The `cleveref` package*

Toby Cubitt
toby-cleveref@dr-qubit.org

17/04/2009

Abstract

The `cleveref` package enhances LaTeX's cross-referencing features, allowing the format of cross-references to be determined automatically according to the "type" of cross-reference (equation, section, etc.) and the context in which the cross-reference is used. The formatting for each cross-reference type can be fully customised in the preamble of your document. In addition, `cleveref` can type-set cross-references to lists of multiple labels, automatically formatting them according to their types, sorting them, and compressing sequences of numerically consecutive labels. Again, the multiple-reference formatting is fully customisable.

Contents

*This document corresponds to cleveref 0.13, dated 17/04/2009.

1 Introduction

When "clever" is used in the name of a computer program, it usually signifies that the programmer is overly smug about his achievements! On the other hand, at the heart of the LaTeX philosophy is the idea that it is clever to delegate as much of the typesetting as possible to the computer, in order to achieve a beautiful, but above all consistent, visual appearance.

Both these points of view are probably valid when it comes to the `cleveref` package. Its goals are two-fold: to use the information that LaTeX inherently has about labels as intelligently as possible when type-setting cross-references to them (clever processing); and to enable you to produce an attractive, consistent formatting of cross-references throughout your document, with the minimum of effort (you'd be clever to use it!).

The `cleveref` package enhances LaTeX's cross-referencing facilities by allowing cross-references to be formatted automatically according to the type of thing they refer to (chapter, section, equation, theorem, etc.) and the context in which the cross-reference is used. It can also automatically format cross-references to multiple labels, sort lists of multiple cross-references, compres cross-references to consecutive labels into a reference range, and all kinds of other clever wizardry.

In standard LaTeX, you have almost certainly found yourself writing things like `Eq.~(\ref{eq1})` and `Theorems~\ref{thm1} to~\ref{thm3}` over and over again. Tedium isn't the only downside to this. What happens if you later decide you want equation references to be type-set as `Equation~\ref{eq1}` instead? What happens if you decide to change the theorem labelled `thm1` into a lemma? You have to search through the entire LaTeX source of your document, modifying all references to equations, and changing all references to `thm1`.

The `cleveref` package allows you to define the format for cross-references once-and-for-all in the preamble of your document. If you later decide to change the format of equation references, you only have to change one preamble definition. If you change a theorem into a lemma, you don't need to change any cross-references at all, because `cleveref` will automatically type-set cross-references to it using the appropriate formatting. This makes it far easier to type-set cross-references uniformly across your whole document, as well as avoiding repetitively typing the same text for each and every cross-reference.

Given how useful this is, there are naturally a number of other packages with similar goals, most notably `varioref`, `fancyref`, `hyperref`'s `\autoref` command, and (for theorem-like environments) `ntheorem` (with the `thref` option). (There are many others, but these come closest to providing similar features to `cleveref`.) However, all have certain deficiencies which `cleveref` attempts to overcome.

The `fancyref` package doesn't automatically determine the type of thing being referred to. Instead, it relies on you adhering to a naming convention for labels. This is usually a good idea in any case, but it can be inconvenient. For example, if you change a theorem into a lemma, you have to change the label name, and therefore also all cross-references to it. So you are back to searching and replacing through the entire document, not to mention missing out on all the other `cleveref` features.

The enhanced referencing feature provided by the `varioref` package's `\labelformat` command decides how to format cross-references when the label is *defined*, rather than when it is *referenced*. Often this isn't a problem. But it makes it impossible to format cross-references differently according the context in which they are referenced, which can sometimes be very useful. For example, if you want cross-references at the beginning of a sentence formatted any other way than by capitalising the first letter of the cross-reference text, it is impossible using `varioref`. Perhaps even more significantly, it makes it impossible to type-set multiple references automatically; you are back to typing `Eqs.~(\ref{eq1}) to~(\ref{eq3})` by hand. Not to mention missing out on automatic sorting and compressing of consecutive references, `ntheorem` support, precise control over hyper-links, etc. `cleveref` fully supports `varioref`, taking over responsibility for type-setting cross-references, whilst retaining all the `varioref` page-referencing magic.

The `hyperref` package's `\autoref` command type-sets a name before a cross-reference, determined by the cross-reference type. This is less flexible than `cleveref`'s fully customisable cross-reference formatting but, when combined with `varioref`, the two packages working together come close. However, even with `hyperref`, it is impossible to customise precisely which part of the cross-reference is made into a hyper-link in PDF documents; this is very easy with `cleveref`. And it still remains impossible to type-set multiple references, have consecutive references sorted and compressed automatically, etc.

The `ntheorem` package (with the `thref` option) does things right with regards how and when the format is defined... except that it only works for theorem-like environments. It is possible to use it for other environments, but only in a bastardized form, by manually supplying an optional argument to `\label` commands that specifies the label type. `cleveref` works equally well when referencing any type of thing, as well as fully supporting `ntheorem`. And again, `cleveref` also provides a number additional features over `ntheorem`, such as multiple cross-references, automatic sorting and compressing of consecutive cross-references, control over the placement of hyper-links, etc.

2 Usage

The `cleveref` package is loaded in the usual way, by putting the line

```
\usepackage{cleveref}
```

in your document's preamble. However, care must be taken when using `cleveref` in conjunction with other packages that modify LaTeX's referencing system (see section 8). Basically, `cleveref` must be loaded *last*.

If you just want to get going quickly with `cleveref`, and come back later to read up on all the features it provides in more detail, here's what you need to do. Wherever you would previously have used `\ref`, use `\cref` instead. (Except at the beginning of a sentence, where you should use `\Cref`.) You no longer need to put the name of the thing you're referencing in front

of the \cref command, because cleveref will sort that out for you: i.e. use \cref{eq1} instead of eq.~(\ref{eq1}). If you want to refer to a range of labels, use the \crefrange command: \crefrange{eq1}{eq5} gives eqs.~(1) to~(5). Finally, if you want to refer to multiple things at once, you can now combine them all into one cross-reference and leave cleveref to sort it out: e.g. \cref{eq2,eq1,eq3,eq5,thm2,def1} produces: eqs.~(1) to~(3) and~(5), theorem~5, and definition~1.

3 Type-Setting Cross-References

\cref To automatically type-set a cross-reference according to the type of thing referred to, simply refer to it using \cref{⟨*label*⟩}. cleveref imposes just one extra restriction on the names of labels: they are no longer allowed to contain commas ",". These are instead used to type-set multiple cross-references (see below).

\Cref As it is very difficult[1] for LaTeX to determine whether a cross-reference appears at the beginning of a sentence or not, a capitalised version exists: \Cref{⟨*label*⟩}. By default, this type-sets the cross-reference with the first letter capitalised. (Though the formatting of the \cref and \Cref forms can be fully and independently customised, see section 5.)

\ref cleveref does *not* modify the standard \ref command[2], so you can still use it to type-set the formatted label counter alone, without any additional text or formatting.

\crefrange To type-set a cross-reference range, e.g. Eqs.~(1.1) to~(1.5), use \Crefrange \crefrange or \Crefrange (depending on the capitalisation you require), which take the beginning and end of the range as arguments:

\crefrange{⟨*label1*⟩}{⟨*label2*⟩}

\cref To type-set multiple cross-references, simply list the lables inside the \cref or \Cref \Cref command, separated by commas (you are not allowed to use commas in label names when using cleveref):

\cref{⟨*label1*⟩,⟨*label2*⟩,⟨*label3*⟩,...}

\cref* When cleveref is used along with the hyperref package (see sections 5 and 8), \Cref* additional starred variants of all the referencing commands are available. The \crefrange* standard referencing commands will make cross-references into hyper-links; the \Crefrange* starred variants prevent this, producing the same type-set text but without creating hyper-links.

4 Sorting and Compressing

When cleveref type-sets lists of multiple cross-references, the default behaviour is to automatically sort the list and compress sequences of consecutive cross-

[1] Actually, very likely impossible!

[2] This is not quite true. The original \ref command no longer works when cleveref is loaded, so cleveref redefines it to recover the original behaviour.

references into a reference range. You can change this behaviour by supplying one of the following package options:

`sort` Sort lists of cross-references, but don't compress consecutive references.

`compress` Compress sequences of consecutive references into a reference range, but don't sort the list of cross-references.

`nosort` Neither sort lists of cross-references, *nor* compress consecutive references.

`sort&compress` Sort lists of cross-references, and compress sequences of consecutive references into a reference range (this is the default).

Occasionally, you may want to prevent a particular sequence of consecutive cross-references from being compressed to a reference range, without disabling this feature globally. To achieve this, you can separate the cross-references in the list by one or more empty references, at the point at which you want to prevent compression. For example,

```
\cref{eq1,eq2,eq3,,eq4}
```

will be type-set as

eqs. (1) to (3) and (4)

or

```
\cref{eq1,eq2,,eq3,eq4,eq5,,eq6,eq7,eq8}
```

will be type-set as

eqs. (1), (2), (3) to (5) and (6) to (8)

You can safely put an empty reference between cross-references that would never be compressed anyway; it will simply be ignored.

If lists of cross-references are also being sorted (the default), it can be a little confusing to work out where the empty reference should go in order to prevent compression of a particular consecutive sequence. It's best to think of the empty reference as being "attached" to the cross-reference preceding it. When the list is sorted, the empty reference will still appear after the same preceding reference, and will prevent it being compressed with any subsequent consecutive cross-references. In other words, an empty reference ensures that the preceding reference will appear explicitly in the final, type-set cross-reference:

```
\cref{eq3,,eq2,eq1,eq6,eq4,eq5}
```

will be type-set as

eqs. (1) to (3) and (4) to (6)

5 Customising the Cross-Reference Format

The `cleveref` package allows you to take full control of the type-setting of cross-references, by allowing the formatting to be customised. Defaults appropriate for English documents are provided for the standard label types[3], and support for German and French is provided via package options (see section 5.3). But if you don't like the defaults, or are writing in a different language[4], or you need to refer to something for which no default format is defined, then you can take charge and define your own formats.

If `cleveref` encounters a cross-reference to a type it does not know, it will produce a "reference type undefined" warning, and type-set the cross-reference as

?? `\ref{`⟨*label*⟩`}`

i.e. the label counter preceded by a double question mark. The error message indicates the name of the unknown cross-reference type, which you will then probably want to define. (References to undefined labels still produce a "reference undefined" warning and appear as a double question mark, as usual.)

The cross-reference formats are usually constructed out of components: the cross-reference name (different for each type of cross-reference), the format for the label itself, and the conjunctions used in reference ranges and lists of multiple cross-references. There are two levels of customisation: you can either customise the components, or you can take full control and override the component-derived format entirely.

5.1 Customising the Cross-Reference Components

5.1.1 Global Customisation

The global customisation commands affect all cross-reference formats, unless they are overridden by lower-level customisation commands.

`\crefdefaultlabelformat`

The format for the label counter itself can be customised globally using

`\crefdefaultlabelformat{`⟨*format*⟩`}`

The ⟨*format*⟩ argument can be any valid LaTeX code, though you will need to `\protect` fragile commands. It can (and almost certainly should!) contain three arguments, `#1`, `#2` and `#3`. The first argument is the formatted version of the label counter (e.g. `\theequation`). The other two are used to mark the beginning and end of the part of the cross-reference that should form the hyper-link when the `hyperref` package is used (see section 8). The hyper-link arguments `#2` and `#3` *must* appear in that order. (Leaving them out completely will not cause an error,

[3] For any pedantic classics scholars out there: "lemmas" is recognised as a valid plural form of "lemma" in all current versions of the Oxford English Dictionary. "Lemmata" was last heard in a mathematical debate that took place in a pub just around the corner from Hadrian's wall... a few years before the Romans pulled out of Britain. `cleveref` might have "clever" in its name, but even that doesn't make it pretentious enough to use "lemmata".

[4] Any contributions of translations for missing languages are very welcome! See section 10.10 for information on how to contribute translations.

but in that case no hyper-link will be created when `hyperref` is used, and there are better ways to achieve this. See sections 3 and 8.)

Note that the default format for equation cross-references already overrides `\crefdefaultlabelformat` in order to surround the label by brackets, so the label format for equations must be customised individually (see section 5.1.2).

`\crefrangeconjunction`

The conjunction used in a reference range can be customised by defining `\crefrangeconjunction`:

```
\newcommand{\crefrangeconjunction}{⟨conjunction⟩}
```

It does not have to be an actual conjunction in the linguistic sense, e.g. it is perfectly reasonable to define it to be an emdash "`--`". `\crefrangeconjunction` is used directly between the start and end references in a reference range, without any additional space surrounding it, e.g. `\crefrange{thm1}{thm2}` is type-set as

```
theorems~\ref{thm1}\crefrangeconjunction\ref{thm2}
```

so you may or may not want to include surrounding space, depending on the formatting you desire. For example,

```
\newcommand{\crefrangeconjunction}{ and~}
```

does require surrounding space, whereas

```
\newcommand{\crefrangeconjunction}{--}
```

probably does not.

`\crefpairconjunction`
`\crefmiddleconjunction`
`\creflastconjunction`

The conjunctions used in lists of multiple cross-references can be customised by defining the commands `\crefpairconjunction`, `\crefmiddleconjunction` and `\creflastconjunction`:

```
\newcommand{\crefpairconjunction}{⟨conjunction⟩}
\newcommand{\crefmiddleconjunction}{⟨conjunction⟩}
\newcommand{\creflastconjunction}{⟨conjunction⟩}
```

`\crefpairconjunction` is used when there are only two cross-references in the list, `\creflastconjunction` is used between the penultimate and final cross-reference in a list of more than two, and `\crefmiddleconjunction` is used between all the others. Again, they do not have to be conjunctions in the linguistic sense, and the same considerations about surrounding space apply as in the case of `\crefrangeconjunction`. For example, the default definition of `\crefmiddleconjunction` is:

```
\newcommand{\crefmiddleconjunction}{, }
```

`\crefpairgroupconjunction`
`\crefmiddlegroupconjunction`
`\creflastgroupconjunction`

By default, the conjunctions used to separate sub-lists of different cross-reference types in a multi-reference are identical to those used to separate cross-references of the same type[5]. You can override this by defining the conjunction commands `\crefpairgroupconjunction`, `\crefmiddlegroupconjunction` and `\creflastgroupconjunction`.

[5] More accurately, if you redefine \crefpairconjunction etc. in your preamble, \crefpairgroupconjunction etc. are automatically redefined so that they match. (In some languages, the default definition of \creflastgroupconjunction has an additional comma lacking in \creflastconjunction.)

For example,

```
\cref{eq1,eq2,eq3,thm1,thm2,fig1,thm3}
```

is type-set as

> eqs. (1)\crefrangeconjunction(3)\crefmiddlegroupconjunction
> theorems 1\crefpairconjunction2\crefmiddlegroupconjunction
> fig. 1\creflastgroupconjunction{}theorem 3

5.1.2 Customising Individual Cross-Reference Types

`\crefname` `\Crefname`

The cross-reference name for a given cross-reference type is customised using the `\crefname` and `\Crefname` commands:

> `\crefname{`⟨*type*⟩`}{`⟨*singular*⟩`}{`⟨*plural*⟩`}`
> `\Crefname{`⟨*type*⟩`}{`⟨*singular*⟩`}{`⟨*plural*⟩`}`

used by the `\cref` and `\Cref` commands, respectively. You must supply both ⟨*singular*⟩ and ⟨*plural*⟩ forms of the name. If the corresponding `\Crefname` is undefined when `\crefname` is called, it will define `\Crefname` to be a capitalised version of `\crefname`, using `\MakeUppercase`. Conversely, if the corresponding `\crefname` is undefined when `\Crefname` is called, it will define `\crefname` to be a lower-case version of `\Crefname`, using `\MakeLowercase`. Obviously, this will only work properly if the names begin with a letter. If the first letter is a special character construct, such as an accented character, you will need to surround it by braces. If the first thing in the name is *not* a letter at all (e.g. if it is a LaTeX command), you *must* define both capitalisation variants explicitly. Otherwise you will get strange and fatal errors when processing the document.

The cross-reference ⟨*type*⟩ is usually the name of the counter for the environment (equation, chapter, section, etc.). The exceptions are appendices, labels whose type has been overridden explicitly by supplying an optional argument (see section 6), and theorem-like environments when the `ntheorem` package is loaded, for which ⟨*type*⟩ should instead be the environment name (lemma, corollary, definition, etc.) even when different environments are part of the same numbering sequence. (`ntheorem` provides extra information about the environment when different theorem-like environments share a common counter, which `cleveref` makes use of to distinguish between them automatically.) In the case of appendices, the ⟨*type*⟩ is "appendix" for the top-level sectioning command (`\chapter` or `\section`, depending on the document class), "subappendix" for the sectioning command one level below (`\section` or `\subsection`), "subsubappendix" for the next level of sectioning command, etc.

For convenience, if they have not been otherwise customised by the end of the preamble, the cross-reference name (and label format) for `subsection` is by default inherited from that of `section`, and that of `subsubsection` is inherited from `subsection` (which might itself have been inherited from `section`). Similarly for `subappendix`, `subsubappendix` and `subsubsubappendix`, and also for

`enumii`, `enumiii`, `enumiv` and `enumv`. Finally, `subfigure` and `subtable` inherit from `figure` and `table`, respectively.

`\creflabelformat` You may want the label format for a particular cross-reference type to differ from the global format set by `\crefdefaultlabelformat` (see section 5.1.1). You can do this using

`\creflabelformat{`⟨*type*⟩`}{`⟨*format*⟩`}`

The ⟨*type*⟩ argument is the cross-reference type to customise, and the ⟨*format*⟩ argument defines the label format for cross-references of that type. As in the case of `\crefdefaultlabelformat`, the latter should contain the three arguments `#1`, `#2` and `#3`, the first being the formatted version of the label counter, the others determining the beginning and end of the portion that becomes a hyper-link when the `hyperref` package is loaded (see section 8). `#2` and `#3` *must* appear in that order.

`\crefrangelabelformat` Normally, the start and end references in a reference range are type-set using the usual label format (as defined by `\crefdefaultlabelformat` or `\creflabelformat`) separated by `\crefrangeconjunction` (section 5.1.1). You can override this for a given cross-reference type using

`\crefrangelabelformat{`⟨*type*⟩`}{`⟨*format*⟩`}`

The ⟨*format*⟩ argument should contain six arguments: `#1`, `#2`, `#3`, `#4`, `#5`, `#6`. The first two (`#1` and `#2`) are the formatted versions of the two label counters defining the reference range. The next two (`#3` and `#4`) denote the beginning and end of the hyper-link for the first reference, the final two (`#5` and `#6`) the hyper-link for the second reference. The hyper-link arguments *must* appear in order. For example,

```
\crefrangelabelformat{equation}{(#3#1#4) to~(#5#2#6)}
```

5.2 Low-Level Customisation: Taking Full Control

If you need more precise control over the cross-reference format than is possible by customising the individual components, then you can take full control of the format for any given type, overriding the component-derived format entirely. The formats for single cross-references, reference ranges and multi-references are customised separately. If you only customise some of these, the other formats will be constructed from components, as usual.

Note that when deciding which cross-references should be grouped together for sorting and/or compressing, `cleveref` does something slightly more complicated than simply checking whether the reference types match. In fact, it checks whether the reference *formats* match[6]. This will always be the case for cross-references of the same type. But it could also be the case for cross-references that have different types, if the cross-reference formats happen to be identical.

The reason for doing this is to allow cross-references to e.g. sections and subsections to be grouped together if they have identical formats. The default formats

[6] To be precise, `cleveref` checks whether the `\crefformat`'s match.

for the sectioning commands, figures and subfigures, tables and subtables, and enumerated lists are set up in this way. If you change any of them using the low-level customisation commands, but still want them to be grouped together, then you must ensure that the formats are *identical*. (It is *not* sufficient for the formats to producing identical type-set text; the format definitions must contain identical LaTeX code.)

5.2.1 Single Cross-References

`\crefformat` `\Crefformat`

Cross-reference formats for *single* cross-references are defined or redefined using the `\crefformat` and `\Crefformat` commands, which are used by the `\cref` and `\Cref` commands respectively. These take two arguments: the cross-reference type, and the formatting code:

`\crefformat{`⟨*type*⟩`}{`⟨*format*⟩`}`
`\Crefformat{`⟨*type*⟩`}{`⟨*format*⟩`}`

The ⟨*type*⟩ is usually the name of the counter, except for labels whose type has been overridden explicitly (see section 6), theorem-like environments when `ntheorem` is loaded, in which case it is the environment name, and appendices. For the latter, the ⟨*type*⟩ is "appendix" for the top-level sectioning command (`\chapter` or `\section`, depending on the document class), "subappendix" for the sectioning command one level below (`\section` or `\ubsection`), "subsubappendix" for the next level of sectioning command, etc.

As in the case of the `\crefname` and `\Crefname` commands, if the corresponding `\Crefformat` is undefined when `\crefformat` is called, it will define the `\Crefformat` to produce a capitalised version of `\crefformat`, using `\MakeUppercase`. Conversely, if the corresponding `\crefformat` is undefined whe `\Crefformat` is called, it will define the `\crefformat` to produce a lower-case version of `\Crefformat`, using `\MakeLowercase`. Obviously, this will only work properly if the format starts with a letter, and letter constructs (such as accented letter constructs) must be surrounded by braces (see section 5.1.1).

The ⟨*format*⟩ argument can be any valid LaTeX code, though you will need to `\protect` fragile commands. It should contain three arguments, `#1`, `#2` and `#3`. The first argument is the formatted version of the label counter (e.g. `\theequation`). The other two are used to mark the beginning and end of the part of the cross-reference that forms the hyper-link when the `hyperref` package is used, and *must* appear in that order (see section 8).

As an example,

```
\crefformat{equation}{Eq.~(#2#1#3)}
```

will type-set equation references as

Eq. (⟨*counter*⟩)

with the counter (excluding the brackets) forming the hyper-link.

Note that the hyper-link arguments are *not* letters, so if `#2` appears at the beginning of ⟨*format*⟩, `cleveref` will not be able to automatically define the other capitalisation variant automatically using `\MakeUppercase` or `\MakeLowercase`. In this case, you will have to define both variants separately. For example, if you wanted to the "Eq." to be part of the hyper-link, you would have to define:

```
\crefformat{equation}{#2eq.~(#1)#3}
\Crefformat{equation}{#2Eq.~(#1)#3}
```

5.2.2 Reference Ranges

`\crefrangeformat` `\Crefrangeformat`

The format for reference ranges is defined by `\crefrangeformat` and `\Crefrangeformat`. Like `\creformat` and `\Crefformat`, the commands take two arguments: the cross-reference type, and the formatting code.

```
\crefrangeformat{⟨type⟩}{⟨format⟩}
\Crefrangeformat{⟨type⟩}{⟨format⟩}
```

The same comments apply as in the case of single cross-references: the ⟨*type*⟩ is usually the name of the counter, except for appendices, labels with explicitly overridden types, and theorem-like environments when `ntheorem` is loaded. Again, if the other-capitalisation variant is not already defined, it will be defined automatically.

The ⟨*format*⟩ argument can again be any valid LaTeX code, with fragile commands `\protect`ed. However, this time it should contain *six* arguments, `#1`–`#6`. The first two (`#1` and `#2`) are the formatted versions of the label counters, the next two (`#3` and `#4`) are used to mark the beginning and end of the hyper-link for the first cross-reference, and the final two (`#5` and `#6`) mark the beginning and end of the second cross-reference's hyper-link.

As an example,

```
\crefrangeformat{equation}{eqs.~(#3#1#4) to~(#5#2#6)}
```

would type-set equation reference ranges as

eqs. (⟨*counter1*⟩) to (⟨*counter2*⟩)

with the counters (excluding the brackets) forming the hyper-links.

5.2.3 Multiple Cross-References

`\crefmultiformat` `\Crefmultiformat` `\crefrangemultiformat` `\Crefrangemultiformat`

The format for multiple cross-references is defined by `\crefmultiformat` and `\Crefmultiformat`, and that of reference ranges within multiple cross-references by `\crefrangemultiformat` and `\Crefrangemultiformat`. Multi-references also require *all* the other cross-reference formats to be defined (see sections 5.2.1 and 5.2.2), including the single reference range formats, even if you never use the `\crefrange` and `\Crefrange` commands.

The commands all take five arguments: the cross-reference type, the format for the first cross-reference in a list, the format for the second cross-reference in a

list of two, the format for the middle cross-references in a list of more than two, and the format for the last cross-reference in a list of more than two.

```
\crefmultiformat{⟨type⟩}{⟨first⟩}{⟨second⟩}{⟨middle⟩}{⟨last⟩}
\Crefmultiformat{⟨type⟩}{⟨first⟩}{⟨second⟩}{⟨middle⟩}{⟨last⟩}
\crefrangemultiformat{⟨type⟩}{⟨first⟩}{⟨second⟩}{⟨middle⟩}{⟨last⟩}
\Crefrangemultiformat{⟨type⟩}{⟨first⟩}{⟨second⟩}{⟨middle⟩}{⟨last⟩}
```

The ⟨*type*⟩ is, as ever, the counter name, except for appendices, explicitly overridden label types, and theorem-like environments then the `ntheorem` package is loaded. The same considerations apply to the formatting arguments ⟨*first*⟩, ⟨*second*⟩, ⟨*middle*⟩ and ⟨*last*⟩ as for the ⟨*format*⟩ argument of `\crefformat` or `\crefrangeformat`, including the meaning of the arguments that should appear in the formatting code (`#1`, `#2` and `#3` for `\crefmultiformat` and `\Crefmultiformat`, `#1`–`#6` for `\crefmultiformat` and `\Crefmultiformat`). However, when the corresponding other-capitalisation variant is automatically defined, only the first letter of the ⟨*first*⟩ argument is upper- or lower-cased; the other arguments are defined to be identical for both variants.

Be careful to get the spaces at the beginning and end of the formatting code correct: the ⟨*first*⟩ and ⟨*second*⟩, or ⟨*first*⟩, ⟨*middle*⟩ and ⟨*last*⟩, LaTeX code is typeset one after another in a multi-reference, with no space separating them. You may or may not want spaces at the beginning and end of the formatting code, depending on the formatting you desire. For example, in the default equation format:

```
\crefmultiformat{equation}{eqs.~(#2#1#3)}%
  { and~(#2#1#3)}{, (#2#1#3)}{ and~(#2#1#3)}
```

the ⟨*middle*⟩ argument should *not* have a space at the beginning, whereas the ⟨*second*⟩ and ⟨*last*⟩ arguments *should* have a space.

5.3 Language and `babel` support

`cleveref` supports different languages via package options, in the usual way, though not all languages are supported yet[7]. The `babel` package is also supported when it is loaded, allowing you to change language using the `babel` language switching commands such as `\selectlanguage` and `\foreignlanguage`.

The `babel` support works by redefining the cross-reference names and conjunctions for the default cross-reference types. Any customisations you make to the default cross-reference names and conjunctions *in the preamble* apply to the main language (i.e. the last language listed in the options). A `\selectlanguage` command or similar will override these customisations, replacing them with the defaults for the new language. If you later use `\selectlanguage` to switch back to the main language, your customisations will be restored. If you want to customise

[7] Currently, only `english`, `german`, `ngerman` and `french` are supported. Contributions of translations for missing languages are very welcome! See section 10.10 for information on how to contribute translations.

cross-reference names or conjunctions for any language other than the main one, you either have to explicitly redefine them after every language switching command, or hook the redefinitions into `babel`'s language switching mechanism. (See section 10.10 and the `babel` documentation.)

If you have defined formats for new cross-reference types for which no defaults are provided, then you're on your own. `cleveref` will not know how to redefine them for other languages, and again you will have to take care of it yourself, either by explicitly redefining them in your document after each language switch, or by hooking the redefinitions into `babel`'s language switching mechanism.

On the other hand, since the language switching commands only modify the cross-reference components, if you use the low-level customisation commands to take full control of the format for a particular cross-reference type, then (unless you're careful) you take it out of the control of `babel` entirely. If you want to use the low-level customisation commands, but *do* still want the language switching commands to work, then you have to use the component macros in your customised formats. The cross-reference names are stored in macros called `\cref@\meta{type}@name`, `\Cref@\meta{type}@name`, `\cref@\meta{type}@name@plural`, and `\Cref@\meta{type}@name@plural`. (Note that since these macro names contain the "@" character, you must use `\makeatletter` and `\makeatother` to access them.)

For example, if you wanted to redefine the equation format so that the cross-reference name ("equation") was also part of the hyper-link, but you still want to be able to switch language using `babel`, you would need something like:

```
\makeatletter
\crefformat{equation}{#2\cref@equation@name~(#1)#3}
...
\makeatother
```

and similarly for `\crefrangeformat`, `\crefmultiformat`, `\Crefformat`, etc.

6 Overriding the Cross-Reference Type

`\label` As described previously, a label's "type" is usually determined by its counter, or in the case of `ntheorem` theorem-like environments by the environment name. Occasionally, you may want to override the cross-reference type for a particular label. You can do this by supplying the desired type as an optional argument to the `\label` command:

`\label[`⟨*type*⟩`]{`⟨*label*⟩`}`

One circumstance in which is useful is when you want to define a special cross-reference format for certain labels of a given type. By supplying a type that doesn't already exist as the optional argument to `\label`, you can then define the cross-reference format for that new type in whatever way you like, without affecting other cross-references of the same type. For example, if a particular

equation contains multiple expressions and you want it to always be referred to in the plural, you could use:

```
\crefname{pluralequation}{eqs.}{eqs.}
...
\label[pluralequation]{eq1}
```

You can of course reuse this format for other plural equations, too.

