Mciteplus: Enhanced Multicitations

Michael Shell*

with special thanks to Joseph Wright

Version 1.2, September 13, 2013

http://www.michaelshell.org/tex/mciteplus/

Abstract

This LaTeX 2$_\varepsilon$ package is an enhanced reimplementation of Thorsten Ohl's mcite package which provides support for the grouping of multiple citations together as is often done in physics journals. An extensive set of features provide for other applications such as reference sublisting.

Contents

*See http://www.michaelshell.org/ for current contact information.

Manuscript originally created on January 15, 2008; revised September 13, 2013. The latest version of this package can be obtained at CTAN [1].

1 Introduction

Mciteplus is a reimplementation of Thorsten Ohl's (http://theorie.physik.uni-wuerzburg.de/~ohl/) mcite LaTeX package [2], which provides for "collapsed" citations (i.e., a grouping of multiple references under a single collective reference) as is often done in physics journals. Using Thorsten Ohl's example in the documentation for mcite:

[1] S. L. Glashow, Nucl. Phys. **22**, 579 (1961).

[2] A. Salam, in *Elementary Particle Theory*, edited by N. Svartholm (Almquist and Wiksell, Stockholm, 1968), pp. 367–377.

[3] S. Weinberg, Phys. Rev. Lett. **19**, 1264 (1967).

becomes:[1]

[1] S. L. Glashow, Nucl. Phys. **22**, 579 (1961); A. Salam, in *Elementary Particle Theory*, edited by N. Svartholm (Almquist and Wiksell, Stockholm, 1968), pp. 367–377; S. Weinberg, Phys. Rev. Lett. **19**, 1264 (1967).

1.1 Features

Mciteplus offers the following features many of which are not available with the original mcite:

- Entry punctuation can be controlled by the bibstyle (.bst) as well as the user.
- No "double periods" when an entry already ends with a period (e.g., an abbreviated journal name).
- Support for sublists.
- Support for multiple bibliographies and/or auxiliary files.
- The bibliography sample label width is automatically updated to account for the changes in the numbering due to the grouped entries. Maximum label width information is available to user.
- Compatible with the natbib package. (The bibstyle must support both mciteplus as well as natbib.)
- Compatible with the REVTeX4 class. (The bibstyle must support both mciteplus as well as natbib.)
- Support for the optional argument of `\cite[]{}`.
- Provides a means to allow users to use almost any cite command via custom command wrappers and the ability to manually disable all automatic internal hooks.
- Mciteplus compatible sorting bibstyles are possible via the use of a "mcitetail" field in the BibTeX database entries. See section 3.1 for details.

[1] Formatting produced by the apsrevM.bst bibstyle.

1.2 File List

The files in this package are as follows:

`mciteplus.sty` The mciteplus LaTeX package.

`mciteplus_doc.pdf` The user manual (this document).

`mciteplus_doc.tex` The LaTeX source of the user manual.

`mciteplus_code.txt` Selected .bst modification and other code listings.

`apsrevM.bst` An mciteplus compatible version of REVTeX4's apsrev.bst [3].

`apsrmpM.bst` An mciteplus compatible version of REVTeX4's apsrmp.bst [3]. **Note: This is a sorting style. For it to work properly, you must set the "mcitetail" BibTeX database field for every tail entry as mentioned in section 3.1.**

`IEEEtranM.bst` An mciteplus compatible version of IEEEtran.bst [4].

`IEEEtranMN.bst` An mciteplus compatible version of IEEEtranN.bst (natbib compatible) [4].

2 Basic Usage

2.1 Bibstyle (.bst) Requirements

Mciteplus does require an mciteplus compatible bibstyle (.bst file). **Unfortunately, bibstyles designed for use with mcite will not work with mciteplus.** See section 3 for information on how to convert existing bibstyles for use with mciteplus.

2.2 Package Loading and Options

Mciteplus is invoked in the standard LaTeX way:

`\usepackage[`*options*`]{mciteplus}`

As mciteplus installs command wrappers over existing `\cite` commands, it should be loaded last, even after other packages that normally are loaded last such as hyperref [5].

Be aware that LaTeX may have to be run twice in order for the correct label width to be calculated and used after entries are collapsed. When this happens, mciteplus will issue the warning "Rerun to ensure correct mciteplus label max width/count".

2.2.1 Package Options

Valid options are:

chapterbibrootbib — This option must be specified when using the rootbib option/mode of the chapterbib package [6]. Mciteplus is not able to auto-detect the rootbib mode of chapterbib directly because when invoking that mode, the user must run LaTeX a second time without the rootbib option. For all of these LaTeX runs, keep this mciteplus option enabled (i.e., as long as there is, or is going to be, a jobname.bbl file under chapterbib). Beware of the possibility of an mciteplus status (i.e., head or tail) conflict in an overall bibliography if the same entry is cited differently in different parts of the document. See section 5.2 for more details on the use of chapterbib.

debug — Invoking this option will cause mciteplus to issue debugging information to the console whenever it processes: (1) a citation list; (2) an mciteplus aware bibliography; and (3) entries within the mciteplus

bibliography. This may be helpful to diagnose problems or to learn what tracking IDs, etc., mciteplus is using.

nohooks — This option will prevent mciteplus from automatically hooking into LaTeX's `\cite` internals and/or autodetecting and interfacing with external packages. The intended use is for advanced users who want to create their own mciteplus' `\cite` wrappers. When using this option, users will then have to manually define their own cite wrappers using the internal mciteplus engine (which is always available to users regardless of whether this option is selected or not). See section 4.2 for how to do this. Beware that this option can cause some packages to issue errors when used with mciteplus, especially those that alter the bibliography environments (e.g., citeref.sty, pageref.sty, etc.)

2.3 Cite Format

In the example given at the start of the introduction, the Glashow entry is known as the "head" entry and those of Salam and Weinberg are known as "tails". As with mcite, tails are declared within the `\cite` command immediately after their head entry by prefixing them with a *:

```
\cite{Glashow,*Salam,*Weinberg}
```

Head entries may or may not have tails. However, every tail must have a head entry. The tail entries for a group must all be declared (without duplication) when their head entry is first cited. After that, the head entry may be recited as often as desired. It is possible, though bad practice, to redeclare the tails (or a subset thereof) when reciting the head. This allows for the case of overall bibliographies in which the same citation group is defined in multiple local bibliographies, all the entries of which are later combined into a single overall bibliography. However, new tails cannot be later added to an existing citation group.

Thus, each of the following lines will generate an error as they are **invalid**:

```
\cite{*Salam} % missing head declaration

\cite{Glashow,*Salam} \cite{Glashow,*Salam,*Weinberg} % tails added after initial definition

\cite{Glashow,*Salam,*Salam} % duplicate tails in initial definition
```

Each of the following lines is **valid**:

```
\cite{Glashow,*Salam,*Weinberg,Smith,*Jones} % multiple groups OK

\cite{Glashow,*Salam,*Weinberg} \cite{Glashow} % head can be recited anytime

\cite{Glashow,*Salam,*Weinberg} \cite{Glashow,*Salam,*Weinberg} % restate previous definition

\cite{Glashow,*Salam,*Weinberg} \cite{Glashow,*Weinberg} % restate part of previous definition

\cite[page 580 of Glashow]{Glashow,*Salam} % optional argument is supported

\nocite{*} % BibTeX wildcard OK, but use with caution
```

The BibTeX "wildcard" entry "*" for `\nocite` is allowed. However, it is important to ensure that the entries BibTeX will automatically add to the bibliography from its databases will not get "in between" any head/tail groupings. This will not be a problem with unsorted bibstyles as long as the head/tail groups are cited prior to the invocation of the wildcard entry or as long as the head/tail group entries are listed together and in the correct order in the BibTeX databases. However, with sorting bibstyles extra care must be taken to keep the head/tail groups together in spite of the reordering process. See section 3.1 for more information.

