The `modref` Package*

Niel de Beaudrap

February 3, 2009

Abstract

This package defines some hacks to allow authors to define customized output for cross-references of different types, based on the reference label.

1 The purpose and overview of `modref`

Usually, when labelling equations, definitions, figures, *etc.* in a document, experienced authors will give their labels intelligent names indicating the kind of reference, such as `eqn:quadratic`, `def-integral`, `fig.instrument-layout`, or similar labels. In the course of using cross-references, the author will then proceed to repeatedly write same text to accompany each instance of these labels throughout their document, as in the following examples:

```
Theorem~\ref{thm:fundmental-calculus}
Theorem~\ref{thm:intermediate-value}
Theorem~\ref{thm:four-colour}

Definition~\ref{def:hilbert-space}
Definition~\ref{def:delta-dirac}

Figure~\ref{fig:instrument-layout}
Figure~\ref{fig:experimental-data}
Figure~\ref{fig:nine-hundred-billion}
Figure~\ref{fig:newton}
```

and so on. Given that the references are retrieved from the values of counters written to the auxiliary file, one solution might be to redefine the relevant counters, *e.g.* as in

```
\renewcommand\thetheorem{Theorem~\arabic{theorem}}
```

but it is occasionally still desirable to retrieve the reference without the title, as when referring to multiple similar references, such as

```
Theorems~\ref{thm:foo} and~\ref{thm:bar}
```

so that a simple strategy of redefining `\thetheorem` is inadequate. More sophisticated approaches involving redefining how counters are written to the auxiliary file will also produce problems, *e.g.*when one counter "counts within" another: for instance, if theorem numbers are of the format `\thechapter-\arabic{theorem}`, the theorem label would also inherit the labelling of the chapter reference (which is unproductive).

However, if the document author names reference labels in a literate manner, it is possible to automate the production of the accompanying text by scanning those reference labels, in

---

[*] This file has version number v0.9 dated 2009/02/02.

a way totally independent of how the counter is defined, and in a way which does not affect how the reference is recorded in the auxiliary file. It is only necessary to allow the formatting to vary somehow depending on the label. However, as not everyone uses the same labelling scheme for their cross-references, the way in which reference-labels map to reference-formatting should be fairly flexible.

The `modref` package provides the functionality to easily define customized output for the `\ref` command by identifying what sort of cross-reference is being made, based on an initial string of characters in the reference label. This functionality is provided in a transparent manner, in that only changes the output of `\ref` for those cases defined by the user, and the way in which those references are formatted is also specified by the user. It also provides a way for authors to perform more powerful tasks, such as having every reference produce a margin note in a completely customizable fashion, and easy customization of the appearance of displayed equation numbers in `amsmath` environments.

1.1 Comparison with `fancyref`

Similar functionality to what `modref` provides has been available since 1999 in the `fancyref` package.[1] The functionality provided by `modref` differs from that of `fancyref` in the following ways:

- The syntax for defining new reference styles is simpler: `modref` does not require the author provide a macro in which to store each new reference style.
- Authors can also use `modref` to define top-level reference customization which is applied to all well-formed references in a *general* way, using the customized cross-references and the page-number of the reference as arguments, using similar syntax to defining any other macro. (The `fancyref` package enables putting all references in parentheses, putting references in margin-notes, and including page-numbers *à la* `varioref` in all references; the documentation also advertises customizable hooks.)
- The behaviour of `\ref` is changed to use the customized reference behaviour. (Purists who find this distasteful but who still with to use `modref` — or any user who just wishes to access the original functionality of `\ref` — can find the original behaviour retained in the macro `\Ref`.)
- The `modref` package also enables the customization of displayed equation tags, and of the output of the command `\eqref`, as defined by the `amsmath` package.

However, the main distinction between `modref` and `fancyref` is in the design philosophy:

— `fancyref` appears to have been written in an attempt to anticipate anything that an author might wish to do: therefore, it has a great many redundant pre-defined macros, which one may change to achieve varying degrees of change from the default configuration;

— `modref` was written to have a simple interface for customizing references, without many assumptions about how an author might wish to customize their references (except that it be not too tedious): any changes defined by an author are relative to the default behaviour of LaTeX.