If you need to do this frequently, it can become tedious specifying the label explicitly each time. An alternative is to use the `aliascnt` package. This lets you define one counter to be an alias for another, so that effectively the same counter has two names. Since `cleveref` determines the label type from the counter name, the two counter aliases can have different cross-reference formats whilst really being the same counter. You have to somehow arrange for the correct counter alias to be used depending on which cross-reference format you want (probably by defining two variants of the environment in question). But the effort involved might be worth the convenience of not having to remember to pass an explicit optional argument to a large number of labels.

You can use this trick to get different cross-reference formats for different theorem-like environments, *without* using the `ntheorem` package[8]. For example,

```
\usepackage{aliascnt}
\usepackage{cleveref}
\newaliascnt{lemma}{theorem}
\newtheorem{lemma}[lemma]{Lemma}
\aliascntresetthe{lemma}
\crefname{lemma}{lemma}{lemmas}
```

7 Poor Man's `cleveref`

Sometimes you may need to send your LaTeX source to someone who can't or won't install the `cleveref` package themselves. For example, many academic journals accept papers in LaTeX format, but only support a small subset of the packages available on CTAN. The `poorman` option was designed specifically to help in this situation.

When the `poorman` option is supplied, your document will be processed as normal. But in addition, a `sed` script will automatically be written, containing rules for replacing all the `\cref` commands with the LaTeX code that they would produce, and using the standard `\ref` command to produce the cross-references themselves. I.e. the script rewrites your document as you would have done if you had had to do it manually!

The advantage, of course, is that you *don't* have to do it manually. Instead, you can use all the features of `cleveref`, and once you've created a version of your document that you want to send elsewere, you can process it through the

[8] This trick seems to belong to LaTeX mythology, and certainly isn't my own idea! But I haven't been able to definitively track down who originally came up with it.

`sed` script to completely remove the `cleveref` dependency. The recipient won't even realise you used `cleveref`!

The `sed` script is written to the same directory as the (main) LaTeX source file, and given the same name as that source file but with the extension `.sed`. To process your document through the script, all you need to do is run the following from your shell:

sed -f ⟨*name*⟩`.sed` ⟨*name*⟩`.tex` >⟨*newname*⟩`.tex`

where ⟨*name*⟩ is the name of the file containing your LaTeX source file minus the `.tex` extension, and ⟨*newname*⟩ is whatever you want to call the new version. *Do not* make ⟨*newname*⟩ the same as ⟨*name*⟩: it won't work. (It's in any case wise to keep the original LaTeX source file containing the `cleveref` commands, in case you need to produce an updated version of your document in the future. Think of the ⟨*newname*⟩`.tex` file in the same way as a `DVI` file: something you can always reproduce from the original source.)

If your document is composed of a number of separate LaTeX source files, combined with `\include` commands, only one `sed` script will be generated, but you will need to run *each* source file through that *same* script (and probably modify the `\include` commands to match the new file names). However, using `babel`'s language switching commands in a document split across multiple separate source files is beyond the capabilities of the `poorman` option. You will almost certainly need to manually tweak the `sed` script in that case.

Note that the `poorman` script cannot fully reproduce the type-setting of the original `cleveref` cross-references in all cases[9]. In particular, any customisation of hyper-links will be lost, and if you're using `varioref` with the `legacyvarioref` option (see section 8, the spacing before the `\vref` command will not be correct.

8 Interaction with Other Packages

The `cleveref` package *must* be loaded *after* all other packages that don't specifically support it[10], i.e. the

```
\usepackage{cleveref}
```

line should always be the last `\usepackage` command in your document's preamble.

Since `cleveref` redefines many internal commands involved in LaTeX's cross-referencing system, it can interact badly with other packages that do the same. The `varioref` package's enhanced referencing features (the ones you make use of by via the `\labelformat` command), the `fancyref` package, and `ntheorem`'s `thref` option are incompatible with `cleveref`. However, since `cleveref` implements a significantly enhanced version of these packages' features, this is not really a problem.

`\thref` `\vref` `\Vref` `\vrefrange` `\Vrefrange` `\fullref`

In fact, if `ntheorem` is loaded with the `thref` option, `cleveref` redefines

ntheorem's \thref command for you, to be an alias for \cref. Similarly, if varioref is loaded, cleveref redefines the \vref, \vrefrange, \fullref commands and variants to instead use the cleveref features for cross-reference formatting, whilst retaining all the varioref page-referencing magic. You can continue to use the other varioref and ntheorem commands (other than \labelformat and the thref option) whilst using cleveref, as long as cleveref is loaded *last*.

\Fullref
\vref*
\Vref*
\vrefrange*
\Vrefrange*
\fullref*
\Fullref*

Note that, whilst in the business of redefining the varioref commands, cleveref seizes the opportunity to get rid of the irritating spacing behaviour of the \vref and \Vref commands, instead making it consistent with the other cleveref cross-referencing commands. This also frees up the starred variants to be used to suppress hyper-links when the hyperref package is loaded, as usual. (Unfortunately, due to lack of support for this in varioref, the page references will still be hyper-links, even when using the starred variants. Go bug the varioref maintainer about this if you don't like it.)

legacyvarioref

However, the cleveref redefinitions break strict compatibility with the original varioref commands. If you need the original spacing behaviour for compatibility reasons, pass the legacyvarioref option to cleveref. If this option is supplied, only the original varioref commands \vref, \vref*, \vrefrange and \fullref will be provided, not any of the additional variants, and the starred variants will change the spacing around the commands as per the original varioref implementation, rather than supressing hyper-links.

Other packages which alter the LaTeX referencing system are unlikely to work properly with cleveref. (For example, although cleveref fully supports the hyperref package's hyper-link features, the backref option is not currently supported.) See section 9 for additional information.

9 Known Bugs and Possible Improvements

9.1 Known Bugs, Non-Bugs, and Work-Arounds

In no particular order:

- When both the amsmath *and* hyperref packages are loaded at the same time, the cleveref cross-referencing commands do not work when used within section titles. If anyone can figure out why, let me know! As a work-around, use \ref within section titles when your document uses both amsmath and hyperref.
- cleveref will not work properly with the standard LaTeX eqnarray environment. There is no intention to fix this. The eqnarray environment is poorly implemented, making it difficult to get it to work properly with

[9] At least, not without resorting to inserting low-level LaTeX code in your document, which would somewhat defeat the purpose of the poorman option.

[10] At the time of writing, I'm not aware of any that do.

`cleveref`. You're better off using the `amsmath` replacements in any case, such as `gather`, `align`, `multline` and `split`, which *do* work properly with `cleveref`. (See `http://www.tug.org/pracjourn/2006-4/madsen/`).

- `cleveref` currently only provides language support for German and French. Please contribute translations for missing languages![11]
- `cleveref` assumes that counters are only ever reset by the standard sectioning commands (`\chapter`, `\section`, etc.). If this is not the case, the automatic compression of consecutive cross-references into a reference range may be incorrect. Making this more flexible would be a simple task, but so far there doesn't seem to be much need for it.
- `cleveref` breaks `hyperref`'s `backref` option, and probably also the `backref` package when used by itself. (This should be fixed in a future version.)
- `cleveref` is almost certainly incompatible with `titleref`, `byname`, `nameref` and the like. This shouldn't be difficult to fix, and if you want this feature, then... patches are always welcome!
- `cleveref` doesn't know about the `subfloat` package, so you have to revert to using `\ref` for cross-references to sub-figures. (This should be fixed in a future version.)
- The `poorman sed` script loses any custom `cleveref` hyper-link formatting you might have defined, and does not reproduce the original spacing around the `\vref` command when `varioref` is used with the `legacyvarioref` option. This is not a bug; it is intentional. The philosophy behind the `poorman` option is to replace `cleveref`'s enhanced cross-referencing with standard LaTeX cross-reference commands that are guaranteed to work with any standard LaTeX installation. Although it would be simple to fix these "bugs", it's almost certainly impossible without using low-level LaTeX code that is unlikely to be supported by journals, thereby defeating the whole purpose of the `poorman` option.

9.2 Possible Improvements

In no particular order:

- The `poorman` option could be enhanced to allow a choice of scripting language rather than just `sed` (e.g. `awk`, `perl`, ...?), but these are unlikely to be much better for those apt to complain about the use of `sed`. The portable option would be to output a TeX "script", but this would be a *much* more work[12] than I'm prepared to invest.

[11] See section 10.10 for information on how to contribute translations.

[12] LaTeX *really* isn't suited to that kind of pattern matching task – just take a look at the code for escaping regexp special characters in this package!

10 Implementation

Essentially, the core of the implementation consists of causing an extra piece of information – the label "type" – to be written to the aux file, and defining `\cref` commands which use this extra information to type-set the reference.

The least invasive implementation of this kind of thing seems to be that used by the `varioref` package. Namely, to redefine the `\refstepcounter` command so that the `\@currentlabel` macro, which usually just contains the type-set version of the counter, now contains the additional type information. (In fact, we write three extra pieces of information: the type, the counter value itself, and the value of the counter that causes the label's counter to be reset, which we call the "prefix" from now on.) `\@currentlabel` eventually gets written to the aux file as an argument to `\newlabel` by the usual LaTeX mechanisms. This involves less hacking to get everything else working again, since very few macros other than `\ref` rely on this particular `\newlabel` argument (nor are other packages likely to, given that `varioref` plays around with it and is a `required` LaTeX package).

10.1 Redefinitions of LaTeX kernel macros

`\refstepcounter` `\@currentlabel` `\cref@old@refstepcounter`

We store the original `\refstepcounter` in `\cref@old@refstepcounter`, then redefine `\refstepcounter` so that it first calls the old version before adding the extra information to `\@currentlabel`. The new `\@refstepcounter` can take an optional argument, which overrides the default "type" inferred from the counter.

```
\let\cref@old@refstepcounter\refstepcounter
\def\refstepcounter{%
  \@ifnextchar[{\refstepcounter@optarg}{\refstepcounter@noarg}%]
}
\def\refstepcounter@noarg#1{%
  \cref@old@refstepcounter{#1}%
  \cref@constructprefix{#1}{\@result}%
  \protected@edef\@currentlabel{%
    [#1][\arabic{#1}][\@result]%
    \csname p@#1\endcsname\csname the#1\endcsname}}
\def\refstepcounter@optarg[#1]#2{%
  \cref@old@refstepcounter{#2}%
  \cref@constructprefix{#2}{\@result}%
  \protected@edef\@currentlabel{%
    [#1][\arabic{#2}][\@result]\@currentlabel}}
```

`\label` `\cref@override@label@type`

We redefine the `\label` command to allow it to take an optional argument that overrides the default reference type in `\@currentlabel`. We have to postpone this redefinition of `\label` until the beginning of the document because some other packages do too, and we need to override their redefinitions. `\cref@override@label@type` is a convenience macro for overriding the label type stored in `\@currentlabel`.

```
\def\cref@override@label@type[#1][#2][#3]#4\@nil#5{[#5][#2][#3]#4}
\AtBeginDocument{%
```

```
\let\cref@old@label\label
\def\label{\@ifnextchar[\label@optarg\cref@old@label}%]
\def\label@optarg[#1]{%
  \protected@edef\@currentlabel{%
    \expandafter\cref@override@label@type\@currentlabel\@nil{#1}}%
  \cref@old@label}%
```

amsmath
\label
\label@in@display

The amsmath package redefines the \label command within equation environments, so if it is loaded we have to extend the behaviour to support the optional argument. With amsmath, the original \label command is stored in \ltx@label, and \label@in@display replaces \abel inside equations. \label@in@display just saves the label for later, and defining it is left until the end of the equation, when \ltx@label is finally called.

To allow \label within equations to support an optional argument, we first store the original \label@in@display and the new \label macro we defined above (since \label will be clobbered inside equations). Then we redefine \abel@in@display so that it wraps all its arguments, including any optional argument, in an extra set of braces. These are stripped away again by \ltx@label before calling the \label macro we defined above (saved in \cref@label).

```
\@ifpackageloaded{amsmath}{%
  \let\cref@label\label
  \let\cref@old@label@in@display\label@in@display
  \def\label@in@display{%
    \@ifnextchar[\label@in@display@optarg\label@in@display@noarg}%]
  \def\label@in@display@noarg#1{\cref@old@label@in@display{{#1}}}
  \def\label@in@display@optarg[#1]#2{%
    \cref@old@label@in@display{[#1]{#2}}}
  \def\ltx@label#1{\cref@label#1}
  \let\cref@old@make@df@tag@@\make@df@tag@@
```

\make@df@tag@@
\make@df@tag@@@

We override the internals of the amsmath \tag command to add the additional information to the label definition.

```
  \def\make@df@tag@@#1{%
    \gdef\df@tag{\maketag@@@{#1}%
      \def\@currentlabel{[equation][][]#1}}}
  \def\make@df@tag@@@#1{%
    \gdef\df@tag{\tagform@{#1}%
      \toks@\@xp{\p@equation{#1}}%
        \edef\@currentlabel{[equation][][]\the\toks@}}}
}{}%
}
```

\ref

The standard \ref macro spits out whatever was in \@currentlabel when the label was written to the aux file, but this now contains additional information which we don't want. Therefore, we redefine \ref to recover the original behaviour. We have to defer redefinition of \ref till the beginning of the document, in case other packages (such as ntheorem) modify it after cleveref is loaded. The apparently

pointless `\cref@reflabel` adds a leyer of indirection to make it easier in case we need to redefine it for `hyperref` later.

```
\def\cref@reflabel#1#2{\@result}
\AtBeginDocument{%
  \DeclareRobustCommand{\ref}{\cref@ref}
  \def\cref@ref#1{%
    \expandafter\ifx\csname r@#1\endcsname\relax%
      \let\@result\relax%
    \else%
      \cref@getlabel{#1}{\@result}%
    \fi%
    \expandafter\@setref\csname r@#1\endcsname{\cref@reflabel}{#1}}%
}
```

`\appendix` `\cref@old@appendix`

The `\appendix` command causes the top-level sectioning commands (`\chapter` or `\section`, depending on the document class) to produce appendices instead. Since we want to be able to format references to appendices separately from references to normal top-level sections, we add to the tasks that `\appendix` does: it redefines `\refstepcounter@noarg` to exceptionally override the label type for chapters or sections, as appropriate, setting it to "appendix" instead. There are two alternative definitions: one if "section" is the top-level sectioning command, and one if "chapter" fulfils that role.

```
\let\cref@old@appendix\appendix
\renewcommand\appendix{%
  \cref@old@appendix%
  \@ifundefined{chapter}{%
    \gdef\refstepcounter@noarg##1{%
      \cref@old@refstepcounter{##1}%
      \cref@constructprefix{##1}{\@result}%
```

We add a large value to the front of the counter data, to force references to anything in appendices to be sorted after everything else.

```
      \edef\@result{{2147483647}\@result}%
```

Override the cross-reference type of sectioning commands.

```
      \def\@tmpa{##1}%
      \def\@tmpb{section}%
      \ifx\@tmpa\@tmpb%
        \protected@edef\@currentlabel{%
          [appendix][\arabic{##1}][\@result]\@currentlabel}%
      \else%
        \def\@tmpa{##1}%
        \def\@tmpb{subsection}%
        \ifx\@tmpa\@tmpb%
          \protected@edef\@currentlabel{%
            [subappendix][\arabic{##1}][\@result]\@currentlabel}%
        \else%
          \def\@tmpa{##1}%
          \def\@tmpb{subsubsection}%
```

```
          \ifx\@tmpa\@tmpb%
            \protected@edef\@currentlabel{%
              [subsubappendix][\arabic{##1}][\@result]%
              \@currentlabel}%
          \else%
            \protected@edef\@currentlabel{%
              [##1][\arabic{##1}][\@result]\@currentlabel}%
          \fi%
        \fi%
      \fi}%
  }{%
    \def\refstepcounter@noarg##1{%
      \cref@old@refstepcounter{##1}%
      \cref@constructprefix{##1}{\@result}%
```

Again, the large value added to the front of the counter data forces references to appendix items to be sorted last.

```
      \edef\@result{{2147483647}\@result}%
```

Override the cross-reference type of sectioning commands.

```
      \def\@tmpa{##1}%
      \def\@tmpb{chapter}%
      \ifx\@tmpa\@tmpb%
        \protected@edef\@currentlabel{%
          [appendix][\arabic{##1}][\@result]\@currentlabel}%
      \else%
        \def\@tmpa{##1}%
        \def\@tmpb{section}%
        \ifx\@tmpa\@tmpb%
          \protected@edef\@currentlabel{%
            [subappendix][\arabic{##1}][\@result]\@currentlabel}%
        \else%
          \def\@tmpa{##1}%
          \def\@tmpb{subsection}%
          \ifx\@tmpa\@tmpb%
            \protected@edef\@currentlabel{%
              [subsubappendix][\arabic{##1}][\@result]%
              \@currentlabel}%
          \else%
            \def\@tmpa{##1}%
            \def\@tmpb{subsubsection}%
            \ifx\@tmpa\@tmpb%
              \protected@edef\@currentlabel{%
                [subsubsubappendix][\arabic{##1}]%
                [\@result]\@currentlabel}%
            \else%
              \protected@edef\@currentlabel{%
                [##1][\arabic{##1}][\@result]\@currentlabel}%
            \fi%
          \fi%
        \fi%
```

```
      \fi}%
  }%
}
```

10.2 Utility Macros

\cref@getlabel
\cref@gettype
\cref@getcounter
\cref@getprefix

Define some utility macros for extracting label, type, and counter information from the contents of `\@currentlabel`, as written to the aux file and stored in `\r@`⟨*label*⟩ when this is re-read on the next pass. Some other packages commandeer the referencing system to write label information to the aux file for other purposes, and probably use `\ref` to recover it later. We still want them to work, so our utility macros must cope with the type information being absent. However, since we need them to be fully expandable in various places, and `\@ifnextchar` is definitely *not* fully expandable, we use the work-around of having the macros store their result in another macro, whose name is passed as the second argument. This other macro *will* then be fully expandable, and can be used e.g. inside an `\edef` or `\csname`...`\endcsname`.

```
\def\cref@getlabel#1#2{%
  \expandafter\let\expandafter\@tmpa\csname r@#1\endcsname%
  \edef\@tmpa{\expandafter\@firstoftwo\@tmpa}%
  \expandafter\@cref@getlabel\@tmpa\@nil#2}
\def\@cref@getlabel{\@ifnextchar[%]
  \@@cref@getlabel{\@@cref@getlabel[][][]}}
\def\@@cref@getlabel[#1][#2][#3]#4\@nil#5{\def#5{#4}}
\def\cref@gettype#1#2{%
  \expandafter\let\expandafter\@tmpa\csname r@#1\endcsname%
  \edef\@tmpa{\expandafter\@firstoftwo\@tmpa}%
  \expandafter\@cref@gettype\@tmpa\@nil#2}
\def\@cref@gettype{\@ifnextchar[%]
  \@@cref@gettype{\@@cref@gettype[][][]}}
\def\@@cref@gettype[#1][#2][#3]#4\@nil#5{\def#5{#1}}
\def\cref@getcounter#1#2{%
  \expandafter\let\expandafter\@tmpa\csname r@#1\endcsname%
  \edef\@tmpa{\expandafter\@firstoftwo\@tmpa}%
  \expandafter\@cref@getcounter\@tmpa\@nil#2}
\def\@cref@getcounter{\@ifnextchar[%]
  \@@cref@getcounter{\@@cref@getcounter[][][]}}
\def\@@cref@getcounter[#1][#2][#3]#4\@nil#5{\def#5{#2}}
\def\cref@getprefix#1#2{%
  \expandafter\let\expandafter\@tmpa\csname r@#1\endcsname%
  \edef\@tmpa{\expandafter\@firstoftwo\@tmpa}%
  \expandafter\@cref@getprefix\@tmpa\@nil#2}
\def\@cref@getprefix{\@ifnextchar[%]
  \@@cref@getprefix{\@@cref@getprefix[][][]}}
\def\@@cref@getprefix[#1][#2][#3]#4\@nil#5{\def#5{#3}}
```

\cref@constructprefix

The `\cref@constructprefix` macro constructs the prefix information stored in `\@currentlabel` (retrieved using `\cref@getprefix`). This information consists of

the numerical value of each counter that's involved in resetting the label's counter, i.e. it contains the numerical values of the chapter, section, subsection... numbers that (ought to) make up the formatted label produced by `\the`⟨*counter*⟩. E.g. if `\theequation` produces "B.1.3", this utility macro will return "[2][1]" (the "3" corresponds to the equation number itself, which is stored separately in `\@currentlabel`). The first argument is the counter in question; the return value is stored in the second argument, which should be a macro name.

The real work is done by the recursive `\@cref@constructprefix` macro, which works its way upwards through the counters' reset lists until it reaches a counter that isn't reset by any other.

```
\def\cref@constructprefix#1#2{%
```

We fully expand the first argument (the counter name) because sometimes a counter name containing a macro gets passed to us.

```
  \edef\@tmpa{\noexpand{\noexpand}\noexpand{#1\noexpand}}%
  \expandafter\def\expandafter\@tmpa\expandafter{\@tmpa{#2}}%
  \expandafter\@cref@constructprefix\@tmpa}
\def\@cref@constructprefix#1#2#3{%
  \cref@resetby{#2}{#3}%
  \ifx#3\relax%
    \edef#3{#1}%
  \else%
    \edef\@tmpa{{{\the\csname c@#3\endcsname}#1}{#3}}%
    \expandafter\expandafter\expandafter\@cref@constructprefix%
      \expandafter\@tmpa\expandafter{\expandafter#3\expandafter}%
  \fi}
```

`\cref@countercmp` The `\cref@countercmp` macro compares two sets of counter data, as returned by `\cref@getcounter`, and `\chardef`'s its third argument to 0 if they're equal, 1 if the first comes earlier than the second, or 2 if the first comes later than the second. This is used later for sorting references.

```
\def\cref@counter@first#1#2\@nil{#1}
\def\cref@counter@rest#1#2\@nil{#2}
\def\cref@countercmp#1#2#3{%
  \begingroup%
  \def\@tmpa{#1}%
```

In order to ensure empty references end up in the right place when sorting lists of multiple references, we make the comparison macro sorts them before a non-empty reference.

```
  \ifx\@tmpa\@empty%
    \def\@result{1}%
  \else%
    \def\@tmpa{#2}%
    \ifx\@tmpa\@empty%
      \def\@result{2}%
    \else%
```

Conversely, undefined references come after everything else.

```
      \expandafter\ifx\csname r@#1\endcsname\relax%
        \def\@result{2}%
      \else%
        \expandafter\ifx\csname r@#2\endcsname\relax%
          \def\@result{1}%
        \else%
```

The real work of comparing two references is done by `\@cref@countercmp`.

```
          \cref@getcounter{#1}{\@countera}%
          \cref@getprefix{#1}{\@prefixa}%
          \cref@getcounter{#2}{\@counterb}%
          \cref@getprefix{#2}{\@prefixb}%
          \edef\@tmpa{{\@prefixa{\@countera}}{\@prefixb{\@counterb}}}%
          \expandafter\@cref@countercmp\@tmpa%
        \fi%
      \fi%
    \fi%
  \fi%
  \expandafter\endgroup\expandafter%
  \chardef\expandafter#3\expandafter=\@result\relax}
```

`\cref@countercmp` The `\@cref@countercmp` macro recursively compares counter components until it runs out of components for one of the references, or finds an two corresponding components that are unequel.

```
\def\@cref@countercmp#1#2{%
  \let\@iterate\relax%
  \def\@counta{#1}%
  \def\@countb{#2}%
  \ifx\@counta\@empty%
    \ifx\@countb\@empty%
      \def\@result{0}%
    \else%
      \def\@result{1}%
    \fi%
  \else%
    \ifx\@countb\@empty%
      \def\@result{2}%
    \else%
      \edef\@counta{\cref@counter@first#1\@nil}%
      \edef\@countb{\cref@counter@first#2\@nil}%
      \ifnum\@counta<\@countb\relax%
        \def\@result{1}%
      \else%
        \ifnum\@counta>\@countb\relax%
          \def\@result{2}%
        \else%
          \edef\@counta{\cref@counter@rest#1\@nil}%
          \edef\@countb{\cref@counter@rest#2\@nil}%
          \edef\@counta{{\@counta}{\@countb}}%
          \expandafter\def\expandafter\@iterate\expandafter{%
```

```
            \expandafter\@cref@countercmp\@counta}%
        \fi%
      \fi%
    \fi%
  \fi%
  \@iterate}
```

`\cref@append@toks` A basic utility macro for appending tokens to a token register.

```
\def\cref@append@toks#1#2{\toks0={#2}%
  \edef\act{\noexpand#1={\the#1\the\toks0}}%
  \act}%
```

`\cref@stack@init` `\cref@stack@top` `\cref@stack@pop` `\cref@stack@push` `\cref@stack@topandbottom` `\cref@stack@add`

We treat multiple references, supplied as a comma-separated list to `\cref` or `\Cref`, as a stack structure. So we define some utility macros for manipulating stacks (`\@nil` is used as an end-of-stack delimeter).

```
\def\cref@stack@init#1{\def#1{\@nil}}
\def\cref@stack@top#1{\expandafter\@cref@stack@top#1}
\def\@cref@stack@top#1,#2\@nil{#1}
\def\cref@stack@pop#1{\expandafter\@cref@stack@pop#1#1}
\def\@cref@stack@pop#1,#2\@nil#3{\def#3{#2\@nil}}
\def\cref@stack@push#1#2{%
  \expandafter\@cref@stack@push\expandafter{#2}{#1}{#2}}
\def\@cref@stack@push#1#2#3{\def#3{#2,#1}}
\def\cref@stack@pull#1#2{\expandafter\@cref@stack@pull#2{#1}{#2}}
\def\@cref@stack@pull#1\@nil#2#3{\def#3{#1#2,\@nil}}
\def\cref@stack@topandbottom#1#2#3{%
  \def#2{}%
  \def#3{}%
  \cref@isstackfull{#1}%
  \if@cref@stackfull%
    \edef#2{\cref@stack@top{#1}}%
    \cref@stack@pop{#1}%
    \cref@isstackfull{#1}%
    \@whilesw\if@cref@stackfull\fi{%
      \edef#3{\cref@stack@top{#1}}%
      \cref@stack@pop{#1}%
      \cref@isstackfull{#1}}%
  \fi}
\def\cref@stack@add#1#2{%
  \begingroup%
    \def\@arg1{#1}%
    \let\@tmpstack#2%
    \newif\if@notthere%
    \@nottheretrue%
    \cref@isstackfull{\@tmpstack}%
    \@whilesw\if@cref@stackfull\fi{%
      \edef\@tmpb{\cref@stack@top{\@tmpstack}}%
      \def\@tmpa{#1}%
      \ifx\@tmpa\@tmpb%
```

```
        \@cref@stackfullfalse%
        \@nottherefalse%
      \else%
        \cref@stack@pop{\@tmpstack}%
        \cref@isstackfull{\@tmpstack}%
      \fi
    }%
  \expandafter\endgroup%
  \if@notthere\cref@stack@push{#1}{#2}\fi}
```

`\if@cref@stackempty` `\if@cref@stackfull` `\cref@isstackempty` `\cref@isstackfull`

The `\cref@isstackempty` and `\cref@isstackfull` macros test whether a stack is empty or full, respectively, and set the corresponding conditionals `\if@cref@stackempty` and `\if@cref@stackfull`.

```
\newif\if@cref@stackempty
\newif\if@cref@stackfull
\def\cref@isstackempty#1{%
  \def\@tmpa{\@nil}%
  \ifx#1\@tmpa\@cref@stackemptytrue%
  \else\@cref@stackemptyfalse\fi}
\def\cref@isstackfull#1{%
  \def\@tmpa{\@nil}%
  \ifx#1\@tmpa\@cref@stackfullfalse%
  \else\@cref@stackfulltrue\fi}
```

`\cref@stack@sort`

The `\cref@stack@sort` macro sorts a stack, using the comparison macro passed in the second argument, which we use later to sort lists of references. We use insertion sort despite its $O(n^2)$ scaling because it's simpler to code, and because we're very unlikely to encounter lists of more than ten or so referrences, so in practice a more complicated $O(n \log n)$ sorting algorithm will very likely be slower anyway.

```
\def\cref@stack@sort#1#2{%
  \begingroup%
  \cref@stack@init{\@sortstack}%
```

Push first element into sorted stack.

```
  \edef\@element{\cref@stack@top{#1}}%
  \expandafter\cref@stack@push\expandafter{\@element}{\@sortstack}%
  \cref@stack@pop{#1}%
```

If empty elements follow first one, need to add them after it in sorted stack.

```
  \cref@isstackfull{#1}%
  \if@cref@stackfull%
    \edef\@tmpa{\cref@stack@top{#1}}%
    \@whilesw\ifx\@tmpa\@empty\fi{%
      \cref@stack@pull{}{\@sortstack}%
      \cref@stack@pop{#1}%
      \cref@isstackempty{#1}%
      \if@cref@stackempty%
        \let\@tmpa\relax%
      \else%
```

```
        \edef\@tmpa{\cref@stack@top{#1}}%
      \fi}%
  \fi%
```

Process elements from stack.

```
  \cref@isstackfull{#1}%
  \@whilesw\if@cref@stackfull\fi{%
    \edef\@element{\cref@stack@top{#1}}%
    \cref@stack@pop{#1}%
```

