Writing LaTeX

Some do's and don'ts

Jendrik Stelzner

June 17, 2022[*]

Available at `https://gitlab.com/cionx/writing-latex`.
Please send comments and corrections to `stelzner@uni-bonn.de`.

[*]Last Changes: `2022-02-04 21:25:45 +0100` Commit: `39c7fb1`

Preface

Das Auge liest mit.

German saying

This text presents some thoughts about what (not) to do in mathematical typesetting using LaTeX. Examples are provided using the environment `tcblisting` from the package `tcolorbox`. This text is not intended as a first introduction to LaTeX. Some of the following remarks are subject to individual taste.

Contents

Preface ii

List of tables vii

List of figures viii

1 Before you start writing 1

- 1.1 Use version control 1
- 1.2 The compilation process 1
  - 1.2.1 Use XƎLaTeX or LuaLaTeX 1
  - 1.2.2 Don't ignore warnings 1
- 1.3 Code quality 2
  - 1.3.1 Make use of whitespace to organize the code 2
  - 1.3.2 Write one sentence per line 3
  - 1.3.3 Properly indent your source code 3
  - 1.3.4 Use `%` for problematic line breaks 3
- 1.4 Splitting up a project into multiple files 4
  - 1.4.1 The commands `\usepackage`, `\include`, `\input` 4
  - 1.4.2 An Example 4
  - 1.4.3 Use `\includeonly` 6
- 1.5 Choosing the document class 8
  - 1.5.1 Use the KOMA-Script classes 8
  - 1.5.2 Deciding on a document class 9
- 1.6 Use `\frontmatter` and its friends 10
- 1.7 Be consistent 12

2 Useful and important packages 13

- 2.1 `microtype` for better typesetting 13
- 2.2 `mathtools` for mathematics 13
- 2.3 `amsthm` for theorem-like environments 13
- 2.4 `tikz-cd` for commutative diagrams 16
- 2.5 `cleveref` and `hyperref` for referencing 17
  - 2.5.1 Recalling basic referencing 17
  - 2.5.2 `cleveref` 19
  - 2.5.3 `hyperref` 21
  - 2.5.4 Order of inclusion 21
- 2.6 `csquotes` for quotation marks 21

2.7 `enumitem` for configuration of lists . . . 23
2.7.1 Style of numbering . . . 24
2.7.2 Resuming lists . . . 25
2.7.3 Spacing . . . 26
2.7.4 Global settings . . . 31
2.7.5 Nested lists . . . 32
2.7.6 Cloning lists . . . 34
2.8 `biblatex` for bibliography . . . 35
2.8.1 The basic setup . . . 35
2.8.2 Creating the `bib`-file . . . 36
2.8.3 Citing the references . . . 42
2.9 `booktabs` for tables . . . 44
2.9.1 Recalling basic tables . . . 44
2.9.2 Problems and solutions . . . 47

3 Writing text 49
3.1 Obey orthography . . . 49
3.2 Proper punctuation . . . 49
3.2.1 A sentence ends with punctuation . . . 49
3.2.2 Punctuation in commutative diagrams? . . . 50
3.2.3 Lists contain punctuation . . . 52
3.2.4 Hyphen and dashes . . . 53
3.2.5 Correct spacing after a period . . . 55
3.3 Text layout . . . 59
3.3.1 Use non-breakable space . . . 59
3.3.2 Don't use `\\` or `\newline` . . . 60
3.3.3 Text consisting of lists . . . 61
3.4 Mathematics in text . . . 65
3.4.1 Don't break inline mathematics . . . 65
3.4.2 Don't begin a sentence with a mathematical symbol . . . 65
3.4.3 Don't put two formulas next to each other . . . 67
3.4.4 Don't begin a line with a mathematical symbol (optional) . . . 67
3.5 Text in mathematics . . . 68
3.5.1 Use `\text` . . . 68
3.5.2 Put space around text in display mathematics . . . 69
3.5.3 Use `\intertext` and `\shortintertext` . . . 71
3.6 Mathematics as text . . . 74
3.6.1 Don't abuse math as text . . . 74
3.6.2 Logical symbols don't replace text . . . 74
3.6.3 Don't use "iff" . . . 76

4 Good behaviour inside math mode 77
4.1 Using the right commands and symbols . . . 77
4.1.1 General symbols . . . 77
4.1.2 Operations . . . 77

4.1.3 Negations . . . 77
4.1.4 Don't use `\limits` . . . 80
4.1.5 Use extensible arrows instead of `\overset` and `\underset` . . . 82
4.1.6 Know your ellipses . . . 84
4.1.7 Know your matrices . . . 86
4.2 Avoid bad notation . . . 88
4.2.1 Don't use `\subset` . . . 88
4.2.2 Don't underline . . . 89
4.2.3 Use $\mathrm{d}x$ instead of dx . . . 89
4.2.4 Don't force fancy fractions . . . 89
4.3 Defining new commands . . . 90
4.3.1 `\DeclareMathOperator` . . . 90
4.3.2 `\operatorname` . . . 91
4.3.3 Don't abuse `\mathrm` . . . 92
4.3.4 `\newcommand` . . . 92
4.3.5 `\DeclarePairedDelimiter` . . . 93
4.3.6 The package `xparse` . . . 95
4.4 Stretch your arrows . . . 98
4.4.1 Stretching columns . . . 98
4.4.2 Stretch rows . . . 100
4.5 Beware of spacings . . . 102

5 Mathematical layout **104**
5.1 Environments for display mathematics . . . 104
5.1.1 Don't use `$$ $$` or `eqnarray` . . . 104
5.1.2 The environments `\[ \]` and `equation*` . . . 104
5.1.3 The environment `gather*` . . . 105
5.1.4 The environments `align*` and `alignat*` . . . 106
5.1.5 Don't use `center` plus `$ $` . . . 108
5.1.6 The environments `gathered`, `aligned` and `alignedat` . . . 108
5.1.7 Overview . . . 108
5.2 Where to break and align formulas . . . 109
5.2.1 When to break a formula . . . 110
5.2.2 Where to break and align a formula . . . 113
5.2.3 Don't break formulas badly . . . 117
5.3 Aligning nearly aligned formulas . . . 120
5.4 Proper spacing before multi-line display mode environments . . . 121
5.5 Use the `cases` environment . . . 123
5.6 Proper placement of the qed-symbol . . . 124
5.7 Tagging and numbering . . . 128
5.7.1 Don't autonumber all formulas . . . 128
5.7.2 `\tag` . . . 128
5.7.3 `\notag` . . . 129
5.8 Multi-line set descriptions . . . 130

Bibliography **131**

List of examples **132**

Index **138**

List of tables

2.1 The commands `\*name*ref`. 20
2.2 Possible labels for the environment `enumerate`. 24

3.1 Kinds of hyphen and dashes. 54
3.2 The Clebsch–Gordan coefficients in different languages. 54
3.3 Spacings after a period (with a zoom of factor 3.5). 55

4.1 Right symbols and wrong symbols. 78
4.2 Binary operation and operator version. 79
4.3 Negated versions of popular symbols. 79
4.4 Extensible arrows in `amsmath`, `mathtools` and `extarrows`. 84
4.5 Ellipses with a specific functions. 85
4.6 Ellipses with a specific orientation. 85
4.7 Different kinds of matrices. 87
4.8 Standard distances for `column sep`. 99
4.9 Standard distances for `row sep`. 101

List of figures

1.1 Splitting up a project into multiple files. 5

3.1 Structure tree of the definition. 69
3.2 Structure tree of the mathematical expression. 69

5.1 Deciding on a math environment. 110

Chapter 1

Before you start writing

A dozen mistakes and bad choices can be made before even the first word of the actual text is written. In this chapter we talk about how to prevent this from happening. Some of the points – those about code quality and code management – are not unique to writing LaTeX.

1.1 Use version control

Use some kind of sensible version control for your code. The author uses Git but has also heard good things about Mercurial. Sending code to yourself on Facebook is not a proper system of version control.

1.2 The compilation process

1.2.1 Use XeLaTeX or LuaLaTeX

Use XeLaTeX or LuaLaTeX for out-of-the-box Unicode support. The author generally recommends LuaLaTeX because the `microtype` package his only a limited functionality under XeLaTeX.

In the following we always talk about "the LaTeX compiler", whether it may `latex`, `lualatex` or `xelatex`.

1.2.2 Don't ignore warnings

There are two ways in which LaTeX will tell you that something went wrong: Errors and warnings.

The occurrence of an error means that LaTeX was unable the process the source code and gave up at some point in the process. In the case of a warning LaTeX again wasn't able to process the source code, but decided to produce some output nevertheless. Hence the only difference between an error and a warning is how LaTeX decided to proceed with it. But there is no intrinsic difference between the two of them.

It is typically hard to ignore errors, as no output file will be generated. But warnings also shouldn't be ignored: While the occurrence of an error means that you get no output at all, the occurrence of a warning means that you're getting a faulty output.

1.3 Code quality

1.3.1 Make use of whitespace to organize the code

The LaTeX compiler almost never cares about unnecessary whitespace in your code. Use generous amounts of whitespace to organize your code in a sensible way. Don't do the following:

Example 1.1: Code is too cramped

```
\[\begin{pmatrix*}[r]a^2+b^2&a^2-b^2\\-a^2+b^2&-a^2-b^2\end{pmatrix*}\]
```

Instead do the following:

Example 1.2: Code is less cramped

```
\[
  \begin{pmatrix*}[r]
     a^2 + b^2 &  a^2 - b^2 \\
    -a^2 + b^2 & -a^2 - b^2
  \end{pmatrix*}
\]
```

You could (and maybe should) even do the following:

Example 1.3: Code isn't cramped

```
\[
  \begin{pmatrix*}[r]
    a^2 + b^2
    &
    a^2 - b^2
    \\
    - a^2 + b^2
    &
    - a^2 - b^2
  \end{pmatrix*}
\]
```

This last version is easiest to write and easiest to navigate.

Also put at least one space between any two symbols that do not belong together. Don't do the following:

Example 1.4: Unrelated symbols need to be separated

```
$y=\sin(x)-e^x$
```

Instead do the following:

Example 1.5: Unrelated symbols are separated

```latex
$y = \sin(x) - e^x$
```

One could (and probably should) go even further and do the following:

Example 1.6: Unrelated symbols are separated even better

```latex
$y = \sin ( x ) - e^x$
```

1.3.2 Write one sentence per line

Related the previous point, write at most one sentence per line. This has at least three advantages:

- When an error or warning occurs, LaTeX will (try to) tell you in which line of source code the problem occured. Having your source code split up over multiple lines makes it much easier to find the problem.
- Organizing the source code into more lines will make the tools provided by your version control system more efficient to use.
- Sentences which are overly long become more easily recognizable.

1.3.3 Properly indent your source code

Writing LaTeX means writing source code. Make sure that your source code is properly indented.

1.3.4 Use `%` for problematic line breaks

Sometimes a sensible line break in your source code can lead to unwanted effects in the resulting output. Consider the following example:

Example 1.7: Wrong spacing before a footnote

```latex
Here is some text.
\footnote{Here is a footnote for
this text.}
```

Here is some text. [1]

[1]Here is a footnote for this text.

Note the unwanted space between the period and the superscript belonging to the footnote. This unwanted space comes from the line break that occurs between them in the source code. By ending the first line with `%` we can “comment out” this line break. We hence do the following:

Example 1.8: Right spacing before a footnote

```
Here is some text.%
\footnote{Here is a footnote for
this text.}
```

Here is some text.[1]

[1] Here is a footnote for this text.

1.4 Splitting up a project into multiple files

Nearly every project should be split up into multiple files.

1.4.1 The commands `\usepackage`, `\include`, `\input`

There are (at least) three useful ways to include another file into your project: `\usepackage`, `\include` and `\input`.

- Most of the preamble information – inclusion packages, configuration of styles, configuration of the look and feels of the document – should be put into one or multiple `sty`-files. These file(s) can then be included into the document via the command `\usepackage`.
- The commands `\include` and `\input` insert the text of the specified documents at the position where they are used, but have slight differences:
  - The command `\include` ensures that the specified content starts on a new page, and also that the following content begins on a new page. The command `\input` simply inserts the specified content at the given position without any such additional formatting.
  - The command `\include` cannot be nested: The specified file is not allowed to include the command `\include` again.

One should use the command `\include` for chapters and the command `\input` for any smaller level of text organization. One should in particular break the main text into smaller units until each of them can be put into its own file of sensible length.

1.4.2 An Example

Suppose that your text consists of two chapters, each of which consists of three sections. Then you should have at least ten files:

- A master file `main.tex`. This file includes all other files in some way, and this is the file which needs to be compiled.
- A file `mystyle.sty` in which packages are included and options are set.
- Two files for the chapters, say `chapter1.tex` and `chapter2.tex`.
- Six files for the sections, say `section1.tex` up to `section6.tex`.

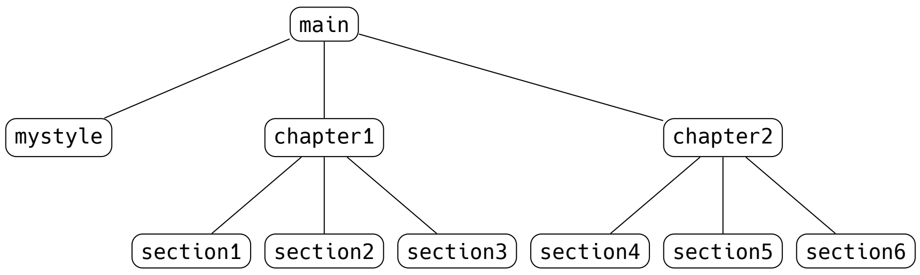


Figure 1.1: Splitting up a project into multiple files.

These files should include each other as shown in Figure 1.1.

Let us suppose for simplicity that all ten files are contained in the same directory. The master file `main.tex` should then look roughly as follows:

Example 1.9: Layout of `main.tex`

```latex
\documentclass[a4paper, 10pt]{scrreprt}

\usepackage{mystyle}

\title{A Report}
\author{John Doe}

\begin{document}

\maketitle

\include{chapter1}
\include{chapter2}

\end{document}
```

The file `mystyle.sty` may look as follows:

Example 1.10: Layout of `mystyle.sty`

```latex
%%%%% PACKAGES

% general mathematics
\usepackage{mathtools}
\usepackage{amssymb}

% commutative diagrams
```

```
\usepackage{tikz-cd}

%%%%% NEW COMMANDS

% new operators
\DeclareMathOperator{\End}{End}
\DeclareMathOperator{\Hom}{Hom}
```

The file `chapter1.tex` should look roughly as follows:

Example 1.11: Layout of `chapter1.tex`

```
\chapter{Name of the first chapter}

Some introduction to this chapter before the first section appears.

\input{section1}
\input{section2}
```

The file `section1.tex` should look roughly as follows:

Example 1.12: Layout of `section1.tex`

```
\section{Name of the first section}

Text of the first section.
```

1.4.3 Use `\includeonly`

As your document grows so does the time needed to compile it. It is therefore often desirable to compile only certain parts of the document. One naive solution to this problem is to simply not include the currently unwanted parts, e.g. by commenting out the corresponding lines of `\include` and `\input`. This approach has the problem that references to the now-excluded parts of the document will not compile properly. This problem can be fixed by using the command `\inludeonly`.

Usage of the command

Suppose that your main document, `main.tex`, includes two other files, say `chapter1.tex` and `chapter2.tex`, via the command `\include`. Hence the file `main.tex` should look roughly as follows:

Example 1.13: Basic example of `main.tex`

```latex
\documentclass[a4paper, 10pt]{scrreprt}

\begin{document}

\include{chapter1}
\include{chapter2}

\end{document}
```

By putting `\includeonly{chapter1}` in the preamble we tell LaTeX to only compile this specific chapter.

Example 1.14: Using the command `\includeonly`

```latex
\documentclass[a4paper, 10pt]{scrreprt}

\includeonly{chapter1}
\begin{document}

\include{chapter1}
\include{chapter2}

\end{document}
```

If in the above situation the file `chapter1.tex` contains some references to labels contained in the file `chapter2.tex` then these references will still compile correctly, even though only `chapter1.tex` is compiled.

Things to be aware of

The command `\includeonly` has two quirks, which follow from the way LaTeX handles referencing:

When LaTeX compiles the file `main.tex` an auxiliary file `main.aux` is created. This file contains various information about the compiled document. It does in particular contain a list of all labels found in `main.tex` during the compilation process. The next compilation process can then access these information to properly typeset all references that refer to these labels.

A feature of the command `\include` (which the command `\input` does not have) is that such an auxiliary file is also generated for the included file. So in the above example the files `chapter1.aux` and `chapter2.aux` will be created whenever corresponding `tex`-files are compiled. These files will in particular contain lists of all the labels found in `chapter1.tex` and `chapter2.tex`.

When `\includeonly{chapter1}` is used, only `chapter1.tex` is recompiled. As a consequence, the auxiliary file `chapter1.aux` may change, but the file `chapter2.aux` remains unchanged. This allows LaTeX to access the labels in `chapter2.tex` (which are now listed in `chapter2.aux`) without going through this file again. However, that the file `chapter2.aux` remains unchanged leads to two quirks that one should be aware of:

- Before inserting the code `\includeonly{chapter1}` one needs to compile the whole document, including `chapter2.tex`, a least once. This needs to be done so that LaTeX can properly create the auxiliary file `chapter2.aux`.
- Changes made to the file `chapter2.tex` will not be seen by LaTeX while the code `\includeonly{chapter1}` is present. This means in particular that LaTeX won't notice any new labels that are added in this file. In this case one need to stop excluding `chapter1.tex` for at least one compilation process to ensure that the auxiliary file `chapter2.aux` is updated.

Specifying multiple files

The command `\includeonly` actually accepts as its argument a list of files to be included:

Example 1.15: Syntax of `\includeonly`

```
\includeonly{file_1, ..., file_n}
```

1.5 Choosing the document class

1.5.1 Use the KOMA-Script classes

Instead of the standard classes `article`, `report` or `book`, the the KOMA-Script classes `scrartcl`, `scrreprt` and `scrbook` should be used. They provide more built-in functionalities than the standard classes and a more unified interface.

The KOMA-Script classes make slightly different design choices than the standard classes. Some of them are the following:

- The KOMA-Script classes have a different chapter title layout than the standard classes. The standard look can be regained by setting the option `chapterprefix` to `true`:

  Example 1.16: Using the option `chapterprefix`

  ```
  \documentclass[10pt, a4paper, chapterprefix = true]{scrbook}
  ```

- Headings and description labels are set in bold and sans-serif. This can be changed to a serif font with help of the command `\setkomafont`.

Example 1.17: Changing KOMA fonts

```latex
\addtokomafont{disposition}{\rmfamily}
\addtokomafont{descriptionlabel}{\rmfamily}
```

One can also use the undocumented option `egregdoesnotlikesansseriftitles`.

Example 1.18: Using the option `egregdoesnotlikesansseriftitles`

```latex
\documentclass[10pt, a4paper, egregdoesnotlikesansseriftitles]{scrbook}
```

1.5.2 Deciding on a document class

The three standard classes to consider for a mathematical project are `scrartcl`, `scrreprt` and `scrbook`. These three classes differ in what functionalities they provide and what their standard settings are.

The class `scrartcl` is the most basic one. It provides the standard sectioning commands `\section`, `\subsection` and `\subsubsection`. An abstract can be added with the `abstract` environment and by giving `\documentclass` the option `abstract = on`:

Example 1.19: Using the environment `abstract`

```latex
\documentclass[10pt, a4paper, abstract = on]{scrartcl}

\begin{document}

\begin{abstract}
  Our results are cool and our proofs have no details.
\end{abstract}

\end{document}
```

The class `scrreprt` extends the previous class: It provides the additional sectioning command `\chapter`, which is higher-level than `\section`. The title has a complete page for itself, although this behavior can be changed by setting the option `titlepage` to `false`:

Example 1.20: Using the option `titlepage`

```latex
\documentclass[10pt, a4paper, titlepage = false]{scrreprt}
```

This class also provides the sectioning command `\appendix` after which all chapters will be counted towards the appendix. The `abstract` environment is still available in the same way as before. (But the abstract is printed only on the second page, after the title page.)

The class scrbook provides the additional sectioning command \part that that is even more higher-level than \chapter. This class also provides the (very useful) additional sectioning commands \frontmatter, \mainmatter and \backmatter. The abstract environment isn't available anymore. This class does by default distinguish between left and right pages, i.e. even and odd pages. This can be disabled by using the option oneside:

Example 1.21: Using the option oneside

```
\documentclass[10pt, a4paper, oneside]{scrbook}
```

By default chapters will begin on the right page. This can be disabled with the option openany:

Example 1.22: Using the option openany

```
\documentclass[10pt, a4paper, openany]{scrbook}
```

When the option oneside then chapters are automatically allowed to start on both left pages and right pages. The option openany is then not needed.

In practice one should use the class scrartcl if only sections and subsections are needed, and otherwise the class scrbook. Instead of scrreprt one should directly use scrbook: The additional commands \frontmatter, \mainmatter and \backmatter that scrbook provides are extremely useful (see Section 1.6), and the visual difference between scrreprt and scrbook (which is their main difference) can easily be adjusted by using the aforementioned option openany.

1.6 Use \frontmatter and its friends

The class scrbook provides the commands \frontmatter, \mainmatter and\backmatter. Together with the sectioning command \appendix these can be used to separate the document into four parts: The front part, the main part, the appendix and the back part. These parts behave slightly differently:

- The front part uses lower case roman numerals for page numbers. In the generated pdf-file these pages will again be numbered with roman numerals (this will be important for the main part). Chapters in this part of the document will not be numbered, but will appear in the table of contents.

  This part of the document should include the title page, preface, table of content, list of figures, list of tables, and the introduction. In other words, everything that occurs before the first chapter.
- The main part uses Arabic numerals for page numbers and the page number is reset at the beginning of the main part. The first page of the main part is therefore numbered “1”. In the generated pdf-file this first page will also be numbered as such. This means that going to page 15 of the pdf-file will indeed

give page 15 of the document. The chapters in this part of the document are numbered and appear in the table of contents.

This part of the document does contain the vast amount of the document. It contains all the chapters excluding the ones belonging into the appendix.

- The appendix part continues the numbering of the main part. It both resets the chapter number and changes its style to upper case letters. The first chapter in this part will hence be numbered “A”. If the option `chapterprefix = true` is set then the word “Chapter” at the beginning of a new chapter will be replaced by “Appendix”.

  This part of the document should contain the appendices.
- The back part continues the numbering of the appendix (and thus the numbering of the main part). The chapters appearing in this part of the document are again unnumbered, and no “Chapter” is printed at the beginning of a new chapter.

  This part of the document should contain the bibliography, list of symbols, and the index. An entry for the bibliography can be added to the table of contents by setting the option `bibliography` to `totoc`.

The document layout with `scrbook` should therefore look roughly as follows:

Example 1.23: General document layout with `scrbook`

```latex
\documentclass[a4paper, 10pt, bibliography = totocnumbered]{scrbook}

% preamble

\begin{document}

\frontmatter
\maketitle
% dedication
% preface
\tableofcontents
% \listoftables
% \listoffigures
% introduction

\mainmatter
% most of the document

\appendix
% the appendices

\backmatter
\printbibliography
% list of symbols
% index
```

```
\end{document}
```

1.7 Be consistent

One of the most important aspects of both mathematical writing and the use of mathematical notation is consistency. If you're making crappy choices then at least do them consistently. Consider the following example:

Example 1.24: Inconsistency looks bad

```
If $X$ and $Y$ are two objects in a category $\mathcal{C}$ then it may happen
that the set $\operatorname{Hom}_{\mathcal{C}}(X,Y)$ is empty even though the
set $Hom_{\mathcal{C}}(Y,X)$ is non-empty.
```

If X and Y are two objects in a category $\mathcal{C}$ then it may happen that the set $\operatorname{Hom}_{\mathcal{C}}(X,Y)$ is empty even though the set $Hom_{\mathcal{C}}(Y,X)$ is non-empty.

The inconsistent use of Hom and Hom makes the already bad Hom even worse.

Chapter 2

Useful and important packages

2.1 `microtype` for better typesetting

Use the package `microtype`. It makes your document look nicer and helps you to circumvent overfull hboxes. Simply including the package is enough to let it work its magic

2.2 `mathtools` for mathematics

The package `amsmath` is a standard for mathematical typesetting. The package `mathtools` is an extension of `amsmath` that fixes some problems and also provides new (and often times very useful) functionalities. The package `mathtools` automatically loads `amsmath`, so instead of `amsmath` include `mathtools`.

Note that the important mathematical packages `amssymb` and `amsthm` are not automatically loaded by `mathtools` and therefore still need to be included by hand. Two other useful packages for mathematical content are `stmaryrd` (which provides some more symbols) and occasionally `extarrows` (which provides certain kinds of extensible arrows, see Table 4.4).

2.3 `amsthm` for theorem-like environments

To define theorem-like environments – like `lemma`, `proposition`, `theorem`, `corollary`, etc., include the package `amsthm`. New environments can then be defined with the command `\newtheorem`:

Example 2.1: Syntax of `\newtheorem`

```
\newtheorem{internal name of the environment}{name to be printed}
```

Example 2.2: Using the command `\newtheorem`

```
% in the preamble
\newtheorem{proposition}{Proposition}
```

```latex
% in the main body
\begin{proposition}
  Every finite subgroup of $k^\times$ is cyclic.
\end{proposition}
```

Proposition 1. *Every finite subgroup of $k^\times$ is cyclic.*

The variant `\newtheorem*` defines unnumbered theorem-like environments:

Example 2.3: Using the command `\newtheorem*`

```latex
% in the preamble
\newtheorem*{claim}{Claim}
% in the main body
\begin{claim}
  The symmetric group $S_3$ is the smallest non-abelian group.
\end{claim}
```

Claim. *The symmetric group S_3 is the smallest non-abelian group.*

If multiple theorem-like environments are defined then their have by default independent counters:

Example 2.4: Theorem-like environment use different counters by default

```latex
%in the preamble
\newtheorem{idea}{Idea}
\newtheorem{problem}{Problem}
%in the main text
\begin{idea}
  Fly to the moon in a car.
\end{idea}
\begin{problem}
  Cars don't fly.
\end{problem}
```

Idea 1. *Fly to the moon in a car.*

Problem 1. *Cars don't fly.*

For mathematical texts this behavior is pretty bad, as it makes it harder to find a specific result. (If page 492 features both Lemma 112 and Proposition 43 then where is Remark 20?)

