The `cleveref` package*

Toby Cubitt
toby-cleveref@dr-qubit.org

2018/02/08

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 typeset 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.

Normally, the latest version of the `cleveref` package is available via CTAN. Occasionally, slightly newer "pre-release" versions are available at `www.dr-qubit.org/latex.php#cleveref` a little before they make their way onto CTAN.

*This document corresponds to cleveref 0.21.1, dated 2018/02/08.

Contents

1 Introduction

When "clever" is used in the name of a computer program, it usually indicates that the programmer is overly smug about his own achievements. But at the heart of the LaTeX philosophy also lies the idea that it is clever to delegate as much of the typesetting as possible to the computer, in order to achieve a beautiful — and above all consistent — visual appearance.

All of this probably applies to the `cleveref` package. Its goals are two-fold: to exploit all the information that LaTeX can collect about labels as intelligently as possible (clever processing), allowing 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 features by automatically formatting cross-references depending on what they refer to (chapter, section, equation, theorem, etc.). It can automatically format cross-references to multiple labels, and can sort lists of multiple cross-references, compresses consecutive labels into a reference range, and all kinds of other clever wizardry. It also does similar things with page references.

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 typeset as `Equation~\ref{eq1}` instead (i.e. without the abbreviation and without the parentheses)? What happens if you decide to change the theorem labelled `thm1` into a lemma?[1] What if you move `thm3` so that it appears (and is numbered) before `thm1`, meaning that references to the sequence of theorems 1 through 3 should now be ordered `Theorems~\ref{thm3} to~\ref{thm1}` (i.e. the other way around)? What if you decide you prefer references to multiple theorems to be written as `Theorems~\ref{thm1}--\ref{thm3}`? Any such change requires you to search through the entire LaTeX source of your document, modifying all references to equations, updating all references to `thm1`, re-ordering all references to `thm1` and `thm3`, and changing all the formatting of references to multiple theorems.

The `cleveref` package allows you to define the format for the different types of cross-references once-and-for-all in the preamble of your document. (Of course, sane default formats are provided, so you only have to redefine the format if you don't like the default for a particular cross-reference type.) 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 use the appropriate name when typesetting any cross-references to it. This makes it far easier to typeset cross-references uniformly across your whole document, as well as avoiding repetitively typing similar text for each and every cross-reference.

[1] Note that to allow `cleveref` to automatically infer the type of theorem, you need to load either the `ntheorem` or the `amsthm` package. See Section 14.1 for more details.

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 13). 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}` produces `eqs.~(1) to~(5)`. If you want to refer to multiple things at once, you can simply throw 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`. Finally, if you want a page reference, use `\cpageref` (and don't write "page" in front), if you want a page range, use `\cpagerefrange`, and if you want to refer to multiple pages, simply throw them all into a single `\cpageref`. Just as with `\cref` (above), `cleveref` will sort it all out for you automaticaly.

`Cleveref` supports a number of languages other than English, and also supports the `babel` and `polyglossia` packages for those languages. Either pass the desired language as an option to `cleveref`, or pass it as a global option to `\documentclass`. Note that if you're writing in a language in which nouns decline, the `\cref` and `\cpageref` commands may be less useful, as they always produce the cross-reference name in the nominative case.[2] In such languages, you may instead prefer to use the `\labelcref` and `\labelcpageref` commands. Unlike `\cref` and `\cpageref`, these don't produce the name in front of the cross-reference, so you must supply it (in the appropriate case) yourself. But they do still cope with multi-references, so you still gain some benefit from using `cleveref`.

3 Comparison with Other Packages

Given how useful automated cross-reference typesetting is, there are naturally a number of other LaTeX packages with similar goals to `cleveref`, most notably `varioref`, `fancyref`, `hyperref`'s `\autoref` command, and (for theorem-like environments) `ntheorem` with the `thref` option. (There are many others, but these

[2] Providing separate variants of the `cleveref` commands for each noun case quickly becomes more cumbersome than just typing the cross-reference name by hand. Trying to determine the appropriate case automatically would be tantamount to solving the full natural-language processing problem in `cleveref`. Check back in a century or so for this feature.

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 occasionally 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 with `fancyref`, you will at times be back to searching and replacing label names throughout your document. Not to mention missing out on all the other `cleveref` features, such as automatic sorting and compressing of consecutive references, `ntheorem` and `amsthm` support, precise control over hyperlinks, etc.