By default, mciteplus will issue an error message if it encounters bibliography[2] entries that it has no record of. This will almost certainly be the case if the BibTeX wildcard entry is used. To disable the error messages and allow mciteplus to automatically assume that all unknown entries are heads, just issue the TeX conditional:

[2] Well, only those in a `mcitethebibliography` environment.

```
\mciteErrorOnUnknownfalse
```

before the bibliography it is to affect. Mciteplus will issue a warning message to the console as a reminder if it detects the use of the BibTeX wildcard entry.

2.4 Punctuation Controls

There are three kinds of punctuation/spacing mciteplus uses in the formatting of entries in the bibliography: (1) Middle punctuation which is used between entries within a collapsed group (typically "; "); (2) End punctuation which is used at the end of each entry group (typically "."); and (3) separation punctuation/spacing which is inserted before each head entry after the very first. Separation punctuation/spacing is usually not used (it is typically "`\relax`"), but it is provided for because some bibliography styles may require unusual spacing or a `\par` between the entries, but under mciteplus, bibstyles cannot directly insert such things as BibTeX does not know at run time which entries are heads and which are tails.

Mciteplus defines the package defaults of each as:[3]

```
\providecommand{\mcitedefaultmidpunct}{;\space}
\providecommand{\mcitedefaultendpunct}{.}
\providecommand{\mcitedefaultseppunct}{\relax}
```

However, both the bibstyle and the user can override the defaults with the user having the final say in the matter.

Bibstyles can override the package defaults via issuing the command:

```
\mciteSetBstMidEndSepPunct{middle punctuation}{end punctuation}{separation punctuation}
```

`\mciteSetBstMidEndSepPunct` must be issued at the end of the entry it is to affect (prior to the next `\bibitem`) and the values given will remain in effect until specified again.

The user can override the bibstyle by issuing the command:

```
\mciteSetMidEndSepPunct{middle punctuation}{end punctuation}{separation punctuation}
```

before the bibliography it is to affect. However, this poses a problem because the bibstyle may or may not specify end punctuation for each entry (e.g., such as when an entry ends with a URL or an abbreviated journal name). To provide for this, mciteplus provides a TeX conditional `\ifmciteBstWouldAddEndPunct`, that a bibstyle can set to true if and only if BibTeX would add end punctuation to the given entry. (See section 3 for how bibstyle designers should implement this feature.) If the bibstyle provides this feature, and it is recommended that they do, then users can take advantage of this conditional in their use of `\mciteSetMidEndSepPunct`:

```
\mciteSetMidEndSepPunct{;\space}{\ifmciteBstWouldAddEndPunct.\else\fi}{\relax}
```

and thus still avoid the "double period" problem under their own custom punctuation.

After issuing a user defined `\mciteSetMidEndSepPunct` for a bibliography, a user can restore control back to the bibstyle for later bibliographies by issuing:

```
\mciteSetMidEndSepPunct{\mcitebstmidpunct}{\mcitebstendpunct}{\mcitebstseppunct}
```

before the bibliography(ies) to be affected.

[3] `\providecommand` is used so that class files can declare their own mciteplus defaults.

2.5 Counters

Mciteplus provides two LaTeX counters, `mcitebibitemcount` and `mcitesubitemcount`, which track the number of head, and tail entries for each head, respectively. At each head entry, `mcitebibitemcount` is incremented and `mcitesubitemcount` is reset, by/to one. At each tail entry (or sublisted head entry under sublist modes "b", "f" and "h" as described in section 2.6.2), `mcitesubitemcount` is incremented by one after each subitem label is rendered. Be aware that `mcitebibitemcount` is *not* what is used to actually generate the entry labels in the bibliography as LaTeX handles this the same way it normally does (typically using it's own counter, `enumiv`).

These counters can be referenced by the user to generate sublist labels (section 2.6.2), to determine maximum label widths (section 2.7.2) as well as to control other counters (e.g., section 4.3).

2.5.1 Bibitem Counter Reset

By default, mciteplus resets `mcitebibitemcount` at the start of each bibliography. However, in some documents with multiple bibliographies, the entries are to be numbered consecutively throughout the document and, thus, the entry count should not reset at the start of each bibliography. For such cases, you can issue a:

```
\mciteResetBibitemCountfalse
```

before a bibliography to disable the counter reset (so that the maximum label widths and counts will be correct, see section 2.7). There is also a `\mciteResetBibitemCounttrue` which reenables the resetting of the counter.

2.6 Sublists

Sublisting within entry groups is supported by mciteplus. For example:

[1] S. L. Glashow, Nucl. Phys. **22**, 579 (1961); a) A. Salam, in *Elementary Particle Theory*, edited by N. Svartholm (Almquist and Wiksell, Stockholm, 1968), pp. 367–377; b) S. Weinberg, Phys. Rev. Lett. **19**, 1264 (1967).

The sublist labels (e.g., "a)", "b)", etc.) are based on the mciteplus provided LaTeX counter, `mcitesubitemcount`, which is reset at the start of each head.

2.6.1 Sublist Modes

The standard sublist mode "s" (as show above) is to begin numbering with the first tail.

Mciteplus can begin numbering with the head entries, but omit the sublabel for the heads via sublist mode "b":

[1] S. L. Glashow, Nucl. Phys. **22**, 579 (1961); b) A. Salam, in *Elementary Particle Theory*, edited by N. Svartholm (Almquist and Wiksell, Stockholm, 1968), pp. 367–377; c) S. Weinberg, Phys. Rev. Lett. **19**, 1264 (1967).

To sublabel the first entries of each group (heads) use the sublist mode "f":

[1] a) S. L. Glashow, Nucl. Phys. **22**, 579 (1961); b) A. Salam, in *Elementary Particle Theory*, edited by N. Svartholm (Almquist and Wiksell, Stockholm, 1968), pp. 367–377; c) S. Weinberg, Phys. Rev. Lett. **19**, 1264 (1967).

Entries without tails (single heads) are not sublisted under sublist mode "f". To enable that, use sublist mode "h".

As with punctuation, the sublist mode can be specified by both the bibstyle as well as the user with the latter overriding the former:

```
\mciteSetBstSublistMode{mode} % for use by bibstyles

\mciteSetSublistMode{mode} % for use by the user in the document
```

where *mode* is one of:

- **d** — "Default", do not alter whatever mode is currently in effect. For bibstyles, this is essentially a NOOP, but when invoked by the user, the bibstyle is allowed to alter the sublist mode.
- **n** — No sublist, do not sublist the entries.
- **s** — Sublist mode "s", sublist the tail entries.
- **b** — Sublist mode "b", sublist the tail entries, start the count with their head, but omit labeling the head.
- **f** — Sublist mode "f", sublist the tail entries including their head.
- **h** — Sublist mode "h", implies sublist mode "f", but sublist all entries, even heads without tails.

Unless specified otherwise, sublisting will not be done.