It is my hope that, as a result of the above differences, `modref` will prove to be a convenient choice for authors who wish to customize how cross-references appear in their documents.

---

[1] The core functionality of `modref` was developped independently of `fancyref`: my sense of whether someone would have done the same task before failed me in this instance, and I discovered the package after having implemented the commands `\refstyle`, `\setrefdelimiter`, and an earlier version of `\eqrefstyle` (not to mention most of this document). However, much of the functionality of this package as it exists currently was prompted by ideas derived from `fancyref`.

1.2 Functionality provided by `modref`

The `modref` package introduces the following concepts, and access to control how these concepts are applied to cross-references in documents.

Special reference styles

> A “special” reference style is a command which is performed on references of a certain type. What command is performed depends on the type of the reference; and the type of any reference is determined from its label.
>
> The author defines one or more styles by specifying a “prefix” or “style name” for each style — such as “`dfn`” for definitions, “`eq`” for equations, *etc.* — and describes the formatting to be applied for that reference type. For each reference, `modref` then attempts to determine the type of the reference by matching the label against known style names. The name is delimited by default using a colon (“`:`”) character (but the delimiter may be chosen by the author, and even changed mid-document): thus, an equation reference might have a label such as “`eq:quadratic`”, a definition might have a label “`dfn:vectorSpace`”, and so on.
>
> The result of applying a reference style to a particular type of reference is a *customized reference*. Any reference whose label does not contain a defined reference type has no special style applied to it.
>
> The command to use to define special reference styles is `\refstyle`; the command to change the delimiter mid-document is `\setrefdelimiter`. (There is also a package option `delimiter` which allows the delimiter to be set at the beginning of the document.)

Global reference styles

> A “global” reference style is a command which is performed at *every* reference. Only two global reference styles may be defined for a given document: a default one and a “variant” one. The inputs to a global reference style are the customized reference (if the reference is of a particular type) and the page number; authors are free to use these arguments however they wish to define the behaviour of the `\ref` function. Initially, the behaviour is just to pass on the (possibly customized) reference unchanged, omitting the page number.
>
> The commands to define the global reference styles are `\GlobalRefStyle` and `\GlobalVarRefStyle`; the commands which apply the two different reference styles are `\ref` and `\varref`.

Equation references and displayed equation tags

> There is exactly one exception to the above description of how references are customized: if the author uses the `amsmath` package, the output of the command `\eqref` remains *totally unmodified* — unless, of course, they wish to modify it; then `modref` facilitates doing this.
>
> Because the output of `\eqref` is by design intended to be different from that of `\ref`, global reference styles are never applied to any application of `\eqref`. As well, `\eqref` is never defined to attempt to infer the type of a reference from its label (presumably, it’s an equation). However, `modref` does allow different reference styles to be defined for `\eqref`.
>
> While we’re customizing cross-references to equations, why not allow how the equations themselves are labelled to be customized? This allows the author to fine-tune the

appearance of equation tags. However, these customizations do not affect the output of `\eqref`, or vice-versa.

The command to redefine the behaviour of `\eqref` is `\eqrefstyle`; the command to redefine how equation numbers appear in displayed equations is `\displaytagstyle`.

No special reference styles are pre-defined by `modref`, and no global styles are defined (except for the "trivial" styles described above); only the reference-styles which the document author defines will be applied.

1.3 Package options

This version of `modref` has the following option:

`delimiter=`⟨*string*⟩

This option controls the string which delimits the prefix defining the type of a particular reference. If omitted, the default value is a single colon character ("`:`"). This string may be set to any value, except **(a)** the empty string, **(b)** any string containing a comma, or **(c)** any string containing characters not allowed in package options by LaTeX (*e.g.* "`{`", "`}`", *etc.*)

1.4 Package dependencies/conflicts

The package `modref` depends on the `kvoptions` package, which should be included in modern LaTeX 2_ε distributions. It is currently incompatible with the `hyperref` package: compatibility is planned for the next version.