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 according to the context in which they are referenced, which can sometimes be very useful. For example, you may want cross-references at the beginning of a sentence formatted differently to cross-references in the middle of a sentence. E.g. you may want to use the abbreviation "eq.", but revert to "Equation" at the beginning of sentences (words at the start of sentences shouldn't be abbreviated in English). This is not possible with `varioref`. Perhaps even more significantly, `varioref`'s `\labelformat` implementation makes it impossible to typeset multiple references automatically; if you want to refer to equations `eq1` through `eq3`, with `varioref` you are back to typing `Eqs.~(\ref{eq1}) to~(\ref{eq3})` by hand. Not to mention missing out on all the other `cleveref` features. In fact, `cleveref` fully supports `varioref`, taking over responsibility for typesetting cross-references, whilst retaining (and even enhancing) all the `varioref` page-referencing magic.

The `hyperref` package's `\autoref` command typesets 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 pretty close. But surprisingly, even with `hyperref`, it is impossible to customise precisely which part of the cross-reference is made into a hyperlink in PDF documents; this is simple with `cleveref`. And it still remains impossible to typeset multiple references, have consecutive references sorted and compressed automatically, etc.

The `ntheorem` package (with the `thref` option) does the right thing when it comes to 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 every `\label` command to specify the label type. `Cleveref` works equally well with any type of cross-reference, as well as fully supporting `ntheorem`. And again, `cleveref` provides a number of additional features over `ntheorem`, such as multi-references, automatic sorting and compressing of consecutive cross-references, control over the placement of hyperlinks, etc.

4 Typesetting Cross-References

\cref To automatically typeset 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 typeset multiple cross-references (see below).

\Cref As it is very difficult[3] for LaTeX to determine whether a cross-reference appears at the beginning of a sentence or not, a beginning-of-sentence variant exists: `\Cref{`⟨*label*⟩`}`. By default, this typesets the cross-reference with the first letter capitalised, and without using an abbreviation in those cases where the standard variant does use one. (However, the formatting of the `\cref` and `\Cref` forms can be fully and independently customised, see Section 8.)

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

`\crefrange{`⟨*label1*⟩`}{`⟨*label2*⟩`}`

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

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

\cref* \Cref* \crefrange* \Crefrange* When `cleveref` is used along with the `hyperref` package (see Sections 8 and 13), additional starred variants of all the referencing commands are available. The standard referencing commands will make cross-references into hyperlinks; the starred variants prevent this, producing the same typeset text but without creating hyperlinks.

\cpageref \Cpageref To typeset a page reference, use `\cpageref{`⟨*label*⟩`}`, which is typeset e.g. as "page 3". At the beginning of a sentence, use `\Cpageref` instead. Since page references are always references to, well... pages, this doesn't gain you so much over `\pageref`. Where `\cpageref` comes into its own is in referring to multiple pages:

`\cpageref{`⟨*label1*⟩`,`⟨*label2*⟩`,`⟨*label3*⟩`,...}`

\cpagerefrange \Cpagerefrange Predictably enough, `\cpagerefrange` and `\Cpagerefrange` are used to typeset references to page ranges:

`\cpagerefrange{`⟨*label1*⟩`}{`⟨*label2*⟩`}`

\ref \pageref `Cleveref` does *not* modify the standard `\ref` or `\pageref` commands, so you can still use them to typeset the formatted label counter or page number alone, without any additional text or formatting.

\namecref \nameCref \lcnamecref \namecrefs \nameCrefs \lcnamecrefs Occasionally, it's useful to produce just the name of a reference, without the label itself. For example, if you want to refer to "this section", but you're not sure

[3] Actually, very likely impossible!

whether you might later change the section into a chapter, it might be useful to produce just the name "section" associated with the section's label. If you later change the section into a chapter, the text will then automatically change to "this chapter". The `\namecref` and `\nameCref` do exactly this:

```
\namecref{sec1}
```

is typeset as "section" (assuming `sec1` labels a section). The `\namecrefs` and `\nameCrefs` commands produce the plural forms. The `\lcnamecref` and `\lcnamecrefs` commands force the reference name to lowercase, for use when the `capitalise` option is enabled (see Section 7.1). (When that option is set, `\namecref` produces an uppercase reference name.)