2.6.2 Sublist Label, Begin and End

Mciteplus allows bibstyles and users to specify the sublist label form used, the code that gets executed at the start of the sublist and the code that ends a sublist. Mciteplus defines the package defaults of each as:

```
\providecommand{\mcitedefaultsublistlabel}{\alph{mcitesubitemcount})\space}
\providecommand{\mcitedefaultsublistbegin}{\relax}
\providecommand{\mcitedefaultsublistend}{\relax}
```

The result of which is as shown in the previous examples. However, both the bibstyle and the user can override the defaults with the user having the final say in the matter. The sublist label/begin/end code can be specified by both the bibstyle as well as the user with the latter overriding the former:

```
\mciteSetBstSublistLabelBeginEnd{label}{begin}{end} % for use by bibstyles

\mciteSetSublistLabelBeginEnd{label}{begin}{end} % for use by the user in the document
```

This flexibility provides a way to use customized environments for the sublists:

```
\mciteSetSublistMode{s}
\mciteSetSublistLabelBeginEnd{\item[\arabic{mcitebibitemcount}.\alph{mcitesubitemcount})]}{\begin{e
numerate}}{\end{enumerate}}
```

which yields:

[1] S. L. Glashow, Nucl. Phys. **22**, 579 (1961);

1.a) A. Salam, in *Elementary Particle Theory*, edited by N. Svartholm (Almquist and Wiksell, Stockholm, 1968), pp. 367–377;

1.b) S. Weinberg, Phys. Rev. Lett. **19**, 1264 (1967).

After issuing a user defined `\mciteSetSublistLabelBeginEnd` for a bibliography, a user can restore control back to the bibstyle for later bibliographies by issuing:

```
\mciteSetSublistLabelBeginEnd{\mcitebstsublistlabel}{\mcitebstsublistbegin}{\mcitebstsublistend}
```

before the bibliography(ies) to be affected.

2.6.3 Referencing Sublist Labels

It is not possible to reference the tails using \cite:

```
\cite{Salam}  % invalid, tail redeclared as a head
\cite{*Salam} % invalid, tail cannot be referenced
```

and even if it were allowed, it would likely cause problems with packages that compress and sort citation numbers.

However, mciteplus does provide two commands that work much like \ref and \pageref that can be used to reference the sublabeled entries and the page numbers they appear on:

```
\mciteSubRef[track ID]{cite key}
\mciteSubPageRef[track ID]{cite key}
```

where *cite key* is a (single) citation key and *track ID* is the tracking ID (section 4.1) string which mciteplus uses to uniquely identify bibliographies. Please note that these commands will not work for entries that are not sublabeled (e.g., heads with tails if the sublist mode is not "f" or "h"). The tracking IDs used by each bibliography can be displayed by loading mciteplus with the "debug" package option (section 2.2.1). For virtually all single bibliography documents, the default tracking ID should work fine. However, when multiple bibliographies are in use, the tracking ID may have to be specified so that mciteplus will know which bibliography entry to reference. Some packages such as chapterbib, may even put the same citation in multiple bibliographies.

For the part/chapter bibliographies of chapterbib.sty, there is no need to specify a tracking ID as mciteplus will automatically use the correct one within each part. The tracking ID of the duplicate bibliographies of chapterbib (produced by chapterbib's "duplicate" package option) is given by "chapterbib*inputfile*.bbl", where "*inputfile*" is the include file the given citation occurred in. For the rootbib bibliography (produced by chapterbib's "rootbib" package option), the tracking ID is of the form "chapterbib*jobname*", where "*jobname*" is the name of the main .tex document file (without the .tex suffix).

For multibib.sty, the tracking ID is given by "multibib*secname*", where "*secname*" is the multibib bibliography section name (as declared via \newcites).

For multibbl.sty, the tracking ID is given by "multibbl*secname*", where "*secname*" is the multibbl bibliography section name.

For bibunits.sty, mciteplus will automatically use the correct tracking ID within each bibunit, but to reference an entry outside of a bibunit, the tracking ID of each bibunit is given by "bibunits*unitname*", where "*unitname*" is the name of the auxiliary file of the desired bibunit without the .aux suffix (e.g., "bu1", "bu2", etc.).

The form of the replacement text of \mciteSubRef is defined by mciteplus as:

```
\providecommand{\mcitesubrefform}{\arabic{mcitebibitemcount}.\alph{mcitesubitemcount}}
```

which can be redefined (via \renewcommand) by the user as desired.

2.6.4 Bibitem Argument Macros

When creating sublist labels (or the maximum width forms discussed in section 2.7.2), the user may wish to have access to the argument(s) of the \bibitem of the entries. Mciteplus provides these as macros. The required argument of the current \bibitem, which is usually the citation key, is provided as \mciteBibitemArgI. The optional argument of the current \bibitem, which, if present, is provided as \mciteBibitemOptArgI. There is also the TEX conditional \ifmciteBibitemOptArgI, which will evaluate true if the optional argument to \bibitem is present.

Lastly, there are \mciteCurheadBibitemArgI, \mciteCurheadBibitemOptArgI and \ifmciteCurheadBibitemOptArgI which contain the same argument information, but for the head of the current entry group.

2.7 Label Widths

Note: Problems with incorrect label widths will usually manifest themselves as incorrect label spacing and/or bibliography entry text that is not properly aligned, but often will not otherwise generate warning messages or errors.

2.7.1 The Bibliography Sample Label

With most bibstyles, BibTeX produces what is known as a "sample label," which is the widest label used by the bibliography entries. This sample label is recorded as the argument to `\thebibliography` when BibTeX creates the .bbl bibliography file so as to provide a way for the `\thebibliography` environment to know in advance how much space to reserve for the labels.

For example, BibTeX may produce something like this in the .bbl file for a bibliography with ten entries:[4]

```
\begin{thebibliography}{10}
```

Mciteplus provides users access to the original BibTeX-produced sample label as the macro `\mciteorgbibsamplelabel`.

A problem arises with the sample label system when mciteplus combines multiple entries in that the BibTeX generated sample label in the .bbl file will not be correct after the tail entries are no longer counted as actual entries. With the original mcite package, this would often result in incorrect label spacing. Mciteplus addresses this problem by actively tracking the maximum width label used in each bibliography.

The sample label form that mciteplus forwards to `thebibliography` is defined as:

```
\def\mcitebibsamplelabel{\rule{\mcitemaxwidthbibitem sp}{0.2pt}}
```

which is simply a rule with a width equal to the widest label as determined by the maximum width system discussed in section 2.7.2. If necessary, this can be redefined by the user via `\renewcommand`.

Most class files will not be bothered by the fact that a rule rather than actual digits is being used for the sample label. However, if this becomes a problem, something like:

```
\renewcommand{\mcitebibsamplelabel}{\mcitemaxcountbibitem}
```

can be tried where `\mcitemaxcountbibitem` will contain the number of the last entry in the current bibliography as discussed in section 2.7.3.

The `thebibliography` environments of some LaTeX classes (e.g., REVTeX) perform their own label width calculations and ignore the sample label entirely making the issue a moot point. However, the most common LaTeX classes do depend on the sample label being accurate.

2.7.2 Maximum Widths

The problem of knowing the widest of a group of labels in advance of their actual rendering[5] is commonly encountered in LaTeX (e.g., lists, section numbering within the table of contents, etc.) and is an issue that, in this author's opinion, has not been dealt with adequately by the base LaTeX system.[6] With regard to mciteplus bibliographies, not only is the width of the head entries (i.e., the sample label) an issue, but potentially so are the sublist (tail) labels if they are rendered in enumerated list form rather than inline.

[4] Some bibstyles use a number equal to the number of entries, others use a form such as "1X", where "X" is a number of zeros such that the sample label length matches that of the number of entries (e.g., "10" if there are 10–99 entries). With most fonts, the numerical digits all have the same width so "10" has the same width as "99".

[5] Obviously, solving this problem requires more than one LaTeX pass as well as the use of the auxiliary file to store information about the widest label which will be read during the second pass.

[6] Packages such as Scott Pakin's eqparbox.sty [7] are most welcome solutions to this type of problem.

To assist the user in determining the required width of labels, mciteplus provides its own maximum width and count tracking facilities. Both the bibitem (main entry) and subitem (sublist entry) label widths can tracked independently of each other.