To solve this problem we define a new counter `alltheorems` and tell all theorem-like environments to use this counter. To define the now counter we use the appropriately named command `\newcounter`:

Example 2.5: Syntax of `\newcounter`

```
\newcounter{name}[dependence]
```

The explicit code looks as follows

Example 2.6: Setting up a common counter

```
% in the preamble
\newcounter{alltheorems}

\newtheorem{assumption}[alltheorems]{Assumption}
\newtheorem{consequence}[alltheorems]{Consequence}

% in the main text
\begin{assumption}
  Cats hunt mice.
\end{assumption}

\begin{assumption}
  Tigers are cats.
\end{assumption}

\begin{consequence}
  Tigers hunt mice.
\end{consequence}
```

Assumption 1. *Cats hunt mice.*

Assumption 2. *Tigers are cats.*

Consequence 3. *Tigers hunt mice.*

In practice one often wants the counter to be bound to the surround section (or chapter). This can be achieved by binding the new counter to the section counter:

Example 2.7: Binding a new counter to the section level

```
% in the preamble
\newcounter{sometheorems}[section]
\renewcommand{\thesometheorems}{\thesection.\arabic{sometheorems}}
\newtheorem{corollary}[sometheorems]{Corollary}
```

```

% in the main text
\section{Free abelian groups}

\begin{corollary}
  Every subgroup of a free abelian group is again free abelian.
\end{corollary}

\begin{corollary}
  Every subgroup of $\mathbb{Z}^n$ admits a basis.
\end{corollary}

\section{More free abelian groups}

\begin{corollary}
  Every subgroup of a subgroup of $\mathbb{Z}^n$ admits a basis.
\end{corollary}
```

2.4 Free abelian groups

Corollary 2.4.1. *Every subgroup of a free abelian group is again free abelian.*

Corollary 2.4.2. *Every subgroup of $\mathbb{Z}^n$ admits a basis.*

2.5 More free abelian groups

Corollary 2.5.1. *Every subgroup of a subgroup of $\mathbb{Z}^n$ admits a basis.*

The addition of `[section]` to the definition of the new counter ensures that the resulting counter `sometheorems` resets every time the counter `section` is increased (which happens every time a new section begins). We also change the way the counter `sometheorems` is printed, namely printing it in the form "(section number).(counter number)" with both numbers printed in Arabic numerals.

2.4 `tikz-cd` for commutative diagrams

There are many packages for drawing commutative diagrams. Many of them have a rather restricted functionality, and quite a lot of them produce bad looking output. You should use the package `tikz-cd`. We refer to the very readable manual [CTN18] for an explanation of this package.

2.5 `cleveref` and `hyperref` for referencing

2.5.1 Recalling basic referencing

To refer to a numbered part of the document, like a theorem, an item of a list, a chapter or a section, one should never write down this number explicitly in the code. The referencing system of LaTeX should be used instead. Using this referencing system always consists of two steps: Setting a label at the position that you want to refer to, and then referencing this label at the desired position.

The basic `\label` and `\ref`

The command for setting a label is `\label`:

Example 2.8: Syntax of `\label`

```
\label{labelname}
```

There are multiple commands for referencing this label, the most basic of which is `\ref`:

Example 2.9: Syntax of `\ref`

```
\ref{labelname}
```

Referencing a label with `\ref` will print the number of whatever the specified label appears in.

Example 2.10: Using the commands `\label` and `\ref`

```
\begin{theorem}
  \label{vector spaces are free}
  Every vector space admits a basis.
\end{theorem}
It follows from \ref{vector spaces are free} that every $k$-vector space is
isomorphic to $k^{\oplus I}$ for some suitable index set $I$.
```

Theorem 1. *Every vector space admits a basis.*

It follows from 1 that every k-vector space is isomorphic to $k^{\oplus I}$ for some suitable index set I.

Instead of giving just a simple number it is customary to also specify what hides behinds this number, e.g. a theorem, table or figure. The specified name and number should then be separated by a tie ~ (as explained in Section 3.3.1) to ensure that no line break occurs at this position. The name of the referred to environment is usually capitalised.

Example 2.11: Specifying the kind of reference

```
\begin{theorem}
  \label{every vector space has a basis}
  Every vector space admits a basis.
\end{theorem}
It follows from Theorem~\ref{vector spaces are free} that every $k$-vector
space is isomorphic to $k^{\oplus I}$ for some suitable index set $I$.
```

Theorem 2. *Every vector space admits a basis.*

It follows from Theorem 1 that every k-vector space is isomorphic to $k^{\oplus I}$ for some suitable index set I.

In praxis this behavior can be automated by using the upcoming `cleveref` package.

`\eqref`

When referring to an equation it is customary to put the resulting number in parentheses. This is done via the command `\eqref`.

Example 2.12: Syntax of `\eqref`

```
\eqref{labelname}
```

The command `\eqref` is used in the same way as the original `\ref`:

Example 2.13: Using the command `\eqref`

```
A classic result in mathematics shows
\begin{equation}
  \label{important formula}
  1 + 1 = 2 \,.
\end{equation}
Note that the identity~\eqref{important formula} shows that ’Fermats conjecture
on the sum $a^n + b^n = c^n$ cannot be generalized to the case $n = 1$.
```

A classic result in mathematics shows

$$1 + 1 = 2\,. \tag{2.1}$$

Note that the identity (2.1) shows that Fermat’s conjecture on the sum $a^n + b^n = c^n$ cannot be generalized to the case $n = 1$.

Choosing labels

Always use descriptive labels. Cryptic sequences of seemingly random letters will backfire on you down the road.

2.5.2 `cleveref`

There are two related problems with hardcoding the type of a reference as done in Example 11: One has to remember or to look up what type of environment the label refers to, and if this type is changed (e.g. by promoting a proposition to a theorem) then the hardcoded types need to be manually adjusted.

These problems can be circumvented by using the package `cleveref`. The author recommends to load this package with the options `capitalise` and `noabbrev`:

Example 2.14: Loading the package `cleveref`

```
\usepackage[capitalise, noabbrev]{cleveref}
```

The command `\cref`

The package `cleveref` provides the command `\cref`

Example 2.15: Syntax of `\cref`

```
\cref{labelname}
```

The command `\cref` differs from the more primitive `\ref` in that it automatically inserts the right kind of type before the reference.

Example 2.16: Using the command `\cref`

```
\begin{lemma}
  \label{dim is well-defined}
  Every two bases of a vector space have the same cardinality.
\end{lemma}

\begin{remark}
  One can generalize \cref{dim is well-defined} to non-commutative rings:
  If $R$ is some ring and $M$ is a semisimple $R$-module then for every
irreducible $R$-module $L$ the multiplicity of $L$ in $M$ is well-defined.
\end{remark}
```

Lemma 3. *Every two bases of a vector space have the same cardinality.*

Remark 4. *One can generalize Lemma 3 to non-commutative rings: If R is some ring and M is a semisimple R-module then for every irreducible R-module*

	lower case	**upper case**	**default case**
singular	`\lcnamecref`	`\nameCref`	`\namecref`
plural	`\lcnamecrefs`	`\nameCrefs`	`\namecrefs`

Table 2.1: The commands `\*name*ref`.

L the multiplicity of L in M is well-defined.

The used options options `capitalise` and `noabbrev` have the following effects:

- The option `capitalise` ensures that the printed type of the reference will begin with an upper case letter. In Example 14 we would otherwise get “lemma 3” instead of “Lemma 3”.
- The option `noabbrev` ensures that printed types won’t be abbreviated. One wil otherwise get “eq. (5)” instead of “equation (5)”.

The command `\cref` has the variant `\Cref` which ensures that the inserted type will start with an upper case letter. One should always use `\Cref` instead of `\cref` at the beginning of a sentence, even if the option `capitalise` is set.

The commands `\*name*ref`

The package `cleveref` provides another family of useful commands aside from `\cref` and `\Cref`. An overview of these commands can be found in Table 2.1. They can be used to print the type of a reference without its number. This can be used be used to circumvent hardcoding types into the source code:

Example 2.17: Using `\lcnamecref`

```
\begin{lemma}
  \label{weak cayley}
  Every group embeds into a non-abelian group.
\end{lemma}
The above \lcnamecref{weak cayley} does directly follow of 'Cayleys~theorem.
```

Lemma 5. *Every group embeds into a non-abelian group.*

The above lemma does directly follow of Cayley’s theorem.

By using the various referencing commands introduced so far it is now possible to avoid (nearly?) every kind of hardcorded type.

2.5.3 hyperref

If the resulting `pdf`-file is supposed to be navigated digitally then the package `hyperref` should be used. This package puts hyperlinks in the `pdf`-file whenever some kind of reference is used.

2.5.4 Order of inclusion

The packages `cleveref` and `hyperref` are a bit peculiar when it comes to where they have to be included in the preamble. The general rule is that the package `hyperref` should be the very last package to be included, to ensure that it interacts properly with all other used packages. There are some rare exceptions to this rule, one of which happened to be `cleveref`.

If you're defining some common counter for your theorem-like environments (which you should do, as explained in Section 2.3) then this needs to be done after `cleverref` was included. Otherwise `cleveref` will have problems knowing what names to print when the command `\cref` is used.

Overall your preamble should have the following order of inclusions:

Example 2.18: Order of preamble with `cleveref` and `hyperref`

```latex
% most packages
...
\usepackage{amsthm}
...

% the last packages
\usepackage{hyperref}
\usepackage{cleveref}

% defining theorem-like environments
\newcounter{everything}
\newtheorem{theorem}[everything]{Theorem}
```

2.6 csquotes for quotation marks

Dealing with quotations marks in LaTeX by hand can be a pain in the ass, for at least two reasons: Different languages use different kinds of quotation marks, and finding the right combination of LaTeX code to get the correct ones can be a non-trivial problem. (Depending on your keyboard layout it may even be non-trival in non-trivial ways.) One way to circumvent this problem is by using the package `csquotes`. This package provides the command `\enquote`:

Example 2.19: Syntax of `\enquote`

```
\enquote{text}
```

This commands inserts quotation marks around the specified text:

Example 2.20: Using the command `\enquote`

```
\enquote{This is a quote.}
```

“This is a quote.”

The command `\enquote` can automatically adjust the quotation marks to the conventions of the used language when this language is specified through the package `babel`. This is done by loading the package `csquotes` with the option `babel` set to `true`.

Example 2.21: `\enquote` chooses the right kind of quotation marks

```
\begin{tabular}{@{}ll@{}}
  \toprule
  American English
  &
  \selectlanguage{american}
  \enquote{quote}
  \\
  British English
  &
  \selectlanguage{british}
  \enquote{quotation}
  \\
  German
  &
  \selectlanguage{ngerman}
  \enquote{Zitat}
  \\
  French
  &
  \selectlanguage{french}
  \enquote{citation}
  \\
  \bottomrule
\end{tabular}
```

American English	“quote”
British English	‘quotation’
German	„Zitat“
French	« citation »

The command `\enquote` automatically handles nested quotation marks:

Example 2.22: Nested quotes with `\enquote`

```
\enquote{This is a \enquote{quote}
inside a quote.}
```

“This is a ‘quote’ inside a quote.”

So for dealing with quotes of any kind use the package `csquotes`.

2.7 `enumitem` for configuration of lists

In LaTeX there are three different kinds of list environments: Numbered lists are provided by the environment `enumerate`:

Example 2.23: Using the environment `enumerate`

```
\begin{enumerate}
  \item
    Assumption
  \item
    ???
  \item
    Contradiction
\end{enumerate}
```

1. Assumption
2. ???
3. Contradiction

Unnumbered lists are provided by the environment `itemize`:

Example 2.24: Using the environment `itemize`

```
\begin{itemize}
  \item
    This is a list item.
  \item
    This is also a list item.
  \item
    And yet another list item.
\end{itemize}
```

- This is a list item.
- This is also a list item.
- And yet another list item.

The environment `description` uses no predefined symbols for the list items and instead expects a descriptive text from the user:

option	description
`\alph*`	lower case alphabetic
`\Alph*`	upper case alphabetic
`\roman*`	lower case Roman numerals
`\Roman*`	upper case Roman numerals
`\arabic*`	Arabic numerals

Table 2.2: Possible labels for the environment `enumerate`.

Example 2.25: Using the environment `description`

```latex
\begin{description}
  \item[Field]
    A special kind of ring.
  \item[Ring]
    A generalization of fields.
\end{description}
```

Field A special kind of ring.

Ring A generalization of fields.

The package `enumitem` is immensely useful for the configuration of the style and behavior of these list environments. It provides (among others) the following features:

2.7.1 Style of numbering

For the environment `enumerate` the style of the numbering can be changed by using the option `label`

Example 2.26: Changing the numbering style of `enumerate` lists

```latex
\begin{enumerate}[label = (\alph*)]
  \item
    First entry.
  \item
    Second entry.
  \item
    Third entry.
\end{enumerate}
```

(a) First entry.

(b) Second entry.

(c) Third entry.

For a list of possible labels see Table 2.2. One can similarly change the symbol for `itemize` lists via the option `label`:

Example 2.27: Changing the symbol for `itemize` lists

```latex
With the standard symbol:
\begin{itemize}[label =
{\textbullet}]
  \item
    First entry.
  \item
    Second entry.
\end{itemize}
Now with a different symbol:
\begin{itemize}[label =
{\textopenbullet}]
  \item
    First entry again.
  \item
    Second entry again.
\end{itemize}
```

With the standard symbol:

- First entry.
- Second entry.

Now with a different symbol:

◦ First entry again.

◦ Second entry again.

2.7.2 Resuming lists

One can resume lists with the option `resume*`. The option `resume` works similarly to `resume` but doesn't copy any style options from the previous list.

Example 2.28: Resuming lists with `resume*`

```latex
Some text before the first \texttt{enumerate} environment.
\begin{enumerate}[label = \roman*.]
  \item
    First entry.
  \item
    Second entry.
\end{enumerate}
Some text between the \texttt{enumerate} environment.
\begin{enumerate}[resume*]
  \item
    Third entry.
  \item
    Fourth entry.
\end{enumerate}
Some text after the second \texttt{enumerate} environment.
\begin{enumerate}[resume]
  \item
    Fifth entry.
  \item
    Sixth entry.
```

```latex
\end{enumerate}
Some text after the third \texttt{enumerate} environmen.
```

Some text before the first `enumerate` environment.

i. First entry.

ii. Second entry.

Some text between the `enumerate` environment.

iii. Third entry.

iv. Fourth entry.

Some text after the second `enumerate` environment.

5. Fifth entry.

6. Sixth entry.

Some text after the third `enumerate` environmen.

2.7.3 Spacing

One can change the various spacings involved in the list environments. For a full overview we refer to [CTN19a], and in particular to [CTN19a, Figure 1]. We give here only some of the more important configurations.

Multiple paragraphs in a list item

If a list item contains multiple paragraphs of text then the new paragraphs are by default not indented but instead separated from the preceding paragraph by some additional vertical space:

Example 2.29: Indentation and distance of pararaphs in lists, default behavior

```latex
\begin{enumerate}
  \item
    Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna.

    In massa tempor nec feugiat. Nunc aliquet bibendum enim facilisis gravida.
\end{enumerate}
```

1. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna.

   In massa tempor nec feugiat. Nunc aliquet bibendum enim facilisis gravida.

One can adjust the indentation of the paragraphs with the option `listparindent` and the additional vertical spacing between paragraphs with the option `parsep`. With this we can regain the standard indentation pattern in lists:

Example 2.30: Indentation and distance of pararaphs in lists, changed behavior

```latex
\begin{enumerate}[listparindent = \parindent, parsep = 0pt]
  \item
    Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna.

    In massa tempor nec feugiat. Nunc aliquet bibendum enim facilisis gravida.
\end{enumerate}
```

1. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna.
   In massa tempor nec feugiat. Nunc aliquet bibendum enim facilisis gravida.

Distance between items

One can control the distance between list items with the option `itemsep`:

Example 2.31: Distance between list items with `itemsep`

```latex
A list with standard item distance:
\begin{enumerate}
  \item
    First entry.
  \item
    Second entry.
\end{enumerate}
```

```latex
Now a list without additional item distance:
\begin{enumerate}[itemsep = 0pt]
  \item
    First entry again.
  \item
    Second entry again.
\end{enumerate}
```

A list with standard item distance:

1. First entry.

2. Second entry.

Now a list without additional item distance:

1. First entry again.
2. Second entry again.

Indentation

The environment `enumerate` has by default two kinds of indentation: An indentation of the labels with respect to the left margin, and an indentation of the list items with respect to the label.

Example 2.32: `enumerate` environment with standard indentation

```latex
Lorem ipsum dolor sit amet,
\begin{enumerate}
  \item
  consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
\end{enumerate}
```

Lorem ipsum dolor sit amet,

1. consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

We can remove both indentations by using the option `wide` with the value `0pt`:

Example 2.33: `enumerate` environment without indentation

```
Lorem ipsum dolor sit amet,
\begin{enumerate}[wide = 0pt]
  \item
  consectetur adipiscing elit, sed
do eiusmod tempor incididunt ut
labore et dolore magna aliqua.
\end{enumerate}
```

Lorem ipsum dolor sit amet,

1. consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

We can add back the identation between the item labels and the item text by also setting the option `leftmargin` to the value `*`:

Example 2.34: `enumerate` environments with only identitation between label and text

```
Lorem ipsum dolor sit amet,
\begin{enumerate}[wide = 0pt,
leftmargin = *]
  \item
  consectetur adipiscing elit, sed
do eiusmod tempor incididunt ut
labore et dolore magna aliqua.
\end{enumerate}
```

Lorem ipsum dolor sit amet,

1. consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

One can also add back the indentation between the left margin and the item labels by also using the option `labelindent`:

Example 2.35: `enumerate` environment with indentation between margin and label

```
Lorem ipsum dolor sit amet,
\begin{enumerate}[wide = 0pt,
leftmargin = \parindent,
labelindent = \parindent]
  \item
  consectetur adipiscing elit, sed
do eiusmod tempor incididunt ut
labore et dolore magna aliqua.
\end{enumerate}
```

Lorem ipsum dolor sit amet,

1. consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

One can also combine the above two indentations to regain something that looks like the standard indentation:

Example 2.36: `enumerate` environment with indentation between margin and label, and label and text

```
Lorem ipsum dolor sit amet,
\begin{enumerate}[wide = 0pt,
leftmargin = *, labelindent =
\parindent]
  \item
  consectetur adipiscing elit, sed
do eiusmod tempor incididunt ut
labore et dolore magna aliqua.
\end{enumerate}
```

Lorem ipsum dolor sit amet,

1. consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

But Example 36 does actually have an advantage over the standard indentation, namely that it properly handels long labels:

Example 2.37: Proper vs. improper placement of long labels

```
The standard identation:
\begin{enumerate}[label = {(Latin \arabic*)}]
  \item
  Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod
tempor incididunt ut labore et dolore magna aliqua.
\end{enumerate}
An indentation with adjusted parameters:
\begin{enumerate}[label = {(Latin \arabic*)}, wide = 0pt, leftmargin = *,
labelindent = \parindent]
  \item
  Dolor sed viverra ipsum nunc aliquet bibendum enim.
  In massa tempor nec feugiat.
\end{enumerate}
```

The standard identation:

(Latin 1) Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

An indentation with adjusted parameters:

(Latin 1) Dolor sed viverra ipsum nunc aliquet bibendum enim. In massa tempor nec feugiat.

The indentitation for the environment `itemize` can be adjusted in the same way as for the environment `description`.

For the environment `description` we can proceed similarly as for `enumerate`. We first observe the standard identation for this environment: There is a fixed identation between the left margin and the items texts. The item labels are aligned at the left margin.

Example 2.38: `description` environment with standard indentation

```
Text, text, text, text, text, text, text.
\begin{description}
  \item[First item]
    Text, text, text, text, text, text, text, text, text, text, text.
  \item[Second item]
     Text, text, text, text, text, text, text, text, text, text, text.
\end{description}
```

Text, text, text, text, text, text, text.

First item Text, text, text, text, text, text, text, text, text, text, text.

Second item Text, text, text, text, text, text, text, text, text, text, text.

We can remove the identation by again using the option `wide`.

Example 2.39: `description` environment with no indentation

```
Text, text, text, text, text, text, text.
\begin{description}[wide = 0pt]
  \item[First item]
    Text, text, text, text, text, text, text, text, text, text.
  \item[Second item]
     Text, text, text, text, text, text, text, text, text, text.
\end{description}
```

Text, text, text, text, text, text, text.

First item Text, text, text, text, text, text, text, text, text, text.

Second item Text, text, text, text, text, text, text, text, text, text.

2.7.4 Global settings

Global settings can be set via the command `\setlist`:

Example 2.40: Syntax of `\setlist`

```
\setlist[kind of list]{options}
```

Consider the following example:

Example 2.41: Global settings for list environments

```latex
\setlist[enumerate]{label =
\roman*)}
\begin{enumerate}
  \item
    First entry.
  \item
    Second entry.
\end{enumerate}
```

i) First entry.

ii) Second entry.

2.7.5 Nested lists

When lists are nested one can use different settings for each list.

Example 2.42: Different settings for nested lists

```latex
\begin{enumerate}[label = \Roman*)]
  \item
    First entry.
    \begin{enumerate}[label = \alph*)]
      \item
        First entry, first subentry.
      \item
        First entry, second subentry.
    \end{enumerate}
  \item
    Second entry.
    \begin{enumerate}[label = \arabic*)]
      \item
        Second entry, first subentry.
      \item
        Second entry, second subentry.
    \end{enumerate}
\end{enumerate}
```

I) First entry.

   a) First entry, first subentry.

   b) First entry, second subentry.

II) Second entry.

   1) Second entry, first subentry.

   2) Second entry, second subentry.

One can also use different global settings for different depths:

Example 2.43: Global settings depending on level

```latex
\setlist[enumerate, 1]{label = (\roman*)}
\setlist[enumerate, 2]{label = (\alph*)}
\begin{enumerate}
  \item
    First entry.
    \begin{enumerate}
      \item
        First entry, first subentry.
      \item
        First entry, second subentry.
    \end{enumerate}
  \item
    Second entry.
    \begin{enumerate}
      \item
        Second entry, first subentry.
      \item
        Second entry, second subentry.
    \end{enumerate}
\end{enumerate}
```

(i) First entry.
  (a) First entry, first subentry.
  (b) First entry, second subentry.

(ii) Second entry.
  (a) Second entry, first subentry.
  (b) Second entry, second subentry.

The counter of the first depth and second depth can be accessed via the counters `enumi` and `enumii`:

Example 2.44: Accessing level counters in settings for list environments

```latex
\setlist[enumerate, 1]{label = (\arabic*)}
\setlist[enumerate, 2]{label = (\arabic{enumi}.\alph*)}
\begin{enumerate}
  \item
    An entry.
    \begin{enumerate}
      \item
      Again an entry.
    \item
      Again an entry.
  \end{enumerate}
 \item
  Another entry.
  \begin{enumerate}
    \item
      Yet another entry.
    \item
      Yet another entry.
  \end{enumerate}
\end{enumerate}
```

(1) An entry.
 (1.a) Again an entry.
 (1.b) Again an entry.

(2) Another entry.
 (2.a) Yet another entry.
 (2.b) Yet another entry.

2.7.6 Cloning lists

New list types can be constructed by cloning an already existing one via the command `\newlist`:

Example 2.45: Syntax of `\newlist`

```
\newlist{new list}{original list}{depth of new list}
```

One can then set global options for this new list type without changing the global options for the original type. This can be used to construct list environments that serve special purposes. In the following example we create a new list environment that is meant for listing equivalent statements:

Example 2.46: Custom clones of list environments

```
% clone enumerate as equivalenceslist, allowing up to 2 levels
\newlist{equivalenceslist}{enumerate}{2}
% set the formatting
\setlist[equivalenceslist,1]{label = (\roman*)}
```

```latex
\setlist[equivalenceslist,2]{label = (\alph*), leftmargin = *}
% an example
Let $M$ be an $R$-module.
For every collection of elements $x_1, \dotsc, x_n \in M$ the following
conditions are equivalent:
\begin{equivalenceslist}
  \item
    For every $R$-module $N$ and every choice of elements $y_1, \dotsc, y_n \in
N$ there exists a unique module homomorphism $f \colon M \to N$ with $f(x_i) =
y_i$ for every $i = 1, \dotsc, n$.
  \item
    The elements $x_1, \dotsc, x_n$ are a basis of $M$, i.e.
    \begin{equivalenceslist}
      \item
        the elements $x_1, \dotsc, x_n$ are linearly independent, and
      \item
        the elements $x_1, \dotsc, x_n$ are a generating set of $M$.
    \end{equivalenceslist}
\end{equivalenceslist}
```

Let M be an R-module. For every collection of elements $x_1, \dots, x_n \in M$ the following conditions are equivalent:

(i) For every R-module N and every choice of elements $y_1, \dots, y_n \in N$ there exists a unique module homomorphism $f\colon M \to N$ with $f(x_i) = y_i$ for every $i = 1, \dots, n$.

(ii) The elements $x_1, \dots, x_n$ are a basis of M, i.e.

  (a) the elements $x_1, \dots, x_n$ are linearly independent, and

  (b) the elements $x_1, \dots, x_n$ are a generating set of M.

2.8 biblatex for bibliography

2.8.1 The basic setup

A bibliography in LaTeX comes about from the interplay of three different actors:

- A `bib`-file that contains the metadata about the literature which should be cited.
- A bibliography package that provides commands for citing these references.
- A back end program that accesses the `bib`-file to extract the needed information and pass them to LaTeX.

One should choose `biblatex` for the bibliography package and biber for the back end program. For this the package `biblatex` needs to be loaded with the option `backend` set to `biber`:

Example 2.47: Loading the package `biblatex`

```
\usepackage[backend = biber]{biblatex}
```

The author also likes to use the following options:

- By default the occurring references will simply be numbered as [1], [2], [3], etc. Often references of the form [Eis04] are preferable, which is achieved by setting the option `style` to `alphabetic`.
- Setting the option `dateabbrev` to `false` ensures that month names like "September" are not abbreviated as "Sept."
- Setting the option `urldate` to `long` ensure that dates concerning URLs are written out as "September 4, 2109" instead of "09/04/2019".

2.8.2 Creating the `bib`-file

To most important step of creating a bibliography is to collect the references and their various metadata in a `bib`-file. For every reference we need to add an entry to this `bib`-file. These entries have the following form:

Example 2.48: Syntax for an entry of the `bib`-file

```
@type{label,
  key1 = {value1},
  key2 = {value2},
  key3 = {value3},
  ...
}
```

Instead of curly braces `{ }` one can also use quotation marks `" "` on the right hand side of the equality signs.

The specifier `@type` in Example 48 will be be replaced by something like `@book` or `@article` to explain what kind of work this entry is. A list of all possible types can be found in [CTN19c, 2.1]. This specified type will determine which of the given data will be printed in the bibliography and how these printed date are formatted.

The given text `label` has no influence on the bibliography itself. It will be used to add the citations to this reference in the main text.

How to choose data for the bibliography

One should follow two guidelines when adding information to the bibliography.

- Provide as much data as possible. The specified type will determine which of these data will be printed. To find out which type will use which information we refer again to [CTN19c, 2.1, 2.2].

- How the printed data are to be formatted is for LaTeX – and more specifically `biblatex` – to decide. So don't try to preformat the provided date in the `bib`-file. Try in particular to give full, unabbreviated names whenever possible.

If you feels strongly about certain data being printed, or how certain data should be formatted when printed out, then you should not try to abuse the `bib`-file for this. Instead complains to `biblatex` by changing the appropriate settings.

Some good sources for finding the data that a bibliography requires are MathSciNet and the websites of the publishers. (Springer is quite good at providing all the needed information on the websites of their books.) Most of the needed data can also be found in the cited resource – e.g. book or article – itself.

In the following we will look at some specific examples of `bib`-file entries.

Entry for a single book

The `bib`-file entry for a single book should roughly look as follows:

Example 2.49: `bib`-file entry for a single book

```
@book{fultonharris2004,
  title     = {Representation Theory},
  subtitle  = {A First Course},
  author    = {Fulton, William and Harris, Joe},
  edition   = {1},
  year      = {2004},
  pagetotal = {xv+551},
  publisher = {Springer-Verlag New York},
  series    = {Graduate Texts in Mathematics},
  number    = {129},
  isbn      = {978-0-387-97527-6},
  doi       = {10.1007/978-1-4612-0979-9}
}
```

The resulting output in the bibliography (see Example 58) will look as follows:

> William Fulton and Joe Harris. *Representation Theory. A First Course.* 1st ed. Graduate Texts in Mathematics 129. Springer-Verlag New York, 2004. xv+551 pp. ISBN: 978-0-387-97527-6. DOI: `10.1007/978-1-4612-0979-9`

The type `@books` tells LaTeX that the entry is a (single) book. The various keys have the following functions:

`title` This key specifies the title of the book. The expected value of this key is a text.