If empty elements follow current one, need to add them to sorted stack, right after element we're currently dealing with.

```
    \def\@empties{}%
    \cref@isstackfull{#1}%
    \if@cref@stackfull%
      \edef\@tmpa{\cref@stack@top{#1}}%
      \@whilesw\ifx\@tmpa\@empty\fi{%
        \edef\@empties{\@empties,}%
        \cref@stack@pop{#1}%
        \cref@isstackempty{#1}%
        \if@cref@stackempty%
          \let\@tmpa\relax%
        \else%
          \edef\@tmpa{\cref@stack@top{#1}}%
        \fi}%
    \fi%
```

Insert current element into sorted stack, appending any following empty elements.

```
    \edef\@tmpa{{\expandafter\noexpand\@element}%
      {\expandafter\noexpand\@empties}%
      {\noexpand\@sortstack}{\noexpand#2}}%
    \expandafter\cref@stack@insert\@tmpa%
    \cref@isstackfull{#1}}%
  \expandafter\endgroup\expandafter%
  \def\expandafter#1\expandafter{\@sortstack}}
```

`\cref@stack@insert` `\cref@stack@insert{#1}{#2}{#3}{#4}` inserts `#1` into the appropriate location in the sorted stack `#3` (appending `#2` onto the end of `#1` when it's inserted), using the comparison macro `#4`.

```
\def\cref@stack@insert#1#2#3#4{%
  \let\@cmp#4%
  \@cref@stack@insert{}{#1}{#2}{#3}%
  \cref@stack@pop{#3}}
```

`\@cref@stack@insert` `\@cref@stack@insert{#1}{#2}{#3}{#4}` prepends `#1` to the stack resulting from inserting `#2` (with `#3` appended to it) into the sorted stack `#4`.

```
\def\@cref@stack@insert#1#2#3#4{%
  \let\@iterate\relax%
  \cref@isstackempty{#4}%
  \if@cref@stackempty%
```

```
    \cref@stack@push{#1,#2#3}{#4}%
  \else%
    \edef\@tmpa{\cref@stack@top{#4}}%
    \expandafter\@cmp\expandafter{\@tmpa}{#2}{\@result}%
    \ifnum\@result=2\relax%
      \cref@stack@push{#1,#2#3}{#4}%
    \else%
      \cref@stack@pop{#4}%
      \edef\@tmpa{{\noexpand#1,\@tmpa}{\noexpand#2}%
        {\noexpand#3}{\noexpand#4}}%
      \expandafter\def\expandafter\@iterate\expandafter%
        {\expandafter\@cref@stack@insert\@tmpa}%
    \fi%
  \fi%
  \@iterate}
```

`\if@cref@inresetlist` `\cref@isinresetlist` `\cref@resetby`

We need to be able to determine which counter is used to reset a given counter. Usually, resets are done by sectioning counters, and we assume that to be the case here. `\cref@isinresetlist` searches through one counter's reset list, stored in `\cl@`⟨*counter*⟩, to determine whether another counter appears there, and sets the new conditional appropriately. `\cref@reset@by` searches through all the sectioning counters' reset lists, from lowest-level (subsubsection) to highest (part), checking whether the given counter is in the list, and returns the first sectioning counter in whose list it appears. (The value is returned by defining its second argument, which should be a macro name.)

```
\newif\if@cref@inresetlist
\def\cref@isinresetlist#1#2{%
  \begingroup%
    \def\@counter{#1}%
```

We locally redefine `\@elt`, which appears at the head of the expansion of `\cl@`⟨*counter*⟩, so that entries in the reset list end up separated by commas, thus can be treated as a stack.

```
    \def\@elt##1{##1,}%
    \expandafter\ifx\csname cl@#2\endcsname\relax%
      \def\cref@resetstack{,\@nil}%
    \else%
      \edef\cref@resetstack{\csname cl@#2\endcsname\noexpand\@nil}%
    \fi%
    \let\@nextcounter\relax%
    \cref@isstackfull{\cref@resetstack}%
    \@whilesw\if@cref@stackfull\fi{%
      \edef\@nextcounter{\cref@stack@top{\cref@resetstack}}%
      \ifx\@nextcounter\@counter%
        \@cref@stackfullfalse%
      \else%
        \let\@nextcounter\relax%
        \cref@stack@pop{\cref@resetstack}%
        \cref@isstackfull{\cref@resetstack}%
```

```
      \fi}%
    \ifx\@nextcounter\relax%
      \def\@next{\@cref@inresetlistfalse}%
    \else%
      \def\@next{\@cref@inresetlisttrue}%
    \fi%
  \expandafter%
  \endgroup%
  \@next}
```

FIXME: We could easily remove the hard-coded search order in `\cref@resetby` and, say, replace it with a customisable list of counters to search in order. But so far I've yet to encounter a need for anything other than the hard-coded default.

```
\def\cref@resetby#1#2{%
  \let#2\relax%
  \def\@tmpa{#1}%
```

If counter in question is `subfigure` or `subtable`, check if it's reset by `figure` or `table`, respectively.

```
  \def\@tmpb{subfigure}%
  \ifx\@tmpa\@tmpb%
    \cref@isinresetlist{#1}{figure}%
    \if@cref@inresetlist%
      \def#2{figure}%
    \fi%
  \fi%
  \def\@tmpa{#1}%
  \def\@tmpb{subtable}%
  \ifx\@tmpa\@tmpb%
    \cref@isinresetlist{#1}{table}%
    \if@cref@inresetlist%
      \def#2{table}%
    \fi%
  \fi%
```

If counter in question is `enum`$\langle x \rangle$, check if it's reset by a higher-level `enum`$\langle x \rangle$.

```
  \def\@tmpa{#1}%
  \def\@tmpb{enumii}%
  \ifx\@tmpa\@tmpb%
    \def#2{enum}%
  \fi%
  \def\@tmpb{enumiii}%
  \ifx\@tmpa\@tmpb%
    \def#2{enum}%
  \fi%
  \def\@tmpb{enumiv}%
  \ifx\@tmpa\@tmpb%
    \def#2{enum}%
  \fi%
  \def\@tmpb{enumv}%
  \ifx\@tmpa\@tmpb%
```

```
    \def#2{enum}%
  \fi%
  \def\@tmpb{enum}%
  \ifx#2\@tmpb%
    \cref@isinresetlist{#1}{enumiv}%
    \if@cref@inresetlist%
      \def#2{enumiv}%
    \else%
      \cref@isinresetlist{#1}{enumiii}%
      \if@cref@inresetlist%
        \def#2{enumiii}%
      \else%
        \cref@isinresetlist{#1}{enumii}%
        \if@cref@inresetlist%
          \def#2{enumii}%
        \else%
          \cref@isinresetlist{#1}{enumi}%
          \if@cref@inresetlist%
            \def#2{enumi}%
          \else%
           \cref@isinresetlist{#1}{part}%
            \if@cref@inresetlist%
              \def#2{part}%
            \else%
              \let#2\relax%
            \fi%
          \fi%
        \fi%
      \fi%
    \fi%
  \fi%
```

If we haven't found anything so far, check if it's reset by a sectioning command.

```
  \ifx#2\relax%
    \cref@isinresetlist{#1}{table}%
    \if@cref@inresetlist%
      \def#2{table}%
    \else%
      \cref@isinresetlist{#1}{subsubsection}%
      \if@cref@inresetlist%
        \def#2{subsubsection}%
      \else%
        \cref@isinresetlist{#1}{subsection}%
        \if@cref@inresetlist%
          \def#2{subsection}%
        \else%
          \cref@isinresetlist{#1}{section}%
          \if@cref@inresetlist%
            \def#2{section}%
          \else%
```

```
            \cref@isinresetlist{#1}{chapter}%
            \if@cref@inresetlist%
              \def#2{chapter}%
            \else%
             \cref@isinresetlist{#1}{part}%
              \if@cref@inresetlist%
                \def#2{part}%
              \else%
                \let#2\relax%
              \fi%
            \fi%
          \fi%
        \fi%
      \fi%
    \fi%
  \fi}
```

`\if@cref@refconsecutive` `\cref@isrefconsecutive`

Define a new conditional to test whether two references are consecutive (needed when type-setting reference ranges). This uses the counter and prefix (i.e. formatted version of the counter that resets the label's counter) information provided by `\r@`⟨*label*⟩ (via the aux file) to check if the prefices are identical (i.e. the references come from the same chapter, section or whatever), and that the label counters differ by 0 or 1.

```
\newif\if@cref@refconsecutive%
\def\cref@isrefconsecutive#1#2{%
  \begingroup%
    \countdef\refa@counter=0%
    \countdef\refb@counter=1%
    \cref@getcounter{#1}{\@result}%
    \refa@counter=\@result%
    \cref@getcounter{#2}{\@result}%
    \refb@counter=\@result%
    \cref@getprefix{#1}{\refa@prefix}%
    \cref@getprefix{#2}{\refb@prefix}%
    \def\@after{\@cref@refconsecutivefalse}%
    \ifx\refa@prefix\refb@prefix%
      \ifnum\refa@counter=\refb@counter\relax%
        \def\@after{\@cref@refconsecutivetrue}%
      \else%
        \advance\refa@counter 1\relax%
        \ifnum\refa@counter=\refb@counter\relax%
          \def\@after{\@cref@refconsecutivetrue}%
        \fi%
      \fi%
    \fi%
  \expandafter\endgroup\@after}
```

`\cref@processgroup`

`\cref@processgroup` processes the first group of references from the stack passed in argument `#1`, by moving references to the stack passed in argument `#2` until it

encounters a reference that has a different type to those that came before. Note that empty references are treated as having the same type as the preceding one.

```
\def\cref@processgroup#1#2{%
  \edef\@nextref{\cref@stack@top{#1}}%
  \expandafter\ifx\csname r@\@nextref\endcsname\relax%
    \def\@grouptype{\@undefined}%
    \def\@groupformat{\@undefined}%
  \else%
    \expandafter\cref@gettype\expandafter{\@nextref}{\@grouptype}%
    \expandafter\expandafter\expandafter\def%
      \expandafter\expandafter\expandafter\@groupformat%
      \expandafter\expandafter\expandafter{%
        \csname cref@\@grouptype @format\endcsname%
        {\@dummya}{\@dummyb}{\@dummyc}}%
  \fi%
  \let\@nexttype\@grouptype%
  \let\@nextformat\@groupformat%
%
  \@whilesw\ifx\@nextformat\@groupformat\fi{%
    \expandafter\cref@stack@pull\expandafter{\@nextref}{#2}%
    \cref@stack@pop{#1}%
    \cref@isstackempty{#1}%
    \if@cref@stackempty%
      \let\@nexttype\relax%
      \let\@nextformat\relax%
    \else%
      \edef\@nextref{\cref@stack@top{#1}}%
      \ifx\@nextref\@empty%
        \let\@nexttype\@grouptype%
        \let\@nextforamt\@groupformat%
      \else%
        \expandafter\ifx\csname r@\@nextref\endcsname\relax%
          \def\@nexttype{\@undefined}%
          \def\@nextformat{\@undefined}%
        \else%
          \expandafter\cref@gettype\expandafter%
            {\@nextref}{\@nexttype}%
          \def\@tmpa{\@undefined}%
          \ifx\@nexttype\@tmpa%
            \def\@nextformat{\@undefined}%
          \else%
            \expandafter\expandafter\expandafter\def%
              \expandafter\expandafter\expandafter\@nextformat%
              \expandafter\expandafter\expandafter{%
                \csname cref@\@nexttype @format\endcsname%
                {\@dummya}{\@dummyb}{\@dummyc}}%
          \fi%
        \fi%
      \fi%
```

```
    \fi}%
}
```

`\cref@processgroupall` `\cref@processgroupall` processes the first group of references from the stack passed in argument `#1`, by moving all references with the same type as the first one into the stack passed in argument `#2`.

```
\def\cref@processgroupall#1#2{%
  \cref@stack@init{\@tmpstack}%
  \edef\@nextref{\cref@stack@top{#1}}%
  \expandafter\ifx\csname r@\@nextref\endcsname\relax%
    \def\@grouptype{\@undefined}%
    \def\@groupformat{\@undefined}%
  \else%
    \expandafter\cref@gettype\expandafter{\@nextref}{\@grouptype}%
    \expandafter\expandafter\expandafter\def%
      \expandafter\expandafter\expandafter\@groupformat%
      \expandafter\expandafter\expandafter{%
        \csname cref@\@grouptype @format\endcsname%
        {\@dummya}{\@dummyb}{\@dummyc}}%
  \fi%
  \let\@lasttype\@grouptype%
  \let\@lastformat\@groupformat%
  \cref@isstackfull{#1}%
%
  \@whilesw\if@cref@stackfull\fi{%
    \edef\@nextref{\cref@stack@top{#1}}%
    \ifx\@nextref\@empty%
      \ifx\@lastformat\@groupformat%
        \let\@nexttype\@grouptype%
        \let\@nextformat\@groupformat%
      \else%
        \let\@nexttype\relax%
        \let\@nextformat\relax%
      \fi%
    \else%
      \expandafter\ifx\csname r@\@nextref\endcsname\relax%
        \def\@nexttype{\@undefined}%
        \def\@nextformat{\@undefined}%
      \else%
        \expandafter\cref@gettype\expandafter%
          {\@nextref}{\@nexttype}%
        \def\@tmpa{\@undefined}%
        \ifx\@nexttype\@tmpa%
          \def\@nextformat{\@undefined}%
        \else%
          \expandafter\expandafter\expandafter\def%
            \expandafter\expandafter\expandafter\@nextformat%
            \expandafter\expandafter\expandafter{%
              \csname cref@\@nexttype @format\endcsname%
              {\@dummya}{\@dummyb}{\@dummyc}}%
```

```
        \fi%
      \fi%
    \fi%
%
    \ifx\@nextformat\@groupformat%
      \expandafter\cref@stack@pull\expandafter{\@nextref}{#2}%
    \else%
      \expandafter\cref@stack@pull\expandafter{\@nextref}{\@tmpstack}%
    \fi%
    \cref@stack@pop{#1}%
    \let\@lasttype\@nexttype%
    \let\@lastformat\@nextformat%
    \cref@isstackfull{#1}}%
  \let#1\@tmpstack}
```

`\cref@processconsecutive` `\cref@processconsecutive` processes the first sequence of consecutive references from the stack passed in `#1`, sets the macro passed as `#2` to the first reference in the sequence, sets `#3` to the last reference, and sets the counter passed in `#4` to the number of consecutive references in the sequence.

```
\def\cref@processconsecutive#1#2#3#4{%
  #4=0%
  \edef\@nextref{\cref@stack@top{#1}}%
  \cref@stack@pop{#1}%
```

If stack only contains one reference, set appropriate return values.

```
  \cref@isstackempty{#1}%
  \if@cref@stackempty%
    \edef#2{\@nextref}%
    \let#3\relax%
    #4=1\relax%
```

If stack contains multiple references, find end of consecutive references.

```
  \else%
    \edef#2{\@nextref}%
    \let#3\relax%
    \edef\@nextref{\cref@stack@top{#1}}%
    #4=1\relax%
    \expandafter\ifx\csname r@#2\endcsname\relax%
      \@cref@refconsecutivefalse%
    \else%
```

If next reference in substack is empty, it indicates that no further compression should take place. Having served its purpose, the empty reference and any consecutive empty references are removed from the stack.

```
      \ifx\@nextref\@empty%
        \@cref@refconsecutivefalse%
        \@whilesw\ifx\@nextref\@empty\fi{%
          \cref@stack@pop{#1}%
          \cref@isstackempty{#1}%
          \if@cref@stackempty%
```

```
            \let\@nextref\relax%
          \else%
            \edef\@nextref{\cref@stack@top{#1}}%
          \fi}%
```

Otherwise, test whether next reference is consecutive or not.

```
      \else%
        \expandafter\ifx\csname r@\@nextref\endcsname\relax%
          \@cref@refconsecutivefalse%
        \else%
          \edef\@tmpa{{#2}{\@nextref}}%
          \expandafter\cref@isrefconsecutive\@tmpa%
        \fi%
      \fi%
    \fi%
```

Remove references from the stack until we find end of consecutive sequence.

```
    \@whilesw\if@cref@refconsecutive\fi{%
      \advance#4 1%
      \let#3\@nextref%
      \cref@stack@pop{#1}%
      \cref@isstackempty{#1}%
      \if@cref@stackempty%
        \@cref@refconsecutivefalse%
      \else%
        \edef\@nextref{\cref@stack@top{#1}}%
```

If next reference is empty, remove any consecutive empty references and we're done.

```
        \ifx\@nextref\@empty%
          \@cref@refconsecutivefalse%
          \@whilesw\ifx\@nextref\@empty\fi{%
            \cref@stack@pop{#1}%
            \cref@isstackempty{#1}%
            \if@cref@stackempty%
              \let\@nextref\relax%
            \else%
              \edef\@nextref{\cref@stack@top{#1}}%
            \fi}%
```

Otherwise, test whether next reference is consecutive or not.

```
        \else%
          \expandafter\ifx\csname r@\@nextref\endcsname\relax%
            \@cref@refconsecutivefalse%
          \else%
            \edef\@tmpa{{#3}{\@nextref}}%
            \expandafter\cref@isrefconsecutive\@tmpa%
          \fi%
        \fi%
      \fi}%
  \fi}
```

10.3 Referencing Commands

\cref
\Cref
\crefrange
\Crefrange

Define the main referencing command `\cref` and the start-of-sentence variant `\Cref`, along with the reference range commands `\crefrange` and `\Crefrange`.

```
\DeclareRobustCommand{\cref}[1]{\@cref{cref}{#1}}
\DeclareRobustCommand{\Cref}[1]{\@cref{Cref}{#1}}
\DeclareRobustCommand{\crefrange}[2]{\@setcrefrange{#1}{#2}{cref}{}}
\DeclareRobustCommand{\Crefrange}[2]{\@setcrefrange{#1}{#2}{Cref}{}}
```

\@cref

To save duplicating code, the referencing macros pass an argument determining the variant to an auxilliary macro `\@cref`, which does the real work. The `\@cref` macro is the behemoth at the heart of all the clever referencing features. It deals with grouping references by type, type-setting the conjunctions between groups, choosing the right formatting macro to use for each reference, and compressing consecutive references into ranges.

```
\def\@cref#1#2{%
  \begingroup%
```

Initialise some things, and put all the references into a stack called `\@refstack`.

```
  \countdef\count@consecutive=0%
  \countdef\count@group=1%
  \count@group=1%
  \newif\if@secondref%
  \cref@stack@init{\@refstack}%
  \cref@stack@push{#2}{\@refstack}%
  \cref@isstackfull{\@refstack}%
```

Loop until the reference stack is empty.

```
  \@whilesw\if@cref@stackfull\fi{%
```

Move next group of references with same type into `\@refsubstack`.

```
    \cref@stack@init{\@refsubstack}%
    \if@cref@sort%
      \cref@processgroupall{\@refstack}{\@refsubstack}%
      \cref@stack@sort{\@refsubstack}{\cref@countercmp}%
    \else%
      \cref@processgroup{\@refstack}{\@refsubstack}%
    \fi%
```

Type-set appropriate conjunction between groups of reference types.

```
    \ifnum\count@group=1\relax%
      \advance\count@group 1%
    \else%
      \cref@isstackfull{\@refstack}%
      \if@cref@stackfull%
        \@setcref@middlegroupconjunction%
      \else%
        \ifnum\count@group=2\relax%
          \@setcref@pairgroupconjunction%
        \else%
          \@setcref@lastgroupconjunction%
```

```
        \fi%
      \fi%
      \advance\count@group 1%
    \fi%
```

Process first group of consecutive references.

```
    \if@cref@compress%
      \cref@processconsecutive%
        {\@refsubstack}{\@beginref}{\@endref}{\count@consecutive}%
    \else%
      \edef\@beginref{\cref@stack@top{\@refsubstack}}%
      \cref@stack@pop{\@refsubstack}%
```

Empty references serve no purpose when we're not compressing consecutive references, so we simply remove them.

```
      \@whilesw\ifx\@beginref\@empty\fi{%
        \cref@stack@pop{\@refsubstack}%
        \cref@isstackempty{\@refsubstack}%
        \if@cref@stackempty%
          \let\@beginref\relax%
        \else%
          \edef\@beginref{\cref@stack@top{\@refsubstack}}%
        \fi}%
      \let\@endref\relax%
      \count@consecutive=1\relax%
    \fi%
```

If there were no consecutive references, type-set the first reference;

```
    \ifnum\count@consecutive=1\relax%
      \cref@isstackfull{\@refsubstack}%
      \if@cref@stackfull%
        \expandafter\@setcref\expandafter{\@beginref}{#1}{@first}%
      \else%
        \expandafter\@setcref\expandafter{\@beginref}{#1}{}%
      \fi%
```

if there were only two consecutive references, type-set the first one and return the second to the substack (we add an empty reference after it just to make sure there's no further compression);

```
    \else%
      \ifnum\count@consecutive=2\relax%
        \expandafter\@setcref\expandafter{\@beginref}{#1}{@first}%
        \expandafter\cref@stack@push\expandafter%
          {\@endref,}{\@refsubstack}%
```

otherwise, type-set a reference range.

```
      \else%
        \edef\@tmpa{{\@beginref}{\@endref}}%
        \if@cref@stackempty%
          \expandafter\@setcrefrange\@tmpa{#1}{}%
        \else%
```

```
          \expandafter\@setcrefrange\@tmpa{#1}{@first}%
        \fi%
      \fi%
    \fi%
```

Process further groups of consecutive references, until substack is empty.

```
    \@secondreftrue%
    \cref@isstackfull{\@refsubstack}%
    \@whilesw\if@cref@stackfull\fi{%
      \if@cref@compress%
        \cref@processconsecutive%
          {\@refsubstack}{\@beginref}{\@endref}{\count@consecutive}%
      \else%
        \edef\@beginref{\cref@stack@top{\@refsubstack}}%
        \cref@stack@pop{\@refsubstack}%
```

Empty references serve no purpose when we're not compressing consecutive references, so we simply remove them.

```
        \@whilesw\ifx\@beginref\@empty\fi{%
          \cref@stack@pop{\@refsubstack}%
          \cref@isstackempty{\@refsubstack}%
          \if@cref@stackempty%
            \let\@beginref\relax%
          \else%
            \edef\@beginref{\cref@stack@top{\@refsubstack}}%
          \fi}%
        \let\@endref\relax%
        \count@consecutive=1\relax%
      \fi%
```

If the substack is now empty, we will need to type-set an "end" reference, otherwise we will need to type-set a "middle" reference.

```
      \cref@isstackempty{\@refsubstack}%
      \if@cref@stackempty%
        \if@secondref%
          \def\@pos{@second}%
        \else%
          \def\@pos{@last}%
        \fi%
      \else%
        \def\@pos{@middle}%
      \fi%
```

If there were no consecutive references, just type-set the next reference;

```
      \ifnum\count@consecutive=1\relax%
        \edef\@tmpa{{\@beginref}{#1}{\@pos}}%
        \expandafter\@setcref\@tmpa%
      \else%
```

if there were only two consecutive references, type-set the first one, and return the second one to the substack,

```
        \ifnum\count@consecutive=2\relax%
          \expandafter\@setcref\expandafter%
            {\@beginref}{#1}{@middle}%
          \expandafter\cref@stack@push\expandafter%
            {\@endref}{\@refsubstack}%
```

otherwise, type-set a reference range.

```
        \else%
          \edef\@tmpa{{\@beginref}{\@endref}{#1}{\@pos}}%
          \expandafter\@setcrefrange\@tmpa%
        \fi%
      \fi%
      \@secondreffalse%
      \cref@isstackfull{\@refsubstack}%
    }% end loop over reference substack
    \cref@isstackfull{\@refstack}%
  }% end loop over main reference stack
  \endgroup}
```

`\@setcref` The internal `\@setcref` macro deals with actually type-setting the reference, by calling the appropriate type-dependent formatting macro defined by `\crefformat` etc.

```
\def\@setcref#1#2#3{%
  \expandafter\ifx\csname r@#1\endcsname\relax%
    \protect\G@refundefinedtrue%
    \nfss@text{\reset@font\bfseries ??}%
    \@latex@warning{Reference `#1' on page \thepage \space undefined}%
  \else%
    \cref@gettype{#1}{\@temptype}%  puts label type in \@temptype
    \cref@getlabel{#1}{\@templabel}%  puts label in \@templabel
    \expandafter\ifx\csname #2@\@temptype @format#3\endcsname\relax%
      \protect\G@refundefinedtrue%
      \nfss@text{\reset@font\bfseries ??}~\@templabel%
      \@latex@warning{\string\Cref \space reference format for label
        type `\@temptype' undefined}%
    \else%
      \expandafter\@@setcref\expandafter%
        {\csname #2@\@temptype @format#3\endcsname}{#1}%
    \fi%
  \fi}
```

`\@@setcref` We separate out the very final type-setting step into a separate macro, in order to make it easier to redefine things later to make them work with the `hyperref` package.

```
\def\@@setcref#1#2{\cref@getlabel{#2}{\@templabel}#1{\@templabel}{}{}}
```

`\@setcrefrange` The internal `\@setcrefrange` macro deals with type-setting reference ranges, just as `\@setcref` does for normal references. The actual type-setting is no more complicated in the range case; it's the error checking that makes the code so much

longer. We now have to check whether *two* references are undefined, whether *two* reference formats are undefined, whether the reference types are consistent, and also combinations of these various errors.

```
\def\@setcrefrange#1#2#3#4{%
  \begingroup%
```

Check if both references are defined.

```
    \expandafter\ifx\csname r@#1\endcsname\relax%
      \protect\G@refundefinedtrue%
      \@latex@warning{Reference `#1' on page \thepage \space%
        undefined}%
      \expandafter\ifx\csname r@#2\endcsname\relax%
        \nfss@text{\reset@font\bfseries ??}--%
        \nfss@text{\reset@font\bfseries ??}%
        \@latex@warning{Reference `#2' on page \thepage \space%
          undefined}%
      \else%
        \cref@getlabel{#2}{\@labelb}%
        \nfss@text{\reset@font\bfseries ??}--\@labelb%
      \fi%
    \else%
      \expandafter\ifx\csname r@#2\endcsname\relax%
        \protect\G@refundefinedtrue%
        \cref@getlabel{#1}{\@labela}%
        \@labela--\nfss@text{\reset@font\bfseries ??}%
        \@latex@warning{Reference `#2' on page \thepage %
          \space undefined}%
```

If both references are defined, check that the reference format is defined.

```
      \else%
        \cref@gettype{#1}{\@typea}%
        \cref@gettype{#2}{\@typeb}%
        \cref@getlabel{#1}{\@labela}%
        \cref@getlabel{#2}{\@labelb}%
        \edef\@formata{\expandafter\noexpand%
          \csname #3range@\@typea @format#4\endcsname}%
        \edef\@formatb{\expandafter\noexpand%
          \csname #3range@\@typeb @format#4\endcsname}%
        \expandafter\ifx\@formata\relax%
          \protect\G@refundefinedtrue%
          \nfss@text{\reset@font\bfseries ??}~\@labela--\@labelb%
          \@latex@warning{#3\space reference range format for label
            type `\@typea' undefined}%
        \else%
```

If reference types are identical, type-set reference range, otherwise display warning. (Note: there's no need to check if reference format for second type is defined, since if it isn't it will be caught here as a non-identical type.)

```
          \ifx\formata\formatb%
            \expandafter\@@setcrefrange\expandafter{\@formata}{#1}{#2}%
```

```
          \else%
            \protect\G@refundefinedtrue%
            \nfss@text{\reset@font\bfseries ??}~\@labela--\@labelb%
            \@latex@warning{Types inconsistent in reference range for
              references `#1' and `#2' on page \thepage}%
          \fi%
        \fi%
      \fi%
    \fi%
  \endgroup}
```

`\@@setcrefrange` We again separate out the very final type-setting step into a separate macro, in order to make it easier to redefine things later to make them work with the `hyperref` package.

```
\def\@@setcrefrange#1#2#3{%
  \cref@getlabel{#2}{\@labela}%
  \cref@getlabel{#3}{\@labelb}%
  #1{\@labela}{\@labelb}{}{}{}{}}
```

The type-setting of conjunctions is also separated out into separate macros, for the same reason.

```
\def\@setcref@pairgroupconjunction{\crefpairgroupconjunction}
\def\@setcref@middlegroupconjunction{\crefmiddlegroupconjunction}
\def\@setcref@lastgroupconjunction{\creflastgroupconjunction}
```

10.4 Reference Format Customisation Commands

10.4.1 Format Component Commands

`\cref@label@types` The reference formats are usually constructed out of components defined by the user-level `\crefname`, `\Crefname`, `\creflabel` and `\crefrangelabel` commands. `\cref@label@types` keeps track of label types for which components have been defined, and therefore need constructing at `\begindocument` (see below).