There is a slight pitfall that you should be aware of when using the `\namecref` commands. They get the reference name from the names defined for the label's reference type using `\crefname` or `\Crefname` (see Section 8.1.2). The default reference formats provide these definitions. However, it is possible to customise reference formats using lower-level commands that do not create `\crefname` definitions (see Section 8.2). If the `\crefname` definitions are missing for a particular reference type, `\namecref` and `\nameCref` will produce errors for labels of that type. You can fix the error by adding explicit `\crefname` definitions for these types.

`\labelcref` Conversely, it is occasionally convenient to produce just the label part of a reference, without the cross-reference name. For example, this can be useful when writing in a language in which nouns decline. The `\labelcref` command does exactly this, and can also cope with multi-references, processing them just as `\cref` does. However, since it typesets a multi-reference without any name, *all* labels in a `\labelcref` multi-reference *must* be of the same type.

The `\labelcref` command will typeset cross-reference labels using the default label format if no type-specific format is defined using `\creflabelformat` (see Sections 8.1.1 and 8.1.2). Note that, if you customise reference formats using the low-level commands, you may want to also explicitly define the `\labelcref` formats to match, using the `\labelcrefformat` etc. commands (see Section 8.2).

`\labelcpageref` Similarly, `\labelcpageref` typesets the page numbers alone, without inserting "page" in front. Like `\cpageref`, it also handles multi-references. Like `\labelcref`, by default `\labelcpageref` typesets the page numbers using the default label format, customised using `\crefdefaultlabelformat`. If you want to define a separate format for `\labelcpageref`, use `\creflabelformat` to customise the label format for the "page" cross-reference type. (see Section 8.2).

5 Sorting and Compressing

When `cleveref` typesets lists of multiple cross-references or page-references, the default behaviour is to automatically sort the list and compress sequences of consecutive cross-references or page numbers 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 typeset as

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

or

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

will be typeset 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, typeset cross-reference:

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

will be typeset as

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

6 Overriding the Cross-Reference Type

`\label` A label's "type" is usually determined by the name of the counter it refers to, or in the case of `ntheorem` and `amsthm` theorem-like environments by the environment name. However, sometimes it is useful to override the type. `Cleveref` provides two different mechanisms for accomplishing this.

You can alias a counter to a different cross-reference type using the `\crefalias` command:

```
\crefalias{⟨counter⟩}{⟨type⟩}
```

⟨*counter*⟩ will then use the cross-reference formatting of ⟨*type*⟩. This can be useful if you want multiple counters to use the same cross-reference format.

Occasionally, you may want to override the cross-reference type for one particular label, one-off. 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,[4] *without* using the `amsthm` or `ntheorem` package (although using one of those packages is a better solution if available). For example,

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

Note that `aliascnt` must be loaded before `cleveref`, and any `\newaliascnt` commands *must* come *after* `cleveref` has been loaded.

[4] 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.

7 Options that Modify the Cross-Reference Format

7.1 Capitalising All Cross-Reference Names

`capitalise` Many authors prefer to always capitalise cross-reference names, regardless of where they appear in the sentence, writing Theorem 1 and Equation 3 (as opposed to theorem 1 and equation 3). If you count yourself among this group, you can pass the `capitalise` option to the `cleveref` package (`capitalize` also works).

All the default cross-reference formats will then have the first letter capitalised, as will the automatically generated `\cref` variants (see Sections 8.1.2 and 8.2). (However, if you explicitly define a `\cref` variant to *not* be capitalised, `cleveref` will still honour your definition. In other words, you're responsible for defining the capitalisation correctly in your own format definitions.)

You should *still* use the `\Cref` variants at the beginning of sentences, for one thing, because abbreviations should not be used at the beginning of a sentence,[5] and for another, in case you later change your mind and remove the `capitalise` option.

7.2 Including Names in Hyperlink Targets

`nameinlink` When using the `hyperref` package, `cleveref` automatically makes all cross-references into hyperlinks to the corresponding reference. By default, only the label itself forms part of the hyperlink target (i.e. the text you can click on to navigate to the cross-reference). The cross-reference name is not part of the hyperlink. By contrast, `hyperref`'s `\autoref` command *does* includes the name as part of the hyperlink. If you prefer to include the names in the hyperlinks when using `cleveref`, you can pass the `nameinlink` option to the `cleveref` package. (For even more control over the placement of the hyperlink target, use the commands for customising the cross-reference format. See Section 8.)