`subtitle` This key specifies the subtitle of the book. The expected value of this key is a text.

`author` This key specifies the author(s) of the book. There are some things to be aware of here:

- The name of an author needs to be given in the format `Lastname, Firstname`. This is required so that biber can properly process this data.
- If multiple authors are given then they need to be separated by the word `and`.

`edition` This key specifies the edition of the book. The value should be given as a number for proper processing, but can in an emergency also be given as a text.

`year` This key specifies the year the book was published. One can also specify a moth with the key `month`. The values for both keys are expected to be numbers.

One can also use the key `date` to specify the date of publication. This key expects a single argument, which is of one of the forms `year`, `year-month`, or `year-month-day`. Here the value of `year` is expected to be a four digit number, and the values of `month` and `day` are expected to be two digit numbers (which may contain a leading zero).

`pagetotal` This key specifies the total number of pages of the book. The value should be an integer but can also be an arbitrary text.

There is however a drawback to simply providing a text: Normally the number of pages is followed by the text "pp." or "p.", depending on whether the reference consists of only a single page. To do so `biblatex` always tried to interpret the input as a number. But if this interpretation fails then neither "pp." nor "p." will be added.

Books often start with pages that are numbered with Roman numbers, followed by pages that are numbered by Arabic numbers. In this case the total number of pages should be given in the form "(Roman number)+(Arabic number)".

As explained above, this will lead to the problem of `biblatex` being unable to interpret the input as a number, which leads by default to a missing "pp." in the output. For this problem one can adjust the settings of `biblatex` to *always* include "pp." after the total page number of a book. This can be done as follows:

Example 2.50: Adjusting the formatting of `pagestotal`

```
\DeclareFieldFormat[book]{pagetotal}{#1~\ppno}
```

`publisher` This key specifies the publisher of the book. The expected value is a text. Note that "Springer" is not a proper reference for a publisher.

`series` Many mathematical books are part of some series, e.g. "Graduate Texts is Mathematics" or "Cambridge Studies in Advanced Mathematics". Such a series can be specified with the key `series`, which expects as its value a text.

`number` This key specifies the number of the book in the previously specified series. The values of this key is (counterintuitively) treated as a text.

If you copy your bibliography data from somewhere else then there is a high chance that instead of the key `number` the key `volume` is used. This is relict from the past that isn't correct with `biblatex`.

`isbn` This key specifies the ISBN number of the book. The value of this field is treated as a text.

`doi` This key specifies the DOI of the book (if it has one).

Entry for a book with multiple volumes

Sometimes a book is just one volume in a small collection of books. In this case one should use the type `@mvbook` to define the overall information of these books, and then an entry of type `@book` which is subordinate to the previously created entry. Let's consider an example:

Example 2.51: `bib`-file entry for a book with multiple volumes

```
@mvbook{benson,
  title     = {Representations and Cohomology},
  author    = {Benson, David J.},
  publisher = {Cambridge University Press},
  series    = {Cambridge Studies in Advanced Mathematics},
  volumes   = {2}
}

@book{benson1991,
  crossref  = {benson},
  volume    = {1},
  number    = {30},
  title     = {Basic Representation Theory of Finite Groups and Associative
Algebras},
  edition   = {1},
  year      = {1991},
  pagetotal = {xii+246},
  isbn      = {978-0-521-36134-7},
  doi       = {10.1017/CBO9780511623615}
}
```

We can then refer to the entry of type `@book` as usual, to get the following output in the bibliography (see Example 58):

> David J. Benson. *Representations and Cohomology.* Vol. 1: *Basic Representation Theory of Finite Groups and Associative Algebras.* 2 vols. Cambridge Studies in Advanced Mathematics 30. Cambridge University Press, 1991. xii+246 pp. ISBN: 978-0-521-36134-7. DOI: `10.1017/CBO9780511623615`

One can also refer to the overall collection:

> David J. Benson. *Representations and Cohomology.* 2 vols. Cambridge Studies in Advanced Mathematics. Cambridge University Press

There are three new keys to talk about here:

`volumes` This key describes the total number of volumes. The value of this key is expected to be an integer.

`volume` This key describes the specific volume of the bibliography entry. The value of this key is expected to be an integer.

`crossref` This key expresses that the given entry is subordinate to some other entry. The expected value is the label of the superior entry.

Entry for an article

We now consider an example for citing an article:

Example 2.52: `bib`-file entry for a single book

```
@article {diamond_lemma,
  title        = {The Diamond Lemma for Ring Theory},
  author       = {Bergman, George Mark},
  year         = {1978},
  month        = {2},
  journaltitle = {Advances in Mathematics},
  issn         = {0001-8708},
  volume       = {29},
  number       = {2},
  pages        = {178--218},
  doi          = {10.1016/0001-8708(78)90010-5}
}
```

The output in the bibliography (see Example 58) will look as follows:

> George Mark Bergman. “The Diamond Lemma for Ring Theory.” In: *Advances in Mathematics* 29.2 (February 1978), pp. 178–218. DOI: `10.1016/0001-8708(78)90010-5`

The type `@article` tells LaTeX that the entry is for a (journal) article. Many fields are as for the type `@book`, so we will focus on the changes:

`journaltitle` This key specifies the name of the journal that the article was published in. The expected value for this key is a text.

`issn` This key specifies the ISSN (International Standard Serial Number) of the journal in question. The value is treated as a text.

`volume`, `number` These keys specify in which volume of the journal the article appeared, and in which number of the volume. The value for `volume` should be an integer, and the value for `number` should be an integer too (although it is treated as text).

`pages` This key specifies in which page range the article appeared. It doesn’t matter how many dashes are used to separate the two page numbers. It also doesn’t matter if the dash(es) are surrounded by space.

It is customary to specify page ranges in the form `pages = number--number` because this gives a right looking output even if this argument were simply to be interpreted as text. (Which lesser packages than `biblatex` may do.)

Entry for an online resource

We consider now an example where we cite an online resource. For this we use the generic type `@online`.

Example 2.53: `bib`-file entry for an online resource

```
@online{cayley_graph,
  title   = {Cayley graphs and the geometry of groups},
  author  = {Tao, Terence},
  date    = {2010-06-10},
  url     =
{https://terrytao.wordpress.com/cayley-graphs-and-the-geometry-of-groups},
  urldate = {2019-09-06}
}
```

The output in the bibliography (see Example 58) will look as follows:

> Terence Tao. *Cayley graphs and the geometry of groups.* July 10, 2010. URL: `https://terrytao.wordpress.com/cayley-graphs-and-the-geometry-of-groups` (visited on September 6, 2019)

We have three (or rather two) new fields to discuss:

`date` This key specifies when the linked-to resource was created. This key takes arguments of the form `year`, `year-month` or `year-month-day`. Here the value of `year` is expected to be a four digit number, and the values of `month` and `day` are expected to be two digit numbers (which may contain a leading zero).
Instead of `date` one can also use the overall less specific keys `year` and `month`.

`url` This key specifies the URL of the online resource.

`urldate` This key specifies the date on which the online resource was accessed. This is an important information since content online may change over time.

Entry for an arXiv article

To cite an article published on arXiv one shouldn't use the key `url` because `biblatex` has a built-in way of dealing with arXiv articles.

Example 2.54: `bib`-file entry for an online resource

```
@online{leinster2014,
  title       = {The bijection between projective indecomposable and simple
modules},
  author      = {Leinster, Tom},
  date        = {2014-10-14},
  eprint      = {1410.3671v1},
  eprinttype  = {arxiv},
  eprintclass = {math.RA},
  urldate     = {2019-09-12}
}
```

The output in the bibliography (see Example 58) will look as follows:

> Tom Leinster. *The bijection between projective indecomposable and simple modules.* October 14, 2014. arXiv: `1410.3671v1 [math.RA]`. (Visited on September 12, 2019)

We have used three new fields:

`eprinttype` This key specifies what kind of resource the entry is. The used value `arxiv` results in a predefined formatting of the resulting output.

`eprint` This key specifies the identifier of the resource.

`eprintclass` This field specifies additional information about the resource.

2.8.3 Citing the references

Suppose now that we have added an entry to our `bib`-file, as outlines in Example 48. In the actual LaTeX project we can then refer to this entry with the command `\cite`:

Example 2.55: Syntax of `\cite`

```
\cite[details]{label}
```

Let's consider an example where we cite the references given in the previous examples:

Example 2.56: Using the command `\cite`

```
We assume that the reader is familiar with the representation theory of the
symmetric groups as discussed in \cite[\S 4]{fultonharris2004}.
The reader may also want to check out \cite{benson1991} and \cite{cayley_graph}.
For a nice proof of the Poincaré--Birkhoff--Witt~Theorem we refer to \cite[\S
3]{diamond_lemma}.
```

We assume that the reader is familiar with the representation theory of the symmetric groups as discussed in [FH04, §4]. The reader may also want to check out [Ben91] and [Tao10]. For a nice proof of the Poincaré–Birkhoff–Witt Theorem we refer to [Ber78, §3].

We will also need to add the bibliography into the LaTeX document. Suppose for this that the `bib`-file is called `references.bib`. We then have to do three things:

- We need to tell LaTeX how the `bib`-file is called. This is done via the command `\bibliography`:

Example 2.57: Syntax of `\bibliography`

```
\bibliography{name of bib-file}
```

In our case we have to add `\bibliography{references.bib}`.

- We have to add the command `\printbibliography` at the position in source code where we want the bibliography to be printed. This typically happens near the end of the document.
- We set the option `bibliography` of `\documentclass` to `totoc` to add the bibliography to the table of contents. (see Example 23).

In our running example(s) we would get the following bibliography:

Example 2.58: Using the command `\printbibliography`

```
\printbibliography
```

Bibliography

[Ben] David J. Benson. *Representations and Cohomology.* 2 vols. Cambridge Studies in Advanced Mathematics. Cambridge University Press.

[Ben91] David J. Benson. *Representations and Cohomology.* Vol. 1: *Basic Representation Theory of Finite Groups and Associative Algebras.* 2 vols. Cambridge Studies in Advanced Mathematics 30. Cambridge University Press, 1991. xii+246 pp. ISBN: 978-0-521-36134-7. DOI: `10.1017/CBO9780511623615`.

[Ber78] George Mark Bergman. "The Diamond Lemma for Ring Theory." In: *Advances in Mathematics* 29.2 (February 1978), pp. 178–218. DOI: `10.1016/0001-8708(78)90010-5`.

[FH04] William Fulton and Joe Harris. *Representation Theory. A First Course.* 1st ed. Graduate Texts in Mathematics 129. Springer-Verlag New York, 2004. xv+551 pp. ISBN: 978-0-387-97527-6. DOI: `10.1007/978-1-4612-0979-9`.

[Lei14] Tom Leinster. *The bijection between projective indecomposable and simple modules.* October 14, 2014. arXiv: `1410.3671v1 [math.RA]`. (Visited on September 12, 2019).

[Tao10] Terence Tao. *Cayley graphs and the geometry of groups.* July 10, 2010. URL: `https://terrytao.wordpress.com/cayley-graphs-and-the-geometry-of-groups` (visited on September 6, 2019).

Compiling the bibliography

To get the output of Example 56 and Example 58 we actually have to compile the document in the right way:

Suppose that the `bib`-file has created, we have put the citations in the text via `\cite` and that we have placed `\printbibliography` in the source code. Suppose that our main file is called `main.tex` and the `bib`-file is called `references.bib`. To get the desired bibliography and cross-references between it and the text we need to complete three steps:

1. We compile the document `main.tex`. The compiler will note down in an auxiliary file `main.bcf` which labels are cited in this document
2. The back end program biber will go through the auxiliary files `main.bcf` and write down all the requested information in a new auxiliary file `main.bbl`.
3. We compile the document `main.tex` again. The compiler will read the various data given in the auxiliary file `main.bll` and, using the settings and commands from the package `biblatex`, will typeset both the citations in the main text and create a bibliography.

If you're using a specialized LaTeX editor or IDE (like TeXStudio, kile, etc.) and have everything properly configured then your editor should take care of the above steps automatically when(ever) the document is compiled. But if you're compiling by hand in the console then you will need three commands:

Example 2.59: Compiling in the console (with bibliography)

```
latex main.tex
biber main.bcf
latex main.tex
```

2.9 booktabs for tables

2.9.1 Recalling basic tables

The table itself

Recall that a table is constructed with the environment `tabular`:

Example 2.60: A basic table

```
\begin{tabular}{lcr}
  longtext & text     & text      \\
  text     & longtext & text      \\
  text     & text     & longtext
\end{tabular}
```

longtext	text	text
text	longtext	text
text	text	longtext

The labels `l`, `c`, `r` specify the alignment of the corresponding column (left aligned, centered, and right aligned).

Horizontal lines

The command `\hline` at the beginning of a row introduces a horizontal line that separates this row from the previous one. To get multiple parallel lines (e.g. a double line) one uses multiple instances of `\hline` directly after each other.

Example 2.61: Full horizontal lines in tables

```
\begin{tabular}{ccc}
  top left & top center & top right \\
  \hline\hline
  text     & text       & text      \\
  \hline
  text     & text       & text
\end{tabular}
```

top left	top center	top right
text	text	text
text	text	text

Partial horizontal lines

To separate only the columns $i, \ldots, j$ by horizontal line, the command `\cline` is used:

Example 2.62: Syntax of `\cline`

```
\cline{start column-end column}
```

The command `\cline` has the same placement as `\hline`:

Example 2.63: Partial horizontal lines in tables

```
\begin{tabular}{ccccc}
  text & text & text & text & text \\
  \cline{2-4}
  text & text & text & text & text \\
  \cline{1-2} \cline{4-5}
  text & text & text & text & text
\end{tabular}
```

text	text	text	text	text
text	text	text	text	text
text	text	text	text	text

Vertical lines

One can put a vertical line between two columns by adding the symbol `|` between the corresponding alignment symbols. Inserting this symbol multiple times will give parallel vertical lines.

Example 2.64: Vertical lines in tables

```
\begin{tabular}{l||c|c}
  first row   & text & text \\
  second row  & text & text \\
  third row   & text & text
\end{tabular}
```

first row	text	text
second row	text	text
third row	text	text

Different kinds of lines

The above effects can also be combined:

Example 2.65: A table with all kinds of lines in it

```
\begin{tabular}{|l||l|r|}
  \hline
  \textbf{Country}  &  \textbf{Town}  & \textbf{Population} \\
  \hline\hline
  France            & Paris           & 2,229,621           \\
  \cline{2-3}
  {}                & Marseille       &   855,393           \\
  \hline
  Germany           & Berlin          & 3,520,031           \\
  \cline{2-3}
  {}                & Hamburg         & 1,787,408           \\
  \hline
  Japan           & Tokyo          & 8,637,098          \\
  \cline{2-3}
  {}              & Yokohama       & 3,697,894          \\
  \hline
\end{tabular}
```

Country	**Town**	**Population**
France	Paris	2,229,621
	Marseille	855,393
Germany	Berlin	3,520,031
	Hamburg	1,787,408
Japan	Tokyo	8,637,098
	Yokohama	3,697,894

2.9.2 Problems and solutions

The above table has a huge problem: It's ugly. This is due to various reasons:

- The spacing between the horizontal lines and the text below them is both bad and inconsistent.
- The above table breaks the first rule of table club: Never, ever use vertical lines.
- The above table also breaks the second rule of table club: Never use double lines.

The last two points are easy to fix. The package `booktabs` gives a way for fixing the first problem:

- This package provides the commands `\toprule`, `\midrule` and `\bottomrule` as replacements for `\hline`. The command `\toprule` is to be used only for the line on top of the table. The command `\bottomrule` is similarly only to be used for the line on the bottom of the table. The horizontal line `\midrule` is meant to separate the main part of the table from the top part and bottom part.
- The command `\cmidrule` is the replacement for `\crule`.

One should also try to minimize the number of horizontal lines. The above example should hence look as follows:

Example 2.66: Using the rules of `booktabs`

```
\begin{tabular}{llr}
  \toprule
  \textbf{Country}  &  \textbf{Town}  & \textbf{Population} \\
  \midrule
  France          & Paris         & 2,229,621         \\
  {}              & Marseille     &   855,393         \\
  Germany         & Berlin        & 3,520,031         \\
  {}              & Hamburg       & 1,787,408         \\
  Japan           & Tokyo         & 8,637,098         \\
  {}              & Yokohama      & 3,697,894         \\
  \bottomrule
\end{tabular}
```

Country	**Town**	**Population**
France	Paris	2,229,621
	Marseille	855,393
Germany	Berlin	3,520,031
	Hamburg	1,787,408
Japan	Tokyo	8,637,098
	Yokohama	3,697,894

Note that we could leave out all non-essential horizontal lines because the table has a very regular form. We refer to [CTN16] for more information about typing tables.

Chapter 3

Writing text

Un texte mathématique est, d'abord, un texte.

Michèle Audin, *Conseils aux auteurs de textes mathématiques*

In this chapter we will talk about the more text focused aspects of writing mathematics with LaTeX. We will in particular talk about the interplay between the mathematical content and the text that surrounds it.

3.1 Obey orthography

A mathematical text has to obey the rules of the language that is it written in (e.g. English, French, German or Russian). A mathematical formula is part of the surrounding text, and has to be treated as such.

3.2 Proper punctuation

3.2.1 A sentence ends with punctuation

Section 3.1 has an important consequence: If a sentence ends with a formula, then this formula need to be followed by some kind of punctuation (in most cases by a period). The following example is very wrong:

Example 3.1: Very wrong punctuation in an equation

```
It follows that
\[
  a^2 + b^2 = c^2
\]
.
This formula is important.
```

It follows that

$$a^2 + b^2 = c^2$$

. This formula is important.

The next example is also wrong:

Example 3.2: Wrong punctuation in an equation I

```
It follows that
\[
  a^2 + b^2 = c^2
\]
This formula is important.
```

It follows that

$$a^2 + b^2 = c^2$$

This formula is important.

The following is still wrong:

Example 3.3: Wrong punctuation in an equation II

```
It follows that:
\[
  a^2 + b^2 = c^2
\]
This formula is important.
```

It follows that:

$$a^2 + b^2 = c^2$$

This formula is important.

The following example finally does it right:

Example 3.4: Right punctuation in an equation

```
It follows that
\[
  a^2 + b^2 = c^2.
\]
This formula is important.
```

It follows that

$$a^2 + b^2 = c^2.$$

This formula is important.

But it is even better if we add some slight spacing between the formula and the period.

Example 3.5: Best punctuation in an equation

```
It follows that
\[
  a^2 + b^2 = c^2 \,.
\]
This formula is important.
```

It follows that

$$a^2 + b^2 = c^2 \,.$$

This formula is important.

This last approach is taken from [TSE14], which we strongly encourage the reader to check out.

3.2.2 Punctuation in commutative diagrams?

There is some disagreement in the mathematical community about whether a commutative diagram is allowed to include punctuation coming from the surrounding text.

The author is of the opinion that a commutative diagram should not contain any such punctuation.

There are two standard ways to achieve this: One can finish up the sentence that precedes a commutative diagram beforehand:

Example 3.6: Finishing the sentence before the commutative diagram

```
We consider the following commutative diagram:
\[
  \begin{tikzcd}
      A
      \arrow{r}
      \arrow{d}
    &
      A'
      \arrow{d}
    \\
      C
      \arrow{r}
    &
      C'
  \end{tikzcd}
\]
The horizontal arrows in this diagram are isomorphisms.
```

We consider the following commutative diagram:

$$\begin{tikzcd} A \arrow[r] \arrow[d] & A' \arrow[d] \\ C \arrow[r] & C' \end{tikzcd}$$

The horizontal arrows in this diagram are isomorphisms.

But it often better to incorporate the diagram in the surrounding sentence in such a way that it contains no punctuation:

Example 3.7: Incorporating the commutative diagram in the sentence

```
In the commutative diagram
  \[
  \begin{tikzcd}
      A
      \arrow{r}
      \arrow{d}
    &
```

```
      A'
      \arrow{d}
    \\
      C
      \arrow{r}
    &
      C'
  \end{tikzcd}
\]
both horizontal arrows are isomorphisms.
```

In the commutative diagram

$$\begin{tikzcd} A \arrow[r] \arrow[d] & A' \arrow[d] \\ C \arrow[r] & C' \end{tikzcd}$$

both horizontal arrows are isomorphisms.

3.2.3 Lists contain punctuation

Text that is organized using list environments still obeys the rules of punctuation. Consider the following counterexample:

Example 3.8: Wrong punctuation in lists

```
A set $B$ is a basis of $V$ if
\begin{enumerate}
  \item
    $B$ is linearly independent
  \item
    $B$ is a generating set
\end{enumerate}
```

A set B is a basis of V if

1. B is linearly independent
2. B is a generating set

To figure out the correct punctuation simply remove the surrounding list and consider the resulting text. In the above example this gives the following:

A set B is a basis of V if B is linearly independent B is a generating set

This is not a proper sentence, and should instead be as follows:

> A set B is a basis of V if B is linearly independent and B is a generating set.

The above counterexample should hence read as follows:

Example 3.9: Right punctuation in lists I

```latex
A set $B$ is a basis of $V$ if
\begin{enumerate}
  \item
    $B$ is linearly independent and
  \item
    $B$ is a generating set.
\end{enumerate}
```

A set B is a basis of V if

1. B is linearly independent and
2. B is a generating set.

There are also some other acceptable versions:

Example 3.10: Right punctuation in lists II

```latex
A set $B$ is a basis of $V$ if it satisfies the following two conditions:
\begin{enumerate}
  \item
    $B$ is linearly independent.
  \item
    $B$ is a generating set.
\end{enumerate}
```

A set B is a basis of V if it satisfies the following two conditions:

1. B is linearly independent.
2. B is a generating set.

3.2.4 Hyphen and dashes

Know your lines

In LaTeX there are (at least) four line-like symbols which need to be distinguished, the hyphen, the en dash, the em dash and the minus sign. See Table 3.1 for their LaTeX code and look. Each of these symbols plays its own role, and these roles depend

name	**code**	**output**
hyphen	`-`	-
en dash	`--`	–
em dash	`---`	—
minus sign	`$-$`	$-$

Table 3.1: Kinds of hyphen and dashes.

language	**name**	**dash used**
English	Clebsch–Gordan coefficients	en dash
German	Clebsch-Gordan-Koeffizienten	hyphen
French	coefficients de Clebsch-Gordan	hyphen
Dutch	Clebsch-Gordan-coëfficienten	hyphen
Russian	Коэффициенты Клебша — Гордана	em dash
Japanese	クレブシュ–ゴルダン係数	en dash

Table 3.2: The Clebsch–Gordan coefficients in different languages.

both on language and on convention. For mathematical writing in English the following usages are important:

- For "n-dimensional" and "3-manifold", use the hyphen.
- For combining names as in "Cauchy–Schwarz inequality" use the en dash.
- To interrupt a sentence—as done here—use the em dash without surrounding space, or – if you're feeling British – use the en dash surrounded by space.

For more information about the use of hyphen and dashes in the English language we refer to [CMS17, 6.75–6.94].

To underline how language dependent the usage of hyphen and dashes is we're considering in Table 3.2 how the the Clebsch–Gordan are called in different languages, as taken from Wikipedia. (We use the same font[1] for all examples to allow a better comparison.)

Don't put a hyphen in math mode

A non-trivial amount of people makes the mistake of putting a hyphen into math mode.

[1]Namely "Noto Serif CJK JP".

right aligned		**left aligned**	
x. x	x. X	x. x	X. x
X. x	X. X	x. X	X. X

Table 3.3: Spacings after a period (with a zoom of factor 3.5).

Example 3.11: An accidental hyphen in math mode

```
Let~$A$ be an $R-$module.
```

Let A be an $R-$module.

We can see that we don't get a hyphen but a minus sign. We hence need to place the symbol `-` outside of the math environment.

Example 3.12: Proper placement of the hyphen

```
Let~$A$ be an $R$-module.
```

Let A be an R-module.

3.2.5 Correct spacing after a period

The problem

Not all periods are treated equally by LaTeX: If a period is both followed by whitespace and not preceded by an uppercase letter then LaTeX will assume that this period is supposed to end a sentence. It will some introduce some additional space following this period. The effect of this can be seen in Table 3.3.

We can see from the first two columns of Table 3.3 (by considering the positions of the periods) that the spacing in "`x. *`" is larger then the corresponding spacing in "`X. *`". This happens because LaTeX thinks that the period in "`x. *`" is supposed to end a sentence. We can also see from the first two tables that this difference of spacing occurs both if the following letter is lower case and if it is upper case. We can see from the third and fourth column that the amount of additional spacing does not depenend on whether the following letter is in lower case or upper case.

First Solution

The default behaviour of LaTeX, to put additional space after a period that ends a sentence, is outdated. So it is best to just deactivate it. This can via the command `\frenchspacing`. By putting this command in the preamble, the whole document will be affected.

Example 3.13: Using `\frenchspacing`

```latex
\documentclass[a4paper, 10pt]{scrartcl}

\frenchspacing

\begin{document}

No additional space after a sentence in this document.

\end{document}
```

The additional spacing can also be reenabled by `\nofrenchspacing`.

Example 3.14: Using `\nofrenchspacing`

```latex
\frenchspacing
Observe the spacing. Between the preceeding period.
\nonfrenchspacing\\
Observe the spacing. Between the preceeding period.
```

Observe the spacing. Between the preceeding period.
Observe the spacing. Between the preceeding period.

Second Solution

If the default behavior should be kept, then one needs to do more manual work.

We can use the command "\ " – which we will write for readability as `\(space)` – to tell LaTeX that a preceding period is not meant to end a sentence. We can similarly use the command `\@` to tell LaTeX that the following period is ending a sentence. Consider the following example:

Example 3.15: Proper spacing after periods

```latex
Imagine now some old and long-dead English kings, e.g.\ Henry III and Henry
IV\@.
(Kings not named Henry are also okay.)
```

Imagine now some old and long-dead English kings, e.g. Henry III and Henry IV. (Kings not named Henry are also okay.)

Note that in the above example one should also use ties ~ (as explained in Section 3.3.1) to ensure that the Henries don't lose their number.

Example 3.16: Proper spacing after periods, and using ~

```
Imagine now some old and long-dead English kings, e.g.\ Henry~III and Henry~IV\@.
(Kings not named Henry are also okay.)
```

Imagine now some old and long-dead English kings, e.g. Henry III and Henry IV. (Kings not named Henry are also okay.)

It should be pointed out that if a problematic period is followed by a closing parenthesis, a closing bracket, or quotations marks then then the problematic spacing will still occur after these symbols. Lets consider the following example (where we use the otherwise forbidden \\ to compare the versions).

Example 3.17: Spacings after periods followed by parentheses or quotation marks

```
Many colours (e.g. red, blue, etc.) are supported by \enquote{ColorX}. Even purple! \\
Many colours (e.g.\ red, blue, etc.) are supported by \enquote{ColorX}. Even purple! \\
Many colours (e.g.\ red, blue, etc.)\ are supported by \enquote{ColorX}. Even purple! \\
Many colours (e.g.\ red, blue, etc.)\ are supported by \enquote{ColorX}\@. Even purple!
```

Many colours (e.g. red, blue, etc.) are supported by “ColorX”. Even purple!
Many colours (e.g. red, blue, etc.) are supported by “ColorX”. Even purple!
Many colours (e.g. red, blue, etc.) are supported by “ColorX”. Even purple!
Many colours (e.g. red, blue, etc.) are supported by “ColorX”. Even purple!