FIXME: we don't check if the label type is already in the list, so some formats may needlessly be redefined identically, multiple times.

```
\cref@stack@init{\cref@label@types}
```

`\crefdefaultlabelformat` `\crefname` `\Crefname` `\creflabelformat` `\crefrangelabelformat` The component customisation commands simply use the supplied arguments to define appropriately named macros containing the formatting components. If the corresponding `\Crefname` or `\crefname` variant is not already defined, `\crefname` and `\Crefname` define it to be a version with the first letter capitalised or lower-cased, respectively.

```
\newcommand{\crefdefaultlabelformat}[1]{%
  \def\cref@default@label##1##2##3{#1}}
\newcommand{\crefname}[3]{%
  \@crefname{cref}{#1}{#2}{#3}{}}
\newcommand{\Crefname}[3]{%
  \@crefname{Cref}{#1}{#2}{#3}{}}
```

```
\newcommand{\creflabelformat}[2]{%
  \expandafter\def\csname cref@#1@label\endcsname##1##2##3{#2}%
  \cref@stack@add{#1}{\cref@label@types}}
\newcommand{\crefrangelabelformat}[2]{%
  \expandafter\def\csname cref@#1@rangelabel\endcsname%
    ##1##2##3##4##5##6{#2}%
  \cref@stack@add{#1}{\cref@label@types}}
```

`\crefname@preamble` `\Crefname@preamble`

The `\crefname@preamble` and `\Crefname@preamble` commands are very like the `\crefname` and `\Crefname` commands, but they tag "`@preamble`" onto the end of the generated macro names. They are used when defining the formats for different languages (see section 10.10).

```
\newcommand{\crefname@preamble}[3]{%
  \@crefname{cref}{#1}{#2}{#3}{@preamble}}
\newcommand{\Crefname@preamble}[3]{%
  \@crefname{Cref}{#1}{#2}{#3}{@preamble}}
```

`\@crefname`

The `\@crefname` utility macro does the real work of defining format names, by defining an appropriately named command to contain the format component, and using the additional first argument ("cref" or "Cref") to determine how to define the corresponding command with the other capitalisation. The extra fifth argument tagged onto the end of the generated macro names.

```
\def\@crefname#1#2#3#4#5{%
  \begingroup%
    \expandafter\gdef\csname #1@#2@name#5\endcsname{#3}%
    \expandafter\gdef\csname #1@#2@name@plural#5\endcsname{#4}%
```

The following `\@tmpa` macro makes use of the fact that the first character of #1 is "c" for lower-case and "C" for upper-case, in order to wrap the capitalisation-dependent parts in macros so that the rest of the code can be capitalisation-variant agnostic.

```
    \def\@tmpa##1##2\@nil{%
      \if##1c%
        \def\@other{C##2}%
        \def\@changecase{\MakeUppercase}%
      \else%
        \def\@other{c##2}%
        \def\@changecase{\MakeLowercase}%
      \fi}%
    \@tmpa#1\@nil%
```

If the other capitalisation variant is not already defined...

```
    \@ifundefined{\@other @#2@name#5}{%
```

Define `\@tmpa` and `\tmpb` to be partial expansions (expanded just once) of the macros for the capitalisation variant we've just defined above. The `\@toska` token register just makes the code less verbose.

```
      \expandafter\expandafter\expandafter\def%
      \expandafter\expandafter\expandafter\@tmpa%
      \expandafter\expandafter\expandafter{%
```

```
      \csname#1@#2@name\endcsname}%
    \expandafter\expandafter\expandafter\def%
    \expandafter\expandafter\expandafter\@tmpb%
    \expandafter\expandafter\expandafter{%
      \csname#1@#2@name@plural\endcsname}%
```

Add the `\@changecase` command to the front of the definitions of `\@tmpa` and `\@tmpb`.

```
    \expandafter\expandafter\expandafter\def%
    \expandafter\expandafter\expandafter\@tmpa%
    \expandafter\expandafter\expandafter{%
      \expandafter\@changecase\@tmpa}%
    \expandafter\expandafter\expandafter\def%
    \expandafter\expandafter\expandafter\@tmpb%
    \expandafter\expandafter\expandafter{%
      \expandafter\@changecase\@tmpb}%
```

Define the other capitalisation variants to be the partial expansions (expanded just once) of `\@tmpa` and `\@tmpb`.

```
    \toksdef\@toksa=0%
    \@toksa={%
      \expandafter\gdef\csname\@other @#2@name#5\endcsname}%
    \expandafter\the\expandafter\@toksa\expandafter{\@tmpa}%
    \@toksa={%
      \expandafter\gdef\csname\@other @#2@name@plural#5\endcsname}%
    \expandafter\the\expandafter\@toksa\expandafter{\@tmpb}%
  }{}%
\endgroup%
```

Add label type to list of types that need defining from components.

```
\cref@stack@add{#2}{\cref@label@types}}
```

`\@crefconstructcomponents` The `\@crefconstructcomponents` utility macro puts the reference format components for the specified reference type into temporary macros, for use by later macros. The ridiculous number of "#" characters ensure that the correct number remain when they come to be used later (pairs "##" are collapsed to a single "#" each time the code is expanded).

```
\def\@crefconstructcomponents#1{%
```

Single cross-reference label format.

```
  \@ifundefined{cref@#1@label}{%
    \let\@tmplabel\cref@default@label%
  }{%
    \expandafter\let\expandafter\@tmplabel%
    \csname cref@#1@label\endcsname%
  }%
```

Reference range label format.

```
  \@ifundefined{cref@#1@rangelabel}{%
    \expandafter\def\expandafter\@tmpa\expandafter{%
      \@tmplabel{####1}{####3}{#####4}}%
```

```
    \expandafter\def\expandafter\@tmpb\expandafter{%
      \@tmplabel{####2}{####5}{####6}}%
    \toksdef\@toksa=0%
    \@toksa={\def\@tmprangelabel##1##2##3##4##5##6}%
    % \expandafter\expandafter\expandafter\expandafter%
    % \expandafter\expandafter\expandafter\the%
    % \expandafter\expandafter\expandafter\expandafter%
    % \expandafter\expandafter\expandafter\@toksa%
    % \expandafter\expandafter\expandafter\expandafter%
    % \expandafter\expandafter\expandafter{%
    %   \expandafter\expandafter\expandafter\@tmpa%
    %   \expandafter\crefrangeconjunction\@tmpb}%
    \expandafter\expandafter\expandafter\the%
    \expandafter\expandafter\expandafter\@toksa%
    \expandafter\expandafter\expandafter{%
      \expandafter\@tmpa\expandafter\crefrangeconjunction\@tmpb}%
  }{%
    \expandafter\let\expandafter\@tmprangelabel%
    \csname cref@#1@rangelabel\endcsname%
  }%
```

Get the correct number of "#"'s into the label format definitions.

```
  \expandafter\def\expandafter\@tmplabel\expandafter{%
    \@tmplabel{########1}{########2}{########3}}%
  \expandafter\def\expandafter\@tmprangelabel\expandafter{%
    \@tmprangelabel{########1}{########2}{########3}%
    {########4}{########5}{########6}}%
```

Lower-case singular cross-reference name.

```
  % \expandafter\expandafter\expandafter\def%
  % \expandafter\expandafter\expandafter\@tmpname%
  % \expandafter\expandafter\expandafter{%
  %   \csname cref@#1@name\endcsname}%
  \expandafter\def\expandafter\@tmpname\expandafter{%
    \csname cref@#1@name\endcsname}%
```

Upper-case singular cross-reference name.

```
  % \expandafter\expandafter\expandafter\def%
  % \expandafter\expandafter\expandafter\@tmpName%
  % \expandafter\expandafter\expandafter{%
  %   \csname Cref@#1@name\endcsname}%
  \expandafter\def\expandafter\@tmpName\expandafter{%
    \csname Cref@#1@name\endcsname}%
```

Lower-case plural cross-reference name.

```
  \expandafter\def\expandafter\@tmpnameplural\expandafter{%
    \csname cref@#1@name@plural\endcsname}%
```

Upper-case plural cross-reference name.

```
  \expandafter\def\expandafter\@tmpNameplural\expandafter{%
    \csname Cref@#1@name@plural\endcsname}%
}
```

`\@crefdefineformat` The `\@crefdefineformat` et al. macros construct calls to `\crefformat` et al. for the supplied reference type that define the corresponding formats in terms of the format components. This is mostly just an arduous exercise in controlling macro expansion order.

```
\def\@crefdefineformat#1{%
  \begingroup%
```

Put format components into tmp macros.

```
    \@crefconstructcomponents{#1}%
```

Assemble the arguments for `\crefformat` and `\Crefformat` from the components.

```
    \expandafter\expandafter\expandafter\def%
    \expandafter\expandafter\expandafter\@tmpfirst%
    \expandafter\expandafter\expandafter{%
      \expandafter\@tmpname\expandafter~\@tmplabel}%
    \expandafter\expandafter\expandafter\def%
    \expandafter\expandafter\expandafter\@tmpFirst%
    \expandafter\expandafter\expandafter{%
      \expandafter\@tmpName\expandafter~\@tmplabel}%
```

Define `\crefformat` and `\Crefformat`.

```
    \toksdef\@toksa=0%
    \@toksa={\crefformat{#1}}%
    \expandafter\the\expandafter\@toksa\expandafter{\@tmpfirst}%
    \@toksa={\Crefformat{#1}}%
    \expandafter\the\expandafter\@toksa\expandafter{\@tmpFirst}%
  \endgroup}
```

`\@crefrangedefineformat` Construct call to `\crefrangeformat`.

```
\def\@crefrangedefineformat#1{%
  \begingroup%
```

Put format components into tmp macros.

```
    \@crefconstructcomponents{#1}%
```

Assemble the arguments for `\crefrangeformat` and `\Crefrangeformat` from the components.

```
    \expandafter\expandafter\expandafter\def%
    \expandafter\expandafter\expandafter\@tmpfirst%
    \expandafter\expandafter\expandafter{%
      \expandafter\@tmpnameplural\expandafter~\@tmprangelabel}%
    \expandafter\expandafter\expandafter\def%
    \expandafter\expandafter\expandafter\@tmpFirst%
    \expandafter\expandafter\expandafter{%
      \expandafter\@tmpNameplural\expandafter~\@tmprangelabel}%
```

Define `\crefrangeformat` and `\Crefrangeformat`.

```
    \toksdef\@toksa=0%
    \@toksa={\crefrangeformat{#1}}%
    \expandafter\the\expandafter\@toksa\expandafter{\@tmpfirst}%
    \@toksa={\Crefrangeformat{#1}}%
```

```
    \expandafter\the\expandafter\@toksa\expandafter{\@tmpFirst}%
  \endgroup}
```

`\@crefdefinemultiformat` Construct call to `\crefmultiformat`.

```
\def\@crefdefinemultiformat#1{%
  \begingroup%
```

Put format components into tmp macros.

```
    \@crefconstructcomponents{#1}%
```

Assemble the arguments for `\crefmultiformat` and `\Crefmultiformat` from the components.

```
    \expandafter\expandafter\expandafter\def%
    \expandafter\expandafter\expandafter\@tmpfirst%
    \expandafter\expandafter\expandafter{%
      \expandafter\@tmpnameplural\expandafter~\@tmplabel}%
    \expandafter\expandafter\expandafter\def%
    \expandafter\expandafter\expandafter\@tmpFirst%
    \expandafter\expandafter\expandafter{%
      \expandafter\@tmpNameplural\expandafter~\@tmplabel}%
    % \expandafter\expandafter\expandafter\def%
    % \expandafter\expandafter\expandafter\@tmpsecond%
    % \expandafter\expandafter\expandafter{%
    %   \expandafter\crefpairconjunction\@tmplabel}%
    \expandafter\def\expandafter\@tmpsecond\expandafter{%
      \expandafter\crefpairconjunction\@tmplabel}%
    % \expandafter\expandafter\expandafter\def%
    % \expandafter\expandafter\expandafter\@tmpmiddle%
    % \expandafter\expandafter\expandafter{%
    %   \expandafter\crefmiddleconjunction\@tmplabel}%
    \expandafter\def\expandafter\@tmpmiddle\expandafter{%
      \expandafter\crefmiddleconjunction\@tmplabel}%
    % \expandafter\expandafter\expandafter\def%
    % \expandafter\expandafter\expandafter\@tmplast%
    % \expandafter\expandafter\expandafter{%
    %   \expandafter\creflastconjunction\@tmplabel}%
    \expandafter\def\expandafter\@tmplast\expandafter{%
      \expandafter\creflastconjunction\@tmplabel}%
```

Bundle all four arguments for `\crefmultiformat` in token register `\@toksb`, then call it.

```
    \toksdef\@toksa=0%
    \toksdef\@toksb=1%
    \@toksb={}%
    \expandafter\cref@append@toks\expandafter\@toksb\expandafter{%
      \expandafter{\@tmpfirst}}%
    \expandafter\cref@append@toks\expandafter\@toksb\expandafter{%
      \expandafter{\@tmpsecond}}%
    \expandafter\cref@append@toks\expandafter\@toksb\expandafter{%
      \expandafter{\@tmpmiddle}}%
    \expandafter\cref@append@toks\expandafter\@toksb\expandafter{%
```

```
      \expandafter{\@tmplast}}%
    \@toksa={\crefmultiformat{#1}}%
    \expandafter\the\expandafter\@toksa\the\@toksb%
```

Bundle all four arguments for `\Crefmultiformat` in token register `\@toksb`, then call it.

```
    \@toksb={}%
    \expandafter\cref@append@toks\expandafter\@toksb\expandafter{%
      \expandafter{\@tmpFirst}}%
    \expandafter\cref@append@toks\expandafter\@toksb\expandafter{%
      \expandafter{\@tmpsecond}}%
    \expandafter\cref@append@toks\expandafter\@toksb\expandafter{%
      \expandafter{\@tmpmiddle}}%
    \expandafter\cref@append@toks\expandafter\@toksb\expandafter{%
      \expandafter{\@tmplast}}%
    \@toksa={\Crefmultiformat{#1}}%
    \expandafter\the\expandafter\@toksa\the\@toksb%
  \endgroup}
```

`\@crefrangedefinemultiformat` Construct call to `\crefrangemultiformat`.

```
\def\@crefrangedefinemultiformat#1{%
  \begingroup%
```

Put format components into tmp macros.

```
    \@crefconstructcomponents{#1}%
```

Assemble the arguments that need to be passed to `\crefrangemultiformat` and `\Crefrangemultiformat` from the reference components.

```
    \expandafter\expandafter\expandafter\def%
    \expandafter\expandafter\expandafter\@tmpfirst%
    \expandafter\expandafter\expandafter{%
      \expandafter\@tmpnameplural\expandafter~\@tmprangelabel}%
    \expandafter\expandafter\expandafter\def%
    \expandafter\expandafter\expandafter\@tmpFirst%
    \expandafter\expandafter\expandafter{%
      \expandafter\@tmpNameplural\expandafter~\@tmprangelabel}%
    % \expandafter\expandafter\expandafter\def%
    % \expandafter\expandafter\expandafter\@tmpsecond%
    % \expandafter\expandafter\expandafter{%
    %   \expandafter\crefpairconjunction\@tmprangelabel}%
    \expandafter\def\expandafter\@tmpsecond\expandafter{%
      \expandafter\crefpairconjunction\@tmprangelabel}%
    % \expandafter\expandafter\expandafter\def%
    % \expandafter\expandafter\expandafter\@tmpmiddle%
    % \expandafter\expandafter\expandafter{%
    %   \expandafter\crefmiddleconjunction\@tmprangelabel}%
    \expandafter\def\expandafter\@tmpmiddle\expandafter{%
      \expandafter\crefmiddleconjunction\@tmprangelabel}%
    % \expandafter\expandafter\expandafter\def%
    % \expandafter\expandafter\expandafter\@tmplast%
    % \expandafter\expandafter\expandafter{%
```

```
    %   \expandafter\creflastconjunction\@tmprangelabel}%
    \expandafter\def\expandafter\@tmplast\expandafter{%
      \expandafter\creflastconjunction\@tmprangelabel}%
```

Bundle all four arguments for `\crefrangemultiformat` in token register `\@toksb`, then call it.

```
    \toksdef\@toksa=0%
    \toksdef\@toksb=1%
    \@toksb={}%
    \expandafter\cref@append@toks\expandafter\@toksb\expandafter{%
      \expandafter{\@tmpfirst}}%
    \expandafter\cref@append@toks\expandafter\@toksb\expandafter{%
      \expandafter{\@tmpsecond}}%
    \expandafter\cref@append@toks\expandafter\@toksb\expandafter{%
      \expandafter{\@tmpmiddle}}%
    \expandafter\cref@append@toks\expandafter\@toksb\expandafter{%
      \expandafter{\@tmplast}}%
    \@toksa={\crefrangemultiformat{#1}}%
    \expandafter\the\expandafter\@toksa\the\@toksb%
```

Bundle all four arguments for `\Crefrangemultiformat` in token register `\@toksb`, then call it.

```
    \@toksb={}%
    \expandafter\cref@append@toks\expandafter\@toksb\expandafter{%
      \expandafter{\@tmpFirst}}%
    \expandafter\cref@append@toks\expandafter\@toksb\expandafter{%
      \expandafter{\@tmpsecond}}%
    \expandafter\cref@append@toks\expandafter\@toksb\expandafter{%
      \expandafter{\@tmpmiddle}}%
    \expandafter\cref@append@toks\expandafter\@toksb\expandafter{%
      \expandafter{\@tmplast}}%
    \@toksa={\Crefrangemultiformat{#1}}%
    \expandafter\the\expandafter\@toksa\the\@toksb%
  \endgroup}
```

`\@crefdefineallformats` `\@crefdefineallformats` calls each of the above, to define all formats for the given type from the corresponding components.

```
\def\@crefdefineallformats#1{%
  \@crefdefineformat{#1}%
  \@crefrangedefineformat{#1}%
  \@crefdefinemultiformat{#1}%
  \@crefrangedefinemultiformat{#1}}
```

10.4.2 Format Definition Commands

`\crefformat` `\Crefformat` `\crefrangeformat` `\Crefrangeformat` `\crefmultiformat` `\Crefmultiformat` `\crefrangemultiformat` `\Crefrangemultiformat`

`\crefformat` et al. are lower-level commands that give complete control over the format of different reference types. They override the component-based formats, simply using the supplied arguments to define appropriately named formatting macros, which are called by `\@setcref`. If the corresponding `\Crefformat` or

`\crefformat` variant is not already defined, they define it to be a version with the first letter capitalised or lower-cased.

```
\newcommand{\crefformat}[2]{\@crefformat{cref}{#1}{#2}}
\newcommand{\Crefformat}[2]{\@crefformat{Cref}{#1}{#2}}
\newcommand{\crefrangeformat}[2]{\@crefrangeformat{crefrange}{#1}{#2}}
\newcommand{\Crefrangeformat}[2]{\@crefrangeformat{Crefrange}{#1}{#2}}
\newcommand{\crefmultiformat}[5]{%
  \@crefmultiformat{cref}{#1}{#2}{#3}{#4}{#5}}
\newcommand{\Crefmultiformat}[5]{%
  \@crefmultiformat{Cref}{#1}{#2}{#3}{#4}{#5}}
\newcommand{\crefrangemultiformat}[5]{%
  \@crefrangemultiformat{crefrange}{#1}{#2}{#3}{#4}{#5}}
\newcommand{\Crefrangemultiformat}[5]{%
  \@crefrangemultiformat{Crefrange}{#1}{#2}{#3}{#4}{#5}}
```

The utility macros do the real work, by using the first argument ("cref" or "Cref", and "crefrange" or "Crefrange") to determine how to define the corresponding command with the other capitalisation.

`\@crefformat` `\@crefformat` defines the macros for single references.

```
\def\@crefformat#1#2#3{%
  \begingroup%
    \expandafter\gdef\csname #1@#2@format\endcsname##1##2##3{#3}%
```

The following `\@tmpa` macro makes use of the fact that the first character of #1 is "c" for lower-case and "C" for upper-case, in order to wrap the capitalisation-dependent parts in macros so that the rest of the code can be capitalisation-variant agnostic.

```
    \def\@tmpa##1##2\@nil{%
      \if##1c%
        \def\@other{C##2}%
        \def\@changecase{\MakeUppercase}%
      \else%
        \def\@other{c##2}%
        \def\@changecase{\MakeLowercase}%
      \fi}%
    \@tmpa#1\@nil%
```

If the other capitalisation variant is not already defined...

```
    \@ifundefined{\@other @#2@format}{%
```

Define `\@tmpa` to be a partial expansion (expanded just once) of the capitalisation variant we've just defined above. The `\@toska` token register just makes the code less verbose.

```
      \toksdef\@toksa=0%
      \@toksa={\def\@tmpa##1##2##3}%
      \expandafter\expandafter\expandafter\the%
      \expandafter\expandafter\expandafter\@toksa%
      \expandafter\expandafter\expandafter{%
        \csname#1@#2@format\endcsname{##1}{##2}{##3}}%
```

Add the `\@changecase` command to the front of the definition of `\@tmpa`.

```
      \expandafter\expandafter\expandafter\the%
      \expandafter\expandafter\expandafter\@toksa%
      \expandafter\expandafter\expandafter{%
        \expandafter\@changecase\@tmpa{##1}{##2}{##3}}%
```

Define the other capitalisation variant to be the partial expansion (expanded just once) of `\@tmpa`.

```
      \@toksa={%
        \expandafter\gdef\csname\@other @#2@format\endcsname##1##2##3}%
      \expandafter\the\expandafter\@toksa\expandafter{%
        \@tmpa{##1}{##2}{##3}}%
    }{}%
  \endgroup}
```

`\@crefrangeformat` `\@crefrangeformat` defines the macros for single reference ranges.

```
\def\@crefrangeformat#1#2#3{%
  \begingroup%
    \expandafter\gdef\csname #1@#2@format\endcsname%
      ##1##2##3##4##5##6{#3}%
```

The following `\@tmpa` macro makes use of the fact that the first character of #1 is "c" for lower-case and "C" for upper-case, in order to wrap the capitalisation-dependent parts in macros so that the rest of the code can be capitalisation-variant agnostic.

```
    \def\@tmpa##1##2\@nil{%
      \if##1c%
        \def\@other{C##2}%
        \def\@changecase{\MakeUppercase}%
      \else%
        \def\@other{c##2}%
        \def\@changecase{\MakeLowercase}%
      \fi}%
    \@tmpa#1\@nil%
```

If the other capitalisation variant is not already defined...

```
    \@ifundefined{\@other @#2@format}{%
```

Define `\@tmpa` to be a partial expansion (expanded just once) of the capitalisation variant we've just defined above. The `\@toska` token register just makes the code less verbose.

```
      \toksdef\@toksa=0%
      \@toksa={\def\@tmpa##1##2##3##4##5##6}%
      \expandafter\expandafter\expandafter\the%
      \expandafter\expandafter\expandafter\@toksa%
      \expandafter\expandafter\expandafter{%
        \csname#1@#2@format\endcsname{##1}{##2}{##3}{##4}{##5}{##6}}%
```

Add the `\@changecase` command to the front of the definition of `\@tmpa`.

```
      \expandafter\expandafter\expandafter\the%
      \expandafter\expandafter\expandafter\@toksa%
```

```
      \expandafter\expandafter\expandafter{%
        \expandafter\@changecase\@tmpa{##1}{##2}{##3}{##4}{##5}{##6}}%
```

Define the other capitalisation variant to be the partial expansion (expanded just once) of `\@tmpa`.

```
      \@toksa={\expandafter\gdef%
        \csname\@other @#2@format\endcsname##1##2##3##4##5##6}%
      \expandafter\the\expandafter\@toksa\expandafter{%
        \@tmpa{##1}{##2}{##3}{##4}{##5}{##6}}%
    }{}%
  \endgroup}
```

`\@crefmultiformat` `\@crefmultiformat` defines the macros for multiple references.

```
\def\@crefmultiformat#1#2#3#4#5#6{%
  \begingroup%
    \expandafter\gdef\csname #1@#2@format@first\endcsname##1##2##3{#3}%
    \expandafter\gdef\csname #1@#2@format@second\endcsname##1##2##3{#4}%
    \expandafter\gdef\csname #1@#2@format@middle\endcsname##1##2##3{#5}%
    \expandafter\gdef\csname #1@#2@format@last\endcsname##1##2##3{#6}%
```

The following `\@tmpa` macro makes use of the fact that the first character of #1 is "c" for lower-case and "C" for upper-case, in order to wrap the capitalisation-dependent parts in macros so that the rest of the code can be capitalisation-variant agnostic.

```
    \def\@tmpa##1##2\@nil{%
      \if##1c%
        \def\@other{C##2}%
        \def\@changecase{\MakeUppercase}%
      \else%
        \def\@other{c##2}%
        \def\@changecase{\MakeLowercase}%
      \fi}%
    \@tmpa#1\@nil%
```

If the other capitalisation variant of the first part of the multi-format definition is not already defined...

```
    \@ifundefined{\@other @#2@format@first}{%
```

Define `\@tmpa` to be a partial expansion (expanded just once) of the capitalisation variant we've just defined above. The `\@toska` token register just makes the code less verbose.

```
      \toksdef\@toksa=0%
      \@toksa={\def\@tmpa##1##2##3}%
      \expandafter\expandafter\expandafter\the%
      \expandafter\expandafter\expandafter\@toksa%
      \expandafter\expandafter\expandafter{%
        \csname#1@#2@format@first\endcsname{##1}{##2}{##3}}%
```

Add the `\@changecase` command to the front of the definition of `\@tmpa`.

```
      \expandafter\expandafter\expandafter\the%
      \expandafter\expandafter\expandafter\@toksa%
```

```
      \expandafter\expandafter\expandafter{%
        \expandafter\@changecase\@tmpa{##1}{##2}{##3}}%
```

Define the other capitalisation variant to be the partial expansion (expanded just once) of `\@tmpa`.

```
      \@toksa={%
        \expandafter\gdef\csname\@other @#2@format@first\endcsname%
          ##1##2##3}%
      \expandafter\the\expandafter\@toksa\expandafter{%
        \@tmpa{##1}{##2}{##3}}%
    }{}%
```

The other parts of the multi-format definition are defined to be identical for both capitalisation variants.

```
    \@ifundefined{\@other @#2@format@second}{%
      \@toksa={%
        \expandafter\global\expandafter\let%
        \csname\@other @#2@format@second\endcsname}%
      \expandafter\the\expandafter\@toksa%
        \csname #1@#2@format@second\endcsname%
    }{}%
    \@ifundefined{\@other @#2@format@middle}{%
      \@toksa={%
        \expandafter\global\expandafter\let%
        \csname\@other @#2@format@middle\endcsname}%
      \expandafter\the\expandafter\@toksa%
        \csname #1@#2@format@middle\endcsname%
    }{}%
    \@ifundefined{\@other @#2@format@last}{%
      \@toksa={%
        \expandafter\global\expandafter\let%
        \csname\@other @#2@format@last\endcsname}%
      \expandafter\the\expandafter\@toksa%
        \csname #1@#2@format@last\endcsname%
    }{}%
  \endgroup}
```

`\@crefrangemultiformat` `\@crefmultiformat` defines the macros for reference ranges within multiple references.

```
\def\@crefrangemultiformat#1#2#3#4#5#6{%
  \begingroup%
    \expandafter\gdef\csname #1@#2@format@first\endcsname%
      ##1##2##3##4##5##6{#3}%
    \expandafter\gdef\csname #1@#2@format@second\endcsname%
      ##1##2##3##4##5##6{#4}%
    \expandafter\gdef\csname #1@#2@format@middle\endcsname%
      ##1##2##3##4##5##6{#5}%
    \expandafter\gdef\csname #1@#2@format@last\endcsname%
      ##1##2##3##4##5##6{#6}%
```

The following `\@tmpa` macro makes use of the fact that the first character of #1 is "c" for lower-case and "C" for upper-case, in order to wrap the capitalisation-dependent parts in macros so that the rest of the code can be capitalisation-variant agnostic.

```
\def\@tmpa##1##2\@nil{%
  \if##1c%
    \def\@other{C##2}%
    \def\@changecase{\MakeUppercase}%
  \else%
    \def\@other{c##2}%
    \def\@changecase{\MakeLowercase}%
  \fi}%
\@tmpa#1\@nil%
```

If the other capitalisation variant of the first part of the multi-format definition is not already defined...

```
\@ifundefined{\@other @#2@format@first}{%
```

Define `\@tmpa` to be a partial expansion (expanded just once) of the capitalisation variant we've just defined above. The `\@toska` token register just makes the code less verbose.

```
  \toksdef\@toksa=0%
  \@toksa={\def\@tmpa##1##2##3##4##5##6}%
  \expandafter\expandafter\expandafter\the%
  \expandafter\expandafter\expandafter\@toksa%
  \expandafter\expandafter\expandafter{%
    \csname#1@#2@format@first\endcsname%
      {##1}{##2}{##3}{##4}{##5}{##6}}%
```

Add the `\@changecase` command to the front of the definition of `\@tmpa`.

```
  \expandafter\expandafter\expandafter\the%
  \expandafter\expandafter\expandafter\@toksa%
  \expandafter\expandafter\expandafter{%
    \expandafter\@changecase\@tmpa{##1}{##2}{##3}{##4}{##5}{##6}}%
```

Define the other capitalisation variant to be the partial expansion (expanded just once) of `\@tmpa`.

```
  \@toksa={%
    \expandafter\gdef\csname\@other @#2@format@first\endcsname%
      ##1##2##3##4##5##6}%
  \expandafter\the\expandafter\@toksa\expandafter{%
    \@tmpa{##1}{##2}{##3}{##4}{##5}{##6}}%
}{}%
```

The other parts of the multi-format definition are defined to be identical for both capitalisation variants.

```
\@ifundefined{\@other @#2@format@second}{%
  \@toksa={%
    \expandafter\global\expandafter\let%
    \csname\@other @#2@format@second\endcsname}%
  \expandafter\the\expandafter\@toksa%
```

```
      \csname #1@#2@format@second\endcsname%
  }{}%
  \@ifundefined{\@other @#2@format@middle}{%
    \@toksa={%
      \expandafter\global\expandafter\let%
      \csname\@other @#2@format@middle\endcsname}%
    \expandafter\the\expandafter\@toksa%
      \csname #1@#2@format@middle\endcsname%
  }{}%
  \@ifundefined{\@other @#2@format@last}{%
    \@toksa={%
      \expandafter\global\expandafter\let%
      \csname\@other @#2@format@last\endcsname}%
    \expandafter\the\expandafter\@toksa%
      \csname #1@#2@format@last\endcsname%
  }{}%
\endgroup}
```

10.5 hyperref Support

hyperref
backref

If the hyperref package is loaded, we add hyper-link support to cleveref. Since hyperref messes around with some of the same LaTeX internals as we do, we also have to override some of its redefinitions so that they work with cleveref.

```
\@ifpackageloaded{hyperref}{%
  \PackageInfo{cleveref}{`hyperref' support loaded}
  \@ifpackagewith{hyperref}{backref}{%
    \PackageError{cleveref}{`cleveref' is currently incompatible with
      `hyperref's `backref' option}{Remove the `backref' option from
      `hyperref' if you want to use `cleveref'}}{}
```

\cref@reflabel
\cref@getlabel

We redefine the utility macros to cope with the extra arguments supplied by hyperref (via the aux file).