1.5 Bugs

There are no known bugs in `modref`.

2 How to use `modref`

The `modref` package is very simple. The user interface essentially consists of two commands: the invocation `\usepackage{modref}` itself (which currently accepts three options), and the command `\refstyle`; a few other macros are provided to perform auxiliary functions. We will describe these in the opposite order.

2.1 Defining reference styles

Special reference styles

`\refstyle` Special reference styles can be defined for different label-prefixes using the `\refstyle` macro, which has the syntax:

`\refstyle{`*style-name*`}{`*style-format*`}`

which defines a style with the given style-name. (The reason for the mixed- The style-format is a macro definition is a similar to what is used for `\def` and `\newcommand`, using at most a single argument. The value of that single argument should be interpreted as the *usual* output of the `\ref` command: *i.e.* style-formats are descriptions of transformations to perform on "ordinary" cross-references.

The `\refstyle` command can be used multiple times for a single reference-style in order to override previous definitions of the style-formats for that style; however, it is recommended only to use it in the pre-amble to avoid confusion.

Use cases. The effects of using `\refstyle` can be demonstrated as follows.

1. Style-names are identified from cross-reference labels by looking for a label-prefix, separated from the rest of the label by a delimiter such as a colon (`:`) character. For any cross-reference label which does not have such a delimiter, such as `genericlabel`, the usual behaviour of `\ref` applies.

   Example. This list item is labelled with the command `\label{firstitem}`. This label doesn't contain an identifiable label-prefix, so `\ref{firstitem}` produces the output 1, as usual. As well, this document doesn't have a cross-references labelled `undef-reference`; therefore, `\ref{undef-reference}` produces the output **??**, as usual.

2. If a cross-reference label has an identifiable prefix, but this prefix does not correspond to a defined style-format, the usual behaviour of `\ref` applies as well.

   Example. This second list item is labelled with the command `\label{item:second}`. However, this document didn't define any style-name called "`item`"; therefore, `\ref{item:second}` produces the output 2.

3. If a cross-reference label has an identifiable prefix which does correspond to a style-format, the style-format is applied to the cross-reference.

   Example. In the header of this document, the following reference style was defined:

   ```
   \refstyle{itemNo}{[Item~\##1]}
   ```

   The label for this list item was defined using `\label{itemNo:third}`. Because the usual meaning of `\ref{itemNo:third}` would be 3, the resulting output of `\ref{itemNo:third}` is [Item #3].

4. The style-formats do not correspond in any way to individual counters, but only to the labels given to the references. Thus, references which use the same counter may be given different styles, if this is a meaningful thing to do, simply by making the label prefixes different; and references with different counters may be given the same reference style.

   Example. In the header of this document, the following reference style was defined:

   ```
   \refstyle{numero}{\texttt{\itshape numero~#1}}
   ```

   The label for this list item was defined using `\label{numero:fourth}`: this gives rise to a different format of reference than the reference style "`itemNo`" — in particular, `\ref{numero:fourth}` produces *`numero 4`*. At the same time, the cross-reference label for this section is defined by `\label{numero:definingModRefs}`: thus, `\ref{numero:definingModRefs}` produces *`numero 2.1`*.

5. The new implementation for `\ref` applies styles only to well-defined cross-references: that is, it acts as though the customized output is the entire cross-reference.