The problem of a sentence ending with an upper case letter does not occur if this letter is in math mode.

Example 3.18: Upper case mathematics followed by a period

```
Let $f$ be an endomorphism of $X$.
Then $f$ is an isomorphism or $f = 0$.\\
Let $f$ be an endomorphism of $X$\@.
Then $f$ is an isomorphism or $f = 0$.\\
Let $f$ be an endomorphism of $X$.\
Then $f$ is an isomorphism or $f = 0$.
\\
Let $f$ be an endomorphism of $\operatorname{X}$.
Then $f$ is an isomorphism or $f = 0$.\\
Let $f$ be an endomorphism of $\operatorname{X}$\@.
Then $f$ is an isomorphism or $f = 0$.\\
Let $f$ be an endomorphism of $\operatorname{X}$.\
Then $f$ is an isomorphism or $f = 0$.
```

Let f be an endomorphism of X. Then f is an isomorphism or $f = 0$.
Let f be an endomorphism of X. Then f is an isomorphism or $f = 0$.
Let f be an endomorphism of X. Then f is an isomorphism or $f = 0$.
Let f be an endomorphism of X. Then f is an isomorphism or $f = 0$.
Let f be an endomorphism of X. Then f is an isomorphism or $f = 0$.
Let f be an endomorphism of X. Then f is an isomorphism or $f = 0$.

In view of the upcoming Section 3.3.1 we get a problem: What if we want the space after an abbreviation, as in "i.e. word", not to be broken? Should we then write `i.e.\ word` or `i.e.~word`?

In this case we write `i.e.~word`, since the tie `~` already contains the effect of `\(space)`. Consider for this the following example:

Example 3.19: Spacing of ~ and "\ "

```
Let $M$ be a $2$-manifold, e.g. $M = S^2$. \\
Let $M$ be a $2$-manifold, e.g.\ $M = S^2$. \\
Let $M$ be a $2$-manifold, e.g.~$M = S^2$.
```

Let M be a 2-manifold, e.g. $M = S^2$.
Let M be a 2-manifold, e.g. $M = S^2$.
Let M be a 2-manifold, e.g. $M = S^2$.

To summarize this section: If an abbreviation or some non-word things appear one should pay heed to periods which may occur at their ends. Most of these special cases are covered by paying attention to "i.e. *" and "e.g. *".

We refer to [L2M19, 19.5.1, 19.6] for more information on this topic.

3.3 Text layout

3.3.1 Use non-breakable space

If two expressions in text mode are separated by a space or a hyphen then LaTeX may separate these expressions in the output `pdf`-file by a line break. But such line breaks are often undesirable and can sometimes be downright wrong. Comsider for this the following examples:

- Abbreviations like "Prof. Dr. John Doe".
- Cmbinations of words and numbers like "Lemma 5".
- Combinations of mathematics and words like "3-dimensional".

In such cases one needs to prevent the line break.

Line breaks at a space

No line break will occur if the tie symbol `~` is used instead of a space. Consider for this the following example:

Example 3.20: An unwanted line break at a space

```
Text text text text text text text
text text text text text text, so
by Lemma 35 text text text text
text text text text
```

Text text text text text text text text
text text text text text, so by Lemma
35 text text text text text text text
text

We don't want a line break to occur in the term "Lemma 35" and thus introduce a tie.

Example 3.21: Using ~ to prevent unwanted line breaks at spaces

```
Text text text text text text text
text text text text text text, so
by Lemma~35 text text text text
text text text text
```

Text text text text text text text
text text text text text text, so by
Lemma 35 text text text text text
text text text

Line breaks at a hyphen

Consider the following example:

Example 3.22: An unwanted line break at a hyphen

```
 textextext textextext textextext textextext textextext textextext textext
123456-dimensional textextext
```

textextext textextext textextext textextext textextext textextext textext 123456-dimensional textextext

To prevent a line break at the hyphen we put `\nobreakdash` before it:

Example 3.23: Using `\nobreakdash` to prevent unwanted line breaks at hyphens

```
textextext textextext textextext textextext textextext textextext textext
123456\nobreakdash-dimensional textextext
```

textextext textextext textextext textextext textextext textextext textext 123456-dimensional textextext

The command `\nobreakdash` works not only with hyphens but also with en dashes and em dashes.

3.3.2 Don't use `\\` or `\newline`

The proper way to separate two succeeding paragraphs in LaTeX is to leave an empty line between them.

Example 3.24: New paragraphs done right

```
This is the first paragraph.
It consists of multiples lines to make this example better.

This is a second paragraph.
It also consists of multiple lines to make this example even more better.

This one is the third and last paragraph in this example.
It also consists of multiple lines.
```

This is the first paragraph. It consists of multiples lines to make this example better.

This is a second paragraph. It also consists of multiple lines to make this example even more better.

This one is the third and last paragraph in this example. It also consists of multiple lines.

Note that new paragraphs start indented.

The use of `\\` or `\newline` doesn't actually end a paragraph but only forces a line break.

Example 3.25: New “paragraphs” done wrong

```
This is the first paragraph in a counterexample.\\
This is not a new paragraph.
The text is just forced to start a new line.\newline
This one is also not a new paragraph.
But this whole thing looks ugly.
```

This is the first paragraph in a counterexample.
This is not a new paragraph. The text is just forced to start a new line.
This one is also not a new paragraph.
But this whole thing looks ugly.

The use of `\\` and `\newline` is only for starting new rows in matrices, tables and arrays.[2] Don’t use is to separate paragraphs.

3.3.3 Text consisting of lists

The two list environments `enumerate` and `itemize` contain by default certain indentations, that can be problematic in mathematical writing.

Identation compared to the left margin

There is by default some indentation between the left margin of the text area and the item labels. This is meant to visually separate the list from the surrounding text. But in mathematical writing there often occur lists that don’t have a surrounding text. The most prominent example for this are propositions (and their proofs) that consist of multiple statements:

Example 3.26: A list without surrounding text

```
\begin{theorem}
  Let $G$ be a finite group.
  \begin{enumerate}
    \item
      If $H$ is subgroup of $G$ then $\operatorname{ord}(H)$ divides $\operatorname{ord}(G)$.
    \item
      If $g$ is any element of $G$ then $\operatorname{ord}(g)$ divides $\operatorname{ord}(G)$.
    \item
      Every group of prime order is cyclic.
    \item
      If $p$ is a prime divisor of $\operatorname{ord}(G)$ then $G$ contains an element of order $p$.
  \end{enumerate}
```

[2]In this text we sometimes use `\\` to tweak examples, to make certain lines more comparable. This is done purely for technical purposes and should not be done for actual mathematical writings.

```latex
\end{theorem}
```

Theorem 1. *Let G be a finite group.*

1. *If H is subgroup of G then* $\operatorname{ord}(H)$ *divides* $\operatorname{ord}(G)$.
2. *If g is any element of G then* $\operatorname{ord}(g)$ *divides* $\operatorname{ord}(G)$.
3. *Every group of prime order is cyclic.*
4. *If p is a prime divisor of* $\operatorname{ord}(G)$ *then G contains an element of order p.*

In such cases one should remove the identation between the left margin of the text area and the item labels, as explained in Section 2.7.3.

Example 3.27: A list without surrounding text

```latex
\begin{theorem}
  Let $G$ be a finite group.
  \begin{enumerate}[wide = 0pt, leftmargin=*]
    \item
      If $H$ is subgroup of $G$ then $\operatorname{ord}(H)$ divides $\operatorname{ord}(G)$.
    \item
      If $g$ is any element of $G$ then $\operatorname{ord}(g)$ divides $\operatorname{ord}(G)$.
    \item
      Every group of prime order is cyclic.
    \item
      If $p$ is a prime divisor of $\operatorname{ord}(G)$ then $G$ contains an element of order $p$.
  \end{enumerate}
\end{theorem}
```

Theorem 2. *Let G be a finite group.*

1. *If H is subgroup of G then* $\operatorname{ord}(H)$ *divides* $\operatorname{ord}(G)$.
2. *If g is any element of G then* $\operatorname{ord}(g)$ *divides* $\operatorname{ord}(G)$.
3. *Every group of prime order is cyclic.*
4. *If p is a prime divisor of* $\operatorname{ord}(G)$ *then G contains an element of order p.*

Theorems and proofs beginning with a list

If a theorem-like environment or a the environment `proof` begins with a list environment then the first item of this list will not be aligned with the other list items:

Example 3.28: Improper aligned of list items

```
\begin{lemma}
\begin{enumerate}
  \item
    Every group $G$ is isomorphic to its opposite group $G^{\mathrm{op}}$.
  \item
    A group $G$ equals its opposite group $G^{\mathrm{op}}$ if and only if it
is abelian.
\end{enumerate}
\end{lemma}
```

Lemma 3. *1. Every group G is isomorphic to its opposite group G^{op}.*

2. A group G equals its opposite group G^{op} if and only if it is abelian.

One way to circumvent this problem is by inserting the command `\leavevmode` before the beginning of the list.

Example 3.29: Proper aligned of list items with `\leavevmode`

```
\begin{lemma}
\leavevmode
\begin{enumerate}
  \item
    Every group $G$ is isomorphic to its opposite group $G^{\mathrm{op}}$.
  \item
    A group $G$ equals its opposite group $G^{\mathrm{op}}$ if and only if it
is abelian.
\end{enumerate}
\end{lemma}
```

Lemma 4.

1. Every group G is isomorphic to its opposite group G^{op}.

2. A group G equals its opposite group G^{op} if and only if it is abelian.

The problem with this approach is that it allows a line break to occur at the position of `\leavevmode`. So it may happen that the header of the theorem-like environment of proof environment occurs at the very bottom of the page while the list appears only on the next page. (This looks quite horrible.)

As far as the author is aware there exists no good solution for this problem.[3] A good way of coping with this problem is to circumvent it by introducing some text before the list environment:

Example 3.30: Proper aligned of list items by introducing text

```
\begin{lemma}
Let $G$ be a group with opposite group $G^{\mathrm{op}}$.
\begin{enumerate}
  \item
    The groups $G$ and $G^{\mathrm{op}}$ are isomorphic.
  \item
    The groups $G$ and $G^{\mathrm{op}}$ are equal if and only if $G$ is
abelian.
\end{enumerate}
\end{lemma}
```

Lemma 5. *Let G be a group with opposite group G^{op}.*

1. *The groups G and G^{op} are isomorphic.*
2. *The groups G and G^{op} are equal if and only if G is abelian.*

An seen above we should also eliminate the indentation between the left margin of the text area and the item labels:

Example 3.31: Proper aligned of list items by introducing text

```
\begin{lemma}
Let $G$ be a group with opposite group $G^{\mathrm{op}}$.
\begin{enumerate}[wide = 0pt, leftmargin=*]
  \item
    The groups $G$ and $G^{\mathrm{op}}$ are isomorphic.
  \item
    The groups $G$ and $G^{\mathrm{op}}$ are equal if and only if $G$ is
abelian.
\end{enumerate}
\end{lemma}
```

Lemma 6. *Let G be a group with opposite group G^{op}.*

1. *The groups G and G^{op} are isomorphic.*

[3]The package `ntheorem` is allegedly able to address this problem, at the price of introducing new problems.

2. The groups G and G^{op} are equal if and only if G is abelian.

3.4 Mathematics in text

3.4.1 Don't break inline mathematics

When inline formulas or inline equations are too long or badly placed then it can happen that a line break occurs inside the formula, tearing it apart:

Example 3.32: Line break in inline math

```
The most important formula of all times is without a doubt is my mind $1 + 2 +
3 + 4 = 10$.
Truly a work of genius!
% fragile example
```

The most important formula of all times is without a doubt is my mind $1 + 2 +$
$3 + 4 = 10$. Truly a work of genius!

This horrible affront to nature can thankfully be eliminated by setting the penalties `\binoppenalty` and `\relpenalty` to their maximum value, which can be accessed via the command `\maxdimen`.

Example 3.33: Preventing inline math in inline math

```
% in the preamble:
\binoppenalty = \maxdimen
\relpenalty   = \maxdimen
% in the main text:
The most important formula of all times is without a doubt is my mind $1 + 2 +
3 + 4 = 10$.
Truly a work of genius!
```

The most important formula of all times is without a doubt is my mind
$1 + 2 + 3 + 4 = 10$. Truly a work of genius!

3.4.2 Don't begin a sentence with a mathematical symbol

Don't begin a sentence with a mathematical symbol. Consider the following example:

Example 3.34: Mathematics at the beginning of a sentence

```
\begin{theorem}
  $\mathbf{Mod}(R)$ is an abelian category.
\end{theorem}
```

Theorem 1. $\mathbf{Mod}(R)$ *is an abelian category.*

Instead do the following:

Example 3.35: No mathematics at the beginning of a sentence

```
\begin{theorem}
  The category $\mathbf{Mod}(R)$ is abelian.
\end{theorem}
```

Theorem 2. *The category* $\mathbf{Mod}(R)$ *is abelian.*

An exception to the above rule are lists, in which short entries may start with a math symbol. Consider for this the following example.

Example 3.36: List items beginning with mathematics

```
Let $V$ be a finite dimensional vector space.
Let $U$ and $W$ be two linear subspaces of $V$ of complementary dimensions,
i.e.\ such that $\dim V = \dim U + \dim W$.
Then the following conditions are equivalent:
\begin{enumerate}[label = \roman*)]
  \item
    $V = U \oplus W$.
  \item
    $V = U + W$.
  \item
    $U \cap W = 0$.
\end{enumerate}
```

Let V be a finite dimensional vector space. Let U and W be two linear subspaces of V of complementary dimensions, i.e. such that $\dim V = \dim U + \dim W$. Then the following conditions are equivalent:

i) $V = U \oplus W$.

ii) $V = U + W$.

iii) $U \cap W = 0$.

3.4.3 Don't put two formulas next to each other

One should not put two mathematical formulas right next to each other. Consider the following example:

Example 3.37: Not properly separating mathematics

```latex
Then $f^2 = g$,$f$ is differentiable and $f = \exp(h)$.
```

Then $f^2 = g$,f is differentiable and $f = \exp(h)$.

The human eye (and brain) parses this sentence as follows:

Then $\quad f^2 = g,f \quad$ is differentiable and $\quad f = \exp(h)$.

But this betrays the actual content of the sentence. This problem can be fixed by introducing some more text. One solution is the following:

Example 3.38: Properly separating mathematics I

```latex
Then $f^2 = g$, the function $f$ is differentiable and $f = \exp(h)$.
```

Then $f^2 = g$, the function f is differentiable and $f = \exp(h)$.

This sentence is parsed as follows:

Then $\quad f^2 = g, \quad$ the function $\quad f \quad$ is differentiable and $\quad f = \exp(h)$.

The following is another solution:

Example 3.39: Properly separating mathematics II

```latex
The function $f$ satisfies $f^2 = g$, it is differentiable and $f = \exp(h)$.
```

The function f satisfies $f^2 = g$, it is differentiable and $f = \exp(h)$.

This sentence is parsed as follows:

The function $\quad f \quad$ satisfies $\quad f^2 = g, \quad$ it is differentiable and $\quad f = \exp(h)$.

3.4.4 Don't begin a line with a mathematical symbol (optional)

One interesting idea for inline mathematics is to never start a line with a mathematical symbol. This can be achieved by prefixing every inline mathematical formula by a tie `~`. The author has taken this approach from [Bel18].

3.5 Text in mathematics

3.5.1 Use `\text`

When text needs to be used in math mode the command `\text` is to be used.

Example 3.40: Syntax of `\text`

```
\text{normal text in math mode}
```

Let's start with the following horrible example:

Example 3.41: Missing text in math mode

```
Let $A \coloneqq \{ x \in [0,1] : x \; is \; rational \}$.
```

Let $A \coloneqq \{ x \in [0,1] : x \; is \; rational \}$.

This next one looks a bit better, but is still not okay:

Example 3.42: Wrong way of using text in math mode

```
Let $A \coloneqq \{ x \in [0,1] : x \text{ is rational} \}$.
```

Let $A \coloneqq \{ x \in [0,1] : x \text{ is rational} \}$.

Instead do this:

Example 3.43: Right way of using text in math mode

```
Let $A \coloneqq \{ x \in [0,1] : \text{$x$ is rational} \}$.
```

Let $A \coloneqq \{ x \in [0,1] : \text{x is rational} \}$.

Note that the right code reflects the actual structure of the displayed expression: Indeed, the diagram in Figure 3.1 shows the structure of the given expression. To get the right kind of code we assign to each node of this diagram the desired content, and whether this node is mathematics or text. If a node contains both mathematics and text then it is primary a text, which just happens to contain some mathematics. We arrive at the diagram in Figure 3.2. By transforming this diagram node-wise into code we arrive at Example 43. Note that both the nodes "$x \in [0,1]$" and "x is rational" both contain mathematics, but that the second is actually a text. It is therefore not proper to combine these two math modes, as done in Example 42.

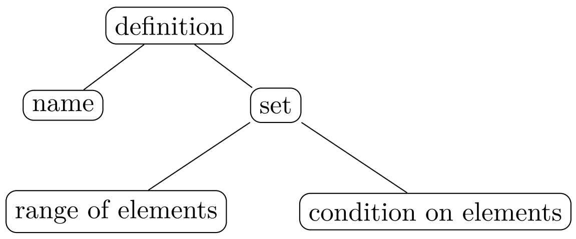


Figure 3.1: Structure tree of the definition.

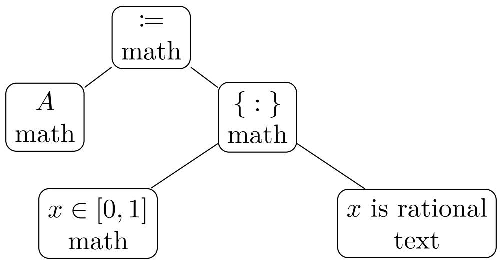


Figure 3.2: Structure tree of the mathematical expression.

3.5.2 Put space around text in display mathematics

Sometimes a single line of displayed mathematics end with some text, as in the following example.

Example 3.44: Wrong spacing around text in math mode I

```
Therefore
\[
  g h g^{-1}
  =
  h
  \text{for all $g, h \in G$}.
\]
```

Therefore

$$ghg^{-1} = h\text{for all } g, h \in G.$$

In such a case one needs to put space between the mathematics and the text. This space should not be part of the text. Consider the following counterexample:

Example 3.45: Wrong spacing around text in math mode II

```
Therefore
\[
  g h g^{-1}
  =
  h
  \text{ for all $g, h \in G$.}
\]
```

Therefore
$$ghg^{-1} = h \text{ for all } g, h \in G.$$

Instead the spacing should be put between the formula and the text.

In the above example, where a single block of mathematics is follows by a single block of text which contains quantifiers for the previous formula, one should use the spacing `\qquad`:

Example 3.46: Right spacing around text in math mode I

```
Therefore
\[
  g h g^{-1}
  =
  h
  \qquad
  \text{for all $g, h \in G$.}
\]
```

Therefore
$$ghg^{-1} = h \qquad \text{for all } g, h \in G.$$

Sometimes two blocks of mathematics are separated by a short text. In this case one should use the spacing `\quad` on both sides of the text:

Example 3.47: Right spacing around text in math mode II

```
That $gh = hg$ can equivalently be expressed as
\[
  g h g^{-1} = h
  \quad\text{and}\quad
  h g h^{-1} = g \,.
\]
```

That $gh = hg$ can equivalently be expressed as

$$ghg^{-1} = h \quad \text{and} \quad hgh^{-1} = g\,.$$

It may even be appropriate to use `\qquad` on both sides when more space is needed.

3.5.3 Use `\intertext` and `\shortintertext`

Multi-line display mode environments like `gather*` and `align*` (see Section 5.1) can be interrupted by inserting some text via the commands `\intersect` and `\shortintertext` to insert some text between different lines of mathematics. This is particularly useful to combine multiple `align*` environments into a single one, which then allows for a common alignment of all lines.

The following is an example for what we don't want.

Example 3.48: Two non-aligned blocks

```
We consider the equalities
\begin{align*}
  H
  &= a_1 + a_2 + a_3 + a_4 \\
  &= b_1 + b_2 + b_3 + b_4 + b_5 + b_6 \\
  &= c_1 + c_2 + c_3 + c_4 + c_5
\end{align*}
and
\begin{align*}
  I
  &= d_1 + d_2 + d_3 + d_4 + d_5 + d_6 \\
  &= e_1 + e_2 + e_3 \\
  &= f_1 + f_2 + f_3 + f_4 + f_5 \,.
\end{align*}
```

We consider the equalities

$$\begin{aligned} H &= a_1 + a_2 + a_3 + a_4 \\ &= b_1 + b_2 + b_3 + b_4 + b_5 + b_6 \\ &= c_1 + c_2 + c_3 + c_4 + c_5 \end{aligned}$$

and

$$\begin{aligned} I &= d_1 + d_2 + d_3 + d_4 + d_5 + d_6 \\ &= e_1 + e_2 + e_3 \\ &= f_1 + f_2 + f_3 + f_4 + f_5\,. \end{aligned}$$

Note the annoying misalignment of the equality signs of the two `align*` environments. We can solve this problem by using only one `align*` environment and inserting the text “and” by using the command `\intertext`:

Example 3.49: Two aligned blocks

```
We consider now the equalities
\begin{align*}
  H
  &= a_1 + a_2 + a_3 + a_4 \\
  &= b_1 + b_2 + b_3 + b_4 + b_5 + b_6 \\
  &= c_1 + c_2 + c_3 + c_4 + c_5
\intertext{and}
  I
  &= d_1 + d_2 + d_3 + d_4 + d_5 + d_6 \\
  &= e_1 + e_2 + e_3 \\
  &= f_1 + f_2 + f_3 + f_4 + f_5 \,.
\end{align*}
```

We consider now the equalities

$$\begin{aligned} H &= a_1 + a_2 + a_3 + a_4 \\ &= b_1 + b_2 + b_3 + b_4 + b_5 + b_6 \\ &= c_1 + c_2 + c_3 + c_4 + c_5 \end{aligned}$$

and

$$\begin{aligned} I &= d_1 + d_2 + d_3 + d_4 + d_5 + d_6 \\ &= e_1 + e_2 + e_3 \\ &= f_1 + f_2 + f_3 + f_4 + f_5 \,. \end{aligned}$$

The command `\shortintertext` works in the same way as `\intertext` but inserts less vertical space around it. To see the effects of `\shortintertext` let us start with the following example:

Example 3.50: Two non-aligned equalities

```
It follows from the identity
\[
  a = b
\]
that
\[
  b = a \,,
\]
which proves the theorem.
```

It follows from the identity

$$a = b$$

that

$$b = a\,,$$

which proves the theorem.

We naturally want to align the two equality signs, which we by using a single `\align*` environment together with `\intertext`, as explained above:

Example 3.51: Two aligned equalities with too much space

```
It follows from the identity
\begin{align*}
  a &= b
\intertext{that}
  b &= a \,,
\end{align*}
which proves the theorem.
```

It follows from the identity

$$a = b$$

that

$$b = a\,,$$

which proves the theorem.

But we can see that the inserted `\intersect` puts a large distance between the two formulas. This distance is far too large for our taste. We can solve this problem by using `\shortintertext` instead of `\intertext`, as the following example shows:

Example 3.52: Two aligned equalities with proper space

```
It follows from the identity
\begin{align*}
  a &= b
\shortintertext{that}
  b &= a \,,
\end{align*}
which proves the theorem.
```

It follows from the identity

$$a = b$$

that

$$b = a\,,$$

which proves the theorem.

3.6 Mathematics as text

3.6.1 Don't abuse math as text

Again similar to Section 3.6.2 one should not abuse mathematical symbols as words:
A mathematical symbol can often replace parts of a sentence, in particular a verb.

Example 3.53: Right way of using mathematical symbols as words

```
Then $a = b$ and thus $b \leq c$.
```

Then $a = b$ and thus $b \leq c$.

But this use of mathematics should be treated with care, and not abused:

Example 3.54: Wrong way of using mathematical symbols as words

```
The integer $a$ is $< 0$ and thus $\notin \mathbb{N}$.
```

The integer a is < 0 and thus $\notin \mathbb{N}$.

This kind of tortue is highly illegal, so don't do it!

3.6.2 Logical symbols don't replace text

Mathematicians often abuse logical symbols like

$$\exists\,, \quad \forall\,, \quad \implies, \quad \iff$$

as replacements for written out words. This is okay in handwriting and on blackboards, but it is not okay in a properly written out mathematical text.

Example 3.55: Abuse of quantifiers

```
It follows that $\forall y \in B$ that $\exists x \in A$ with $f(x) = y$.
```

It follows that $\forall y \in B$ that $\exists x \in A$ with $f(x) = y$.

One needs to instead write out the used quantifiers.

Example 3.56: Writing out quantifiers

```
It follows that for every element $y \in B$ there exists an element $x \in A$
with $f(x) = y$.
```

It follows that for every element $y \in B$ there exists an element $x \in A$ with $f(x) = y$.

In this example one should also write out the element relation:

Example 3.57: Best use of quantifiers

```
It follows that for every element $y$ of $B$ there exists an element $x$ of $A$
with $f(x) = y$.
```

It follows that for every element y of B there exists an element x of A with $f(x) = y$.

The use of the above logical symbols is of course okay if they are used in their actual logical function:

Example 3.58: Okay way to use logical symbols

```
Let $f \colon X \to Y$ be a morphism in a category $\mathcal{C}$.
Then
\begin{align*}
  {}&
    \text{$f$ is an epimorphism}
  \\
  \iff{}&
  \bigl[
  \forall Z \in \operatorname{Ob}(\mathcal{C}):
  \forall g, h \in \mathcal{C}(Y,Z):
  g \circ f = h \circ f \implies g = h
  \bigr]
  \\
  \iff{}&
  \bigl[
  \forall Z \in \operatorname{Ob}(\mathcal{C}):
  \text{$f^* \colon \mathcal{C}(Y,Z) \to \mathcal{C}(X,Z)$ is injective}
  \bigr]
  \\
  \implies{}&
  \text{$f^* \colon \mathcal{C}(Y,-) \to \mathcal{C}(X,-)$ is a monomorphism
in~$\mathbf{Fun}(\mathcal{C}, \mathbf{Set})$}
\end{align*}
```

Let $f\colon X \to Y$ be a morphism in a category $\mathcal{C}$. Then

$$\begin{aligned}
& f \text{ is an epimorphism} \\
\iff & \big[\forall Z \in \mathrm{Ob}(\mathcal{C}) : \forall g, h \in \mathcal{C}(Y, Z) : g \circ f = h \circ f \implies g = h\big] \\
\iff & \big[\forall Z \in \mathrm{Ob}(\mathcal{C}) : f^*\colon \mathcal{C}(Y, Z) \to \mathcal{C}(X, Z) \text{ is injective}\big] \\
\implies & f^*\colon \mathcal{C}(Y, -) \to \mathcal{C}(X, -) \text{ is a monomorphism in } \mathbf{Fun}(\mathcal{C}, \mathbf{Set})
\end{aligned}$$

3.6.3 Don’t use “iff”

Similarly to Section 3.6.2 abbreviations like “iff” need to be written out in a proper mathematical text.

Chapter 4

Good behaviour inside math mode

4.1 Using the right commands and symbols

4.1.1 General symbols

Use the correct symbols. Table 4.1 shows some popular sources of this problem. Note that the commands `\rightarrow` and `\to` give the same arrow. So use whichever is more appropriate in the given situation.