\cref@gettype
\cref@getcounter
\cref@getprefix

```
  \def\cref@reflabel#1#2#3#4#5{\@result}
  \def\cref@hyperref#1{\expandafter\expandafter\expandafter%
    \@fourthoffive\csname r@#1\endcsname}
  \def\cref@getlabel#1#2{%
    \expandafter\let\expandafter\@tmpa\csname r@#1\endcsname%
    \edef\@tmpa{\expandafter\@firstoffive\@tmpa}%
    \expandafter\@cref@getlabel\@tmpa\@nil#2}
  \def\cref@gettype#1#2{%
    \expandafter\let\expandafter\@tmpa\csname r@#1\endcsname%
    \edef\@tmpa{\expandafter\@firstoffive\@tmpa}%
    \expandafter\@cref@gettype\@tmpa\@nil#2}
  \def\cref@getcounter#1#2{%
    \expandafter\let\expandafter\@tmpa\csname r@#1\endcsname%
    \edef\@tmpa{\expandafter\@firstoffive\@tmpa}%
    \expandafter\@cref@getcounter\@tmpa\@nil#2}
  \def\cref@getprefix#1#2{%
    \expandafter\let\expandafter\@tmpa\csname r@#1\endcsname%
```

```
    \edef\@tmpa{\expandafter\@firstoffive\@tmpa}%
    \expandafter\@cref@getprefix\@tmpa\@nil#2}
```

`\H@refstepcounter` The `hyperref` package stores the original `\refstepcounter` definition as `\H@refstepcounter`, which we therefore need to modify so that it adds the extra information to `\@currentlabel`.

```
\def\H@refstepcounter#1{%
  \stepcounter{#1}%
  \cref@constructprefix{#1}{\@result}%
  \protected@edef\@currentlabel{%
    [#1][\arabic{#1}][\@result]%
    \csname p@#1\endcsname\csname the#1\endcsname}}
```

`\refstepcounter@noarg` `\refstepcounter@optarg` The original `\refstepcounter`, as stored earlier in `\cref@old@refstepcounter`, already calls `\H@refstepcounter` if `hyperref` is loaded, and we just redefined the latter to store the type information. So we only need to change `\@currentlabel` in our `\refstepcounter` if an optional argument was supplied.

```
\def\refstepcounter@noarg#1{\cref@old@refstepcounter{#1}}
\def\refstepcounter@optarg[#1]#2{%
  \cref@old@refstepcounter{#2}%
  \expandafter\@cref@getlabel\@currentlabel\@nil{\@tmplabel}%
  \cref@constructprefix{#2}{\@tmpreset}%
  \protected@edef\@currentlabel{%
    [#1][\arabic{#2}][\@tmpreset]\@tmplabel}}
```

`\appendix` We again make `\appendix` redefine things so that the label type for chapters or sections is exceptionally overridden and set to “appendix” instead. But this time, it is `\H@refstepcounter` that needs to be redefined.

```
\renewcommand\appendix{%
  \cref@old@appendix%
  \@ifundefined{chapter}{%
    \def\H@refstepcounter##1{%
      \stepcounter{##1}%
      \cref@constructprefix{##1}{\@result}%
```

We add a large value to the front of the counter data, to force references to anything in appendices to be sorted after everything else.

```
      \edef\@result{{2147483647}\@result}%
```

Override the cross-reference type of sectioning commands.

```
      \def\@tmpa{##1}%
      \def\@tmpb{section}%
      \ifx\@tmpa\@tmpb%
        \protected@edef\@currentlabel{%
          [appendix][\arabic{##1}][\@result]%
          \csname p@##1\endcsname\csname the##1\endcsname}%
      \else%
        \def\@tmpa{##1}%
```

```
      \def\@tmpb{subsection}%
      \ifx\@tmpa\@tmpb%
        \protected@edef\@currentlabel{%
          [subappendix][\arabic{##1}][\@result]%
          \csname p@##1\endcsname\csname the##1\endcsname}%
      \else%
        \def\@tmpa{##1}%
        \def\@tmpb{subsubsection}%
        \ifx\@tmpa\@tmpb%
          \protected@edef\@currentlabel{%
            [subsubappendix][\arabic{##1}][\@result]%
            \csname p@##1\endcsname\csname the##1\endcsname}%
        \else%
          \protected@edef\@currentlabel{%
            [##1][\arabic{##1}][\@result]%
            \csname p@##1\endcsname\csname the##1\endcsname}%
        \fi%
      \fi%
    \fi}%
  }{%
    \def\H@refstepcounter##1{%
      \stepcounter{##1}%
      \cref@constructprefix{##1}{\@result}%
```

Again, the large value added to the front of the counter data forces references to appendix items to be sorted last.

```
      \edef\@result{{2147483647}\@result}%
```

Override the cross-reference type of sectioning commands.

```
      \def\@tmpa{##1}%
      \def\@tmpb{chapter}%
      \ifx\@tmpa\@tmpb%
        \protected@edef\@currentlabel{%
          [appendix][\arabic{##1}][\@result]%
          \csname p@##1\endcsname\csname the##1\endcsname}%
      \else%
        \def\@tmpa{##1}%
        \def\@tmpb{section}%
        \ifx\@tmpa\@tmpb%
          \protected@edef\@currentlabel{%
            [subappendix][\arabic{##1}][\@result]%
            \csname p@##1\endcsname\csname the##1\endcsname}%
        \else%
          \def\@tmpa{##1}%
          \def\@tmpb{subsection}%
          \ifx\@tmpa\@tmpb%
            \protected@edef\@currentlabel{%
              [subsubappendix][\arabic{##1}][\@result]%
              \csname p@##1\endcsname\csname the##1\endcsname}%
          \else%
            \def\@tmpa{##1}%
```

```
            \def\@tmpb{subsubsection}%
            \ifx\@tmpa\@tmpb%
              \protected@edef\@currentlabel{%
                [subsubsubappendix][\arabic{##1}][\@result]%
                \csname p@##1\endcsname\csname the##1\endcsname}%
            \else%
              \protected@edef\@currentlabel{%
                [##1][\arabic{##1}][\@result]%
                \csname p@##1\endcsname\csname the##1\endcsname}%
            \fi%
          \fi%
        \fi%
      \fi}%
  }%
}
```

`\cref*` `\Cref*` `\crefrange*` `\Crefrange*` `\@crefstar` `\@crefrangestar` `\@crefrangenostar`

Redefine `\cref` and all the others to allow starred variants, which don't create hyper-links. The starred variants simply set a flag, which is tested in `\@@setcref` and `\@@setrangeref` (below).

```
\newif\if@crefstarred
\DeclareRobustCommand{\cref}{%
  \@ifstar{\@crefstar{cref}}{\@crefnostar{cref}}}
\DeclareRobustCommand{\Cref}{%
  \@ifstar{\@crefstar{Cref}}{\@crefnostar{Cref}}}
\def\@crefnostar#1#2{\@cref{#1}{#2}}
\def\@crefstar#1#2{%
  \@crefstarredtrue\@cref{#1}{#2}\@crefstarredfalse}
\DeclareRobustCommand{\crefrange}{%
  \@ifstar{\@crefrangestar{cref}}{\@crefrangenostar{cref}}}
\DeclareRobustCommand{\Crefrange}{%
  \@ifstar{\@crefrangestar{Cref}}{\@crefrangenostar{Cref}}}
\def\@crefrangenostar#1#2#3{\@setcrefrange{#2}{#3}{#1}{}}
\def\@crefrangestar#1#2#3{%
  \@crefstarredtrue\@setcrefrange{#2}{#3}{#1}{}\@crefstarredfalse}
```

`\@@setcref` `\@@setcrefrange`

Redefine `\@@setcref` and `\@@setrangeref` to create hyper-links (unless the starred flag is set), using the extra arguments supplied in `\r@`⟨*label*⟩ (via the aux file) by `hyperref`.

```
\def\@@setcref#1#2{%
  \cref@getlabel{#2}{\@tmplabel}%
  \if@crefstarred%
    #1{\@tmplabel}{}{}%
  \else%
    \edef\@tmplink{\cref@hyperref{#2}}%
    #1{\@tmplabel}{\hyper@linkstart{link}{\@tmplink}}{\hyper@linkend}%
  \fi}
\def\@@setcrefrange#1#2#3{%
  \cref@getlabel{#2}{\@labela}%
  \cref@getlabel{#3}{\@labelb}%
```

```
    \if@crefstarred%
      #1{\@labela}{\@labelb}{}{}{}{}%
    \else%
      \edef\@linka{\cref@hyperref{#2}}%
      \edef\@linkb{\cref@hyperref{#3}}%
      #1{\@labela}{\@labelb}%
        {\hyper@linkstart{link}{\@linka}}{\hyper@linkend}%
        {\hyper@linkstart{link}{\@linkb}}{\hyper@linkend}%
    \fi}%
  }{}%  end of \@ifpackageloaded
```

`\ref*` Redefine `\ref` command to provide a starred variant with the same behaviour as `hyperref`'s `\ref*`. As before, we defer the redefinition until the beginning of the document, to ensure it 'takes'. (`hyperref` stores the original `\@setref` in `\real@setref`.)

```
\AtBeginDocument{%
 \DeclareRobustCommand{\ref}{\@ifstar\cref@refstar\cref@ref}
 \def\cref@refstar#1{%
   \expandafter\ifx\csname r@#1\endcsname\relax%
     \let\@result\relax%
   \else%
     \cref@getlabel{#1}{\@result}%
   \fi%
   \expandafter\real@setref\csname r@#1\endcsname{\cref@reflabel}{#1}}%
}
```

10.6 `ntheorem` Support

`ntheorem` `thref` If `ntheorem` is loaded, we need to modify its theorem referencing features so that they work with `cleveref`.

```
  \@ifpackageloaded{ntheorem}{%
  \PackageInfo{cleveref}{'ntheorem' support loaded}
  \@ifpackagewith{ntheorem}{thref}{%
    \PackageWarning{cleveref}{'cleveref' supersedes 'ntheorem's 'thref'
      option}%
    \renewcommand{\thref}{\cref}}{}
```

`\theorem@prework` Newer versions of `ntheorem` require a call to `\theorem@prework` when type-setting theorems. If an older version of `ntheorem` is being used, we just `\let` it to `\relax` to make sure it's defined.

```
  \@ifundefined{theorem@prework}{\let\theorem@prework\relax}{}
```

`\@thm` We modify `ntheorem`'s version of the `\@thm` macro very slightly, to have it call `\refstepcounter` with an optional argument containing the theorem type.

```
  \gdef\@thm#1#2#3{%
    \if@thmmarks%
      \stepcounter{end\InTheoType ctr}%
    \fi%
```

```
    \renewcommand{\InTheoType}{#1}%
    \if@thmmarks%
      \stepcounter{curr#1ctr}%
      \setcounter{end#1ctr}{0}%
    \fi%
    \refstepcounter[#1]{#2}%  <<<<<
    \theorem@prework%
    \thm@topsepadd \theorempostskipamount%
    \ifvmode \advance\thm@topsepadd\partopsep\fi%
    \trivlist%
    \@topsep \theorempreskipamount%
    \@topsepadd \thm@topsepadd%
    \advance\linewidth -\theorem@indent%
    \advance\@totalleftmargin \theorem@indent%
    \parshape \@ne \@totalleftmargin \linewidth%
    \@ifnextchar[{\@ythm{#1}{#2}{#3}}{\@xthm{#1}{#2}{#3}}%]
  }}{}%  end of \@ifpackageloaded
```

10.7 `varioref` Support

`varioref` If `varioref` is loaded, we redefine its commands to use `\cref` instead of `\ref` to produce the reference. Since `\cref` can cope with multiple references, We extend the page referencing magic of `\vref` et al. so that they check whether they need to use `\vpagerefrange` instead of `\vpageref`.

```
\@ifpackageloaded{varioref}{%
    \PackageInfo{cleveref}{`varioref' support loaded}
    \PackageInfo{cleveref}{`cleveref' supersedes `varioref's %
      $\backslash$\labelformat command, which will not work}
  \AtBeginDocument{%
    \def\cref@vref#1#2{%
```

Since we're modifying the `varioref` commands anyway, we also (by default) take this opportunity to get rid of the irritating spacing issues of `\vref` et al. However, this breaks strict compatibility with the original `varioref` spacing behaviour, so we also provide a `legacyvarioref` option to restore the spacing behaviour, in case full compatibility is required.

```
      \if@cref@legacyvarioref%
        \leavevmode\unskip\vref@space
      \fi%
      \@cref{#1}{#2} % space here is deliberate
      \begingroup%
        \def\@tmpstack{#2,\@nil}%
        \cref@stack@topandbottom{\@tmpstack}{\@firstref}{\@lastref}%
        \ifx\@lastref\@empty%
          \vpageref{#2}%
        \else%
          \edef\@tmpa{{\@firstref}{\@lastref}}%
          \expandafter\vpagerefrange\@tmpa%
```

```
      \fi%
    \endgroup}
  \def\cref@vrefrange#1#2#3{%
    \@setcrefrange{#2}{#3}{#1}{} \vpagerefrange{#2}{#3}}
  \def\cref@fullref#1#2{%
    \@cref{#1}{#2} % space here is deliberate
    \begingroup%
      \def\@tmpstack{#2,\@nil}%
      \cref@stack@topandbottom{\@tmpstack}{\@firstref}{\@lastref}%
      \ifx\@lastref\@empty%
        \reftextfaraway{#2}%
      \else%
        \expandafter\vrefpagenum\expandafter%
          \@tmpa\expandafter{\@firstref}%
        \expandafter\vrefpagenum\expandafter%
          \@tmpb\expandafter{\@lastref}%
        \ifx\@tmpa\@tmpb
          \expandafter\reftextfaraway\expandafter{\@firstref}%
        \else
          \edef\@tmpa{{\@firstref}{\@lastref}}%
          \expandafter\reftextpagerange\@tmpa%
        \fi%
      \fi%
    \endgroup}
```

`\vref` `\vref*` `\vrefrange` `\fullref` `\vr@f` `\Vr@f`

If `legacyvarioref` is set, we only modify the original `varioref` commands, and don't define any new ones.

```
  \if@cref@legacyvarioref%
    \def\vr@f#1{\cref@vref{cref}{#1}}
    \def\Vr@f#1{\cref@vref{Cref}{#1}}
    \renewcommand\vrefrange[3][\reftextcurrent]{%
      \crefrange{#2}{#3} \vpagerefrange{#2}{#3}}
    \def\fullref#1{\cref@fullref{cref}{#1}}
```

`\vref` `\vref*` `\Vref` `\Vref*` `\vrefrange` `\vrefrange*` `\Vrefrange` `\Vrefrange*` `\fullref` `\fullref*` `\Fullref` `\Fullref*`

If we're not providing legacy compatibility with `varioref`, we define `\vref` et al. to be consistent with the other `cleveref` referencing commands. This frees up the starred variants to be used to suppress hyperlinks when `hyperref` is loaded, as usual.

```
  \else%
    \@ifpackageloaded{hyperref}{%
      \DeclareRobustCommand{\vref}{%
        \@ifstar{\cref@vrefstar{cref}}{\cref@vref{cref}}}
      \DeclareRobustCommand{\Vref}{%
        \@ifstar{\cref@vrefstar{Cref}}{\cref@vref{Cref}}}
      \DeclareRobustCommand{\vrefrange}{%
        \@ifstar{\cref@vrefrangestar{cref}}{\cref@vrefrange{cref}}}
      \DeclareRobustCommand{\Vrefrange}{%
        \@ifstar{\cref@vrefrangestar{Cref}}{\cref@vrefrange{Cref}}}%
```

```
        \DeclareRobustCommand{\fullref}{%
          \@ifstar{\cref@fullrefstar{cref}}{\cref@fullref{cref}}}
        \DeclareRobustCommand{\Fullref}{%
          \@ifstar{\cref@fullrefstar{Cref}}{\cref@fullref{Cref}}}
        \def\cref@vrefstar#1#2{%
          \@crefstarredtrue%
          \cref@vref{#1}{#2}%
          \@crefstarredfalse}
        \def\cref@vrefrangestar#1#2#3{%
          \@crefstarredtrue%
          \cref@vrefrange{#1}{#2}{#3}%
          \@crefstarredfalse}
        \def\cref@fullrefstar#1#2{%
          \@crefstarredtrue%
          \cref@fullref{#1}{#2}%
          \@crefstarredfalse}
      }{%
        \DeclareRobustCommand{\vref}{\cref@vref{cref}}
        \DeclareRobustCommand{\Vref}{\cref@vref{Cref}}
        \DeclareRobustCommand{\vrefrange}{\cref@vrefrange{cref}}
        \DeclareRobustCommand{\Vrefrange}{\cref@vrefrange{Cref}}
        \DeclareRobustCommand{\fullref}{\cref@fullref{cref}}
        \DeclareRobustCommand{\Fullref}{\cref@fullref{Cref}}
      }
    \fi%
   }%  end of \AtBeginDocument
 }{}%  end of \@ifpackageloaded
%   \end{macrocode}
%
% \begin{macro}{legacyvarioref}
%   The \option{legacyvarioref} option just sets a flag, checked in the
%   redefinitions set up at the beginning of the document, above.
% \end{macro}
%    \begin{macrocode}
\let\if@cref@legacyvarioref\iffalse
\DeclareOption{legacyvarioref}{%
  \PackageInfo{cleveref}{legacy `varioref' compatibility enabled}
  \let\if@cref@legacyvarioref\iftrue}
```

10.8 Poor Man's `cleveref`

`poorman` The `poorman` option causes a `sed` script to automatically be written. When the original LaTeX source file is processed through this script, it strips out all the `cleveref` commands, type-setting all the reference formatting explicitly, and using the standard `\ref` command to produce the references themselves.

```
\DeclareOption{poorman}{%
  \PackageInfo{cleveref}{option `poorman' loaded}
```

`\cref@poorman@text` Define global macro `\cref@poorman@text` to store the text produced by the `\cref`

commands, and open an output stream for writing the script before starting to process the document body.

```
\gdef\cref@poorman@text{}
\AtBeginDocument{%
  \newwrite\@crefscript%
  \immediate\openout\@crefscript=\jobname.sed}
```

select@language
forreign@language

If `babel` is loaded, we add to the `\select@language` and `\forreign@language` commands to make them write substitution rules to the script that replace the cross-reference name and conjunction component macros with the appropriate language-dependent names. We use `sed` line-number addresses in the rules to ensure they are only applied to the regions in which that particular language was in use.

Note that we write substitution rules for the *previou* language block when the language is changed, because we need the rules to appear in the script *after* all the cross-reference substitution rules for that language block. `\ref@inputlineno` stores the input-file line-number of the start of the previous language block.

We postpone the redefinitions until the beginning of the document not only to ensure that they don't get clobbered by other package's redefinitions, but also because we don't want the redefenitions to take effect until after `babel` has called `\selectlanguage` for the main language (remember, the substitution rules for this first language block will get written at the next language change).

Note that, since we're writing to the script file within `\AtBeginDocument` and `\AtEndDocument`, this code has to come *after* the above `\AtBeginDocument` code which opens the script file for writing, and *before* the later `\AtEndDocument` code (below) which closes it.

The `\if@cref@switched@language` flag is set when a `babel` language switching command is called. It is checked by `\cref@writelanguagerules` when writing substitution rules.

```
\newif\if@cref@switched@language
\@ifpackageloaded{babel}{%
  \AtBeginDocument{%
    \let\cref@old@select@language\select@language
    \def\select@language{%
      \@cref@switched@languagetrue%
      \cref@writelanguagerules%
      \cref@old@select@language}
    \let\cref@old@forreign@language\forreign@language
    \def\forreign@language{%
      \@cref@switched@languagetrue%
      \cref@writelanguagerules%
      \cref@old@forreign@language}
    \edef\cref@inputlineno{\the\inputlineno}}
```

The final set of substitution rules gets written at the end of the document.

```
  \AtEndDocument{%
    \let\select@language\cref@old@select@language%
```

```
      \let\forreign@language\cref@old@forreign@language%
      \cref@writelanguagerules}
```

`\cref@writelanguagerules` `\cref@writelanguagerules` does the grunt work of writing out the necessary substitution rules.

```
    \def\cref@writelanguagerules{%
      \begingroup%
```

If `\if@cref@switched@language` hasn't been set, then we must be writing the final set of substitution rules at the end of a document, in which no language switching command was ever used. In which case, the substitution rules don't specify a line-number address.

```
        \if@cref@switched@language%
          \edef\@address{\cref@inputlineno,\the\inputlineno}%
        \else%
          \def\@address{}%
        \fi%
        \expandafter\def\expandafter\cref@poorman@text\expandafter{%
          \crefrangeconjunction}%
        \expandafter\def\expandafter\@tmpa\expandafter{%
          \expandafter{\@address}{\string\crefrangeconjunction}}
        \expandafter\cref@writescript\@tmpa%
        \expandafter\def\expandafter\cref@poorman@text\expandafter{%
          \crefpairconjunction}%
        \expandafter\def\expandafter\@tmpa\expandafter{%
          \expandafter{\@address}{\string\crefpairconjunction}}
        \expandafter\cref@writescript\@tmpa%
        \expandafter\def\expandafter\cref@poorman@text\expandafter{%
          \crefmiddleconjunction}%
        \expandafter\def\expandafter\@tmpa\expandafter{%
          \expandafter{\@address}{\string\crefmiddleconjunction}}
        \expandafter\cref@writescript\@tmpa%
        \expandafter\def\expandafter\cref@poorman@text\expandafter{%
          \creflastconjunction}%
        \expandafter\def\expandafter\@tmpa\expandafter{%
          \expandafter{\@address}{\string\creflastconjunction}}
        \expandafter\cref@writescript\@tmpa%
        \expandafter\def\expandafter\cref@poorman@text\expandafter{%
          \crefpairgroupconjunction}%
        \expandafter\def\expandafter\@tmpa\expandafter{%
          \expandafter{\@address}{\string\crefpairgroupconjunction}}
        \expandafter\cref@writescript\@tmpa%
        \expandafter\def\expandafter\cref@poorman@text\expandafter{%
          \crefmiddlegroupconjunction}%
        \expandafter\def\expandafter\@tmpa\expandafter{%
          \expandafter{\@address}{\string\crefmiddlegroupconjunction}}
        \expandafter\cref@writescript\@tmpa%
        \expandafter\def\expandafter\cref@poorman@text\expandafter{%
          \creflastgroupconjunction}%
        \expandafter\def\expandafter\@tmpa\expandafter{%
```

```
      \expandafter{\@address}{\string\creflastgroupconjunction}}
    \expandafter\cref@writescript\@tmpa%
```

We write substitution rules for all component-derived cross-reference formats, as listed in `\cref@label@types`.

```
    \let\@tmpstack\cref@label@types%
    \cref@isstackfull{\@tmpstack}%
    \@whilesw\if@cref@stackfull\fi{%
```

`\cref@`⟨*type*⟩`@name` substitution rules.

```
      \edef\@tmpa{\cref@stack@top{\@tmpstack}}%
      \expandafter\expandafter\expandafter\def%
      \expandafter\expandafter\expandafter\cref@poorman@text%
      \expandafter\expandafter\expandafter{%
        \csname cref@\@tmpa @name\endcsname}%
      \edef\@tmpa{%
        \string\cref@\expandafter\noexpand\@tmpa @name\space}%
      \expandafter\expandafter\expandafter\def%
      \expandafter\expandafter\expandafter\@tmpa%
      \expandafter\expandafter\expandafter{%
        \expandafter\expandafter\expandafter{%
          \expandafter\@address\expandafter}%
        \expandafter{\@tmpa}}
      \expandafter\cref@writescript\@tmpa%
```

`\cref@`⟨*type*⟩`@name@plural` substitution rules.

```
      \edef\@tmpa{\cref@stack@top{\@tmpstack}}%
      \expandafter\expandafter\expandafter\def%
      \expandafter\expandafter\expandafter\cref@poorman@text%
      \expandafter\expandafter\expandafter{%
        \csname cref@\@tmpa @name@plural\endcsname}%
      \edef\@tmpa{%
        \string\cref@\expandafter\noexpand\@tmpa%
        @name@plural\space}%
      \expandafter\expandafter\expandafter\def%
      \expandafter\expandafter\expandafter\@tmpa%
      \expandafter\expandafter\expandafter{%
        \expandafter\expandafter\expandafter{%
          \expandafter\@address\expandafter}%
        \expandafter{\@tmpa}}
      \expandafter\cref@writescript\@tmpa%
```

`\Cref@`⟨*type*⟩`@name` substitution rules.

```
      \edef\@tmpa{\cref@stack@top{\@tmpstack}}%
      \expandafter\expandafter\expandafter\def%
      \expandafter\expandafter\expandafter\cref@poorman@text%
      \expandafter\expandafter\expandafter{%
        \csname Cref@\@tmpa @name\endcsname}%
      \edef\@tmpa{%
        \string\Cref@\expandafter\noexpand\@tmpa @name\space}%
      \expandafter\expandafter\expandafter\def%
```

```
          \expandafter\expandafter\expandafter\@tmpa%
          \expandafter\expandafter\expandafter{%
            \expandafter\expandafter\expandafter%
            {\expandafter\@address\expandafter}%
            \expandafter{\@tmpa}}
          \expandafter\cref@writescript\@tmpa%
```

`\Cref@`⟨*type*⟩`@name@plural` substitution rules.

```
          \edef\@tmpa{\cref@stack@top{\@tmpstack}}%
          \expandafter\expandafter\expandafter\def%
          \expandafter\expandafter\expandafter\cref@poorman@text%
          \expandafter\expandafter\expandafter{%
            \csname Cref@\@tmpa @name@plural\endcsname}%
          \edef\@tmpa{%
            \string\Cref@\expandafter\noexpand\@tmpa%
            @name@plural\space}%
          \expandafter\expandafter\expandafter\def%
          \expandafter\expandafter\expandafter\@tmpa%
          \expandafter\expandafter\expandafter{%
            \expandafter\expandafter\expandafter%
            {\expandafter\@address\expandafter}%
            \expandafter{\@tmpa}}
          \expandafter\cref@writescript\@tmpa%
```

After the loop over cross-reference types, we set `\cref@inputlineno` to the current input-file line, in preparation for the next language block.

```
          \cref@stack@pop{\@tmpstack}%
          \cref@isstackfull{\@tmpstack}}%
      \endgroup%
      \edef\cref@inputlineno{\the\inputlineno}}%
 }{}% end of \@ifpackageloaded
```

After processing the document body, we re-read in the temporary script file, and write it out again to the final `sed` script file, escaping regexp special characters in the process. The escaping is carried out by turning the regexp special characters into active characters, and defining them to expand to their escaped form. This involves a lot of juggling of catcodes and lccodes!

Both `\DeclareOption` and `\AtEndDocument` store their arguments in token lists, so all the following TEXcode is already tokenised long before it is expanded and evaluated. Thus there is no (easy) way to change the catcodes of the characters appearing here before they are tokenised. In one way this is convenient: the catcode changes we make don't "take" until evaluated, so we can continue to use the standard TEXcharacters (\, {, } etc.) even after the lines containing the catcode commands. But in another, more significant, way, it is very inconvenient: it makes it difficult to define the regexp special characters as active characters, since it's impossible to directly create tokens with the correct char- and catcodes.

