global.max$ substring$ 's :=
      #1 'ptr :=
    }
    { incr.ptr }
    if$
  }
  while$
```

```
  s write$ newline$
}
```

begin.bin

```
FUNCTION {begin.bib} {
    preamble$ empty$
        'skip$
        { preamble$ write.with.newlines }
    if$
    "% \bib, bibdiv, biblist are defined by the amsrefs package."
        write$ newline$
    "\begin{bibdiv}" write$ newline$
    "\begin{biblist}" write$ newline$
}
```

end.bin

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

EXECUTE {begin.bib}

ITERATE {call.type$}

EXECUTE {end.bib}
```

⟨/bst⟩

13 Journal abbreviations

By putting the ISSN into the LaTeX document, we make it easier to construct www queries from the bibliography data. These string definitions tie together journal abbreviations, ISSN numbers, and journal names.

These are the journals that are cited most frequently by American Mathematical Society authors at the present time (2000 CE).

The format used here is designed to make it easy for BibTeX to retrieve an ISSN number when given a journal abbreviation.

⟨∗abbrevs⟩

```
@string{aa="aa/0065-1036/"
  #"Acta Arithmetica"}
@string{actamath="actamath/0001-5962/"
  #"Acta Mathematica"}
@string{asms="asms/0001-6969/"
  #"Acta Scientiarum Mathematicarum"}
@string{advmath="advmath/0001-8708/"
  #"Advances in Mathematics"}
@string{ajm="ajm/0002-9327/"
  #"American Journal of Mathematics"}
@string{amm="amm/0002-9890/"
  #"American Mathematical Monthly"}
@string{aifg="aifg/0373-0956/"
  #"Annales de l'Institut Fourier"}
@string{am2="am2/0003-486X/Annals of Mathematics"}
@string{ap="ap/0091-1798/"
```

```
  #"The Annals of Probability"}
@string{asens4="asens4/0012-9593/"
  #"Annales Scientifiques de l'\'{E}cole Normale Sup\'{e}rieure"}
@string{amb="amb/0003-889X/Archiv der Mathematik"}
@string{arma="arma/0003-9527/"
  #"Archive for Rational Mechanics and Analysis"}
@string{bamsn="bamsn/0273-0979/"
  #"Bulletin (New Series) of the American Mathematical Society"}
@string{bams="bams/0004-9727/"
  #"Bulletin of the Australian Mathematical Society"}
@string{blms="blms/0024-6093/"
  #"Bulletin of the London Mathematical Society"}
@string{bsmf="bsmf/0037-9484/"
  #"Bulletin de la Soci\'{e}t\'{e} Math\'{e}matique de France"}
@string{craspi="craspi/0764-4442/"
  #"Comptes Rendus de l'Acad\'{e}mie des Sciences (Paris)"
  #" S\'{e}rie I Math\'{e}matique"}
@string{cjm="cjm/0008-414X/Canadian Journal of Mathematics"}
@string{cmb="cmb/0008-4395/"
  #"Canadian Mathematical Bulletin"}
@string{colloqmath="colloqmath/0010-1354/"
  #"Colloquium Mathematicum"}
@string{commalgebra="commalgebra/0092-7872/"
  #"Communications in Algebra"}
@string{cmp="cmp/0010-3616/"
  #"Communications in Mathematical Physics"}
@string{cpam="cpam/0010-3640/"
  #"Communications on Pure and Applied Mathematics"}
@string{cmh="cmh/0010-2571/"
  #"Commentarii Mathematici Helvetici"}
@string{cvta="cvta/0278-1077/"
  #"Complex Variables"}
@string{compositiomath="compositiomath/0010-437X/"
  #"Compositio Mathematica"}
@string{c="c/0010-485X/Computing"}
@string{constrapprox="constrapprox/0176-4276/"
  #"Constructive Approximation"}
@string{dmj="dmj/0012-7094/"
  #"Duke Mathematical Journal"}
@string{eraams="eraams/1079-6762/"
  #"Electronic Research Announcements"
  #" of the American Mathematical Society"}
@string{etds="etds/0143-3857/"
  #"Ergodic Theory and Dynamical Systems"}
@string{fm="fm/0016-2736/"
  #"Fundamenta Mathematicae"}
@string{gd="gd/0046-5755/"
  #"Geometriae Dedicata"}
@string{illinoisjmath="illinoisjmath/0019-2082/"
  #"Illinois Journal of Mathematics"}
@string{iumj="iumj/0022-2518/"
  #"Indiana University Mathematics"}
@string{im="im/0020-9910/"
  #"Inventiones Mathematicae"}
```

```
@string{israeljmath="israeljmath/0021-2172/"
  #"Israel Journal of Mathematics"}
@string{iansm="iansm/??/"
  #"Izvestiya Akademii Nauk SSSR. Seriya Matematicheskaya"}
@string{ja="ja/0021-8693/"
  #"Journal of Algebra"}
@string{jams="jams/0894-0347/"
  #"Journal of the American Mathematical Society"}
@string{jam="jam/0021-7670/"
  #"Journal d'Analyse Math\'{e}matique"}
@string{jamm="jamm/0021-8928/"
  #"Journal of Applied Mathematics and Mechanics"}
@string{jat="jat/0021-9045/"
  #"Journal of Approximation Theory"}
@string{jamsa="jamsa/0263-6115/"
  #"Journal of the Australian Mathematical Society"}
@string{jcta="jcta/0097-3165/"
  #"Journal of Combinatorial Theory"}