   Example. This document doesn't have any cross-reference `itemNo:undef-reference`; therefore, despite the fact that this label contains the style-prefix "`itemNo`", the reference `\ref{itemNo:undef-reference}` produces the output **??**, as with any ill-defined reference.

```
\refstyle{def}{Definition~#1}
\refstyle{lemma}{Lemma~#1}
\refstyle{thm}{Theorem~#1}
\refstyle{eqn}{Eqn.~#1}

\eqrefstyle{\textbf{#1]}}
\displaytagstyle{{\Large%
   $\bigcirc\mspace{-12mu}$}#1}

% Main text:

\begin{definition}
   \label{def:lorentz} ...
\end{definition}

\begin{lemma}
   \label{lemma:simultaneity} ...
\end{lemma}

\begin{theorem}
   \label{thm:einstein} ...
   \begin{gather}
      \label{eqn:einstein}
         E = mc^2
   \end{gather}
   ...
\end{theorem}

\ref{thm:einstein} ...
\eqref{eqn:einstein} ...

\ref{def:lorentz} and
\ref{lemma:simultaneity} ...
\ref{eqn:einstein} ...
```

Definition V. A theory of physics is *Lorentz invariant* if [. . .]

Lemma 11. *For two distinct events A and B, there exists a reference frame in which they are simultaneous if* [. . .]

Theorem 4. *The equivalent mass m which is associated with some amount of energy E is determined by the equation*

$$E = mc^2 \qquad ②$$

where c is the speed of light.

Theorem 4 is actually a corollary of a more general theorem, which generalizes **(2)** by adding a dependency on momentum.

Definition V and Lemma 11 are also important ideas forming part of the Special Theory of Relativity; but Eqn. 2 is its most famous formula.

Figure 1: Illustration of (some of) the functionality of `modref` package.

A more coherent illustration of how to use `\refstyle`, and how it affects cross-references, is illustrated in Figure 1.

`\eqrefstyle` The syntax of `\eqrefstyle` is similar: it takes a single argument, specifying the style for any cross-reference using `\eqref`. Global reference styles are not applied to uses of `\eqref`; and as with `\ref`, `\eqref` produces the output **??** when applied to undefined references.

Global reference styles

Two different global reference styles can be defined using the `\GlobalRefStyle` and `\GlobalVarRefStyle` macros, which have the syntax:

`\GlobalRefStyle{`*style-format*`}`
`\GlobalVarRefStyle{`*style-format*`}`

Again, the style-format is a macro definition, in this case using at most *two* arguments. The operands of these two arguments are the (customized) cross-reference, and the page number of that reference: thus, the effect of a global reference style on a cross-reference might look something like

```
\GlobalStyleMacro{Definition~V}{13}
```

for a definition on page thirteen, if definitions have been given a special reference style which inserts "`Definition~`" at the beginning of every reference.[2]

`\varref` The global style defined by `\GlobalRefStyle` is applied to every cross-reference made with `\ref`; the variant global style is applied for cross-references made with `\varref`. (This is the only difference between `\ref` and `\varref`: the functionality will be otherwise identical.)

Initially, both of the global reference styles do nothing except discard the page number, similarly to how `\ref` usually works. In a typical application, it would be quite reasonable to leave the "main" global reference style unchanged, only to customize the "variant" style. For instance, one might sensibly choose to declare

```
\GlobalVarRefStyle{#1 (on page #2)}
```

to facilitate combining cross-referencing with page numbers, or

```
\GlobalVarRefStyle{(#1)}
```

to make it easy to turn a cross-reference into a parenthetical remark, in both cases by using `\varref` to perform the cross-reference.

To avoid too much non-uniformity in the appearance of cross-references, neither `\GlobalRefStyle` nor `\GlobalVarRefStyle` are allowed outside of the document pre-amble.

2.2 Other features

Customizing the delimiter for reference types

`\setrefdelimiter` In the examples above, the style-name was determined from cross-reference labels by scanning for the shortest substring at the beginning of the label which did not contain a "`:`" character. However, authors can change this delimiter to any (non-empty) sequence of characters. This can be done at the beginning of the document by invoking the package as

```
\usepackage[delimiter=⟨string⟩]{modref}
```

or mid-document by using the command `\setrefdelimiter{`*string*`}`; all subsequent uses of `\ref` or `\varref` will attempt to discover reference types using the new delimiter.[3]

Customizing tags in `amsmath`-style displayed equations

`\displaytagstyle` The style in which equation numbers are shown in the displayed-math environments defined by the `amsmath` package can be customized using the `\displaytagstyle` command, which again has similar syntax to `\refstyle` and `\eqref`:

```
\displaytagstyle{style-format}
```

This also has the more general effect of defining the style for the `\tag` command, which can be used to insert *ad-hoc* equation tags in displayed math environments; the effect of the `\tag*` macro, however, is unaffected.