4.1.2 Operations

Many mathematical operations have both a binary version and an operator version, where the operation can range over some index set. One should not confuse the two of them. Table 4.2 shows some popular binary operations and their operator counterpart. The command `\bigsqcap` requires the package `stmaryrd`.

4.1.3 Negations

For some mathematical symbols there also exists a negated version, which is formed by diagonal line through the symbol. A general way of introducing such a line is the command `\not`.

Example 4.1: Using `\not`

```
It follows that $A \not\ni x$.
```

It follows that $A \not\ni x$.

But in practice this command should seldom be used, as it often produces very bad looking output. Consider the following example:

Example 4.2: Why not to use `\not`

```
Hence $A \not\implies B$.
```

Hence $A \not\implies B$.

There are two solutions to this problem:

symbol	right commands		wrong commands	
element relation	`\in`	$\in$	`\epsilon`	ϵ
			`\varepsilon`	ε
	`\ni`	$\ni$	`\backepsilon`	$\backepsilon$
empty set	`\emptyset`	$\emptyset$	`\phi`	ϕ
	`\varnothing`	$\varnothing$		
set difference	`A \setminus B`	$A \setminus B$	`A \backslash B`	$A\backslash B$
	`A \smallsetminus B`	$A \smallsetminus B$		
	`A - B`	$A - B$		
implication	`\implies`	$\implies$	`\Rightarrow`	$\Rightarrow$
			`=>`	$=>$
	`\impliedby`	$\impliedby$	`\Leftarrow`	$\Leftarrow$
			`<=`	$<=$
equivalence	`\iff`	$\iff$	`\Leftrightarrow`	$\Leftrightarrow$
			`<=>`	$<=>$
definition	`\coloneqq`	$\coloneqq$	`:=`	$:=$
	`\eqqcolon`	$\eqqcolon$	`=:`	$=:$
norm	`| x |`	$\|x\|$	`|| x ||`	$||x||$
	`\lVert x \rVert`			
pointy brackets	`\langle x \rangle`	$\langle x\rangle$	`< x >`	$< x >$
infinity	`\infty`	∞	`oo`	oo
function colon	`f \colon X \to Y`	$f\colon X \to Y$	`f : X \to Y`	$f : X \to Y$

Table 4.1: Right symbols and wrong symbols.

Many symbols already have a predefined lined-out version available. A few of them are collected in Table 4.3. If the required symbol is not predefined then the command `\centernot` from the package `centernot` often produces better looking output than `\not`:

Example 4.3: Using the command `\centernot`

```
Consider
\[
  A \centernot\ni x
  \quad\text{versus}\quad
  \quad
  A \not\ni x \,.
\]
```

operation	**binary**		**generalized**	
sum	`+`	$a + b$	`\sum`	$\sum_i x_i$
multiplication	`\cdot`	$a \cdot b$	`\prod`	$\prod_i x_i$
direct sum	`\oplus`	$A \oplus B$	`\bigoplus`	$\bigoplus_i X_i$
tensor product	`\otimes`	$A \otimes B$	`\bigotimes`	$\bigotimes_i X_i$
wedge	`\wedge`	$a \wedge b$	`\bigwedge`	$\bigwedge_i X_i$
union	`\cup`	$A \cup B$	`\bigcup`	$\bigcup_i X_i$
intersection	`\cap`	$A \cap B$	`\bigcap`	$\bigcap_i X_i$
product	`\times`	$A \times B$	`\prod`	$\prod_i X_i$
	`\sqcap`	$A \sqcap B$	`\bigsqcap`	$\bigsqcap_i X_i$
coproduct	`\amalg`	$A \amalg B$	`\coprod`	$\coprod_i X_i$
	`\sqcup`	$A \sqcup B$	`\bigsqcup`	$\bigsqcup_i X_i$

Table 4.2: Binary operation and operator version.

right		**wrong**	
`\notin`	$\notin$	`\not\in`	$\not\in$
`\nexists`	$\nexists$	`\not\exists`	$\not\exists$
`\neq`	$\neq$	`\not =`	$\not=$
`\nleq`	$\nleq$	`\not\leq`	$\not\leq$
`\nrightarrow`	$\nrightarrow$	`\not\rightarrow`	$\not\rightarrow$

Table 4.3: Negated versions of popular symbols.

```
Consider also
\[
  A \centernot\implies B
  \quad\text{versus}\quad
  A \not\implies B \,.
\]
```

Consider

$$A \not\ni x \quad\text{versus}\qquad A \not\ni x\,.$$

Consider also

$$A \centernot\implies B \quad\text{versus}\quad A \not\implies B\,.$$

4.1.4 Don't use `\limits`

People seem to think that they have to put `\limits` after a command to add limits to it:

Example 4.4: Using `\limits`

```
\[
  \sum\limits_{k=1}^n k
  =
  \frac{n(n+1)}{2}
\]
```

$$\sum_{k=1}^n k = \frac{n(n+1)}{2}$$

But this is not only unnecessary, but also dangerous. It is unnecessary because (most of) the commands in questions already have this functionality built in:

Example 4.5: Using built-in limits

```
\[
  \sum_{k=1}^n k
  =
  \frac{n(n+1)}{2}
\]
```

$$\sum_{k=1}^n k = \frac{n(n+1)}{2}$$

These built-in limits have the advantage that they can distinguish between inline math and display math:

Example 4.6: Inline math vs. display math with built-in limits

```
The sum $\sum_{k=0}^n 2^k = 2^{n+1}
 - 1$ is inline while the sum
\[
  \sum_{k=0}^n 3^k
  \neq
  3^{n+1} - 1
\]
is in display mode.
```

The sum $\sum_{k=0}^n 2^k = 2^{n+1}-1$ is inline while the sum

$$\sum_{k=0}^n 3^k \neq 3^{n+1} - 1$$

is in display mode.

We can see that the inline version does not only use the smaller summation sign but also sets the limits to the right of the summation sign. This is a feature which `\limits` version is missing:

Example 4.7: Inline math with `\limits`

```
So here is some text which will generate some lines.
The text itself 'isnt important, but we really want it to fill some lines.
The important thing is the sum $\sum\limits_{k=0}^n k$.
Well, not really the sum itself, but the use of \texttt{{\textbackslash}limits} for typesetting it.
```

So here is some text which will generate some lines. The text itself isn't important, but we really want it to fill some lines. The important thing is the sum $\sum\limits_{k=0}^n k$. Well, not really the sum itself, but the use of `\limits` for typesetting it.

The limits are still placed above the top and below the summation sign. This has its price: The line in which the sum resides breaks the usual vertical space between lines, which gives the text an inconsistent and unorganized look. Compare this to the version without `\limits`:

Example 4.8: Inline with built-in limits

```
So here is some text which will generate some lines.
The text itself 'isnt important, but we really want it to fill some lines.
The important thing is the sum $\sum_{k=0}^n k$.
Well, not really the sum itself, but use of built-in limits for typesetting it.
```

So here is some text which will generate some lines. The text itself isn't important, but we really want it to fill some lines. The important thing is the sum $\sum_{k=0}^n k$. Well, not really the sum itself, but use of built-in limits for typesetting it.

Here the line distance is nicely consistent and pleasing to the eye.

The usual predefined commands on which one would expect limits already have them defined, e.g. `\sum`, `\prod` or `\lim`, as the following example shows:

Example 4.9: Inline vs. display for different commands with built-in limits

```
Compare the inline versions $\sum_{k=0}^n k$ and $\prod_{k=1}^n k$ and $\lim_{n \to \infty} a_n$ with the display versions
\[
  \sum_{k=0}^n k \,,
  \quad
  \prod_{k=1}^n k \,,
  \quad
  \lim_{n \to \infty} a_n \,.
\]
```

Compare the inline versions $\sum_{k=0}^{n} k$ and $\prod_{k=1}^{n} k$ and $\lim_{n\to\infty} a_n$ with the display versions

$$\sum_{k=0}^{n} k\,, \quad \prod_{k=1}^{n} k\,, \quad \lim_{n\to\infty} a_n\,.$$

If a new commands are defined with `\DeclareMathOperator` or `\operatorname` then one can make them support limits by using `\DeclareMathOperator*` and `\operatorname*` instead. Suppose for example that we have made in the preamble the following definition:

Example 4.10: Using `\DeclareMathOperator*` to define `\colim`

```
\DeclareMathOperator*{\colim}{colim}
```

We can then do the following:

Example 4.11: Using `\colim`

```
Inline we have $\colim_{X' \leq X} F(X')$ and in display mode we get
\[
  \colim_{X' \leq X} F(X') \,.
\]
```

Inline we have $\operatorname{colim}_{X'\leq X} F(X')$ and in display mode we get

$$\operatorname*{colim}_{X'\leq X} F(X')\,.$$

The command `\operatorname*` was named `\operatornamewithlimits` in the past, but this name is deprecated.

4.1.5 Use extensible arrows instead of `\overset` and `\underset`

Some people put text above or under arrows by wrongly using the commands `\overset` or `\underset`:

Example 4.12: Using `\overset` and `\underset` to put text above or below an arrow

```
\[
  X
  \overset{f}{\longrightarrow}
  Y
  \underset{g \circ h \circ k}{\longrightarrow}
  Z
\]
```

$$X \xrightarrow{f} Y \underset{g \circ h \circ k}{\longrightarrow} Z$$

We can see above that the length of the arrow does not adjust to the size of the text above it or below it. The proper way to put text on top of an arrow or bellow an arrow of the form "→" is therefore the command `\xrightarrow`:

Example 4.13: Using `\xrightarrow` to put text above or below an arrow

```
\[
  X
  \xrightarrow{f}
  Y
  \xrightarrow[g \circ h \circ k]{}
  Z \,.
\]
```

$$X \xrightarrow{f} Y \xrightarrow[g \circ h \circ k]{} Z \,.$$

The package `amsmath` defines only the two most basic extensible arrows. Many more kinds of extensible arrows are provided by the package `mathtools` and some more are contained in the package `extarrows`. An overview of the various kinds of extensible arrows can be found in Table 4.4.

The author recommends to define custom commands as shortcuts for the most used arrows.

Example 4.14: Defining arrow commands as shortcuts

```
\newcommand{\xto}{\xrightarrow}
\newcommand{\xlongto}[1]{\xlongrightarrow{\;#1\;}}
The function $A \xto{f} B$ is the same as
\[
  A \xlongto{f} B \,.
\]
```

The function $A \xrightarrow{f} B$ is the same as

$$A \xrightarrow{\;f\;} B \,.$$

`amsmath`	$A \xrightarrow{f} B$ `\xrightarrow`	$A \xleftarrow{f} B$ `\xleftarrow`	
`mathtools`	$A \xmapsto{f} B$ `\xmapsto`	$A \xleftrightarrow{f} B$ `\xleftrightarrow`	$A \xRightarrow{f} B$ `\xRightarrow`
	$A \xLeftarrow{f} B$ `\xLeftarrow`	$A \xLeftrightarrow{f} B$ `\xLeftrightarrow`	$A \xhookleftarrow{f} B$ `\xhookleftarrow`
	$A \xhookrightarrow{f} B$ `\xhookrightarrow`	$A \xrightharpoondown{f} B$ `\xrightharpoondown`	$A \xrightharpoonup{f} B$ `\xrightharpoonup`
	$A \xrightleftharpoons{f} B$ `\xrightleftharpoons`	$A \xleftharpoondown{f} B$ `\xleftharpoondown`	$A \xleftharpoonup{f} B$ `\xleftharpoonup`
	$A \xleftrightharpoons{f} B$ `\xleftrightharpoons`		
`extarrows`	$A \xlongequal{f} B$ `\xlongequal`	$A \xleftrightarrow{f} B$ `\xleftrightarrow`	$A \xLeftrightarrow{f} B$ `\xLeftrightarrow`
	$A \xlongleftarrow{f} B$ `\xlongleftarrow`	$A \xlongrightarrow{f} B$ `\xlongrightarrow`	$A \xlongleftrightarrow{f} B$ `\xlongleftrightarrow`
	$A \xLongleftarrow{f} B$ `\xLongleftarrow`	$A \xLongrightarrow{f} B$ `\xLongrightarrow`	$A \xLongleftrightarrow{f} B$ `\xLongleftrightarrow`

Table 4.4: Extensible arrows in `amsmath`, `mathtools` and `extarrows`.

4.1.6 Know your ellipses

An ellipsis should *never* by written as `...` (three single dots). LaTeX instead provides different kinds of ellipses to use in math mode, namely

`\dotsb`, `\dotsc`, `\dotsm`, `\dotsi`, `\dotso`,
`\cdots`, `\ddots`, `\vdots`, `\ldots`.

Each of these have they own role and (mostly) distinct look and feel. There are two groups of ellipses. The first group consists of `\dotsb`, `\dotsc`, `\dotsm`, `\dotsi`,`\dotso`, whereas the second group consists of `\cdots`, `\ddots`, `\vdots`, `\ldots`.

The ellipses in the first group are named after their function, see Table 4.5. Note that the members of this group all have names of the form `\dots*`, where `*` is a letter that specifies the semantic use of the ellipsis.

The ellipses in the second group are not named after their function but after their orientation, see Table 4.6 The ellipses `\cdots`, `\ddots` and `\vdots` are typically used in

command	**where to use**	**example**
`\dotsb`	between binary relations	$x_1 \leq \dots \leq x_n$
	and binary operations	$x_1 + \dots + x_n$
`\dotsc`	between commas	$x_1, \dots, x_n$
`\dotsm`	abbreviating multiplication	$x_1 \dotsm x_n$
`\dotsi`	iterated integrals	$\int_{X_1} \dotsi \int_{X_n}$
`\dotso`	others	

Table 4.5: Ellipses with a specific functions.

command	**description**	**look**
`\cdots`	Horizontal dots, vertically centered.	$\cdots$
`\ddots`	Diagonal dots.	$\ddots$
`\vdots`	Vertical dots, horizontally centered.	$\vdots$
`\ldots`	Lowered dots.	$\ldots$

Table 4.6: Ellipses with a specific orientation.

matrices.

Example 4.15: Typical usage of the ellipses `\cdots`, `\vdots`, `\ddots`

```
\[
  \begin{pmatrix}
    a_{11} & \cdots & a_{1n} \\
    \vdots & \ddots & \vdots \\
    a_{n1} & \cdots & a_{nn}
  \end{pmatrix}
\]
```

$$\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix}$$

The command `\ldots` may be used to denote left out digits or symbols.

Example 4.16: Using the ellipsis `\ldots`

```
We find that
\[
  x = 0.1234567891011\ldots
\]
Consider the word $w = a_1 \ldots a_n$ where $a_1, \dotsc, a_n$ are letters in
an alphabet $\Sigma$.
```

We find that

$$x = 0.1234567891011\ldots$$

Consider the word $w = a_1 \ldots a_n$ where $a_1, \dotsc, a_n$ are letters in an alphabet Σ.

Note that the members of this second group of dots all have names of the form `\*dots`, where `*` is a letter that specified the positioning of these dots.

So overall one should use the ellipses `\cdots`, `\ddots` and `\vdots` for matrices, and otherwise the ellipses `\dotsb`, `\dotsc`, `\dotsm`, `\dotsi` and occasionally `\dotso`.

There also exists the generic command `\dots` which tries to automagically use the right kind of positioning and spacing. But the author recommends not using this command as it can lead to inconsistent results:

Example 4.17: Inconsistent results with `\dots`

```
The ellipses in $x_1 \leq x_2 \leq \dots \leq x_n$ and $y_1 \leq y_2 \leq
\dots$ should look the same.
```

The ellipses in $x_1 \leq x_2 \leq \cdots \leq x_n$ and $y_1 \leq y_2 \leq \ldots$ should look the same.

4.1.7 Know your matrices

The package `amsmath` provides various environments for matrices, the most basic of which is the environment `matrix`.

Example 4.18: The basic matrix environment `matrix`

```
\[
  \begin{matrix}
    a & b \\
    c & d
  \end{matrix}
\]
```

$$\begin{matrix} a & b \\ c & d \end{matrix}$$

There are five more variants of this basic matrix environment, that put different delimiters around the matrix. The package `mathtools` also provides small versions of all six kinds of matrices. See Table 4.7 for a table of all these kinds of matrices.

size	**matrices**		
normal	$\begin{matrix} a & b \\ c & d \end{matrix}$ `matrix`	$\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ `pmatrix`	$\begin{bmatrix} a & b \\ c & d \end{bmatrix}$ `bmatrix`
	$\begin{Bmatrix} a & b \\ c & d \end{Bmatrix}$ `Bmatrix`	$\begin{vmatrix} a & b \\ c & d \end{vmatrix}$ `vmatrix`	$\begin{Vmatrix} a & b \\ c & d \end{Vmatrix}$ `Vmatrix`
small	$\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}$ `smallmatrix`	$\left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right)$ `psmallmatrix`	$\left[\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right]$ `bsmallmatrix`
	$\left\{\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right\}$ `Bsmallmatrix`	$\left\lvert\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right\rvert$ `vsmallmatrix`	$\left\lVert\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right\rVert$ `Vsmallmatrix`

Table 4.7: Different kinds of matrices.

The package `mathtools` also provides starred versions of the matrices. These starred versions allow to specify as an optional argument the alignment of the columns, see Table 4.7. The three possible arguments are `l` for left-alignment, `c` for centered alignment and `r` for right-alignment. The standard alignment is `c`.

Example 4.19: Aligning matrix entries

```
\[
  \begin{bmatrix*}[l]
      a & -b \\
     -c &  d
  \end{bmatrix*}
  \qquad
  \begin{bmatrix*}[c]
    a & -b \\
    -c &  d
  \end{bmatrix*}
  \qquad
  \begin{bmatrix*}[r]
     a & -b \\
    -c &  d
  \end{bmatrix*}
  \qquad
  \begin{bmatrix}
     a & -b \\
    -c &  d
  \end{bmatrix}
\]
```

$$\begin{bmatrix} a & -b \\ -c & d \end{bmatrix} \quad \begin{bmatrix} a & -b \\ -c & d \end{bmatrix} \quad \begin{bmatrix} a & -b \\ -c & d \end{bmatrix} \quad \begin{bmatrix} a & -b \\ -c & d \end{bmatrix}$$

One should choose the kind of matrix, its size and alignment dependent on convention, usage and the matrix entries.

Example 4.20: Using the right kind of matrix

```
We consider the morphism
\[
  X \oplus Y
  \xrightarrow{\,
    \begin{bsmallmatrix*}[r]
      f & -d \\
      0 &  g
    \end{bsmallmatrix*}
    \,
  }
  X' \oplus Y' \,.
\]
```

We consider the morphism

$$X \oplus Y \xrightarrow{\,\left[\begin{smallmatrix} f & -d \\ 0 & g \end{smallmatrix}\right]\,} X' \oplus Y' \,.$$

4.2 Avoid bad notation

4.2.1 Don't use `\subset`

Some people use $\subset$ (`\subset`) to denote inclusion and $\subsetneq$ (`\subsetneq`) to denote proper inclusion, while some other people use $\subseteq$ (`\subseteq`) to denote inclusion and $\subset$ to denote proper inclusion. The second convention has the advantage of making sense and being consistent with the usual use of $\leq$ and $<$, whereas the first convention has the non-advantage of existing. The problem is that both conventions are wide-spread while using the symbol $\subset$ is different ways.

The conflict between the above two conventions has abused the symbol $\subset$ to a point that it should *never* be used. Instead one should always use $\subseteq$ for inclusion, $\subsetneq$ for proper inclusion and $\nsubseteq$ (`\nsubseteq`) for non-inclusion.

Some people prefer the symbols $\subseteqq$ (`\subseteqq`) and $\subsetneqq$ (`\subsetneqq`) instead. The author thinks that these symbols are unnecessary large and recommends not to use them. (But they do at least leave poor old $\subset$ alone.)

4.2.2 Don't underline

Underlining works well on the blackboard, in handwriting, and was useful in the in the (dark) age of typewriters. Don't do it in LaTeX.

4.2.3 Use $\mathrm{d}x$ instead of dx

A differential is written as `\mathrm{d}x`, not as `dx`. When it occurs at the end of an integral then a slight spacing `\,` is also introduced in front of it. So don't do the following:

Example 4.21: Wrong kind of differentials

```
\[
  \int_a^b f(x) dx
\]
```

$$\int_a^b f(x)dx$$

Instead do the following:

Example 4.22: Right kind of differential

```
\[
  \int_a^b f(x) \,\mathrm{d}x
\]
```

$$\int_a^b f(x)\,\mathrm{d}x$$

4.2.4 Don't force fancy fractions

When fractions are placed inline, are part of an exponent or part of an index, then they should be of the form a/b. The notation

$$\frac{a}{b}$$

is reserved for display style. So don't do the following:

Example 4.23: Full fractions in exponent, index and inline

```
Consider $e^{\frac{1}{x}}$ and $x_{\frac{1}{n}}$ and $\frac{2}{3}$.
```

Consider $e^{\frac{1}{x}}$ and $x_{\frac{1}{n}}$ and $\frac{2}{3}$.

Do the following instead:

Example 4.24: Flat fractions in exponent, index and inline

```
Consider $e^{1/x}$ and $x_{1/n}$ and $2/3$.
```

Consider $e^{1/x}$ and $x_{1/n}$ and $2/3$.

Don't use funky fractions like $^a\!/_b$. They'll make you go blind and burn down your house.

4.3 Defining new commands

Many mathematical operators have predefined commands, which should be used when needed.

Example 4.25: Using commands vs. not using them

```
Use $\sin(x)$ instead of $sin x$, use $\dim V$ instead of $dim V$ and use
$\lim_{n \to \infty} a_n$ instead of $lim_{n \to \infty} a_n$.
```

Use $\sin(x)$ instead of $sin x$, use $\dim V$ instead of $dim V$ and use $\lim_{n \to \infty} a_n$ instead of $lim_{n \to \infty} a_n$.

New commands can be defined in various ways:

4.3.1 `\DeclareMathOperator`

Commands of the form `\WordCommand`, that are supposed to print "WordOutupt", can easily be defined using the command `\DeclareMathOperator`.

Example 4.26: Syntax of `\DeclareMathOperator`

```
\DeclareMathOperator{\WordCommand}{WordOutput}
```

The command `\DeclareMathOperator` can only be used in the preamble.

To define the command `\End` we use the following text in the preamble:

Example 4.27: Declaring a mathematical operator with `\DeclareMathOperator`

```
% in the preamble:
\DeclareMathOperator{\End}{End}
```

The command `\End` can then be used in the usual way:

Example 4.28: Using a declared mathematical operator

```
Thus $\End(V) = \End_k(V)$ becomes a vector space.
```

Thus $\operatorname{End}(V) = \operatorname{End}_k(V)$ becomes a vector space.

If a command `\WordCommand` is defined with `\DeclareMathOperator` then LaTeX will automatically insert some space around `\WordCommand` when needed:

Example 4.29: Automatic spacing of `\DeclareMathOperator`

```
\begin{align*}
&x \End V
\\
&x \End(V)
\\
&x \End {(V)}
\end{align*}
```

$$x \operatorname{End} V$$
$$x \operatorname{End}(V)$$
$$x \operatorname{End} {(V)}$$

Note that in the first expression LaTeX inserts some spacing both to the left and to the right of End. In the second expression LaTeX observes that the used math operator is follows by a parenthesis and thus inserts no additional spacing. For the third expression we prevent LaTeX from making such an observation by using a pair of curly brackets.

This behavior leads to a bad looking output when `\DeclareMathoperator` is abused. Suppose that we want a command `\Complex` that inserts the code `\mathbb{C}`.

Example 4.30: Wrong way of using `\DeclareMathOperator`

```
% in the preamble:
\DeclareMathOperator{\Complex}{\mathbb{C}}
```

This will lead to the following problem:

Example 4.31: Wrong output when `\DeclareMathOperator` is abused

```
The span of $x_1, \dotsc, x_n \in \Complex^m$ equals $\Complex x_1 + \dotsb +
\Complex x_n$.
```

The span of $x_1, \dots, x_n \in \mathbb{C}^m$ equals $\mathbb{C}\, x_1 + \dots + \mathbb{C}\, x_n$.

We expect the output $\mathbb{C}x_1 + \dots + \mathbb{C}x_n$ but get some unwanted spacing instead.

4.3.2 `\operatorname`

The command `\operatorname` can be used to give the formatting of a mathemical operator without defining a new command.

Example 4.32: Using `\operatorname`

```
Thus $\operatorname{Hom}(V,W) = \operatorname{Hom}_k(V,W)$ becomes a vector
space.
```

Thus $\operatorname{Hom}(V,W) = \operatorname{Hom}_k(V,W)$ becomes a vector space.

If the same command is used multiple times then one should use `\DeclareMathOperator` instead of `\operatorname`, to keep the code clean.

4.3.3 Don't abuse `\mathrm`

The commands `\mathrm` and `\operatorname` do not give the same formatting. With `\operatorname` we get the necessary spacing when not using parentheses We don't get this from `\mathrm`.

Example 4.33: Missing spacing after `\mathrm`

```
Compare $\operatorname{End} V$ to $\mathrm{End} V$, and also~$2
\operatorname{Fr}(x)$ to~$\mathrm{Fr}(x)$.
```

Compare $\operatorname{End} V$ to $\mathrm{End} V$, and also $2\operatorname{Fr}(x)$ to $\mathrm{Fr}(x)$.

4.3.4 `\newcommand`

A very general way of defining new commands is given by `\newcommand`. Its syntax is as follows:

Example 4.34: Syntax of `\newcommand`

```
\newcommand{\name}[number of arguments n]{ definition including #1, ..., #n }
```

Consider the following example:

Example 4.35: Using the command `\newcommand`

```
\newcommand{\bimodule}[2]{#1-#2-bimodule}
Let $M$ be an \bimodule{$A$}{$B$}.
```

Let M be an A-B-bimodule.

One may think about `\DeclareMathOperator` as a combination of `\newcommand` and `\operatorname`:

Example 4.36: `\DeclareMathOperator` = `\newcommand` plus `\operatorname`

```
\newcommand{\Ouv}{\operatorname{Ouv}}
$\Ouv X$
```

$\operatorname{Ouv} X$

Trying to define an already existing command with `\newcommand` will lead to an error. To overwrite an already existing command one can use `\renewcommand` instead. But this shouldn't really be done (unless you really, *really* know what you're doing). Even if you don't like a particular command it may happen that some of the packages which you're using rely on it: overwriting the command can then easily lead to some unexpected new problems.

4.3.5 `\DeclarePairedDelimiter`

Mathematical operations like the absolute value $|\cdot|$ and a norm $\|\cdot\|$ are denoted by putting certain delimiters around the argument. To define a corresponding LaTeX command like `\abs` or `\norm` one should use the command `\DeclarePairedDelimiter` (which is provided by the package `mathtools`).

Example 4.37: Syntax of `\DeclarePairedDelimiter`

```
\DeclarePairedDelimiter{\name}{left delimiter}{right delimiter}
```

As an example we use `\DeclarePairedDelimiter` to define a command `\abs` for absolute value:

Example 4.38: Defining the command `\abs` with `\DeclarePairedDelimiter`

```
\DeclarePairedDelimiter{\abs}{\lvert}{\rvert}
```

The defined command can now be used as expected:

Example 4.39: Using a declared delimiter

```
\[
  \abs{-5} = 5
\]
```

$$|-5| = 5$$

Declaring a command via `\DeclarePairedDelimiter` will automatically also define a starred version that scales the surrounding delimiters according to the given content between them. One can also specify a scaling size like `\big`, `\bigg`, etc. to scale the delimiters.