We get around this by creating the unusual charcode/catcode combinations using the `\lowercase` trick (`\lowercase` changes the charcodes of all characters in its argument to their lccodes, but *leaves* their catcodes alone). That way,

the argument of `\AtEndDocument` is tokenised correctly, and when it comes to be expanded and evaluated, the `\lowercase` commands create tokens with the correct char- and catcodes.

```
\AtEndDocument{%
  \immediate\closeout\@crefscript%
  \newread\@crefscript%
  \immediate\openin\@crefscript=\jobname.sed%
  \begingroup%
    \newif\if@not@eof%
    \def\@eof{\par }%
```

Change catcodes of regexp special characters to make them active characters and define them to expand to their escaped forms. Change those of TEXspecial characters to make them normal letters.

```
    \catcode`.=13 \catcode`*=13
    \catcode`[=13 \catcode`]=13
    \catcode`^=13 \catcode`$=13 %$
    \catcode`\=0 \catcode`<=1 \catcode`>=2
    \catcode`\\=13 \catcode`\{=12 \catcode`\}=12 \catcode`_=12
    \lccode`/=92
    \lccode`~=92\lowercase{\def~{\string/\string/}}%
    \lccode`~=42\lowercase{\def~{\string/\string*}}%
    \lccode`~=46\lowercase{\def~{\string/\string.}}%
    \lccode`~=91\lowercase{\def~{\string/\string[}}%
    \lccode`~=93\lowercase{\def~{\string/\string]}}%
    \lccode`~=94\lowercase{\def~{\string/\string^}}%
    \lccode`~=36\lowercase{\def~{\string/\string$}}% $
    \lccode`~=0 \lccode`/=0 \catcode`~=12
```

Read lines from the temporary script file, expand them to escape regexp special characters, and store them in `\cref@poorman@text`.

```
    \def\cref@poorman@text{}%
    \immediate\read\@crefscript to \@tmpa%
    \ifx\@tmpa\@eof%
      \@not@eoffalse%
    \else%
      \@not@eoftrue%
      \edef\@tmpa{\@tmpa}%
    \fi%
    \@whilesw\if@not@eof\fi{%
      \expandafter\g@addto@macro\expandafter%
        \cref@poorman@text\expandafter{\@tmpa^^J}%
      \immediate\read\@crefscript to \@tmpa%
      \ifx\@tmpa\@eof%
        \@not@eoffalse%
      \else%
        \@not@eoftrue%
        \edef\@tmpa{\@tmpa}%
      \fi}%
  \endgroup%
```

```
\immediate\closein\@crefscript%
```

Add some rules to remove other `cleveref` commands. We use the `\lowercase` trick again for writing the \, { and } characters. (This could be done in other ways, but since we're in `\lowercase` mood, why not stick with it.)

```
\begingroup%
  \lccode`|=92 \lccode`<=123 \lccode`>=125 \lccode`C=67
  \lowercase{\def\@tmpa{%[
      s/||label|[[^]]*|]/||label/g}}
  \expandafter\g@addto@macro\expandafter%
    \cref@poorman@text\expandafter{\@tmpa^^J}%
  \lowercase{\edef\@tmpa{s/||usepackage|(|[.*|]|)|?<cleveref>//g}}%
  \expandafter\g@addto@macro\expandafter%
    \cref@poorman@text\expandafter{\@tmpa^^J}%
  \lowercase{\edef\@tmpa{s/||[cC]refformat<.*><.*>//g}}%
  \expandafter\g@addto@macro\expandafter%
    \cref@poorman@text\expandafter{\@tmpa^^J}%
  \lowercase{\edef\@tmpa{s/||[cC]refrangeformat<.*><.*>//g}}%
  \expandafter\g@addto@macro\expandafter%
    \cref@poorman@text\expandafter{\@tmpa^^J}%
  \lowercase{\edef\@tmpa{s/||[cC]refmultiformat<.*><.*><.*><.*>//g}}%
  \expandafter\g@addto@macro\expandafter%
    \cref@poorman@text\expandafter{\@tmpa^^J}%
  \lowercase{\edef\@tmpa{%
      s/||[cC]refrangemultiformat<.*><.*><.*><.*>//g}}%
  \expandafter\g@addto@macro\expandafter%
    \cref@poorman@text\expandafter{\@tmpa^^J}%
  \lowercase{\edef\@tmpa{s/||[cC]refname<.*><.*>//g}}%
  \expandafter\g@addto@macro\expandafter%
    \cref@poorman@text\expandafter{\@tmpa^^J}%
  \lowercase{\edef\@tmpa{s/||[cC]reflabelformat<.*><.*>//g}}%
  \expandafter\g@addto@macro\expandafter%
    \cref@poorman@text\expandafter{\@tmpa^^J}%
  \lowercase{\edef\@tmpa{s/||[cC]refrangelabelformat<.*><.*>//g}}%
  \expandafter\g@addto@macro\expandafter%
    \cref@poorman@text\expandafter{\@tmpa^^J}%
  \lowercase{\edef\@tmpa{s/||[cC]refdefaultlabelformat<.*>//g}}%
  \expandafter\g@addto@macro\expandafter%
    \cref@poorman@text\expandafter{\@tmpa^^J}%
  \lowercase{\edef\@tmpa{%
      s/||renewcommand<||crefpairconjunction><.*>//g}}%
  \expandafter\g@addto@macro\expandafter%
    \cref@poorman@text\expandafter{\@tmpa^^J}%
  \lowercase{\edef\@tmpa{%
      s/||renewcommand<||crefpairgroupconjunction><.*>//g}}%
  \expandafter\g@addto@macro\expandafter%
    \cref@poorman@text\expandafter{\@tmpa^^J}%
  \lowercase{\edef\@tmpa{%
      s/||renewcommand<||crefmiddleconjunction><.*>//g}}%
  \expandafter\g@addto@macro\expandafter%
```

```
      \cref@poorman@text\expandafter{\@tmpa^^J}%
    \lowercase{\edef\@tmpa{%
        s/||renewcommand<||crefmiddlegroupconjunction><.*>//g}}%
    \expandafter\g@addto@macro\expandafter%
      \cref@poorman@text\expandafter{\@tmpa^^J}%
    \lowercase{\edef\@tmpa{%
        s/||renewcommand<||creflastconjunction><.*>//g}}%
    \expandafter\g@addto@macro\expandafter%
      \cref@poorman@text\expandafter{\@tmpa^^J}%
    \lowercase{\edef\@tmpa{%
        s/||renewcommand<||creflastgroupconjunction><.*>//g}}%
    \expandafter\g@addto@macro\expandafter%
      \cref@poorman@text\expandafter{\@tmpa^^J}%
    \lowercase{\edef\@tmpa{s/||renewcommand<||[cC]ref><.*>//g}}%
    \expandafter\g@addto@macro\expandafter%
      \cref@poorman@text\expandafter{\@tmpa^^J}%
    \lowercase{\edef\@tmpa{s/||renewcommand<||[cC]refrange><.*>//g}}%
    \expandafter\g@addto@macro\expandafter%
      \cref@poorman@text\expandafter{\@tmpa^^J}%
  \endgroup%
```

Overwrite the script file with the new, escaped regexp rules.

```
  \newwrite\@crefscript%
  \immediate\openout\@crefscript=\jobname.sed%
  \immediate\write\@crefscript{\cref@poorman@text}%
  \immediate\closeout\@crefscript%
}
```

`\cref@writescript` The `\cref@writescript` utility macro does the actual writing of the substitution rule to the script. The first argument is the "address", the second argument is the regexp pattern to match, whilst the substitution must be stored in `\cref@poorman@text`.

```
\def\cref@getmeaning#1{\expandafter\@cref@getmeaning\meaning#1\@nil}
\def\@cref@getmeaning#1->#2\@nil{#2}
\def\cref@writescript#1#2{%
  \edef\@tmpa{\cref@getmeaning{\cref@poorman@text}}%
  \immediate\write\@crefscript{#1 s/#2/\@tmpa/g}}
```

`\cref` `\Cref` `\crefrange` `\Crefrange` `\@crefstar` `\@crefnostar` `\@crefrangestar` `\@crefrangenostar` Redefine the user-level referencing commands so that they write a substitution rule for the reference to the script, as well as type-setting the reference itself.

```
\@ifpackageloaded{hyperref}{%
  \def\@crefnostar#1#2{%
    \gdef\cref@poorman@text{}%
    \@cref{#1}{#2}%
    \begingroup%
      \lccode`|=92 \lccode`<=123 \lccode`>=125 \lccode`C=67
      \lowercase{\cref@writescript{}{|#1<#2>}}%
    \endgroup}
  \def\@crefstar#1#2{%
    \gdef\cref@poorman@text{}%
```

```
      \@crefstarredtrue\@cref{#1}{#2}\@crefstarredfalse%
      \begingroup%
        \lccode`|=92 \lccode`<=123 \lccode`>=125 \lccode`C=67
        \lowercase{\cref@writescript{}{|#1*<#2>}}%
      \endgroup}
    \def\@crefrangenostar#1#2#3{%
      \gdef\cref@poorman@text{}%
      \@setcrefrange{#2}{#3}{#1}{}%
      \begingroup%
        \lccode`|=92 \lccode`<=123 \lccode`>=125 \lccode`C=67
        \lowercase{\cref@writescript{}{|#1range<#2><#3>}}%
      \endgroup}
    \def\@crefrangestar#1#2#3{%
      \gdef\cref@poorman@text{}%
      \@crefstarredtrue\@setcrefrange{#2}{#3}{#1}{}\@crefstarredfalse%
      \begingroup%
        \lccode`|=92 \lccode`<=123 \lccode`>=125 \lccode`C=67
        \lowercase{\cref@writescript{}{|#1range*<#2><#3>}}%
      \endgroup}
%
  }{%
    \DeclareRobustCommand{\cref}[1]{%
      \edef\cref@poorman@text{}%
      \@cref{cref}{#1}%
      \cref@writescript{}{\string\cref\string{#1\string}}}
    \DeclareRobustCommand{\Cref}[1]{%
      \edef\cref@poorman@text{}%
      \@cref{Cref}{#1}%
      \cref@writescript{}{\string\Cref\string{#1\string}}}
    \DeclareRobustCommand{\crefrange}[2]{%
      \edef\cref@poorman@text{}%
      \@setcrefrange{#1}{#2}{cref}{}%
      \cref@writescript{}{%
        \string\crefrange\string{#1\string}\string{#2\string}}}
    \DeclareRobustCommand{\Crefrange}[2]{%
      \edef\cref@poorman@text{}%
      \@setcrefrange{#1}{#2}{Cref}{}%
      \cref@writescript{}{%
        \string\Crefrange\string{#1\string}\string{#2\string}}}
  }
```

\vref
\vref*
\Vref
\Vref*

If `varioref` is loaded, do the same for the `\vref` et al. commands. Note that we now need to set the `\if@crefstarred` flag for the starred variant `\vref*` even if `legacyvarioef` is set and it changes the spacing rather than suppressing hyperlinks, so that we can write the correct substitution rule.

\vrefrange
\vrefrange*
\Vrefrange
\Vrefrange*
\fullref
\fullref*
\Fullref
\Fullref*

```
  \@ifpackageloaded{varioref}{%
    \AtBeginDocument{%
      \if@cref@legacyvarioref%
        \DeclareRobustCommand{\vref}{%
          \@ifstar{\cref@vrefstar{cref}}{\cref@vref{cref}}}
```

```
      \def\cref@vrefstar#1#2{%
        \@crefstarredtrue\cref@vref{#1}{#2}\@crefstarredfalse}
    \fi
    \def\cref@vref#1#2{%
      \gdef\cref@poorman@text{}%
      \if@cref@legacyvarioref%
        \leavevmode\unskip\vref@space
      \fi%
      \begingroup%
```

If `legacyvarioref` is set, `\vref*` shouldn't suppress hyper-links, so temporarily disable then restore the flag whilst type-setting the cross-reference.

```
        \let\if@tmp\if@crefstarred%
        \if@cref@legacyvarioref\@crefstarredfalse\fi%
        \@cref{#1}{#2} % space here is deliberate
        \let\if@crefstarred\if@tmp%
        \def\@tmpstack{#2,\@nil}%
        \cref@stack@topandbottom{\@tmpstack}{\@firstref}{\@lastref}%
        \ifx\@lastref\@empty%
          \vpageref{#2}%
          \g@addto@macro\cref@poorman@text{ \vpageref{#2}}%
        \else%
          \g@addto@macro\cref@poorman@text{ }%
          \edef\@tmpa{{\@firstref}{\@lastref}}%
          \expandafter\vpagerefrange\@tmpa%
          \expandafter\g@addto@macro\expandafter\cref@poorman@text%
            \expandafter{\expandafter\vpagerefrange\@tmpa}%
        \fi%
        \def\@tmpa##1##2\@nil{%
          \if##1c%
            \if@crefstarred%
              \cref@writescript{}{\string\vref*\string{#2\string}}%
            \else%
              \cref@writescript{}{\string\vref\string{#2\string}}%
            \fi%
          \else%
            \if@crefstarred%
              \cref@writescript{}{\string\Vref*\string{#2\string}}%
            \else%
              \cref@writescript{}{\string\Vref\string{#2\string}}%
            \fi%
          \fi}%
        \@tmpa#1\@nil%
      \endgroup}
%
    \def\cref@vrefrange#1#2#3{%
      \gdef\cref@poorman@text{}%
      \@setcrefrange{#2}{#3}{#1}{} \vpagerefrange{#2}{#3}%
      \g@addto@macro\cref@poorman@text{ \vpagerefrange{#2}{#3}}%
      \def\@tmpa##1##2\@nil{%
```

```
      \if##1c%
        \if@crefstarred%
          \cref@writescript{}{%
            \string\vrefrange*\string{#2\string}\string{#3\string}}%
        \else%
          \cref@writescript{}{%
            \string\vrefrange\string{#2\string}\string{#3\string}}%
        \fi%
      \else%
        \if@crefstarred%
          \cref@writescript{}{%
            \string\Vrefrange*\string{#2\string}\string{#3\string}}%
        \else%
          \cref@writescript{}{%
            \string\Vrefrange\string{#2\string}\string{#3\string}}%
        \fi%
      \fi}%
    \@tmpa#1\@nil}
%
  \def\cref@fullref#1#2{%
    \gdef\cref@poorman@text{}%
    \begingroup%
      \@cref{#1}{#2} % space here is deliberate
      \def\@tmpstack{#2,\@nil}%
      \cref@stack@topandbottom{\@tmpstack}{\@firstref}{\@lastref}%
      \ifx\@lastref\@empty%
        \reftextfaraway{#2}%
        \def\@pageref{\reftextfaraway{#1}}%
      \else%
        \expandafter\vrefpagenum\expandafter%
          \@tmpa\expandafter{\@firstref}%
        \expandafter\vrefpagenum\expandafter%
          \@tmpb\expandafter{\@lastref}%
        \ifx\@tmpa\@tmpb
          \expandafter\reftextfaraway\expandafter{\@firstref}%
          \expandafter\def\expandafter\@pageref\expandafter{%
            \expandafter\reftextfaraway\expandafter{\@firstref}}%
        \else
          \edef\@tmpa{{\@firstref}{\@lastref}}%
          \expandafter\reftextpagerange\@tmpa%
          \expandafter\def\expandafter\@pageref\expandafter{%
            \expandafter\reftextpagerange\@tmpa}%
        \fi%
      \fi%
      \g@addto@macro\cref@poorman@text{ }%
      \expandafter\g@addto@macro\expandafter\cref@poorman@text%
        \expandafter{\@pageref}%
      \def\@tmpa##1##2\@nil{%
        \if##1c%
          \if@crefstarred%
```

```
            \cref@writescript{}{\string\fullref*\string{#2\string}}%
          \else%
            \cref@writescript{}{\string\fullref\string{#2\string}}%
          \fi%
        \else%
          \if@crefstarred%
            \cref@writescript{}{\string\Fullref*\string{#2\string}}%
          \else%
            \cref@writescript{}{\string\Fullref\string{#2\string}}%
          \fi%
        \fi}%
      \@tmpa#1\@nil
    \endgroup}
  }% end of \AtBeginDocument
}{}% end of \@ifpackageloaded
```

\@@setcref
\@@setcrefrange

Redefine \@@setcref and \@@setrangeref, as well as the conjunction macros \@setcref@middlegroupconjunction, \@setcref@lastgroupconjunction and \@setcref@pairgroupconjunction, to append text they type-set to the \cref@poorman@text macro, as well as actually doing the type-setting.

```
\def\@setcref@pairgroupconjunction{%
  \crefpairgroupconjunction%
  \expandafter\g@addto@macro\expandafter\cref@poorman@text%
    \expandafter{\crefpairgroupconjunction}}
\def\@setcref@middlegroupconjunction{%
  \crefmiddlegroupconjunction%
  \expandafter\g@addto@macro\expandafter\cref@poorman@text%
    \expandafter{\crefmiddlegroupconjunction}}
\def\@setcref@lastgroupconjunction{%
  \creflastgroupconjunction%
  \expandafter\g@addto@macro\expandafter\cref@poorman@text%
    \expandafter{\creflastgroupconjunction}}
\@ifpackageloaded{hyperref}{%
  \def\@@setcref#1#2{%
    \cref@getlabel{#2}{\@tmplabel}%
    \if@crefstarred%
      #1{\@tmplabel}{}{}%
      \expandafter\g@addto@macro\expandafter\cref@poorman@text%
        \expandafter{#1{\ref*{#2}}{}{}}%
    \else%
      \edef\@tmplink{\cref@hyperref{#2}}%
      #1{\@tmplabel}{\hyper@linkstart{link}{\@tmplink}}%
        {\hyper@linkend}%
      \expandafter\g@addto@macro\expandafter\cref@poorman@text%
        \expandafter{#1{\ref{#2}}{}{}}%
    \fi}
  \def\@@setcrefrange#1#2#3{%
    \cref@getlabel{#2}{\@labela}%
    \cref@getlabel{#3}{\@labelb}%
    \if@crefstarred%
```

```
        #1{\@labela}{\@labelb}{}{}{}{}%
        \expandafter\g@addto@macro\expandafter\cref@poorman@text%
          \expandafter{#1{\ref*{#2}}{\ref*{#3}}{}{}{}{}}%
      \else%
        \edef\@linka{\cref@hyperref{#2}}%
        \edef\@linkb{\cref@hyperref{#3}}%
        #1{\@labela}{\@labelb}%
          {\hyper@linkstart{link}{\@linka}}{\hyper@linkend}%
          {\hyper@linkstart{link}{\@linkb}}{\hyper@linkend}%
        \expandafter\g@addto@macro\expandafter\cref@poorman@text%
          \expandafter{#1{\ref{#2}}{\ref{#3}}{}{}{}{}}%
      \fi}
  }{%
    \def\@@setcref#1#2{%
      \old@@setcref{#1}{#2}%
      \expandafter\g@addto@macro\expandafter{%
        \expandafter\cref@poorman@text\expandafter}\expandafter{%
        #1{\ref{#2}}{}{}}}
    \def\@@setcrefrange#1#2#3{%
      \old@@setcrefrange{#1}{#2}{#3}%
      \expandafter\g@addto@macro%
        \expandafter{\expandafter\cref@poorman@text\expandafter}%
        \expandafter{#1{\ref{#2}}{\ref{#3}}{}{}{}{}}}
  }
}%  end of poorman option
```

10.9 Sort and Compress options

`sort` `compress` `nosort`

The `sort`, `compress` or `nosort` options determine whether to sort and/or compress lists of multiple references (default is to do both). They work simply by setting the `\if@cref@sort` and `\if@cref@compress` flags appropriately.

`\if@cref@sort` `\if@cref@compress`

```
\newif\if@cref@sort
\newif\if@cref@compress
```

Default is to both sort and compress references.

```
\@cref@sorttrue
\@cref@compresstrue
```

Options override default.

```
\DeclareOption{sort}{%
  \PackageInfo{cleveref}{sorting but not compressing references}
  \@cref@sorttrue
  \@cref@compressfalse}
\DeclareOption{compress}{%
  \PackageInfo{cleveref}{compressing but not sorting references}
  \@cref@sortfalse
  \@cref@compresstrue}
\DeclareOption{sort&compress}{%
  \PackageInfo{cleveref}{sorting and compressing references}
  \@cref@sorttrue
```

```
  \@cref@compresstrue}
\DeclareOption{nosort}{%
  \PackageInfo{cleveref}{neither sorting nor compressing references}
  \@cref@sortfalse
  \@cref@compressfalse}
```

10.10 Language and `babel` Support

Default reference formats for different languages are supported via package options, in the usual way.

Any contributions of translations for missing languages are most welcome! If you can contribute definitions for a missing language, ideally you should add them below the existing ones (using those as a model), generate a patch against the original `cleveref.dtx` file, and send the patch by email to the package author. However, if you don't know how to produce a patch, you can instead just send the translations as a plain text file.

`\crefdefaultlabelformat` `\creflabelformat` We first define the default label formats, which don't depend on language. We override the default format for equations, to follow the near universal convention of enclosing equation labels in brackets.

```
\crefdefaultlabelformat{#2#1#3}
\creflabelformat{equation}{\textup{(#2#1#3)}}
```

`\cref@addto` Utility macro to use instead of `babel`'s flawed `\addto` (copied and modified from `varioref`).

```
\def\cref@addto#1#2{%
  \@temptokena{#2}%
  \ifx#1\undefined
    \edef#1{\the\@temptokena}%
  \else
    \toks@\expandafter{#1}%
    \edef#1{\the\toks@\the\@temptokena}%
  \fi
  \@temptokena{}\toks@\@temptokena%
}
\@onlypreamble\cref@addto
```

Passing a language option to `cleveref` defines the cross-reference names and conjunctions as appropriate for that lanauge. We can't make the definitions straight away, since they would prevent the automatic definition of the other capitalisation variant from working if the user chooses to change a default definition in the preamble, so we postpone them until the beginning of the document. However, if each language option were to simply to define any formats that aren't already defined by the end of the preamble, the *first* language option would override all the others. Unfortunately, the convention in LaTeX and `babel` is for the *last* language option to take precedence. So we instead used the `\crefname@preamble` command to save the definitions in `\cref@\meta{type}@name@preamble` etc., and

after all the language options have been processed, use the contents of these to set the default definitions for any undefined formats.

For `babel` support, we add the appropriate redefinitions to the `\extras`⟨*language*⟩ macro, which is called by `babel`'s `\selectlanguage` at al. commands. The main language (the last one listed in the options) is set up by an automatic call to `\selectlanguage` at the beginning of the document, which would clobber any redefinitions made by the user in the preamble. To avoid this, we postpone adding the redefinitions to `\extras`⟨*language*⟩ until the beginning of the document. Since `cleveref` must always be loaded *after* `babel`, the redefinitions won't be added to `\extras`⟨*language*⟩ until *after* `babel` has already called `\selectlanguage` for the main language. Thus the redefinitions will only be in effect when `\selectlanguage` is called explicitly within the document. (The definitions for the main language are taken care of by the language options passed to `cleveref`, independently of `babel`.)

Note that we define both capitalisation variants explicitly throughout, rather than relying on the automatic definition of the other variant, in order to make the code produced by the poor man's `sed` script slightly cleaner.

`english` English definitions (these are used by default).

```
\DeclareOption{english}{%
  \PackageInfo{cleveref}{loaded 'english' language definitions}
```

First, we set up the definitions used at the beginning of the document to define the formats created by the document preamble.

```
  \AtBeginDocument{%
    \def\crefrangeconjunction@preamble{ to~}
    \def\crefpairconjunction@preamble{ and~}
    \def\crefmiddleconjunction@preamble{, }
    \def\creflastconjunction@preamble{ and~}
```

We have to define the group conjunctions explicitly here, rather than relying on fall-back definitions in terms of the above conjunctions (see section 10.10.1), in case any other language option defines them explicitly and we need to override those.

```
    \def\crefpairgroupconjunction@preamble{ and~}
    \def\crefmiddlegroupconjunction@preamble{, }
    \def\creflastgroupconjunction@preamble{, and~}
    \crefname@preamble{equation}{eq.}{eqs.}
    \Crefname@preamble{equation}{Equation}{Equations}
    \crefname@preamble{chapter}{chapter}{chapters}
    \Crefname@preamble{chapter}{Chapter}{Chapters}
    \crefname@preamble{section}{section}{sections}
    \Crefname@preamble{section}{Section}{Sections}
    \crefname@preamble{appendix}{appendix}{appendices}
    \Crefname@preamble{appendix}{Appendix}{Appendices}
    \crefname@preamble{figure}{fig.}{figs.}
    \Crefname@preamble{figure}{Figure}{Figures}
    \crefname@preamble{table}{table}{tables}
```

```
\Crefname@preamble{table}{Table}{Tables}
\crefname@preamble{theorem}{theorem}{theorems}
\Crefname@preamble{theorem}{Theorem}{Theorems}
\crefname@preamble{enumi}{item}{items}
\Crefname@preamble{enumi}{Item}{Items}
\crefname@preamble{lemma}{lemma}{lemmas}
\Crefname@preamble{lemma}{Lemma}{Lemmas}
\crefname@preamble{corollary}{corollary}{corollaries}
\Crefname@preamble{corollary}{Corollary}{Corollaries}
\crefname@preamble{proposition}{proposition}{propositions}
\Crefname@preamble{proposition}{Proposition}{Propositions}
\crefname@preamble{definition}{definition}{definitions}
\Crefname@preamble{definition}{Definition}{Definitions}
\crefname@preamble{result}{result}{results}
\Crefname@preamble{result}{Result}{Results}
\def\cref@language{english}%
```

Next, we add the definitions to `\extras`...so that `babel`'s `\selectlanguage` command will change the format appropriately.

```
\cref@addto\extrasenglish{%
  \renewcommand{\crefrangeconjunction}{ to~}
  \renewcommand{\crefpairconjunction}{ and~}
  \renewcommand{\crefmiddleconjunction}{, }
  \renewcommand{\creflastconjunction}{ and~}
  \renewcommand{\crefpairgroupconjunction}{ and~}
  \renewcommand{\crefmiddlegroupconjunction}{, }
  \renewcommand{\creflastgroupconjunction}{, and~}
  \crefname{equation}{eq.}{eqs.}
  \Crefname{equation}{Equation}{Equations}
  \crefname{chapter}{chapter}{chapters}
  \Crefname{chapter}{Chapter}{Chapters}
  \crefname{section}{section}{sections}
  \Crefname{section}{Section}{Sections}
  \crefname{subsection}{section}{sections}
  \Crefname{subsection}{Section}{Sections}
  \crefname{subsubsection}{section}{sections}
  \Crefname{subsubsection}{Section}{Sections}
  \crefname{appendix}{appendix}{appendices}
  \Crefname{appendix}{Appendix}{Appendices}
  \crefname{subappendix}{appendix}{appendices}
  \Crefname{subappendix}{Appendix}{Appendices}
  \crefname{subsubappendix}{appendix}{appendices}
  \Crefname{subsubappendix}{Appendix}{Appendices}
  \crefname{subsubsubappendix}{appendix}{appendices}
  \Crefname{subsubsubappendix}{Appendix}{Appendices}
  \crefname{figure}{fig.}{figs.}
  \Crefname{figure}{Figure}{Figures}
  \crefname{subfigure}{fig.}{figs.}
  \Crefname{subfigure}{Figure}{Figures}
  \crefname{table}{table}{tables}
```

```
    \Crefname{table}{Table}{Tables}
    \crefname{subtable}{table}{tables}
    \Crefname{subtable}{Table}{Tables}
    \crefname{theorem}{theorem}{theorems}
    \Crefname{theorem}{Theorem}{Theorems}
    \crefname{enumi}{item}{items}
    \Crefname{enumi}{Item}{Items}
    \crefname{enumii}{item}{items}
    \Crefname{enumii}{Item}{Items}
    \crefname{enumiii}{item}{items}
    \Crefname{enumiii}{Item}{Items}
    \crefname{enumiv}{item}{items}
    \Crefname{enumiv}{Item}{Items}
    \crefname{enumv}{item}{items}
    \Crefname{enumv}{Item}{Items}
    \crefname{lemma}{lemma}{lemmas}
    \Crefname{lemma}{Lemma}{Lemmas}
    \crefname{corollary}{corollary}{corollaries}
    \Crefname{corollary}{Corollary}{Corollaries}
    \crefname{proposition}{proposition}{propositions}
    \Crefname{proposition}{Proposition}{Propositions}
    \crefname{definition}{definition}{definitions}
    \Crefname{definition}{Definition}{Definitions}
    \crefname{result}{result}{results}
    \Crefname{result}{Result}{Results}%
  }}}
```

`german` German translations kindly provided by Stefan Pinnow, with a few additions by the package author (so you know to blame the latter for any errors!).

```
\DeclareOption{german}{%
  \PackageInfo{cleveref}{loaded 'german' language definitions}
```

First, we set up the definitions used at the beginning of the document to define the formats created by the document preamble.

```
  \AtBeginDocument{%
    \def\crefrangeconjunction@preamble{ bis~}
    \def\crefpairconjunction@preamble{ und~}
    \def\crefmiddleconjunction@preamble{, }
    \def\creflastconjunction@preamble{ und~}
```