Accessing the original functionality of `\ref`

`\Ref` Sometimes it is useful to access a cross-reference without any special formatting. To do this, one may use the `\Ref` command: this preserves the standard LaTeX 2ε functionality of `\ref`.

---

[2] No macro specifically named *`\GlobalStyleMacro`* is defined by `modref`; this example is only intended for illustrative purposes.

[3] An immediate consequence of this is that references which previously had a defined "reference type" using the old delimiter may not have a defined type using the new delimiter. This will then have the effect of temporarily rescinding all special reference styles for those references using the old delimiter; the styles can be restored by restoring that delimiter.

3 Implementation

3.1 Preliminary definitions

`\modref@error`

First, we define a generic command `\modref@error` for producing errors.

```
\newcommand\modref@error{\PackageError{modref}}
```

`\Ref`

As most of the subsequent definitions revolve around redefining aspects of the `\ref` command and friends, we define the command `\Ref` which preserves the original behaviour for reference. (If the macro `\Ref` is already defined, we produce an error.)

```
\edef\reserved@a{Ref}%
\@ifundefined\reserved@a{%
  \let\Ref\ref
}{%
  \modref@error{%
    Command \string\Ref\ defined already; refer to the help message.%
  }{%
    The "modref" package defines the command \string\Ref\ to allow you (and
    the package) to use the original functionality of \string\ref. However, in
    this instance, \string\Ref\ already had a meaning when "modref" started
    running. Please determine what is defining \string\Ref, and whether you need
    it.}}
```

`\@ifempty`
`\@xifempty`

We also crib the code for `\@ifempty` from `amsgen.sty` for simplicity.

```
\def\@ifempty#1{\@xifempty#1@@..\@nil}
\long\def\@xifempty#1#2@#3#4#5\@nil{      %
  \ifx#3#4\expandafter\@firstoftwo\else\expandafter\@secondoftwo\fi}
```

3.2 Declaration and processing of the package options

Using the `kvoptions` package, we define the options for `modref`, and the default value for the `fixtagsize` option.

```
\DeclareStringOption[:]{delimiter}
\ProcessKeyvalOptions*
```

3.3 The core functionality, and customization of label delimiters

`\setrefselimiter`

The macro `\setrefdelimiter` is a meta-macro which takes a single argument: this argument will be used as a delimiter in further macros which define the infrastructure of `modref`.

```
\newcommand\setrefdelimiter[1]{%
```

Throughout the following, `#1` denotes the delimiter to be used for identifying the style prefix in a cross-reference label.

`\ref`
`\varref`

The replacement for `\ref` will pass its argument onto another macro `\modref@ref`, which will attempt to parse the argument for a delimiter; we define `\varref` similarly. They each pass a macro (`\modref@basestyle` and `\modref@varbasestyle`, respectively) as a final argument, to define the “global” formatting which is to be used for the reference regardless of which particular reference style (if any) is to be used. We employ the standard strategy of appending a delimiter to the end of the argument, together with an unlikely terminal command sequence.

```
  \def\ref##1{\modref@ref##1#1\egroup\modref@basestyle}%
  \def\varref##1{\modref@ref##1#1\egroup\modref@varbasestyle}%
```

`\modref@ref` The command `\modref@ref` attempts to split the argument into a prefix, delimiter, and suffix. If the suffix is empty, the original argument to `\ref` had no delimiter (and thus indicates no style): in that case, `\modref@ref` invokes `\@setref` on the reference, using the global format specified in `##3` and the identity function as the "style format". Otherwise, it passes it on to a second macro `\modref@@ref` for further processing.

```
   \def\modref@ref##1#1##2\egroup##3{%
      \@ifempty{##2}{%
         \expandafter\@setref\csname r@##1\endcsname{##3\expandafter\@iden}{##1}%
      }{%
         \modref@@ref##1#1##2\egroup##3}}%
```