Example 4.40: Scaling of declared delimiters

```
\begin{align*}
  \abs{-\frac{1}{2}}
  &=
  \frac{1}{2} \,,
  \\
  \abs*{-\frac{1}{2}}
  &=
  \frac{1}{2} \,,
  \\
  \abs[\bigg]{-\frac{1}{2}}
  &=
  \frac{1}{2}
\end{align*}
```

$$\begin{aligned}
|-\tfrac{1}{2}| &= \frac{1}{2}\,, \\
\left|-\frac{1}{2}\right| &= \frac{1}{2}\,, \\
\bigg|-\frac{1}{2}\bigg| &= \frac{1}{2}
\end{aligned}$$

The variant `\DeclarePairedDelimiterX` allows building more sophisticated commands:

Example 4.41: Syntax of `\DeclarePairedDelimiterX`

```
\DeclarePairedDelimiterX{\name}
  [number of arguments n]
  {left delimiter}{right delimiter}
  {expression build from #1, #2, ..., #n}
```

If the built up expression contains a delimiter, then by prefixing this delimiter with the command `\delimsize` will ensure that the delimiter scales in the same way as the two surrounding delimiters. As an example we define a command `\inner` for inner product.

Example 4.42: Using `\DeclarePairedDelimiterX` for more advanced delimiters

```
\DeclarePairedDelimiterX{\inner}[2]{\langle}{\rangle}{#1 \,\delimsize\vert\, #2}
```

The defined command `\inner` takes two arguments and inserts a line (`\vert`) with some surrounding space (`\,`) between them.

Example 4.43: Using more advanced delimiters

```
\[
  \inner{\psi_1}{\psi_2}
  \quad
  \inner*{\frac{f}{g}}{\frac{h}{k}}
\]
```

$$\langle \psi_1 \mid \psi_2 \rangle \quad \left\langle \frac{f}{g} \middle| \frac{h}{k} \right\rangle$$

There is also the variant `\DeclarePairedDelimiterXPP` that is even more flexible than `\DeclarePairedDelimiterX`. We refer to [CTN19b] for more details on these commands.

4.3.6 The package `xparse`

A useful way for defining commands – in particular more involved ones – is provided by the package `xparse`. This package provides the command `\NewDocumentCommand`.

Example 4.44: Syntax of `\NewDocumentCommand`

```
\NewDocumentCommand{\name}{arguments}{definition}
```

The field `arguments` specifies what kinds of arguments the command `\name` will take:

Mandatory arguments

Mandatory arguments can be declared with the option `m`:

Example 4.45: Mandatory arguments with `\NewDocumentCommand` II

```
% in the preamble
\NewDocumentCommand{\double}{m}{#1 #1}
% in the main text
\double{sometext}
```

sometext sometext

One can also specify multiple arguments:

Example 4.46: Mandatory arguments with `\NewDocumentCommand` II

```
% in the preamble
\NewDocumentCommand{\swap}{m m}{#2 #1}
% in the main text
```

```
\swap{first}{second}
```

second first

Optional arguments without default value

Optional arguments can be declared with the option `o`. It needs to be checked with the command `\IfNoValueTF` if this optional argument was assigned a value.

Example 4.47: Syntax of `\IfValueTF`

```
\IfValueTF{#number of argument}
  {if the argument has been set}
  {if the argument has not been set}
```

One can similarly use `\IfNoValueTF` instead of `\IfNoValueTF`. This has the same effect as switching the cases in Example 47.

Let's look at a specific example:

Example 4.48: Optional arguments with `\NewDocumentCommand`, I

```
% in the preamble
\NewDocumentCommand
{\module}
{m o}
{\IfNoValueTF{#2}{#1-module}{#1-#2-bimodule}}
% in the main text
Let $M$ be an \module{$R$} and let $N$ be an \module{$R$}[$S$].
```

Let M be an R-module and let N be an R-S-bimodule.

It is useful to properly indent the source code. But we have to be careful here: It can easily happen that this indentation introduces some unwanted whitespace. Let's consider the naive code first:

Example 4.49: Optional arguments with `\NewDocumentCommand`, I

```
% in the preamble
\NewDocumentCommand
{\module}
{m o}
{
  \IfNoValueTF{#2}
    {#1-module}
    {#1-#2-bimodule}
}
% in the main text
Let $M$ be an \module{$R$} and let $N$ be an \module{$R$}[$S$].
```

Let M be an R-module and let N be an R-S-bimodule .

Now let's “comment out” the additional whitespace.

Example 4.50: Optional arguments with `\NewDocumentCommand`, I

```
% in the preamble
\NewDocumentCommand
{\module}
{m o}
{%
  \IfNoValueTF{#2}
    {#1-module}
    {#1-#2-bimodule}%
}
% in the main text
Let $M$ be an \module{$R$} and let $N$ be an \module{$R$}[$S$].
```

Let M be an R-module and let N be an R-S-bimodule.

Optional arguments with default value

One can use `O{default value}` instead of `o` to declare an optional argument that has a default value.

Let's start with an example where this default value is empty:

Example 4.51: Optional arguments with `\NewDocumentCommand`, II

```
% in the preamble
\NewDocumentCommand{\restrict}{m m O{}}{#1|_{#2}^{#3}}
% in the main text
Let $f = \restrict{g}{X}$ and $f' = \restrict{g'}{X}[Y]$
```

Let $f = g|_X$ and $f' = g'|_X^Y$

Now let's do an example which has a standard value:

Example 4.52: Optional arguments with `\NewDocumentCommand`, III

```
% in the preamble
\NewDocumentCommand{\favorite}{O{colour} m}{My favorite #1 is #2.}
% in the main text
\favorite{blue}
\favorite[food]{spam}
```

My favorite colour is blue. My favorite food is spam.

Starred versions

One can use the argument `*` together with the command `\IfBooleanTF` to check for the occurrence of a star. This can be used to also define a starred version of a command.

Example 4.53: Starred versions with `\NewDocumentCommand`

```
% in the preamble
\NewDocumentCommand{\choice}{s m m}{\IfBooleanTF{#1}{#3}{#2}}
% in the main text'
Dont confuse \choice{first}{second} with \choice*{first}{second}.
```

Don’t confuse first with second.

4.4 Stretch your arrows

If some expression occurs atop or below an arrow, then this arrow must be stretched suffciently long to accommodate this expression. The extensible arrows introduced in Section 4.1.5 automatically do so.

4.4.1 Stretching columns

If an arrow in a commutative diagram isn’t long enough then this arrow must also be stretched:

Example 4.54: Commutative diagram with an arrow too short

```
\[
\begin{tikzcd}
  X \arrow{r}{f \circ g - g \circ f} \arrow{d}
  &
  Y \arrow{r}{k} \arrow{d}
  &
```

name	tiny	small	scriptsize	normal	large	huge
distance	0.6em	1.2em	1.8em	2.4em	3.6em	4.8em

Table 4.8: Standard distances for `column sep`.

```latex
  Z \arrow[equal]{d}
  \\
  X' \arrow[dashed]{r}{h'}
  &
  Y' \arrow{r}{k'}
  &
  Z'
\end{tikzcd}
\]
```

$$\begin{tikzcd}
X \arrow[r, "f\circ g - g\circ f"] \arrow[d] & Y \arrow[r, "k"] \arrow[d] & Z \arrow[d, equal] \\
X' \arrow[r, dashed, "h'"] & Y' \arrow[r, "k'"] & Z'
\end{tikzcd}$$

The column distance of a commutative diagram is governed by the option `column sep`. The value of `column sep` is expected to be a distance, e.g. `4em`. Some standard distances are predefined, see Table 4.8. With `column sep` one can fix the above diagram.

Example 4.55: Using `column sep`

```latex
\[
\begin{tikzcd}[column sep = huge]
  X \arrow{r}{f \circ g - g \circ f} \arrow{d}
  &
  Y \arrow{r}{k} \arrow{d}
  &
  Z \arrow[equal]{d}
  \\
  X' \arrow[dashed]{r}{h'}
  &
  Y' \arrow{r}{k'}
  &
  Z'
\end{tikzcd}
\]
```

$$\begin{tikzcd}[column sep=huge]
X \arrow[r, "f\circ g-g\circ f"] \arrow[d] & Y \arrow[r, "k"] \arrow[d] & Z \arrow[d, equal] \\
X' \arrow[r, dashed, "h'"] & Y' \arrow[r, "k'"] & Z'
\end{tikzcd}$$

One can also increase the width of a specific column by specifying the additional width at the correct &–symbol in the first row. (In the above examples, the difference between `normal` and `huge` is `2.4em`.)

Example 4.56: Explicit column spacing

```
\[
\begin{tikzcd}
  X \arrow{r}{f \circ g - g \circ f} \arrow{d}
  &[2.4em]
  Y \arrow{r}{k} \arrow{d}
  &
  Z \arrow[equal]{d}
  \\
  X' \arrow[dashed]{r}{h'}
  &
  Y' \arrow{r}{k'}
  &
  Z'
\end{tikzcd}
\]
```

$$\begin{tikzcd}
X \arrow[r, "f\circ g-g\circ f"] \arrow[d] &[2.4em] Y \arrow[r, "k"] \arrow[d] & Z \arrow[d, equal] \\
X' \arrow[r, dashed, "h'"] & Y' \arrow[r, "k'"] & Z'
\end{tikzcd}$$

4.4.2 Stretch rows

One can similarly use the option `row sep` to change the distance of the rows. Some predefined distances can be found in Table 4.9. They are the same as for `column sep` but scaled down by a factor of 0.75. One can also explicitely specify an additional space between two rows after the correct `\\`.

name	`tiny`	`small`	`scriptsize`	`normal`	`large`	`huge`
distance	`0.45em`	`0.9em`	`1.35em`	`1.8em`	`2.7em`	`3.6em`

Table 4.9: Standard distances for `row sep`.

Example 4.57: Using `row sep` and explicit row spacing

```
\[
  \begin{tikzcd}[row sep = huge]
    A \arrow{r} \arrow{d}
    &
    B \arrow{d}
    \\
    C \arrow{r} \arrow{d}
    &
    D \arrow{d}
    \\
    E \arrow{r}
    &
    F
  \end{tikzcd}
  \qquad
  \begin{tikzcd}
    A \arrow{r} \arrow{d}
    &
    B \arrow{d}
    \\[1.8em]
    C \arrow{r} \arrow{d}
    &
    D \arrow{d}
    \\
    E \arrow{r}
    &
    F
  \end{tikzcd}
\]
```

$$\begin{tikzcd}
A \arrow[r] \arrow[d] & B \arrow[d] \\
C \arrow[r] \arrow[d] & D \arrow[d] \\
E \arrow[r] & F
\end{tikzcd}
\qquad
\begin{tikzcd}
A \arrow[r] \arrow[d] & B \arrow[d] \\
C \arrow[r] \arrow[d] & D \arrow[d] \\
E \arrow[r] & F
\end{tikzcd}$$

4.5 Beware of spacings

LaTeX classifies symbols and expressions into certain groups and then adds spacing around these symbols, which depends on the group they belong to. Three of these groups are “operators”, “relation symbols” and “binary operations”. The symbols `=` and `<` are for example treated as relations symbols, and the symbols `+` and `\cdot` as binary operations. We can see in the following example how some space is automatically added around these symbols:

Example 4.58: Standard spacing around relation symbols and binary operators

```
\[
  a = b  \qquad  a < b  \qquad  a + b  \qquad  a \cdot b
\]
```

$$a = b \qquad a < b \qquad a + b \qquad a \cdot b$$

To compare this to a version without spacing we can surround the symbols by a pair of curly brackets. This circumvents LaTeX from taking the surrounding code into consideration.

Example 4.59: Disabling the standard spacing around a symbol

```
\[
  a {=} b   \qquad  a {<} b  \qquad  a {+} b  \qquad  a {\cdot} b
\]
```

$$a {=} b \qquad a {<} b \qquad a {+} b \qquad a {\cdot} b$$

The automatic spacing can become a problem, as the following examples illustrate:

Example 4.60: Clashing spacings around symbols

```
\[
  X/\sim
  \quad
  R/\operatorname{J}(R)
  \quad
  \operatorname{id} \otimes h
\]
```

$$X/\sim \quad R/\operatorname{J}(R) \quad \operatorname{id} \otimes h$$

These problems can be fixed by surrounding the respective symbols with curly brackets.

Example 4.61: Preventing a clash of spacings

```
\[
  X/{\sim}
  \quad
  R/{\operatorname{J}(R)}
  \quad
  {\operatorname{id}} \otimes h
\]
```

$$X/{\sim} \quad R/{\operatorname{J}(R)} \quad {\operatorname{id}} \otimes h$$

One can also tell LaTeX a symbol should be treated.

Example 4.62: Specifying the role (and thus spacing) of a symbol

```
\[
  a | b
  \quad
  a \mathop{|} b
  \quad
  a \mathrel{|} b
  \quad
  a \mathbin{|} b
\]
```

$$a | b \quad a \mathop{|} b \quad a \mathrel{|} b \quad a \mathbin{|} b$$

We can thus define a command `\divides`, which expresses that a number n divides a number m, as follows:

Example 4.63: Defining and using `\divides`

```
\newcommand{\divides}{\mathrel{|}}
\[
  n \divides m
\]
```

$$n \mathrel{|} m$$

For more on this topic we refer to [TSE11].

Chapter 5

Mathematical layout

5.1 Environments for display mathematics

5.1.1 Don't use `$$ $$` or `eqnarray`

There are many good ways to put mathematical content into display mode, but `$$ $$` and `eqnarray` are none of them. The method `$$ $$` is too low level, and the environment `eqnarray` has too many problem and has long been deprecated. One can instead use any of the following, depending on the planned usage.

5.1.2 The environments `\[ \]` and `equation*`

The environments `\[ \]` and `equation*` can be used for a single line of display math mode. Both commands do the exactly same thing (when the package `amsmath` is loaded).

Example 5.1: Using the environments `\[ \]` and `equation*`

```
Suppose that both the formula
\[
  a + b = c
\]
and the formula
\begin{equation*}
  2a - b = c \,.
\end{equation*}
hold.
Then $a$ and $b$ are unique.
```

Suppose that both the formula

$$a + b = c$$

and the formula

$$2a - b = c\,.$$

hold. Then $a$ and $b$ are unique.

The non-starred version `equation` numbers the equation.

Example 5.2: Using the environment `equation`

```
The formula
\begin{equation}
  a^2 - b^2 = (a + b)(a - b)
\end{equation}
is one of the binomial formulas.
```

The formula

$$a^2 - b^2 = (a + b)(a - b) \tag{5.1}$$

is one of the binomial formulas.

5.1.3 The environment `gather*`

The environment `gather*` is meant for multiple lines that are non-aligned.

Example 5.3: Using the environment `gather*`

```
We consider for every integer $n \geq 0$ the polynomial
\[
  p_n
  =
  \sum_{k=0}^n x^k \,.
\]
In particular
\begin{gather*}
  p_0 = 1 \,,
  \qquad
  p_1 = 1 + x \,,
  \qquad
  p_2 = 1 + x + x^2 \,,
  \qquad
  p_3 = 1 + x + x^2 + x^3 \,,
  \\
  p_4 = 1 + x + x^2 + x^3 + x^4 \,,
  \qquad
  p_5 = 1 + x + x^2 + x^3 + x^4 + x^5 \,.
\end{gather*}
```

We consider for every integer $n \geq 0$ the polynomial

$$p_n = \sum_{k=0}^n x^k \,.$$

In particular

$$\begin{gather*} p_0 = 1 \,, \qquad p_1 = 1 + x \,, \qquad p_2 = 1 + x + x^2 \,, \qquad p_3 = 1 + x + x^2 + x^3 \,, \\ p_4 = 1 + x + x^2 + x^3 + x^4 \,, \qquad p_5 = 1 + x + x^2 + x^3 + x^4 + x^5 \,. \end{gather*}$$

The non-starred version `gather` numbers each line.

Example 5.4: Using the environment `gather`

```
We have the polynomials
\begin{gather}
  p_0 = 1 \,,
  \qquad
  p_1 = 1 + x \,,
  \qquad
  p_2 = 1 + x + x^2 \,,
  \qquad
  p_3 = 1 + x + x^2 + x^3 \,,
  \\
  p_4 = 1 + x + x^2 + x^3 + x^4 \,,
  \qquad
  p_5 = 1 + x + x^2 + x^3 + x^4 + x^5 \,.
\end{gather}
```

We have the polynomials

$$p_0 = 1\,, \qquad p_1 = 1 + x\,, \qquad p_2 = 1 + x + x^2\,, \qquad p_3 = 1 + x + x^2 + x^3\,, \tag{5.2}$$

$$p_4 = 1 + x + x^2 + x^3 + x^4\,, \qquad p_5 = 1 + x + x^2 + x^3 + x^4 + x^5\,. \tag{5.3}$$

5.1.4 The environments `align*` and `alignat*`

The environment `align*` allows for multiple lines. Each line contains the symbol `&` once, and the occurrences of this symbol are then aligned.

Example 5.5: Using the environment `align*`

```
We find that
\begin{align*}
  a + b + c
  &=
  d + e + f + g
  \\
  &=
  h + i + j
  \\
  &=
  k + l + m + n \,.
\end{align*}
```

We find that

$$\begin{aligned} a + b + c &= d + e + f + g \\ &= h + i + j \\ &= k + l + m + n\,. \end{aligned}$$

The unstarred version `align` numbers the lines.

Example 5.6: Using the environment `align`

```
We find again that
\begin{align}
  a + b + c
  &=
  d + e + f + g
  \\
  &=
  h + i + j
  \\
  &=
  k + l + m + n \,.
\end{align}
```

We find again that

$$
\begin{align}
a+b+c &= d+e+f+g \tag{5.4}\\
&= h+i+j \tag{5.5}\\
&= k+l+m+n\,. \tag{5.6}
\end{align}
$$

One can also use multiple aligned columns, which then need to be separated by an additional `&`. For n aligned columns we hence need $2n-1$ occurrences of `&` per line.

Example 5.7: Using `align*` with multiple columns

```
We consider the values
\begin{align*}
  x_1 &= 1 \,,  &   x_2 &= 2 \,,  &   x_3 &= 3 \,,  \\
  x_4 &= 4 \,,  &   x_5 &= 5 \,,  &   x_6 &= 6 \,,  \\
  x_7 &= 7 \,,  &   x_8 &= 8 \,,  &   x_9 &= 9 \,.
\end{align*}
```

We consider the values

$$
\begin{align*}
x_1 &= 1\,, & x_2 &= 2\,, & x_3 &= 3\,,\\
x_4 &= 4\,, & x_5 &= 5\,, & x_6 &= 6\,,\\
x_7 &= 7\,, & x_8 &= 8\,, & x_9 &= 9\,.
\end{align*}
$$

The environment `alignat*` is similar to the environment `align*` but doesn't add any built-in spacing between the aligned columns. Any such spacing must therefore by added by hand. One also has to specify the number of columns beforehand.

Example 5.8: Using the environment `alignat*`

```
We also consider the values
\begin{alignat*}{3}
  y_1 &= 9 \,,  &\qquad   y_2 &= 8 \,,  &\qquad   y_3 &= 7 \,,  \\
  y_4 &= 6 \,,  &         y_5 &= 5 \,,  &         y_6 &= 4 \,,  \\
  y_7 &= 3 \,,  &         y_8 &= 2 \,,  &         y_9 &= 1 \,.
\end{alignat*}
```

We also consider the values

$$\begin{alignedat}{3}
y_1 &= 9\,, \qquad & y_2 &= 8\,, \qquad & y_3 &= 7\,, \\
y_4 &= 6\,, & y_5 &= 5\,, & y_6 &= 4\,, \\
y_7 &= 3\,, & y_8 &= 2\,, & y_9 &= 1\,.
\end{alignedat}$$

To align multiple columns one should use `alignat*` instead of `align*` to get (manually) a good looking distance between the aligned columns. The environment `alignat` works in the same way as `alignat*` but automatically numbers the lines.

5.1.5 Don't use `center` plus `$ $`

By all that is holy, don't do the following:

Example 5.9: Using `\center` with `$ $`

```
It holds that
\begin{center}
  $\sum_{i=0}^n 2^i = 2^{n+1} - 1$.
\end{center}
This can be shown by induction.
```

It holds that

$\sum_{i=0}^n 2^i = 2^{n+1} - 1$.

This can be shown by induction.

5.1.6 The environments `gathered`, `aligned` and `alignedat`

The environments `gathered`, `aligned` and `alignedat` are variations of the environments `gather*`, `align*` and `alignat*` that can be used inside an already existing math environment.

Example 5.10: Using `aligned`

```
\[
  \left\{
    \begin{aligned}
      a + b &= c      \\
      d     &= e + f
    \end{aligned}
  \right\}
\]
```

$$\left\{ \begin{aligned} a + b &= c \\ d &= e + f \end{aligned} \right\}$$

5.1.7 Overview

One should always use the most basic environment that does the job: Using overpowered environments can lead to unexpected problems. Consider the following example:

Example 5.11: Improper use of `align*`

```
\begin{align*}
  ABCD = EF = GHI = JKL = M = NOP
  \\
  QRS = TUV = WX = Y = Z
\end{align*}
```

$$\begin{aligned} ABCD = EF = GHI = JKL = M = NOP \\ QRS = TUV = WX = Y = Z \end{aligned}$$

The environment `align*` automatically alignes both lines on the right since no information about alignment was given. In the above situation one should use `gather*` instead.

Example 5.12: Using `gather*` for non-aligned equations

```
\begin{gather*}
  ABCD = EF = GHI = JKL = M = NOP
  \\
  QRS = TUV = WX = Y = Z
\end{gather*}
```

$$\begin{gathered} ABCD = EF = GHI = JKL = M = NOP \\ QRS = TUV = WX = Y = Z \end{gathered}$$

If a multi-line display mode environment is used for a single line then one can get spacing issues, see Section 5.4.

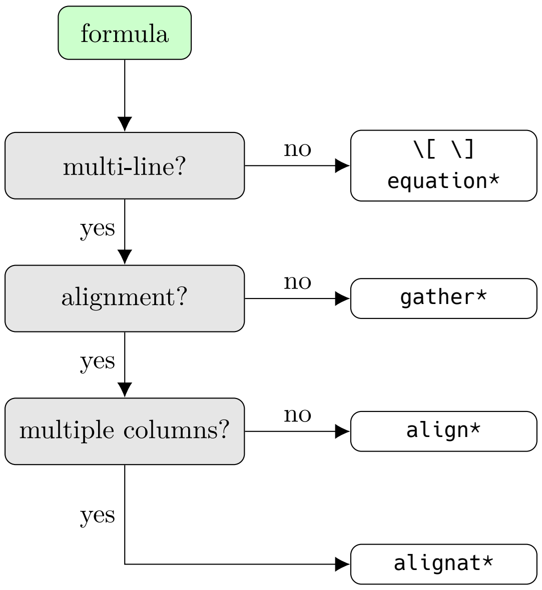
The flowchart in Figure 5.1 explains how to choose the correct display environment. (This flowchart is partly inspired by [Kot15].)

5.2 Where to break and align formulas

Oftentimes a formula is broken among multiple lines. This is done for at least two reasons:

- To improve the readability of the given formula.
- To prevent that the formula goes over the margins of the text area.

If a formula is broken among multiple lines then one has to choose at which places the formula should be broken, and how the resulting parts of the formula will then be aligned. In this section we present some standard ways of doing so.



If numbering of the line(s) is needed then the unstarred version is to be used.

Figure 5.1: Deciding on a math environment.

5.2.1 When to break a formula

We first discuss when a formula needs to be broken.

When the formula is too long

If a formula is too long to physically fit into the text area, i.e. if it goes over the margins of the text area, then it must be broken up. These occurrences are easy to spot since they give will give an “overfull hbox” warning. Consider the following example:

Example 5.13: An overfull hbox

```
It follows from
\[
  aaaaaaa
  =
  bbbbbbb
  =
  ccccccc
  \leq
  ddddddd
  =
  eeeeeee
  =
  fffffff
  \leq
  ggggggg
  =
  hhhhhhh
\]
that $a \leq h$.
```

It follows from

$$aaaaaaa = bbbbbbb = ccccccc \leq ddddddd = eeeeeee = fffffff \leq ggggggg = hhhhhhh$$

that $a \leq h$.

When the formula is visually too long

Sometimes a formula does fit into a single line, but barely so. Consider the following example:

Example 5.14: Visually overfull hbox

```
Here is some text.
\[
  aaaaaaaaa = bbbbbbbbbbbbbb = ccccccccccccc = dddddddddddddd = eeeeeeeeee
\]
Here is some more text.
```

Here is some text.

$$aaaaaaaaa = bbbbbbbbbbbbbb = ccccccccccccc = dddddddddddddd = eeeeeeeeee$$

Here is some more text.

The formula in the above example does – technically speaking – not go over the margin. But it has stretched the display mode beyond its visual limits. The formula does therefore need to be broken up.

Expressing structure

Often the breaking up of a formula is done to better express the structure – and thus content – of the displayed formula. Consider the following example:

Example 5.15: A formula that should be broken up for readability

```
If $k$ is algebraically closed with~$\operatorname{char}(k) \neq 2$ and $i$ is
a square root of $-1$ then
\[
  k[x]/(x^2 + 1)
  =
  k[x]/( (x - i) (x + i) )
  \cong
  k[x]/(x - i) \times k[x]/(x + i)
  \cong
  k \times k
\]
by the Chinese remainder theorem.
```

If k is algebraically closed with $\operatorname{char}(k) \neq 2$ and i is a square root of -1 then

$$k[x]/(x^2 + 1) = k[x]/((x - i)(x + i)) \cong k[x]/(x - i) \times k[x]/(x + i) \cong k \times k$$

by the Chinese remainder theorem.

The above output still has an appropriate length to be put into a single line, and if space is sparse then this is an acceptable solution. But this single-line approach to the formula does not help to display its internal structure. This can be done by splitting up the formula as done in the next example:

Example 5.16: Broken up version of Example 15

```
If $k$ is algebraically closed and $i$ is a square root of $-1$ then
\begin{align*}
  k[x]/(x^2 + 1)
  &=
  k[x]/( (x - i) (x + i) )
  \\
  &\cong
  k[x]/(x - i) \times k[x]/(x + i)
  \\
  &\cong
  k \times k
\end{align*}
by the Chinese remainder theorem.
```

If k is algebraically closed and i is a square root of -1 then

$$\begin{aligned} k[x]/(x^2+1) &= k[x]/((x-i)(x+i)) \\ &\cong k[x]/(x-i) \times k[x]/(x+i) \\ &\cong k \times k \end{aligned}$$

by the Chinese remainder theorem.

This form makes it clear where the equalities and isomorphisms occur.

5.2.2 Where to break and align a formula

We now discuss at which points a formula can be broken, and how these broken parts can then be aligned.

Aligning at relation symbols I

A first approach is to put all relation symbols underneath each other.