However, use of this option is discouraged on stylistic grounds. Firstly, when producing PDF output `hyperref` by default surrounds hyperlinks with red boxes, which looks particularly ugly when the entire cross-reference name is surrounded by a red box (though this unfortunate default can be changed using `hyperref` package options; see the `hyperref` documentation for details). Secondly, and more significantly, when using multi-references only the first reference in a group can include the cross-reference name as part of its hyperlink target, for obvious reasons. The hyperlink targets for the other references in the group will necessarily be just the labels. This makes for somewhat non-uniform typesetting of hyperlinks, with the first cross-reference in a multi-reference having a much larger hyperlink target than the others.

[5] At least in English; I'm not sure about other languages.

7.3 Abbreviations in Cross-Reference Names

`noabbrev` The default cross-reference names for some languages use common abbreviations for some of the names (e.g. in the default English format, `\cref{eq1}` will be typeset as `eq.~(1)`). Some authors may prefer to always use the full name, rather than an abbreviation (`equation~(1)` instead of `eq.~(1)`). To disable all use of abbreviations in the default cross-reference names, pass the `noabbrev` option to the `cleveref` package.

Note that the default names *never* use abbreviations for the start-of-sentence variants (`\Cref` etc.) This is because in good written English (and likely other languages too), abbreviations should never be used at the beginning of a sentence. Many of TEX's default settings (e.g. page margins) are specifically chosen to encourage good typesetting style. `Cleveref` tried to follow the same philosophy. If despite this you insist on using abbreviations at the start of sentences, you will need to customise the start-of-sentence formats yourself.

8 Customising the Cross-Reference Formats

The `cleveref` package allows you to take full control of the typesetting of cross-references, by allowing the formatting to be customised. Defaults appropriate for English documents are provided for the standard label types,[6] and support for a number of languages is provided via package options (see Section 10). But if you don't like the defaults, or are writing in a language that is not supported yet,[7] 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 typeset 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.

[6] 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" for the plural of "lemma".

[7] Any contributions of translations for missing languages are very welcome! See Section 14.3 for information on how to contribute translations.

`Cleveref` treats page references, as produced e.g. by `\cpageref`, as cross-references with the type "page". Therefore, all of the mechanisms for customising cross-references apply equally well to page references, simply by using "page" as the cross-reference type.

8.1 Customising the Cross-Reference Components

8.1.1 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. `\thesection`). The other two are used to mark the beginning and end of the part of the cross-reference that should form the hyperlink when the `hyperref` package is used (see Section 13). For example, if you wanted to surround all labels with square brackets, without the square brackets themselves being part of the hyperlink, you would need:

```
\crefdefaultlabelformat{[#2#1#3]}
```

The hyperlink arguments `#2` and `#3` *must* appear in that order. (Leaving them out completely will not cause an error, but in that case no hyperlink will be created when `hyperref` is used, and there are better ways to achieve this. See Sections 4 and 13.)

Note that the default format for equation cross-references already overrides `\crefdefaultlabelformat` in order to surround the label with parentheses, so the redefining `\crefdefaultlabelformat` will have no effect on equations. The label format for equations must be customised separately if you want to change it (see Section 8.1.2).

`\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 endash "`--`". `\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 typeset 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}{--}
```

does not.

\crefrangepreconjunction
\crefrangepostconjunction

There are two other "conjunction" commands available for customizing the formatting for reference ranges. These are `\crefrangepreconjunction` and `\crefrangepostconjunction`, which insert text before the first label defining the range, and after the second label, respectively. For example, when these commands are defined, `\crefrange{thm1}{thm2}` is typeset as

```
theorems~\crefrangepreconjunction\ref{thm1}
 ➥ \crefrangeconjunction\ref{thm2}\crefrangepostconjunction
```

These commands are not used in the default English format definitions, but they are needed in some languages to correctly express a range. For example, the Italian format defines `\crefrangepreconjunction` to be "da", so that `\crefrange{thm1}{thm2}` produces