@string{jde="jde/0022-0396/"
  #"Journal of Differential Equations"}
@string{jdg="jdg/0022-040X/Journal of Differential Geometry"}
@string{jfa="jfa/0022-1236/"
  #"Journal of Functional Analysis"}
@string{jlms2="jlms2/0024-6107/"
  #"Journal of the London Mathematical Society"}
@string{jmaa="jmaa/0022-247X/"
  #"Journal of Mathematical Analysis and Applications"}
@string{jmp="jmp/0022-2488/"
  #"Journal of Mathematical Physics"}
@string{jmsj="jmsj/0025-5645/"
  #"Journal of the Mathematical Society of Japan"}
@string{jnt="jnt/0022-314X/Journal of Number Theory"}
@string{jot="jot/0379-4024/"
  #"Journal of Operator Theory"}
@string{jpaa="jpaa/0022-4049/"
  #"Journal of Pure and Applied Algebra"}
% Braces instead of quotes, to hide the embedded \" control sequence
@string{jram={jram/0075-4102/}
  #{Journal f\"{u}r die Reine und Angewandte Mathematik}}
@string{jsl="jsl/0022-4812/"
  #"The Journal of Symbolic Logic"}
@string{laa="laa/0024-3795/"
  #"Linear Algebra and its Applications"}
@string{mm="mm/0025-2611/"
  #"Manuscripta Mathematica"}
@string{matzametki="matzametki/0025-567X/Matematicheskie Zametki"}
@string{ma="ma/0025-5831/"
  #"Mathematische Annalen"}
@string{mc="mc/0025-5718/"
  #"Mathematics of Computation"}
@string{mj="mj/0025-5513/"
  #"Mathematica Japonica"}
@string{mn="mn/0025-584X/Mathematische Nachrichten"}
@string{mpcps="mpcps/0305-0041/"
```

```
  #"Mathematical Proceedings of the"
  #" Cambridge Philosophical Society"}
@string{mrl="mrl/1073-2780/"
  #"Mathematical Research Letters"}
@string{ms="ms/0025-5521/"
  #"Mathematica Scandinavica"}
@string{mathz="mathz/0025-5874/"
  #"Mathematische Zeitschrift"}
@string{m="m/0025-5793/"
  #"Mathematika"}
@string{mmj="mmj/0026-2285/"
  #"The Michigan Mathematical Journal"}
@string{nmj="nmj/0027-7630/"
  #"Nagoya Mathematical Journal"}
@string{na="na/0362-546X/Nonlinear Analysis"}
@string{nm="nm/0029-599X/Numerische Mathematik"}
@string{ojm="ojm/0030-6126/"
  #"Osaka Journal of Mathematics"}
@string{pjm="pjm/0030-8730/"
  #"Pacific Journal of Mathematics"}
@string{pams="pams/0002-9939/"
  #"Proceedings of the American Mathematical Society"}
@string{pems2="pems2/0013-0915/"
  #"Proceedings of the Edinburgh Mathematical Society"}
@string{pjaa="pjaa/0386-2194/"
  #"Proceedings of the Japan Academy"}
@string{plms3="plms3/0024-6115/"
  #"Proceedings of the London Mathematical Society"}
@string{qjmo2="qjmo2/0033-5606/"
  #"The Quarterly Journal of Mathematics"}
@string{rmi="rmi/0213-2230/"
  #"Revista Matem\'{a}tica Iberoamericana"}
@string{rmjm="rmjm/0035-7596/"
  #"Rocky Mountain Journal of Mathematics"}
@string{sjam="sjam/0036-1399/"
  #"SIAM Journal on Applied Mathematics"}
@string{sjc="sjc/0097-5397/"
  #"SIAM Journal on Computing"}
@string{sjma="sjma/0036-1410/"
  #"SIAM Journal on Mathematical Analysis"}
@string{sjna="sjna/0036-1429/"
  #"SIAM Journal on Numerical Analysis"}
@string{sr="sr/0036-1445/"
  #"SIAM Review"}
@string{sm="sm/0039-3223/"
  #"Studia Mathematica"}
@string{tjm="tjm/0387-3870/"
  #"Tokyo Journal of Mathematics"}
@string{ta="ta/0166-8641/"
  #"Topology and its Applications"}
@string{t="t/0040-9383/"
  #"Topology"}
@string{tams="tams/0002-9947/"
  #"Transactions of the American Mathematical Society"}
```

937 ⟨/abbrevs⟩

14 To Do

—When there is no author or editor info to create a label from: Use the initial letters of all the words in the title. Plus the year. Maybe year first.

—Add linelength and indent options to control the output?

—Discuss apacite "originalfoo" fields etc.

—Discuss Beebe's work.

—Add bibliography.

—Add AUDIO, VIDEO entry types.

—Add a field for MR subject classification numbers and the like?

—Write a converter to convert from typical .bib data to preferred form. Have to go via perl or something to capture STRING defs and PREAMBLE and interspersed comments? Or emacs lisp? Provide .bst version as fallback.

—Alpha labels: add numbers after sorting? subscript option? L3Y2a option.

—Test multi-author articles with ten-plus authors. What happens with the alpha labels then?

—For url field especially, check length and prebreak by hand instead of relying on BibTeX's slapdash stick-in-a-percent-at-column-72 regardless of where it happens to fall.

—How about a general 'contributor' field.

—Test with a wimpy older version of BibTeX.

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.