Example 5.17: Aligning relation symbols I

```
Let $x$, $y$ be two commuting, nilpotent elements of~$A$.
Then
\begin{align*}
  \exp(x) \exp(y)
  &=
  \left( \sum_{k=0}^\infty \frac{x^k}{k!} \right)
  \left( \sum_{l=0}^\infty \frac{y^l}{l!} \right)
  \\
  &=
  \sum_{k,l=0}^\infty \frac{x^k y^l}{k! \, l!}
  \\
  &=
  \sum_{n=0}^\infty \, \sum_{k+l = n} \frac{x^k y^l}{k! \, l!}
  \\
  &=
  \sum_{n=0}^\infty \frac{1}{n!} \sum_{k=0}^n \binom{n}{k} x^k y^{n-k}
  \\
  &=
  \sum_{n=0}^\infty \frac{1}{n!} (x + y)^n
  \\
  &=
  \exp(x + y) \,.
\end{align*}
```

Let x, y be two commuting, nilpotent elements of A. Then

$$\begin{aligned}
\exp(x)\exp(y) &= \left(\sum_{k=0}^{\infty} \frac{x^k}{k!}\right)\left(\sum_{l=0}^{\infty} \frac{y^l}{l!}\right) \\
&= \sum_{k,l=0}^{\infty} \frac{x^k y^l}{k!\,l!} \\
&= \sum_{n=0}^{\infty} \sum_{k+l=n} \frac{x^k y^l}{k!\,l!} \\
&= \sum_{n=0}^{\infty} \frac{1}{n!} \sum_{k=0}^{n} \binom{n}{k} x^k y^{n-k} \\
&= \sum_{n=0}^{\infty} \frac{1}{n!} (x+y)^n \\
&= \exp(x+y)\,.
\end{aligned}$$

Aligning at relation symbols II

One can also align all relation symbols to the left, so that the broken up parts of the formula all lie on top of each other.

Example 5.18: Aligning relation symbols II

```
Let~$R$ and~$S$ be two commutative rings.
Then for every~$(r,s) \in R \times S$,
\begin{align*}
  {}&
  (r, s) \in (R \times S)^\times
  \\
  \iff{}&
  \text{there exist $(r', s') \in R \times S$ with $(r,s)(r',s') = (1,1)$}
  \\
  \iff{}&
  \text{there exist $r' \in R$ and $s' \in S$ with $rr' = 1$ and $ss' = 1$}
  \\
  \iff{}&
  \text{$r \in R^\times$ and $s \in S^\times$}
  \\
  \iff{}&
  (r,s) \in R^\times \times S^\times \,,
\end{align*}
```

```
and therefore $(R \times S)^{\times} = R^\times \times S^\times$.
```

Let R and S be two commutative rings. Then for every $(r,s) \in R \times S$,

$$\begin{aligned}
&(r,s) \in (R \times S)^\times \\
\iff{}& \text{there exist } (r',s') \in R \times S \text{ with } (r,s)(r',s') = (1,1) \\
\iff{}& \text{there exist } r' \in R \text{ and } s' \in S \text{ with } rr' = 1 \text{ and } ss' = 1 \\
\iff{}& r \in R^\times \text{ and } s \in S^\times \\
\iff{}& (r,s) \in R^\times \times S^\times \,,
\end{aligned}$$

and therefore $(R \times S)^\times = R^\times \times S^\times$.

The empty pair of curly brackets in `\iff{}&` ensures that the spacings coming from `\iff` and `&` do not interfere with each other. Otherwise something like this happens:

Example 5.19: Wrong spacing when alignment points are set wrong

```
\begin{align*}
   &\text{some stuff} \\
  =&\text{some other stuff}
\end{align*}
```

$$\begin{aligned}
&\text{some stuff} \\
=&\text{some other stuff}
\end{aligned}$$

Breaking at a binary operator

Sometimes it is also useful to break a long term of a formula at a binary operator. In this case this operator needs to occur in the line after the break. The following example does it wrong:

Example 5.20: Wrong aligning at a binary operator I

```
\begin{align*}
  & a + b + c + \\
  & d + e
\end{align*}
```

$$\begin{aligned}
& a + b + c + \\
& d + e
\end{aligned}$$

The following should be done instead:

Example 5.21: Right breaking at a binary operator I

```
\begin{align*}
  & a + b + c \\
  & + d + e
\end{align*}
```

$$\begin{aligned}
& a + b + c \\
& + d + e
\end{aligned}$$

If a formula is broken at relation symbols and one of the resulting terms is broken at a binary operator, then the operator is not aligned together with the relation symbols. Instead the binary operator appears after the relation symbols. The following example does it wrong:

Example 5.22: Wrong breaking at a binary operator II

```
\begin{align*}
  a + a
  &=
  b + b + b + b
  \\
  &+
  b + b + b + b
  \\
  &=
  c + c + c + c + c
\end{align*}
```

$$\begin{aligned} a + a &= b + b + b + b \\ &+ b + b + b + b \\ &= c + c + c + c + c \end{aligned}$$

Instead the following has to be done:

Example 5.23: Right breaking at a binary operator II

```
\begin{align*}
  a + a
  ={}&
  b + b + b + b
  \\
  {}&
  +b + b + b + b
  \\
  ={}&
  c + c + c + c + c
\end{align*}
```

$$\begin{aligned} a + a ={}& b + b + b + b \\ {}& + b + b + b + b \\ ={}& c + c + c + c + c \end{aligned}$$

A single term in another line

If a single line equation is too long, then it is sometimes appropriate to put the last term in a new line, such that the last term occurs below the second to last term.

Example 5.24: Single term in new line

```
It follows that
\begin{align*}
  aaaaaaaaaaaa
  =
  bbbbbbbbbbb
  =
  cccccccccc
  =
  ddddddddd
  &=
  eeeeeeee
  \\
  &=
  ffffff
\end{align*}
and hence $2 + 2 = 5$.
```

It follows that

$$\begin{aligned} aaaaaaaaaaaa = bbbbbbbbbbb = cccccccccc = ddddddddd &= eeeeeeee \\ &= ffffff \end{aligned}$$

and hence $2 + 2 = 5$.

5.2.3 Don’t break formulas badly

One should always keep in mind that breaking up a formula isn’t just meant to prevent technical problems, but more importantly to let the resulting output better display the structure – and thus part of the content – of the formula. A badly broken up formula is harder to understand for both the reader and the author. Consider the following example:

Example 5.25: Badly broken formula

```
It follows that
\begin{align*}
  aaaaaaaaa
  &=
  bbbbbbbbbbb
  =
  cccccccc
  \leq
  dddddd
  =
  eeeeeee
  \\
  &=
  eee
  \leq
  fffffff
  =
  gggggggg
  \leq
  hhhhhhhhh
  \\
  &<
  kkkkkkk
  =
  llll
  \leq
  mmmma
  =
  nnnnnnn
  =
  pp \,.
\end{align*}
```

It follows that

$$\begin{aligned} aaaaaaaaa = bbbbbbbbbbb = cccccccc \leq dddddd = eeeeeee \\ &= eee \leq fffffff = gggggggg \leq hhhhhhhhh \\ &< kkkkkkk = llll \leq mmmma = nnnnnnn = pp\,. \end{aligned}$$

In such a case the breaking of the formula should be done in a consistent way. There seem to be two sensible approaches for the above example, which we will now explain.

Align everything

One can align all occurring relation symbols:

Example 5.26: Aligning all relation symbols

```
It follows that
\begin{align*}
  aaaaaaaaa
  &= bbbbbbbbbbb \\
  &= cccccccc \\
  &\leq dddddd \\
  &= eeeeeee \\
  &= eee \\
  &\leq fffffff \\
  &= gggggggg \\
  &\leq hhhhhhhhh \\
  &< kkkkkkk \\
  &= llll \\
  &\leq mmmma \\
  &= nnnnnnn \\
  &= pp \,.
\end{align*}
```

It follows that

$$\begin{aligned}
aaaaaaaaa &= bbbbbbbbbbb \\
&= cccccccc \\
&\leq dddddd \\
&= eeeeeee \\
&= eee \\
&\leq fffffff \\
&= gggggggg \\
&\leq hhhhhhhhh \\
&< kkkkkkk \\
&= llll \\
&\leq mmmma \\
&= nnnnnnn \\
&= pp \,.
\end{aligned}$$

This approach has the advantage of being very consistent. But it has the disadvantage of taking a lot of space. It might also not reflect the structure of the formula particularly well, as this layout gives all (in)equalities the same importance.

Align at inequalities

On could align all the inequality symbols, to make it clear where these occur:

Example 5.27: Aligning all inequalities

```
It follows that
\begin{align*}
  aaaaaaaaa
  &= bbbbbbbbbbb
  = cccccccc
  \\
  &\leq
  dddddd
  = eeeeeee
  = eee
  \\
  &\leq
  fffffff
  = gggggggg
  \\
  &\leq
  hhhhhhhhh
  \\
  &<
  kkkkkkk
  = llll \\
  &\leq mmmma
  = nnnnnnn
  \\
  &= pp \,.
\end{align*}
```

It follows that

$$\begin{aligned} aaaaaaaaa &= bbbbbbbbbb = cccccccc \\ &\leq dddddd = eeeeeee = eee \\ &\leq fffffff = gggggggg \\ &\leq hhhhhhhhh \\ &< kkkkkkk = llll \\ &\leq mmmma = nnnnnnn \\ &= pp\,. \end{aligned}$$

This layout emphasizes the importance of the inequalities, while relegating the equalities to a less important position. Note that we have also aligned the first and last equality signs to make it clear where the manipulations begin and end. If some other equalities are also particularly important (e.g. if they follows from some previously hard-earned proposition) then they too should be aligned

5.3 Aligning nearly aligned formulas

Sometimes formulas turn out to look nearly aligned in the compiled output, even though this wasn't planned. But the formulas may still be non-aligned enough to look jarring. In such a case it is often best to align these formulas.

Consider the following example:

Example 5.28: Accidental jarringly non-aligned expressions

```
\begin{gather*}
  KK^{-1} = 1 = K^{-1}K \,,
  \quad
  EF - FE = \frac{ K - K^{-1} }{ q - q^{-1} } \,,
  \\
  KE = q^2 EK \,,
  \quad
  KF = q^{-2} FK \,.
\end{gather*}
```

$$KK^{-1} = 1 = K^{-1}K\,, \quad EF - FE = \frac{K - K^{-1}}{q - q^{-1}}\,, \\ KE = q^2 EK\,, \quad KF = q^{-2} FK\,.$$

Note that the first line of the output looks slightly shifted to the left when compared to the second line. This impression vanishes when both lines are properly aligned, and instead gives rise to more coherent look and fell.

Example 5.29: Intentional well-aligned expressions

```
\begin{align*}
  KK^{-1} = 1 = K^{-1}K \,,
  \quad
  &EF - FE = \frac{ K - K^{-1} }{ q - q^{-1} } \,,
  \\
  KE = q^2 EK \,,
  \quad
  &KF = q^{-2} FK \,.
\end{align*}
```

$$\begin{aligned} KK^{-1} = 1 = K^{-1}K\,, \quad &EF - FE = \frac{K - K^{-1}}{q - q^{-1}}\,, \\ KE = q^2 EK\,, \quad &KF = q^{-2} FK\,. \end{aligned}$$

5.4 Proper spacing before multi-line display mode environments

If mathematical content is put into display mode then this content will not only be horizontally centered but also receive some vertical spacing around it to separate it from the surrounding text. Observe in the following example that both above and below the formula there is an additional spacing of roughly (exactly?) one line:

Example 5.30: Vertical space around display mode

```
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et
\[
  a = b \,.
\]
Nunc aliquet bibendum enim facilisis gravida. Nisl nunc mi ipsum faucibus vitae aliquet nec ullamcorper.
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et

$$a = b\,.$$

Nunc aliquet bibendum enim facilisis gravida. Nisl nunc mi ipsum faucibus vitae aliquet nec ullamcorper.

If the line before a display mode environment is too short then this vertical spacing may be visually too large. The one-line display environments `\[ \]` and `equation*` do therefore automatically adjust this spacing. Observe that in the following example the distance between the first two text lines is far shorter than the distance between the second and third text lines.

Example 5.31: Adjusted spacing before one-line display mode

```
Lorem
\[
  a = b
\]
 ipsum dolor sit amet, consectetur
\[
  a = b \,.
\]
ipsum elit.
```

Lorem

$$a = b$$

ipsum dolor sit amet, consectetur

$$a = b\,.$$

ipsum elit.

We can say more precisely that the additional spacing inserted by LaTeX is either `\abovedisplayskip` or its shorter version `\abovedisplayshortskip`.

The multi-line display mode environments (like `align*`) don't have this feature (for technical reasons). Instead, it always uses the spacing `\abovedisplayskip`. This can lead to some unpleasant spacing:

Example 5.32: Unpleasant spacing around a multi-line environment

```
text text text text text text text
text text
\begin{alignat*}{1}
  x &= y_i
\shortintertext{text}
  x &= z_{i+1}
\end{alignat*}
text text text text text text text
text
```

text text text text text text text text
text

$$x = y_i$$

text

$$x = z_{i+1}$$

text text text text text text text text

The package `mathools` provides a partial fix to this problem: By putting the command `\SwapAboveDisplaySkip` at the beginning of a multi-line display mode environment we force LaTeX to use the shorter spacing `\abovedisplayshortskip`.

Example 5.33: Pleasant spacing around a multi-line environment

```
text text text text text text text
text text
\begin{alignat*}{1}
\SwapAboveDisplaySkip
  x &= y_i
\shortintertext{text}
  x &= z_{i+1}
\end{alignat*}
text text text text text text text
text
```

text text text text text text text text
text

$$x = y_i$$

text

$$x = z_{i+1}$$

text text text text text text text text

5.5 Use the `cases` environment

Use the environment `cases` for case distinctions:

Example 5.34: Using `cases`

```
It follows that
\[
  A(x)
  =
  \begin{cases}
    x^2  & \text{if $x \leq 0$,} \\
    3x   & \text{if $x = 0$.}
  \end{cases}
\]
```

It follows that
$$A(x) = \begin{cases} x^2 & \text{if } x \leq 0, \\ 3x & \text{if } x = 0. \end{cases}$$

In most cases one should actually use the environment `cases*`, which ensures that the second column will be treated as text.

Example 5.35: Using `cases*`

```
It follows that
\[
  A(x)
  =
  \begin{cases*}
    x^2  & if $x \leq 0$, \\
    3x   & if $x = 0$.
  \end{cases*}
\]
```

It follows that
$$A(x) = \begin{cases} x^2 & \text{if } x \leq 0, \\ 3x & \text{if } x = 0. \end{cases}$$

The package `mathtools` defines some more useful variants of the environment `cases`. We refer to [CTN19b, 3.4.3] for more information on this.

5.6 Proper placement of the qed-symbol

The `proof` environment automatically places a qed-symbol at its end.

Example 5.36: Using the `proof` environment

```
\begin{proof}
This is obviously a proof.
\end{proof}
```

Proof. This is obviously a proof. □

It can happen that this automatic placement of the qed-symbol gives a bad looking result. The general rule is that the qed-symbol should not occur at the end of an otherwise empty line. To fix such a bad placement one can use the command `\qedhere`. Let’s look at some specific examples of this problem:

qed-symbol after lists

If a proof consists of a list then the qed-symbol will by default be placed after the list, and thus in a new line.

Example 5.37: Improper placement of the qed-symbol after a list I

```
\begin{proof}
  This proof consists of a list.
  \begin{enumerate}
    \item
      Some part of the proof.
    \item
      Another part of the proof.
  \end{enumerate}
\end{proof}
```

Proof. This proof consists of a list.

1. Some part of the proof.
2. Another part of the proof.

□

This can be fixed by placing the command `\qedhere` just before the list ends.

Example 5.38: Proper placement of the qed-symbol after a list

```
\begin{proof}
  This proof consists of a list.
  \begin{enumerate}
    \item
      Some part of the proof.
    \item
      Another part of the proof.
    \qedhere
  \end{enumerate}
\end{proof}
```

Proof. This proof consists of a list.

1. Some part of the proof.
2. Another part of the proof. □

But one has to be careful no to introduce a new line before the command `\qedhere`.

Example 5.39: Improper placement of the qed-symbol after a list II

```
\begin{proof}
  This proof consists of a list.
  \begin{enumerate}
    \item
      Some part of the proof.
    \item
      Another part of the proof.

    \qedhere
  \end{enumerate}
\end{proof}
```

Proof. This proof consists of a list.

1. Some part of the proof.
2. Another part of the proof.

□

qed-symbol after displaystyle

If a proof ends with display style mathematics then the qed-symbol will by default be placed after this display mode.

Example 5.40: Improper placement of the qed-symbol after display mode I

```
\begin{proof}
  This proof ends with a some display~mode mathematics, namely
  \[
    \sin(x+y)
    =
    \cos(x)\sin(y) + \sin(x)\cos(y) \,.
  \]
\end{proof}
```

Proof. This proof ends with a some display mode mathematics, namely

$$\sin(x+y) = \cos(x)\sin(y) + \sin(x)\cos(y)\,.$$

□

There are at least two different approaches to fixing this situation:

Some authors prever to bring the qed-symbol to the height of the display environment, as done in the following example.

Example 5.41: Somewhat proper placement of the qed-symbol after display mode

```
\begin{proof}
  This proof ends with a some display~mode mathematics, namely
  \[
    \sin(x+y)
    =
    \cos(x)\sin(y) + \sin(x)\cos(y) \,.
    \qedhere
  \]
\end{proof}
```

Proof. This proof ends with a some display mode mathematics, namely

$$\sin(x+y) = \cos(x)\sin(y) + \sin(x)\cos(y)\,. \qquad \square$$

But this approach does not work if the (last line) of the display environment has additional height. Then the mathematical formula goes below the line on which the qed-symbol rests.

Example 5.42: Improper placement of the qed-symbol after display mode II

```
\begin{proof}
  This proof ends with a some display~mode mathematics, namely
  \[
    1
    + \frac{a}{b - a}
    =
    \frac{(b - a) + a}{b - a}
    =
    \frac{b}{b - a} \,.
    \qedhere
  \]
\end{proof}
```

Proof. This proof ends with a some display mode mathematics, namely

$$1 + \frac{a}{b-a} = \frac{(b-a)+a}{b-a} = \frac{b}{b-a}\,. \qquad \square$$

The author is the opinion that in such a cases there is no good placement for the qed-symbol.

Aside the placement of the qed-symbol there is another problem to ending a proof with a display environment: One of the functions of display mode is to put an emphasis on the displayed mathematical content. Putting such an emphasis directly before the end of a proof can lead to the proof missing a sense of closure.

This problem leads to the second solution for the placement of the qed-symbol. Never end a proof with a display environment. This can be done by rewriting the end of proof, or by adding a closing sentence to it.

qed-symbol at the end of a long text

It can also happen that the qed-symbol is pushed to a new line if the previous line is completely filled with text. (Although LaTeX will actually try quite hard to prevent this from happening.) If this happens then one should (slightly) rewrite the text to circumvent this problem.

5.7 Tagging and numbering

A finer control over tags can be achieved via the commands `\tag` and `\notag`.

5.7.1 Don't autonumber all formulas

Don't indiscriminately number every occurring formula. Instead, an equation should be numbered only if it will be referred to later on. This numbering should then be done automatically by using a suitable environment like `equation`, `gather`, `align` or `alignat`.

5.7.2 `\tag`

With the command `\tag` a custom tag can be set. This is useful for marking selected equations by special symbols:

Example 5.43: Using `\tag` for marking a line

```
Consider the equation
\begin{equation}
\label{important equation}
  2 + 2 = 5 \,.
  \tag{\ast}
\end{equation}
Note that \cref{important equation} can equivalently be expressed as~$5 = 2 + 2$.
```

Consider the equation

$$2 + 2 = 5\,. \tag{$\ast$}$$

Note that equation ($\ast$) can equivalently be expressed as $5 = 2 + 2$.

The argument of `\tag` is in text mode, and the resulting tag is automatically enclosed in parentheses. These parentheses can be removed by using the starred command `\tag*` instead.

The command `\tag` should not be used for regular numbering of equations. It should be used to tag only certain (often a single) equations in a special way.

It can also be used to express that certain transformations have been used, as the following example demonstrates:

Example 5.44: Using `\tag` to explain steps

```
It follows from the Chinese remainder theorem that
\begin{align*}
  \mathbb{R}[x] / ( x^3 + x^2 + x + 1 )
  &=
  \mathbb{R}[x] / ( (x^2 + 1) (x + 1) )
  \\
  &\cong
  \mathbb{R}[x] / ( x^2 + 1 ) \times \mathbb{R}[x] / ( x + 1 )
  \tag{CRT}
  \\
  &\cong
  \mathbb{C} \times \mathbb{R}
\end{align*}
```

It follows from the Chinese remainder theorem that

$$\begin{aligned}
\mathbb{R}[x]/(x^3+x^2+x+1) &= \mathbb{R}[x]/((x^2+1)(x+1)) \\
&\cong \mathbb{R}[x]/(x^2+1) \times \mathbb{R}[x]/(x+1) \qquad \text{(CRT)} \\
&\cong \mathbb{C} \times \mathbb{R}
\end{aligned}$$

5.7.3 `\notag`

According to Section 5.7.1 a formula should be numbered only if it needs to be referred to. But if this formula occurs in a multi-line environment like `align` then all occuring lines will be numbered, even unwanted ones. The prevent the numbering of the unrequired lines the command `\notag` can then be used:

Example 5.45: Using `\notag` to prevent selected line numbers

```
\begin{align}
  a
  &= b \notag \\
  &= c \\
  &= d \notag \\
  &= e
\end{align}
```

$$\begin{aligned} a &= b \\ &= c \qquad (5.7) \\ &= d \\ &= e \qquad (5.8) \end{aligned}$$

5.8 Multi-line set descriptions

Multi-line set descriptions of the form

$$\left\{ (e_1, \dots, e_n) \,\middle|\, \begin{tabular}{@{}c@{}} $e_1, \dots, e_n \in R$ \\ is a complete set of \\ pairwise orthogonal \\ idempotents \end{tabular} \right\}$$

can be typeset by using a `tabular` environment for the right hand side of the set description:

Example 5.46: Multi-line set descriptions with tabular