```
teorema da~\ref{thm1} a~\ref{thm2}
```

\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.[8] You can override this by defining the conjunction commands `\crefpairgroupconjunction`, `\crefmiddlegroupconjunction` and `\creflastgroupconjunction`.

For example,

[8] 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.)

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

is typeset as

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

8.1.2 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 automatically define `\Crefname` to be a capitalised version of `\crefname`, using `\MakeUppercase`. Conversely, if the corresponding `\crefname` is undefined when `\Crefname` is called, it will automatically 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` of `amsthm` packages are 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` and `amsthm` provide 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`, which inherit from `enumi`. 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 8.1.1). 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 hyperlink when the `hyperref` package is loaded (see Section 13). `#2` and `#3` *must* appear in that order.

`\crefrangelabelformat`

Normally, the start and end references in a reference range are typeset using the usual label format (as defined by `\crefdefaultlabelformat` or `\creflabelformat`) separated by `\crefrangeconjunction` (Section 8.1.1). 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 hyperlink for the first reference, the final two (`#5` and `#6`) the hyperlink for the second reference. The hyperlink arguments *must* appear in order. For example,

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

8.1.3 Automatic `\newtheorem` Definitions

`\newtheorem`

The standard LaTeX `\newtheorem` command for defining new theorem-like environments provides enough information to deduce a reasonable cross-reference name for the new environment. So `cleveref` automatically defines an appropriate cross-reference name for new theorem-like environments. This automatic definition is only used if no default definition is provided by `cleveref` itself, and if no `\crefname` or `\Crefname` definition is given explicitly (see Section 8.1.2).

The caveat with this automatic definition is that, although `\newtheorem` essentially provides the singular form of the cross-reference name, it doesn't provide the plural form. And there is no reliable way of constructing the plural form from the singular.[9] Therefore, if the plural form is ever required, `cleveref` will produce a "reference type undefined" warning, and typeset the cross-reference where the plural form is required as:

?? `\ref{`⟨*label*⟩`}` ...

[9] If you're a native English-speaker, you might think that just adding an 's' would work, though a moment's thought will provide examples of words where this will fail. If you're a non-English speaker, it probably won't even occur to you to claim that plurals can reliably be constructed automatically!

In this case, you will have to provide an explicit `\crefname` or `\Crefname` definition yourself, to define the plural form as well as the singular form.

Note that this has *nothing whatsoever* to do with automatically determining the type of theorem-like environment in a cross-reference! For that, you need to load either the `ntheorem` or the `amsthm` package. See Section 14.1 for more details.

8.2 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.[10] 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 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 produce identical typeset text; the format definitions must contain identical LaTeX code.)

Note that if you use the low-level customisation commands, you might still want to provide `\crefname` and `\Crefname` definitions too, so that the `\namecref` commands will work (see Section 4).

8.2.1 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 the `ntheorem` or `amsthm` package is loaded*, in which case it is the environment name,

[10] To be precise, `cleveref` checks whether the `\crefformat` definitions match.

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 `subsection`), "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 when `\Crefformat` is called, it will define the `\crefformat` to produce a lowercase 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 8.1.1).

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 hyperlink when the `hyperref` package is used, and *must* appear in that order (see Section 13).

As an example,

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

will typeset equation references as

Eq. (⟨*counter*⟩)

with the counter (excluding the parentheses) forming the hyperlink.

Note that the hyperlink 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 the "Eq." to be part of the hyperlink, you would have to explicitly define:

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

8.2.2 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` or `amsthm` are

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 hyperlink for the first cross-reference, and the final two (`#5` and `#6`) mark the beginning and end of the second cross-reference's hyperlink.

As an example,

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

would typeset equation reference ranges as

eqs. (⟨*counter1*⟩) to (⟨*counter2*⟩)

with the counters (excluding the parentheses) forming the hyperlinks.

8.2.3 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 8.2.1 and 8.2.2), 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 when the `ntheorem` or `amsthm` packages are 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 `\crefrangemultiformat` and `\Crefrangemultiformat`). 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 snippets are typeset one after another in a multi-reference, with no space separating them. You may or may not want spaces at the beginning or 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.

8.2.4 Label Cross-References

If you define the format for a particular cross-reference type using the low-level customisation commands, and still want to use the `\labelcref` command to produce just the label part of the cross-reference, then you must also define the appropriate `\labelcref` formats for that type. This is done using the `\labelcrefformat`, `\labelcrefrangeformat`, `\labelcrefmultiformat` and `\labelcrefrangemultiformat` commands. Their syntax is identical to that of the corresponding `\crefformat`, `\crefrangmeformat`, `\crefmultiformat` or `\crefrangemultiformat` command. Typically, the `\labelcref` formats should be defined identically to the standard `\cref` formats, except for the ⟨*first*⟩ part, which should leave off the cross-reference name. This is not enforced, however.

9 Advanced Cross-Reference Formating

When you define a custom cross-reference format using `\creflabelformat`, `\creformat` et al. (see Section 8), you're not merely defining a pattern with placeholders to be filled in. You're really defining the body of a LaTeX macro, with the formatted labels as arguments. This is a very powerful tool. It means that the only limit on how you can process the labels is your ability to code it in TeX![11] Which potentially allows for very sophisticated cross-reference formatting.

One example of this is removing common prefixes from reference ranges. E.g. if you're numbering equations within sections, and `eq1`, `eq2` and `eq3` are all in the section 1.2, then you might want to typeset `\cref{eq1,eq2,eq3}` as "eqs. (1.2.1–3)" instead of "eqs. (1.2.1) to (1.2.3)". Similarly, if `eq1a`, `eq1b` and `eq1c` are `amsmath` subequations, you might want to typeset `\cref{eq1a,eq1b,eq1c}` as "eqs. (1a–c)" instead of "eqs. (1a) to (1c)".

`Cleveref` provides a useful utility macro for this: `\crefstripprefix`, which takes two strings as arguments, and returns the second one with any common prefix stripped off. (However, the very last run of digits or letters in the string is

[11] And since TeX is Turing-complete, that means you can do anything you like short of solving the Halting Problem.

retained in its entirety, even if it has a part in common.) With the help of this macro, you can produce the desired reference-range formatting with:

```
\crefrangelabelformat{equation}
 ➥ {(#3#1#4--#5\crefstripprefix{#1}{#2}#6)}
\crefrangelabelformat{subequation}
 ➥ {(#3#1#4--#5\crefstripprefix{#1}{#2}#6)}
```

10 Language, `babel` and `polyglossia` support

`Cleveref` supports different languages via package options, in the usual way, though not all languages are supported yet.[12] Basic `cleveref` language support will work even if `babel` or `polyglossia` are not loaded. The only exception currently is Catalan (which requires the `\lgem` command provided by these packages).

The `babel` package is fully supported if it is loaded, allowing you to change the language used in cross-references using the `babel` language switching commands, such as `\selectlanguage` and `\foreignlanguage`. Similar support is provided for the `polyglossia` babel replacement package.

Note that when using `babel`, you still need to tell `cleveref` which language it should use for the default cross-reference formats. It is *not* sufficient to pass the language option to `babel` alone. You *must* also *either* pass the language options to `cleveref` package directly when loading it:

```
\usepackage[⟨language⟩]{cleveref}
```

or specify the desired language globally as a document class option:

```
\documentclass[⟨language⟩]{⟨class⟩}
\usepackage{babel}
\usepackage{cleveref}
```

The latter method is strongly recommended. LaTeX automatically passes document class options to *every* loaded package. So specifying the language as a global option causes the appropriate language support to be enabled automatically in every package that supports it.

When writing multi-language documents, you may need to specify multiple language options in order to load `babel` support for all of them. In this case, `babel` sets the initial document language to the *last* language option. (See the `babel` documentation for more details.) `Cleveref` does the same: the last language in the option list determines the language for the initial cross-reference format definitions; additional language options load `cleveref` support for switching between those languages.

[12] Contributions of translations for missing languages are very welcome! See Section 14.3 for information on how to contribute translations.

`Polyglossia` uses a different mechanism for selecting and loading languages, and ignores package language options entirely. The default language must be set using `\setdefaultlanguage`, and additional languages are loaded using `\setotherlanguage`. `Cleveref` recognises these commands, so you should *not* pass language options to `cleveref` when using `polyglossia`. (Passing language options to `cleveref`, either as package options or global options, will override the default language set by `polyglossia`'s `\setdefaultlanguage`.) Note that the `\setdefaultlanguage` option *must* come before `cleveref` is loaded, so that `cleveref` knows what default language you want. (If you don't do this, `cleveref` will generate a warning message in the log.)

The `babel` and `polyglossia` 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` `babel` command (or similar) in the document body will override these customisations, replacing them with the defaults for the newly selected language. If you later use `\selectlanguage` to switch back to the main language, any customisations from the preamble will be restored. If you want to customise 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` or `polyglossia`'s language switching mechanism. (See section "Language and `babel` Support" in the full implementation documentation, and the `babel` or `polyglossia` package 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` or `polyglossia`'s language switching mechanisms.

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` or `polyglossia` 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@`⟨*type*⟩`@name`, `\Cref@`⟨*type*⟩`@name`, `\cref@`⟨*type*⟩`@name@plural`, and `\Cref@`⟨*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 hyperlink,[13] but you still want to be able to switch language using `babel` or `polyglossia`, you would need something like:

[13] This is merely as an example. Including names in hyperlinks is more easily accomplished by setting the `nameinlink` package option.

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

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

Note that if you define an empty cross-reference name for some type using an empty `\crefname`, e.g. for equations