`\modref@@ref` Having been given an alleged style prefix, and a suffix making up the remainder of the reference label, the command `\modref@@ref` attempts to determine whether or not the prefix corresponds to a defined style. We do this by testing for the existence of the command sequence `\modref@`⟨*label-prefix*⟩, which will be the macro which is applied for that style. If that macro does not exist, we invoke `\@setref` on the reference, using the global format specified in `##3` and the identity function as the "style format". Otherwise, we obtain the command sequence for the reference's style-format, and invoke `\@setref` using the global format in `##3` and that style-format.

```
   \def\modref@@ref##1#1##2#1\egroup##3{%
      \@ifundefined{@modref@##1}{%
         \expandafter\@setref\csname r@##1#1##2\endcsname{%
            ##3\expandafter\@iden}{##1#1##2}%
      }{%
         \expandafter\def\expandafter\@tempa\expandafter{%
            \csname @modref@##1\endcsname}%
         \expandafter\@setref\csname r@##1#1##2\endcsname{%
            ##3\expandafter\@tempa}{##1#1##2}%
   }}%
```

This completes the definition of `\setrefdelimiter`: it only remains to execute it for the delimiter which has been chosen. This causes `\ref` to be redefined using the code above, using whatever string is contained in `\modref@delimiter` in the syntax of the definitions. However, if the `delimiter` option was used, and set to the empty string for some reason, we produce an error.

```
}%
\ifx\modref@delimiter\@empty
   \modref@error{Option "delimiter" must be set to a non-empty value}{%
         The option "delimiter" was used, but seems to have been set to the
         empty string. I require a character, or a multi-character string, to
         delimit reference style names within cross-reference labels. If in
         doubt, just remove the "delimiter" option, and this should fix things.}%
\else
   \expandafter\setrefdelimiter\expandafter{\modref@delimiter}%
\fi
```

3.4 Defining global reference formats and special reference styles

`\GlobalRefStyle` `\GlobalVarRefStyle` We provide two commands for modifying the global format for `\ref` and `\varref`, each of which accept an argument describing a two-argument macro. They each create such a macro: these will be used to redefine `\modref@basestyle` or `\modref@varbasestyle`.

```
\newcommand\GlobalRefStyle[1]{\def\modref@@basestyle##1##2{#1}}
\newcommand\GlobalVarRefStyle[1]{\def\modref@@varbasestyle##1##2{#1}}
\@onlypreamble\GlobalRefStyle
```

```
\@onlypreamble\GlobalVarRefStyle
```

`\modref@basestyle`
`\modref@varbasestyle`

We have already met `\modref@basestyle` and `\modref@varbasestyle`, which were used by `\ref` and `\varref` respectively; these commands are used to store the user-customizable "global style" that gets applied to all references, whether or not they have a defined style type. They both take three arguments: a macro to apply as a style type, and two more corresponding to the cross-reference value and the page number of the reference. They are designed in such a way that using the argument

`{\modref@basestyle\expandafter\@modref@`⟨*style-name*⟩`}`

in place of `\@firstoftwo` in a call to `\@setref`, as in the original definition of `\ref` in `latex.ltx`, will have the effect of applying `\modref@basestyle` to precisely the specified style-format, and the cross-reference and page number. It is easy to verify that this is exactly what is done in `\modref@@ref` when called by the new implementation of `\ref`; similar statements hold for `\varref`.