In order to measure widths, mciteplus must know what exactly it is to measure. This is specified by macros that contain the "width forms" which are simply "text" whose width will be measured for each bibitem or subitem. The package defaults of each are defined as:

```
\providecommand{\mcitedefaultmaxwidthbibitemform}{\arabic{mcitebibitemcount}}
\providecommand{\mcitedefaultmaxwidthsubitemform}{\alph{mcitesubitemcount})}
```

There are also the initial width forms that will be used only during the very first LaTeX run before the maximum label widths are actually known (i.e., when the auxiliary file is absent):

```
\providecommand{\mcitedefaultmaxwidthbibitemforminit}{\mciteorgbibsamplelabel}
\providecommand{\mcitedefaultmaxwidthsubitemforminit}{a)}
```

The initial width forms will be evaluated with the bibitem and subitem counters both set to one. The purpose of the initial width forms is just to provide a rough guess as to the label widths to try to avoid large formatting changes or errors within label formatting code that cannot deal with unknown, or zero, widths.

As with the punctuation and sublist controls, both the bibstyle and the user can specify the width forms with the latter overriding the former:

```
\mciteSetBstMaxWidthForm[init]{type}{form} % for use by bibstyles

\mciteSetMaxWidthForm[init]{type}{form} % for use by the user in the document
```

where *type* is "`bibitem`" or "`subitem`". The optional argument *init* is used to specify the initial width form that will be used. If it is not specified, the current initial width form in use will not be altered. The code specified in the forms should be self contained (i.e., not have commands which require an external environment such as `\item`) as well as allow reevaluation at any time (e.g., should not do things such as alter counters). Note that these restrictions do not apply to the actual sublist code (section 2.6.2) and is one reason why mciteplus distinguishes between that which measures the width of the a label (discussed in this section) and that which actually forms the label (section 2.6.2). If desired, the current width forms (see below) can be used within the actual sublist code.

After issuing a user defined `\mciteSetMaxWidthForm` for a bibliography, a user can restore control back to the bibstyle for later bibliographies by issuing:

```
\mciteSetMaxWidthForm[\mcitebstmaxwidthbibitemforminit]{bibitem}{\mcitebstmaxwidthbibitemform}
\mciteSetMaxWidthForm[\mcitebstmaxwidthsubitemforminit]{subitem}{\mcitebstmaxwidthsubitemform}
```

before the bibliography(ies) to be affected.

Please note that for bibstyles, there is no use in specifying an initial width form via `\mciteSetBstMaxWidthForm[]` after the bibliography begins because, by that point, it is too late as the sample label has already been forwarded. However, this is not a problem because by default, the initial width form is the original sample label as given by the bibstyle.[7] This is not an issue for the normal bibitem width form or the subitem width forms as they are not evaluated until the first `\bibitem`.

The user has access to the currently used bibitem and subitem width forms and their maximum widths as the macros:

```
\mcitemaxwidthbibitemform % current bibitem width form
\mcitemaxwidthsubitemform % current subitem width form
\mcitemaxwidthbibitem % the maximum width of the bibitem form
\mcitemaxwidthsubitem % the maximum width of the subitem form
```

[7] Another possible approach would be to specify the bibitem width form before the bibliography begins, possibly surrounding the entire environment in a group to keep the changes local.

The maximum widths are macros containing an integer which is the width in TEX scaled points (sp).[8] The widths are not length commands (e.g., a dimension or skip register), but their values can be assigned to a length command by appending a "sp" to the length specification:

```
\newlength{\MYlength}
\setlength{\MYlength}{\mcitemaxwidthsubitem sp}
```

2.7.3 Maximum Counts

Also available are the macros:

```
\mcitemaxcountbibitem
\mcitemaxcountsubitem
```

which contain the value of the highest count of the bibitem or subitem[9] counters, respectively, in the current bibliography. The maxcount macros are not counters, but they can be used to set them, among other things. There are no "forms" associated with the maxcounts and their initial values (during the very first LATEX run in the absence of an auxiliary file) will be zero. So, any code that references them should be able to handle (without error) the case of zero.

If, at the end of the bibliography, the correct maximum widths and counts do not match what mciteplus used at the start of the run, mciteplus will issue a package warning to request that LATEX be rerun.

2.7.4 Non-Numerical Labels

The maxwidth forms may have to be changed for some non-numerical bibstyles. For example, under alpha.bst [8], the entry labels are carried within the optional arguments of the `bibitem`, in which case

```
\mciteSetMaxWidthForm{bibitem}{\mciteBibitemOptArgI}
```

would be appropriate when using such a bibstyle under mciteplus. Actually, it is better to do such setup within the bibliography code of the bibstyle (.bst) so that the maxwidth form will be correct by default without any effort on the part of the user:

```
\mciteSetBstMaxWidthForm{bibitem}{\mciteBibitemOptArgI}
```

This should not be an issue with natbib compatible bibstyles, even under the author-date mode, as natbib automatically does its own label generation and management.

2.8 Example of Use

Perhaps the maxwidth system is easiest to understand from an example:

```
\mciteSetSublistMode{f}% start numbering with the first item
\mciteSetMaxWidthForm{subitem}{\scriptsize\textbf{\roman{mcitesubitemcount})}}% small, bold, roman

\newcommand{\MYlistsetup}{\relax
\setlength{\labelwidth}{\mcitemaxwidthsubitem sp}% reserve room for widest label
\setlength{\labelsep}{5pt}% 5pt label/text separation
\setlength{\itemindent}{0pt}\setlength{\leftmargin}{\labelwidth}% text to left of label
\addtolength{\leftmargin}{\labelsep}}% and separation space

\mciteSetSublistLabelBeginEnd{\item}{\begin{list}{\hfill\mcitemaxwidthsubitemform}{\MYlistsetup}}{
\end{list}}
```

[8] In TEX, a scaled point is the smallest dimensional unit available and is equal to $\frac{1}{65536}$ of a point.

[9] For subitems, it is the highest count that occurred *during* the generation of the sublist labels. Because the subitem counter is advanced by one *after* each sublist label is rendered, the actual counter itself does reach one count higher than its maxcount, but that value was not actually used.

which produces a sublist via a customized list with small, bold and right aligned roman numerals as labels and with block indented text. Note that in the list setup code, we need to know the amount of room to reserve for the labels. Mciteplus provides this as `\mcitemaxwidthsubitem` (to which the units "sp" must be appended when assigning it to a length command). The maxwidth form itself, `\mcitemaxwidthsubitemform` was used in the specification of the labels within the list setup, but its definition (as used within `\mciteSetMaxWidthForm`) could have been respecified in the list setup code if desired. Once the list is setup, each sublabel is generated by a simple call to `\item`. So, this example shows a case in which the code actually used to produced the sublabels at run time (`\item`) is not (directly) the same as that code which is used to measure the sublabel widths. Indeed, as it is not self contained, `\item` cannot even be used in the definition of a maxwidth form.

3 Bibstyle (.bst) Modification

Original mcite bibstyles cannot be used with mciteplus. They will have to be remodified as described below.

To modify a standard bibstyle (.bst) for use with mciteplus (unsrt.bst is used as an example here), find the instances of `\begin{thebibliography}` and `\end{thebibliography}` in the `begin.bib` and `end.bib` functions and change them to use `mcitethebibliography`. It may also be a good idea to test for the existence of `mcitethebibliography` in case the user ever forgets to load mciteplus.sty.

Note: Many of the functions, or parts of functions, shown below are listed in the included file `mciteplus_code.txt`, which can be used to cut and paste from. Beware of slight code differences between the different bibstyles. For example, some use `longest.label` like this example, but others may use something else like `number.label int.to.str$` and/or may have extra code after the `\begin{mcitethebibliography}` line. Leave these other things as they are.

Thus,

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

.
.