```
\crefname{equation}{}{}
```

then the empty cross-reference name will be retained when switching languages. This is probably what you want anyway.

11 The `cleveref.cfg` File

If `cleveref` finds a `cleveref.cfg` file somewhere in the LaTeX search path, it automatically loads any definitions found in that file. (For details of which directories LaTeX searches, consult the documentation for your site's TeX installation.) The main use of `cleveref.cfg` is to store any cross-reference format customisations that you want to use in every document you write, so that you don't have to include them explicitly in every document's preamble.

12 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 `cleveref` 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 elsewhere, you can process it through the `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 typesetting of the original `cleveref` cross-references in all cases.[14] In particular, if you're using the `hyperref` package (see Section 13) to turn cross-references into hyperlinks, any customisation of hyperlinks will be lost. And if you're using the `varioref` package (see Section 13), you may need to manually tweak the spacing in front of some of the `varioref` commands in the document produced by the `sed` script.

13 Interaction with Other Packages

The `cleveref` package *must* be loaded *after* all other packages that don't specifically support it,[15] i.e. the

```
\usepackage{cleveref}
```

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

`Cleveref` tries as far as possible to minimise its impact on the standard LaTeX cross-referencing machinery, allowing it to work alongside many of the other packages that also enhance LaTeX's cross-referencing features, though it can occasionally interact badly with packages that redefine the same core LaTeX commands. Beyond peacefully co-existing with many packages, `cleveref` includes specific support for a number other packages, allowing it to integrate its clever cross-referencing features with the features provided by these packages: `babel`, `polyglossia`, `hyperref`, `varioref`, `ntheorem`, `amsthm`, `aliascnt`, `subfig`, `algorithmicx`[16], `algorithm2e`, `listings`.

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

[15] At the time of writing, the only packages I'm aware of that should be loaded after `cleveref` are the `hypdvips` and `autonum` packages.

[16] The `algorithmic` package is *not* supported.

Cleveref implements a significantly enhanced version of the features found in the fancyref package, ntheorem's thref option, and varioref's \labelformat command. Although these features may (or may not) work correctly alongside cleveref, there is no good reason to use them when using cleveref, and their use is unsupported. (Note that varioref *is* fully supported by cleveref, just that cleveref's features supersede varioref's \labelformat feature. Similarly, ntheorem is fully supported and even recommended, only the thref option is superseded by cleveref.)

\thref
\vref
\Vref
\vrefrange
\Vrefrange
\fullref
\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*.

\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 of the varioref commands to be used for suppressing hyperlinks when the hyperref package is loaded, as usual. (Unfortunately, due to lack of support for this in varioref, the page references will still sometimes be hyperlinks, even when using the starred variants. Go bug the varioref maintainer about this if you don't like it.)

Cleveref is currently incompatible with the mathtools package's showonlyrefs option, which automatically labels only those equations that are cross-referenced. The autonum package provides a possible alternative, which implements similar features in a cleveref-compatible manner.

14 Known Bugs, Non-Bugs, and Possible Improvements

14.1 Non-Bugs

The following are *not* bugs. They are either intentional behaviour, unavoidable behaviour, or are caused by LaTeX misunderstandings:

- If you are using both varioref and hyperref, *make sure you are loading them in the correct order*, otherwise cross-references will reference completely the wrong thing *without any warning in the LaTeX output or log!* The packages *must* be loaded in the following order: varioref, hyperref, cleveref.
- Cleveref on its own won't automatically infer the type of theorem-like environment you're referring to in a cross-reference. Cross-references to all theorem-like environments will use the same name, "theorem". To allow the

theorem type to be determined automatically, you need to load either the `ntheorem` or the `amsthm` package. Also note that all `\newtheorem` definitions must be placed *after* the `cleveref` package is loaded.

- Due to the way TEX parses arguments, you have to be a little careful when using `\label` inside an optional argument to another command. `\label{`⟨*label*⟩`}` will work, but trying to pass an optional argument `\label[`⟨*type*⟩`]{`⟨*label*⟩`}` will fail. If you want to pass an optional argument to `\label` whilst already within an optional argument to some other command, you must surround the entire label command with braces: `{\label[`⟨*type*⟩`]{`⟨*label*⟩`}}`. This crops up e.g. when adding labels to subfigure subcaptions in the `memoir` document class. A simpler solution in this particular case is to define the label in the subfigure body, instead of in the subcaption.