We set `\modref@basestyle` to apply the any format defined by `\GlobalRefStyle`, with the styled reference as the first argument, and the page number as the second; and similarly for `\modref@varbasestyle`. We then define the global reference format in each case to the identity function. In the case where the style-format is replaced by the `\@iden` macro, the effect of the reference formatting then reduces the usage above to the traditional meaning of `\ref`, defined in terms of passing the macro `\@firstoftwo` to `\@setref`.

```
\def\modref@basestyle#1#2#3{\modref@@basestyle{#1{#2}}{#3}}%
\def\modref@varbasestyle#1#2#3{\modref@@varbasestyle{#1{#2}}{#3}}%
\GlobalRefStyle{#1}%
\GlobalVarRefStyle{#1}%
```

`\refstyle`

The `\refstyle` macro for defining particular reference styles takes a style-name ⟨*style-name*⟩ as its first argument, and the formatting of the style as the second argument; the latter may use at most one argument, for the reference label. The command `\refstyle` defines a command sequence `\@modref@`⟨*style-name*⟩ which takes one argument and applies the required formatting. In order to allow easy overriding of reference styles (*e.g.* in order to redefine the formatting of `\eqref` when using the `eqref` option), we do not require that reference style names be unique. We require `\refstyle` to only be used in the document pre-amble; and if `\refstyle` is already defined, we produce an error.

```
\edef\reserved@a{refstyle}%
\@ifundefined\reserved@a{%
   \newcommand\refstyle[2]{%
      \expandafter\def\csname @modref@#1\endcsname##1{#2}}%
}{%
   \modref@error{%
      Command \string\refstyle\ defined already; refer to the help message.
   }{%
      The "modref" package defines the command \string\refstyle; however,
      in this instance, \string\refstyle\ already had a meaning when "modref"
      started running. Please determine what is defining \string\refstyle, and
      whether you need it.}}
```

3.5 Customizing `\eqref` and displayed equation tags

`\eqrefstyle`
`\modref@trivbasestyle`

Changes to the style of `\eqref` can be made, independently of changes to the displayed equation tags, by redefining `\eqref` to call the macro `\modref@eqref`, and redefining that macro accordingly. The way in which `\eqref` invokes `\modref@eqref` is to use it as part of an argument to `\@setref` in such a way as to apply the style before resolving the reference (so that undefined references will result in the output “ **??**” rather than a transformation of it). However, we only redefine `\eqref` if `\eqrefstyle` is called, so that it retains the meaning given to it by `amsmath` unless it is explicitly redefined. We also define a “trivial” base style, *à la* `\modref@basestyle`, to facilitate this.

```
\newcommand\eqrefstyle[1]{%
   \def\eqref##1{%
      \maketag@@@{%
         \expandafter\@setref\csname r@##1\endcsname{%
            \modref@trivbasestyle\expandafter\modref@eqref}{##1}}}%
   \def\modref@eqref##1{#1}}%
\def\modref@trivbasestyle#1#2#3{#1{#2}}
```

`\displaytagstyle`

To set the style of tags in displayed equations, we redefine the internal macro `\tagform@` from `amsmath.sty` to apply the style that we want, which will be defined by a macro `\modref@tagstyle`.

```
\newcommand\displaytagstyle[1]{%
   \def\modref@tagstyle##1{#1}%
   \def\tagform@##1{\maketag@@@{\modref@tagstyle{##1}}}%
```

Because `\eqref` also uses the command `\tagform@` for its implementation, a call to `\displayTagStyle` also re-implements `\eqref` using `\eqrefstyle` — but only if `\eqref` hasn’t already been re-implemented. Whether or not it has is determined by testing if `\modref@eqref` is well-defined. The style that it sets is the original meaning of `\eqref` as defined in `amsmath.sty`, except for expanding the macros `\tagform@` and `\ref`, and removing the macro `\maketag@@@` (which is already part of the new definition of `\eqref`).

```
   \@ifundefined{modref@eqref}{%
      \eqrefstyle{\textup{(\ignorespaces#1\unskip\@@italiccorr)}}%
   }\relax
}%
```

4 Feedback

I would appreciate any feedback on this package that anyone may have, including error reports, suggestions for modest extensions, and criticism (whether about the purpose or about the implementation of the package). Please send your feedback to `jdebeaud@math.uwaterloo.ca`, with the phrase “`modref` package” (or something very similar) in the subject line. Thanks!