We don't want the extra comma before "und" that would be added by the default fall-back definitions in terms of the above conjunctions, so we define `\crefpairgroupconjunction` explicitly. In fact, we have to define the other group conjunctions explicitly too here, in case any other language option defines them explicitly and we need to override them .

```
    \def\crefpairgroupconjunction@preamble{ und~}
    \def\crefmiddlegroupconjunction@preamble{, }
    \def\creflastgroupconjunction@preamble{ und~}
    \crefname@preamble{equation}{Gleichung}{Gleichungen}
```

```
\Crefname@preamble{equation}{Gleichung}{Gleichungen}
\crefname@preamble{chapter}{Kapitel}{Kapitel}
\Crefname@preamble{chapter}{Kapitel}{Kapitel}
\crefname@preamble{section}{Abschnitt}{Abschnitte}
\Crefname@preamble{section}{Abschnitt}{Abschnitte}
\crefname@preamble{appendix}{Anhang}{Anh\"ange}
\Crefname@preamble{appendix}{Anhang}{Anh\"ange}
\crefname@preamble{figure}{Abbildung}{Abbildungen}
\Crefname@preamble{figure}{Abbildung}{Abbildungen}
\crefname@preamble{table}{Tabelle}{Tabellen}
\Crefname@preamble{table}{Tabelle}{Tabellen}
\crefname@preamble{theorem}{Theorem}{Theoremen}
\Crefname@preamble{theorem}{Theorem}{Theoremen}
\crefname@preamble{enumi}{Punkt}{Punkte}
\Crefname@preamble{enumi}{Punkt}{Punkte}
\crefname@preamble{lemma}{Lemma}{Lemmata}
\Crefname@preamble{lemma}{Lemma}{Lemmata}
\crefname@preamble{corollary}{Korollar}{Korollaren}
\Crefname@preamble{corollary}{Korollar}{Korollaren}
\crefname@preamble{proposition}{Satz}{S\"atze}
\Crefname@preamble{proposition}{Satz}{S\"atze}
\crefname@preamble{definition}{Definition}{Definitionen}
\Crefname@preamble{definition}{Definition}{Definitionen}
\crefname@preamble{result}{Ergebnis}{Ergebnisse}
\Crefname@preamble{result}{Ergebnis}{Ergebnisse}%
\def\cref@language{german}%
```

Next, we add the definitions to `\extras`...so that `babel`'s `\selectlanguage` command will change the format appropriately.

```
\cref@addto\extrasgerman{%
  \renewcommand{\crefrangeconjunction}{ bis~}%
  \renewcommand{\crefpairconjunction}{ und~}%
  \renewcommand{\crefmiddleconjunction}{, }%
  \renewcommand{\creflastconjunction}{ und~}%
  \renewcommand{\crefpairgroupconjunction}{ und~}%
  \renewcommand{\crefmiddlegroupconjunction}{, }%
  \renewcommand{\creflastgroupconjunction}{ und~}%
  \crefname{equation}{Gleichung}{Gleichungen}%
  \Crefname{equation}{Gleichung}{Gleichungen}%
  \crefname{chapter}{Kapitel}{Kapitel}%
  \Crefname{chapter}{Kapitel}{Kapitel}%
  \crefname{section}{Abschnitt}{Abschnitte}%
  \Crefname{section}{Abschnitt}{Abschnitte}%
  \crefname{subsection}{Abschnitt}{Abschnitte}%
  \Crefname{subsection}{Abschnitt}{Abschnitte}%
  \crefname{subsubsection}{Abschnitt}{Abschnitte}%
  \Crefname{subsubsection}{Abschnitt}{Abschnitte}%
  \crefname{appendix}{Anhang}{Anh\"ange}%
  \Crefname{appendix}{Anhang}{Anh\"ange}%
  \crefname{subappendix}{Anhang}{Anh\"ange}
```

```
    \Crefname{subappendix}{Anhang}{Anh\"ange}
    \crefname{subsubappendix}{Anhang}{Anh\"ange}
    \Crefname{subsubappendix}{Anhang}{Anh\"ange}
    \crefname{subsubsubappendix}{Anhang}{Anh\"ange}
    \Crefname{subsubsubappendix}{Anhang}{Anh\"ange}
    \crefname{figure}{Abbildung}{Abbildungen}%
    \Crefname{figure}{Abbildung}{Abbildungen}%
    \crefname{subfigure}{Abbildung}{Abbildungen}%
    \Crefname{subfigure}{Abbildung}{Abbildungen}%
    \crefname{table}{Tabelle}{Tabellen}%
    \Crefname{table}{Tabelle}{Tabellen}%
    \crefname{subtable}{Tabelle}{Tabellen}%
    \Crefname{subtable}{Tabelle}{Tabellen}%
    \crefname{theorem}{Theorem}{Theoremen}%
    \Crefname{theorem}{Theorem}{Theoremen}%
    \crefname{enumi}{Punkt}{Punkte}%
    \Crefname{enumi}{Punkt}{Punkte}%
    \crefname{enumii}{Punkt}{Punkte}%
    \Crefname{enumii}{Punkt}{Punkte}%
    \crefname{enumiii}{Punkt}{Punkte}%
    \Crefname{enumiii}{Punkt}{Punkte}%
    \crefname{enumiv}{Punkt}{Punkte}%
    \Crefname{enumiv}{Punkt}{Punkte}%
    \crefname{enumv}{Punkt}{Punkte}%
    \Crefname{enumv}{Punkt}{Punkte}%
    \crefname{lemma}{Lemma}{Lemmata}%
    \Crefname{lemma}{Lemma}{Lemmata}%
    \crefname{corollary}{Korollar}{Korollaren}%
    \Crefname{corollary}{Korollar}{Korollaren}%
    \crefname{proposition}{Satz}{S\"atze}%
    \Crefname{proposition}{Satz}{S\"atze}%
    \crefname{definition}{Definition}{Definitionen}%
    \Crefname{definition}{Definition}{Definitionen}%
    \crefname{result}{Ergebnis}{Ergebnisse}%
    \Crefname{result}{Ergebnis}{Ergebnisse}%
  }}}
```

`ngerman` It so happens that none of the cross-reference names differ in the "Neuerechtschreibung", so we make `ngerman` execute `german`. However, we still need to add the definitions to `\extrasngerman` (note the "n") so that `\selectlanguage` etc. will work.

```
\DeclareOption{ngerman}{%
  \PackageInfo{cleveref}{loaded 'ngerman' language definitions}
  \ExecuteOptions{german}
  \def\cref@language{ngerman}
  \AtBeginDocument{%
    \cref@addto\extrasngerman{%
      \renewcommand{\crefrangeconjunction}{ bis~}%
      \renewcommand{\crefpairconjunction}{ und~}%
```

```
\renewcommand{\crefmiddleconjunction}{, }%
\renewcommand{\creflastconjunction}{ und~}%
\renewcommand{\crefpairgroupconjunction}{ und~}%
\renewcommand{\crefmiddlegroupconjunction}{, }%
\renewcommand{\creflastgroupconjunction}{ und~}%
\crefname{equation}{Gleichung}{Gleichungen}%
\Crefname{equation}{Gleichung}{Gleichungen}%
\crefname{chapter}{Kapitel}{Kapitel}%
\Crefname{chapter}{Kapitel}{Kapitel}%
\crefname{section}{Abschnitt}{Abschnitte}%
\Crefname{section}{Abschnitt}{Abschnitte}%
\crefname{subsection}{Abschnitt}{Abschnitte}%
\Crefname{subsection}{Abschnitt}{Abschnitte}%
\crefname{subsubsection}{Abschnitt}{Abschnitte}%
\Crefname{subsubsection}{Abschnitt}{Abschnitte}%
\crefname{appendix}{Anhang}{Anh\"ange}%
\Crefname{appendix}{Anhang}{Anh\"ange}%
\crefname{subappendix}{Anhang}{Anh\"ange}
\Crefname{subappendix}{Anhang}{Anh\"ange}
\crefname{subsubappendix}{Anhang}{Anh\"ange}
\Crefname{subsubappendix}{Anhang}{Anh\"ange}
\crefname{subsubsubappendix}{Anhang}{Anh\"ange}
\Crefname{subsubsubappendix}{Anhang}{Anh\"ange}
\crefname{figure}{Abbildung}{Abbildungen}%
\Crefname{figure}{Abbildung}{Abbildungen}%
\crefname{subfigure}{Abbildung}{Abbildungen}%
\Crefname{subfigure}{Abbildung}{Abbildungen}%
\crefname{table}{Tabelle}{Tabellen}%
\Crefname{table}{Tabelle}{Tabellen}%
\crefname{subtable}{Tabelle}{Tabellen}%
\Crefname{subtable}{Tabelle}{Tabellen}%
\crefname{theorem}{Theorem}{Theoremen}%
\Crefname{theorem}{Theorem}{Theoremen}%
\crefname{enumi}{Punkt}{Punkte}%
\Crefname{enumi}{Punkt}{Punkte}%
\crefname{enumii}{Punkt}{Punkte}%
\Crefname{enumii}{Punkt}{Punkte}%
\crefname{enumiii}{Punkt}{Punkte}%
\Crefname{enumiii}{Punkt}{Punkte}%
\crefname{enumiv}{Punkt}{Punkte}%
\Crefname{enumiv}{Punkt}{Punkte}%
\crefname{enumv}{Punkt}{Punkte}%
\Crefname{enumv}{Punkt}{Punkte}%
\crefname{lemma}{Lemma}{Lemmata}%
\Crefname{lemma}{Lemma}{Lemmata}%
\crefname{corollary}{Korollar}{Korollaren}%
\Crefname{corollary}{Korollar}{Korollaren}%
\crefname{proposition}{Satz}{S\"atze}%
\Crefname{proposition}{Satz}{S\"atze}%
\crefname{definition}{Definition}{Definitionen}%
```

```
      \Crefname{definition}{Definition}{Definitionen}%
      \crefname{result}{Ergebnis}{Ergebnisse}%
      \Crefname{result}{Ergebnis}{Ergebnisse}%
    }}}
```

`french` French translations attempted by the package author (please report any corrections that might be needed!).

```
\DeclareOption{french}{%
  \PackageInfo{cleveref}{loaded 'french' language definitions}
```

First, we set up the definitions used at the beginning of the document to define the formats created by the document preamble.

```
  \AtBeginDocument{%
    \def\crefrangeconjunction@preamble{ \`a~}
    \def\crefpairconjunction@preamble{ et~}
    \def\crefmiddleconjunction@preamble{, }
    \def\creflastconjunction@preamble{ et~}
```

Erring on the side of caution, I've left off the extra comma before "et" between groups, pending more knowledgeable input on punctuation rules from a native Franch speaker.

```
    \def\crefpairgroupconjunction@preamble{ et~}
    \def\crefmiddlegroupconjunction@preamble{, }
    \def\creflastgroupconjunction@preamble{, et~}
    \crefname@preamble{equation}{{\'e}quation}{{\'e}quations}
    \Crefname@preamble{equation}{{\'E}quation}{{\'E}quations}
    \crefname@preamble{chapter}{chapitre}{chapitres}
    \Crefname@preamble{chapter}{Chapitre}{Chapitres}
    \crefname@preamble{section}{section}{sections}
    \Crefname@preamble{section}{Section}{Sections}
    \crefname@preamble{appendix}{appendice}{appendices}
    \Crefname@preamble{appendix}{Appendice}{Appendices}
    \crefname@preamble{figure}{figure}{figures}
    \Crefname@preamble{figure}{Figure}{Figures}
    \crefname@preamble{table}{tableau}{tableaux}
    \Crefname@preamble{table}{Tableau}{Tableaux}
    \crefname@preamble{theorem}{th\'eor\`eme}{th\'eor\`emes}
    \Crefname@preamble{theorem}{Th\'eor\`eme}{Th\'eor\`emes}
    \crefname@preamble{enumi}{point}{points}
    \Crefname@preamble{enumi}{Point}{Points}
    \crefname@preamble{lemma}{lemme}{lemmes}
    \Crefname@preamble{lemma}{Lemme}{Lemmes}
    \crefname@preamble{corollary}{corollaire}{corollaires}
    \Crefname@preamble{corollary}{Corollaire}{Corollaires}
    \crefname@preamble{proposition}{proposition}{propositions}
    \Crefname@preamble{proposition}{Proposition}{Propositions}
    \crefname@preamble{definition}{d\'efinition}{d\'efinitions}
    \Crefname@preamble{definition}{D\'efinition}{D\'efinitions}
    \crefname@preamble{result}{r\'esultat}{r\'esultats}
```

```
    \Crefname@preamble{result}{R\'esultat}{R\'esultats}
    \def\cref@language{french}%
```

Next, we add the definitions to `\extras`... so that `babel`'s `\selectlanguage` command will change the format appropriately.

```
    \cref@addto\extrasfrench{%
      \renewcommand{\crefrangeconjunction}{ \`a~}%
      \renewcommand{\crefpairconjunction}{ et~}%
      \renewcommand{\crefmiddleconjunction}{, }%
      \renewcommand{\creflastconjunction}{ et~}%
      \renewcommand{\crefpairgroupconjunction}{ et~}%
      \renewcommand{\crefmiddlegroupconjunction}{, }%
      \renewcommand{\creflastgroupconjunction}{ et~}%
      \crefname{equation}{{\'e}quation}{{\'e}quations}%
      \Crefname{equation}{{\'E}quation}{{\'E}quations}%
      \crefname{chapter}{chapitre}{chapitres}%
      \Crefname{chapter}{Chapitre}{Chapitres}%
      \crefname{section}{section}{sections}%
      \Crefname{section}{Section}{Sections}%
      \crefname{subsection}{section}{sections}%
      \Crefname{subsection}{Section}{Sections}%
      \crefname{subsubsection}{section}{sections}%
      \Crefname{subsubsection}{Section}{Sections}%
      \crefname{appendix}{appendice}{appendices}%
      \Crefname{appendix}{Appendice}{Appendices}%
      \crefname{subappendix}{appendice}{appendices}
      \Crefname{subappendix}{Appendice}{Appendices}
      \crefname{subsubappendix}{appendice}{appendices}
      \Crefname{subsubappendix}{Appendice}{Appendices}
      \crefname{subsubsubappendix}{appendice}{appendices}
      \Crefname{subsubsubappendix}{Appendice}{Appendices}
      \crefname{figure}{figure}{figures}%
      \Crefname{figure}{Figure}{Figures}%
      \crefname{subfigure}{figure}{figures}%
      \Crefname{subfigure}{Figure}{Figures}%
      \crefname{table}{tableau}{tableaux}%
      \Crefname{table}{Tableau}{Tableaux}%
      \crefname{subtable}{tableau}{tableaux}%
      \Crefname{subtable}{Tableau}{Tableaux}%
      \crefname{theorem}{th\'eor\`eme}{th\'eor\`emes}%
      \Crefname{theorem}{Th\'eor\`eme}{Th\'eor\`emes}%
      \crefname{enumi}{point}{points}%
      \Crefname{enumi}{Point}{Points}%
      \crefname{enumii}{point}{points}%
      \Crefname{enumii}{Point}{Points}%
      \crefname{enumiii}{point}{points}%
      \Crefname{enumiii}{Point}{Points}%
      \crefname{enumiv}{point}{points}%
      \Crefname{enumiv}{Point}{Points}%
      \crefname{enumv}{point}{points}%
```

```
      \Crefname{enumv}{Point}{Points}%
      \crefname{lemma}{lemme}{lemmes}%
      \Crefname{lemma}{Lemme}{Lemmes}%
      \crefname{corollary}{corollaire}{corollaires}%
      \Crefname{corollary}{Corollaire}{Corollaires}%
      \crefname{proposition}{proposition}{propositions}%
      \Crefname{proposition}{Proposition}{Propositions}%
      \crefname{definition}{d\'efinition}{d\'efinitions}%
      \Crefname{definition}{D\'efinition}{D\'efinitions}%
      \crefname{result}{r\'esultat}{r\'esultats}%
      \Crefname{result}{R\'esultat}{R\'esultats}%
    }}}
```

10.10.1 Default Cross-Reference Formats

Setup default English format definitions, then process options in the order they were supplied.

```
\ExecuteOptions{english}
\ProcessOptions*\relax
```

Define the component-derived formats.

```
\AtBeginDocument{%
```

Use whatever's in the ... `@preamble` definitions at the beginning of the document to set up the default cross-reference names.

```
  \edef\@tmpa{%
    \expandafter\noexpand\csname extras\cref@language\endcsname}%
  \@ifundefined{crefrangeconjunction}{%
    \let\crefrangeconjunction\crefrangeconjunction@preamble%
  }{%
    \expandafter\def\expandafter\@tmpb\expandafter{%
      \expandafter\renewcommand\expandafter%
      {\expandafter\crefrangeconjunction\expandafter}%
      \expandafter{\crefrangeconjunction}}%
    \expandafter\expandafter\expandafter\cref@addto%
      \expandafter\@tmpa\expandafter{\@tmpb}%
  }%
  \@ifundefined{crefpairconjunction}{%
    \let\crefpairconjunction\crefpairconjunction@preamble%
  }{%
    \expandafter\def\expandafter\@tmpb\expandafter{%
      \expandafter\renewcommand\expandafter%
      {\expandafter\crefpairconjunction\expandafter}%
      \expandafter{\crefpairconjunction}}%
    \expandafter\expandafter\expandafter\cref@addto%
      \expandafter\@tmpa\expandafter{\@tmpb}%
  }%
  \@ifundefined{crefmiddleconjunction}{%
    \let\crefmiddleconjunction\crefmiddleconjunction@preamble%
  }{%
```

```
    \expandafter\def\expandafter\@tmpb\expandafter{%
      \expandafter\renewcommand\expandafter%
      {\expandafter\crefmiddleconjunction\expandafter}%
      \expandafter{\crefmiddleconjunction}}%
    \expandafter\expandafter\expandafter\cref@addto%
      \expandafter\@tmpa\expandafter{\@tmpb}%
  }%
  \@ifundefined{creflastconjunction}{%
    \let\creflastconjunction\creflastconjunction@preamble%
  }{%
    \expandafter\def\expandafter\@tmpb\expandafter{%
      \expandafter\renewcommand\expandafter%
      {\expandafter\creflastconjunction\expandafter}%
      \expandafter{\creflastconjunction}}%
    \expandafter\expandafter\expandafter\cref@addto%
      \expandafter\@tmpa\expandafter{\@tmpb}%
  }%
  \@ifundefined{crefpairgroupconjunction}{%
    \let\crefpairgroupconjunction%
      \crefpairgroupconjunction@preamble%
  }{%
    \expandafter\def\expandafter\@tmpb\expandafter{%
      \expandafter\renewcommand\expandafter%
      {\expandafter\crefpairgroupconjunction\expandafter}%
      \expandafter{\crefpairgroupconjunction}}%
    \expandafter\expandafter\expandafter\cref@addto%
      \expandafter\@tmpa\expandafter{\@tmpb}%
  }%
  \@ifundefined{crefmiddlegroupconjunction}{%
    \let\crefmiddlegroupconjunction%
      \crefmiddlegroupconjunction@preamble%
  }{%
    \expandafter\def\expandafter\@tmpb\expandafter{%
      \expandafter\renewcommand\expandafter%
      {\expandafter\crefpairmiddleconjunction\expandafter}%
      \expandafter{\crefpairmiddleconjunction}}%
    \expandafter\expandafter\expandafter\cref@addto%
      \expandafter\@tmpa\expandafter{\@tmpb}%
  }%
  \@ifundefined{creflastgroupconjunction}{%
    \let\creflastgroupconjunction%
      \creflastgroupconjunction@preamble%
  }{%
    \expandafter\def\expandafter\@tmpb\expandafter{%
      \expandafter\renewcommand\expandafter%
      {\expandafter\crefpairlastconjunction\expandafter}%
      \expandafter{\crefpairlastconjunction}}%
    \expandafter\expandafter\expandafter\cref@addto%
      \expandafter\@tmpa\expandafter{\@tmpb}%
  }%
```

```
% \end{macrocode}
% If the group conjunctions haven't been defined, define them to be
% identical to the reference conjunctions.
%    \begin{macrocode}
  \@ifundefined{crefpairgroupconjunction}{%
    \let\crefpairgroupconjunction\crefpairconjunction}{}%
  \@ifundefined{crefmiddlegroupconjunction}{%
    \let\crefmiddlegroupconjunction\crefmiddleconjunction}{}%
```

Define the last group conjunction to include an extra comma.

```
  \@ifundefined{creflastgroupconjunction}{%
    \edef\creflastgroupconjunction{, \creflastconjunction}}{}%
```

Define any undefined formats listed in `\cref@label@types` using the components.

```
  \let\@tmpstack\cref@label@types%
  \cref@isstackfull{\@tmpstack}%
  \@whilesw\if@cref@stackfull\fi{%
    \edef\@tmpa{\cref@stack@top{\@tmpstack}}%
    \@ifundefined{cref@\@tmpa @name}{%
      \expandafter\def\expandafter\@tmpb\expandafter{%
        \csname cref@\@tmpa @name\endcsname}%
      \expandafter\def\expandafter\@tmpc\expandafter{%
        \csname cref@\@tmpa @name@preamble\endcsname}%
      \expandafter\expandafter\expandafter\let\expandafter\@tmpb\@tmpc%
      \expandafter\def\expandafter\@tmpb\expandafter{%
        \csname cref@\@tmpa @name@plural\endcsname}%
      \expandafter\def\expandafter\@tmpc\expandafter{%
        \csname cref@\@tmpa @name@plural@preamble\endcsname}%
      \expandafter\expandafter\expandafter\let\expandafter\@tmpb\@tmpc
    }{%
      \edef\@tmpb{%
        \expandafter\noexpand\csname extras\cref@language\endcsname}%
      \expandafter\def\expandafter\@tmpc\expandafter{%
        \expandafter\crefname\expandafter{\@tmpa}}%
      \expandafter\expandafter\expandafter\cref@addto%
      \expandafter\expandafter\expandafter\@tmpc%
      \expandafter\expandafter\expandafter{%
        \expandafter\expandafter\expandafter{%
          \csname cref@\@tmpa @name\endcsname}}%
      \expandafter\expandafter\expandafter\cref@addto%
      \expandafter\expandafter\expandafter\@tmpc%
      \expandafter\expandafter\expandafter{%
        \expandafter\expandafter\expandafter{%
          \csname cref@\@tmpa @name@plural\endcsname}}%
      \expandafter\expandafter\expandafter\cref@addto%
        \expandafter\@tmpb\expandafter{\@tmpc}%
    }%
    \@ifundefined{Cref@\@tmpa @name}{%
      \expandafter\def\expandafter\@tmpb\expandafter{%
        \csname Cref@\@tmpa @name\endcsname}%
```

```
    \expandafter\def\expandafter\@tmpc\expandafter{%
      \csname Cref@\@tmpa @name@preamble\endcsname}%
    \expandafter\expandafter\expandafter\let\expandafter\@tmpb\@tmpc%
    \expandafter\def\expandafter\@tmpb\expandafter{%
      \csname Cref@\@tmpa @name@plural\endcsname}%
    \expandafter\def\expandafter\@tmpc\expandafter{%
      \csname Cref@\@tmpa @name@plural@preamble\endcsname}%
    \expandafter\expandafter\expandafter\let\expandafter\@tmpb\@tmpc
  }{%
    \edef\@tmpb{%
      \expandafter\noexpand\csname extras\cref@language\endcsname}%
    \expandafter\def\expandafter\@tmpc\expandafter{%
      \expandafter\Crefname\expandafter{\@tmpa}}%
    \expandafter\expandafter\expandafter\cref@addto%
    \expandafter\expandafter\expandafter\@tmpc%
    \expandafter\expandafter\expandafter{%
      \expandafter\expandafter\expandafter{%
        \csname Cref@\@tmpa @name\endcsname}}%
    \expandafter\expandafter\expandafter\cref@addto%
    \expandafter\expandafter\expandafter\@tmpc%
    \expandafter\expandafter\expandafter{%
      \expandafter\expandafter\expandafter{%
        \csname Cref@\@tmpa @name@plural\endcsname}}%
    \expandafter\expandafter\expandafter\cref@addto%
      \expandafter\@tmpb\expandafter{\@tmpc}%
  }%
  \@ifundefined{cref@\@tmpa @format}{%
    \expandafter\@crefdefineformat\expandafter{\@tmpa}}{}%
  \@ifundefined{crefrange@\@tmpa @format}{%
    \expandafter\@crefrangedefineformat\expandafter{\@tmpa}}{}%
  \@ifundefined{cref@\@tmpa @format@first}{%
    \expandafter\@crefdefinemultiformat\expandafter{\@tmpa}}{}%
  \@ifundefined{crefrange@\@tmpa @format@first}{%
    \expandafter\@crefrangedefinemultiformat\expandafter{\@tmpa}}{}%
  \cref@stack@pop{\@tmpstack}%
  \cref@isstackfull{\@tmpstack}}%
```

If formats for subsections are undefined, define them to be identical to the formats for sections.

```
  \@ifundefined{cref@subsection@format}{%
    \let\cref@subsection@format%
    \cref@section@format}{}%
  \@ifundefined{Cref@subsection@format}{%
    \let\Cref@subsection@format%
    \Cref@section@format}{}%
  \@ifundefined{crefrange@subsection@format}{%
    \let\crefrange@subsection@format%
    \crefrange@section@format}{}%
  \@ifundefined{Crefrange@subsection@format}{%
    \let\Crefrange@subsection@format%
```

```
    \Crefrange@section@format}{}%
  \@ifundefined{cref@subsection@format@first}{%
    \let\cref@subsection@format@first%
    \cref@section@format@first}{}%
  \@ifundefined{Cref@subsection@format@first}{%
    \let\Cref@subsection@format@first%
    \Cref@section@format@first}{}%
  \@ifundefined{cref@subsection@format@second}{%
    \let\cref@subsection@format@second%
    \cref@section@format@second}{}%
  \@ifundefined{Cref@subsection@format@second}{%
    \let\Cref@subsection@format@second%
    \Cref@section@format@second}{}%
  \@ifundefined{cref@subsection@format@middle}{%
    \let\cref@subsection@format@middle%
    \cref@section@format@middle}{}%
  \@ifundefined{Cref@subsection@format@middle}{%
    \let\Cref@subsection@format@middle%
    \Cref@section@format@middle}{}%
  \@ifundefined{cref@subsection@format@last}{%
    \let\cref@subsection@format@last%
    \cref@section@format@last}{}%
  \@ifundefined{Cref@subsection@format@last}{%
    \let\Cref@subsection@format@last%
    \Cref@section@format@last}{}%
  \@ifundefined{crefrange@subsection@format@first}{%
    \let\crefrange@subsection@format@first%
    \crefrange@section@format@first}{}%
  \@ifundefined{Crefrange@subsection@format@first}{%
    \let\Crefrange@subsection@format@first%
    \Crefrange@section@format@first}{}%
  \@ifundefined{crefrange@subsection@format@second}{%
    \let\crefrange@subsection@format@second%
    \crefrange@section@format@second}{}%
  \@ifundefined{Crefrange@subsection@format@second}{%
    \let\Crefrange@subsection@format@second%
    \Crefrange@section@format@second}{}%
  \@ifundefined{crefrange@subsection@format@middle}{%
    \let\crefrange@subsection@format@middle%
    \crefrange@section@format@middle}{}%
  \@ifundefined{Crefrange@subsection@format@middle}{%
    \let\Crefrange@subsection@format@middle%
    \Crefrange@section@format@middle}{}%
  \@ifundefined{crefrange@subsection@format@last}{%
    \let\crefrange@subsection@format@last%
    \crefrange@section@format@last}{}%
  \@ifundefined{Crefrange@subsection@format@last}{%
    \let\Crefrange@subsection@format@last%
    \Crefrange@section@format@last}{}%
%
```

```
\@ifundefined{cref@subsubsection@format}{%
  \let\cref@subsubsection@format%
  \cref@subsection@format}{}%
\@ifundefined{Cref@subsubsection@format}{%
  \let\Cref@subsubsection@format%
  \Cref@subsection@format}{}%
\@ifundefined{crefrange@subsubsection@format}{%
  \let\crefrange@subsubsection@format%
  \crefrange@subsection@format}{}%
\@ifundefined{Crefrange@subsubsection@format}{%
  \let\Crefrange@subsubsection@format%
  \Crefrange@subsection@format}{}%
\@ifundefined{cref@subsubsection@format@first}{%
  \let\cref@subsubsection@format@first%
  \cref@subsection@format@first}{}%
\@ifundefined{Cref@subsubsection@format@first}{%
  \let\Cref@subsubsection@format@first%
  \Cref@subsection@format@first}{}%
\@ifundefined{cref@subsubsection@format@second}{%
  \let\cref@subsubsection@format@second%
  \cref@subsection@format@second}{}%
\@ifundefined{Cref@subsubsection@format@second}{%
  \let\Cref@subsubsection@format@second%
  \Cref@subsection@format@second}{}%
\@ifundefined{cref@subsubsection@format@middle}{%
  \let\cref@subsubsection@format@middle%
  \cref@subsection@format@middle}{}%
\@ifundefined{Cref@subsubsection@format@middle}{%
  \let\Cref@subsubsection@format@middle%
  \Cref@subsection@format@middle}{}%
\@ifundefined{cref@subsubsection@format@last}{%
  \let\cref@subsubsection@format@last%
  \cref@subsection@format@last}{}%
\@ifundefined{Cref@subsubsection@format@last}{%
  \let\Cref@subsubsection@format@last%
  \Cref@subsection@format@last}{}%
\@ifundefined{crefrange@subsubsection@format@first}{%
  \let\crefrange@subsubsection@format@first%
  \crefrange@subsection@format@first}{}%
\@ifundefined{Crefrange@subsubsection@format@first}{%
  \let\Crefrange@subsubsection@format@first%
  \Crefrange@subsection@format@first}{}%
\@ifundefined{crefrange@subsubsection@format@second}{%
  \let\crefrange@subsubsection@format@second%
  \crefrange@subsection@format@second}{}%
\@ifundefined{Crefrange@subsubsection@format@second}{%
  \let\Crefrange@subsubsection@format@second%
  \Crefrange@subsection@format@second}{}%
\@ifundefined{crefrange@subsubsection@format@middle}{%
  \let\crefrange@subsubsection@format@middle%
```

```
    \crefrange@subsection@format@middle}{}%
  \@ifundefined{Crefrange@subsubsection@format@middle}{%
    \let\Crefrange@subsubsection@format@middle%
    \Crefrange@subsection@format@middle}{}%
  \@ifundefined{crefrange@subsubsection@format@last}{%
    \let\crefrange@subsubsection@format@last%
    \crefrange@subsection@format@last}{}%
  \@ifundefined{Crefrange@subsubsection@format@last}{%
    \let\Crefrange@subsubsection@format@last%
    \Crefrange@subsection@format@last}{}%
```