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

- If you are using `babel`, you *must still* pass the appropriate language option to `cleveref`, as well as to `babel`. Passing it to `babel` alone is *not* sufficient (you will get the default English cross-reference formats). The best way to set the document language is as a global option in the `\documentclass` line. (This is standard LATEX and `babel` practice – read up on LATEX's option handling for more detail.)

- `Cleveref` can't cope with active characters being present in cross-reference label names. For example, if French `babel` support is loaded, the commonly used ":" in label names will often fail, spewing the usual random selection of mysterious TEX errors that accompany such deep-seated errors. The solution is to avoid using active characters in label names. (You may need to consult the `babel` documentation to discover which active characters are defined in your language.)

- The `poorman sed` script loses any custom `cleveref` hyperlink formatting you might have defined, and does not always reproduce the original spacing around the `varioref` commands when `varioref` is used. This is not a bug; it is a side-effect of the intended purpose of the `poorman` option. The philosophy behind `poorman` 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 e.g. academic journals, thereby defeating the whole purpose of the `poorman` option.

14.2 Known Bugs and Work-Arounds

In rough order of significance:

- 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`.

- When using `varioref` and `hyperref` with `cleveref`, the `cleveref` `nameinlink` option will not cause the word "page" in the page-reference part of a `\vref` (or other `varioref`) command to be included in the hyperlink, nor will the "on the previous page" (or similar) text produced by `\vref` be hyperlinked. This is not strictly speaking a `cleveref` issue. It is the normal behaviour of the `hyperref`-enhanced version of `varioref`'s `\vpageref` command, which `cleveref` uses to produce the page references in its enhanced `\vref` command. (This *might* be improved in a future version by partially overriding `hyperref`.)

- `Cleveref` doesn't know about the `subfloat` package, so you have to revert to using `\ref` for cross-references to sub-figures. (Might be fixed in a future version.)

- The `beamer` document class redefines the `\label` command in a particularly devious way that breaks `cleveref`'s optional argument to that command. (Might be fixed in a future version.)

- `Cleveref` is incompatible with the `showonlyrefs` option of the `mathtools` package, though it should be compatible with the rest of `mathtools`. (Might be fixed in a future version.) The `autonum` package, which provides similar functionality and *is* designed to be `cleveref`-compatible, is a possible alternative.

- `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.

14.3 Possible New Features and 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 *much* more work[17] than I'm prepared to invest.

- `Cleveref` doesn't include support for all languages yet. 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 in the implementation (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.

15 Thanks

A number of people have helped improve `cleveref` by contributing code and translations. Thanks to Michael Ummels for contributing the `amsthm` support code, and to Stefan Pinnow, Gonzalo Medina, Massimo Redaelli, Philip Hölzenspies, Aleksander Gorohovski, Benjamin Høyer, Johannes Mueller, Paulo Roberto Massa Cereda, Simon Sigurdhsson, Rafel Jaume Deyà and Eva Bosch Roura for contributing translations. Thanks also to Susanna Goldschmidt for additional help with the translations.

Many people have suggested improvements or reported bugs – indeed, many have put significant effort into helping investigate and fix them. So thanks (in alphabetical order) to: Adrian Knoth, Akim Demaille, Alan Munn, Aleksander Gorohovski, Amar Ghaisas, Anand Deopurkar, Andreas Haselbacher, Arne Meier, Bas Ploeger, Christian Tuma, Dan Luecking, David Gleich, Denis Bitouzé, Domenic Denicola, Donald Arseneau, Eric Ahlberg, Frank Mittlebach, Hendrik Maryns, Iain Cunningham, Ingolf Becker, James Sharam, Jens Mueller, Joel C. Salomon, Jonas Nyrup, Joris Pinkse, Kristian Debrabant, Leo Shidai Liu, Lev Bishop, Mak Trifkovic, Matej Batic, Matt Gately, Matthew Skala, Michael Barber, Michael Gorven, Michal Kaut, Mico Loretan, Milania, Nicolas Dudebout, Olivier Roy, Patrick Häcker, Paul Gomme, Ricardo de Aldama Sánchez, Robert Fischer, Sebastian Ørsted, Simon Spiegel, Stefan Pinnow, Steve Dower, Ted Pavlic, Thomas Arildsen, Tobias Jores, Uwe Lück and Vadim Makarov for their help. (If I've inadvertently missed you out, please let me know!)

[17] 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!