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

becomes (change the string "unsrtM.bst" to the name of your new .bst file):

```
FUNCTION {begin.bib}
{ preamble$ empty$
    'skip$
    { preamble$ write$ newline$ }
  if$
  "\ifx\mcitethebibliography\mciteundefinedmacro"
  write$ newline$
  "\PackageError{unsrtM.bst}{mciteplus.sty has not been loaded}"
  write$ newline$
  "{This bibstyle requires the use of the mciteplus package.}\fi"
  write$ newline$  "\begin{mcitethebibliography}{"  longest.label  * "}" * write$ newline$
```

```
}

.
.

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

You may also want to declare the sublist mode via a `\mciteSetBstSublistMode`, as mentioned in section 2.6, just after the `mcitethebibliography` environment begins, that will be used if the user does not otherwise specify a sublist mode, as well as other bibliography setup commands.

For example:

```
FUNCTION {begin.bib}
{ preamble$ empty$
    'skip$
    { preamble$ write$ newline$ }
  if$
  "\ifx\mcitethebibliography\mciteundefinedmacro"
  write$ newline$
  "\PackageError{unsrtM.bst}{mciteplus.sty has not been loaded}"
  write$ newline$
  "{This bibstyle requires the use of the mciteplus package.}\fi"
  write$ newline$
  "\begin{mcitethebibliography}{"  longest.label  * "}" * write$ newline$
  "\mciteSetBstSublistMode{b}"
  write$ newline$
  "\mciteSetBstMaxWidthForm{subitem}{\alph{mcitesubitemcount})}"
  write$ newline$
  "\mciteSetBstSublistLabelBeginEnd{\mcitemaxwidthsubitemform\space}"
  write$ newline$
  "{\relax}{\relax}"
  write$ newline$
}
```

will specify a `mcitethebibliography` that defaults to using the sublist "b" mode and enumerates the subentries using letters. The middle and end punctuation was not be specified via `\mciteSetBstMidEndSepPunct` and so the package defaults will be used. In this example, the subitem maxwidth form is not so important as the labels are rendered inline, but other types of lists may require it.

Next, find the `fin.entry` function:

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

and change it to (which you can cut and paste from `mciteplus_code.txt`):

```
% mciteplus fin.entry
%
% pushes true (1), if add.period$ would add a period to the string on the stack
% pushes false (0), otherwise
% Uses text.length$ to avoid full string comparison and two copies of string.
% Requires one copy of string on stack.
```

```
INTEGERS {would.add.period.textlen}
FUNCTION {would.add.period}
{ duplicate$
  add.period$
  text.length$
  'would.add.period.textlen :=
  duplicate$
  text.length$
  would.add.period.textlen =
    { #0 }
    { #1 }
  if$
}

FUNCTION {fin.entry}
{ would.add.period
    { "\relax" * write$ newline$
      "\mciteBstWouldAddEndPuncttrue" write$ newline$
      "\mciteSetBstMidEndSepPunct{\mcitedefaultmidpunct}"
      write$ newline$
      "{\mcitedefaultendpunct}{\mcitedefaultseppunct}\relax"
    }
    { "\relax" * write$ newline$
      "\mciteBstWouldAddEndPunctfalse" write$ newline$
      "\mciteSetBstMidEndSepPunct{\mcitedefaultmidpunct}"
      write$ newline$
      "{}{\mcitedefaultseppunct}\relax"
    }
  if$
  write$
  newline$
  "\EndOfBibitem" write$
}
% end mciteplus fin.entry
```

The `\mciteSetBstMidEndSepPunct` line is broken into two lines (after the first argument) to prevent BibTeX from breaking what would be a long single line (over 80 columns) at an unacceptable place.

By providing a way for the .bst to tell mciteplus not to use an end period if BibTeX's `add.period$` command would not, we avoid the double end period problem. The `\mciteBstWouldAddEndPuncttrue/false` flag does not in itself do anything. However, if set appropriately by the .bst, it can be employed by the user in the definition custom punctuation via `\mciteSetMidEndSepPunct`.

Other .bst files may hardcode values other than the defaults and/or use more complex conditions as is needed for that particular bibstyle.

The `\EndOfBibitem` is defined by mciteplus.sty as a macro containing `\relax`. Although it does nothing at present, it's purpose is to make it easier to identify the end of each bibliography entry for the purposes of inspection or parsing.

3.1 Sorting Bibstyles

Perhaps surprisingly, it is possible to use mciteplus with sorting bibstyles. The difficulty is that BibTeX must keep an entry group together, in citation order with the tails immediately following their respective head, despite sorting the entries. This is possible with the use of special bibstyle code if the user is willing and able to declare all the tails via a special entry field "mcitetail", which should be set to "yes" for each tail,

in the BibTeX database.[10] Entries without this field, or that have it set to "no", will be sorted normally (as head entries). Of course, bibstyles that do not recognize the "mcitetail" field will silently ignore it.

Thus, for the example used in the introduction, the database entries might look like:

```
@article{Glashow,
  author        = "Sheldon Lee Glashow",
  title         = "Partial-symmetries of Weak Interactions",
  journal       = "Nucl. Phys.",
  volume        = "22",
  number        = "4",
  month         = feb,
  year          = "1961",
  pages         = "579-588"
}

@incollection{Salam,
  author        = "Abdus Salam",
  editor        = "Nils Svartholm",
  title         = "Weak and Electromagnetic Interactions",
  booktitle     = "Elementary Particle Theory",
  publisher     = "Almquist and Wiksell",
  address       = "Stockholm",
  year          = "1968",
  pages         = "367-377",
  mcitetail     = "yes"
}

@article{Weinberg,
  author        = "Steven Weinberg",
  title         = "A Model of Leptons",
  journal       = "Phys. Rev. Lett.",
  volume        = "19",
  number        = "21",
  month         = nov,
  year          = "1967",
  pages         = "1264-1266",
  mcitetail     = "yes"
}
```

The basic idea is for BibTeX to use the sorting key of the head, but with an appended tail count, as the sorting key for all the tails of that head. If the bibstyle properly generates unique sorting keys, then the entries in each group will be kept together and in order despite the sorting process. For example, apsrmpM.bst uses the following sort keys for the above three entries:

```
glashow  s l    1961    partial symmetries of weak interactions
glashow  s l    1961    partial symmetries of weak interactions__0000000001
glashow  s l    1961    partial symmetries of weak interactions__0000000002
```

The sorting modification to the bibstyle consists of three parts which must be done in addition to the mciteplus compatibility modifications previously described.

First of all, add an "mcitetail" field to the list of entry fields (usually located near the beginning of the bibstyle):

[10] Thus, we have to specify the tails twice — once for LaTeX and once for BibTeX because there is no other way to get the information to BibTeX. This limitation helped to inspire LaTeX-based bibliography processing as is done with the amsrefs [9] and biblatex [10] packages.