Similarly for subsections within appendices.

```
% \begin{macrocode}
  \@ifundefined{cref@subappendix@format}{%
    \let\cref@subappendix@format%
    \cref@appendix@format}{}%
  \@ifundefined{Cref@subappendix@format}{%
    \let\Cref@subappendix@format%
    \Cref@appendix@format}{}%
  \@ifundefined{crefrange@subappendix@format}{%
    \let\crefrange@subappendix@format%
    \crefrange@appendix@format}{}%
  \@ifundefined{Crefrange@subappendix@format}{%
    \let\Crefrange@subappendix@format%
    \Crefrange@appendix@format}{}%
  \@ifundefined{cref@subappendix@format@first}{%
    \let\cref@subappendix@format@first%
    \cref@appendix@format@first}{}%
  \@ifundefined{Cref@subappendix@format@first}{%
    \let\Cref@subappendix@format@first%
    \Cref@appendix@format@first}{}%
  \@ifundefined{cref@subappendix@format@second}{%
    \let\cref@subappendix@format@second%
    \cref@appendix@format@second}{}%
  \@ifundefined{Cref@subappendix@format@second}{%
    \let\Cref@subappendix@format@second%
    \Cref@appendix@format@second}{}%
  \@ifundefined{cref@subappendix@format@middle}{%
    \let\cref@subappendix@format@middle%
    \cref@appendix@format@middle}{}%
  \@ifundefined{Cref@subappendix@format@middle}{%
    \let\Cref@subappendix@format@middle%
    \Cref@appendix@format@middle}{}%
  \@ifundefined{cref@subappendix@format@last}{%
    \let\cref@subappendix@format@last%
    \cref@appendix@format@last}{}%
  \@ifundefined{Cref@subappendix@format@last}{%
    \let\Cref@subappendix@format@last%
    \Cref@appendix@format@last}{}%
  \@ifundefined{crefrange@subappendix@format@first}{%
```

```
      \let\crefrange@subappendix@format@first%
      \crefrange@appendix@format@first}{}%
    \@ifundefined{Crefrange@subappendix@format@first}{%
      \let\Crefrange@subappendix@format@first%
      \Crefrange@appendix@format@first}{}%
    \@ifundefined{crefrange@subappendix@format@second}{%
      \let\crefrange@subappendix@format@second%
      \crefrange@appendix@format@second}{}%
    \@ifundefined{Crefrange@subappendix@format@second}{%
      \let\Crefrange@subappendix@format@second%
      \Crefrange@appendix@format@second}{}%
    \@ifundefined{crefrange@subappendix@format@middle}{%
      \let\crefrange@subappendix@format@middle%
      \crefrange@appendix@format@middle}{}%
    \@ifundefined{Crefrange@subappendix@format@middle}{%
      \let\Crefrange@subappendix@format@middle%
      \Crefrange@appendix@format@middle}{}%
    \@ifundefined{crefrange@subappendix@format@last}{%
      \let\crefrange@subappendix@format@last%
      \crefrange@appendix@format@last}{}%
    \@ifundefined{Crefrange@subappendix@format@last}{%
      \let\Crefrange@subappendix@format@last%
      \Crefrange@appendix@format@last}{}%
%
    \@ifundefined{cref@subsubappendix@format}{%
      \let\cref@subsubappendix@format%
      \cref@subappendix@format}{}%
    \@ifundefined{Cref@subsubappendix@format}{%
      \let\Cref@subsubappendix@format%
      \Cref@subappendix@format}{}%
    \@ifundefined{crefrange@subsubappendix@format}{%
      \let\crefrange@subsubappendix@format%
      \crefrange@subappendix@format}{}%
    \@ifundefined{Crefrange@subsubappendix@format}{%
      \let\Crefrange@subsubappendix@format%
      \Crefrange@subappendix@format}{}%
    \@ifundefined{cref@subsubappendix@format@first}{%
      \let\cref@subsubappendix@format@first%
      \cref@subappendix@format@first}{}%
    \@ifundefined{Cref@subsubappendix@format@first}{%
      \let\Cref@subsubappendix@format@first%
      \Cref@subappendix@format@first}{}%
    \@ifundefined{cref@subsubappendix@format@second}{%
      \let\cref@subsubappendix@format@second%
      \cref@subappendix@format@second}{}%
    \@ifundefined{Cref@subsubappendix@format@second}{%
      \let\Cref@subsubappendix@format@second%
      \Cref@subappendix@format@second}{}%
    \@ifundefined{cref@subsubappendix@format@middle}{%
      \let\cref@subsubappendix@format@middle%
```

```
    \cref@subappendix@format@middle}{}%
  \@ifundefined{Cref@subsubappendix@format@middle}{%
    \let\Cref@subsubappendix@format@middle%
    \Cref@subappendix@format@middle}{}%
  \@ifundefined{cref@subsubappendix@format@last}{%
    \let\cref@subsubappendix@format@last%
    \cref@subappendix@format@last}{}%
  \@ifundefined{Cref@subsubappendix@format@last}{%
    \let\Cref@subsubappendix@format@last%
    \Cref@subappendix@format@last}{}%
  \@ifundefined{crefrange@subsubappendix@format@first}{%
    \let\crefrange@subsubappendix@format@first%
    \crefrange@subappendix@format@first}{}%
  \@ifundefined{Crefrange@subsubappendix@format@first}{%
    \let\Crefrange@subsubappendix@format@first%
    \Crefrange@subappendix@format@first}{}%
  \@ifundefined{crefrange@subsubappendix@format@second}{%
    \let\crefrange@subsubappendix@format@second%
    \crefrange@subappendix@format@second}{}%
  \@ifundefined{Crefrange@subsubappendix@format@second}{%
    \let\Crefrange@subsubappendix@format@second%
    \Crefrange@subappendix@format@second}{}%
  \@ifundefined{crefrange@subsubappendix@format@middle}{%
    \let\crefrange@subsubappendix@format@middle%
    \crefrange@subappendix@format@middle}{}%
  \@ifundefined{Crefrange@subsubappendix@format@middle}{%
    \let\Crefrange@subsubappendix@format@middle%
    \Crefrange@subappendix@format@middle}{}%
  \@ifundefined{crefrange@subsubappendix@format@last}{%
    \let\crefrange@subsubappendix@format@last%
    \crefrange@subappendix@format@last}{}%
  \@ifundefined{Crefrange@subsubappendix@format@last}{%
    \let\Crefrange@subsubappendix@format@last%
    \Crefrange@subappendix@format@last}{}%
%
  \@ifundefined{cref@subsubsubappendix@format}{%
    \let\cref@subsubsubappendix@format%
    \cref@subsubappendix@format}{}%
  \@ifundefined{Cref@subsubsubappendix@format}{%
    \let\Cref@subsubsubappendix@format%
    \Cref@subsubappendix@format}{}%
  \@ifundefined{crefrange@subsubsubappendix@format}{%
    \let\crefrange@subsubsubappendix@format%
    \crefrange@subsubappendix@format}{}%
  \@ifundefined{Crefrange@subsubsubappendix@format}{%
    \let\Crefrange@subsubsubappendix@format%
    \Crefrange@subsubappendix@format}{}%
  \@ifundefined{cref@subsubsubappendix@format@first}{%
    \let\cref@subsubsubappendix@format@first%
    \cref@subsubappendix@format@first}{}%
```

```
\@ifundefined{Cref@subsubsubappendix@format@first}{%
  \let\Cref@subsubsubappendix@format@first%
  \Cref@subsubappendix@format@first}{}%
\@ifundefined{cref@subsubsubappendix@format@second}{%
  \let\cref@subsubsubappendix@format@second%
  \cref@subsubappendix@format@second}{}%
\@ifundefined{Cref@subsubsubappendix@format@second}{%
  \let\Cref@subsubsubappendix@format@second%
  \Cref@subsubappendix@format@second}{}%
\@ifundefined{cref@subsubsubappendix@format@middle}{%
  \let\cref@subsubsubappendix@format@middle%
  \cref@subsubappendix@format@middle}{}%
\@ifundefined{Cref@subsubsubappendix@format@middle}{%
  \let\Cref@subsubsubappendix@format@middle%
  \Cref@subsubappendix@format@middle}{}%
\@ifundefined{cref@subsubsubappendix@format@last}{%
  \let\cref@subsubsubappendix@format@last%
  \cref@subsubappendix@format@last}{}%
\@ifundefined{Cref@subsubsubappendix@format@last}{%
  \let\Cref@subsubsubappendix@format@last%
  \Cref@subsubappendix@format@last}{}%
\@ifundefined{crefrange@subsubsubappendix@format@first}{%
  \let\crefrange@subsubsubappendix@format@first%
  \crefrange@subsubappendix@format@first}{}%
\@ifundefined{Crefrange@subsubsubappendix@format@first}{%
  \let\Crefrange@subsubsubappendix@format@first%
  \Crefrange@subsubappendix@format@first}{}%
\@ifundefined{crefrange@subsubsubappendix@format@second}{%
  \let\crefrange@subsubsubappendix@format@second%
  \crefrange@subsubappendix@format@second}{}%
\@ifundefined{Crefrange@subsubsubappendix@format@second}{%
  \let\Crefrange@subsubsubappendix@format@second%
  \Crefrange@subsubappendix@format@second}{}%
\@ifundefined{crefrange@subsubsubappendix@format@middle}{%
  \let\crefrange@subsubsubappendix@format@middle%
  \crefrange@subsubappendix@format@middle}{}%
\@ifundefined{Crefrange@subsubsubappendix@format@middle}{%
  \let\Crefrange@subsubsubappendix@format@middle%
  \Crefrange@subsubappendix@format@middle}{}%
\@ifundefined{crefrange@subsubsubappendix@format@last}{%
  \let\crefrange@subsubsubappendix@format@last%
  \crefrange@subsubappendix@format@last}{}%
\@ifundefined{Crefrange@subsubsubappendix@format@last}{%
  \let\Crefrange@subsubsubappendix@format@last%
  \Crefrange@subsubappendix@format@last}{}%
```

Ditto for subfigures and subtables.

```
\@ifundefined{cref@subfigure@format}{%
  \let\cref@subfigure@format%
  \cref@figure@format}{}%
```

```
\@ifundefined{Cref@subfigure@format}{%
  \let\Cref@subfigure@format%
  \Cref@figure@format}{}%
\@ifundefined{crefrange@subfigure@format}{%
  \let\crefrange@subfigure@format%
  \crefrange@figure@format}{}%
\@ifundefined{Crefrange@subfigure@format}{%
  \let\Crefrange@subfigure@format%
  \Crefrange@figure@format}{}%
\@ifundefined{cref@subfigure@format@first}{%
  \let\cref@subfigure@format@first%
  \cref@figure@format@first}{}%
\@ifundefined{Cref@subfigure@format@first}{%
  \let\Cref@subfigure@format@first%
  \Cref@figure@format@first}{}%
\@ifundefined{cref@subfigure@format@second}{%
  \let\cref@subfigure@format@second%
  \cref@figure@format@second}{}%
\@ifundefined{Cref@subfigure@format@second}{%
  \let\Cref@subfigure@format@second%
  \Cref@figure@format@second}{}%
\@ifundefined{cref@subfigure@format@middle}{%
  \let\cref@subfigure@format@middle%
  \cref@figure@format@middle}{}%
\@ifundefined{Cref@subfigure@format@middle}{%
  \let\Cref@subfigure@format@middle%
  \Cref@figure@format@middle}{}%
\@ifundefined{cref@subfigure@format@last}{%
  \let\cref@subfigure@format@last%
  \cref@figure@format@last}{}%
\@ifundefined{Cref@subfigure@format@last}{%
  \let\Cref@subfigure@format@last%
  \Cref@figure@format@last}{}%
\@ifundefined{crefrange@subfigure@format@first}{%
  \let\crefrange@subfigure@format@first%
  \crefrange@figure@format@first}{}%
\@ifundefined{Crefrange@subfigure@format@first}{%
  \let\Crefrange@subfigure@format@first%
  \Crefrange@figure@format@first}{}%
\@ifundefined{crefrange@subfigure@format@second}{%
  \let\crefrange@subfigure@format@second%
  \crefrange@figure@format@second}{}%
\@ifundefined{Crefrange@subfigure@format@second}{%
  \let\Crefrange@subfigure@format@second%
  \Crefrange@figure@format@second}{}%
\@ifundefined{crefrange@subfigure@format@middle}{%
  \let\crefrange@subfigure@format@middle%
  \crefrange@figure@format@middle}{}%
\@ifundefined{Crefrange@subfigure@format@middle}{%
  \let\Crefrange@subfigure@format@middle%
```

```
      \Crefrange@figure@format@middle}{}%
    \@ifundefined{crefrange@subfigure@format@last}{%
      \let\crefrange@subfigure@format@last%
      \crefrange@figure@format@last}{}%
    \@ifundefined{Crefrange@subfigure@format@last}{%
      \let\Crefrange@subfigure@format@last%
      \Crefrange@figure@format@last}{}%
%
%   \begin{macrocode}
    \@ifundefined{cref@subtable@format}{%
      \let\cref@subtable@format%
      \cref@table@format}{}%
    \@ifundefined{Cref@subtable@format}{%
      \let\Cref@subtable@format%
      \Cref@table@format}{}%
    \@ifundefined{crefrange@subtable@format}{%
      \let\crefrange@subtable@format%
      \crefrange@table@format}{}%
    \@ifundefined{Crefrange@subtable@format}{%
      \let\Crefrange@subtable@format%
      \Crefrange@table@format}{}%
    \@ifundefined{cref@subtable@format@first}{%
      \let\cref@subtable@format@first%
      \cref@table@format@first}{}%
    \@ifundefined{Cref@subtable@format@first}{%
      \let\Cref@subtable@format@first%
      \Cref@table@format@first}{}%
    \@ifundefined{cref@subtable@format@second}{%
      \let\cref@subtable@format@second%
      \cref@table@format@second}{}%
    \@ifundefined{Cref@subtable@format@second}{%
      \let\Cref@subtable@format@second%
      \Cref@table@format@second}{}%
    \@ifundefined{cref@subtable@format@middle}{%
      \let\cref@subtable@format@middle%
      \cref@table@format@middle}{}%
    \@ifundefined{Cref@subtable@format@middle}{%
      \let\Cref@subtable@format@middle%
      \Cref@table@format@middle}{}%
    \@ifundefined{cref@subtable@format@last}{%
      \let\cref@subtable@format@last%
      \cref@table@format@last}{}%
    \@ifundefined{Cref@subtable@format@last}{%
      \let\Cref@subtable@format@last%
      \Cref@table@format@last}{}%
    \@ifundefined{crefrange@subtable@format@first}{%
      \let\crefrange@subtable@format@first%
      \crefrange@table@format@first}{}%
    \@ifundefined{Crefrange@subtable@format@first}{%
      \let\Crefrange@subtable@format@first%
```

```
    \Crefrange@table@format@first}{}%
  \@ifundefined{crefrange@subtable@format@second}{%
    \let\crefrange@subtable@format@second%
    \crefrange@table@format@second}{}%
  \@ifundefined{Crefrange@subtable@format@second}{%
    \let\Crefrange@subtable@format@second%
    \Crefrange@table@format@second}{}%
  \@ifundefined{crefrange@subtable@format@middle}{%
    \let\crefrange@subtable@format@middle%
    \crefrange@table@format@middle}{}%
  \@ifundefined{Crefrange@subtable@format@middle}{%
    \let\Crefrange@subtable@format@middle%
    \Crefrange@table@format@middle}{}%
  \@ifundefined{crefrange@subtable@format@last}{%
    \let\crefrange@subtable@format@last%
    \crefrange@table@format@last}{}%
  \@ifundefined{Crefrange@subtable@format@last}{%
    \let\Crefrange@subtable@format@last%
    \Crefrange@table@format@last}{}%
```

Ditto for enums.

```
% \begin{macrocode}
  \@ifundefined{cref@enumii@format}{%
    \let\cref@enumii@format%
    \cref@enumi@format}{}%
  \@ifundefined{Cref@enumii@format}{%
    \let\Cref@enumii@format%
    \Cref@enumi@format}{}%
  \@ifundefined{crefrange@enumii@format}{%
    \let\crefrange@enumii@format%
    \crefrange@enumi@format}{}%
  \@ifundefined{Crefrange@enumii@format}{%
    \let\Crefrange@enumii@format%
    \Crefrange@enumi@format}{}%
  \@ifundefined{cref@enumii@format@first}{%
    \let\cref@enumii@format@first%
    \cref@enumi@format@first}{}%
  \@ifundefined{Cref@enumii@format@first}{%
    \let\Cref@enumii@format@first%
    \Cref@enumi@format@first}{}%
  \@ifundefined{cref@enumii@format@second}{%
    \let\cref@enumii@format@second%
    \cref@enumi@format@second}{}%
  \@ifundefined{Cref@enumii@format@second}{%
    \let\Cref@enumii@format@second%
    \Cref@enumi@format@second}{}%
  \@ifundefined{cref@enumii@format@middle}{%
    \let\cref@enumii@format@middle%
    \cref@enumi@format@middle}{}%
  \@ifundefined{Cref@enumii@format@middle}{%
```

```
      \let\Cref@enumii@format@middle%
      \Cref@enumi@format@middle}{}%
    \@ifundefined{cref@enumii@format@last}{%
      \let\cref@enumii@format@last%
      \cref@enumi@format@last}{}%
    \@ifundefined{Cref@enumii@format@last}{%
      \let\Cref@enumii@format@last%
      \Cref@enumi@format@last}{}%
    \@ifundefined{crefrange@enumii@format@first}{%
      \let\crefrange@enumii@format@first%
      \crefrange@enumi@format@first}{}%
    \@ifundefined{Crefrange@enumii@format@first}{%
      \let\Crefrange@enumii@format@first%
      \Crefrange@enumi@format@first}{}%
    \@ifundefined{crefrange@enumii@format@second}{%
      \let\crefrange@enumii@format@second%
      \crefrange@enumi@format@second}{}%
    \@ifundefined{Crefrange@enumii@format@second}{%
      \let\Crefrange@enumii@format@second%
      \Crefrange@enumi@format@second}{}%
    \@ifundefined{crefrange@enumii@format@middle}{%
      \let\crefrange@enumii@format@middle%
      \crefrange@enumi@format@middle}{}%
    \@ifundefined{Crefrange@enumii@format@middle}{%
      \let\Crefrange@enumii@format@middle%
      \Crefrange@enumi@format@middle}{}%
    \@ifundefined{crefrange@enumii@format@last}{%
      \let\crefrange@enumii@format@last%
      \crefrange@enumi@format@last}{}%
    \@ifundefined{Crefrange@enumii@format@last}{%
      \let\Crefrange@enumii@format@last%
      \Crefrange@enumi@format@last}{}%
%
%   \begin{macrocode}
    \@ifundefined{cref@enumiii@format}{%
      \let\cref@enumiii@format%
      \cref@enumii@format}{}%
    \@ifundefined{Cref@enumiii@format}{%
      \let\Cref@enumiii@format%
      \Cref@enumii@format}{}%
    \@ifundefined{crefrange@enumiii@format}{%
      \let\crefrange@enumiii@format%
      \crefrange@enumii@format}{}%
    \@ifundefined{Crefrange@enumiii@format}{%
      \let\Crefrange@enumiii@format%
      \Crefrange@enumii@format}{}%
    \@ifundefined{cref@enumiii@format@first}{%
      \let\cref@enumiii@format@first%
      \cref@enumii@format@first}{}%
    \@ifundefined{Cref@enumiii@format@first}{%
```

```
      \let\Cref@enumiii@format@first%
      \Cref@enumii@format@first}{}%
    \@ifundefined{cref@enumiii@format@second}{%
      \let\cref@enumiii@format@second%
      \cref@enumii@format@second}{}%
    \@ifundefined{Cref@enumiii@format@second}{%
      \let\Cref@enumiii@format@second%
      \Cref@enumii@format@second}{}%
    \@ifundefined{cref@enumiii@format@middle}{%
      \let\cref@enumiii@format@middle%
      \cref@enumii@format@middle}{}%
    \@ifundefined{Cref@enumiii@format@middle}{%
      \let\Cref@enumiii@format@middle%
      \Cref@enumii@format@middle}{}%
    \@ifundefined{cref@enumiii@format@last}{%
      \let\cref@enumiii@format@last%
      \cref@enumii@format@last}{}%
    \@ifundefined{Cref@enumiii@format@last}{%
      \let\Cref@enumiii@format@last%
      \Cref@enumii@format@last}{}%
    \@ifundefined{crefrange@enumiii@format@first}{%
      \let\crefrange@enumiii@format@first%
      \crefrange@enumii@format@first}{}%
    \@ifundefined{Crefrange@enumiii@format@first}{%
      \let\Crefrange@enumiii@format@first%
      \Crefrange@enumii@format@first}{}%
    \@ifundefined{crefrange@enumiii@format@second}{%
      \let\crefrange@enumiii@format@second%
      \crefrange@enumii@format@second}{}%
    \@ifundefined{Crefrange@enumiii@format@second}{%
      \let\Crefrange@enumiii@format@second%
      \Crefrange@enumii@format@second}{}%
    \@ifundefined{crefrange@enumiii@format@middle}{%
      \let\crefrange@enumiii@format@middle%
      \crefrange@enumii@format@middle}{}%
    \@ifundefined{Crefrange@enumiii@format@middle}{%
      \let\Crefrange@enumiii@format@middle%
      \Crefrange@enumii@format@middle}{}%
    \@ifundefined{crefrange@enumiii@format@last}{%
      \let\crefrange@enumiii@format@last%
      \crefrange@enumii@format@last}{}%
    \@ifundefined{Crefrange@enumiii@format@last}{%
      \let\Crefrange@enumiii@format@last%
      \Crefrange@enumii@format@last}{}%
%
    \@ifundefined{cref@enumiv@format}{%
      \let\cref@enumiv@format%
      \cref@enumiii@format}{}%
    \@ifundefined{Cref@enumiv@format}{%
      \let\Cref@enumiv@format%
```

```
    \Cref@enumiii@format}{}%
  \@ifundefined{crefrange@enumiv@format}{%
    \let\crefrange@enumiv@format%
    \crefrange@enumiii@format}{}%
  \@ifundefined{Crefrange@enumiv@format}{%
    \let\Crefrange@enumiv@format%
    \Crefrange@enumiii@format}{}%
  \@ifundefined{cref@enumiv@format@first}{%
    \let\cref@enumiv@format@first%
    \cref@enumiii@format@first}{}%
  \@ifundefined{Cref@enumiv@format@first}{%
    \let\Cref@enumiv@format@first%
    \Cref@enumiii@format@first}{}%
  \@ifundefined{cref@enumiv@format@second}{%
    \let\cref@enumiv@format@second%
    \cref@enumiii@format@second}{}%
  \@ifundefined{Cref@enumiv@format@second}{%
    \let\Cref@enumiv@format@second%
    \Cref@enumiii@format@second}{}%
  \@ifundefined{cref@enumiv@format@middle}{%
    \let\cref@enumiv@format@middle%
    \cref@enumiii@format@middle}{}%
  \@ifundefined{Cref@enumiv@format@middle}{%
    \let\Cref@enumiv@format@middle%
    \Cref@enumiii@format@middle}{}%
  \@ifundefined{cref@enumiv@format@last}{%
    \let\cref@enumiv@format@last%
    \cref@enumiii@format@last}{}%
  \@ifundefined{Cref@enumiv@format@last}{%
    \let\Cref@enumiv@format@last%
    \Cref@enumiii@format@last}{}%
  \@ifundefined{crefrange@enumiv@format@first}{%
    \let\crefrange@enumiv@format@first%
    \crefrange@enumiii@format@first}{}%
  \@ifundefined{Crefrange@enumiv@format@first}{%
    \let\Crefrange@enumiv@format@first%
    \Crefrange@enumiii@format@first}{}%
  \@ifundefined{crefrange@enumiv@format@second}{%
    \let\crefrange@enumiv@format@second%
    \crefrange@enumiii@format@second}{}%
  \@ifundefined{Crefrange@enumiv@format@second}{%
    \let\Crefrange@enumiv@format@second%
    \Crefrange@enumiii@format@second}{}%
  \@ifundefined{crefrange@enumiv@format@middle}{%
    \let\crefrange@enumiv@format@middle%
    \crefrange@enumiii@format@middle}{}%
  \@ifundefined{Crefrange@enumiv@format@middle}{%
    \let\Crefrange@enumiv@format@middle%
    \Crefrange@enumiii@format@middle}{}%
  \@ifundefined{crefrange@enumiv@format@last}{%
```

```
      \let\crefrange@enumiv@format@last%
      \crefrange@enumiii@format@last}{}%
    \@ifundefined{Crefrange@enumiv@format@last}{%
      \let\Crefrange@enumiv@format@last%
      \Crefrange@enumiii@format@last}{}%
%
    \@ifundefined{cref@enumv@format}{%
      \let\cref@enumv@format%
      \cref@enumiv@format}{}%
    \@ifundefined{Cref@enumv@format}{%
      \let\Cref@enumv@format%
      \Cref@enumiv@format}{}%
    \@ifundefined{crefrange@enumv@format}{%
      \let\crefrange@enumv@format%
      \crefrange@enumiv@format}{}%
    \@ifundefined{Crefrange@enumv@format}{%
      \let\Crefrange@enumv@format%
      \Crefrange@enumiv@format}{}%
    \@ifundefined{cref@enumv@format@first}{%
      \let\cref@enumv@format@first%
      \cref@enumiv@format@first}{}%
    \@ifundefined{Cref@enumv@format@first}{%
      \let\Cref@enumv@format@first%
      \Cref@enumiv@format@first}{}%
    \@ifundefined{cref@enumv@format@second}{%
      \let\cref@enumv@format@second%
      \cref@enumiv@format@second}{}%
    \@ifundefined{Cref@enumv@format@second}{%
      \let\Cref@enumv@format@second%
      \Cref@enumiv@format@second}{}%
    \@ifundefined{cref@enumv@format@middle}{%
      \let\cref@enumv@format@middle%
      \cref@enumiv@format@middle}{}%
    \@ifundefined{Cref@enumv@format@middle}{%
      \let\Cref@enumv@format@middle%
      \Cref@enumiv@format@middle}{}%
    \@ifundefined{cref@enumv@format@last}{%
      \let\cref@enumv@format@last%
      \cref@enumiv@format@last}{}%
    \@ifundefined{Cref@enumv@format@last}{%
      \let\Cref@enumv@format@last%
      \Cref@enumiv@format@last}{}%
    \@ifundefined{crefrange@enumv@format@first}{%
      \let\crefrange@enumv@format@first%
      \crefrange@enumiv@format@first}{}%
    \@ifundefined{Crefrange@enumv@format@first}{%
      \let\Crefrange@enumv@format@first%
      \Crefrange@enumiv@format@first}{}%
    \@ifundefined{crefrange@enumv@format@second}{%
      \let\crefrange@enumv@format@second%
```

```
      \crefrange@enumiv@format@second}{}%
    \@ifundefined{Crefrange@enumv@format@second}{%
      \let\Crefrange@enumv@format@second%
      \Crefrange@enumiv@format@second}{}%
    \@ifundefined{crefrange@enumv@format@middle}{%
      \let\crefrange@enumv@format@middle%
      \crefrange@enumiv@format@middle}{}%
    \@ifundefined{Crefrange@enumv@format@middle}{%
      \let\Crefrange@enumv@format@middle%
      \Crefrange@enumiv@format@middle}{}%
    \@ifundefined{crefrange@enumv@format@last}{%
      \let\crefrange@enumv@format@last%
      \crefrange@enumiv@format@last}{}%
    \@ifundefined{Crefrange@enumv@format@last}{%
      \let\Crefrange@enumv@format@last%
      \Crefrange@enumiv@format@last}{}%
%
  \let\cref@language\relax%
}
```

Change History

v0.01
General: Initial version 1
v0.02
General: Complete rewrite 1
v0.03
General: Added reference ranges . . 1
v0.04
General: Renamed "cleveref" – first public release 1
v0.05
General: Added poorman option . . 1
v0.06
General: Poorman no longer writes extra .tmp file 1
v0.07
General: Allow prevention of reference range collapsing 1
v0.08
General: Made referencing commands robust, improved default formats, made starred versions of referencing commands for hyperref, allowed multiple consecutive empty references in list to prevent range collapsing, and fixed bugs. Phew. 1
v0.08.1
General: Bug fixes. 1
v0.09
General: Added an extra conjunction and format component, used when a list only contains a single pair of references or reference sub-lists. Formats are now constructed from customisable components, unless overridden using the old format definition commands. 1
v0.10
General: Allow optional argument to
label to override reference type. Removed hyperref and ntheorem options; support for these and amsmath is enabled automatically if those packages are loaded. 1
v0.11
General: Made referencing aware of appendices. 1
v0.11.1
General: Bug fixes. 1
v0.12
General: Restore working
ref* when hyperref is loaded – first release mirrored on www.dr-qubit.org 1
v0.13
General: Finally implemented sort and compress package options. Support amsmath
tag command. Added babel support for English, German and French. Added varioref support. 1

Index

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