```
\[
  \left\{
    x \in X
  \,\middle|\,
    \begin{tabular}{@{}c@{}}
      $x$ satisfies \\
      certain conditions
    \end{tabular}
  \right\}
\]
```

$$\left\{ x \in X \,\middle|\, \begin{tabular}{@{}c@{}} x satisfies \\ certain conditions \end{tabular} \right\}$$

Note that the entries of the environment `tabular` are automatically in text mode. The argument `{}` ensure that the environment `tabular` does not insert additional spacing to its left and right.

Bibliography

[Bel18] Pieter Belmans. *My good practice concerning inline mathematical formulas.* December 18, 2018. URL: https://pbelmans.ncag.info/blog/2010/12/18/my-good-practice-concerning-inline-mathematical-formulas (visited on September 4, 2019).

[CMS17] *The Chicago Manual of Style. The Essential Guide for Writers, Editors and Publishers.* 17th ed. The University of Chicago Press, 2017. xvii+1144 pp. ISBN: 978-0-226-28705-8. DOI: 10.7208/cmos17.

[CTN16] Simon Fear. *Publication quality tables in LaTeX.* Version 1.618033. April 29, 2016. 17 pp. URL: https://www.ctan.org/pkg/booktabs (visited on September 1, 2019).

[CTN18] Augusto Stoffel. *{tikzcd}. Commutative diagrams with TikZ.* Version 0.9f. November 19, 2018. 17 pp. URL: https://ctan.org/pkg/tikz-cd (visited on September 4, 2019).

[CTN19a] Javier Bezos. *Customizing lists with the enumitem package.* Version 3.9. June 20, 2019. 23 pp. URL: https://ctan.org/pkg/enumitem (visited on September 14, 2019).

[CTN19b] Morten Høgholm and Lars Madsen. *The `mathtools` package.* Version 1.22. July 31, 2019. 37 pp. URL: https://ctan.org/pkg/mathtools (visited on September 12, 2019).

[CTN19c] Philip Kime, Moritz Wemheuer, and Philipp Lehman. *The `biblatex` Package. Programmable Bibliographies and Citations.* Version 3.13. August 17, 2019. 336 pp. URL: https://ctan.org/pkg/biblatex (visited on September 6, 2019).

[Kot15] Stefan Kottwitz. *Example: Flowchart.* August 29, 2015. URL: http://www.texample.net/tikz/examples/math-flowchart/ (visited on August 27, 2019).

[L2M19] *LaTeX2e. An unofficial reference manual.* June 2019. ix + 262 pp. URL: https://latexref.xyz (visited on September 4, 2019).

[TSE11] Nobby. *What is the difference between \mathbin vs. \mathrel?* December 21, 2011. URL: https://tex.stackexchange.com/questions/38982/what-is-the-difference-between-mathbin-vs-mathrel (visited on August 19, 2019).

[TSE14] Lupino. *For formal articles, should a displayed equation be followed by a punctuation to conform to the language grammar ?* April 10, 2014. URL: https://tex.stackexchange.com/a/170691/108717 (visited on August 18, 2019).

List of examples

1.1 Code is too cramped . . . 2
1.2 Code is less cramped . . . 2
1.3 Code isn't cramped . . . 2
1.4 Unrelated symbols need to be separated . . . 2
1.5 Unrelated symbols are separated . . . 3
1.6 Unrelated symbols are separated even better . . . 3
1.7 Wrong spacing before a footnote . . . 3
1.8 Right spacing before a footnote . . . 4
1.9 Layout of `main.tex` . . . 5
1.10 Layout of `mystyle.sty` . . . 5
1.11 Layout of `chapter1.tex` . . . 6
1.12 Layout of `section1.tex` . . . 6
1.13 Basic example of `main.tex` . . . 7
1.14 Using the command `\includeonly` . . . 7
1.15 Syntax of `\includeonly` . . . 8
1.16 Using the option `chapterprefix` . . . 8
1.17 Changing KOMA fonts . . . 9
1.18 Using the option `egregdoesnotlikesansseriftitles` . . . 9
1.19 Using the environment `abstract` . . . 9
1.20 Using the option `titlepage` . . . 9
1.21 Using the option `oneside` . . . 10
1.22 Using the option `openany` . . . 10
1.23 General document layout with `scrbook` . . . 11
1.24 Inconsistency looks bad . . . 12

2.1 Syntax of `\newtheorem` . . . 13
2.2 Using the command `\newtheorem` . . . 13
2.3 Using the command `\newtheorem*` . . . 14
2.4 Theorem-like environment use different counters by default . . . 14
2.5 Syntax of `\newcounter` . . . 15
2.6 Setting up a common counter . . . 15
2.7 Binding a new counter to the section level . . . 15
2.8 Syntax of `\label` . . . 17
2.9 Syntax of `\ref` . . . 17
2.10 Using the commands `\label` and `\ref` . . . 17
2.11 Specifying the kind of reference . . . 18
2.12 Syntax of `\eqref` . . . 18
2.13 Using the command `\eqref` . . . 18

2.14 Loading the package `cleveref` . . . 19
2.15 Syntax of `\cref` . . . 19
2.16 Using the command `\cref` . . . 19
2.17 Using `\lcnamecref` . . . 20
2.18 Order of preamble with `cleveref` and `hyperref` . . . 21
2.19 Syntax of `\enquote` . . . 22
2.20 Using the command `\enquote` . . . 22
2.21 `\enquote` chooses the right kind of quotation marks . . . 22
2.22 Nested quotes with `\enquote` . . . 23
2.23 Using the environment `enumerate` . . . 23
2.24 Using the environment `itemize` . . . 23
2.25 Using the environment `description` . . . 24
2.26 Changing the numbering style of `enumerate` lists . . . 24
2.27 Changing the symbol for `itemize` lists . . . 25
2.28 Resuming lists with `resume*` . . . 25
2.29 Indentation and distance of pararaphs in lists, default behavior . . . 27
2.30 Indentation and distance of pararaphs in lists, changed behavior . . . 27
2.31 Distance between list items with `itemsep` . . . 27
2.32 `enumerate` environment with standard indentation . . . 28
2.33 `enumerate` environment without indentation . . . 29
2.34 `enumerate` environments with only identitation between label and text 29
2.35 `enumerate` environment with indentation between margin and label . . 29
2.36 `enumerate` environment with indentation between margin and label, and label and text . . . 30
2.37 Proper vs. improper placement of long labels . . . 30
2.38 `description` environment with standard indentation . . . 31
2.39 `description` environment with no indentation . . . 31
2.40 Syntax of `\setlist` . . . 31
2.41 Global settings for list environments . . . 32
2.42 Different settings for nested lists . . . 32
2.43 Global settings depending on level . . . 33
2.44 Accessing level counters in settings for list environments . . . 33
2.45 Syntax of `\newlist` . . . 34
2.46 Custom clones of list environments . . . 34
2.47 Loading the package `biblatex` . . . 36
2.48 Syntax for an entry of the `bib`-file . . . 36
2.49 `bib`-file entry for a single book . . . 37
2.50 Adjusting the formatting of `pagestotal` . . . 38
2.51 `bib`-file entry for a book with multiple volumes . . . 39
2.52 `bib`-file entry for a single book . . . 40
2.53 `bib`-file entry for an online resource . . . 41
2.54 `bib`-file entry for an online resource . . . 41
2.55 Syntax of `\cite` . . . 42
2.56 Using the command `\cite` . . . 42
2.57 Syntax of `\bibliography` . . . 43

2.58 Using the command `\printbibliography` . . . 43
2.59 Compiling in the console (with bibliography) . . . 44
2.60 A basic table . . . 44
2.61 Full horizontal lines in tables . . . 45
2.62 Syntax of `\cline` . . . 45
2.63 Partial horizontal lines in tables . . . 45
2.64 Vertical lines in tables . . . 46
2.65 A table with all kinds of lines in it . . . 46
2.66 Using the rules of `booktabs` . . . 47

3.1 Very wrong punctuation in an equation . . . 49
3.2 Wrong punctuation in an equation I . . . 50
3.3 Wrong punctuation in an equation II . . . 50
3.4 Right punctuation in an equation . . . 50
3.5 Best punctuation in an equation . . . 50
3.6 Finishing the sentence before the commutative diagram . . . 51
3.7 Incorporating the commutative diagram in the sentence . . . 51
3.8 Wrong punctuation in lists . . . 52
3.9 Right punctuation in lists I . . . 53
3.10 Right punctuation in lists II . . . 53
3.11 An accidental hyphen in math mode . . . 55
3.12 Proper placement of the hyphen . . . 55
3.13 Using `\frenchspacing` . . . 56
3.14 Using `\nofrenchspacing` . . . 56
3.15 Proper spacing after periods . . . 56
3.16 Proper spacing after periods, and using `~` . . . 57
3.17 Spacings after periods followed by parentheses or quotation marks . . . 57
3.18 Upper case mathematics followed by a period . . . 57
3.19 Spacing of `~` and "`\ `" . . . 58
3.20 An unwanted line break at a space . . . 59
3.21 Using `~` to prevent unwanted line breaks at spaces . . . 59
3.22 An unwanted line break at a hyphen . . . 59
3.23 Using `\nobreakdash` to prevent unwanted line breaks at hyphens . . . 60
3.24 New paragraphs done right . . . 60
3.25 New "paragraphs" done wrong . . . 61
3.26 A list without surrounding text . . . 61
3.27 A list without surrounding text . . . 62
3.28 Improper aligned of list items . . . 63
3.29 Proper aligned of list items with `\leavevmode` . . . 63
3.30 Proper aligned of list items by introducing text . . . 64
3.31 Proper aligned of list items by introducing text . . . 64
3.32 Line break in inline math . . . 65
3.33 Preventing inline math in inline math . . . 65
3.34 Mathematics at the beginning of a sentence . . . 66
3.35 No mathematics at the beginning of a sentence . . . 66

3.36 List items beginning with mathematics . . . 66
3.37 Not properly separating mathematics . . . 67
3.38 Properly separating mathematics I . . . 67
3.39 Properly separating mathematics II . . . 67
3.40 Syntax of `\text` . . . 68
3.41 Missing text in math mode . . . 68
3.42 Wrong way of using text in math mode . . . 68
3.43 Right way of using text in math mode . . . 68
3.44 Wrong spacing around text in math mode I . . . 69
3.45 Wrong spacing around text in math mode II . . . 70
3.46 Right spacing around text in math mode I . . . 70
3.47 Right spacing around text in math mode II . . . 70
3.48 Two non-aligned blocks . . . 71
3.49 Two aligned blocks . . . 72
3.50 Two non-aligned equalities . . . 73
3.51 Two aligned equalities with too much space . . . 73
3.52 Two aligned equalities with proper space . . . 73
3.53 Right way of using mathematical symbols as words . . . 74
3.54 Wrong way of using mathematical symbols as words . . . 74
3.55 Abuse of quantifiers . . . 74
3.56 Writing out quantifiers . . . 74
3.57 Best use of quantifiers . . . 75
3.58 Okay way to use logical symbols . . . 75

4.1 Using `\not` . . . 77
4.2 Why not to use `\not` . . . 77
4.3 Using the command `\centernot` . . . 78
4.4 Using `\limits` . . . 80
4.5 Using built-in limits . . . 80
4.6 Inline math vs. display math with built-in limits . . . 80
4.7 Inline math with `\limits` . . . 81
4.8 Inline with built-in limits . . . 81
4.9 Inline vs. display for different commands with built-in limits . . . 81
4.10 Using `\DeclareMathOperator*` to define `\colim` . . . 82
4.11 Using `\colim` . . . 82
4.12 Using `\overset` and `\underset` to put text above or below an arrow . . 82
4.13 Using `\xrightarrow` to put text above or below an arrow . . . 83
4.14 Defining arrow commands as shortcuts . . . 83
4.15 Typical usage of the ellipses `\cdots`, `\vdots`, `\ddots` . . . 85
4.16 Using the ellipsis `\ldots` . . . 86
4.17 Inconsistent results with `\dots` . . . 86
4.18 The basic matrix environment `matrix` . . . 86
4.19 Aligning matrix entries . . . 87
4.20 Using the right kind of matrix . . . 88
4.21 Wrong kind of differentials . . . 89

4.22 Right kind of differential . . . 89
4.23 Full fractions in exponent, index and inline . . . 89
4.24 Flat fractions in exponent, index and inline . . . 90
4.25 Using commands vs. not using them . . . 90
4.26 Syntax of `\DeclareMathOperator` . . . 90
4.27 Declaring a mathematical operator with `\DeclareMathOperator` . . . 90
4.28 Using a declared mathematical operator . . . 91
4.29 Automatic spacing of `\DeclareMathOperator` . . . 91
4.30 Wrong way of using `\DeclareMathOperator` . . . 91
4.31 Wrong output when `\DeclareMathOperator` is abused . . . 91
4.32 Using `\operatorname` . . . 92
4.33 Missing spacing after `\mathrm` . . . 92
4.34 Syntax of `\newcommand` . . . 92
4.35 Using the command `\newcommand` . . . 92
4.36 `\DeclareMathOperator` = `\newcommand` plus `\operatorname` . . . 93
4.37 Syntax of `\DeclarePairedDelimiter` . . . 93
4.38 Defining the command `\abs` with `\DeclarePairedDelimiter` . . . 93
4.39 Using a declared delimiter . . . 93
4.40 Scaling of declared delimiters . . . 94
4.41 Syntax of `\DeclarePairedDelimiterX` . . . 94
4.42 Using `\DeclarePairedDelimiterX` for more advanced delimiters . . . 94
4.43 Using more advanced delimiters . . . 95
4.44 Syntax of `\NewDocumentCommand` . . . 95
4.45 Mandatory arguments with `\NewDocumentCommand` II . . . 95
4.46 Mandatory arguments with `\NewDocumentCommand` II . . . 95
4.47 Syntax of `\IfValueTF` . . . 96
4.48 Optional arguments with `\NewDocumentCommand`, I . . . 96
4.49 Optional arguments with `\NewDocumentCommand`, I . . . 96
4.50 Optional arguments with `\NewDocumentCommand`, I . . . 97
4.51 Optional arguments with `\NewDocumentCommand`, II . . . 97
4.52 Optional arguments with `\NewDocumentCommand`, III . . . 98
4.53 Starred versions with `\NewDocumentCommand` . . . 98
4.54 Commutative diagram with an arrow too short . . . 98
4.55 Using `column sep` . . . 99
4.56 Explicit column spacing . . . 100
4.57 Using `row sep` and explicit row spacing . . . 101
4.58 Standard spacing around relation symbols and binary operators . . . 102
4.59 Disabling the standard spacing around a symbol . . . 102
4.60 Clashing spacings around symbols . . . 103
4.61 Preventing a clash of spacings . . . 103
4.62 Specifying the role (and thus spacing) of a symbol . . . 103
4.63 Defining and using `\divides` . . . 103

5.1 Using the environments `\[ \]` and `equation*` . . . 104
5.2 Using the environment `equation` . . . 104

5.3 Using the environment `gather*` . . . 105
5.4 Using the environment `gather` . . . 105
5.5 Using the environment `align*` . . . 106
5.6 Using the environment `align` . . . 107
5.7 Using `align*` with multiple columns . . . 107
5.8 Using the environment `alignat*` . . . 107
5.9 Using `\center` with `$ $` . . . 108
5.10 Using `aligned` . . . 108
5.11 Improper use of `align*` . . . 109
5.12 Using `gather*` for non-aligned equations . . . 109
5.13 An overfull hbox . . . 110
5.14 Visually overfull hbox . . . 111
5.15 A formula that should be broken up for readability . . . 112
5.16 Broken up version of Example 15 . . . 112
5.17 Aligning relation symbols I . . . 113
5.18 Aligning relation symbols II . . . 114
5.19 Wrong spacing when alignment points are set wrong . . . 115
5.20 Wrong aligning at a binary operator I . . . 115
5.21 Right breaking at a binary operator I . . . 115
5.22 Wrong breaking at a binary operator II . . . 116
5.23 Right breaking at a binary operator II . . . 116
5.24 Single term in new line . . . 116
5.25 Badly broken formula . . . 117
5.26 Aligning all relation symbols . . . 118
5.27 Aligning all inequalities . . . 119
5.28 Accidental jarringly non-aligned expressions . . . 121
5.29 Intentional well-aligned expressions . . . 121
5.30 Vertical space around display mode . . . 122
5.31 Adjusted spacing before one-line display mode . . . 122
5.32 Unpleasant spacing around a multi-line environment . . . 123
5.33 Pleasant spacing around a multi-line environment . . . 123
5.34 Using `cases` . . . 123
5.35 Using `cases*` . . . 124
5.36 Using the `proof` environment . . . 124
5.37 Improper placement of the qed-symbol after a list I . . . 125
5.38 Proper placement of the qed-symbol after a list . . . 125
5.39 Improper placement of the qed-symbol after a list II . . . 126
5.40 Improper placement of the qed-symbol after display mode I . . . 126
5.41 Somewhat proper placement of the qed-symbol after display mode . . . 127
5.42 Improper placement of the qed-symbol after display mode II . . . 127
5.43 Using `\tag` for marking a line . . . 128
5.44 Using `\tag` to explain steps . . . 129
5.45 Using `\notag` to prevent selected line numbers . . . 130
5.46 Multi-line set descriptions with tabular . . . 130

Index

\(space), 56
\@, 56
$ $, 108
$$ $$, 104
%, 3
&, 106, 107
~, 59
\|, 78
\\, 60
\[\], 104, 122

abstract, 9
\addtokomafont, 8
align, 106
align*, 71, 106, 122
alignat, 108
alignat*, 107
aligned, 108
alignedat, 108
aligning formulas, 71, 113–117, 120–121
    see also \align*
    see also \alignat*
Alph*, 24
alph*, 24
amsmath, 13, 83, 86, 104
amssymb, 13
amsthm, 13
    \newtheorem, 13
    proof, 63, 124
\appendix, 10
Arabic numeral, 24
Arabic numerals, 10, 38
arabic*, 24
arrows
    extensible, 83
    \Leftarrow, 78
    \Leftrightarrow, 78
    \notrightarrow, 79
    \Rightarrow, 78
    \rightarrow, 77
    \to, 77
    \xhookleftarrow, 84
    \xhookrightarrow, 84
    \xLeftarrow, 84
    \xleftarrow, 84
    \xleftharpoondown, 84
    \xleftharpoonup, 84
    \xLeftrightarrow, 84
    \xleftrightarrow, 84
    \xleftrightharpoons, 84
    \xlongequal, 84
    \xLongleftarrow, 84
    \xlongleftarrow, 84
    \xlongleftrightarrow, 84
    \xLongrightarrow, 84
    \xlongrightarrow, 84
    \xmapsto, 84
    \xRightarrow, 84
    \xrightarrow, 83, 84
    \xrightharpoondown, 84
    \xrightharpoonup, 84
    \xrightleftharpoons, 84
@article, 40
article, 8
arXiv, 41
author, 37

babel, 22
babel, 22
backend, 35
\backepsilon, 78
\backmatter, 10

\backslash, 78
before a footnote, 3
bib-file, 35–41
biber, 35, 44
biblatex, 35
    \cite, 42
    keys
        author, 37
        crossref, 40
        date, 38, 41
        doi, 38
        edition, 38
        isbn, 38
        issn, 40
        journaltitle, 40
        number, 38, 40
        pages, 40
        pagetotal, 38
        publisher, 38
        series, 38
        subtitle, 37
        title, 37
        url, 41
        urldate, 41
        volume, 39, 40
        volumes, 39
        year, 38
    options
        backend, 35
        dateabbrev, 36
        style, 36
        urldate, 36
    \printbibliography, 43
    types
        @article, 40
        @book, 37
        @mvbook, 39
        @online, 41
bibliography, 11, 35–44
    arXiv, 41
    back end, 35
    bib-file, 35–41
    biber, 35, 44
    biblatex, 35
    \bibliography, 42
    bibliography, 11, 43
    \cite, 42
    MathSciNet, 37
    \printbibliography, 43
\bibliography, 42
bibliography, 11, 43
\big, 94
\bigg, 94
\binoppenality, 65
Bmatrix, 87
bmatrix, 87
@book, 37
book, 8
booktabs, 47
    \bottomrule, 47
    \cmidrule, 47
    \midrule, 47
    \toprule, 47
\bottomrule, 47
Bsmallmatrix, 87
bsmallmatrix, 87

c, 45, 87
capitalise, 19
cases, 123
cases*, 124
\cdots, 85
center, 108
\centernot, 78
centernot, 78
\chapter, 9
chapterprefix, 8
\cite, 42
cleveref, 18, 19
    babel, 22
    capitalise, 19
    \Cref, 20
    \cref, 19
    \lcnamecref, 20
    \lcnamecrefs, 20
    \nameCref, 20
    \namecref, 20
    \nameCrefs, 20
    \namecrefs, 20
    noabbrev, 19

\cline, 45
\cmidrule, 47
\colon, 78
\coloneqq, 78
column sep, 99
commutative diagrams, 16, 50–52
consistency, 12
corollary, 13
counter, 14, 15
\Cref, 20
\cref, 19
crossref, 40
csquotes, 21
  \enquote, 21

date, 38, 41
dateabbrev, 36
\ddots, 85
\DeclareMathOperator, 82, 90
\DeclareMathOperator*, 82
\DeclarePairedDelimiter, 93
\DeclarePairedDelimiterX, 94
\DeclarePairedDelimiterXPP, 95
defining commands
  \DeclareMathOperator, 82, 90
  \DeclareMathOperator*, 82
  \DeclarePairedDelimiter, 93
  \DeclarePairedDelimiterX, 94
  \DeclarePairedDelimiterXPP, 95
  \newcommand, 92
  \NewDocumentCommand, 95
  \operatorname, 82, 91
  \operatorname*, 82
  \renewcommand, 93
delimiters
  \|, 78
  \langle, 78
  \lVert, 78
  \rangle, 78
  \rVert, 78
  see also \DeclarePairedDelimiter
\delimsize, 94
description, 23, 30
differential, 89
display math environment
  $$ $$, 104
  \[\], 104, 122
  align, 106
  align*, 71, 106, 122
  alignat, 108
  alignat*, 107
  eqnarray, 104
  equation, 104
  equation*, 104, 122
  gather, 105
  gather*, 71, 105
document class, 9
  classes
    article, 8
    book, 8
    report, 8
    scrartcl, 8, 9
    scrbook, 8, 9
    scrreprt, 8, 9
  options
    abstract, 9
    bibliography, 11, 43
    chapterprefix, 8
    egregdoesnotlikesansserif-titles, 9
    oneside, 10
    openany, 10
    titlepage, 9
doi, 38
dots
  see ellipsis
\dots, 86
\dotsb, 85
\dotsc, 85
\dotsi, 85
\dotsm, 85
\dotso, 85

edition, 38
egregdoesnotlikesansseriftitles, 9
ellipses, 84–86
  \cdots, 85
  \ddots, 85
  \dots, 86
  \dotsb, 85

    \dotsc, 85
    \dotsi, 85
    \dotsm, 85
    \dotso, 85
    \ldots, 85
    \vdots, 85
\emptyset, 78
em dash, 53, 60
\enquote, 21
enumerate, 23, 28
    see also enumitem
enumi, 33
enumii, 33
enumitem, 24
    Alph*, 24
    alph*, 24
    arabic*, 24
    enumi, 33
    enumii, 33
    itemsep, 27
    label, 24
    leftmargin, 29
    listparindent, 27
    \newlist, 34
    parsep, 27
    resume, 25
    resume*, 25
    Roman*, 24
    roman*, 24
    setlist, 31
    wide, 28, 31
en dash, 53, 60
\epsilon, 78
eqnarray, 104
\eqqcolon, 78
\eqref, 18
equation, 104
equation*, 104, 122
error, 1, 3
extarrows, 13, 83
extensible arrows, 83

Facebook, 1
file inclusion
    \include, 4
    \includeonly, 6
    \input, 4
    \usepackage, 4
fonts
    \addtokomafont, 8
footnote, 3
fractions, 89–90
\frenchspacing, 56
\frontmatter, 10

gather, 105
gather*, 71, 105
gathered, 108
Git, 1

\hline, 45
huge, 99, 101
hyperlink, 21
hyperref, 21
hyphen, 53, 54, 60

\IfBooleanTF, 98
iff, 76
\iff, 78
\impliedby, 78
\implies, 78
\in, 78
\include, 4
\includeonly, 6
including files
    see file inclusion
indent, 28
indentation
    for list environments, 28–31
    in source code, 3
    of paragraphs, 26, 60
\infty, 78
\input, 4
integral, 85, 89
\intertext, 71
introduction, 10
isbn, 38
issn, 40
itemize, 23, 30
itemsep, 27

journal, 40
journaltitle, 40

kile, 44
KOMA-Script classes
  \addtokomafont, 8
  bibliography, 11, 43
  chapterprefix, 8
  egregdoesnotlikesansseriftitles, 9
  oneside, 10
  openany, 10
  scrartcl, 8, 9
  scrbook, 8, 9
  scrreprt, 8, 9
  titlepage, 9
  totoc, 11

l, 45, 87
\label, 17
label, 24
\langle, 78
large, 99, 101
\lcnamecref, 20
\lcnamecrefs, 20
\ldots, 85
\leavevmode, 63
\Leftarrow, 78
leftmargin, 29
\Leftrightarrow, 78
lemma, 13
limits
  \DeclareMathOperator*, 82
  \limits, 80
  \operatorname*, 82
\limits, 80
line breaks
  in math mode, 110–120
  in text mode, 59, 60, 65
list environments
  description, 23
  enumerate, 23
  itemize, 23
listparindent, 27
lists, 52, 66
logical symbols, 74
LuaLaTeX, 1
\lVert, 78

magic
  see microtype
\mainmatter, 10
\mathrm, 92
MathSciNet, 37
mathtools, 13, 83, 86, 87, 124
  \SwapAboveDisplaySkip, 123
matrices, 86–88
  normal sized
    Bmatrix, 87
    bmatrix, 87
    matrix, 86, 87
    pmatrix, 87
    Vmatrix, 87
    vmatrix, 87
  small sized
    Bsmallmatrix, 87
    bsmallmatrix, 87
    psmallmatrix, 87
    smallmatrix, 87
    Vsmallmatrix, 87
    vsmallmatrix, 87
matrix, 86, 87
\maxdimen, 65
Mercurial, 1
microtype, 1, 13
\midrule, 47
minus sign, 53, 78
multi-line mathematics
  align, 106
  align*, 71, 106, 122
  alignat, 108
  alignat*, 107
  aligned, 108
  alignedat, 108
  cases, 123
  cases*, 124
  gather, 105
  gather*, 71, 105
  gathered, 108
@mvbook, 39

\nameCref, 20
\namecref, 20
\nameCrefs, 20
\namecrefs, 20
negation
  \centernot, 78
  \neq, 79
  \nexists, 79
  \nleq, 79
  \not, 77
  \notin, 79
  \notrightarrow, 79
\neq, 79
\newcommand, 92
\newcounter, 15
\NewDocumentCommand, 95
  \IfBooleanTF, 98
  mandatory argument, 95
  optional argument
    with default value, 97
    without default value, 96
\newline, 60
\newlist, 34
\newtheorem, 13
\newtheorem*, 14
\nexists, 79
\ni, 78
\nleq, 79
noabbrev, 19
\nofrenchspacing, 56
normal, 99, 101
\not, 77
\notin, 79
\notrightarrow, 79
\nsubseteq, 88
ntheorem, 64
number, 38, 40
numerals
  Arabic, 10, 24, 38
  Roman, 24, 38
  roman, 10

o, 96
O{ }, 97
oneside, 10
@online, 41
openany, 10
\operatorname, 82, 91
\operatorname*, 82
orthography, 49
\overset, 82

packages
  amsmath, 13, 83, 86, 104
  amssymb, 13
  amsthm, 13
  babel, 22
  biblatex, 35
  booktabs, 47
  centernot, 78
  cleveref, 18, 19
  csquotes, 21
  enumitem, 24
  extarrows, 13, 83
  hyperref, 21
  mathtools, 13, 83, 86, 87, 124
  microtype, 13
  ntheorem, 64
  stmaryrd, 13
  tikz-cd, 16
  xparse, 95
pages, 40
pagetotal, 38
paragraphs, 26, 60
parsep, 27
\part, 10
penalties
  \binoppenality, 65
  \relpenalty, 65
\phi, 78
pmatrix, 87
preface, 10
\printbibliography, 43
proof, 63, 124
proposition, 13
psmallmatrix, 87
publisher, 38
punctuation
  em dash, 53, 60
  en dash, 53, 60

- hyphen, 53, 54, 60
- in commutative diagrams, 50–52
- in formulas, 49–50
- in lists, 52–53

qed-symbol, 124–128
- *see also* \qedhere

\qedhere, 124
\qquad, 70
\quad, 70
quotation marks, 21
- nested, 23

r, 45, 87
\rangle, 78
\ref, 17
referencing
- capitalise, 19
- cleveref, 18, 19
- \Cref, 20
- \cref, 19
- \eqref, 18
- hyperref, 21
- \label, 17
- \lcnamecref, 20
- \lcnamecrefs, 20
- \nameCref, 20
- \namecref, 20
- \nameCrefs, 20
- \namecrefs, 20
- noabbrev, 19
- \ref, 17

\relpenalty, 65
\renewcommand, 93
report, 8
resume, 25
resume*, 25
\Rightarrow, 78
\rightarrow, 77
Roman numerals, 24, 38
roman numerals, 10
Roman*, 24
roman*, 24
\rVert, 78

scaling
- \big, 94
- \bigg, 94
- \delimsize, 94

scrartcl, 8, 9
scrbook, 9
scrbook, 8, 9
scriptsize, 99, 101
scrreprt, 8, 9
\section, 9
sectioning
- \appendix, 10
- \backmatter, 10
- \chapter, 9
- \frontmatter, 10
- \mainmatter, 10
- \part, 10
- \section, 9
- \subsection, 9
- \subsubsection, 9

series, 38
setlist, 31
\setminus, 78
\shortintertext, 71
small, 99, 101
smallmatrix, 87
\smallsetminus, 78
spacing, 26, 72
- after a period, 55–58
  - \(space), 56
  - \@, 56
  - \frenchspacing, 56
  - \nofrenchspacing, 56
- around display mode, 121
  - \SwapAboveDisplaySkip, 123
- in lists, 26
- in math mode, 89, 91, 92, 102–103, 130
- in math mode, 69
- \qquad, 70
- \quad, 70

standard classes, 8
- article, 8
- book, 8
- report, 8

stmaryrd, 13

structure of an expression, 68
`style`, 36
`\subsection`, 9
`\subset`, 88
`\subseteq`, 88
`\subseteqq`, 88
`\subsetneq`, 88
`\subsetneqq`, 88
`\subsubsection`, 9
`subtitle`, 37
`\SwapAboveDisplaySkip`, 123
symbols, 77

table of contents, 10
tables, 44–48
- `booktabs`, 47
- `\bottomrule`, 47
- `c`, 45
- `\cline`, 45
- `\cmidrule`, 47
- `\hline`, 45
- horizontal lines, 45
- `l`, 45
- `\midrule`, 47
- `r`, 45
- `tabular`, 44
- `\toprule`, 47
- vertical lines, 46

`tabular`, 44
`\tag`, 128, 129
tagging and numbering, 128
- `align`, 106
- `equation`, 104
- `gather`, 105
- `\tag`, 128, 129

TEXStudio, 44
text in math mode
- `\intertext`, 71
- `\shortintertext`, 71

`theorem`, 13
theorem-like environments, 13, 63
- `corollary`, 13
- `lemma`, 13
- `proposition`, 13
- `theorem`, 13

tie, 59
`tikz-cd`, 16
- `column sep`
  - `huge`, 99
  - `large`, 99
  - `normal`, 99
  - `scriptsize`, 99
  - `small`, 99
  - `tiny`, 99
- `row sep`
  - `huge`, 101
  - `large`, 101
  - `normal`, 101
  - `scriptsize`, 101
  - `small`, 101
  - `tiny`, 101
- `column sep`, 99

`tiny`, 99, 101
`title`, 37
title page, 10
`titlepage`, 9
`\to`, 77
`\toprule`, 47
`totoc`, 11

unbreakable
- dash, 60
- space, 59

underlining, 89
`\underset`, 82
Unicode, 1
URL, 41
`url`, 41
`urldate`, 36, 41
`\usepackage`, 4

`\varepsilon`, 78
`\varnothing`, 78
`\vdots`, 85
version control, 1, 3
`Vmatrix`, 87
`vmatrix`, 87
`volume`, 39, 40
`volumes`, 39
`Vsmallmatrix`, 87

vsmallmatrix, 87

warning, 1, 3
wide, 28, 31

XƎLaTeX, 1
\xhookleftarrow, 84
\xhookrightarrow, 84
\xLeftarrow, 84
\xleftarrow, 84
\xleftharpoondown, 84
\xleftharpoonup, 84
\xLeftrightarrow, 84
\xleftrightarrow, 84
\xleftrightharpoons, 84
\xlongequal, 84
\xLongleftarrow, 84
\xlongleftarrow, 84
\xlongleftrightarrow, 84
\xLongrightarrow, 84
\xlongrightarrow, 84
\xmapsto, 84
xparse, 95
\xRightarrow, 84
\xrightarrow, 83, 84
\xrightharpoondown, 84
\xrightharpoonup, 84
\xrightleftharpoons, 84

year, 38