```
ENTRY
  { address
    archive
    author
    booktitle
    .
    .
    key
    mcitetail
    month
    .
    .
```

Then, add the following code (which you can cut and paste from `mciteplus_code.txt`) somewhere near the start of the bibstyle code (after the first "STRINGS" definition(s) is fine):

```
% mciteplus mcitetail field and sort key adjust support
INTEGERS {mcitetailcnt is.mcitetail}
STRINGS  {mciteheadsortkey}
% convert the strings "yes" or "no" to #1 or #0 respectively
FUNCTION {mciteplus.yes.no.to.int}
{ "l" change.case$ duplicate$
    "yes" =
    { pop$ #1 }
    { duplicate$ "no" =
        { pop$ #0 }
        { "unknown boolean " quote$ * swap$ * quote$ *
          " in " * cite$ * warning$
          #0
        }
      if$
    }
  if$
}
FUNCTION {mciteplustail.adj.sort.key}
{ mcitetail
  empty$
    { #0 'is.mcitetail :=  }
    { mcitetail
      mciteplus.yes.no.to.int
      'is.mcitetail :=
    }
  if$
  is.mcitetail
    { #1 mcitetailcnt
      'mcitetailcnt :=
      mciteheadsortkey
      "__" *
      "000000000"
      mcitetailcnt
      int.to.str$
      *
      #-1 #10 substring$
      *
      'sort.key$ :=
    }
    { #0 'mcitetailcnt :=
      sort.key$ 'mciteheadsortkey :=
```

```
    }
  if$
}
% END mciteplus mcitetail field support
```

Lastly, you have to locate the place(s) where the bibstyle makes the (final) assignment to `sort.key$` (typically of the form `'sort.key$ :=`) before sorting is performed and add a call to `mciteplustail.adj.sort.key` (which will modify the sort keys of the tails as needed to keep the groups together) just after it. With apsrmpM.bst, this is at the end of the `FUNCTION{presort}`:

```
.
.
  'sort.key$ :=
  mciteplustail.adj.sort.key
}
```

as well as at the end of `FUNCTION{bib.sort.order}`.

4 Advanced Usage

The vast majority of users will not need to know the details of the internal operation of mciteplus. However, the following information may be of use to those who are interested in it or who need to interface other packages to mciteplus.

Mciteplus consists of two main parts: (1) a part that intercepts and records the status of entries the user has cited; and (2) a part that acts on this information when formatting the entries in the bibliography.

Upon the start of the `\begin{document}`, mciteplus stores the original `\cite` and `\nocite` as `\mciteOrgcite` and `\mciteOrgnocite`, respectively, and replaces the originals with its own versions. The mciteplus versions of the citation commands: (1) process the citation list, recording which entries are heads and which are tails; (2) writes out the citation list to the auxiliary file, much like `\nocite` does, to ensure that all the given entries will appear in the bibliography as well as being in the correct order; and (3) forwards the list of heads in the citation list to the original cite command. Thus, LaTeX remains unaware that the tail entries were ever cited.

The second part of mciteplus operates inside the `mcitethebibliography` environment where mciteplus: (1) intercepts and adjusts the bibliography sample label as needed; (2) forwards to the original `thebibliography`; and (3) places a wrapper around `\bibitem`, storing the original `\bibitem` as `\mciteOrgbibitem`. For head entries, the mciteplus `\bibitem` behaves in fairly normal fashion and forwards to the original `\bibitem`. However, for tail entries, the mciteplus `\bibitem`, does not forward them to the original `\bibitem`. Thus, the tail entries become part of the previous entry. To LaTeX, it is as if the `\bibitem` of the tails never happened.

4.1 Tracking ID and Aux. Files

Things get more complicated when a document has more than one bibliography. To support multiple bibliographies, mciteplus uses a "tracking ID" which is simply an identification string which is unique to each bibliography. Thus, the same citation key can be used in different ways in different bibliographies (e.g., serving as a head in one and a tail in another) without conflict because each instance will be considered different as their tracking IDs are different.[11]

The standard tracking ID is simply the string "main" which is carried as:

```
\def\mcitetrackID{main}
\def\mcitebibtrackID{main}
```

[11] Of course, a "status conflict" is still possible for "overall" bibliographies.

the former of which is used by the mciteplus citation commands and the latter of which is used by the `mcitethebibliography` environment. However, these definitions may be automatically altered depending on the specific needs of other LaTeX packages that have been loaded (section 5). Some packages that provide support for multiple bibliographies require that the tracking ID be closely coupled with the individual citation commands. In such cases, mciteplus may not even refer to `\mcitetrackID` or `\mcitebibtrackID`.

For the normal part/chapter bibliographies of chapterbib.sty, the tracking ID form "chapterbib*inputfile*" is used, where "*inputfile*" is the include file (without the .tex suffix) the given citation occurred in. The tracking ID of the duplicate bibliographies of chapterbib (produced by chapterbib's "duplicate" package option) is given by "chapterbib*inputfile*.bbl". For the rootbib bibliography (produced by chapterbib's "rootbib" package option), the tracking ID is of the form "chapterbib*jobname*", where "*jobname*" is the name of the main .tex document file (without the .tex suffix).

For multibib.sty, the tracking ID is given by "multibib*secname*", where "*secname*" is the multibib bibliography section name (as declared via `\newcites`).

For multibbl.sty, the tracking ID is given by "multibbl*secname*", where "*secname*" is the multibbl bibliography section name.

For bibunits.sty, the tracking ID of each bibunit is given by "bibunits*unitname*", where "*unitname*" is the name of the auxiliary file of the desired bibunit without the .aux suffix (e.g., "bu1", "bu2", etc.). Outside of the bibunits, where the global bibliography is in effect, the tracking ID of "main" is used.

Likewise, mciteplus must know which auxiliary file(s) to write its citation lists and bibliography maximum widths and counts to. The standard auxiliary file handles, which are simply integers that represent files TeX has opened, are defined as:

```
\def\mciteauxout{\@auxout}
\def\mcitebibauxout{\@mainaux}
```

the former of which is used by the mciteplus citation commands and the latter of which is used by the `mcite thebibliography` environment. As with the tracking ID, these may be altered to support other packages that have been loaded. Under some packages, such as chapterbib, LaTeX may reuse the same file handle number during the course of processing the document. Thus, a given file handle number may not refer to the same file in different parts of the document.

Note that mciteplus writes all of its maximum width and count information to the main auxiliary file even though multiple auxiliary files may be in use because it is not certain that the other auxiliary files will be read in during LaTeX runs. As the tracking ID is written along with the maximum width and count information, there will be no confusion as to which bibliography the information pertains to.

4.2 Custom Command Wrappers

Mciteplus provides two rather complex commands that allow users to define their own mciteplus citation command wrappers:

```
\mciteCiteA*{aux out}{track ID}{prehandler}{posthandler}{fwd}*[opt1][opt2]{cite list}
\mciteCiteB*{aux out}{track ID}{prehandler}{posthandler}{fwd}*[opt1][opt2]{sec ID}{cite list}
```

Both of these are "robust" and can safely be called within footnotes, etc. The difference between the two is that the "B" form supports citation commands that use two arguments, as is the case with the `\cite` of multibbl.sty. `\mciteCiteA/B` acquires all of the arguments listed above (the "*" and "[]" are optional arguments), but those that appear after the *fwd* argument are forwarded (possibly with modification) to the command named in the *fwd* argument. The arguments before *fwd* are only seen and used by `\mciteCiteA/B`.

The meaning of each of the options from left to right is as follows:

***** The star forms (e.g., `\mciteCiteA/B*`) disable automatic internal processing (via `\mciteDoList` as discussed below). The presence of the star form is indicated by the TeX conditional `\ifmciteMacroStarForm`.

aux out The auxiliary file handle number to be used when writing out the citation list. Use of the special string "noauxwrite" will disable auxiliary writes.[12] This argument is stored in the macro `\mciteCiteAuxArg`.

track ID The tracking ID to be used. This argument is stored in the macro `\mciteCiteTrackArg`.

prehandler The user's prehandler code which will be executed before internal `\mciteDoList` processing. This argument is stored in the macro `\mciteCitePrehandlerArg`. The purpose of the prehandler code is to provide the user a way to alter the arguments as needed before further processing.

posthandler The user's posthandler code which will be executed after internal `\mciteDoList` processing. This argument is stored in the macro `\mciteCitePosthandlerArg`.

fwd The citation command the later arguments will be forwarded to. This argument is stored in the macro `\mciteCiteFwdArg`.

***** The star for the star form of the forward command. The presence of the star form is indicated by the TEX conditional `\ifmciteCiteStarFwdArg`.

opt1 The first optional argument of the forwarded citation command. The presence of this optional argument is indicated by the TEX conditional `\ifmciteMacroOptArgI`. This argument, if present, is stored in the macro `\mciteCiteOptArgI`.

opt2 The second optional argument of the forwarded citation command. The presence of this optional argument is indicated by the TEX conditional `\ifmciteMacroOptArgII`. This argument, if present, is stored in the macro `\mciteCiteOptArgII`.

sec ID Only used for `\mciteCiteB`, this is the first argument of two used by the forwarded citation command. Under multibbl.sty, this contains the bibliography section name. This argument is stored in the macro `\mciteCiteSecIDArg`.

cite list This is the citation list and is stored in the macro `\mciteCiteListArg`. It is important to know that `\mciteCiteListArg` is *not* forwarded as-is, but rather only after the tails are removed. The forwarded citation list is the macro `\mciteFwdCiteListArg`.

4.2.1 The MciteDoList Engine

After acquiring its arguments, `\mciteCiteA/B` executes the prehandler code. This is to allow the prehandler code to alter the arguments as needed, specifically if the contents of one argument need to be based on another. For example, under multibbl.sty, the tracking ID has to be based in part on `\mciteCiteSecIDArg`.

After the prehandler, the `\mciteDoList` command, which forms the core engine of mciteplus, is executed:

`\mciteDoList{`*aux out*`}{`*track ID*`}{`*cite list*`}`

as:

```
\mciteDoList{\mciteCiteAuxArg}{\mciteCiteTrackArg}{\mciteCiteListArg}
```

`\mciteDoList` records the status of each of the entries, writes the citations to the auxiliary file, and then creates `\mciteheadlist` which contains only the head entries. A copy of `\mciteheadlist` called `\mciteFwdCiteListArg` is made. This will be the actual citation list that is forwarded to the *fwd* command.

Next, `\mciteExtraDoLists` is called. This command is normally, just `\relax`, but it can be used to create additional sets of entry status, such as when duplicate/global bibliographies are being used.

Then, the user's posthandler code is executed.

Finally, the forward command is executed with the (rightmost) `\mciteCiteA/B` arguments forwarded to it, but using `\mciteFwdCiteListArg` for the citation list.

As a practical example, a standard `\cite` wrapper can be created via:

[12] As will the traditional TEX conditional, which is defined and used by LATEX, `\if@filesw`, if it is false.

```
\newcommand{\MYcite}{\mciteCiteA{\mciteauxout}{\mcitetrackID}{\relax}{\relax}{\mciteOrgcite}}
```

If `\MYcite` is invoked as:

```
\MYcite{Glashow,*Salam,*Weinberg}
```

mciteplus will do a:

```
\mciteCiteA{\mciteCiteAuxArg}{\mciteCiteTrackArg}{\relax}{\relax}{\mciteOrgcite}{\mciteCiteListArg}
```

which is the same as:

```
\mciteCiteA{\mciteauxout}{\mcitetrackID}{\relax}{\relax}{\mciteOrgcite}{Glashow,*Salam,*Weinberg}
```

which in turn would call:

```
\mciteDoList{\mciteauxout}{\mcitetrackID}{Glashow,*Salam,*Weinberg}
```

which will create a `\mciteheadlist` and `\mciteFwdCiteListArg` containing "Glashow", which will then be forwarded to the original LaTeX `\cite`:

```
\mciteOrgcite{\mciteFwdCiteListArg}
```

which is effectively the same as:

```
\mciteOrgcite{Glashow}
```

Mciteplus carefully builds up the forwarded arguments using token lists so that the forward cite command isn't even aware that it has been wrapped within an earlier command. Thus, even cite.sty's automatic punctuation and spacing adjustments will work as normal.

Note that command structures that result in "double calls" to the miteplus engine (i.e., a cite command goes through mciteplus and then forwards to another cite command that also goes through mciteplus) although undesirable, will still work as intended without error.

4.3 Mcitethebibliography Hooks

Mciteplus supplies the macros `\mciteBIBdecl` and `\mciteBIBenddecl` which are executed at the beginning and ending of the `mcitethebibliography` environment, respectively, before the the normal `thebibliography` environment is begun or ended. By default they are both defined as macros containing only `\relax`, but a user may redefine them as needed to alter the `mcitethebibliography` environment setup.

There is also a `\mcitefwdBIBdecl` which is executed just after the original `thebibliography` has been started (i.e., forwarded) by `mcitethebibliography`. This can be used to adjust the setup of `thebibliography` before the first bibliography entry. One potential application is to sync the `thebibliography` counter to `mcitebibitemcount`. This trick can be used to obtain continuous numbering of the references under multibbl.sty even though that package does not normally provide this feature:

```
\documentclass{article}
\usepackage{multibbl}
\usepackage{mciteplus}
\mciteResetBibitemCountfalse
\renewcommand{\mcitefwdBIBdecl}{\setcounter{enumiv}{\value{mcitebibitemcount}}}
\begin{document}
.
.
```

5 Use With External Packages

Unless the "nohooks" option is invoked, mciteplus will automatically reconfigure itself and hook into the compatible packages it detects. Mciteplus should be loaded after other packages so that it can detect the other packages and to ensure that the definitions of the system cite commands have been finalized before mciteplus installs its wrappers at the beginning of the document.

It is not possible to test or list all of the possible interactions among all the various packages. Remember, just because the packages below are compatible with mciteplus, does not mean they all are compatible with each other, even in the absence of mciteplus. Here are some notes regarding the compatibility and use of mciteplus with some of the more common LaTeX bibliography related packages.

5.1 Bibunits

Mciteplus is fully compatible with Thorsten Hansen's bibunits package [11]. When using the global bibliography via the "globalcitecopy" and/or the star form of `\cite`, beware of the possibility of an mciteplus status (i.e., head or tail) conflict if the same entry is cited differently in different parts of the document.

5.2 Chapterbib

Mciteplus is fully compatible with Donald Arseneau's chapterbib package [6] including the use of the chapterbib "duplicate" and "rootbib" package options. **When using the "rootbib" option, be sure and enable mciteplus' "chapterbibrootbib" option.** Mciteplus is not able to auto-detect the rootbib mode of chapterbib directly because when invoking that mode, the user must run LaTeX a second time without the rootbib option. For all of these LaTeX runs, keep this mciteplus option enabled (i.e., as long as there is, or is going to be, a jobname.bbl file under chapterbib). When using the rootbib option, beware of the possibility of an mciteplus status (i.e., head or tail) conflict if the same entry is cited differently in different parts of the document.

5.3 Cite

Mciteplus is fully compatible with Donald Arseneau's cite package [12], including cite's `\citen`, `\citenum` and `\citeonline` variants.

5.4 Citeref

Mciteplus will work with Björn Briel and Uni Oldenburg's citeref package [13].

5.5 Drftcite

Mciteplus is fully compatible with Donald Arseneau's drftcite package [14], including drftcite's `\citen` variant. To get block aligned entry text in the bibliography, set the bibitem maximum width form to use the citation keys drftcite uses for the bibliography labels:

```
\mciteSetMaxWidthForm{bibitem}{\mciteBibitemArgI}
```

which, in turn, will revise the bibliography sample label accordingly. This is not done automatically by mciteplus because drftcite does not update the bibliography sample label (even in the absence of mciteplus).

5.6 Hyperref/Backref

Mciteplus should work fine with Sebastian Rahtz and Heiko Oberdiek's hyperref package [5] including it's backref option.

5.7 Multibbl

Mciteplus is fully compatible with Apostolos Syropoulos's multibbl package [15].

5.8 Multibib

Mciteplus is fully compatible with Thorsten Hansen's multibib package [16]. Under mciteplus with natbib, multibib is patched to support all of natbib's `\cite` variants even though the original multibib only supported `\citep`, `\citet`, `\citealp` and `\citealt`. Multibib's internal command name list hook `\@mb@citenamelist` is ignored.

5.9 Natbib

Mciteplus is fully compatible with Patrick W. Daly's natbib package [17] as long as the bibstyle supports both. All of natbib's `\cite` variants are supported including: `\citenum`, `\citep`, `\Citep`, `\citet`, `\Citet`, `\citealp`, `\Citealp`, `\citealt`, `\Citealt`, `\citeauthor`, `\Citeauthor`, `\citeyear`, `\citeyearpar`, `\citepalias` and `\citetalias`.

5.10 Notes2bib

Mciteplus is fully compatible with Joseph Wright's notes2bib package [18]. However, version 1.3 (January 2008) or later must be used for the notes2bib "tail" or "head" options to work with mciteplus.

5.11 REVTEX

Mciteplus is fully compatible with Arthur Ogawa and David Carlisle's REVTEX (Version 4) class [3], including support for end notes in the bibliography. Note that to support this feature, REVTEX uses its own internal bibliography sample label and ignores the ones provided by BIBTEX and mciteplus. One annoyance is that REVTEX writes the footnote "citations" to the auxiliary file which causes BIBTEX to complain that it can't find these "entries" in its database (e.g., "`Warning-I didn't find a database entry for "endnote7"`". To stop this problem, add the following patch code (which you can cut and paste from `mciteplus_code.txt`) right after REVTEX is loaded:

```
% Patch REVTeX to prevent BibTeX from seeing endnotes as citations
% Insert just after REVTeX is loaded
\makeatletter
\let\@ORGREVTEXendnotemark\@endnotemark
\let\@ORGREVTEX@makefnmark@cite\@makefnmark@cite
\def\@endnotemark{\bgroup\@fileswfalse\@ORGREVTEXendnotemark\egroup}
\def\@makefnmark@cite{\bgroup\@fileswfalse\@ORGREVTEX@makefnmark@cite\egroup}
\makeatother
```

5.12 Partially Supported Packages

Footbib: Eric Domenjoud's footbib package [19] "peacefully coexists" with mciteplus in the sense that they do not interact. At present, an mciteplus compatible bibstyle cannot be used for the footbibliography.

5.13 Incompatible Packages

Unfortunately, the following packages are currently known not to work with mciteplus. Many of them clash with mciteplus on a fundamental level. However, others may be supported in the future.

Amsrefs: David Jones' amsrefs package [9].

Apacite: Erik Meijer's apacite package [20].

Biblatex: Philipp Lehman's biblatex package [10]. Future versions of biblatex may well offer features like those of mciteplus.

Bibtopic: Stefan Ulrich and Pierre Basso's bibtopic package [21].

Inlinebib: René Seindal's inlinebib package [22].

Jurabib: Jens Berger's jurabib package [23].

Opcit: Federico Garcia's opcit package [24].

Splitbib: Nicolas Markey's splitbib package [25].

Acknowledgments

First and foremost, the author would like to thank Joseph Wright. Joseph beta tested prototype versions as well as provided many ideas that improved the final architecture of mciteplus.

Thorsten Ohl's mcite package identified the need for a LaTeX package that provides this type of bibliography handling and set the standard for so doing. Although coded differently, mciteplus uses the same basic approach to the problem (i.e., wrap the citation command, drop the tails, forward the head list to the original cite command, and selectively drop items in the bibliography) as originally implemented in mcite.sty.

References

[1] M. Shell. (2013, Sep.) The mciteplus.sty package. [Online]. Available: http://www.ctan.org/tex-archive/macros/latex/contrib/mciteplus/

[2] T. Ohl. (1996, Jan.) The mcite.sty package. [Online]. Available: http://www.ctan.org/tex-archive/macros/latex/contrib/mcite/

[3] A. Ogawa and D. Carlisle. (2010, Aug.) The REVTeX package. [Online]. Available: http://www.ctan.org/tex-archive/macros/latex/contrib/revtex/

[4] M. Shell. (2008, Sep.) The IEEEtran BibTeX style. [Online]. Available: http://www.ctan.org/tex-archive/macros/latex/contrib/IEEEtran/bibtex/

[5] S. Rahtz and H. Oberdiek. (2012, Nov.) The hyperref.sty package. [Online]. Available: http://www.ctan.org/tex-archive/macros/latex/contrib/hyperref/

[6] D. Arseneau. (2010, Sep.) The chapterbib.sty package. [Online]. Available: http://www.ctan.org/tex-archive/macros/latex/contrib/cite/

[7] S. Pakin. (2013, Mar.) The eqparbox.sty package. [Online]. Available: http://www.ctan.org/tex-archive/macros/latex/contrib/eqparbox/

[8] O. Patashnik. (2010, Dec.) The alpha.bst bibstyle. [Online]. Available: http://www.ctan.org/tex-archive/biblio/bibtex/base/

[9] M. Downes and D. M. Jones. (2013, Mar.) The amsrefs package. [Online]. Available: http://www.ctan.org/tex-archive/macros/latex/contrib/amsrefs/

[10] P. Lehman and P. Kime. (2013, Jul.) The biblatex package. [Online]. Available: http://www.ctan.org/tex-archive/macros/latex/contrib/biblatex

[11] T. Hansen. (2004, May) The bibunits.sty package. [Online]. Available: http://www.ctan.org/tex-archive/macros/latex/contrib/bibunits/

[12] D. Arseneau. (2010, Sep.) The cite.sty package. [Online]. Available: http://www.ctan.org/tex-archive/macros/latex/contrib/cite/

[13] B. Briel and U. Oldenburg. (1999, May) The citeref.sty package. [Online]. Available: http://www.ctan.org/tex-archive/macros/latex/contrib/citeref/citeref.sty

[14] D. Arseneau. (2010, Sep.) The drftcite.sty package. [Online]. Available: http://www.ctan.org/tex-archive/macros/latex/contrib/cite/

[15] A. Syropoulos. (2004, Jul.) The multibbl.sty package. [Online]. Available: http://www.ctan.org/tex-archive/macros/latex/contrib/multibbl/

[16] T. Hansen. (2008, Dec.) The multibib.sty package. [Online]. Available: http://www.ctan.org/tex-archive/macros/latex/contrib/multibib/

[17] P. W. Daly. (2010, Sep.) The natbib.sty package. [Online]. Available: http://www.ctan.org/tex-archive/macros/latex/contrib/natbib/

[18] J. Wright. (2013, Jul.) The notes2bib.sty package. [Online]. Available: http://www.ctan.org/tex-archive/macros/latex/contrib/notes2bib/

[19] E. Domenjoud. (2010, Feb.) The footbib.sty package. [Online]. Available: http://www.ctan.org/tex-archive/macros/latex/contrib/footbib/

[20] E. Meijer. (2013, Jul.) The apacite package. [Online]. Available: http://www.ctan.org/tex-archive/biblio/bibtex/contrib/apacite/

[21] S. Ulrich and P. Basso. (2006, Sep.) The bibtopic.sty package. [Online]. Available: http://www.ctan.org/tex-archive/macros/latex/contrib/bibtopic/

[22] R. Seindal. (1999, Jul.) The inlinebib.sty package. [Online]. Available: http://www.ctan.org/tex-archive/biblio/bibtex/contrib/inlinebib/

[23] J. Berger. (2004, Jan.) The jurabib.sty package. [Online]. Available: http://www.ctan.org/tex-archive/macros/latex/contrib/jurabib/

[24] F. Garcia. (2007, Jun.) The opcit.sty package. [Online]. Available: http://www.ctan.org/tex-archive/macros/latex/contrib/opcit/

[25] N. Markey. (2005, Dec.) The splitbib.sty package. [Online]. Available: http://www.ctan.org/tex-archive/macros/latex/contrib/splitbib/